DISCLAIMER:

This document does not meet the
current format guidelines of
the Graduate School at
The University of Texas at Austin.

It has been published for
informational use only.

The Dissertation Committee for Zhao Song
certifies that this is the approved version of the following dissertation:

Matrix Theory:
Optimization, Concentration, and Algorithms

Committee:

Eric Price, Supervisor

Georgios-Alex Dimakis

Adam Klivans

Yin Tat Lee

C. Greg Plaxton

Matrix Theory:
Optimization, Concentration, and Algorithms

by

Zhao Song

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN

August 2019

Dedicated to my parents Li Song and Mei Sun, also my grandparents Kexin Song, Weifeng Guan, Yongzhe Sun, and Fenglin Yang.

Thank Wei Ye for accompanying me over a half of my Ph.D. time.

Acknowledgments

First, I would like to thank my advisor, Professor Eric Price, for all the inspirations and support that he has given me throughout my Ph.D. career. He has been, and will always be, a great source of wisdom and encouragement for me. I genuinely want to thank Eric for giving me so many freedoms during the Ph.D. I thank Eric for allowing me to work on topics from algorithm to complexity, from theoretical computer science to machine learning, and from applications to applied Mathematics. I thank Eric for allowing me to travel around the country to pursue the best scholarship in the field. I also wanted to thank Eric for introducing David and Yin Tat to me. I could never imagine a Ph.D life without Eric as my advisor. It is an honor of life for me to be the first student of Eric.

I am very thankful for my spritual advisor, Professor David P. Woodruff, who had been teaching and training me during the first half of my Ph.D. time.

I really want to thank my informal advisor, Professor Yin Tat Lee, who had been helping and guiding me during the second half of my Ph.D. time.

I would appreciate the contribution from David and Yin Tat on my thesis.

I wish to thank the multitudes of people who helped me. Time would fail me to tell of . . .

Coauthors The main purpose of writing this thesis is to express my gratitude to all co-authors and collaborators. I want to thank all of you x times, where x is the number of pages in this thesis. The list includes, but is not limited to: Alexandr Andoni, Josh Alman, Sanjeev Arora, Zeyuan Allen-Zhu, Maria-Florina Balcan, Duane Boning, Peter L. Bartlett, Aritra Banik, Binay K. Bhattacharya, Dana H. Ballard, Paul Beame, Jan van den Brand, Mitali Bafna, Hongge Chen, Michael Cohen, Xue Chen, Sitan Chen, Constantine Caramanis, Timothy Chu, Chi-Ning Chou, Inderjit S. Dhillon, Luca Daniel, Huaian Diao, Sandip Das, Alexandros G. Dimakis, Minati De, Simon S. Du, Martin Ester, Ankit Garg, Anna Gál, Cho-Jui Hsieh, Wei Hu, Bo Hu, Prateek Jain, Ziheng Jiang, Rajesh Jayaram, Chi Jin, Haotian Jiang, Daniel Kane, Tsunehiko Kameda, Weihao Kong, Rasmus Kyng, Ka Yui Lee, Yin Tat Lee, Yingyu Liang, Yibo Lin, David Liau, Jerry Li, Meng Li, Xingguo Li, Yuanzhi Li, Daogao Liu, Kyle Luh, Dana Moshkovitz, Vasileios Nakos, Michael Orshansky, Eric Price, Aviad Rubinstein, Milan Rubinstein, Ilya P. Razenshteyn, Ankit Singh Rawat, Sasanka Roy, Victor Reis, Xiaorui Sun, Aaron Schild, Aaron Sidford, Nikhil Srivastava, Wen Sun, Vladyslav Sokol, Seyed Abbas Sadat, Saeed Seddighin, Ruoqi Shen, Clifford Stein, Ewin Tang, Richard T. Vaughan, Santosh S. Vempala, Ye Wang, Mengdi Wang, Ruosong Wang, Zhengyu Wang, Tsui-Wei Weng, David P. Woodruff, Lin F. Yang, Xin Yang, Ger Yang, Xinyang Yi, Huacheng Yu, Yi Zhang, Qiuyi Zhang, Huan Zhang, Jiong Zhang, Hongyang Zhang, Ruohan Zhang, Ruizhe Zhang, Kai Zhong, Peilin Zhong, Yuke Zhu.

Useful Discussions I want to thank the researchers who had done super useful discussions with me, and they probably even didn't notice our conversation lead to a paper finally. More surprisingly, perhaps most of the researchers mentioned below couldn't remember they've ever talked to me. I'd like to thank Udit Agarwal, Alexandr Andoni, Josh Alman, Zeyuan Allen-Zhu, Amir Abboud, Megasthenis Asteris, Scott Aaronson, Arturs Backurs, Saugata Basu, Boaz Barak, Jarosław Błasiok, Saugata Basu, Sebastien Bubeck, Peter L. Bartlett, Michael Borokhovich, Mitali Bafna, Paul Beame, Abhishek Bhowmick, Ainesh Bakshi, Eshan Chattopadhyay, Lijie Chen, Xue Chen, Kenneth Clarkson, Hongge Chen, Danqi Chen, Sitan Chen, Chi-ning Chou, Thomas Dillig, Isil Dillig, Alexandros G. Dimakis, Ethan Elenberg, Ronen Eldan, Yu Feng, Vitaly Feldman, Uriel Feige, Ronald Fagin, Rong Ge, Anna Gál, Ankit Garg, Mika Göös, Surbhi Goel, Badih Ghazi, Daniel Hsu, Changyong Hu, Yige Hu, Cho-Jui Hsieh, Wei Hu, Samuel Hopkins, William Hoza, Bernhard Haeupler, Russell Impagliazzo, Chi Jin, Prateek Jain, T.S. Jayram, Ravindran Kannan, Michael Kapralov, Anna Karlin, Rasmus Kyng, Daniel Kane, Janardhan Kulkarni, Murat Kocaoglu, Pritish Kamath, Mrinal Kumar, J. M. Landsberg, Qi Lei, Fu Li, Yixuan Li, Yin Tat Lee, Yibo Lin, Jason D. Lee, James Lee, Yuanzhi Li, Zhiyuan Li, Xin Li, Yingyu Liang, Kyle Luh, Meng Li, Wuxi Li, Daogao Liu, Zhixian Lei, David Liau, Kasper Green Larsen, Syed Mohammad Meesum, Ankur Moitra, Dana Moshkovitz, Cameron Musco, Christopher Musco, Raghu Meka, Jelani Nelson, Vasileios Nakos, Preetum Nakkiran, Richard Peng, Eric Price, Greg Plaxton, Daniel Perrucci, Dimitris Papailiopoulos, Govind Ramnarayan, Ilya Razenshteyn, Anup Rao, James Renegar, Aviad Rubinstein, Rocco Servedio, Tselil Schramm, Clifford Stein, Wen Sun, Madhu Sudan, Aaron Sidford, Aaron Schild, Nikhil Srivastava, Xiaorui Sun, Tianxiao Shen, Saeed Seddighin, Thomas Steinke, Karthikeyan Shanmugam, Xiaofei Shi, Ewin Tang, Elias

Tsigaridas, Rashish Tandon, Avishay Tal, Santosh S. Vempala, Salil Vadhan, Jan Vondrák, Yuhao Wan, Yining Wang, Zhengyu Wang, Zhaoran Wang, Xinyu Wang, Ye Wang, David P. Woodruff, Ryan R. Williams, John Wright, Omri Weinstein, Mary Wootters, Shanshan Wu, Xiaoxia Wu, Bo Xiong, Yuanzhong Xu, Huacheng Yu, Bei Yu, Huacheng Yu, Xin Yang, Lin F. Yang, Ger Yang, Huan Zhang, Hongyang Zhang, Yi Zhang, Jiong Zhang, Ruizhe Zhang, Kai Zhong, Yuke Zhu, Danyang Zhuo, and David Zuckerman for useful discussions.

Visiting My research has benefited enormously from a number of visits to research institutions over the past few years. I would like to thank all my hosts, with whom I have established invaluable friendships: David P. Woodruff at IBM Almaden, Jelani Nelson at Harvard University, Yin Tat Lee at the University of Washington, Zeyuan Allen-Zhu, Jerry Li, and Ilya Razenshteyn at MSR Redmond, Peter L. Bartlett, David P. Woodruff, and Santosh Vempala at Simons Institute at the University of California, Berkeley, Nikhil Srivastava at the University of California, Berkeley, Omri Weinstein, Alex Andoni and Peilin Zhong at Columbia University, and Sanjeev Arora at Princeton University.

I would like to thank all my office mates at the institutions that I have stayed in over the last few years: Pravesh Kothari during my first year at UT-Austin; Mika Göös and Ilya Razenshteyn at IBM in 2015; Cameron Musco, Badih Ghazi, and Xingguo Li at IBM in 2016; Chen Shao, Lin F. Yang, Peilin Zhong, Hongyang Zhang at IBM in 2017. I had a productive year during my visit to Harvard University in 2017 and 2018, where I shared the office with Aviad Rubinstein, Mika Göös, Kyle Luh, Tselil Schramm, Rasmus Kyng, Tobias Christiani, and Huacheng Yu. I enjoyed my leisure time with Josh Alman, Chi-Ning Chou, Tobias Christiani, Zhixian Lei, Zhengyu Wang and Huacheng Yu while I was at Harvard. I had a productive summer when I was interning MSR Redmond in 2018 while sharing the office with Mark Sellke, and Yuanzhi Li. During my visit to UC Berkeley in 2019 Fall, I shared an office with Lijie Chen, Gautam Kamath, Jerry Li, Samuel Hopkins, Ainesh Bakshi, Simon S. Du, Themis Gouleakis, Rajesh Jayaram, Ruosong Wang, Xiaoxia Wu, Xiaofei Shi, and Hongyang Zhang. I was struggling on my thesis while I was interning MSR Redmond in 2019 Summer while sharing the office with Ronen Eldan, Mark Sellke, Sitan Chen, Lixin Huang, Kevin Tian. I enjoyed my time at the University of Washington where I shared the

office with Xin Yang, Haotian Jiang, Swati Padmanabhan, Victor Reis, Ruoqi Shen, Ewin Tang, and Robbie Weber in 2018 and 2019. I enjoyed my leisure time with Lequn Chen, Zihou Gao, Ziheng Jiang, Weihao Kong, Jialin Li, Hanchuan Li, Daogao Liu, Xuhai Xu, Tianyi Zhou, and Danyang Zhuo while I was at the University of Washington. I thank my office mates Fu Li, Kai Zhong, Qi Lei, Jiong Zhang, and Ruizhe Zhang at UT-Austin for their support during the latter part of my Ph.D. career. I enjoy my leisure time with Ruizhe Zhang during the last a few days of my Ph.D. program at UT-Austin.

Traveling As a frequent traveler, I was homeless for the majority of my Ph.D. years. I would like to thank all the friends who have kindly provided me a place to stay: Feng Xiao at IBM; Xindi Hu and Aviad Rubinstein at Harvard; Xin Yang and Jerry Li during my visit to UW and MSR; and Lijie Chen, Themis Gouleakis, Chi Jin, Gautam Kamath, and Santosh Vempala who shared their house with me while I was visiting UC Berkeley.

Over my entire Ph.D., I don't know how to drive. My life would be very inconvenient if I don't know the following friends who had given me a ride many times. I want to thank for Beidi Chen, Qiren Chen, Lequn Chen, Yu Feng, Chi Jin, Weihao Kong, Meng Li, Xingguo Li, Yuanzhi Li , Yibo Lin, Jialin Li, Jinsong Liu, Aviad Rubinstein, Ilya Razenshteyn, Ruoqi Shen, Aaron Sidford, Xinyu Wang, Ye Wang, David P. Woodruff, Yun Wu, Biying Xu, Feng Xiao, Xin Yang, Lin F. Yang, Bei Yu, Zhuoran Zhao, Ruohan Zhang, Kai Zhong, Danyang Zhuo.

RA and TA Over the past few years, I have received fundings from many generous professors, including Alexandros Dimakis, Inderjit Dhillon, Adam Klivans, Rasmus Kyng, Yin Tat Lee and Eric Price.

Over the past few years, I have been doing TA so many times. I'd like to thank Rasmus Kyng and Bill Young for working with at Harvard and UT-Austin.

Undergraduate The decision to pursue a Ph.D. is already a brave one for me. I really want to thank for my undergraduate advisors Richard Vaughan, Tsunehiko Kameda, Binay K. Bhattacharya, Ke Wang, Martin Ester, Ze-Nian Li, Hao Zhang (at Simon Fraser University) for kindly training and guiding a young researcher. I would like to thank for my undergraduate friends Song Chen, Tiaoxiang Gao, Junsong Hu, Wen Sun, Chi Jin, Yatao Wang, Mingzhou Yang, Huan Zhang, Zhuoran Zhao, Yuke Zhu who encouraged me to pursue Ph.D.

Struggling Time Over the entire Ph.D. time, I have been thinking about quitting the Ph.D. program seriously thousands of times. I really want to thank all the friends who had suggested me not to give up. I had a very struggling time during my first year at Austin. I truly appreciate Adam Klivans and Pravesh Kothari's support without which I might have quitted Ph.D. I really like to thank Alexandros G. Dimakis, Anna Gál, and Ankit Singh Rawat for helping me publishing the first theory paper during my Ph.D. This was extremely important to me, as it gave me the confidence to continue my Ph.D. research.

During the second year, I met my advisor Eric. I want to thank him for giving me a set of toys (Fox, etc.). Those inspire my research for many years. I finally collected six before graduation.

In the summer of my second year, I was an intern at IBM Almaden. This was the first time that I did research outside my home institution. I would like to thank my intern advisor David Woodruff for his help on many things, especially on giving me the first stock in my life. When I was planning to graduate, that stock is five times than before. Since then, I started to collaborate with David heavily, and he became my pseudo-advisor.

While I was at the University of Washington, I was struggling with many things. I want to thank Yin Tat Lee for chatting with me about many non-research kinds of stuff, sharing so many funny stories with me, and giving many bits of help in my life.

Even during the last year of my Ph.D., I was still considering quitting the Ph.D. due to a variety of reasons. I really appreciate the support from so many professors (Scott Aaronson, Greg Plaxton, Adam Klivans, Eric Price, David P. Woodruff) and friends (Zeyuan Allen-Zhu, Josh Alman, Lijie Chen, Gautam Kamath Rasmus Kyng, Yin Tat Lee, Xingguo

Li, Jerry Li, Ilya Razenshteyn, Aviad Rubinstein, Ruosong Wang, Zhengyu Wang, Lin Yang, Xin Yang, Peilin Zhong) who convinced me to finish the Ph.D. program.

??? Friends I wanted to thank Lijie Chen, Zhengyu Wang, and Peilin Zhong for their enourmous help on my research. Their contributions range from proposing new research ideas to paper proofreading. Many times they kindly declined the authorship they deserved. I am so grateful for their help and friendship that I could not find a proper category for them. This is why I put "???" above.

I wanted to thank Ziheng Jiang for his generous help at the University of Washington, without which I would not be able to write this thesis.

I really wanted to thank Billy (Vasileios) and Zhengyu for writing my only student papers with me.

At the end of the third year, I started to get interested in Machine Learning and Deep Learning. I would like to thank my friend Kai Zhong for introducing this new field for me.

Thesis I want to thank Zeyuan Allen-Zhu and Yin Tat Lee for the suggestion of the title of the thesis. I want to thank Jerry Li for choosing the final title. I want to thank Jerry Li (Sana Minatozaki) for his generous help on the abstract, introduction, and the open problem section of the thesis. I want to thank Lijie Chen for his help on acknowledgment. I want to thank Ruoqi Shen for her generous advice on the abstract of the thesis. I appreciate Zhengyu Wang's generous help on the acknowledgment, abstract, introduction and open problem section of the thesis. I want to thank Yin Tat Lee, Victor Reis, Xiaorui Sun, Xin Yang, Peilin Zhong for their generous help on the open problem section.

Defense and Graduation I want to thank Ruizhe Zhang for his generous help during my defense day. I want to thank all my committees and Vijaya Ramachandran for coming to my defense at UT-Austin. I want to thank Yin Tat Lee and Xin Yang for creating two Skype call lives of my defense. One is at MSR Redmond, and the other is at the University of Washington. I want to thank Weihao Kong, Jialin Li, Daogao Liu, Swati Padmanabhan, Ruoqi Shen, Xin Yang, and Danyang Zhuo for joining my defense remotely. I want to thank Beidi Chen, Lixin Huang, Gautam Kamath, Yin Tat Lee, Jerry Li, Chen Luo, Aaron Sidford, and Sam Wong for attending my defense remotely.

I want to thank Katie Dahm (who is the CS Graduate Program Coordinator at The University of Texas at Austin) for giving all kinds of help for letting me graduate on time.

To Junior Ph.D. Student To be honest, I never thought I was a successful Ph.D. student. At the end of my thesis defense, someone asked a question: "do you have any suggestions for junior students?". I am still not sure if I have a satisfactory answer to that question today. Instead, I wanted to write down how I do research, and what I feel when something happened, for the reference of future readers.

1. After every meeting, I will write down the notes as formal as possible via latex. Even if the idea proposed in the meeting is completely wrong, I still write down the details of all the calculations and mark why it doesn't work. To me, the definition of the research progress is not just solving the problem. Sometimes, ruling out some wrong directions also counts as remarkable progress.

2. For the first three months of my Ph.D., I almost spent all the time on solving one problem. But, it didn't go anywhere. I felt truly sad and wanted to quit the Ph.D. program. Then someone suggested me to go to some classes to learn more things and read more papers. I took the suggestion. Luckily, after that class, I got my first theory paper during my Ph.D. This gives me very strong confidence to continue my Ph.D.

3. After that first three months of bad experience, I usually spent more time reading papers than thinking about how to solve problems. I am reading ArXiv papers every day. Usually, once a conference released the accepted paper list, I will google every title of the paper and try to take a quick look.

4. I am in general happy to talk about research with anyone in any field. I am also open to working on any problems in any field. As long as my coauthors want to work on something with me, I am happy to work on it. As far as I remember, I never say "no" to

anyone or any problem.

To clarify, I am just sharing my experience with everyone. I am NOT suggesting anyone try to follow my way.

Parents Finally, it is time to say something for my parents.

Although I am neither a lousy Ph.D. student nor a stupid coauthor, I have to admit that I am a horrible son. Over my entire Ph.D. career, I have only met my parents twice for a total of one month. I still remember the shock that I had during our first meeting. I could barely recognize their faces, and they became so much older compared to the time I left them.

My parents do not know much about Computer Science or graduate school, and I was not patient enough to explain to them what was going on in my life. For all of these, I want to say "sorry" to my parents. I truly appreciate their support in my life. Mom and dad, I love you!

It is difficult to forget the pleasant time with my grandparents during my childhood. I felt really sorry that I did not give enough time to them as I grew older. That is the reason why I have mentioned all of my grandparents' names in this thesis.

Previous Publications

This thesis is based on several previous publications. We list the chapters together with the corresponding papers.

Optimization

Chapter 2 is based upon the following previous publication

- Michael B. Cohen, Yin Tat Lee, Zhao Song

  Solving Linear Programs in the Current Matrix Multiplication Time.

  STOC 2019 [CLS19]

Chapter 3 is based upon the following previous publication

- Yin Tat Lee, Zhao Song, Qiuyi Zhang

  Solving Empirical Risk Minimization in the Current Matrix Multiplication.

  COLT 2019 [LSZ19]

Chapter 4 is based upon the following previous publication

- Yin Tat Lee, Zhao Song, Santosh S. Vempala

  Algorithmic Theory of ODEs and Sampling from Well-conditioned Logconcave Densi-

ties.

Manuscript 2018 [LSV18]

Chapter 5 is based upon the following previous publication

- Zeyuan Allen-Zhu, Yuanzhi Li, Zhao Song

  A Convergence Theory for Deep Learning via Over-Parameterization.

  ICML 2019 [AZLS19]

Chapter 6 is based upon the following previous publication

- Zeyuan Allen-Zhu, Yuanzhi Li, Zhao Song

  On the convergence rate of training recurrent neural networks.

  Manuscript 2018 [AZLS18]

Chapter 7 is based upon the following previous publication

- Ankit Garg, Yin Tat Lee, Zhao Song, Nikhil Srivastava

  A Matrix Expander Chernoff Bound.

  STOC 2018 [GLSS18]

Chapter 8 is based upon the following previous publication

- Rasmus Kyng, Zhao Song

  A Matrix Chernoff Bound for Strongly Rayleigh Distributions and Spectral Sparsifiers

from a few Random Spanning Trees.

FOCS 2018 [KS18]

Chapter 9 is based upon the following previous publication

- Rasmus Kyng, Kyle Luh, Zhao Song

  Four Deviations Suffice for Rank 1 Matrices.

  Manuscript 2019 [KLS19]

Chapter 10 is based upon the following previous publication

- Vasileios Nakos, Zhao Song

  Stronger L2/L2 Compressed Sensing; Without Iterating.

  STOC 2019 [NS19]

Chapter 11 is based upon the following previous publication

- Eric Price, Zhao Song

  A Robust Sparse Fourier Transform in the Continuous Setting.

  FOCS 2015 [PS15]

Chapter 12 is based upon the following previous publication

- Xue Chen, Eric Price, Daniel Kane, Zhao Song.

  Fourier-sparse interpolation without a frequency gap.

  FOCS 2016 [CKPS16]

Chapter 13 is based upon the following previous publication

- Vasileios Nakos, Zhao Song, Zhengyu Wang

  (Nearly) Sample-Optimal Sparse Fourier Transform in Any Dimension; RIPless and Filterless.

  FOCS 2019 [NSW19a]

Chapter 14 is based upon the following previous publication

- Vasileios Nakos, Zhao Song, Zhengyu Wang

  The Power of (Careful) Iterative Loop Analysis in Compressed Sensing.

  Manuscript 2019 [NSW18]

Chapter 16 is based upon the following previous publication

- Ilya Razenshteyn, Zhao Song, David Woodruff

  Weighted low rank approximations with provable guarantees.

  STOC 2016 [RSW16]

Chapter 17 is based upon the following previous publication

- Zhao Song, David Woodruff, Peilin Zhong

  Low Rank Approximation with Entrywise ℓ_1-Norm Error.

  STOC 2017 [SWZ17]

Chapter 19 and 20 are based upon the following previous publication

- Zhao Song, David Woodruff, Peilin Zhong

  Towards a Zero-One Law for Entrywise Low Rank Approximation.

  Manuscript 2019 [SWZ18]

Chapter 21 is based upon the following previous publication

- Zhao Song, David Woodruff, Peilin Zhong

  Relative Error Tensor Low Rank Approximation.

  SODA 2019 [SWZ19b]

Chapter 22 is based upon the following previous publication

- Zhao Song, David Woodruff, Huan Zhang

  Sublinear Time Orthogonal Tensor Decomposition.

  NeurIPS 2016 [SWZ16]

Chapter 23 is an open problem section, which is based on the all the publications corresponding to Chapters 2∼22 and also the following previous publications

- Alexandr Andoni, Zhao Song, Clifford Stein, Zhengyu Wang, Peilin Zhong

  Parallel graph connectivity in log diameter rounds.

  FOCS 2018 [ASS+18]

- Aviad Rubinstein, Saeed Seddighin, Zhao Song, Xiaorui Sun

  Approximation Algorithms for LCS and LIS with Truly Improved Running Times.

  FOCS 2019 [RSSS19]

- Aviad Rubinstein, Zhao Song

  Reducing approximate Longest Common Subsequence to approximate Edit Distance.

  Manuscript 2019 [RS19b]

We remark that the papers covered by this thesis are only about part of the author's gradschool-publications. Topics not covered in this thesis are deep learning theory [ZSJ+17, ZSD17, WZC+18, LSY19, SY19], recurrent neural network [ZLSD18], finding adversarial examples [WZC+18, ZCS+19], reinforcement learning [ZSB15, SS19], coding theory [DGRS15], unique games [BCS18], optimization [LPS19, vdBLSS19], learning mixture distributions [CLS20], multi-arm bandits [LPSY18], matrix completion [ZSJD18, BLS+19], Fourier transform with restricted samples [LSS19], regression [PSW17, DSSW18, DSWY19, DJS+19, SWY+19], Laplacian system [ASS19], low-rank approximation [LSW+19], EDA design [WLY+15], fast multipole method [ASS19], geometry algorithms [BBD+18, ASS19], robotics [ZS16], quantum algorithm [TS19] and discrepancy [RS19a].

The author also had done a few research when he was an undergraduate student. Since the author never wrote a undergraduate thesis, we like to list those papers here. The topics are robotics [SV12b, SSV12, SV13, SS14], graph and geometry algorithms [BKS12b, BKS12c, BKS14, BDK+14, BKS15b, BBD+16], data-mining [HSE12, SZ13].

Matrix Theory:
Optimization, Concentration, and Algorithms

Publication No. _____

Zhao Song, Ph.D.
The University of Texas at Austin, 2019

Supervisor: Eric Price

The matrix plays an essential role in many theoretical computer science and machine learning problems. In this thesis, we develop a better understanding of matrices with a view towards these applications. Our insights yield improvements for a number of old, well-studied algorithmic problems.

In this thesis, we study matrices from three perspectives. We first consider their role in *optimization*. We study a number of matrix optimization problems, and propose new solvers and results for linear programs, empirical risk minimization, ordinary differential equations, deep neural networks. We next consider how random matrices *concentrate*. Specifically, we generalize a number of scalar Chernoff-type concentration inequalities and the Spencer-type discrepancy theorems to matrices. Finally, we develop new *algorithms* for problems on matrices. These fall roughly into two sub-categories, namely matrix factorization problems and structured recovery problems. In the first category, we propose a number of new algorithms

for a variety of low-rank matrix factorization problems. In the second category, we give new algorithms for some recovery tasks with structured matrices. We design matrices and corresponding algorithms for compressed sensing tasks, and we give fast algorithms for the sparse Fourier transform problem, which can be thought of as a sparse recovery problem where one cannot freely choose the matrix. We now describe our contributions in more detail.

Linear programming is one of the fundamental problems in computer science. In both theory and practice, many problems can be solved via linear programming, and doing so often yields the best-known runtime. We present an algorithm that runs in the current matrix multiplication time, which breaks a thirty-year-old barrier. Furthermore, our technique can be generalized to speed up a large family of convex optimization problems, i.e., empirical risk minimization.

Sampling logconcave functions which arise in statistics and machine learning has been the subject of intensive study. This problem can be thought of as a special case of solving multivariate ordinary differential equations (ODEs). Under sufficiently strong smoothness conditions, discrete algorithms of Hamiltonian Monte Carlo (HMC) have runtime and number of function evaluations growing with the dimension. We give new algorithms that obtain a nearly linear implementation of HMC for a broad class of smooth, strongly logconcave densities, with the number of iterations (parallel depth) and gradient evaluations being polylogarithmic in the dimension (rather than polynomial, as in previous work).

Deep neural networks (DNNs) have demonstrated dominating performance in many fields; since AlexNet, networks used in practice are going wider and deeper. We prove why stochastic gradient descent (SGD) can find global minima on the training objective of DNNs in polynomial time. We only make two assumptions: the inputs are non-degenerate and

the network is over-parameterized. Furthermore, our results also hold for Recurrent Neural Networks (RNNs) which are multi-layer networks widely used in natural language processing. They are harder to analyze than feedforward neural networks, because the same recurrent unit is repeatedly applied across the entire time horizon.

The Chernoff bound for the concentration of scalar random variables is a fundamental tool in the analysis of randomized algorithms. Over the past decade, a matrix generalization of the Chernoff bound has also found widespread application, but this generalization is fairly restrictive. For a number of decades, it has been open whether one can remove some of these restrictions. We answer this question in the affirmative by giving a number of new matrix Chernoff bounds under more relaxed independence assumptions than before.

Spencer's theorem is a famous result in discrepancy theory, and it is an important open question how to generalize this result to the matrix setting. We make progress on this, and prove a matrix generalization of this result, in some restricted settings. Our result also generalizes the famous Kadison-Singer conjecture.

The classical low rank approximation problem is : given a matrix A, find a rank-k matrix B such that the Frobenius norm of $A-B$ is minimized. It can be solved in polynomial-time using, for instance, singular value decomposition. If one allows randomization and approximation, it can be solved in time proportional to the number of non-zero entries of A with high probability. We consider a number of natural generalizations of this important problem. We generalize the problem to consider a number of different norms, including weighted norms, entry-wise L1, and more general loss functions including Huber and Tukey loss. We also consider the natural tensor version of the problem. For all these settings, we give new state-of-the-art algorithms, including a number of new fixed parameter tractable

algorithms.

Compressed Sensing, or sparse recovery, is a powerful mathematical framework whose goal is to reconstruct an approximately sparse signal from linear measurements. We consider the extensively studied problem of L2/L2 compressed sensing. Our algorithm has faster decoding time and significantly smaller column sparsity, answering two open questions of prior work. Previous work on sublinear-time compressed sensing employed an iterative procedure, recovering the heavy coordinates in phases. We completely depart from that framework, and give the first sublinear-time algorithm which achieves the optimal number of measurements without iterating.

The Fourier transform is ubiquitous in computer science, mathematics, and beyond. In recent years, a number of works have studied methods for computing the Fourier transform in sublinear time if the output is sparse. We consider two important settings in which this occurs: namely, the sparse high-dimensional setting, and the sparse continuous setting. In the high dimensional setting, we consider the extensively studied problem of computing a k-sparse approximation to the d-dimensional Fourier transform of a length n signal. Our algorithm achieves a nearly optimal sample complexity and runs in time comparable to the Fast Fourier Transform. All previous algorithms proceed either via the Restricted Isometry Property or via filter functions. Our approach totally departs from the aforementioned techniques, and we believe is a new approach to the sparse Fourier transform problem.

While many of the works on sparse Fourier transforms have focused on the discrete setting, in many applications the input signal is continuous and naive discretization significantly worsens the sparsity level. Assuming the frequency gap is known, we present an algorithm for robustly computing sparse Fourier transforms in the continuous setting. Know-

ing the frequency gap is necessary for robustly identifying individual frequencies. However, was unknown whether interpolating the signal also requires a frequency gap. We resolve this problem by giving an algorithm which shows that such a gap not necessary for estimating the signal.

Table of Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

1.1 Introduction

Many important questions in computer science can be formulated as problems on matrices. Having a better understanding of the matrix is crucial for devising fast algorithms for many fundamental problems, including a majority of the 10 most important algorithms of the 20th century [Cip00]. Examples of this include linear programming, Metropolis algorithm for Monte Carlo, matrix/tensor factorization, low-rank approximation, Fourier transforms, etc. In this thesis, we make progress on a number of long-standing open problems related to this areas. Along the way, we propose a number of new techniques which be could be useful in the future. We first summarize several core ideas, new insights, and new techniques related to the matrix. These techniques will eventually lead to fast algorithms. Later on in the introduction, we go to the details of each category.

- **Sketching.** A powerful technique we will use in many places throughout this thesis is sketching. Given a matrix $A \in \mathbb{R}^{n \times d}$, our goal is to produce a concise representation of A to summarize informationa about A without using A directly. This is typically done by choosing a "small" sketching matrix S, and replacing A with SA. We say SA is a sketch of A. There are many previously studied choices for S, depending on the specific problem instance, including random Gaussian matrices, subsampled

randomized Hadamard matrices, and Count-Sketch matrices. By using SA, which is typically much smaller than A, we are able to speed up a number of algorithms.

This idea is often called "sketch and solve" in numerical linear algebra. Suppose that, for a given problem, we can produce a solver, which we wish to speed up. Oftentimes, we can apply a sketching matrix to reduce the dimensionality/freedom/size of the original problem, and then run the original solver on this smaller problem, and obtain a faster algorithm. Some classical examples are ℓ_2 norm linear regression and Frobenius norm low-rank approximation. A number of results in this thesis use much more complicated versions of these sketching techniques, such as "guess and sketch" and "sketch and iterate," to achieve similar speed-ups in other problem settings. Many results in this thesis modify the classical iterating algorithms more decently with a better understanding of hashing-trick, sampling, and concentration techniques.

- **Sampling.** Given a matrix $A \in \mathbb{R}^{n \times d}$, our goal is to obtain a concise representation of A by sampling some rows of A and reweighting them. Let $D \in \mathbb{R}^{n \times n}$ denote a sparse diagonal matrix. Then DA can be viewed as a sampled version of A. The goal of this is similar to sketching, as the goal is to produce a smaller representation of A, so that our algorithms run faster. In our context, the main difference between sketching and sampling is that in sketching, each row of SA is a linear combination of rows in A, and whereas when we sample, each row of DA is a row of A with reasonable rescaling factor. There are several well-known examples, e.g. importance sampling, leverage scores sampling, and Lewis weights sampling.

  – By sampling (in conjunction with some other ideas), we are able improve the

runtime for solving some convex problems, e.g. linear programs [CLS19]. We propose an algorithm that solves linear programs in the current matrix multiplication time. Our result breaks a thirty-year-old barrier.

- Sampling can also be used to solve some non-convex problems. We give a nearly optimal matrix CUR decomposition [SWZ17, SWZ19b] algorithm. We also extend the definition of matrix CUR problem to tensor CURT problem [SWZ19b] and provide a nearly optimal algorithm.
- Sampling can be used to speed up the iterating algorithm for orthogonal tensor decomposition [SWZ16]. This leads to a sublinear time finally.

- **Matrix Maintainance.** Given a matrix A and a sequence of T matrices $W^1, W^2, \cdots, W^T$, our goal is to approximately maintain $f(A, W^t)$ over all the iterations. Formally speaking, whenever we receives W^t, we want to generate $g(A, W^t)$ such that $g(A, W^t) \approx f(A, W^t)$. The objective is to do so faster than simply calculating $f(A, W^t)$ independently for each $t = 1, \ldots, T$, by leveraging some structure in the sequence of update matrices.

  - In [LSZ19], we propose a technique for doing this, which we call "iterate and sketch", when f takes the form of a projection. Specifically, we consider functions f of the form

    $$f(A, W) = \sqrt{W} A^\top (A W A^\top)^{-1} A \sqrt{W}.$$

    When using the central path method to solve linear programs, in each iteration we need to multiply $f(A, W^t)$ by some vector. However, since the updates W^t are

very structured, we don't need to compute this from scratch. By instead cleverly maintaining the sequence of updates, we are able to decrease the running time of the solver. Note that this idea is very different from "sketch and solve" described in the early paragraph. The main difference between "sketch and solve" and "iterate and sketch" is, the former just uses the original solver as a black-box, whereas the latter changes the original solver and uses a different sketching matrix over each iteration of the solver. There are other examples in the thesis can be thought of as "iterate and sketch", e.g. [CLS19, SWZ16].

- **Matrix Concentration.** The Chernoff bound for the concentration of scalar random variables is a fundamental and ubiquitous tool in the analysis of randomized algorithms. One common form of the bound is the following: given a list of independent random variables $x_1, \cdots, x_k \in [0, 1]$ with mean μ, then

$$\Pr\left[\left|\frac{1}{k}\sum_{i=1}^{k} x_i - \mu\right| > \epsilon\right] \leq 2\exp(-\Omega(k\epsilon^2)).$$

There is a natural generalization of this to matrices. Namely, if $X_1, \cdots, X_k$ are independent $n \times n$ complex Hermitian random matrices with $\|X_i\| \leq 1$ and mean μ, then the following is true:

$$\Pr\left[\left\|\frac{1}{k}\sum_{i=1}^{k} X_i - \mu\right\| > \epsilon\right] \leq 2n\exp(-\Omega(k\epsilon^2)).$$

Over the past decade, the matrix generalization of the Chernoff bound has also found widespread application, but this generalization is fairly restrictive. In particular, this bound requires strong independence of the individual X_i. In contrast, in the scalar setting, it is known that often times one can get away with some weak dependences

between the summands. For a number of decades, it has been open whether one can remove some of these restrictions in the matrix setting as well.

  - In the scalar setting, it is well-known that if the random variables follow the dependence structure of a random walk on a graph, then the random variables still obey a Chernoff-style bound. It was conjectured by Wigderson and Xiao that a similar result holds in the matrix setting. In [GLSS18], we resolve this conjecture in the affirmative. Formally speaking, let $v_1, \cdots, v_k$ denote a random walk on expander graph G, let f denote a mapping $f : V \to \mathbb{R}^{n\times n}$ with $\|f(v)\| \leq 1$, $\forall v \in V$, then $\sum_{i=1}^{k} f(v_i)$ follows matrix Chernoff bound.
  - We show a matrix Chernoff bound for random spanning trees [KS18]. Such a result was conjectured by Batson, Spielman, Srivastava and Teng [BSST13]. Formally speaking, let $A_1, \cdots, A_k \in \mathbb{R}^{n\times n}$ denote a list of matrices where each of them corresponds to a random spanning tree. We show that $k = O(\log^2 n)$ suffices to imply that $\sum_{i=1}^{k} A_i$ gives spectral sparsifier for graph G.

- **Sketch and Guess.** The "sketch and guess" is the following, surprisingly powerful, paradigm: to solve matrix (or tensor) factorization problem, first demonstrate that the optimal solution may be represented as a concise sketch with fewer degrees of freedom, and then simply "guess" this sketch by exhaustively searching over all possible sketches. Since we have reduced the degrees of freedom, this exhaustive search can be done relatively efficiently. For instance, consider the problem of low-rank factorization. Given a matrix $A \in \mathbb{R}^{n\times n}$, our goal is to find two matrices $U \in \mathbb{R}^{n\times k}$ and $V \in \mathbb{R}^{k\times n}$ so that $\|UV - A\|_F$ is minimized. The idea is to write a polynomial system with a small

number of variables and constraints, abd low degree, and then run a polynomial system solver on this small polynomial system. For example, in order to solve $\min \|UV - A\|_F^2$, we can create $2nk$ variable (one for every entry of U and V), and minimize the resulting degree-4 polynomial system. The polynomial system solver has exponential dependence on the number of variables, and thus has runtime $2^{O(nk)}$. The trick to reduce the number of variables is choosing two sketching matrices S (a fat one, it has size $O(k) \times n$) and R (a tall one, it has size $n \times O(k)$), then focusing on $\min_{X,Y} \|ASXYRA - A\|_F^2$. To solve this problem, we can create $O(k^2)$ variables to minimize a degree-4 polynomial system, resulting in a runtime that, while exponential in k^2, is polynomial in n. Of course, for this specific problem, this technique is overkill, as we can solve this problem in time which is polynomial in both n and k via SVD. However, in situations where the problem we are trying to solve is NP-hard, this technique (and extensions thereof) can provide a nearly optimal approximation algorithms.

- We first propose the "guess a sketch" idea in [RSW16]. The combination of guessing and sketching can be used to get FPT-type algorithms for several matrix/tensor factorization problems [SWZ17, SWZ19b].
- For the entry-wise L1 norm, we propose "sketch and relax" idea in [SWZ17]. We want to solve $\min \|UV - A\|_1$. To write a polynomial, we need to create variables for U, V and also n^2 sign variables. To reduce the number of variables, we use L1 embedding matrix to reduce the dimension from n to $r = \text{poly}(k)$ and then relax the problem to Frobenius by losing some factors in the approximation ratio.

Eventually, we just need to solve

$$\min_{X,Y} \|T_1 ASXYRAT_2 - T_1 AT_2\|_1.$$

Since the number of rows of T_1 and $T_2^\top$ is small, thus, we only need to create small number of sign variables, and as a result we obtain a faster runtime.

– For the tensor case, we propose "sketch and rotate" idea in [SWZ19b]. We want to solve $\min \|U \otimes V \otimes W - A\|_F^2$. Slightly different from matrix case, we can create $3nk$ variables to minimize a degree-6 polynomial system. Applying the sketching trick again, we can write

$$\min_{X,Y,Z} \|A_1 SX \otimes A_2 RY \otimes A_3 TZ - A\|_F^2$$

where $A_1, A_2, A_3 \in \mathbb{R}^{n \times n^2}$ are the three possible flattening of tensor A and S, R, T are random sketching matrices.

- **Iterating and Concentration.** Deep neural networks (DNNs) have demonstrated dominating performance in many fields; since AlexNet, networks used in practice are going wider and deeper. We want to give some explanation about why stochastic gradient descent (SGD) can find global minima on the training objective of DNNs and Recurrent Neural Networks (RNNs). In previous paragraphs, we have discussed several techniques for improving convex, non-convex, and NP-hard problems. Unfortunately, the theory of deep learning cannot benefit from those "sketch and solve", "sketch and iterate", and "sketch and guess" techniques.

  – One view of the over-parameterization theory of DNNs [AZLS19] is, $\|\phi(Wx)\|_2$ has a good concentration when x is some fixed unit vector, W is random Gaussian matrix, and $\phi(t) = \max\{t, 0\}$ is the ReLU activation function. Technically

speaking, if we normalize W properly, $\|\phi(Wx)\| \approx (1+\epsilon)\|x\|_2$ with very high probability. In each of iteration of SGD, we update $W \leftarrow W + \Delta W$. There, we need to argue such concentration-type result for different Ws in each iteration. We call it "iterate and concentrate". A variation of this idea appears in [LSZ19] (not involved ReLU), where we use subsampled randomized Hadamard matrix to show concentration over each iteration.

  - RNNs are multi-layer networks widely used in natural language processing. They are harder to analyze than feedforward neural networks, because the same recurrent unit is repeatedly applied across the entire time horizon. By doing a more careful randomness decomposition (in conjunction with other ideas, e.g., concentrations, Gram–Schmidt process), we can show the provable guarantees of RNNs [AZLS18].

- **Hashing.** For a given vector $x \in \mathbb{C}^n$ and a parameter k, we consider the problem where we observe $y = \Phi x$, and our goal is to figure out a $O(k)$-sparse approximation to x. There are two situations in which this problem is often considered. In one setting, we can design the matrix Φ. In this case, we want Φ to implement some hashing-based function. The major things to minimize are, the number of rows of Φ, the column sparsity of Φ (encoding time), and decoding time [GI10]. Alternatively, in some other settings, we cannot design the matrix Φ. Typically, in this setting, Φ is a discrete Fourier matrix, and the problem is referred to as the sparse Fourier transform problem. In this case, we want to smartly take some samples from Φx, and those samples are able to do a hashing job. The major things to minimize are sample complexity and decoding time.

The sparse Fourier transform also can be defined in the continuous setting. This problem can be thought of as the mathematical abstraction of supperresolution problems [Moi15], the development of which was awarded the 2014 Nobel Prize in Chemistry. Intuitively, in the discrete setting we are working with a matrix Φ of finite size, but in the continuous setting, we need to understand the "infinite matrix" Φ, which is more formally a linear operator over functions. However, by generalizing intuitions from the finite setting, many of the ideas can be transferred over to this infinite setting. In the continuous setting, there is one more important parameter to consider, in addition to the ones mentioned above. Namely, we should try to minimize the duration of the signal our algorithm needs to observe.

- The setting where we can design Φ is often called the *compressed sensing* problem [Don06, CRT06b]. We give a new construction of Φ and the corresponding algorithm that improves the classical L2/L2 compressed sensing task [NS19].
- We give a new iterating algorithm for the d-dimensional discrete sparse Fourier transform with nearly optimal sample complexity [NSW19a].
- In the continuous setting, in [PS15] we propose an algorithm for the robust sparse Fourier transform under the assumption of a frequency gap. The sample complexity is linear in k and logarithmic in the signal-to-noise ratio and the frequency resolution. Previous results with similar sample complexities could not tolerate an infinitesimal amount of i.i.d. Gaussian noise, and even algorithms with higher sample complexities increased the noise by a polynomial factor. We also give new results for how precisely the individual frequencies of the signal can be recovered.

– The previously mentioned work of [PS15] relies on recovering the frequencies, and as a result, requires a dependence on the frequency gap. However, if all we care about is reconstructing a k-sparse signal which is close to the original, but which potentially uses different frequencies, such a gap is not necessary. In [CKPS16], we demonstrate an algorithm for the robust sparse Fourier transform that does not require any dependence on the frequency gap. In some ways, both [PS15] and [CKPS16] can be thought of generalizations of the aforementioned hashing techniques to the continuous setting.

1.2 Optimization

Linear programming is one of the key problems in computer science. In both theory and practice, many problems can be reformulated as linear programs to take advantage of fast algorithms. For an arbitrary linear program $\min_{Ax=b,x\geq 0} c^\top x$ with n variables and d constraints[1], the fastest algorithm takes $O^*(\sqrt{d}\cdot \text{nnz}(A) + d^{2.5})$[2] where $\text{nnz}(A)$ is the number of non-zeros in A [LS14, LS15].

For the generic case $d = \Omega(n)$ we focus in this thesis, the current fastest runtime is dominated by $O^*(n^{2.5})$. This runtime has not been improved since the result by Vaidya on 1989 [Vai87, Vai89b]. The $n^{2.5}$ bound originated from two factors: the cost per iteration n^2 and the number of iterations $\sqrt{n}$. The n^2 cost per iteration looks optimal because this is the cost to compute Ax for a dense A. Therefore, many efforts [Kar84, Ren88, NN89, Vai89a,

[1] Throughout this thesis, we assume there is no redundant constraints and hence $n \geq d$. Note that papers in different communities uses different symbols to denote the number of variables and constraints in a linear program.

[2] We use O^* to hide $n^{o(1)}$ and $\log^{O(1)}(1/\delta)$ factors and $\widetilde{O}$ to hide $\log^{O(1)}(n/\delta)$ factors.

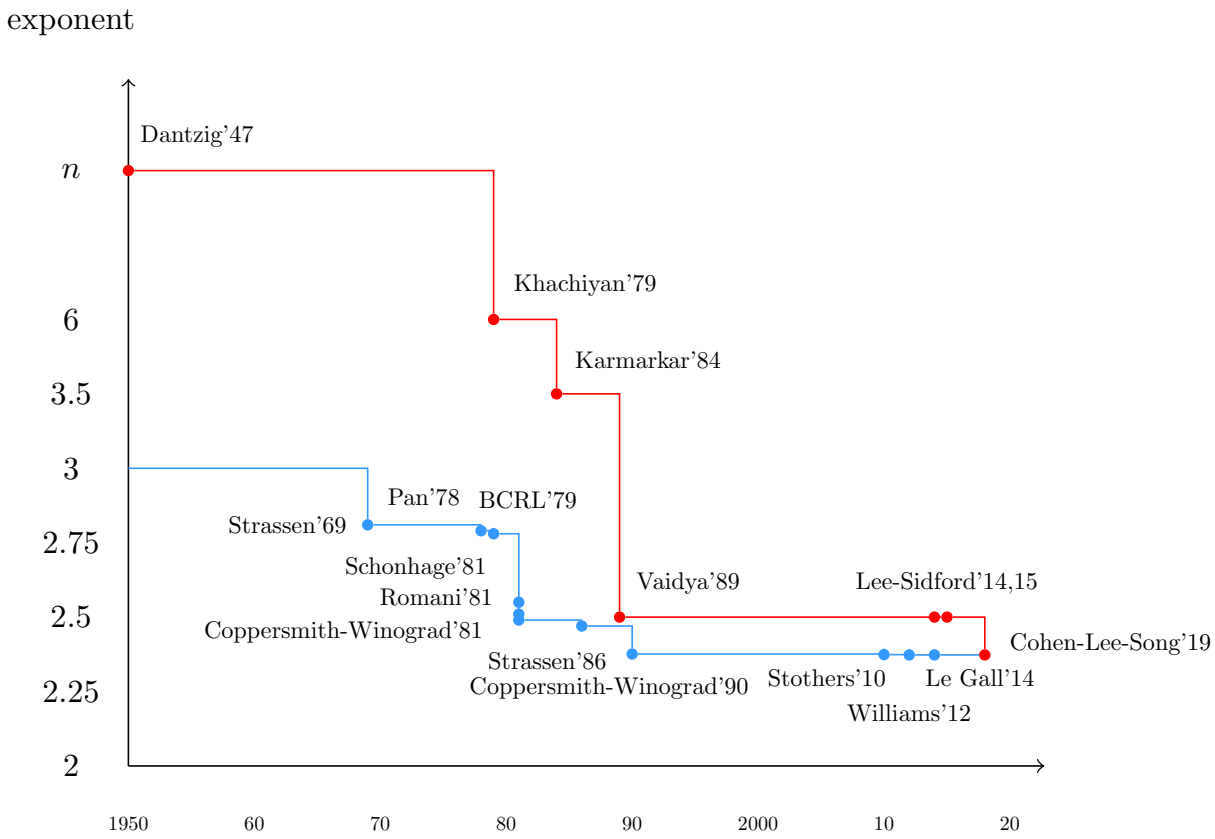


Figure 1.1: Linear Programming is Chasing Matrix Multiplication

LS14] have been focused on decreasing the number of iterations while maintaining the cost per iteration. As for many important linear programs (and convex programs), the number of iterations has been decreased, including maximum flow [Mad13, Mad16], minimum cost flow [CMSV17], geometric median [CLM+16], matrix scaling and balancing [CMTV17], and ℓ_p regression [BCLL18]. Unfortunately, beating $\sqrt{n}$ iterations (or $\sqrt{d}$ when $d \ll n$) for the general case remains one of the biggest open problems in optimization.

Avoiding this open problem, this thesis develops a stochastic central path method that has a runtime of $O^*(n^\omega + n^{2.5-\alpha/2} + n^{2+1/6})$, where ω is the exponent of matrix multiplication

and α is the dual exponent of matrix multiplication[3]. For the current value of $\omega \sim 2.38$ and $\alpha \sim 0.31$, the runtime is simply $O^*(n^\omega)$. This achieves the natural barrier for solving linear programs because linear system is a special case of linear program and that the currently fastest way to solve general linear systems involves matrix multiplication. Despite the exact approach used in [CW87, Wil12, DS13, LG14] cannot give a bound on ω better than 2.3078 [AFLG15] and all known approaches cannot achieve the bound $\omega = 2$ [AW18b], it is still possible that $\omega = 2.01$ using all known approaches. Therefore, we believe improving the additive $2 + 1/6$ term remains an interesting open problem.

Our method is a stochastic version of the short step central path method. This short step method takes $O^*(\sqrt{n})$ steps and each step decreases $x_i s_i$ by a $1 - 1/\sqrt{n}$ factor for all i where s is the dual variable [Ren88] (See the definition of s in (2.1)). This results in $O^*(\sqrt{n}) \times n = O^*(n^{1.5})$ coordinate updates. Our method takes the same number of step but only updates $\widetilde{O}(\sqrt{n})$ coordinates each step. Therefore, we only update $O^*(n)$ coordinates in total, which is nearly optimal.

Our framework is efficient enough to take a much smaller step while maintaining the same running time. For the current value of $\omega \sim 2.38$, we show how to obtain the same runtime of $O^*(n^\omega)$ by taking $O^*(n)$ steps and $\widetilde{O}(1)$ coordinates update per steps. This is because the complexity of each step decreases proportionally when the step size decreases. Beyond the cost per iteration, we remark that our algorithm is one of the very few central path algorithms [PRT02, Mad13, Mad16] that does not maintain $x_i s_i$ close to some ideal vector in ℓ_2 norm. We are hopeful that our stochastic method and our proof will be useful

[3]The dual exponent of matrix multiplication α is the supremum among all $a \geq 0$ such that it takes $n^{2+o(1)}$ time to multiply an $n \times n$ matrix by an $n \times n^a$ matrix.

Year	Author	Refs	#Iters	Cost/Iter	Total
1984	Karmarkar	[Kar84]	$O(nL)$	$\widetilde{O}(n^{2.2})$	$\widetilde{O}(n^{3.2}L)$
1986	Renegar	[Ren88]	$O(\sqrt{n}L)$		$\widetilde{O}(n^{2.7}L)$
1989	Vaidya	[Vai89b]	$O(\sqrt{n}L)$	$O(n^2)$	$O(n^{2.5}L)$
1994	Nesterov and Nemirovskii	[NN94]	$O(\sqrt{n}L)$	$O(n^2)$	$O(n^{2.5}L)$
2014	Lee, Sidford	[LS14]	$\widetilde{O}(\sqrt{n}L)$	$\widetilde{O}(n^2)$	$O(n^{2.5}L)$
2015	Lee, Sidford	[LS15]	$\widetilde{O}(\sqrt{n}L)$	$\widetilde{O}(n^2)$	$O(n^{2.5}L)$
2019	Cohen, Lee, Song	[CLS19](thesis)	$\widetilde{O}(\sqrt{n}L)$	$\widetilde{O}(n^{\omega-1/2})$	$O(n^{\omega}L)$

Table 1.1: Let ω denote the exponent of the current matrix multiplication. "Iters" denotes the "number of iterations". "Cost/Iter" denotes the "cost per iteration". LP has n variables, $d = n$ constraints, and can be encoded in L input bits. We consider the case where A is a dense full rank matrix. We remark that all of results have difference when $\mathrm{rank}(A)$ is low rank, $\mathrm{nnz}(A)$ is sparse and $n \not\approx d$. However, when $d = n$, $\mathrm{rank}(A) = n$ and $\mathrm{nnz}(A) = n^2$, the running time remains to be $O(n^{2.5})$ for three decades.

for future research on interior point methods. In particular, it would be interesting to see how this can be combined with techniques in [Cla95, LS14] to get a faster algorithm for linear programs with $d \ll n$.

Besides the applications to linear programs, some of our techniques are probably useful for studying other important problems in convex optimization. In particular, our framework should be naturally extendable to a larger class of convex programs.

Empirical Risk Minimization Empirical Risk Minimization (ERM) problem is a fundamental question in statistical machine learning. There are a huge number of papers that have considered this topic [Nes83, Vap92, PJ92, Nes04, BBM05, BB08, NJLS09, MB11, FGRW12, LRSB12, JZ13, Vap13, SSZ13, DB14, DBLJ14, FGKS15, DB16, SLC+17, ZYJ17, ZX17, ZWX+17, GSS17, MS17, NS17, AKK+17, Csi18, JLGJ18] as almost all convex optimization

machine learning can be phrased in the ERM framework [SSBD14, Vap92]. While the statistical convergence properties and generalization bounds for ERM are well-understood, a general runtime bound for general ERM is not known although fast runtime bounds do exist for specific instances [AKPS19].

Examples of applications of ERM include linear regression, LASSO [Tib96], elastic net [ZH05], logistic regression [Cox58, HJLS13], support vector machines [CV95], ℓ_p regression [Cla05, DDH+09, BCLL18, AKPS19], quantile regression [Koe00, KH01, Koe05], AdaBoost [FS97], kernel regression [Nad64, Wat64], and mean-field variational inference [XJR02].

The classical Empirical Risk Minimization problem is defined as

$$\min_x \sum_{i=1}^m f_i(a_i^\top x + b_i)$$

where $f_i : \mathbb{R} \to \mathbb{R}$ is a convex function, $a_i \in \mathbb{R}^d$, and $b_i \in \mathbb{R}$, $\forall i \in [m]$. Note that this formulation also captures most standard forms of regularization as well.

Letting $y_i = a_i^\top x + b_i$, and $z_i = f_i(a_i^\top x + b_i)$ allows us to rewrite the original problem in the following sense,

$$\begin{aligned} \min_{x,y,z} \quad & \sum_{i=1}^m z_i \\ \text{s.t.} \quad & Ax + b = y \\ & (y_i, z_i) \in K_i = \{(y_i, z_i) : f_i(y_i) \le z_i\}, \forall i \in [m] \end{aligned} \tag{1.1}$$

We can consider a more general version where dimension of K_i can be arbitrary, e.g. n_i. Therefore, we come to study the general n-variable form

$$\min_{x \in \prod_{i=1}^m K_i, Ax=b} c^\top x$$

where $\sum_{i=1}^{m} n_i = n$. We give a result for solving the general model.

First-order algorithms for ERM are well-studied and a long series of accelerated stochastic gradient descent algorithms have been developed and optimized [Nes98, JZ13, XZ14, SSZ14, FGKS15, LMH15, MLF15, AY16, RHS+16, SS16, AH16, SLRB17, MS17, LMH17, LJCJ17, All17b, All17a, All18b, All18a]. However, these rates depend polynomially on the Lipschitz constant of ∇f_i and in order to achieve a $\log(1/\epsilon)$ dependence, the runtime will also have to depend on the strong convexity of the $\sum_i f_i$. In this thesis, we want to focus on algorithms that depend logarithmically on diameter/smoothness/strong convexity constants, as well as the error parameter ϵ. Note that gradient descent and a direct application of Newton's method do not belong to these class of algorithms, but for example, interior point method and ellipsoid method does.

Therefore, in order to achieve high-accuracy solutions for non-smooth and non strongly convex case, most convex optimization problems will rely on second-order methods, often under the general interior point method (IPM) or some sort of iterative refinement framework. So, we note that our algorithm is thus optimal in this general setting since second-order methods require at least n^ω runtime for general matrix inversion.

Our algorithm applies the interior point method framework to solve ERM. The most general interior point methods require $O(\sqrt{n})$-iterations of linear system solves [Nes98], requiring a naive runtime bound of $O(n^{\omega+1/2})$. Using the inverse maintenance technique [Vai89b, CLS19], one can improve the running time for LP to $O(n^\omega)$. This essentially implies that almost all convex optimization problems can be solved, up to subpolynomial factors, as fast as linear regression or matrix inversion!

The specific case of ℓ_2 regression can be solved in $O(n^\omega)$ time since the solution is explicitly given by solving a linear system. In the more general case of ℓ_p regression, [BCLL18] proposed a $\widetilde{O}_p(n^{|1/2-1/p|})$-iteration iterative solver with a naive $O(n^\omega)$ system solve at each step. Recently, [AKPS19] improved the runtime to $\widetilde{O}_p(n^{\max(\omega,7/3)})$, which is current matrix multiplication time as $\omega > 7/3$. However, both these results depend exponentially on p and fail to be impressive for large p. Otherwise, we are unaware of other ERM formulations that have have general runtime bounds for obtaining high-accuracy solutions.

Recently several works [AW18a, AW18b, Alm19] try to show the limitation of current known techniques for improving matrix multiplication time. Alman and Vassilevska Williams [AW18b] proved limitations of using the Galactic method applied to many tensors of interest (including Coppersmith-Winograd tensors [CW87]). More recently, Alman [Alm19] proved that by applying the Universal method on those tensors, we cannot hope to achieve any running time better than $n^{2.168}$ which is already above our $n^{2+1/6}$.

Algorithmic Theory of ODEs The complexity of sampling a high-dimensional density of the form $e^{-f(x)}$ where f is a convex function is a fundamental problem with many applications [LS90, LS92, LS93, LV06a, LV06b, Dal17, DK17, DRD18, DCWY18]. The focus of this part is to give very fast, i.e., nearly linear time algorithms, for a large subclass of such densities. A motivating and important case is the loss function for logistic regression, widely used in machine learning applications [Ber44, Paa00, NJ02, HJLS13, Bou14]:

$$\sum_{i=1}^{n} \phi_i(a_i^\top x)$$

where ϕ_i are convex functions; a popular choice is $\phi(t) = \log(1 + e^{-t})$. Sampling according to e^{-f} for this choice of f corresponds to sampling models according to their KL-divergence,

a natural and effective choice for classification problems [HJLS13].

A general approach to sampling is by an ergodic Markov chain whose stationary distribution is designed to have the desired density. Traditionally, this is done via a Metropolis filter, which accepts a proposed (random) next step y from the current point x with probability $\min\{1, \frac{f(y)}{f(x)}\}$. While very general, one downside of this approach is the possibility of high rejection probabilities, which typically force local steps to be very small. Nevertheless, for arbitrary logconcave functions (including nonsmooth ones), this approach has the current best guarantees [LV06b].

Another family of algorithms is derived from an underlying *continuous* stochastic process with the desired stationary density. A classic example of such a continuous process is Brownian motion. To sample a convex body for example, one could use Brownian motion with a boundary reflection condition. This is written as the stochastic equation:

$$\mathrm{d}X_t = \mathrm{d}W_t$$

with reflection at the boundary of the domain, and dW_t being infinitesimal Brownian motion. To sample from the density proportional to $e^{-f(x)}$, one can use the stochastic differential equation,

$$\mathrm{d}X_t = -\nabla f(X_t)\mathrm{d}t + \sqrt{2}\mathrm{d}W_t.$$

By the classical Fokker-Planck equation, under mild assumptions on f, the stationary density of this process is proportional to $e^{-f(x)}$.

How can we turn these continuous processes into algorithms? One approach is to take small rather than infinitesimal steps, and this leads to the Langevin dynamics, of which there are multiple flavors [Dal17, DK17, ZLC17, RRT17, DRD18, CCBJ18, CCAY+18, CFM+18].

Year	Refs.	Method	Iters.	Gra/Iter	Total time
2006	[LV06b]	B./H.*	d^3, d^4	1	d^5, d^6
2017	[Dal17]	LMC *	$\kappa^2 d$, $\kappa^3 d^3$	1	$\kappa^2 d^2$, $\kappa^3 d^4$
2017	[Dal17]	LMCO *	$\kappa^2 d$, $\kappa^2 d^{2.5}$	1	$\kappa^2 d^4$, $\kappa^2 d^{5.5}$
2018	[CCBJ18]	DL	$\kappa^2 d^{0.5}$	1	$\kappa^2 d^{1.5}$
2018	[DCWY18]	MALA *	κd, κd^2	1	κd^2, κd^3
2017	[MS17]	HMC	$\kappa^{6.5} d^{0.5}$	1	$\kappa^{6.5} d^{1.5}$
2018	[MV18]	HMC *,†	$\kappa^{2.75} d^{0.25}$, $\kappa^{3.5} d^{0.25}$	1	$\kappa^{2.75} d^{1.25}$, $\kappa^{3.5} d^{1.25}$
2018	[LSV18]	HMC †	$\kappa^{1.5}$	1	$\kappa^{1.5} d$
	(thesis)	HMC	$\kappa^{1.5}$	$\kappa^{0.25} d^{0.5}$	$\kappa^{1.75} d^{1.5}$

* have different bounds for warm start and general (cold) start. We stated the runtime for cold start in green color.

† make smoothness and incoherence assumptions motivated by and applicable to Bayesian logistic regression.

Table 1.2: Summary of results, d is the dimension, κ is the condition number of $\nabla^2 f$. "Iters" denotes "the number of iterations / parallel depth". "Gra/Iter" denotes the "the number of gradients per iteration". We use the parallel depth of the algorithm as the number of iterations. We suppress polylogarithmic terms and dependence on the error parameter. Ball walk/hit-and-run apply to general logconcave distributions, the rest assume strongly logconcave with Lipschitz gradient and possibly more. "B./H." denotes "Ball Walk/Hit-and-run". "DL" denotes "Damped Langevin". In all previous work, for simplicity, we report the most favorable bounds by making various assumptions such as $\kappa \ll d$.

Starting with Dalalyan [Dal17], it has been established that these dynamics converge in polynomial (in dimension) time for strongly logconcave functions, with the underdamped version converging in $O(\sqrt{d})$ iterations (and polynomial dependences on appropriate condition numbers) [CCBJ18]. The dependence on dimension seems unavoidable in the discretized algorithm, even though the continuous process has no such dependence.

Hamiltonian Monte Carlo. HMC is a random process that maintains a position x and velocity pair v. To sample according to e^{-f}, we define a Hamiltonian $H(x, v) = f(x) + \frac{1}{2}\|v\|^2$.

At each step v is chosen randomly from $N(0, I)$ and x is updated using the following Ordinary Differential Equation (ODE) for a some fixed time interval.

$$\frac{\mathrm{d}x(t)}{\mathrm{d}t} = v(t), \quad \frac{\mathrm{d}v(t)}{\mathrm{d}t} = -\nabla f(x(t)). \tag{1.2}$$

This process has the particularly nice property that it conserves the value of H, and as a result there is no need to apply a Metropolis filter. HMC has been studied in many works [MS17, MV18, LV18]. Mangoubi and Smith [MS17] gave the following guarantee for strongly logconcave densities.

Note that Eq. (1.2) is a special case of the following first order ODE

$$\frac{\mathrm{d}}{\mathrm{d}t}x(t) = F(x(t), t), x(0) = v$$

where $F : \mathbb{R}^{d+1} \to \mathbb{R}^d$, $x(t) \in \mathbb{R}^d$ and $v \in \mathbb{R}^d$.

Moreover, we can write the k-th order ODE in the following way

$$\frac{\mathrm{d}^k}{\mathrm{d}t^k}x(t) = F(\frac{\mathrm{d}^{k-1}}{\mathrm{d}t^{k-1}}x(t), \cdots, x(t), t)$$
$$\frac{\mathrm{d}^i}{\mathrm{d}t^i}x(0) = v_i, \forall i \in \{k-1, \cdots, 1, 0\}.$$

where $F : \mathbb{R}^{kd+1} \to \mathbb{R}^d$, $x(t) \in \mathbb{R}^d$, and $v_0, v_1, \cdots, v_{k-1} \in \mathbb{R}^d$.

To solve HMC such special case ODE efficiently, it requires a better understanding of the ODE system. We analyze the collocation method for solving ODEs [Ise09]. This method is classical in numerical analysis.

We present the multivariate high-oder ODE guarantee. This generalizes and improves on the guarantee from [LV17]. Note that our result is a general result about solving ODE

efficiently, independent of the application to sampling. The only assumptions needed are that the ODE function is Lipschitz and that the solution is close to the span of small number of basis of functions. These natural assumptions suffice to get around the worst-case complexity lower bounds for solving such general ODEs [KF82, Ko83, Ko10, Kaw10, KC12].

Deep Neural Networks Neural networks have demonstrated a great success in numerous machine-learning tasks [KSH12, GMH13, LHP+15, AAA+16, HZRS16, SHM+16, SSS+17]. One of the empirical findings is that neural networks, trained by first-order methods from random initialization, have a remarkable ability to fit training data [ZBH+17].

From an *expressibility* perspective, this may not be surprising since modern neural networks are often over-parameterized: they have much more parameters than the number of training samples. There certainly *exist* parameter choices with zero training error as long as data is non-degenerate.

Yet, from an optimization perspective, the fact that randomly-initialized first-order methods can find global minima on the training data is *quite non-trivial*: neural networks are often equipped with the ReLU activation, making the training objective not only non-convex, but even non-smooth. Even the general convergence for finding approximate critical points of a non-convex, non-smooth function is not fully-understood [BLO05] and appears to be a challenging question on its own. This is in direct contrast to practice, in which ReLU networks trained by stochastic gradient descent (SGD) from random initialization *almost never* suffer from non-smoothness or non-convexity, and can avoid local minima for a variety of network architectures (see [GVS15]). A theoretical justification was missing to explain this phenomenon.

There are quite a few papers trying to understand the success of neural networks from optimization perspective. Many of them focus on the case when the inputs are random Gaussian, and work only for two-layer neural networks [BG17, Son17, Tia17, LY17, DLT+18, GLM17, PRSZ18, ZSJ+17, ZSD17]. [LL18] show that for a two-layer network with ReLU activation, SGD finds nearly-global optimal (say, 99% classification accuracy) solutions on the training data, as long as the network is *over-parameterized*, meaning that the number of neurons is polynomially large comparing to the input size. Moreover, if the data is sufficiently structured (say, coming from mixtures of separable distributions), this accuracy extends also to *test data*. As a separate note, over-parameterization is suggested as the possible key to avoid bad local minima by [SS18] even for two-layer networks.

There are also results that go beyond two-layer networks with limitations. Some consider deep *linear* neural networks without any activation functions [HM17, ACGH18, BHL18, Kaw16]. [Dal17] studies multi-layer neural networks but essentially only with respect to the *convex* task of training the last layer.[4] [SC16] show that under over-parameterization and under random input perturbation, there is bad local minima for multi-layer neural networks. [JGH18] derive global convergence using neural tangent kernel for *infinite-width* neural networks.

In this thesis, we study the following fundamental questions

Can DNN be trained close to zero training error efficiently under mild assumptions?

If so, can the running time depend only polynomially *in the network depth and input size?*

[4] [Dal17] works in a parameter regime where the weight changes of all layers except the last one make negligible contribution to the final output.

Motivation In 2012, AlexNet was born with 5 convolutional layers [KSH12]. The later VGG network uses 19 layers [SSZ14], and GoogleNet uses 22 layers [SLJ+15]. In practice, we cannot go deeper by naively stacking layers together, due to the so-called vanishing/exploding gradient problem. To deal with this issue, networks with residual links (ResNet) were proposed with the capability of handling at least 152 layers [HZRS16]. Compared with practical networks that go much deeper, existing theory has been mostly around two-layer (thus one-hidden-layer) neural networks, even just for the training process alone. Thus,

Can we theoretically justify how the training process has worked for multi-layer neural networks?

In this thesis, we extend the over-parameterization theory to *multi-layer* neural networks.

Recurrent Neural Networks Among different architectures of neural networks, one of the *least* theoretically-understood structure is the recurrent one [Elm90]. A recurrent neural network recurrently applies the same network unit to a sequence of input tokens, such as a sequence of words in a language sentence. RNN is particularly useful when there are long-term, non-linear interactions between input tokens in the same sequence. These networks are widely used in practice for natural language processing, language generation, machine translation, speech recognition, video and music processing, and many other tasks [MKB+10, MKB+11, SSN12, KB13, SSB14, SVL14, CVMBB14, CGCB14]. On the theory side, while there are some attempts to show that an RNN is more expressive than a feedforward neural network [KNO18], when and how an RNN can be efficiently learned has little theoretical explanation.

In practice, RNN is usually trained by simple local-search algorithms such as SGD. However, unlike shallow networks, the training process of RNN often runs into the trouble of vanishing or exploding gradient [SMH11]. That is, the value of the gradient becomes exponentially small or large in the time horizon, even when the training objective is still constant. [5]

In practice, one of the popular ways to resolve this is by the long short term memory (LSTM) structure [HS97]. However, one can also use rectified linear units (ReLUs) as activation functions to avoid vanishing or exploding gradient [SBS+17]. In fact, one of the earliest adoptions of ReLUs was on applications of RNNs for this purpose twenty years ago [Hah98, SA96].

Since the RNN structure was proposed, a large number of variations have been designed over past decades. For a more detailed survey, we refer the readers to [SBS+17].

In this thesis, we study the following general question

- *Can ReLU provably stabilize the training process and avoid vanishing/exploding gradient?*
- *Can RNN be trained close to zero training error* efficiently *under mild assumptions?*

Remark 1.2.1. When there is no activation function, RNN is known as *linear dynamical system*. Hardt, Ma and Recht [HMR18] first proved the convergence of finding global minima for

[5] Intuitively, an RNN recurrently applies the same network unit for L times if the input sequence is of length L. When this unit has "operator norm" larger than one or smaller than one, the final output can possibly exponentially explode or vanish in L. More importantly, when one back propagates through time —which intuitively corresponds to applying the reverse unit multiple times— the gradient can also vanish or explode. Controlling the operator norm of a non-linear operator can be quite challenging.

such linear dynamical systems. Followups in this line of research include [HSZ17, HLS+18].

This part is based on the following papers:

- Michael B. Cohen, Yin Tat Lee, Zhao Song

  Solving Linear Programs in the Current Matrix Multiplication Time.

  STOC 2019 [CLS19]

- Yin Tat Lee, Zhao Song, Qiuyi Zhang

  Solving Empirical Risk Minimization in the Current Matrix Multiplication.

  COLT 2019 [LSZ19]

- Yin Tat Lee, Zhao Song, Santosh S. Vempala

  Algorithmic Theory of ODEs and Sampling from Well-conditioned Logconcave Densities.

  Manuscript 2018 [LSV18]

- Zeyuan Allen-Zhu, Yuanzhi Li, Zhao Song

  A Convergence Theory for Deep Learning via Over-Parameterization.

  ICML 2019 [AZLS19]

- Zeyuan Allen-Zhu, Yuanzhi Li, Zhao Song

  On the convergence rate of training recurrent neural networks.

  Manuscript 2018 [AZLS18]

1.3 Concentration

The study of concentration of sums of random variables dates back to Central Limit Theorems, and hence de Moivre and Laplace [Tij], while modern concentration bounds for sums of random variables were perhaps first established by Bernstein [Ber24], and a popular variant now known as Chernoff bounds was introduced by Rubin and published by Chernoff [Che52].

Concentration of measure for matrix-valued random variables is the phenomenon that many matrix valued distributions are to close their mean with high probability, closeness usually being measured by spectral norm. Modern quantitative bounds of the form often used in theoretical computer science were derived by Rudelson [Rud99], while Ahlswede and Winter [AW02] established a useful matrix-version of the Laplace transform that plays a central role in scalar concentration results such as those of Bernstein. [AW02] combined this with the Golden-Thompson trace inequality to prove matrix concentration results. Tropp refined this approach, and by replacing the use of Golden-Thompson with deep a theorem on concavity of certain trace functions due to Lieb, Tropp was able to recover strong versions of a wide range of scalar concentration results, including matrix Chernoff bounds, Azuma and Freedman's inequalities for matrix martingales [Tro12].

Matrix concentration results have had an enormous range of applications in computer science, and are ubiquitous throughout spectral graph theory [ST04, SS11, CKP+17], sketching [Coh16a], approximation algorithms [HSSS16], numerical linear algebra, quantum information theory, and deep learning [ZSJ+17, ZSD17, SY19]. Most applications are based on results for independent random matrices, but more flexible bounds, such as Tropp's Matrix Freedman Inquality [Tro11a], have been used to greatly simplify algorithms, e.g.

for solving Laplacian linear equations [KS16] and for semi-streaming graph sparsification [AG09, KPPS17]. Matrix concentration results are also closely related to other popular tools sampling tools, such as Karger's techniques for generating sparse graphs that approximately preserve the cuts of denser graphs [BK96].

We formally state the natural generalization of the Chernoff bound appeared in the works of Rudelson [Rud99], Ahlswede-Winter [AW02], and Tropp [Tro12]. They showed that a similar concentration phenomenon is true for *matrix-valued* random variables. In particular, if $X_1, \ldots, X_k$ are independent $d \times d$ complex Hermitian random matrices with $\|X_i\| \leq 1$, then the following is true:

$$\Pr\left[\left\|\frac{1}{k}\sum_{i=1}^{k} X_i - \mathbb{E}[X]\right\| > \epsilon\right] \leq 2d \cdot \exp(-\Omega(k\epsilon^2)). \tag{1.3}$$

The only difference between this and the usual Chernoff bound is the factor of d in front of the deviation probability; to see that it is necessary, notice that the diagonal case simply corresponds to a direct sum of d arbitrarily correlated instances of the scalar Chernoff bound, so by the union bound the probability should be d times as large in the worst case. This so called "Matrix Chernoff Bound".

Expander Walk An important generalization of this bound was achieved by Gillman [Gil98] (with refinements later by [Lez98, Kah97, LP04, WX05, Hea08, Wag08, CLLM12, RR17]), who significantly relaxed the independence assumption to Markov dependence. In particular, suppose G is a regular graph with vertex set $V = [n]$, $X : V \to \mathbb{C}$ is a bounded function, and $v_1, \ldots, v_k$ is a stationary random walk[6] of length k on G. Then, even though

[6]That is the first vertex v_1 is chosen uniformly at random – which is the stationary distribution of the graph G.

the random variables $X(v_i)$ are in not independent (except when G is the complete graph with self loops), it is shown that:

$$\Pr\left[\left|\frac{1}{k}\sum_{i=1}^{k} X(v_i) - \mathbb{E}[X]\right| > \epsilon\right] \leq 2 \cdot \exp(-\Omega((1-\lambda)k\epsilon^2)), \tag{1.4}$$

where $1-\lambda$ is the spectral gap of the transition matrix of the random walk. The gain here is that sampling a stationary random walk of length k on a constant degree graph with constant spectral gap requires $\log(n) + O(k)$ random bits, which is much less than the $k\log(n)$ bits required to produce k independent samples. Since such graphs can be explicitly constructed, this leads to a generic "derandomization" of the Chernoff bound, which has had several important applications (see [WX05] for a detailed discussion). In particular, it leads to the following randomness efficient sampler for scalar-valued functions ([Gil98]) using known strongly explicit constructions of expander graphs [RVW00, LPS88]:

Theorem 1.3.1 ([Gil98])**.** *For any $\epsilon > 0$ and $k \geq 1$, there is a $\mathrm{poly}(r)$-time computable sampler $\sigma : \{0,1\}^r \to [n]^k$, where $r = \log(n) + O(k)$ s.t. for all functions $f : [n] \to [-1,1]$ satisfying $\mathbb{E}\, f = 0$, we have that*

$$\Pr_{w \in_R \{0,1\}^r}\left[\left|\frac{1}{k}\sum_{i=1}^{k} f(\sigma(w)_i)\right| \geq \epsilon\right] \leq 2\exp\left(-\Omega\left(\epsilon^2 k\right)\right).$$

In many applications of interest k is about $\log(n)$, and going from $O(\log^2(n))$ to $O(\log(n))$ random bits leads to a complete derandomization by cycling over all seeds $w \in \{0,1\}^r$.

It is natural to wonder whether there is a common generalization of (1.4) and (1.3), i.e., a "Matrix Expander Chernoff Bound". Such a result was conjectured by Wigderson and

Xiao in [WX06]. In this thesis, we prove the Wigderson and Xiao conjecture. In order to prove that conjecture, we also propose a new Golden-Thompson inequality. The original Golden-Thompson inequality only holds for two matrices [Gol65, Tho65]. Our new Golden-Thompson inequality holds for multiple matrices.

Strongly Rayleigh and Matrix Chernoff for Random Spanning Trees Negative dependence of random variables is an appealing property that intuition suggests should help with concentration of measure. Notions of negative dependence can be formalized in many ways. Roughly speaking, these notions characterize distributions where where some event occurring ensures that other events of interest become less likely. A simple example is the distribution of a sequence of coin flips, conditioned on the total number of heads in the outcome. In this distribution, conditioning on some coin coming out heads makes all other coins less likely to come out heads. Unfortunately, negative dependence phenomena are not as robust as positive association which can be established from local conditions using the powerful FKG theorem [FKG71].

Strongly Rayleigh distributions were introduced recently by Borcea, Brändén, and Liggett [BBL09] as a class of negatively dependent distributions of binary-valued random variables with many useful properties. Strongly Rayleigh distributions satisfy useful negative dependence properties, and retain these properties under natural conditioning operations. Strongly Rayleigh distributions also satisfy a powerful stability property under conditioning known as *Stochastic Covering* [PP14], which is useful for analyzing them through martingale techniques. A measure on $\{0,1\}^n$ is said to be Strongly Rayleigh if its generating polynomial is real stable [BBL09]. There are many interesting examples of Strongly Rayleigh distribu-

tions [PP14]: The example mentioned earlier of heads of independent coin flips conditional on the total number of heads in the outcome; symmetric exclusion processes; determinental point processes and determinental measures on a boolean lattice. An example of particular interest to us is the edges of uniform or weighted random spanning trees, which form a Strongly Rayleigh distribution.

We prove a Matrix Chernoff bound for the case of k-homogeneous Strongly Rayleigh distributions. Our bound is slightly weaker than the bound for independent variables. We give lower bounds that show our bounds are close to tight in some regimes, but importantly, our lower bounds do not establish separation from the behaviour of indepedent random matrices, leaving open the question of whether the true bound should match the independent case in all regimes – which seems plausible.

We use our bound to show new concentration results related to random spanning trees of graphs. Random spanning trees are one among the most well-studied probabilistic objects in graph theory, going back to the work of Kirchoff [Kir47] in 1847, who gave formula relating the number of spanning trees in a graph to the determinant of the Laplacian of the same graph.

Algorithms for sampling of random spanning trees have been studied extensively, [Gue83, Bro89, Ald90, Kul90, Wil96, CMN96, KM09, MST15, HX16, DKP$^+$17, DPPR17, Sch18], and a random spanning tree can now be sampled in almost linear time [Sch18]. There is also a long line [BK96, ST11, SS11, BSS12, Zou12, LS17, LS18] of research about generating graph spectral sparsifier as fast as possible. However, the connection between those two objects is open for decades. We answer the question positively by providing the following result

Theorem 1.3.2. *Given as input a weighted graph G with n vertices and a parameter $\epsilon > 0$, let $T_1, T_2, \cdots, T_t$ denote t independent inverse leverage score weighted random spanning trees, if we choose $t = O(\epsilon^{-2} \log^2 n)$ then with probability $1 - 1/\operatorname{poly}(n)$,*

$$(1-\epsilon)L_G \preceq \frac{1}{t}\sum_{i=1}^{t} L_{T_i} \preceq (1+\epsilon)L_G.$$

In addition, we also show $t = \Omega(\epsilon^{-2} \log n)$ is necessary.

Discrepancy Discrepancy theory is an area of combinatorics that studies how well continuous objects can be approximated by discrete ones. It lies at the heart of numerous problems in mathematics and computer science [Cha00]. Although closely tied to probability theory, direct randomized approaches rarely yield the best bounds. In a classical formulation in discrepancy theory, we have n sets on n elements, and would like to two-color the elements so that each set has roughly the same number of elements of each color. Using a simple random coloring, it is an easy consequence of Chernoff's bound that there exists a coloring such that the discrepancy in all n sets is $O(\sqrt{n \log n})$ [AS16]. However, in a celebrated result, Spencer showed that in fact there is a coloring with discrepancy at most $6\sqrt{n}$ [Spe85].

Recently, there has been significant success in generalizing Chernoff [Che52], Hoeffding, Bernstein, and Bennett-type concentration bounds for scalar random variables to matrix-valued random variables [Rud99, AW02, Tro12]. Consider the following matrix concentration bound, which is a direct consequence of a matrix Hoeffding bound.

Theorem 1.3.3 ([Tro12])**.** *Let $\xi_i \in \{\pm 1\}$ be independent, symmetric random signs and $A_1, \ldots, A_n \in \mathbb{C}^{m \times m}$ be positive semi-definite matrices. Suppose $\max_{i \in [n]} \|A_i\| \leq \epsilon$ and*

$\|\sum_{i=1}^n A_i\| \leq 1$. *Then,*

$$\Pr\left[\left\|\sum_{i=1}^n \xi_i A_i\right\| \geq t\sqrt{\epsilon}\right] \leq 2m\exp(-t^2/2).$$

A consequence of this theorem is that with high probability

$$\left\|\sum_{i=1}^n \xi_i A_i\right\| = O(\sqrt{\log m})\sqrt{\epsilon}. \tag{1.5}$$

The Kadison-Singer theorem of [MSS15b] is essentially equivalent to the following statement (which can readily be derived from the bipartition statement in [MSS15b]).

Theorem 1.3.4 (Kadison-Singer [MSS15b])**.** *Let* $u_1, \ldots, u_n \in \mathbb{C}^m$ *and suppose* $\max_{i\in[n]} \|u_i u_i^*\| \leq \epsilon$ *and* $\sum_{i=1}^n u_i u_i^* = I$*. Then, there exists signs* $\xi_i \in \{\pm 1\}$ *s.t.*

$$\left\|\sum_{i=1}^n \epsilon_i u_i u_i^*\right\| \leq O(\sqrt{\epsilon}).$$

Thus, for rank 1 matrices the theorem improves on the norm bound in Equation (1.5), by a factor $\sqrt{\log m}$, in a manner analogous to the improvement of Spencer's theorem over the bound based on the scalar Chernoff bound.

For random signings of matrices one can establish bounds in some cases that are much stronger than Theorem 1.3.3.

Theorem 1.3.5 ([Tro12])**.** *Let* $\xi_i \in \{\pm 1\}$ *be independent random signs, and let* $A_1, \ldots, A_n \in \mathbb{C}^{m\times m}$ *be Hermitian matrices. Let* $\sigma^2 = \|\sum_{i=1}^n \mathbb{V}[\xi_i]A_i^2\|$*. Then,*

$$\Pr\left[\left\|\sum_{i=1}^n \mathbb{E}[\xi_i]A_i - \sum_{i=1}^n \xi_i A_i\right\| \geq t\cdot\sigma\right] \leq 2m\exp(-t^2/2).$$

From this theorem we deduce that with high probability

$$\left\| \sum_{i=1}^{n} \mathbb{E}[\xi_i] A_i - \sum_{i=1}^{n} \xi_i A_i \right\| = O(\sqrt{\log m})\sigma. \tag{1.6}$$

Of course, this implies that there exists a choice of signs $\epsilon_1, \ldots, \epsilon_n \in \{\pm 1\}$ such that

$$\left\| \sum_{i=1}^{n} \mathbb{E}[\xi_i] A_i - \sum_{i=1}^{n} \epsilon_i A_i \right\| = O(\sqrt{\log m})\sigma.$$

For rank-1 matrices, we want to show there exists a choice of signs with a stronger guarantee. Formally speaking, we can prove the following result.

Theorem 1.3.6. *Consider any independent scalar random variables $\xi_1, \ldots, \xi_n$ with finite support. Let $u_1, \ldots, u_n \in \mathbb{C}^m$ and*

$$\sigma^2 = \left\| \sum_{i=1}^{n} \mathbb{V}[\xi_i](u_i u_i^*)^2 \right\|.$$

Then there exists a choice of outcomes $\epsilon_1, \ldots, \epsilon_n$ in the support of $\xi_1, \ldots, \xi_n$

$$\left\| \sum_{i=1}^{n} \mathbb{E}[\xi_i] u_i u_i^* - \sum_{i=1}^{n} \epsilon_i u_i u_i^* \right\| \leq O(\sigma).$$

This part is based on the following papers:

- Ankit Garg, Yin Tat Lee, Zhao Song, Nikhil Srivastava

  A Matrix Expander Chernoff Bound.

  STOC 2018 [GLSS18]

- Rasmus Kyng, Zhao Song

  A Matrix Chernoff Bound for Strongly Rayleigh Distributions and Spectral Sparsifiers from a few Random Spanning Trees.

  FOCS 2018 [KS18]

- Rasmus Kyng, Kyle Luh, Zhao Song

  Four Deviations Suffice for Rank 1 Matrices.

  Manuscript 2019 [KLS19]

1.4 Compressed Sensing and Sparse Fourier Transform

Compressed Sensing, or sparse recovery, is a powerful mathematical framework the goal of which is to reconstruct an approximately k-sparse vector $x \in \mathbb{R}^n$ from linear measurements $y = \Phi x$, where $\Phi \in \mathbb{R}^{m \times n}$. The most important goal is to reduce the number of measurements m needed to approximate the vector x, avoiding the linear dependence on n. In discrete signal processing, where this framework was initiated [CRT06b, Don06], the core principle that the sparsity of a signal can be exploited to recover it using much fewer samples than the Shannon-Nyquist Theorem. We refer to the matrix Φ as the sketching or sensing matrix, and $y = \Phi x$ as the sketch of vector x.

Sparse recovery is the primary task of interest in a number of applications, such as image processing [TLW+06, LDP07a, DDT+08], design pooling schemes for biological tests [ECG+09, DWG+13], pattern matching [CEPR07], combinatorial group testing [SAZ09, ESAZ09, KBG+10], localizing sources in sensor networks [ZBSG05, ZPB06], as well as neuroscience [GS12]. Furthermore, not surprisingly, tracking heavy hitters in data streams, also known as frequent items, can be captured by the sparse recovery framework [Mut05a, CH09, KSZC03, Ind07]. In practice, streaming algorithms for detecting heavy hitters have been used to find popular destination addresses and heavy bandwidth users by AT&T [CJK+04] or answer "iceberg queries" in databases [FSGM+99].

Sparse recovery attracts researchers from different communities, from both theoret-

ical and practical perspective. During the last ten years, hundreds of papers have been published by theoretical computer scientists, applied mathematicians and electrical engineers that specialize in compressed sensing. While numerous algorithms using space linear in the universe size n are known, [Don06, CRT06b, IR08a, NT09a, BIR08a, BD09a, SV16] to name a few, our goal is to obtain algorithms that are sublinear, something that is crucial in many applications.

The desirable quantities we want to optimize may vary depending on the application. For example, in network management, x_i could denote the total number of packets with destination i passing through a network router. In such an application, storing the sketching matrix explicitly is typically not a tenable solution, since this would lead to an enormous space consumption; the number of possible IP addresses is 2^{32}. Moreover, both the query and the update time should be very fast, in order to avoid congestion on the network. Incremental updates to x come rapidly, and the changes to the sketch should also be implemented very fast; we note that in this case, even poly-logarithmic factors might be prohibitive. Interested readers can refer to [KSZC03, EV03] for more information about streaming algorithms for network management applications.

"The goal of that research is to obtain encoding and recovery schemes with good compression rate (i.e., short sketch lengths) as well as good algorithmic properties (i.e., low encoding, update and recovery times)." – *Anna Gilbert and Piotr Indyk* [GI10]

In this thesis, we consider the extensively studied problem of L2/L2 compressed sensing. The main contribution of our work is an improvement over [GLPS10] with faster decoding time and significantly smaller column sparsity, answering two open questions of the aforementioned work.

Year	Reference	Measurements	Decoding Time	Encoding Time
2006	[Don06, CRT06b]	$k \log(n/k)$	LP	$k \log(n/k)$
2006	[CCF02, CM06]	$\epsilon^{-2} k \log n$	$\epsilon^{-1} n \log n$	$\log n$
2008	[NT09a]	$k \log(n/k)$	$nk \log(n/k)$	$\log(n/k)$
2004	[CM04]	$\epsilon^{-2} k \log^2 n$	$\epsilon^{-1} k \log^c n$	$\log^2 n$
2006	[CCF02, CM06]	$\epsilon^{-2} k \log^c n$	$\epsilon^{-1} k \log^2 n$	$\log^c n$
2010	[GLPS10]	$\epsilon^{-1} k \log(n/k)$	$\epsilon^{-1} k \log^c n$	$\log(n/k) \cdot \log^2 k$
2019	[NS19] (thesis)	$\epsilon^{-1} k \log(n/k)$	$\epsilon^{-1} k \log^2(n/k)$	$\log(n/k)$

Table 1.3: (A list of ℓ_2/ℓ_2-sparse recovery results). We ignore the "O" for simplicity. LP denotes the time of solving Linear Programs [CLS19], and the state-of-the-art algorithm takes n^ω time where ω is the exponent of matrix multiplication. The results in [Don06, CRT06b, NT09a] do not explicitly state the ℓ_2/ℓ_2 guarantee, but their approach obtains it by an application of the Johnson-Lindenstrauss Lemma; they also cannot facilitate $\epsilon < 1$, obtaining thus only a 2-approximation. The c in previous work is a sufficiently large constant, not explicitly stated, which is defined by probabilistically picking an error-correcting code of short length and iterating over all codewords. We estimate $c \geq 4$. We note that our runtime is (almost) achieved.

Previous work on sublinear-time compressed sensing employed an iterative procedure, recovering the heavy coordinates in phases. We completely depart from that framework, and give the first sublinear-time L2/L2 scheme which achieves the optimal number of measurements without iterating; this new approach is the key step to our progress. Towards that, we satisfy the L2/L2 guarantee by exploiting the heaviness of coordinates in a way that was not exploited in previous work. Via our techniques we obtain improved results for various sparse recovery tasks, and indicate possible further applications to problems in the field, to which the aforementioned iterative procedure creates significant obstructions.

High-dimensional Discrete Fourier Transform Probably the most important subtopic of compressed sensing/sparse recovery is the sparse Fourier transform, where one desires to

reconstruct a k-sparse vector from Fourier measurements. In other words, measurements are not allowed to be generic, but have to belong to the so-called Fourier ensemble. In Optics imaging [Goo05, Voe11] and Magnetic resonance imaging (MRI) [ASSN08], the physics [Rey89] of the underlying device restricts us to the Fourier ensemble, where the sparse Fourier problem becomes highly relevant. In fact, one of the initial motivations of Candes, Romberg and Tao came out due to the aforementioned applications. The number of samples plays a crucial role: they determine the amount of radiation a patient receives in CT scans, and taking fewer samples can reduce the amount of time the patient needs to stay in the machine. The framework has found its way in practical life-changing applications. Software includes the Compressed Sensing GRAB-VIBE, CS SPACE, CS SEMAC and CS TOF by Siemens [Sie], as well as Compressed Sense by Phillips [Phi]. Its incorporation in the MRI technology allows faster acquisition rates, depiction of dynamic processes or moving organs, as well as acceleration of MRI scanning up to a factor of 40. On the webpage of SIEMENS Healthineers, for example, one can see the following, as well as numerous similar statements.

This allows bringing the advantages of Compressed Sensing GRASP-VIBE *to daily clinical routine.*

- *Perform push-button, free-breathing liver dynamics.*
- *Overcome timing challenges in dynamic imaging and respiratory artifacts.*
- *Expand the patient population eligible for abdominal* MRI.

The Fourier transform is in fact ubiquitous: image processing, audio processing, telecommunications, seismology, polynomial multiplication, Subset Sum and other text-

Year	Reference	Samples	Time	Filter	RIP	Gua.
2005	[GMS05]	$k \log^{O(d)} n$	$k \log^{O(d)} n$	Yes	No	ℓ_2/ℓ_2
2006	[CT06]	$k \log^6 n$	$\text{poly}(n)$	No	Yes	ℓ_2/ℓ_1
2008	[RV08]	$k \log^2 k \log(k \log n) \log n$	$\widetilde{O}(n)$	No	Yes	ℓ_2/ℓ_1
2012	[HIKP12a]	$k \log^d n \log(n/k)$	$k \log^d n \log(n/k)$	Yes	No	ℓ_2/ℓ_2
2013	[CGV13]	$k \log^3 k \log n$	$\widetilde{O}(n)$	No	Yes	ℓ_2/ℓ_1
2014	[IK14]	$2^{d \log d} k \log n$	$\widetilde{O}(n)$	Yes	No	ℓ_∞/ℓ_2
2014	[Bou14]	$k \log k \log^2 n$	$\widetilde{O}(n)$	No	Yes	ℓ_2/ℓ_1
2016	[HR16]	$k \log^2 k \log n$	$\widetilde{O}(n)$	No	Yes	ℓ_2/ℓ_1
2016	[Kap16]	$2^{d^2} k \log n \log \log n$	$2^{d^2} k \log^{d+3} n$	Yes	No	ℓ_2/ℓ_2
2019	[KVZ19]	$k^3 \log^2 k \log^2 n$	$k^3 \log^2 k \log^2 n$	Yes	Yes	E.k
2019	[NSW19a] (thesis)	$k \log k \log n$	$\widetilde{O}(n)$	No	No	ℓ_∞/ℓ_2

Table 1.4: $n = p^d$. E.k denotes the "Exactly k-sparse" setting, which is a noiseless setting. "Gua" denotes "Guarantee". We ignore the O for simplicity. The ℓ_∞/ℓ_2 is the strongest possible guarantee, with ℓ_2/ℓ_2 coming second, ℓ_2/ℓ_1 third and exactly k-sparse being the less strong. We note that [CT06, RV08, CGV13, Bou14, HR16] obtain a uniform guarantee, i.e. with $1 - 1/\text{poly}(n)$ they allow reconstruction of all vectors; ℓ_∞/ℓ_2 and ℓ_2/ℓ_2 are impossible in the uniform case [CDD09]. We also note that [RV08, CGV13, Bou14, HR16] give improved analysis of the Restricted Isometry property; the algorithm is suggested and analyzed (modulo the RIP property) in [BD08]. The work in [HIKP12a] does not explicitly state the extension to the d-dimensional case, but can easily be inferred from the arguments. [HIKP12a, IK14, Kap16, KVZ19] work when the universe size in each dimension are powers of 2. We also assume that the signal-to-noise ratio is bounded by a polynomial of n, which is a standard assumption in the sparse Fourier transform literature [HIKP12a, IK14, Kap16, Kap17, LN19].

book algorithms are a few of the examples where the Fast Fourier Transform finds applications. The Fast Fourier Transform by Cooley and Tukey [CT65] runs in $O(n \log n)$ time, and has far-reaching applications in all of the aforementioned cases. It is thus expected that algorithms which exploit sparsity assumptions about the input, and can outperform FFT in applications are of high practical value. Generally, the two most important parameters one

would like to optimize are the sample complexity, i.e. the numbers needed to obtain from the time domain, as well as time needed to approximate the Fourier Transform.

Two different lines of research exist for the problem: the one focuses solely on sample complexity, while the other tries to achieve sublinear time while keeping the sample complexity as low as possible. The first line of research operates via the renowned Restricted Isometry Property (RIP), which proceeds by taking random samples and solving a linear/convex program, or an iterative thresholding procedure [CT06, DDTS06, TG07, BD08, DM08, RV08, BD09b, BD09a, NT09b, NV09, GK09, BD10, NV10, Fou11, Bou14, HR16]. The analysis of the algorithms is performed in the following way, in two steps. The first step ensures that, after sampling an appropriate number of points from the time domain, the inverse DFT matrix restricted on the rows indexed by those points acts as a near isometry on the space of k-sparse vectors. All of the state of the art results [CT06, RV08, Bou14, HR16] employ chaining arguments to make the analysis of this sampling procedure as tight as possible. The second part is how to exploit the aforementioned near-isometry property to find the best k-sparse approximation to the signal. There the approaches either follow an iterative procedure which gradually denoise the signal [BD08, NT09b, NV09], or perform ℓ_1 minimization [CT06], a method that promotes sparsity of solutions.

The second line of research tries to implement arbitrary linear measurements via sampling Fourier coefficients [GL89, Man92, KM93, GGI+02, AGS03, GMS05, Iwe08, Iwe10, HIKP12a, HIKP12b, LWC13, Iwe13, PR14, IKP14, IK14, Kap16, Kap17, CI17, BZI17, MZIC17, LN19] and use sparse functions (in the time domain) which behave like bandpass filters in the frequency domain. The seminal work of Kapralov [Kap17] achieves $O(k \log n)$ samples and running time that is some log factors away from the sample complexity. This

would be the end of the story, apart from the fact that this algorithm does not scale well with dimension, since it has an exponential dependence on d. Indeed, in many applications, one is interested in higher dimensions, rather than the one-dimensional case. The main reason[7] why this curse of dimensionality appears is due to the lack of dimension-independent ways to construct functions that approximate the ℓ_∞ ball and are sufficiently sparse in the time domain. A very nice work of Kapralov, Velingker and Zandieh [KVZ19] tries to remedy that by combining the standard execution of FFT with careful aliasing, but their algorithm works in a noiseless setting, and has a polynomial, rather than linear, dependence on k; the running time is polynomial in $k, \log n$ and the exponential dependence is avoided. It is an important and challenging question whether a robust and more efficient algorithm can be found.

We note that in many applications, such as MRI or computed tomography (CT), the main focus is the sample complexity; the algorithms that have found their way to industry are, to the best of our knowledge, not concerned with sublinear running time, but with the number of measurements, which determine the acquisition time, or in CT the radiation dose the patient receives.

In this work, we consider the extensively studied problem of computing a k-sparse approximation to the d-dimensional Fourier transform of a length n signal. Our algorithm uses $O(k \log k \log n)$ samples, is dimension-free, operates for any universe size, and achieves the strongest ℓ_∞/ℓ_2 guarantee, while running in time comparable to the Fast Fourier Transform. All previous algorithms proceed either via the Restricted Isometry Property or via

[7]But not the only one: pseudorandom permutations for sparse FT in high dimensions also incur an exponential loss, and it is not known whether this can be avoided.

Year	Refs.	Duration	Sample/Time	Approx. Ratio
2012	[BCG+12]	$> \frac{1}{\eta} \cdot k$	sublinear	NO
2015	[Moi15]	$> \frac{1}{\eta}$	linear	poly(k)
2015	[PS15] (thesis)	$> \frac{1}{\eta} \cdot \log(k)$	sublinear	$O(\log k)$
2015	[PS15] (thesis)	$> \frac{1}{\eta} \cdot \log^2(k)$	sublinear	$O(1)$
2016	[CKPS16] (thesis)	> 0	sublinear	$O(1)$

Table 1.5: Reconstructing the signal in the continuous setting. Let η denote the frequency gap. The first three results requires a frequency gap, and the last result doesn't require a frequency gap. We reconstruct an $x'(t)$ such that $\frac{1}{T}\int_0^T |x'(t) - x(t)|^2 \mathrm{d}t \leq \alpha \cdot \frac{1}{T}\int_0^T |g(t)|^2 \mathrm{d}t$, where α is the Approx. Ratio.

filter functions. Our approach totally departs from the aforementioned techniques, and we believe it is a fresh look to the sparse Fourier transform problem.

Continuous Fourier Transform In many situations, much of the reason for using Fourier transforms is because the transformed signal is *sparse*—i.e., the energy is concentrated in a small set of k locations. In such situations, one could hope for a dependency that depends nearly linearly on k rather than n. Moreover, one may be able to find these frequencies while only sampling the signal for some period of time. This idea has led to a number of results on *sparse Fourier transforms*, including [GGI+02, GMS05, HIKP12a, IK14], that can achieve $O(k \log(n/k) \log n)$ running time and $O(k \log(n/k))$ sample complexity (although not quite both at the same time) in a robust setting.

These works apply to the discrete Fourier transform, but lots of signals including audio or radio originally come from a continuous domain. The standard way to convert a continuous Fourier transform into a discrete one is to apply a window function then subsample. Unfortunately, doing so "smears out" the frequencies, blowing up the sparsity. Thus,

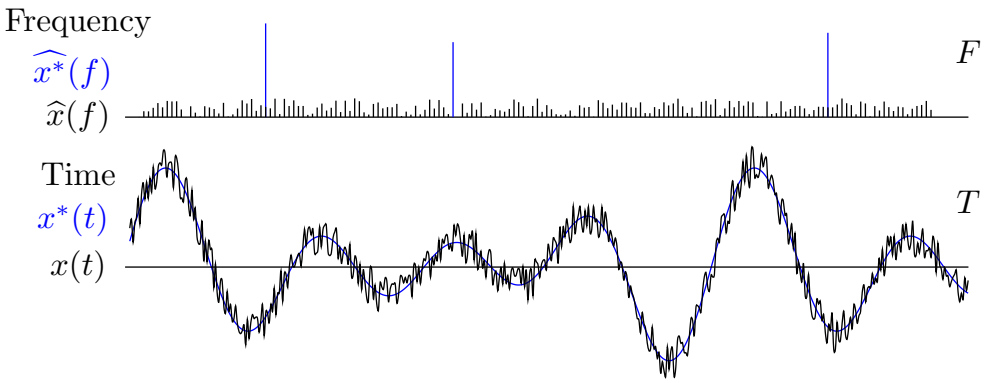


Figure 1.2: Sampling Model. x^* is a perfect signal. We are only allowed to samples from x which is a signal that has noise.

one can hope for significant efficiency gains by directly solving the sparse Fourier transform problem in the continuous setting. This has led researchers to adapt techniques from the discrete setting to the continuous both in theory [BCG+12, TBSR13, CF14, DB13] and in practice [SAH+13]. However, these results are not robust to noise: if the signal is sampled with a tiny amount of Gaussian noise or decays very slightly over time, no method has been known for computing a sparse Fourier transform in the continuous setting with sample complexity linear in k and logarithmic in other factors. That is what we present in this paper.

Formally, a vector $x^*(t)$ has a k-sparse Fourier transform if it can be written as

$$x^*(t) = \sum_{i=1}^{k} v_i e^{2\pi \mathbf{i} f_i t}$$

for some *tones* $\{(v_i, f_i)\}$. We consider the problem where we can sample some signal

$$x(t) = x^*(t) + g(t)$$

at any t we choose in some interval $[0, T]$, where $x^*(t)$ has a k-sparse Fourier transform and

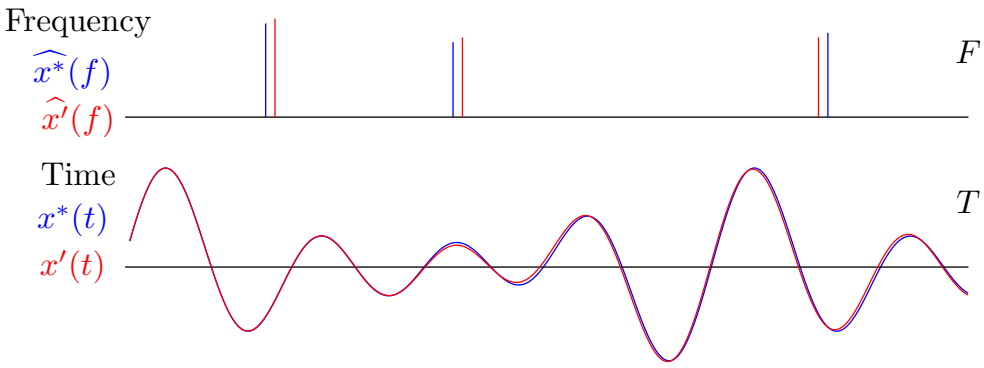


Figure 1.3: Recovered signal. Our goal is to output signal x' such that x' is close to x^*.

$g(t)$ is arbitrary noise. As long as g is "small enough," one would like to recover a good approximation to x (or to x^*, or to $\{(v_i, f_i)\}$) using relatively few samples $t \in [0, T]$ and fast running time. Our algorithm achieves several results of this form, but a simple one is an ℓ_2/ℓ_2 guarantee: we reconstruct an $x'(t)$ with k-sparse Fourier transform such that

$$\frac{1}{T}\int_0^T |x'(t) - x(t)|^2 \mathrm{d}t \lesssim \frac{1}{T}\int_0^T |g(t)|^2 \mathrm{d}t$$

using a number of samples that is k times logarithmic factors[8]. To the best of our knowledge, this is the first algorithm achieving such a constant factor approximation with a sample complexity sublinear in T and the signal-to-noise ratio.

Our algorithm also gives fairly precise estimates of the individual tones (v_i, f_i) of the signal x^*. To demonstrate what factors are important, it is helpful to think about a concrete setting. Let us consider sound from a simplified model of a piano.

Thought experiment: piano tuning In a simplified model of a piano, we have keys corresponding to frequencies over some range $[-F, F]$. The noise $g(t)$ comes from ambient

[8]We use $f \lesssim g$ to denote that $f \leq Cg$ for some universal constant C.

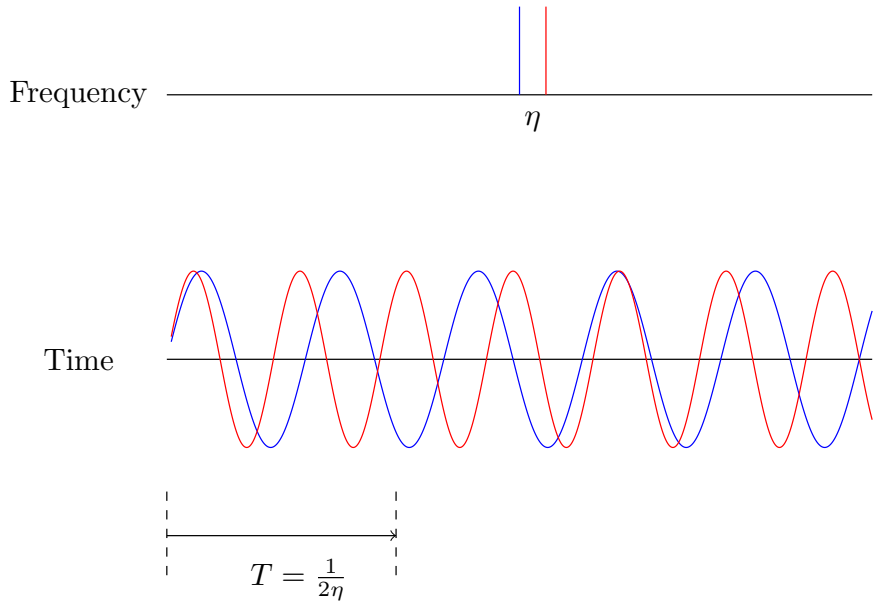


Figure 1.4: Thought Experiments

noise and the signals not being pure tones (because, for example, the notes might decay slowly over time). For concrete numbers, a modern piano has 88 keys spaced from about 27.5 Hz to $F = 4200$Hz. The space between keys ranges from a few Hz to a few hundred Hz, but most chords will have an $\eta = 30$Hz or more gap between the frequencies being played. One typically would like to tune the keys to within about $\pm\nu = 1$Hz. And piano music typically has k around 5.

Now, suppose you would like to build a piano tuner that can listen to a chord and tell you what notes are played and how they are tuned. For such a system, how long must we wait for the tuner to identify the frequencies? How many samples must the tuner take? And how robust is it to the noise?

If you have a constant signal-to-noise ratio, you need to sample for a time T of at

least order $1/\nu = 1$ second in order to get 1Hz precision—frequencies within 1Hz of each other will behave very similarly over small fractions of a second, which noise can make indistinguishable. You also need at least $\Omega(k \log(\frac{F}{k\nu})) \approx 50$ samples, because the support of the signal contains that many bits of information and you only get a constant number per measurement (at constant SNR). At higher signal-to-noise ratios ρ, these results extend to $\Omega(\frac{1}{\nu\rho})$ duration and $\Omega(k \log_\rho(\frac{F}{k\nu}))$ samples. But as the signal-to-noise ratio gets very high, there is another constraint on the duration: for $T < \frac{1}{\eta} \approx 33$ milliseconds the different frequencies start becoming hard to distinguish, which causes the robustness to degrade exponentially in k [Moi15] (though the lower bound there only directly applies to a somewhat restricted version of our setting).

This suggests the form of a result: with a duration $T > \frac{1}{\eta}$, one can hope to recover the frequencies to within $\frac{1}{\rho T}$ using $O(k \log_\rho(\frac{FT}{k}))$ samples. We give an algorithm that is within logarithmic factors of this ideal: with a duration $T > \frac{O(\log(k/\delta))}{\eta}$, we recover the frequencies to within $O(\frac{1}{\rho T})$ using $O(k \log_\rho(FT) \cdot \log(k/\delta) \cdot \log k)$ samples, where ρ and $1/\delta$ are (roughly speaking) the minimum and maximum signal-to-noise ratios that you can tolerate, respectively.

Instead of trying to tune the piano by recovering the frequencies precisely, one may simply wish to record the sound for future playback with relatively few samples. Our algorithm works for this as well: the combination $x'(t)$ of our recovered frequencies satisfies

$$\frac{1}{T}\int_0^T |x'(t) - x(t)|^2 \mathrm{d}t \lesssim \frac{1}{T}\int_0^T |g(t)|^2 \mathrm{d}t.$$

Continuous Fourier Transform without Frequency Gap In an interpolation problem, one can observe $x(t) = x^*(t) + g(t)$, where $x^*(t)$ is a structured signal and $g(t)$ denotes noise,

at points t_i of one's choice in some interval $[0, T]$. The goal is to recover an estimate $\widetilde{x}$ of x^* (or of x). Because we can sample over a particular interval, we would like our approximation to be good on that interval, so for any function $y(t)$ we define

$$\|y\|_T^2 = \frac{1}{T}\int_0^T |y(t)|^2 \mathrm{d}t.$$

to be the ℓ_2 error on the sample interval. For some parameters C and δ, we would then like to get

$$\|\widetilde{x} - x^*\|_T \leq C\,\|g\|_T + \delta\,\|x^*\|_T \tag{1.7}$$

while minimizing the number of samples and running time. Typically, we would like C to be $O(1)$ and to have δ be very small (either zero, or exponentially small). Note that, if we do not care about changing C by $O(1)$, then by the triangle inequality it doesn't matter whether we want to estimate x^* or x (i.e. we could replace the LHS of (1.7) by $\|\widetilde{x} - x\|_T$).

Of course, to solve an interpolation problem one also needs x^* to have structure. One common form of structure is that x^* have a sparse Fourier representation. We say that a function x^* is k-Fourier-sparse if it can be expressed as a sum of k complex exponentials:

$$x^*(t) = \sum_{j=1}^{k} v_j e^{2\pi \mathbf{i} f_j t}.$$

for some $v_j \in \mathbb{C}$ and $f_j \in [-F, F]$, where F is the "bandlimit". Given F, T, and k, how many samples must we take for the interpolation (1.7)?

If we ignore sparsity and just use the bandlimit, then Nyquist sampling and Shannon-Whittaker interpolation uses $FT + 1/\delta$ samples to achieve (1.7). Alternatively, in the absence of noise, x^* can be found from $O(k)$ samples by a variety of methods, including Prony's

method from 1795 or Reed-Solomon syndrome decoding [Mas69], but these methods are not robust to noise.

If the signal is periodic with period T—i.e., the frequencies are multiples of $1/T$—then we can use sparse discrete Fourier transform methods, which take $O(k \log^c(FT/\delta))$ time and samples (e.g. [GGI+02, HIKP12a, IKP14]). If the frequencies are not multiples of $1/T$ (are "off the grid"), then the discrete approximation is only k/δ sparse, making the interpolation less efficient; and even this requires that the frequencies be well separated.

A variety of algorithms have been designed to recover off-grid frequencies directly, but they require the minimum gap among the frequencies to be above some threshold. With frequency gap at least $1/T$, we can achieve a k^c approximation factor using $O(FT)$ samples [Moi15], and with gap above $O(\log^2 k)/T$ we can get a constant approximation using $O(k \log^c(FT/\delta))$ samples and time [PS15].

Having a dependence on the frequency gap is natural. If two frequencies are very close together—significantly below $1/T$—then the corresponding complex exponentials will be close on $[0, T]$, and hard to distinguish in the presence of noise. In fact, from a lower bound in [Moi15], below $1/T$ frequency gap one cannot recover the frequencies in the presence of noise as small as $2^{-\Omega(k)}$. The lower bound proceeds by constructing two signals using significantly different frequencies that are exponentially close over $[0, T]$.

But if two signals are so close, do we need to distinguish them? Such a lower bound doesn't apply to the interpolation problem, it just says that you can't solve it by finding the frequencies. Our question becomes: can we benefit from Fourier sparsity in a regime where we can't recover the individual frequencies?

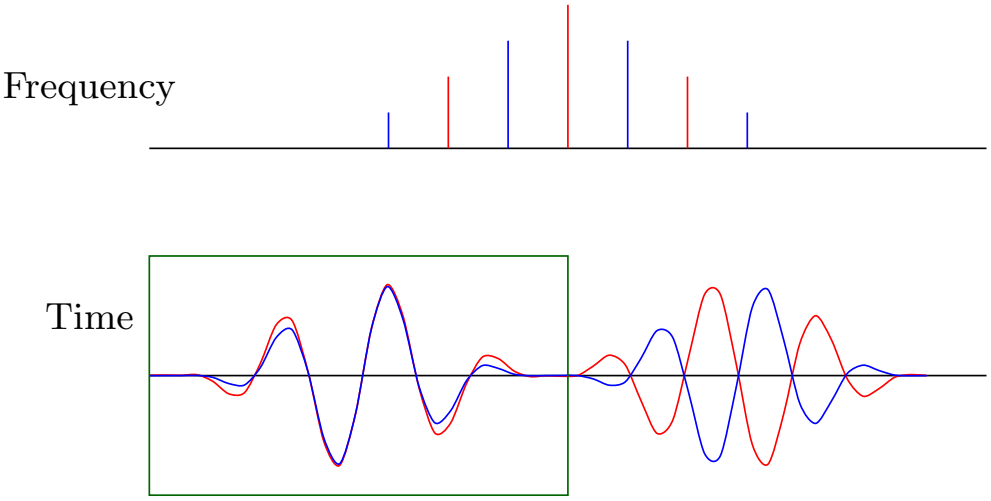


Figure 1.5: Moitra's lower bound [Moi15]. $\forall \epsilon > 0$, if sample duration $T < \frac{(1-\epsilon)}{\eta}$, then $\exists$ two k-Fourier-sparse signals: $x(t)$ and $x'(t)$, each has η frequency gap. To tell them apart, need noise $\leq 2^{-\Omega(\epsilon k)}$. This example only implies that frequency recovery requires frequency gap, but what if we just want to interpolate signal?

We answer in the affirmative, giving an algorithm for the interpolation using $O(\mathrm{poly}(k \log(FT/\delta))$ samples.

This part is based on the following papers:

- Eric Price, Zhao Song

  A Robust Sparse Fourier Transform in the Continuous Setting.

  FOCS 2015 [PS15]

- Xue Chen, Eric Price, Daniel Kane, Zhao Song

  Fourier-sparse Interpolation Without a Frequency Gap.

  FOCS 2016 [CKPS16]

- Vasileios Nakos, Zhao Song, Zhengyu Wang

The Power of (careful) Iterative Loop Analysis in Compressed Sensing.

Manuscript 2018 [NSW18]

- Vasileios Nakos, Zhao Song

  Stronger L2/L2 Compressed Sensing; Without Iterating.

  STOC 2019 [NS19]

- Vasileios Nakos, Zhao Song, Zhengyu Wang

  (Nearly) Sample-Optimal Sparse Fourier Transform in Any Dimension; RIPless and Filterless.

  FOCS 2019 [NSW19a]

1.5 Factorization

Low rank approximation is arguably one of the most well-studied problems in randomized numerical linear algebra, with diverse applications to clustering [DFK+04, FSS13, LBKW14, CEM+15], data mining [AFK+01], distance matrix completion [Cha12], information retrieval [PRTV00], learning mixtures of distributions [AM05, KSV08], recommendation systems [DKR02], and web search [AFKM01, Kle99]. In practice one often has a low rank matrix which has been corrupted with noise of bounded norm, and low rank approximation allows one to approximately recover the original matrix. Low rank approximation may also help explain a dataset, revealing low dimensional structure in high dimensional data. Given a low rank approximation, one can store a matrix and compute a matrix-vector product much more efficiently by storing the corresponding factorization. It can also be used as a preprocessing step in applications, that is, by first projecting data onto a lower-dimensional

subspace one preserves important properties of the input, but can now run subsequent algorithms in the lower-dimensional space. For example, it has been proposed to reduce the data dimension in Non-Negative Matrix Factorization [LS00] (more on this below), and Latent Dirichlet Allocation (LDA) [BNJ01].

The basic low rank approximation problem is: given an $n \times n$ matrix A, find a matrix $\widehat{A}$ of rank at most k for which $\|A - \widehat{A}\|_F$ is minimized, where for a matrix B, $\|B\|_F = \left(\sum_{i,j} B_{i,j}^2\right)^{1/2}$ is its Frobenius norm. This formulation intuitively corresponds to the matrix $\widehat{A}$ capturing as much of the variance of A as possible. It is well-known that the optimal solution is given by A_k, which if $U\Sigma V^\top$ is the singular value decomposition (SVD) of A, where U and V are orthogonal matrices and Σ is a non-negative diagonal matrix with $\Sigma_{1,1} \geq \Sigma_{2,2} \geq \cdots \geq \Sigma_{n,n}$, then $A_k = U\Sigma_k V^\top$, where Σ_k agrees with Σ on its first k diagonal entries and is 0 otherwise. Although the SVD is computable in polynomial time, it is often acceptable to output a matrix $\widehat{A}$ for which $\|A - \widehat{A}\|_F \leq (1+\epsilon)\|A - A_k\|_F$ with high probability. In the latter case, much more efficient algorithms are known, and it is possible to compute such an $\widehat{A}$ in $\mathrm{nnz}(A) + n \cdot \mathrm{poly}(k/\epsilon)$ time, where $\mathrm{nnz}(A)$ denotes the number of non-zero entries of A [CW13, MM13, NN13a]. We note that for typical applications k and $1/\epsilon$ are assumed to be much smaller than n, e.g., in [Har14] they are treated as absolute constants.

Weighted Despite the large body of work on low rank approximation, the *weighted* case is not well understood. In this case one is given an $n \times n$ matrix A and an $n \times n$ matrix W

with $W_{i,j} \geq 0$, and one seeks to solve:

$$\min_{\text{rank-}k \text{ matrices } \widehat{A}} \|W \circ (A - \widehat{A})\|_F^2 = \min_{\text{rank-}k \text{ matrices } \widehat{A}} \sum_{i,j} W_{i,j}^2 (A_{i,j} - \widehat{A}_{i,j})^2.$$

The classical low rank approximation is a special case in which $W_{i,j} = 1$ for all i and j. However, in general there may not be a good reason to weight all elements of the approximation error $A - \widehat{A}$ equally, especially if one is given prior knowledge about the distribution of the errors. For example, suppose the columns of A each come from a low-dimensional subspace but one of the columns is then shifted by a fixed large vector so that its mean is different. One may first want to recenter the data by subtracting off the mean from each of the columns. While this is possible without weighted low rank approximation, suppose instead that each of the columns of A comes from a perturbation of columns in a low dimensional subspace but one of the columns has a much larger variance. Then if all weights were equal, it would be enough for $\widehat{A}$ to fit this one single large variance column, which fails to capture the entire low-dimensional subspace. One way of fixing this is to reweight each entry of A by the inverse of its variance. This is a common technique used in gene expression analysis, where the error model for microarray measurements provides entry-specific noise estimates, or when entries of A represent aggregates of many samples such as in word co-occurrence matrices and non-uniform weights are needed to appropriately capture any differences in the sample sizes; see [SJ03] for a discussion, and also the Wikipedia entry on weighted low rank approximation for a brief introduction.

While the extension of low rank approximation to the weighted case goes back to work of Young in 1940 [You40], its complexity is not well-understood, partly because the weighted case does not admit a solution via the SVD and may have many local minima [SJ03]. Early

work by Shpak [Shp90] looked at gradient-based approaches while Lu et al. [LPW97, LA03] looked at alternating minimization methods. These were significantly sped up in practice by the work of Srebro and Jaakkola [SJ03], with success in various applications such as color image restoration [MES08], though there are no provable time bounds and in the worst case the running times could be exponential or worse. In fact, weighted low rank approximation is known to be NP-hard to approximate up to a $(1 \pm 1/\operatorname{poly}(n))$ factor [GG11]. We note that this also follows from the fact that matrix completion, arguably one of the most important special cases of weighted low rank approximation in which case all weights are 0 or 1, which we discuss more below, is also known to be NP-hard [Pee96, HMRW14]. Typically, though, assumptions such as incoherence and randomly sampled entries allow one to circumvent this hardness [CR09, LLR16]. There is some debate as to whether these assumptions are valid, for instance in [SW15] an argument is made why randomly missing entries may not hold for real-world datasets.

Many natural questions are left open from previous work. In particular one question as we see it is the following:

For which weight matrices W is the problem tractable? More generally, is it possible to identify a natural parameter of W and to obtain parameterized complexity bounds in terms of that parameter?

Entry-wise L1 norm Two well-studied problems in numerical linear algebra are regression and low rank approximation. In regression, one is given an $n \times d$ matrix A, and an $n \times 1$ vector b, and one seeks an $x \in \mathbb{R}^d$ which minimizes $\|Ax - b\|$ under some norm. For example, for least squares regression one minimizes $\|Ax - b\|_2$. In low rank approximation, one is

given an $n \times d$ matrix A, and one seeks a rank-k matrix $\widehat{A}$ which minimizes $\|A - \widehat{A}\|$ under some norm. For example, in Frobenius norm low rank approximation, one minimizes $\|A - \widehat{A}\|_F = \left(\sum_{i,j}(A_{i,j} - \widehat{A}_{i,j})^2\right)^{1/2}$. Algorithms for regression are often used as subroutines for low rank approximation. Indeed, one of the main insights of [DMM06c, DMM06b, Sar06, DMM08, CW09] was to use results for generalized least squares regression for Frobenius norm low rank approximation. Algorithms for ℓ_1-regression, in which one minimizes $\|Ax - b\|_1 = \sum_i |(Ax)_i - b_i|$, were also used [BD13, SW11] to fit a set of points to a hyperplane, which is a special case of entrywise ℓ_1-low rank approximation, the more general problem being to find a rank-k matrix $\widehat{A}$ minimizing $\sum_{i,j} |A_{i,j} - \widehat{A}_{i,j}|$.

Randomization and approximation were introduced to significantly speed up algorithms for these problems, resulting in algorithms achieving relative error approximation with high probability. Such algorithms are based on sketching and sampling techniques; we refer to [Woo14b] for a survey. For least squares regression, a sequence of work [Sar06, CW13, MM13, NN13a, LMP13, BDN15, Coh16a] shows how to achieve algorithms running in $\text{nnz}(A) + \text{poly}(d)$ time. For Frobenius norm low rank approximation, using the advances for regression this resulted in $\text{nnz}(A) + (n + d)\,\text{poly}(k)$ time algorithms. For ℓ_1-regression, sketching and sampling-based methods [Cla05, SW11, CDMI+13, CW13, MM13, LMP13, WZ13, CW15b, CP15] led to an $\text{nnz}(A) + \text{poly}(d)$ time algorithm.

Just like Frobenius norm low rank approximation is the analogue of least squares regression, entrywise ℓ_1-low rank approximation is the analogue of ℓ_1-regression. Despite this analogy, *no non-trivial upper bounds with provable guarantees* are known for ℓ_1-low rank approximation. Unlike Frobenius norm low rank approximation, which can be solved exactly using the singular value decomposition, no such algorithm or closed-form solution is known

for ℓ_1-low rank approximation. Moreover, the problem was recently shown to be NP-hard [GV15]. A major open question is whether there exist approximation algorithms, sketching-based or otherwise, for ℓ_1-low rank approximation. Indeed, the question of obtaining betters algorithms was posed in section 6 of [GV15], in [Exc13], and as the second part of open question 2 in [Woo14b], among other places. The earlier question of NP-hardness was posed in Section 1.4 of [KV09], for which the question of obtaining approximation algorithms is a natural followup. The goal of our work is to answer this question.

We now formally define the ℓ_1-low rank approximation problem: we are given an $n \times d$ matrix A and approximation factor $\alpha \geq 1$, and we would like, with large constant probability, to output a rank-k matrix $\widehat{A}$ for which

$$\|A - \widehat{A}\|_1 \leq \alpha \cdot \min_{\text{rank-}k \text{ matrices } A'} \|A - A'\|_1,$$

where for an $n \times d$ matrix C, we let $\|C\|_1 = \sum_{i=1}^{n} \sum_{j=1}^{d} |C_{i,j}|$. This notion of low rank approximation has been proposed as a more robust alternative to Frobenius norm low rank approximation [KK03, KK05, KLC+15, Kwa08, ZLS+12, BJ12, BD13, BDB13, MXZZ13, MKP13, MKP14, MKCP16, PK16], and is sometimes referred to as ℓ_1-matrix factorization or robust PCA. ℓ_1-low rank approximation gives improved results over Frobenius norm low rank approximation since outliers are less exaggerated, as one does not square their contribution in the objective. The outlier values are often erroneous values that are far away from the nominal data, appear only a few times in the data matrix, and would not appear again under normal system operation. These works also argue ℓ_1-low rank approximation can better handle missing data, is appropriate in noise models for which the noise is not Gaussian, e.g., it produces the maximum likelihood estimator for Laplacian noise [Gao08, KAC+08, VT01], and can be used in image processing to prevent image occlusion [YZD12].

To see that ℓ_1-low rank approximation and Frobenius norm low rank approximation can give very different results, consider the $n \times n$ matrix $A = \left[\begin{smallmatrix} n & 0 \\ 0 & B \end{smallmatrix}\right]$, where B is *any* $(n-1) \times (n-1)$ matrix with $\|B\|_F < n$. The best rank-1 approximation with Frobenius norm error is given by $\widehat{A} = n \cdot e_1 e_1^\top$, where e_1 is the first standard unit vector. Here $\widehat{A}$ ignores all but the first row and column of A, which may be undesirable in the case that this row and column represent an outlier. Note $\|A - \widehat{A}\|_1 = \|B\|_1$. If, for example, B is the all 1s matrix, then $\widehat{A} = [0, 0; 0, B]$ is a rank-1 approximation for which $\|A - \widehat{A}\|_1 = n$, and therefore this solution is a much better solution to the ℓ_1-low rank approximation problem than $n \cdot e_1 e_1^\top$, for which $\|A - n \cdot e_1 e_1^\top\|_1 = (n-1)^2$.

Despite the advantages of ℓ_1-low rank approximation, its main disadvantage is its computationally intractability. It is not rotationally invariant and most tools for Frobenius low rank approximation do not apply. To the best of our knowledge, all previous works only provide heuristics. Using that for an $n \times d$ matrix C, $\|C\|_F \leq \|C\|_1 \leq \sqrt{nd}\|C\|_F$, a Frobenius norm low rank approximation gives a $\sqrt{nd}$ approximation for ℓ_1-low rank approximation. A bit better is to use algorithms for low rank approximation with respect to the sum of distances, i.e., to find a rank-k matrix $\widehat{A}$ minimizing $\|A - \widehat{A}\|_{1,2}$, where for an $n \times d$ matrix C, $\|C\|_{1,2} = \sum_{i=1}^n \|C_i\|_2$, where C_i is the i-th row of C. A sequence of work [DV07, FMSW10, FL11, SV12a, CW15a] shows how to obtain an $O(1)$-approximation to this problem in $\mathrm{nnz}(A) + (n+d)\,\mathrm{poly}(k) + \exp(k)$ time, and using that $\|C\|_{1,2} \leq \|C\|_1 \leq \sqrt{d}\|C\|_{1,2}$ results in an $O(\sqrt{d})$-approximation.

Zero-one Law To understand the role of the Frobenius norm in the algorithms above, we recall a standard motivation for this error measure. Suppose one has n data points in

a k-dimensional subspace of $\mathbb{R}^d$, where $k \ll d$. We can write these points as the rows of an $n \times d$ matrix A^* which has rank k. The matrix A^* is often called the *ground truth matrix*. In a number of settings, due to measurement noise or other kinds of noise, we only observe the matrix $A = A^* + \Delta$, where each entry of the *noise matrix* $\Delta \in \mathbb{R}^{n \times n}$ is an i.i.d. random variable from a certain mean-zero noise distribution $\mathcal{D}$. One method for approximately recovering A^* from A is maximum likelihood estimation. Here one tries to find a matrix B maximizing the log-likelihood: $\max_{\text{rank-}k\ B} \sum_{i,j} \log p(A_{i,j} - B_{i,j})$, where $p(\cdot)$ is the probability density function of the underlying noise distribution $\mathcal{D}$. For example, when the noise distribution is Gaussian with mean zero and variance σ^2, denoted by $N(0, \sigma^2)$, then the optimization problem is $\max_{\text{rank-}k\ B} \sum_{i,j} \left(\log(1/\sqrt{2\pi\sigma^2}) - (A_{i,j} - B_{i,j})^2/(2\sigma^2) \right)$, which is equivalent to solving the Frobenius norm loss low rank approximation problem defined above.

The Frobenius norm loss, while having nice statistical properties for Gaussian noise, is well-known to be sensitive to outliers. Applying the same maximum likelihood framework above to other kinds of noise distributions results in minimizing other kinds of loss functions. In general, if the density function of the underlying noise $\mathcal{D}$ is $p(z) = c \cdot e^{-g(z)}$, where c is a normalization constant, then the maximum likelihood estimation problem for this noise distribution becomes the following generalized entry-wise loss low rank approximation problem: $\min_{\text{rank-}k\ B} \sum_{i,j} g(A_{i,j} - B_{i,j}) = \min_{\text{rank-}k\ B} \|A - B\|_g$, which is a central topic of recent work on *generalized low-rank models* [UHZ+16]. For example, when the noise is Laplacian, the entrywise ℓ_1 loss is the maximum likelihood estimation, which is also robust to sparse outliers. A natural setting is when the noise is a mixture of small Gaussian noise and sparse outliers; this noise distribution is referred to as the *Huber density*. In this case

the Huber loss function gives the maximum likelihood estimate [UHZ+16], where the Huber function [Hub64] is defined to be: $g(x) = x^2/(2\tau)$ if $|x| < \tau/2$, and $g(x) = |x| - \tau/2$ if $|x| \geq \tau$. Another nice property of the Huber error measure is that it is differentiable everywhere, unlike the ℓ_1-norm, yet still enjoys the robustness properties as one moves away from the origin, making it less sensitive to outliers than the ℓ_2-norm. There are many other kinds of loss functions, known as M-estimators [Zha97], which are widely used as loss functions in robust statistics [HRRS11].

Although several specific cases have been studied, such as entry-wise ℓ_p loss [CLMW11, SWZ17, CGK+17a, BKW17, BBB+19a], weighted entry-wise ℓ_2 loss [RSW16], and cascaded $\ell_p(\ell_2)$ loss [DVTV09, CW15a], the landscape of general entry-wise loss functions remains elusive. There are no results known for any loss function which is not scale-invariant, much less any kind of characterization of which loss functions admit efficient algorithms. This is despite the importance of these loss functions; we refer the reader to [UHZ+16] for a survey of generalized low rank models. This motivates the main question in our work:

Question 1.5.1 (General Loss Functions)**.** *For a given approximation factor $\alpha > 1$, which functions g allow for efficient low-rank approximation algorithms? Formally, given an $n \times d$ matrix A, can we find a* rank-k *matrix B for which $\|A - B\|_g \leq \alpha \min_{\text{rank}-k\ B'} \|A - B'\|_g$, where for a matrix C, $\|C\|_g = \sum_{i \in [n], j \in [d]} g(C_{i,j})$? What if we also allow B to have rank* $\text{poly}(k \log n)$*?*

For Question 1.5.1, one has $g(x) = |x|^p$ for p-norms, and note the Huber loss function also fits into this framework. Allowing B to have slightly larger rank than k, namely, $\text{poly}(k \log n)$, is often sufficient for applications as it still allows for the space savings and

computational gains outlined above. These are referred to as bicriteria approximations and are the focus of our work.

Tensor Tensors are often more useful than matrices for capturing higher order relations in data. Computing low rank factorizations of approximations of tensors is the primary task of interest in a number of applications, such as in psychology[Kro83], chemometrics [Paa00, SBG04], neuroscience [AAB+07, KB09, CLK+15], computational biology [CV15, SC15], natural language processing [CYYM14, LZBJ14, LZMB15, BNR+15], computer vision [VT02, WA03, SH05, HPS05, HD08, AFdLGTL09, PLY10, LFC+16, CLZ17], computer graphics [VT04, WWS+05, Vas09], security [AÇKY05, ACY06, KB06], cryptography [FS99, Sch12, KYFD15, SHW+16] data mining [KS08, RST10, KABO10, Mør11], machine learning applications such as learning hidden Markov models, reinforcement learning, community detection, multi-armed bandit, ranking models, neural network, Gaussian mixture models and Latent Dirichlet allocation [MR05, AFH+12, HK13, ALB13, ABSV14, AGH+14, AGHK14, BCV14, JO14a, GHK15, PBLJ15, JSA15, ALA16, AGMR16, ZSJ+17], programming languages [RTP16], signal processing [Wes94, DLDM98, Com09, CMDL+15], and other applications [YCS11, LMWY13, OS14, ZCZJ14, STLS14, YCS16, RNSS16].

Despite the success for matrices, the situation for order-q tensors for $q > 2$ is much less understood. There are a number of works based on alternating minimization [CC70, Har70, FMPS13, FT15, ZG01, BS15] gradient descent or Newton methods [ES09, ZG01], methods based on the Higher-order SVD (HOSVD) [LMV00a] which provably incur $\Omega(\sqrt{n})$-inapproximability for Frobenius norm error [LMV00b], the power method or orthogonal iteration method [LMV00b], additive error guarantees in terms of the flattened (unfolded)

tensor rather than the original tensor [MMD08], tensor trains [Ose11], the tree Tucker decomposition [OT09], or methods specialized to orthogonal tensors [KM11, AGH+14, MHG15, WTSA15, WA16, SWZ16]. There are also a number of works on the problem of tensor completion, that is, recovering a low rank tensor from missing entries [WM01, AKDM10, TSHK11, LMWY13, MHWG14, JO14b, BM16]. There is also another line of work using the sum of squares (SOS) technique to study tensor problems [BKS15a, GM15, HSS15, HSSS16, MSS16, PS17, SS17], other recent work on tensor PCA [All12b, All12a, RM14, JMZ15, ADGM16, ZX17], and work applying smoothed analysis to tensor decomposition [BCMV14]. Several previous works also consider more robust norms than the Frobenius norm for tensors, e.g., the R_1 norm (ℓ_1-ℓ_2-ℓ_2 norm in our work) [HD08], ℓ_1-PCA [PLY10], entry-wise ℓ_1 regularization [GGH14], M-estimator loss [YFS16], weighted approximation [Paa97, TK11, LRHG13], tensor-CUR [OST08, MMD08, CC10, FMMN11, FT15], or robust tensor PCA [GQ14, LFC+16, CLZ17].

Some of the above works, such as ones based on the tensor power method or alternating minimization, require incoherence or orthogonality assumptions. Others, such as those based on the simultaneous SVD, require an assumption on the minimum singular value. See the monograph of Moitra [Moi14a] for further discussion. Unlike the situation for matrices, there is no work for tensors that is able to achieve the following natural relative error guarantee: given a q-th order tensor $A \in \mathbb{R}^{n^{\otimes q}}$ and an arbitrary accuracy parameter $\epsilon > 0$, output a rank-k tensor B for which

$$\|A - B\|_F^2 \leq (1+\epsilon)\,\mathrm{OPT}, \tag{1.8}$$

where $\mathrm{OPT} = \inf_{\text{rank-}k\ B'} \|A - B'\|_F^2$, and where recall the rank of a tensor B is the minimal integer k for which B can be expressed as $\sum_{i=1}^{k} U_1^i \otimes U_2^i \otimes \cdots \otimes U_q^i$. General speaking, a q-th

order tensor can have rank at most n^{q-1}. A third order tensor, for example, has rank which is an integer in $\{0, 1, 2, \ldots, n^2\}$.

For notational simplicity, we will start by assuming third order tensors with all dimensions of equal size, but we extend all of our main theorems below to tensors of any constant order $q > 3$ and dimensions of different sizes.

The first caveat regarding (1.8) for tensors is that an optimal rank-k solution may not even exist! This is a well-known problem for tensors (see, e.g., [KHL89, Paa00, KDS08, Ste06, Ste08] and more details in section 4 of [DSL08]), for which for any rank-k tensor B, there always exists another rank-k tensor B' for which $\|A - B'\|_F^2 < \|A - B\|_F^2$. If OPT $= 0$, then in this case for any rank-k tensor B, necessarily $\|A - B\|_F^2 > 0$, and so (1.8) cannot be satisfied. This fact was known to algebraic geometers as early as the 19th century, which they refer to as the fact that the locus of r-th secant planes to a Segre variety may not define a (closed) algebraic variety [DSL08, Lan12]. It is also known as the phenomenon underlying the concept of *border rank*[9][Bin80, Bin86, BCS97, Knu98, Lan06]. In this case it is natural to allow the algorithm to output an arbitrarily small $\gamma > 0$ amount of additive error. Note that unlike several additive error algorithms for matrices, the additive error here can in fact be an arbitrarily small positive function of n. If, however, OPT > 0, then for any $\epsilon > 0$, there exists a rank-k tensor B for which $\|A - B\|_F^2 \leq (1 + \epsilon)$ OPT, and in this case we should still require the algorithm to output a relative-error solution. If an optimal rank-k solution B exists, then as for matrices, it is natural to require the algorithm to output a relative-error solution.

[9] https://en.wikipedia.org/wiki/Tensor_rank_decomposition#Border_rank

Besides the above definitional issue, a central reason that (1.8) has not been achieved is that computing the rank of a third order tensor is well-known to be NP-hard [Hås90, HL13]. Thus, if one had such a polynomial time procedure for solving the problem above, one could determine the rank of A by running the procedure on each $k \in \{0, 1, 2, \ldots, n^2\}$, and check for the first value of k for which $\|A - B\|_F^2 = 0$, thus determining the rank of A. However, it is unclear if approximating the tensor rank is hard. This question will also be answered in this work.

This part is based on the following papers:

- Ilya Razenshteyn, Zhao Song, David Woodruff

  Weighted low rank approximations with provable guarantees.

  STOC 2016 [RSW16]

- Zhao Song, David Woodruff, Peilin Zhong

  Low Rank Approximation with Entrywise ℓ_1-Norm Error.

  STOC 2017 [SWZ17]

- Zhao Song, David Woodruff, Peilin Zhong

  Towards a Zero-One Law for Entrywise Low Rank Approximation.

  Manuscript 2018 [SWZ18]

- Zhao Song, David Woodruff, Peilin Zhong

  Relative Error Tensor Low Rank Approximation.

  SODA 2019 [SWZ19b]

- Zhao Song, David Woodruff, Huan Zhang

  Sublinear Time Orthogonal Tensor Decomposition.

  NeurIPS 2016 [SWZ16]

1.6 More Algorithms

Regression Sketching has emerged as a powerful technique for speeding up problems in numerical linear algebra, such as regression. In the overconstrained regression problem, one is given an $n \times d$ matrix A, with $n \gg d$, as well as an $n \times 1$ vector b, and one wants to find a vector $\widehat{x}$ so as to minimize the residual error $\|Ax - b\|_2$. Using the sketch and solve paradigm, one first computes $S \cdot A$ and $S \cdot b$ for a randomly chosen matrix S, then outputs $x' = (SA)^\dagger Sb$ so as to minimize $\|SAx' - Sb\|_2$.

The sketch-and-solve paradigm gives a bound on $\|x' - x^*\|_2$ when A is well-conditioned. One of result cares about $\|x' - x^*\|_\infty$, we call it ℓ_∞ regression problem. We also studied regression problem under general norms (i.e. symmetric norm regression $\min \|Ax - b\|_{\text{symmetric norm}}$) and general structure (i.e. tensor regression $\min_x \|(A_1 \otimes A_2)x - b\|$).

Clustering We consider the k-means clustering problem in the dynamic streaming setting, where points from a discrete Euclidean space $\{1, 2, \ldots, \Delta\}^d$ can be dynamically inserted to or deleted from the dataset. For this problem, we provide a one-pass coreset construction algorithm using space $\widetilde{O}(k \cdot \mathrm{poly}(d, \log \Delta))$, where k is the target number of centers. To our knowledge, this is the first dynamic geometric data stream algorithm for k-means using space polynomial in dimension and nearly optimal (linear) in k.

LCS Binary String Given a pair of n-character strings, the problems of computing their Longest Common Subsequence and Edit Distance have been extensively studied for decades. For exact algorithms, LCS and Edit Distance (with character insertions and deletions) are equivalent; the state of the art running time is (almost) quadratic in n, and this is tight under plausible fine-grained complexity assumptions. But for approximation algorithms the picture is different: there is a long line of works with improved approximation factors for Edit Distance, but for LCS (with binary strings) only a trivial 1/2-approximation was known. In this work we give a reduction from approximate LCS to approximate Edit Distance, yielding the first efficient $(1/2 + \epsilon)$-approximation algorithm for LCS for some constant $\epsilon > 0$.

LCS Longest common subsequence (LCS) is a classic and central problem in combinatorial optimization. While LCS admits a quadratic time solution, recent evidence suggests that solving the problem may be impossible in truly subquadratic time. A special case of LCS wherein each character appears at most once in every string is equivalent to the longest increasing subsequence problem (LIS) which can be solved in quasilinear time. In this work, we present novel algorithms for approximating LCS in truly subquadratic time and LIS in truly sublinear time. Our approximation factors depend on the ratio of the optimal solution size over the input size. We denote this ratio by λ and obtain the following results for LCS and LIS without any prior knowledge of λ.

- A truly subquadratic time algorithm for LCS with approximation factor $O(\lambda^3)$.
- A truly sublinear time algorithm for LIS with approximation factor $O(\lambda^3)$.

Triangle inequality was recently used by Boroujeni *et al.* [BEG+18] and Chakraborty

et al. [CDG+18] to present new approximation algorithms for edit distance. Our techniques for LCS extend the notion of triangle inequality to non-metric settings.

Map-Reduce Many modern parallel systems, such as MapReduce, Hadoop and Spark, can be modeled well by the MPC model. The MPC model captures well coarse-grained computation on large data — data is distributed to processors, each of which has a sublinear (in the input data) amount of memory and we alternate between rounds of computation and rounds of communication, where each machine can communicate an amount of data as large as the size of its memory. This model is stronger than the classical PRAM model, and it is an intriguing question to design algorithms whose running time is smaller than in the PRAM model.

One fundamental graph problem is connectivity. On an undirected graph with n nodes and m edges, $O(\log n)$ round connectivity algorithms have been known for over 35 years. However, no algorithms with better complexity bounds were known. In this work, we give **fully scalable, faster algorithms for the connectivity problem**, by parameterizing the time complexity as a function of the *diameter* of the graph. Our main result is a $O(\log D \log \log_{m/n} n)$ time connectivity algorithm for diameter-D graphs, using $\Theta(m)$ total memory. If our algorithm can use more memory, it can terminate in fewer rounds, and there is no lower bound on the memory per processor.

We extend our results to related graph problems such as spanning forest, finding a DFS sequence, exact/approximate minimum spanning forest, and bottleneck spanning forest. We also show that achieving similar bounds for reachability in *directed graphs* would imply faster boolean matrix multiplication algorithms.

We introduce several new algorithmic ideas. We describe a general technique called *double exponential speed problem size reduction* which roughly means that if we can use total memory N to reduce a problem from size n to n/k, for $k = (N/n)^{\Theta(1)}$ in one phase, then we can solve the problem in $O(\log \log_{N/n} n)$ phases. In order to achieve this fast reduction for graph connectivity, we use a multistep algorithm. One key step is a carefully constructed truncated broadcasting scheme where each node broadcasts neighbor sets to its neighbors in a way that limits the size of the resulting neighbor sets. Another key step is *random leader contraction*, where we choose a smaller set of leaders than many previous works do.

Part I

Optimization

Chapter 2

Linear Programs

This chapter shows how to solve linear programs of the form $\min_{Ax=b,x\geq 0} c^\top x$ with n variables in time

$$O^*((n^\omega + n^{2.5-\alpha/2} + n^{2+1/6}) \log(n/\delta))$$

where ω is the exponent of matrix multiplication, α is the dual exponent of matrix multiplication, and δ is the relative accuracy. For the current value of $\omega \sim 2.37$ and $\alpha \sim 0.31$, our algorithm takes $O^*(n^\omega \log(n/\delta))$ time. When $\omega = 2$, our algorithm takes $O^*(n^{2+1/6} \log(n/\delta))$ time.

Our algorithm utilizes several new concepts that we believe may be of independent interest:

- We define a stochastic central path method.
- We show how to maintain a projection matrix $\sqrt{W}A^\top (AWA^\top)^{-1} A\sqrt{W}$ in sub-quadratic time under ℓ_2 multiplicative changes in the diagonal matrix W.

2.1 Introduction

Linear programming is one of the key problems in computer science. In both theory and practice, many problems can be reformulated as linear programs to take advantage of fast algorithms. For an arbitrary linear program $\min_{Ax=b,x\geq 0} c^\top x$ with n variables and d constraints[1], the fastest algorithm takes $O^*(\sqrt{d}\cdot \mathrm{nnz}(A)+d^{2.5})$[2] where $\mathrm{nnz}(A)$ is the number of non-zeros in A [LS14, LS15].

For the generic case $d=\Omega(n)$ we focus in this paper, the current fastest runtime is dominated by $O^*(n^{2.5})$. This runtime has not been improved since the result by Vaidya on 1989 [Vai87, Vai89b]. The $n^{2.5}$ bound originated from two factors: the cost per iteration n^2 and the number of iterations $\sqrt{n}$. The n^2 cost per iteration looks optimal because this is the cost to compute Ax for a dense A. Therefore, many efforts [Kar84, Ren88, NN89, Vai89a, LS14] have been focused on decreasing the number of iterations while maintaining the cost per iteration. As for many important linear programs (and convex programs), the number of iterations has been decreased, including maximum flow [Mad13, Mad16], minimum cost flow [CMSV17], geometric median [CLM+16], matrix scaling and balancing [CMTV17], and ℓ_p regression [BCLL18]. Unfortunately, beating $\sqrt{n}$ iterations (or $\sqrt{d}$ when $d\ll n$) for the general case remains one of the biggest open problems in optimization.

Avoiding this open problem, we develop a stochastic central path method that has a runtime of $O^*(n^\omega+n^{2.5-\alpha/2}+n^{2+1/6})$, where ω is the exponent of matrix multiplication

[1] Throughout this paper, we assume there is no redundant constraints and hence $n\geq d$. Note that papers in different communities uses different symbols to denote the number of variables and constraints in a linear program.

[2] We use O^* to hide $n^{o(1)}$ and $\log^{O(1)}(1/\delta)$ factors and $\widetilde{O}$ to hide $\log^{O(1)}(n/\delta)$ factors.

and α is the dual exponent of matrix multiplication[3]. For the current value of $\omega \sim 2.38$ and $\alpha \sim 0.31$, the runtime is simply $O^*(n^\omega)$. This achieves the natural barrier for solving linear programs because linear system is a special case of linear program and that the currently fastest way to solve general linear systems involves matrix multiplication. Despite the exact approach used in [CW87, Wil12, DS13, LG14] cannot give a bound on ω better than 2.3078 [AFLG15] and all known approaches cannot achieve the bound $\omega = 2$ [AW18b], it is still possible that $\omega = 2.01$ using all known approaches. Therefore, we believe improving the additive $2 + 1/6$ term remains an interesting open problem.

Our method is a stochastic version of the short step central path method. This short step method takes $O^*(\sqrt{n})$ steps and each step decreases $x_i s_i$ by a $1 - 1/\sqrt{n}$ factor for all i where s is the dual variable [Ren88] (See the definition of s in (2.1)). This results in $O^*(\sqrt{n}) \times n = O^*(n^{1.5})$ coordinate updates. Our method takes the same number of step but only updates $\widetilde{O}(\sqrt{n})$ coordinates each step. Therefore, we only update $O^*(n)$ coordinates in total, which is nearly optimal.

Our framework is efficient enough to take a much smaller step while maintaining the same running time. For the current value of $\omega \sim 2.38$, we show how to obtain the same runtime of $O^*(n^\omega)$ by taking $O^*(n)$ steps and $\widetilde{O}(1)$ coordinates update per steps. This is because the complexity of each step decreases proportionally when the step size decreases. Beyond the cost per iteration, we remark that our algorithm is one of the very few central path algorithms [PRT02, Mad13, Mad16] that does not maintain $x_i s_i$ close to some ideal vector in ℓ_2 norm. We are hopeful that our stochastic method and our proof will be useful

[3]The dual exponent of matrix multiplication α is the supremum among all $a \geq 0$ such that it takes $n^{2+o(1)}$ time to multiply an $n \times n$ matrix by an $n \times n^a$ matrix.

for future research on interior point methods. In particular, it would be interesting to see how this can be combined with techniques in [Cla95, LS14] to get a faster algorithm for linear programs with $d \ll n$.

Besides the applications to linear programs, some of our techniques are probably useful for studying other important problems in convex optimization. In particular, our framework should be naturally extendable to a larger class of convex programs.

2.1.1 Related Work

Interior point method has a long history, for more detailed surveys, we refer the readers to [Wri97, Ye97, Ren01, RTV05, Meg12, Ter13]. This paper in part inspired by the use of data-structure in Laplacian solvers [ST04, KMP10, KMP11, CKM+11, KOSZ13, CKM+14, KLP+16, KS16, CKK+18, KPSZ18], in particular the cycle update in [KOSZ13].

2.2 Results and Techniques

Theorem 2.2.1 (Main result)**.** *Given a linear program* $\min_{Ax=b,x\geq 0} c^\top x$ *with no redundant constraints. Assume that the polytope has diameter* R *in* ℓ_1 *norm, namely, for any* $x \geq 0$ *with* $Ax = b$*, we have* $\|x\|_1 \leq R$*.*

Then, for any $0 < \delta \leq 1$, $\textsc{Main}(A, b, c, \delta)$ *outputs* $x \geq 0$ *such that*

$$c^\top x \leq \min_{Ax=b,x\geq 0} c^\top x + \delta \cdot \|c\|_\infty R \quad \text{and} \quad \|Ax - b\|_1 \leq \delta \cdot \left(R \sum_{i,j} |A_{i,j}| + \|b\|_1 \right)$$

in expected time

$$\left(n^{\omega+o(1)} + n^{2.5-\alpha/2+o(1)} + n^{2+1/6+o(1)}\right) \cdot \log(\frac{n}{\delta})$$

where ω *is the exponent of matrix multiplication,* α *is the dual exponent of matrix multiplication.*

For the current value of $\omega \sim 2.38$ *and* $\alpha \sim 0.31$*, the expected time is simply* $n^{\omega+o(1)} \log(\frac{n}{\delta})$.

Remark 2.2.1. See [Ren88] and [LS13, Sec E, F] on the discussion on converting an approximation solution to an exact solution. For integral A, b, c, it suffices to pick $\delta = 2^{-O(L)}$ to get an exact solution where $L = \log(1 + d_{\max} + \|c\|_\infty + \|b\|_\infty)$ is the bit complexity and $d_{\max}$ is the largest absolute value of the determinant of a square sub-matrix of A. For many combinatorial problems, $L = O(\log(n + \|b\|_\infty + \|c\|_\infty))$.

In this paper, we assume all floating point calculations are done exactly for simplicity. In general, the algorithm can be carried out with $O(L)$ bits of accuracy. This is necessary because each coordinate in the solution could require as much as $\Omega(L)$ bits to represent. See [Ren88] for some discussions on the numerical stability of the interior point methods.

If $T(n)$ is the current cost of matrix multiplication and inversion with $T(n) \sim n^{2.38}$, our runtime is simply $O(T(n) \log n \log(\frac{n}{\delta}))$. The $\log(\frac{n}{\delta})$ comes from iteration count and the $\log n$ factor comes from the doubling trick ($|y_{\pi(1.5r)}| \geq (1 - 1/\log n)|y_{\pi(r)}|$) in the projection maintenance section. We left the problem of obtaining $O(T(n) \log(\frac{n}{\delta}))$ as an open problem.

Finally, we note that our runtime holds for any square and rectangular matrix multiplication algorithm as long as $\omega \leq 3 - \alpha$ (See Lemma 2.6.3) For example, Strassen algorithm together with a simple rectangular multiplication algorithm gives a runtime of roughly $n^{2.807}$.

2.2.1 Central Path Method

Our algorithm relies on two new ingredients: stochastic central path and projection maintenance. The central path method consider the linear programs

$$\min_{Ax=b, x\geq 0} c^\top x \quad \text{(primal)} \quad \text{and} \quad \max_{A^\top y \leq c} b^\top y \quad \text{(dual)}$$

with $A \in \mathbb{R}^{d\times n}$. Any solution of the linear program satisfies the following optimality conditions:

$$\begin{aligned} x_i s_i &= 0 \text{ for all } i, \\ Ax &= b, \\ A^\top y + s &= c, \\ x_i, s_i &\geq 0 \text{ for all } i. \end{aligned} \tag{2.1}$$

We call (x, s, y) feasible if it satisfies the last three equations above. For any feasible (x, s, y), the duality gap is $\sum_i x_i s_i$. The central path method find a solution of the linear program by following the central path which uniformly decrease the duality gap. The central path $(x_t, s_t, y_t) \in \mathbb{R}^{n+n+d}$ is a path parameterized by t and defined by

$$\begin{aligned} x_{t,i} s_{t,i} &= t \text{ for all } i, \\ Ax_t &= b, \\ A^\top y_t + s_t &= c, \\ x_{t,i}, s_{t,i} &\geq 0 \text{ for all } i. \end{aligned} \tag{2.2}$$

It is known [YTM94] how to transform linear programs by adding $O(n)$ many variables and constraints so that:

- The optimal solution remains the same.
- The central path at $t = 1$ is near $(1_n, 1_n, 0_d)$ where 1_n and 0_d are all 1 and all 0 vectors with lengths n and d.
- It is easy to convert an approximate solution of the transformed program to the original one.

For completeness, a theoretical version of such result is included in Lemma 3.8.2. This result shows that it suffices to move gradually (x_1, s_1, y_1) to (x_t, s_t, y_t) for small enough t.

2.2.1.1 Short Step Central Path Method

The short step central path method maintains $x_i s_i = \mu_i$ for some vector μ such that

$$\sum_i (\mu_i - t)^2 = O(t^2) \quad \text{for some scalar } t > 0. \tag{2.3}$$

Since the duality gap is $\sum_i \mu_i$, it suffices to find x and s satisfying the above equation with small enough t. There are many variants of central path methods. We will focus on the version that decreases t and takes a step of μ at the same time. The purpose of moving μ is to maintain the invariant (2.3) and the purpose of decreasing t is decrease the duality gap, which is roughly nt. One natural way to maintain the invariant (2.3) is to do a gradient descent step on the energy $\sum_i (\mu_i - t)^2$ defined in (2.3), namely, moving μ to $\mu - h(\mu - t)$ with step size h[4]. Compared to other versions, we only take one step instead of multiple steps to move μ closer to the central path t per update of t.

[4]The classical view of central path method is to take a Newton step on the system (2.2), which turns out to be same as taking a gradient step on the energy defined in (2.3). However, our main algorithm will choose a different energy and this gradient descent view is crucial for designing our algorithm.

More generally, say we want to move from μ to $\mu + \delta_\mu$, we approximate the term $(x + \delta_x)_i(s + \delta_s)_i$ by $x_i s_i + x_i \delta_{s,i} + s_i \delta_{x,i}$ and obtain the following system:

$$\begin{aligned} X\delta_s + S\delta_x &= \delta_\mu, \\ A\delta_x &= 0, \\ A^\top \delta_y + \delta_s &= 0, \end{aligned} \tag{2.4}$$

where $X = \mathrm{diag}(x)$ and $S = \mathrm{diag}(s)$. This equation is the linear approximation of the original goal (moving from μ to $\mu + \delta_\mu$), and that the step is explicitly given by the formula

$$\delta_x = \frac{X}{\sqrt{XS}}(I - P)\frac{1}{\sqrt{XS}}\delta_\mu \text{ and } \delta_s = \frac{S}{\sqrt{XS}}P\frac{1}{\sqrt{XS}}\delta_\mu, \tag{2.5}$$

where $P = \sqrt{\frac{X}{S}}A^\top \left(A\frac{X}{S}A^\top\right)^{-1} A\sqrt{\frac{X}{S}}$ is an orthogonal projection and the formulas $\frac{X}{\sqrt{XS}}, \frac{X}{S}, \cdots$ are the diagonal matrices of the corresponding vectors.

In turns out that one can decrease t by $1 - \frac{1}{\sqrt{n}}$ multiplicative factor every iteration while maintain the invariant (2.3). This requires $\widetilde{O}(\sqrt{n})$ iterations to converge. Combining this with the inverse maintenance technique [Vai87], this gives a total runtime of $n^{2.5}$. More precisely, the algorithm maintains the invariant $\sum_i (\mu_i - t)^2 = O(t^2)$ by making steps bring μ_i closer to t while taking steps to decrease μ_i uniformly. The progress of the whole algorithm is measured by t because the duality gap is bounded by nt.

2.2.1.2 Stochastic Central Path Method

This part discuss how to modify the short step central path to decrease the cost per iteration to roughly $n^{\omega - \frac{1}{2}}$. Since our goal is to implement a central path method in sub-quadratic time per iteration, we even do not have the budget to compute Ax every

iterations. Therefore, instead of maintaining $\left(A\frac{X}{S}A^\top\right)^{-1}$ shown in previous papers, we will study the problem of maintaining a projection matrix $P = \sqrt{\frac{X}{S}}A^\top\left(A\frac{X}{S}A^\top\right)^{-1}A\sqrt{\frac{X}{S}}$ due to the formula of δ_x and δ_s (2.5).

However, even if the projection matrix P is given explicitly for free, it is difficult to multiply the dense projection matrix with a dense vector δ_μ in time $o(n^2)$. To avoid moving along a dense δ_μ, we move along an $O(k)$ sparse direction $\widetilde{\delta}_\mu$ defined by

$$\widetilde{\delta}_{\mu,i} = \begin{cases} \delta_{\mu,i}/p_i, & \text{with probability } p_i \stackrel{\text{def}}{=} k \cdot \left(\frac{\delta_{\mu,i}^2}{\sum_l \delta_{\mu,l}^2} + \frac{1}{n}\right); \\ 0, & \text{else.} \end{cases} \tag{2.6}$$

The sparse direction is defined so that we are moving in the same direction in expectation ($\mathbb{E}[\widetilde{\delta}_{\mu,i}] = \delta_{\mu,i}$) and that the direction has as small variance as possible ($\mathbb{E}[\widetilde{\delta}_{\mu,i}^2] \leq \frac{\sum_i \delta_{\mu,i}^2}{k}$). If the projection matrix is given explicitly, we can apply the projection matrix on $\widetilde{\delta}_\mu$ in time $O(nk)$. This paper picks $k \sim \sqrt{n}$ and the sum of the cost of projection vector multiplications in the whole algorithm is about $nk^2 = n^2$.

During the whole algorithm, we maintain a projection matrix

$$\overline{P} = \sqrt{\frac{\overline{X}}{\overline{S}}}A^\top\left(A\frac{\overline{X}}{\overline{S}}A^\top\right)^{-1}A\sqrt{\frac{\overline{X}}{\overline{S}}}$$

for vectors $\overline{x}$ and $\overline{s}$ such that $\overline{x}_i$ and $\overline{s}_i$ are multiplicative approximations of x_i and s_i respectively for all i. Since we maintain the projection at a nearby point $(\overline{x}, \overline{s})$, our stochastic step $x \leftarrow x + \widetilde{\delta}_x$, $s \leftarrow s + \widetilde{\delta}_s$ and $y \leftarrow y + \widetilde{\delta}_y$ are defined by

$$\begin{aligned} \overline{X}\widetilde{\delta}_s + \overline{S}\widetilde{\delta}_x &= \widetilde{\delta}_\mu, \\ A\widetilde{\delta}_x &= 0, \\ A^\top\widetilde{\delta}_y + \widetilde{\delta}_s &= 0, \end{aligned} \tag{2.7}$$

which is different from (2.4) on both sides of the first equation. Note that this system use $\overline{X}$ and $\overline{S}$ because we have only maintained this projection matrix. The main goal of Section 2.4 is to show $\overline{X} = \Theta(X)$ and $\overline{S} = \Theta(S)$ is good enough for our interior point method. Similar to (2.5), Lemma 2.4.2 shows that

$$\widetilde{\delta}_x = \frac{\overline{X}}{\sqrt{\overline{X}\overline{S}}}(I - \overline{P})\frac{1}{\sqrt{\overline{X}\overline{S}}}\widetilde{\delta}_\mu \text{ and } \widetilde{\delta}_s = \frac{\overline{S}}{\sqrt{\overline{X}\overline{S}}}\overline{P}\frac{1}{\sqrt{\overline{X}\overline{S}}}\widetilde{\delta}_\mu. \tag{2.8}$$

The previously fastest algorithm involves maintaining the matrix inverse $(A\frac{X}{S}A^\top)^{-1}$ using subspace embedding techniques [Sar06, CW13, NN13a] and leverage score sampling [SS11]. In this paper, we maintain the projection directly using lazy update.

The key departure from the central path we present is that we can only maintain

$$0.9t \leq \mu_i = x_i s_i \leq 1.1t \quad \text{for some } t > 0$$

instead of μ close to t in ℓ_2 norm. We will further explain the proof in Section 2.4.1.

2.2.2 Projection Maintenance via Lazy Update

The projection matrix we maintain is of the form $\sqrt{W}A^\top \left(AWA^\top\right)^{-1} A\sqrt{W}$ where $W = \mathrm{diag}(x/s)$. For intuition, we only explain how to maintain the matrix

$$M_w \stackrel{\mathrm{def}}{=} A^\top (AWA^\top)^{-1}A$$

for the short step central path step here. In this case, we have $\sum_i \left(\frac{w_i^{\mathrm{new}} - w_i}{w_i}\right)^2 = O(1)$ for each step. Given this, there are mainly two extreme cases, w changes uniformly on all coordinates and w changes only on a few coordinates.

If the changes $\left(\frac{w_i^{\mathrm{new}} - w_i}{w_i}\right)^2$ is uniformly across all the coordinates, then $w_i^{\mathrm{new}} = (1 \pm \frac{1}{\sqrt{n}})w_i$ for all i. Since it takes $\sqrt{n}$ steps to change all coordinates by a constant factor and we only need to maintain M_v for some $v_i = \Theta(w_i)$ for all i, we can update the matrix every $\sqrt{n}$ steps. Hence, the average cost per iteration of maintaining the projection matrix is $n^{\omega - \frac{1}{2}}$, which is exactly what we desired.

For the other extreme case that the "adversary" puts all of his ℓ_2 budget on few coordinates, only $\sqrt{n}$ coordinates are changed by a constant factor during all $\sqrt{n}$ iterations. In this case, instead of updating M_w every step, we can compute $M_w h$ online by the Woodbury matrix identity.

Fact 2.2.2 ([Woo50]). *The Woodbury matrix identity is*

$$(M + UCV)^{-1} = M^{-1} - M^{-1}U(C^{-1} + VM^{-1}U)^{-1}VM^{-1}.$$

Let $S \subset [n]$ denote the set of coordinates that is changed by more than a constant factor and $r = |S|$. Using the identity above, we have that

$$M_{w^{\mathrm{new}}} = M_w - (M_w)_S(\Delta_{S,S}^{-1} + (M_w)_{S,S})^{-1}((M_w)_S)^\top, \tag{2.9}$$

where $\Delta = \mathrm{diag}(w^{\mathrm{new}} - w)$, $(M_w)_S \in \mathbb{R}^{n \times r}$ is the r columns from S of M_w and $(M_w)_{S,S}, \Delta_{S,S} \in \mathbb{R}^{r \times r}$ are the r rows and columns from S of M_w and Δ.

As long as there are only few coordinates violating $v_i = \Theta(w_i)$, (2.9) can be applied online efficiently. In another case, we can use (2.9) instead to update the matrix M_w and the cost is dominated by multiplying a $n \times n$ matrix with a $n \times n^r$ matrix.

Theorem 2.2.3 (Rectangular matrix multiplication, [LGU18])**.** *Let the dual exponent of matrix multiplication α be the supremum among all $a \geq 0$ such that it takes $n^{2+o(1)}$ time to multiply an $n \times n$ matrix by an $n \times n^a$ matrix.*

Then, for any $n \geq r$, multiplying an $n \times r$ with an $r \times n$ matrix or $n \times n$ with $n \times r$ takes time

$$n^{2+o(1)} + r^{\frac{\omega-2}{1-\alpha}} n^{2-\frac{\alpha(\omega-2)}{1-\alpha}+o(1)}.$$

Furthermore, we have $\alpha > 0.31389$.

See Lemma 2.6.4 for the origin of the formula. Since the cost of multiplying $n \times n$ matrix by a $n \times 1$ matrix is same as the cost for $n \times n$ with $n \times n^{0.31}$, (2.9) should be used to update at least $n^{0.31}$ coordinates. In the extreme case only few w_i are changing, we only need to update the matrix $n^{\frac{1}{2}-0.31}$ times during the whole algorithm and each takes n^2 time, and hence the total cost is less than n^ω for the current value of $\omega \sim 2.37$.

In previous papers [Kar84, Vai89b, NN91, NN94, LS14, LS15], the matrix is updated in a fixed schedule independent of the input sequence w. This leads to sub-optimal bounds if used in this paper. We instead define a potential function to measure the distance between the approximate vector v and the target vector w. When there are less than n^α coordinates

of v that is far from w, we are lazy and do not update the matrix. We simply apply the Woodbury matrix identity online. When there are more than n^α coordinates, we update v by a certain greedy step. As in the extreme cases, the worst case of our algorithm is that the "adversary" puts his ℓ_2 budget across all coordinates uniformly and hence the worst case runtime is $n^{\omega-\frac{1}{2}}$ per iteration. We will further explain the potential in Section 2.5.1.

2.3 Notations

For notation convenience, we assume the number of variables $n \geq 10$ and there is no redundant constraints. In particular, this implies that the constraint matrix A is full rank and $n \geq d$.

For a positive integer n, let $[n]$ denote the set $\{1, 2, \cdots, n\}$.

For any function f, we define $\widetilde{O}(f)$ to be $f \cdot \log^{O(1)}(f)$. In addition to $O(\cdot)$ notation, for two functions f, g, we use the shorthand $f \lesssim g$ (resp. $\gtrsim$) to indicate that $f \leq Cg$ (resp. $\geq$) for some absolute constant C.

We use $\sinh x$ to denote $\frac{e^x - e^{-x}}{2}$ and $\cosh x$ to denote $\frac{e^x + e^{-x}}{2}$.

For vectors $a, b \in \mathbb{R}^n$ and accuracy parameter $\epsilon \in (0, 1)$, we use $a \approx_\epsilon b$ to denote that $(1 - \epsilon)b_i \leq a_i \leq (1 + \epsilon)b_i, \forall i \in [n]$. Similarly, for any scalar t, we use $a \approx_\epsilon t$ to denote that $(1 - \epsilon)t \leq a_i \leq (1 + \epsilon)t, \forall i \in [n]$.

For a vector $x \in \mathbb{R}^n$ and $s \in \mathbb{R}^n$, we use xs to denote a length n vector with the i-th coordinate $(xs)_i$ is $x_i \cdot s_i$. Similarly, we extend other scalar operations to vector coordinate-wise.

Given vectors $x, s \in \mathbb{R}^n$, we use X and S to denote the diagonal matrix of those two vectors. We use $\frac{X}{S}$ to denote the diagonal matrix given $(\frac{X}{S})_{i,i} = x_i/s_i$. Similarly, we extend other scalar operations to diagonal matrix diagonal-wise. Note that matrix $\sqrt{\frac{X}{S}} A^\top (A \frac{X}{S} A^\top)^{-1} A \sqrt{\frac{X}{S}}$ is an orthogonal projection matrix.

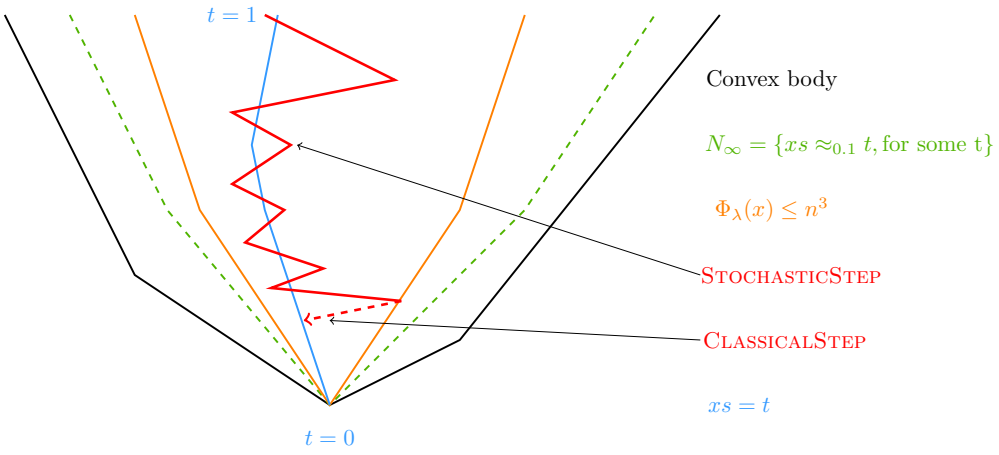


Figure 2.1: ClassicalStep happens with n^{-2} probability

2.4 Stochastic Central Path Method

2.4.1 Proof Outline

The short step central path method is defined using the approximation $(x+\delta_x)_i(s+\delta_s)_i \sim x_i s_i + x_i\delta_{s,i} + s_i\delta_{x,i}$. This approximate is only accurate if $\|X^{-1}\delta_x\|_\infty \le 1/2$ and $\|S^{-1}\delta_s\|_\infty \le 1/2$. For the δ_x step, we have

$$X^{-1}\delta_x = \frac{1}{\sqrt{XS}}(I-P)\frac{1}{\sqrt{XS}}\delta_\mu \sim \frac{1}{t}(I-P)\delta_\mu, \tag{2.10}$$

where we used $x_i s_i \sim t$ for all i.

If we know that $\|\delta_\mu\|_2 \le t/4$, then the ℓ_∞ norm can be bounded as follows:

$$\|X^{-1}\delta_x\|_\infty \le \|X^{-1}\delta_x\|_2 \lesssim \frac{1}{t}\|(I-P)\delta_\mu\|_2 \le \frac{1}{t}\|\delta_\mu\|_2 \le 1/2,$$

where we used that $I-P$ is an orthogonal projection matrix. This is the reason why a standard choice of $\delta_{\mu,i}$ is $-ct/\sqrt{n}$ for all i for some small constant c.

For the stochastic step, $\widetilde{\delta}_{\mu,i} \sim -\frac{t}{\sqrt{n}}\frac{n}{k}$ for roughly k coordinates where the term $\frac{n}{k}$ is used to preserve the expectation of the step. Therefore, the ℓ_2 norm of $\widetilde{\delta}_\mu$ is very large

Algorithm 2.1 Stochastic Step

procedure StochasticStep($\mathrm{mp}, x, s, \delta_\mu, k, \epsilon$) ▷ Lemma 2.4.2,2.4.3,2.4.7

  $w \leftarrow \frac{x}{s}$, $\widetilde{v} \leftarrow \mathrm{mp.Update}(w)$ ▷ Algorithm 2.3

  $\overline{x} \leftarrow x\sqrt{\frac{\widetilde{v}}{w}}$, $\overline{s} \leftarrow s\sqrt{\frac{w}{\widetilde{v}}}$ ▷ It guarantees that $\frac{\overline{x}}{\overline{s}} = \widetilde{v}$ and $\overline{xs} = xs$

  repeat

    Generate $\widetilde{\delta}_\mu$ such that ▷ Compute a sparse direction

    $$\widetilde{\delta}_{\mu,i} \leftarrow \begin{cases} \delta_{\mu,i}/p_i, & \text{with prob. } p_i = \min(1, k \cdot ((\delta_{\mu,i}^2 / \sum_{l=1}^n \delta_{\mu,l}^2) + 1/n)); \\ 0 & \text{else.} \end{cases}$$

    ▷ Compute an approximate step

    ▷ Find $(\widetilde{\delta}_x, \widetilde{\delta}_s, \widetilde{\delta}_y)$ such that these three equations hold

$$\begin{aligned} \overline{X}\widetilde{\delta}_s + \overline{S}\widetilde{\delta}_x &= \widetilde{\delta}_\mu, \\ A\widetilde{\delta}_x &= 0, \\ A^\top \widetilde{\delta}_y + \widetilde{\delta}_s &= 0. \end{aligned}$$

    $p_\mu \leftarrow \mathrm{mp.Query}(\frac{1}{\sqrt{\overline{X}\overline{S}}}\widetilde{\delta}_\mu)$ ▷ Algorithm 2.3

    $\widetilde{\delta}_s \leftarrow \frac{\overline{S}}{\sqrt{\overline{X}\overline{S}}} p_\mu$ ▷ According to (2.11)

    $\widetilde{\delta}_x \leftarrow \frac{1}{\overline{S}}\widetilde{\delta}_\mu - \frac{\overline{X}}{\sqrt{\overline{X}\overline{S}}} p_\mu$ ▷ According to (2.12)

  until $\|\overline{s}^{-1}\widetilde{\delta}_s\|_\infty \leq \frac{1}{100\log n}$ and $\|\overline{x}^{-1}\widetilde{\delta}_x\|_\infty \leq \frac{1}{100\log n}$

  return $(x + \widetilde{\delta}_x, s + \widetilde{\delta}_s)$

end procedure

($\|\widetilde{\delta}_\mu\|_2 \sim t\sqrt{\frac{n}{k}}$). After the projection, we have $\|X^{-1}\delta_x\|_2 \sim \frac{1}{t}\|(I-P)\delta_\mu\|_2 \sim \sqrt{\frac{n}{k}}$. Hence, the bound of $\|X^{-1}\delta_x\|_\infty$ using $\|X^{-1}\delta_x\|_2$ is too weak. To improve the bound, we use Chernoff bounds to estimate $\|X^{-1}\delta_x\|_\infty$.

Beside the ℓ_∞ norm bound, the proof sketch in (2.10) also requires using $x_i s_i \sim t$ for all i. The short step central path proof maintains an invariant that $\sum_i (x_i s_i - t)^2 = O(t^2)$. However, since our stochastic step has a stochastic noise with ℓ_2 norm as large as $t\sqrt{\frac{n}{k}}$, one cannot hope to maintain $x_i s_i$ close to t in ℓ_2 norm. Instead, we follow an idea in

Algorithm 2.2 Our Main Algorithm

procedure MAIN(A, b, c, δ) ▷ Theorem 2.2.1
  $\epsilon \leftarrow \frac{1}{40000 \log n}$, $\epsilon_{mp} \leftarrow \frac{1}{40000}$, $k \leftarrow \frac{1000\epsilon\sqrt{n}\log^2 n}{\epsilon_{mp}}$.
  $\lambda \leftarrow 40 \log n$, $\delta \leftarrow \min(\frac{\delta}{2}, \frac{1}{\lambda})$, $a \leftarrow \min(\alpha, 2/3)$.
  Modify the linear program and obtain an initial x and s according to Lemma 3.8.2.
  MAINTAINPROJECTION mp
  mp.INITIALIZE($A, \frac{x}{s}, \epsilon_{mp}, a$) ▷ Algorithm 2.3
  $t \leftarrow 1$ ▷ Initialize t
  while $t > \delta^2/(2n)$ **do** ▷ We stopped once the precision is good
    $t^{\mathrm{new}} \leftarrow (1 - \frac{\epsilon}{3\sqrt{n}})t$
    $\mu \leftarrow xs$
    $\delta_\mu \leftarrow (\frac{t^{\mathrm{new}}}{t} - 1)xs - \frac{\epsilon}{2} \cdot t^{\mathrm{new}} \cdot \frac{\nabla\Phi_\lambda(\mu/t-1)}{\|\nabla\Phi_\lambda(\mu/t-1)\|_2}$ ▷ Φ_λ is defined in Lemma 2.4.11
    $(x^{\mathrm{new}}, s^{\mathrm{new}}) \leftarrow$ STOCHASTICSTEP(mp, $x, s, \delta_\mu, k, \epsilon$) ▷ Algorithm 2.1
    if $\Phi_\lambda(\mu^{\mathrm{new}}/t^{\mathrm{new}} - 1) > n^3$ **then** ▷ When potential function is large
      $(x^{\mathrm{new}}, s^{\mathrm{new}}) \leftarrow$ CLASSICALSTEP(x, s, t^{new}) ▷ Lemma 2.6.2, [Vai89b]
      mp.INITIALIZE($A, \frac{x^{\mathrm{new}}}{s^{\mathrm{new}}}, \epsilon_{mp}, a$) ▷ Restart the data structure
    end if
    $(x, s) \leftarrow (x^{\mathrm{new}}, s^{\mathrm{new}})$, $t \leftarrow t^{\mathrm{new}}$
  end while
  Return an approximate solution of the original linear program according to Lemma 3.8.2.
end procedure

[LS14, LSW15] and maintain the following potential

$$\sum_{i=1}^{n} \cosh\left(\lambda\left(\frac{x_i s_i}{t} - 1\right)\right) = n^{O(1)}$$

with $\lambda = \Theta(\log n)$. This potential is a variant of soft-max. Note that the potential bounded by $n^{O(1)}$ implies that $x_i s_i$ is a multiplicative approximation of t. To bound the potential, consider $r_i = \frac{x_i s_i}{t}$ and $\Phi(r)$ be the potential above. Then, we have that

$$\mathbb{E}[\Phi(r^{\text{new}})] \leq \Phi(r) + \langle \nabla\Phi(r), \mathbb{E}[r^{\text{new}} - r]\rangle + O(1)\,\mathbb{E}\,\|r^{\text{new}} - r\|^2_{\nabla^2\Phi(r)}.$$

The first order term can be bounded efficiently because $\mathbb{E}[r^{\text{new}} - r]$ is close to the short step central path step. The second term is a variance term which scales like $1/k$ due to the k independent coordinates. Therefore, the potential changed by $1/k \sim 1/\sqrt{n}$ factor each step. Hence, we can maintain it for roughly $\sqrt{n}$ steps.

To make sure the potential Φ is bounded during the whole algorithm, our step is the mixtures of two steps of the form $\delta_\mu \sim -\frac{t}{\sqrt{n}} - t\frac{\nabla\Phi}{\|\nabla\Phi\|_2}$. The first term is to decrease t and the second term is to decrease Φ.

Since the algorithm is randomized, there is a tiny probability that Φ is large. In that case, we switch to a short step central path method. See Figure 2.1, Algorithm 2.1, and Algorithm 3.6. The first part of the proof involves bounding every quantity listed in Table 2.1. In the second part, we are using these quantities to bound the expectation of Φ.

To decouple the proof in both parts, we will make the following assumption in first part. It will be verified in the second part.

Assumption 2.4.1. *Assume the following for the input of the procedure* StochasticStep *(see Algorithm 2.1):*

- $xs \approx_{0.1} t$ *with* $t > 0$.
- $mp.\textsc{Update}(w)$ *outputs* $\widetilde{v}$ *such that* $w \approx_{\epsilon_{mp}} \widetilde{v}$ *with* $\epsilon_{mp} \leq 1/40000$.
- $\|\delta_\mu\|_2 \leq \epsilon t$ *with* $0 < \epsilon < 1/(40000 \log n)$.
- $k \geq 1000\epsilon\sqrt{n}\log^2 n/\epsilon_{mp}$.

Quantity	Bound	Place
$\|\mathbb{E}[s^{-1}\widetilde{\delta}_s]\|_2, \|\mathbb{E}[x^{-1}\widetilde{\delta}_x]\|_2, \|\mathbb{E}[\mu^{-1}\widetilde{\delta}_\mu]\|_2$	$O(\epsilon)$	Part 1, Lemma 2.4.3
$\|\mathbb{E}[\mu^{-1}(\mu^{\mathrm{new}} - \mu - \widetilde{\delta}_\mu)]\|_2$	$O(\epsilon_{mp} \cdot \epsilon)$	Part 1, Lemma 2.4.7
$\|\mathbb{E}[\mu^{-1}(\mu^{\mathrm{new}} - \mu)]\|_2$	$O(\epsilon)$	Part 1, Lemma 2.4.7
$\mathbb{V}[s_i^{-1}\widetilde{\delta}_{s,i}], \mathbb{V}[x_i^{-1}\widetilde{\delta}_{x,i}], \mathbb{V}[\mu_i^{-1}\widetilde{\delta}_{\mu,i}]$	$O(\epsilon^2/k)$	Part 2, Lemma 2.4.3
$\mathbb{V}[\mu_i^{-1}\mu^{\mathrm{new}}]$	$O(\epsilon^2/k)$	Part 2, Lemma 2.4.7
$\|s^{-1}\widetilde{\delta}_s\|_\infty, \|x^{-1}\widetilde{\delta}_x\|_\infty, \|\mu^{-1}\widetilde{\delta}_\mu\|_\infty$	$O(1/\log n)$	Part 3, Lemma 2.4.3
$\|\mu^{-1}(\mu^{\mathrm{new}} - \mu)\|_\infty$	$O(1/\log n)$	Part 3, Lemma 2.4.7

Table 2.1: The bound of each quantity under Assumption 2.4.1. For intuition, think $\epsilon \sim \epsilon_{mp} \sim 1/10$ and $k \sim \sqrt{n}$.

2.4.2 Bounding each quantity of stochastic step

First, we give an explicit formula for our step, which will be used in all subsequent calculations.

Lemma 2.4.2. *The procedure* StochasticStep$(\mathrm{mp}, x, s, \delta_\mu, k, \epsilon)$ *(see Algorithm 2.1) finds a solution* $\widetilde{\delta}_x$, $\widetilde{\delta}_s \in \mathbb{R}^n$ *to* (2.7) *by the formula*

$$\widetilde{\delta}_x = \frac{\overline{X}}{\sqrt{\overline{X}\overline{S}}}(I - \overline{P})\frac{1}{\sqrt{\overline{X}\overline{S}}}\widetilde{\delta}_\mu \tag{2.11}$$

$$\widetilde{\delta}_s = \frac{\overline{S}}{\sqrt{\overline{X}\overline{S}}}\overline{P}\frac{1}{\sqrt{\overline{X}\overline{S}}}\widetilde{\delta}_\mu \tag{2.12}$$

with

$$\overline{P} = \sqrt{\frac{\overline{X}}{\overline{S}}}A^\top \left(A\frac{\overline{X}}{\overline{S}}A^\top\right)^{-1} A\sqrt{\frac{\overline{X}}{\overline{S}}}. \tag{2.13}$$

Proof. For the first equation of (2.7), we multiply $A\overline{S}^{-1}$ on both sides,

$$A\overline{S}^{-1}\overline{X}\widetilde{\delta}_s + A\widetilde{\delta}_x = A\overline{S}^{-1}\widetilde{\delta}_\mu.$$

Since the second equation gives $A\widetilde{\delta}_x = 0$, then we know that $A\overline{S}^{-1}\overline{X}\widetilde{\delta}_s = A\overline{S}^{-1}\widetilde{\delta}_\mu$.

Multiplying $A\overline{S}^{-1}\overline{X}$ on both sides of the third equation of (2.7), we have

$$-A\overline{S}^{-1}\overline{X}A^\top\widetilde{\delta}_y = A\overline{S}^{-1}\overline{X}\widetilde{\delta}_s = A\overline{S}^{-1}\widetilde{\delta}_\mu.$$

Thus,

$$\begin{aligned}
\widetilde{\delta}_y &= -(A\overline{S}^{-1}\overline{X}A^\top)^{-1}A\overline{S}^{-1}\widetilde{\delta}_\mu, \\
\widetilde{\delta}_s &= A^\top(A\overline{S}^{-1}\overline{X}A^\top)^{-1}A\overline{S}^{-1}\widetilde{\delta}_\mu, \\
\widetilde{\delta}_x &= \overline{S}^{-1}\widetilde{\delta}_\mu - \overline{S}^{-1}\overline{X}A^\top(A\overline{S}^{-1}\overline{X}A^\top)^{-1}A\overline{S}^{-1}\widetilde{\delta}_\mu.
\end{aligned}$$

Recall we define $\overline{P}$ as (2.13), then we have

$$\widetilde{\delta}_s = \frac{\overline{S}}{\sqrt{\overline{X}\overline{S}}}\cdot\sqrt{\frac{\overline{X}}{\overline{S}}}A^\top(A\frac{\overline{X}}{\overline{S}}A^\top)^{-1}\sqrt{\frac{\overline{X}}{\overline{S}}}\cdot\frac{1}{\sqrt{\overline{X}\overline{S}}}\widetilde{\delta}_\mu = \frac{\overline{S}}{\sqrt{\overline{X}\overline{S}}}\overline{P}\frac{1}{\sqrt{\overline{X}\overline{S}}}\widetilde{\delta}_\mu,$$

and

$$\widetilde{\delta}_x = \overline{S}^{-1}\widetilde{\delta}_\mu - \frac{\overline{X}}{\sqrt{\overline{X}\overline{S}}}\cdot\sqrt{\frac{\overline{X}}{\overline{S}}}A^\top(A\frac{\overline{X}}{\overline{S}}A^\top)^{-1}\sqrt{\frac{\overline{X}}{\overline{S}}}\cdot\frac{1}{\sqrt{\overline{X}\overline{S}}}\widetilde{\delta}_\mu = \frac{\overline{X}}{\sqrt{\overline{X}\overline{S}}}(I-\overline{P})\frac{1}{\sqrt{\overline{X}\overline{S}}}\widetilde{\delta}_\mu.$$

which are matching (2.11) and (2.12).

To see why the StochasticStep outputs $\widetilde{\delta}_x$, $\widetilde{\delta}_s$ satisfying (2.11) and (2.12), we note that

$$p_\mu = \sqrt{\widetilde{V}}A^\top\left(A\frac{\overline{X}}{\overline{S}}A^\top\right)^{-1}A\sqrt{\widetilde{V}}\frac{1}{\sqrt{\overline{X}\overline{S}}}\widetilde{\delta}_\mu = \overline{P}\frac{1}{\sqrt{\overline{X}\overline{S}}}\widetilde{\delta}_\mu$$

because of Theorem 2.5.1. □

Using the explicitly formula, we are ready to bound all quantities we needed in the following two subsubsections.

2.4.2.1 Bounding $\widetilde{\delta}_s$, $\widetilde{\delta}_x$ and $\widetilde{\delta}_\mu$

Lemma 2.4.3. *Under the Assumption 2.4.1, the two vectors $\widetilde{\delta}_x$ and $\widetilde{\delta}_s$ found by* StochasticStep *satisfy :*

1. $\|\mathbb{E}[\overline{s}^{-1}\widetilde{\delta}_s]\|_2 \leq 2\epsilon, \|\mathbb{E}[\overline{x}^{-1}\widetilde{\delta}_x]\|_2 \leq 2\epsilon, \|\mathbb{E}[s^{-1}\widetilde{\delta}_s]\|_2 \leq 2\epsilon, \|\mathbb{E}[x^{-1}\widetilde{\delta}_x]\|_2 \leq 2\epsilon, \|\mathbb{E}[\mu^{-1}\widetilde{\delta}_\mu]\|_2 \leq 4\epsilon$.

2. $\mathbb{V}[\frac{\widetilde{\delta}_{s,i}}{\overline{s}_i}] \leq \frac{2\epsilon^2}{k}, \mathbb{V}[\frac{\widetilde{\delta}_{x,i}}{\overline{x}_i}] \leq \frac{2\epsilon^2}{k}, \mathbb{V}[\frac{\widetilde{\delta}_{s,i}}{s_i}] \leq \frac{2\epsilon^2}{k}, \mathbb{V}[\frac{\widetilde{\delta}_{x,i}}{x_i}] \leq \frac{2\epsilon^2}{k}, \mathbb{V}[\frac{\widetilde{\delta}_{\mu,i}}{\mu_i}] \leq \frac{8\epsilon^2}{k}$.

3. $\|\overline{s}^{-1}\widetilde{\delta}_s\|_\infty \leq \frac{0.01}{\log n}, \|s^{-1}\widetilde{\delta}_s\|_\infty \leq \frac{0.02}{\log n}, \|\overline{x}^{-1}\widetilde{\delta}_x\|_\infty \leq \frac{0.01}{\log n}, \|x^{-1}\widetilde{\delta}_x\|_\infty \leq \frac{0.02}{\log n}, \|\mu^{-1}\widetilde{\delta}_\mu\|_\infty \leq \frac{0.02}{\log n}$.

Remark 2.4.1. For notational simplicity, the $\mathbb{E}$ and $\mathbb{V}$ in the proof are for the case without resampling (Line 12). Since the all the additional terms due to resampling are polynomially bounded and since we can set failure probability to an arbitrarily small inverse polynomial (see Claim 2.4.6), the proof does not change and the result remains the same.

Proof.

Claim 2.4.4 (Part 1, bounding the ℓ_2 norm of expectation)**.**

$$\|\mathbb{E}[\overline{s}^{-1}\widetilde{\delta}_s]\|_2 \leq 2\epsilon, \|\mathbb{E}[\overline{x}^{-1}\widetilde{\delta}_x]\|_2 \leq 2\epsilon, \|\mathbb{E}[s^{-1}\widetilde{\delta}_s]\|_2 \leq 2\epsilon, \|\mathbb{E}[x^{-1}\widetilde{\delta}_x]\|_2 \leq 2\epsilon, \|\mathbb{E}[\mu^{-1}\widetilde{\delta}_\mu]\|_2 \leq 4\epsilon.$$

Proof. For $\|\overline{s}^{-1}\widetilde{\delta}_s\|_\infty$, we consider the i-th coordinate of the vector

$$\overline{s}_i^{-1}\widetilde{\delta}_{s,i} = \frac{1}{\sqrt{\overline{x}_i\overline{s}_i}}\sum_{j=1}^{n}\overline{P}_{i,j}\frac{\widetilde{\delta}_{\mu,j}}{\sqrt{\overline{x}_j\overline{s}_j}}.$$

Then, we have

$$\mathbb{E}\left[\overline{s}_i^{-1}\widetilde{\delta}_{s,i}\right] = \frac{1}{\sqrt{\overline{x}_i\overline{s}_i}}\sum_{j=1}^{n}\overline{P}_{i,j}\frac{\mathbb{E}[\widetilde{\delta}_{\mu,j}]}{\sqrt{\overline{x}_j\overline{s}_j}} = \frac{1}{\sqrt{\overline{x}_i\overline{s}_i}}\sum_{j=1}^{n}\overline{P}_{i,j}\frac{\delta_{\mu,j}}{\sqrt{\overline{x}_j\overline{s}_j}}.$$

Since $xs \approx_{0.1} t$ and $\|\delta_\mu\| \leq \epsilon t$, we have $\|\frac{\delta_\mu}{\sqrt{xs}}\|_2 \leq \frac{1.1\epsilon t}{\sqrt{t}}$. Since $\overline{P}$ is an orthogonal projection matrix, we have $\|\overline{P}\frac{\delta_\mu}{\sqrt{\overline{xs}}}\|_2 \leq \|\frac{\delta_\mu}{\sqrt{\overline{xs}}}\|_2$. Putting all the above facts and $xs = \overline{xs}$, we can show

$$\begin{aligned}
\left\|\mathbb{E}[\overline{s}^{-1}\widetilde{\delta}_s]\right\|_2^2 &= \sum_{i=1}^n \left(\frac{1}{\sqrt{\overline{x}_i\overline{s}_i}}\sum_{j=1}^n \overline{P}_{i,j}\frac{\delta_{\mu,j}}{\sqrt{\overline{x}_j\overline{s}_j}}\right)^2 = \sum_{i=1}^n \frac{1}{\overline{x}_i\overline{s}_i}\left(\sum_{j=1}^n \overline{P}_{i,j}\frac{\delta_{\mu,j}}{\sqrt{\overline{x}_j\overline{s}_j}}\right)^2 \\
&\leq \frac{1}{0.9t}\sum_{i=1}^n \left(\sum_{j=1}^n \overline{P}_{i,j}\frac{\delta_{\mu,j}}{\sqrt{\overline{x}_j\overline{s}_j}}\right)^2 = \frac{1}{0.9t}\|\overline{P}\frac{\delta_\mu}{\sqrt{\overline{xs}}}\|_2^2 \\
&\leq \frac{1}{0.9t}\|\frac{\delta_\mu}{\sqrt{\overline{xs}}}\|_2^2 \leq \frac{(1.1)^2}{0.9t}\cdot\frac{(\epsilon t)^2}{t} \leq 1.4\epsilon^2,
\end{aligned}$$

which implies that

$$\left\|\mathbb{E}[\overline{s}^{-1}\widetilde{\delta}_s]\right\|_2 \leq 1.2\epsilon. \tag{2.14}$$

Notice that the proof for x is identical to the proof for s because $(I - \overline{P})$ is also a projection matrix. Since $\overline{s} \approx_{0.1} s$ and $\overline{x} \approx_{0.1} x$, then we can also prove the next two inequalities in the Claim statement.

Now, we are ready to bound $\|\mathbb{E}[\mu^{-1}\widetilde{\delta}_\mu]\|_2$

$$\|\mathbb{E}[\mu^{-1}\widetilde{\delta}_\mu]\|_2 = \|\mathbb{E}[\overline{s}^{-1}\overline{x}^{-1}(\overline{x}\widetilde{\delta}_s + \overline{s}\widetilde{\delta}_x)]\|_2 \leq \|\mathbb{E}[\overline{s}^{-1}\widetilde{\delta}_s]\|_2 + \|\mathbb{E}[\overline{x}^{-1}\widetilde{\delta}_x]\|_2 \leq 4\epsilon.$$

by using $\mu = xs = \overline{xs}$ and $\overline{x}\widetilde{\delta}_s + \overline{s}\widetilde{\delta}_x = \widetilde{\delta}_\mu$ from (2.7). □

Claim 2.4.5 (Part 2, bounding the variance per coordinate)**.**

$$\mathbb{V}[\overline{s}_i^{-1}\widetilde{\delta}_{s,i}] \leq \frac{2\epsilon^2}{k}, \mathbb{V}[\overline{x}_i^{-1}\widetilde{\delta}_{x,i}] \leq \frac{2\epsilon^2}{k}, \mathbb{V}[s_i^{-1}\widetilde{\delta}_{s,i}] \leq \frac{2\epsilon^2}{k}, \mathbb{V}[x_i^{-1}\widetilde{\delta}_{x,i}] \leq \frac{2\epsilon^2}{k}, \mathbb{V}[\mu_i^{-1}\widetilde{\delta}_{\mu,i}] \leq \frac{8\epsilon^2}{k}.$$

Proof. Consider the i-th coordinate of the vector

$$\overline{s}_i^{-1}\widetilde{\delta}_{s,i} = \frac{1}{\sqrt{\overline{x}_i\overline{s}_i}}\sum_{j=1}^n \overline{P}_{i,j}\frac{\widetilde{\delta}_{\mu,j}}{\sqrt{\overline{x}_j\overline{s}_j}}.$$

For variance of $\overline{s}_i^{-1}\widetilde{\delta}_{s,i}$, we have

$$\begin{aligned}
\mathbb{V}[\overline{s}_i^{-1}\widetilde{\delta}_{s,i}] &= \frac{1}{\overline{x}_i\overline{s}_i}\sum_{j=1}^n \frac{\overline{P}_{i,j}^2}{\overline{x}_j\overline{s}_j}\mathbb{V}[\widetilde{\delta}_{\mu,j}] && \text{by all } \widetilde{\delta}_{\mu,j} \text{ are independent} \\
&\leq \frac{1}{\overline{x}_i\overline{s}_i}\sum_{j=1}^n \frac{\overline{P}_{i,j}^2}{\overline{x}_j\overline{s}_j}\frac{1}{k}\frac{\delta_{\mu,j}^2}{\frac{\delta_{\mu,j}^2}{\sum_{l=1}^n \delta_{\mu,l}^2}+\frac{1}{n}} && \text{by (2.6)} \\
&\leq \frac{1}{\overline{x}_i\overline{s}_i}\sum_{j=1}^n \frac{\overline{P}_{i,j}^2}{\overline{x}_j\overline{s}_j}\frac{1}{k}\sum_{l=1}^n \delta_{\mu,l}^2 \\
&\leq \frac{1.3}{t^2}\sum_{j=1}^n \overline{P}_{i,j}^2\frac{1}{k}\sum_{l=1}^n \delta_{\mu,l}^2 \leq \frac{1.3\epsilon^2}{k}, && \text{by } \overline{x}_i\overline{s}_i = x_is_i \approx_{1/10} t
\end{aligned}$$

where we used that $\sum_{j=1}^n \overline{P}_{i,j}^2 = \overline{P}_{i,i} \leq 1$, $\|\delta_\mu\|_2 \leq \epsilon t$ at the end.

The proof for the other three inequalities in the Claim statement are identical to this one. We omit here.

For the variance of $\mu_i^{-1}\widetilde{\delta}_{\mu,i}$,

$$\begin{aligned}
\mathbb{V}[\mu_i^{-1}\widetilde{\delta}_{\mu,i}] =\ & \mathbb{V}[\overline{x}_i^{-1}\overline{s}_i^{-1}(\overline{x}_i\widetilde{\delta}_{s,i} + \overline{s}_i\widetilde{\delta}_{x,i})] \\
&\leq 2\,\mathbb{V}[\overline{x}_i^{-1}\overline{x}_i\overline{s}_i^{-1}\widetilde{\delta}_{s,i}] + 2\,\mathbb{V}[\overline{s}_i^{-1}\overline{s}_i\overline{x}_i^{-1}\widetilde{\delta}_{x,i}] \\
&= 2\,\mathbb{V}[\overline{s}_i^{-1}\widetilde{\delta}_{s,i}] + 2\,\mathbb{V}[\overline{x}_i^{-1}\widetilde{\delta}_{x,i}] \leq 8\epsilon^2/k.
\end{aligned}$$

where the first step follows by definition of $\mu = xs = \overline{x}\overline{s}$ and (2.7), the second step follows by triangle inequality and, the last step follows by $\mathbb{V}[\overline{s}_i^{-1}\widetilde{\delta}_{s,i}], \mathbb{V}[\overline{x}_i^{-1}\widetilde{\delta}_{x,i}] \leq 2\epsilon^2/k$ □

Claim 2.4.6 (Part 3, bounding the probability of success)**.** *Without resampling, the following holds with probability* $1 - 2n\exp(-\frac{0.003k}{\epsilon\sqrt{n}\log n})$.

$$\|\overline{s}^{-1}\widetilde{\delta}_s\|_\infty \leq \frac{0.01}{\log n}, \|s^{-1}\widetilde{\delta}_s\|_\infty \leq \frac{0.02}{\log n}, \|\overline{x}^{-1}\widetilde{\delta}_x\|_\infty \leq \frac{0.01}{\log n}, \|x^{-1}\widetilde{\delta}_x\|_\infty \leq \frac{0.02}{\log n}, \|\mu^{-1}\widetilde{\delta}_\mu\|_\infty \leq \frac{0.02}{\log n}.$$

With resampling, it always holds.

Proof. We can write $\overline{s}_i^{-1}\widetilde{\delta}_{s,i} - \mathbb{E}[\overline{s}_i^{-1}\widetilde{\delta}_{s,i}] = \sum_j Y_j$ where Y_j are independent random variables defined by

$$Y_j = \frac{1}{\sqrt{\overline{x}_i\overline{s}_i}}\overline{P}_{i,j}\frac{\widetilde{\delta}_{\mu,j}}{\sqrt{\overline{x}_j\overline{s}_j}} - \frac{1}{\sqrt{\overline{x}_i\overline{s}_i}}\overline{P}_{i,j}\frac{\delta_{\mu,j}}{\sqrt{\overline{x}_j\overline{s}_j}}.$$

We bound the sum using Bernstein inequality (Lemma A.1.2). Note that Y_j are mean 0 and that Claim 2.4.5 shows that $\sum_{j=1}^n \mathbb{E}[Y_j^2] = \mathbb{V}[\overline{s}_i^{-1}\widetilde{\delta}_{s,i}] \le \frac{2\epsilon^2}{k}$. We also need to give an upper bound for Y_j

$$\begin{aligned}
|Y_j| &= \left|\frac{1}{\sqrt{\overline{x}_i\overline{s}_i}}\overline{P}_{i,j}\left(\frac{\widetilde{\delta}_{\mu,j}-\delta_{\mu,j}}{\sqrt{\overline{x}_j\overline{s}_j}}\right)\right| \\
&\le \frac{1.2}{t}|\widetilde{\delta}_{\mu,j}-\delta_{\mu,j}| && \text{by } |\overline{P}_{i,j}| \le 1, x_i s_i \approx_{1/10} t \\
&\le \frac{1.2}{t}|\delta_{\mu,j}/p_j| && \text{by } \widetilde{\delta}_{\mu,j} \in [0, \delta_{\mu,j}/p_j] \\
&= \frac{1.2}{t}\frac{1}{k}\frac{1}{(\frac{\delta_{\mu,i}}{\sum_{l=1}^n \delta_{\mu,l}^2} + \frac{1}{n\delta_{\mu,i}})} && \text{by (2.6)} \\
&\le \frac{0.6}{t}\frac{1}{k}\left(n\sum_{l=1}^n \delta_{\mu,l}^2\right)^{1/2} && \text{by } a^2+b^2 \ge 2ab \\
&\le \frac{0.6\epsilon\sqrt{n}}{k} \stackrel{\text{def}}{=} M. && \text{by } \|\delta_\mu\|_2 \le \epsilon t
\end{aligned}$$

Now, we can apply Bernstein inequality (Lemma A.1.2)

$$\begin{aligned}
\Pr\left[\left|\sum_{j=1}^n Y_j\right| > b\right] &\le 2\exp\left(-\frac{b^2/2}{\sum_{j=1}^n \mathbb{E}[Y_j^2] + Mb/3}\right) \\
&\le 2\exp\left(-\frac{b^2/2}{2\epsilon^2/k + (0.6\epsilon\sqrt{n}/k)\cdot b/3}\right).
\end{aligned}$$

We choose $b = \frac{0.005}{\log n}$ and use $\epsilon \le \frac{1}{400\log n}$ and $n \ge 10$ to get

$$\Pr\left[\left|\sum_{j=1}^n Y_j\right| \ge \frac{0.05}{\log n}\right] \le 2\exp\left(-\frac{0.003k}{\epsilon\sqrt{n}\log n}\right).$$

Since $\|\mathbb{E}[\overline{s}_i^{-1}\widetilde{\delta}_{s,i}]\|_2 \le 2\epsilon \le \frac{0.005}{\log n}$, we have that $|\overline{s}_i^{-1}\widetilde{\delta}_{s,i}| \le \frac{0.01}{\log n}$ with probability $1 - 2\exp(-\frac{0.003k}{\epsilon\sqrt{n}\log n})$. Taking a union bound, we have that $\|\overline{s}^{-1}\widetilde{\delta}_s\|_\infty \le \frac{0.01}{\log n}$ with probability $1 - 2n\exp(-\frac{0.003k}{\epsilon\sqrt{n}\log n})$. Similarly, this holds for the other 3 terms.

Now, the last term follows by

$$|\mu_i^{-1}\widetilde{\delta}_{\mu,i}| = |\overline{x}_i^{-1}\overline{s}_i^{-1}(\overline{x}_i\widetilde{\delta}_{s,i} + \overline{s}_i\widetilde{\delta}_{x,i})| = |\overline{s}_i^{-1}\widetilde{\delta}_{s,i}| + |\overline{x}_i^{-1}\widetilde{\delta}_{x,i}| \le \frac{0.02}{\log n}.$$

□

□

2.4.2.2 Bounding $\mu^{\text{new}} - \mu$

Lemma 2.4.7. *Under the Assumption 2.4.1, the vector* $\mu_i^{\text{new}} \stackrel{\text{def}}{=} (x_i + \widetilde{\delta}_{x,i})(s_i + \widetilde{\delta}_{s,i})$ *satisfies*

1. $\|\mathbb{E}[\mu^{-1}(\mu^{\text{new}} - \mu - \widetilde{\delta}_\mu)]\|_2 \le 10\epsilon_{mp}\cdot\epsilon$ *and* $\|\mathbb{E}[\mu^{-1}(\mu^{\text{new}} - \mu)]\|_2 \le 5\epsilon$.
2. $\mathbb{V}[\mu_i^{-1}\mu_i^{\text{new}}] \le 50\epsilon^2/k$ *for all* i.
3. $\|\mu^{-1}(\mu^{\text{new}} - \mu)\|_\infty \le \frac{0.021}{\log n}$.

Claim 2.4.8 (Part 1 of Lemma 2.4.7)**.**

$$\|\mathbb{E}[\mu^{-1}(\mu^{\text{new}} - \mu - \widetilde{\delta}_\mu)]\|_2 \le 10\epsilon_{mp}\cdot\epsilon, \text{ and } \|\mathbb{E}[\mu^{-1}(\mu^{\text{new}} - \mu)]\|_2 \le 5\epsilon.$$

Proof.

$$\mu^{\text{new}} = (x + \widetilde{\delta}_x)(s + \widetilde{\delta}_s) = \mu + x\widetilde{\delta}_s + s\widetilde{\delta}_x + \widetilde{\delta}_x\widetilde{\delta}_s = \mu + \underbrace{\overline{x}\widetilde{\delta}_s + \overline{s}\widetilde{\delta}_x}_{\widetilde{\delta}_\mu} + \underbrace{(x-\overline{x})\widetilde{\delta}_s + (s-\overline{s})\widetilde{\delta}_x + \widetilde{\delta}_x\widetilde{\delta}_s}_{\epsilon_\mu}.$$

Taking the expectation on both sides, we have

$$\mathbb{E}[\mu^{\text{new}} - \mu - \widetilde{\delta}_\mu] = (x - \overline{x})\,\mathbb{E}[\widetilde{\delta}_s] + (s - \overline{s})\,\mathbb{E}[\widetilde{\delta}_x] + \mathbb{E}[\widetilde{\delta}_x\widetilde{\delta}_s].$$

Hence, we have that

$$\begin{aligned}
&\|\mu^{-1}\,\mathbb{E}[\mu^{\text{new}} - \mu - \widetilde{\delta}_\mu]\|_2 \\
&\le \|\mu^{-1}(x-\overline{x})s\cdot s^{-1}\,\mathbb{E}[\widetilde{\delta}_s]\|_2 + \|\mu^{-1}(s-\overline{s})x\cdot x^{-1}\,\mathbb{E}[\widetilde{\delta}_x]\|_2 + \|\mu^{-1}\,\mathbb{E}[\widetilde{\delta}_x\widetilde{\delta}_s]\|_2 \\
&\le \|\mu^{-1}(x-\overline{x})s\|_\infty\cdot\|s^{-1}\,\mathbb{E}[\widetilde{\delta}_s]\|_2 + \|\mu^{-1}(s-\overline{s})x\|_\infty\cdot\|x^{-1}\,\mathbb{E}[\widetilde{\delta}_x]\|_2 + \|\mu^{-1}\,\mathbb{E}[\widetilde{\delta}_x\widetilde{\delta}_s]\|_2 \\
&\le \epsilon_{mp}\cdot\|s^{-1}\,\mathbb{E}[\widetilde{\delta}_s]\|_2 + \epsilon_{mp}\cdot\|x^{-1}\,\mathbb{E}[\widetilde{\delta}_x]\|_2 + \|\mu^{-1}\,\mathbb{E}[\widetilde{\delta}_x\widetilde{\delta}_s]\|_2 \\
&\le 4\epsilon_{mp}\cdot\epsilon + \|\mu^{-1}\,\mathbb{E}[\widetilde{\delta}_x\widetilde{\delta}_s]\|_2, \qquad (2.15)
\end{aligned}$$

where the first step follows by triangle inequality, the second step follows by $\|ab\|_2 \le \|a\|_\infty \cdot \|b\|_2$, the third step follows by $\|\mu^{-1}(x-\overline{x})s\|_\infty \le \epsilon_{mp}$ and $\|\mu^{-1}(s-\overline{s})x\|_\infty \le \epsilon_{mp}$ (since $\overline{x} \approx_{\epsilon_{mp}} x$, $\overline{s} \approx_{\epsilon_{mp}} s$), the last step follows by $\|\,\mathbb{E}[s^{-1}\widetilde{\delta}_s]\|_2 \le 2\epsilon$ and $\|\,\mathbb{E}[x^{-1}\widetilde{\delta}_x]\|_2 \le 2\epsilon$ (Part 1 of Lemma 2.4.3).

To bound the last term, using $\mathbb{E}[\widetilde{\delta}_s] = \delta_s$ and $\mathbb{E}[\widetilde{\delta}_x] = \delta_x$, we note that

$$\mathbb{E}[\widetilde{\delta}_{x,i}\widetilde{\delta}_{s,i}] = \delta_{x,i}\delta_{s,i} + \mathbb{E}[(\widetilde{\delta}_{x,i} - \delta_{x,i})(\widetilde{\delta}_{s,i} - \delta_{s,i})].$$

Hence, we have

$$\begin{aligned}
\|\mu^{-1}\,\mathbb{E}[\widetilde{\delta}_x\widetilde{\delta}_s]\|_2 &\le \|\mu^{-1}\delta_x\delta_s\|_2 + \left(\sum_{i=1}^n\left(\mathbb{E}\left[x_i^{-1}(\widetilde{\delta}_{x,i}-\delta_{x,i})\cdot s_i^{-1}(\widetilde{\delta}_{s,i}-\delta_{s,i})\right]\right)^2\right)^{1/2} \\
&\le 4\epsilon^2 + \frac{1}{2}\left(\sum_{i=1}^n\left(\mathbb{V}[x_i^{-1}\widetilde{\delta}_{x,i}] + \mathbb{V}[s_i^{-1}\widetilde{\delta}_{s,i}]\right)^2\right)^{1/2} \\
&\le 4\epsilon^2 + \frac{1}{2}\left(\sum_{i=1}^n 2(\mathbb{V}[x_i^{-1}\widetilde{\delta}_{x,i}])^2 + 2(\mathbb{V}[s_i^{-1}\widetilde{\delta}_{s,i}])^2\right)^{1/2} \\
&\le 4\epsilon^2 + 2\sqrt{n\cdot\epsilon^4/k^2} \le 4\epsilon^2 + 2\epsilon\cdot\epsilon_{mp} \le 6\epsilon\cdot\epsilon_{mp}, \qquad (2.16)
\end{aligned}$$

where the second step follows by $\|\mu^{-1}\delta_x\delta_s\|_2 \le \|x^{-1}\delta_x\|_2\cdot\|s^{-1}\delta_s\|_2 \le 4\epsilon^2$ (Part 1 of Lemma 2.4.3) and $2ab \le a^2+b^2$, the third step follows by $(a+b)^2 \le 2a^2+2b^2$, the fourth step follows by

$\mathbb{V}[x_i^{-1}\widetilde{\delta}_{x,i}] \le 2\epsilon^2/k$ and $\mathbb{V}[s_i^{-1}\widetilde{\delta}_{s,i}] \le 2\epsilon^2/k$ (Part 2 of Lemma 2.4.3), the last step follows by $k \ge \frac{\epsilon\sqrt{n}}{\epsilon_{mp}}$.

Combining (2.15) and (2.16), we have that

$$\|\mu^{-1}(\mathbb{E}[\mu^{\text{new}} - \mu - \widetilde{\delta}_\mu])\|_2 \le 4\epsilon_{mp}\cdot\epsilon + \|\mu^{-1}\,\mathbb{E}[\widetilde{\delta}_x\widetilde{\delta}_s]\|_2 \le 10\epsilon_{mp}\cdot\epsilon.$$

where we used $\epsilon \le \epsilon_{mp}$.

From Part 1 of Lemma 2.4.3, we know that $\|\mu^{-1}\,\mathbb{E}[\widetilde{\delta}_\mu]\|_2 \le 4\epsilon$. Thus using triangle inequality, we know

$$\|\mu^{-1}(\mathbb{E}[\mu^{\text{new}} - \mu])\|_2 \le 10\epsilon_{mp}\cdot\epsilon + 4\epsilon \le 5\epsilon.$$

□

Claim 2.4.9 (Part 2 of Lemma 2.4.7)**.** $\mathbb{V}[\mu_i^{-1}\mu_i^{\text{new}}] \le 50\epsilon^2/k$ *for all* i.

Proof. Recall that

$$\mu^{\text{new}} = \mu + \widetilde{\delta}_\mu + (x-\overline{x})\widetilde{\delta}_s + (s-\overline{s})\widetilde{\delta}_x + \widetilde{\delta}_x\widetilde{\delta}_s.$$

We can upper bound the variance of $\mu_i^{-1}\mu_i^{\text{new}}$,

$$\begin{aligned}
\mathbb{V}[\mu_i^{-1}\mu_i^{\text{new}}] &\le 4\,\mathbb{V}[\mu_i^{-1}\widetilde{\delta}_{\mu,i}] + 4\,\mathbb{V}[\mu_i^{-1}(x_i-\overline{x}_i)\widetilde{\delta}_{s,i}] + 4\,\mathbb{V}[\mu_i^{-1}(s_i-\overline{s}_i)\widetilde{\delta}_{x,i}] + 4\,\mathbb{V}[\mu_i^{-1}\widetilde{\delta}_{x,i}\widetilde{\delta}_{s,i}] \\
&\le 32\frac{\epsilon^2}{k} + 4\frac{\epsilon^2}{k} + 4\frac{\epsilon^2}{k} + \mathbb{V}[\mu_i^{-1}\widetilde{\delta}_{x,i}\widetilde{\delta}_{s,i}] \\
&= 40\frac{\epsilon^2}{k} + \mathbb{V}[x_i^{-1}\widetilde{\delta}_{x,i}\cdot s_i^{-1}\widetilde{\delta}_{s,i}] \\
&\le 40\frac{\epsilon^2}{k} + 2\,\mathbf{Sup}[(x_i^{-1}\widetilde{\delta}_{x,i})^2]\cdot\mathbb{V}[s_i^{-1}\widetilde{\delta}_{s,i}] + 2\,\mathbf{Sup}[(y_i^{-1}\widetilde{\delta}_{y,i})^2]\cdot\mathbb{V}[x_i^{-1}\widetilde{\delta}_{x,i}] \\
&\le 40\frac{\epsilon^2}{k} + 2\cdot(\frac{0.02}{\log n})^2\cdot\frac{\epsilon^2}{k} + 2\cdot(\frac{0.02}{\log n})^2\cdot\frac{\epsilon^2}{k} \le 50\frac{\epsilon^2}{k}.
\end{aligned}$$

where the second step follows by $\mathbb{V}[\mu_i^{-1}\widetilde{\delta}_{\mu,i}] \le 8\epsilon^2/k$ (Part 2 of Lemma 2.4.3),

$$\mathbb{V}[\mu_i^{-1}(x_i - \overline{x}_i)\widetilde{\delta}_{s,i}] = \mathbb{V}[x_i^{-1}(x_i - \overline{x}_i)s_i^{-1}\widetilde{\delta}_{s,i}] \le 2\epsilon_{mp}^2\,\mathbb{V}[s_i^{-1}\widetilde{\delta}_{s,i}] \le \epsilon^2/k.$$

and a similar inequality $\mathbb{V}[\mu_i^{-1}(s_i - \overline{s}_i)\widetilde{\delta}_{x,i}] \le \epsilon^2/k$, the third step follows by $\mu = xs$, the fourth step follows by $\mathbb{V}[xy] \le 2\,\mathbf{Sup}[x^2]\,\mathbb{V}[y] + 2\,\mathbf{Sup}[y^2]\,\mathbb{V}[x]$ (Lemma 2.6.1) with **Sup** denoting the deterministic maximum of the random variable, the fifth step follows by $\mathbb{V}[s_i^{-1}\widetilde{\delta}_{s,i}] \le 2\epsilon^2/k$ and $\mathbb{V}[x_i^{-1}\widetilde{\delta}_{x,i}] \le 2\epsilon^2/k$ (Part 2 of Lemma 2.4.3). □

Claim 2.4.10 (Part 3 of Lemma 2.4.7)**.** $\|\mu^{-1}(\mu^{\text{new}} - \mu)\|_\infty \le \frac{0.021}{\log n}$.

Proof. We again note that

$$\mu^{\text{new}} = \mu + \widetilde{\delta}_\mu + (x - \overline{x})\widetilde{\delta}_s + (s - \overline{s})\widetilde{\delta}_x + \widetilde{\delta}_x\widetilde{\delta}_s.$$

Hence, we have

$$\begin{aligned}
&|\mu_i^{-1}(\mu_i^{\text{new}} - \mu_i - \widetilde{\delta}_{\mu,i})| \\
&\le |(x - \overline{x})_i\mu_i^{-1}\widetilde{\delta}_{s,i}| + |(s - \overline{s})_i\mu_i^{-1}\widetilde{\delta}_{x,i}| + |\mu_i^{-1}\widetilde{\delta}_{x,i}\widetilde{\delta}_{s,i}| \\
&= |(x - \overline{x})_i x_i^{-1}| \cdot |s_i^{-1}\widetilde{\delta}_{s,i}| + |(s - \overline{s})_i s_i^{-1}| \cdot |x_i^{-1}\widetilde{\delta}_{x,i}| + |x_i^{-1}\widetilde{\delta}_{x,i}| \cdot |s_i^{-1}\widetilde{\delta}_{s,i}| \\
&\le \epsilon_{mp}|s_i^{-1}\widetilde{\delta}_{s,i}| + \epsilon_{mp}|x_i^{-1}\widetilde{\delta}_{x,i}| + |s_i^{-1}\widetilde{\delta}_{s,i}||x_i^{-1}\widetilde{\delta}_{x,i}| \\
&\le \epsilon_{mp} \cdot \frac{0.2}{\log n} + \epsilon_{mp} \cdot \frac{0.02}{\log n} + (\frac{0.02}{\log n})^2 \\
&\le \frac{1}{1000 \log n},
\end{aligned}$$

where the first step follows by triangle inequality, the second step follows by $\mu_i = x_i s_i$, the third step follows by $x \approx_{\epsilon_{mp}} \overline{x}$ and $s \approx_{\epsilon_{mp}} \overline{s}$, the fifth step follows by $|s_i^{-1}\widetilde{\delta}_{s,i}| \le \frac{0.02}{\log n}$ and $|x_i^{-1}\widetilde{\delta}_{x,i}| \le \frac{0.02}{\log n}$ (Part 3 of Lemma 2.4.3).

Since we know that $|\mu_i^{-1}\widetilde{\delta}_{\mu,i}| \leq \frac{0.02}{\log n}$ (Part 3 of Lemma 2.4.3), we have

$$|\mu_i^{-1}(\mu_i^{\text{new}} - \mu_i)| \leq \frac{1}{1000 \log n} + \frac{0.02}{\log n} \leq \frac{0.021}{\log n}.$$

□

2.4.3 Stochastic central path

Now, we are ready to prove $x_i s_i \approx_{0.1} t$ during the whole algorithm. As explained in the proof outline (see Section 2.4.1), we will prove this bound by analyzing the potential $\Phi_\lambda(\mu/t - 1)$ where $\Phi_\lambda(r) = \sum_{i=1}^n \cosh(\lambda r_i)$.

First, we give some basic properties of Φ_λ.

Lemma 2.4.11 (Basic properties of potential function)**.** *Let $\Phi_\lambda(r) = \sum_{i=1}^n \cosh(\lambda r_i)$ for some $\lambda > 0$. For any vector $r \in \mathbb{R}^n$,*

1. For any vector $\|v\|_\infty \leq 1/\lambda$, we have that

$$\Phi_\lambda(r+v) \leq \Phi_\lambda(r) + \langle \nabla \Phi_\lambda(r), v \rangle + 2\|v\|^2_{\nabla^2 \Phi_\lambda(r)}.$$

2. $\|\nabla \Phi_\lambda(r)\|_2 \geq \frac{\lambda}{\sqrt{n}}(\Phi_\lambda(r) - n)$.

3. $\left(\sum_{i=1}^n \lambda^2 \cosh^2(\lambda r_i)\right)^{1/2} \leq \lambda\sqrt{n} + \|\nabla \Phi_\lambda(r)\|_2$.

Proof. For each $i \in [n]$, we use r_i to denote the i-th coordinate of vector r.

Proof of Part 1. We have that

$$\cosh(\lambda(r_i + v_i)) = \cosh(\lambda r_i) + \lambda \sinh(\lambda r_i) v_i + \frac{\lambda^2}{2} \cosh(\zeta_i) v_i^2,$$

where ζ_i is between λr_i and $\lambda(r_i + v_i)$. By definition of cosh and the assumption that $\|v\|_\infty \leq \frac{1}{2\lambda}$, we have that

$$\cosh(\zeta_i) = \frac{1}{2}\exp(\zeta_i) + \frac{1}{2}\exp(-\zeta_i) \leq \exp(1) \cdot \frac{1}{2}(\exp(\lambda r_i) + \exp(-\lambda r_i)) \leq 3\cosh(\lambda r_i).$$

Hence, we have

$$\cosh(\lambda(r_i + v_i)) \leq \cosh(\lambda r_i) + \lambda \sinh(\lambda r_i) v_i + 2\lambda^2 \cosh(\lambda r_i) v_i^2.$$

Summing over all the coordinates gives

$$\begin{aligned}
&\sum_{i=1}^{n} \cosh(\lambda(r_i + v_i)) \leq \sum_{i=1}^{n} \left[\cosh(\lambda r_i) + 2\lambda \sinh(\lambda r_i) v_i + \lambda^2 \cosh(\lambda r_i) v_i^2\right] \\
\implies\ &\Phi_\lambda(r + v) \leq \Phi_\lambda(r) + \langle \nabla \Phi_\lambda(r), v \rangle + 2\|v\|^2_{\nabla^2 \Phi_\lambda(r)}.
\end{aligned}$$

Proof of Part 2. Since $\Phi_\lambda(r) = \sum_{i=1}^{n} \cosh(\lambda r_i)$, then

$$\nabla \Phi_\lambda(r) = \begin{bmatrix} \lambda \sinh(\lambda r_1) & \lambda \sinh(\lambda r_2) & \cdots & \lambda \sinh(\lambda r_n) \end{bmatrix}^\top.$$

Thus, we can lower bound $\|\nabla \Phi_\lambda(r)\|_2$ in the following way,

$$\begin{aligned}
\|\nabla \Phi_\lambda(r)\|_2 &= \left(\sum_{i=1}^{n} \lambda^2 \sinh^2(\lambda r_i)\right)^{1/2} \\
&= \left(\sum_{i=1}^{n} \lambda^2 (\cosh^2(\lambda r_i) - 1)\right)^{1/2} && \text{by } \cosh^2(y) - \sinh^2(y) = 1, \forall y \\
&\geq \frac{\lambda}{\sqrt{n}} \sum_{i=1}^{n} \sqrt{\cosh^2(\lambda r_i) - 1} && \text{by } \|\cdot\|_2 \geq \frac{1}{\sqrt{n}} \|\cdot\|_1 \\
&\geq \frac{\lambda}{\sqrt{n}} \sum_{i=1}^{n} (\cosh(\lambda r_i) - 1) && \text{by } \cosh(\lambda r_i) \geq 1 \\
&= \frac{\lambda}{\sqrt{n}} (\Phi_\lambda(r) - n). && \text{by def of } \Phi(r)
\end{aligned}$$

Proof of Part 3. We have

$$\begin{aligned}
\left(\sum_{i=1}^{n} \lambda^2 \cosh^2(\lambda r_i)\right)^{1/2} &= \left(\sum_{i=1}^{n} \lambda^2 + \lambda^2 \sinh^2(\lambda r_i)\right)^{1/2} \\
&\leq (n\lambda^2)^{1/2} + \left(\sum_{i=1}^{n} \lambda^2 \sinh^2(\lambda r_i)\right)^{1/2} \\
&= \lambda\sqrt{n} + \|\nabla \Phi_\lambda(r)\|_2.
\end{aligned}$$

where the first step follows from $\cosh^2(y) - \sinh^2(y) = 1, \forall y$. □

The following lemma shows that the potential Φ is decreasing in expectation when Φ is large.

Lemma 2.4.12. *Under the Assumption 2.4.1, we have*

$$\mathbb{E}\left[\Phi_\lambda\left(\frac{\mu^{\text{new}}}{t^{\text{new}}}-1\right)\right] \le \Phi_\lambda\left(\frac{\mu}{t}-1\right) - \frac{\lambda\epsilon}{15\sqrt{n}}\left(\Phi_\lambda\left(\frac{\mu}{t}-1\right)-10n\right).$$

Proof. Let $\epsilon_\mu = \mu^{\text{new}} - \mu - \widetilde{\delta}_\mu$. From the definition, we have

$$\mu^{\text{new}} - t^{\text{new}} = \mu + \widetilde{\delta}_\mu + \epsilon_\mu - t^{\text{new}},$$

which implies

$$\begin{aligned}
\frac{\mu^{\text{new}}}{t^{\text{new}}} - 1 &= \frac{\mu}{t^{\text{new}}} + \frac{1}{t^{\text{new}}}(\widetilde{\delta}_\mu + \epsilon_\mu) - 1 \\
&= \frac{\mu}{t}\frac{t}{t^{\text{new}}} + \frac{1}{t^{\text{new}}}(\widetilde{\delta}_\mu + \epsilon_\mu) - 1 \\
&= \frac{\mu}{t} + \frac{\mu}{t}(\frac{t}{t^{\text{new}}} - 1) + \frac{1}{t^{\text{new}}}(\widetilde{\delta}_\mu + \epsilon_\mu) - 1 \\
&= \frac{\mu}{t} - 1 + \underbrace{\frac{\mu}{t}(\frac{t}{t^{\text{new}}} - 1) + \frac{1}{t^{\text{new}}}(\widetilde{\delta}_\mu + \epsilon_\mu)}_{v}. \qquad (2.17)
\end{aligned}$$

To apply Lemma 2.4.11 with $r = \mu/t - 1$ and $r + v = \mu^{\text{new}}/t^{\text{new}} - 1$, we first compute the expectation of v

$$\begin{aligned}
\mathbb{E}[v] &= \frac{\mu}{t}(\frac{t}{t^{\text{new}}} - 1) + \frac{1}{t^{\text{new}}}(\mathbb{E}[\widetilde{\delta}_\mu] + \mathbb{E}[\epsilon_\mu]) \\
&= \frac{\mu}{t}(\frac{t}{t^{\text{new}}} - 1) + \frac{1}{t^{\text{new}}}(\delta_\mu + \mathbb{E}[\epsilon_\mu]) \\
&= \frac{\mu}{t}(\frac{t}{t^{\text{new}}} - 1) + \frac{1}{t^{\text{new}}}\left(\left((\frac{t^{\text{new}}}{t} - 1)\mu - \frac{\epsilon}{2}t^{\text{new}}\frac{\nabla\Phi_\lambda(\mu/t-1)}{\|\nabla\Phi_\lambda(\mu/t-1)\|_2}\right) + \mathbb{E}[\epsilon_\mu]\right) \\
&= -\frac{\epsilon}{2}\frac{\nabla\Phi_\lambda(\mu/t-1)}{\|\nabla\Phi_\lambda(\mu/t-1)\|_2} + \frac{1}{t^{\text{new}}}\mathbb{E}[\epsilon_\mu], \qquad (2.18)
\end{aligned}$$

where the third step follows by definition of δ_μ.

Next, we bound the $\|v\|_\infty$ as follows

$$\begin{aligned}\|v\|_\infty &\le \left\|\frac{\mu}{t}(\frac{t}{t^{\text{new}}}-1)\right\|_\infty + \left\|\frac{1}{t^{\text{new}}}(\widetilde{\delta}_\mu+\epsilon_\mu)\right\|_\infty \le \frac{\epsilon}{\sqrt{n}} + \frac{\|\mu^{-1}(\mu^{\text{new}}-\mu)\|_\infty}{0.9} \\ &\le \frac{\epsilon}{\sqrt{n}} + \frac{0.021}{0.9\log n} \le \frac{1}{\lambda}.\end{aligned}$$

where we used Part 3 of Lemma 2.4.7 and $\epsilon \le \frac{1}{400\log n}$.

Since $\|v\|_\infty \le \frac{1}{\lambda}$, we can apply Part 1 of Lemma 2.4.11 and get

$$\begin{aligned}&\mathbb{E}[\Phi_\lambda(\mu/t+v-1)] \\ &\le \Phi_\lambda(\mu/t-1) + \langle \nabla\Phi_\lambda(\mu/t-1), \mathbb{E}[v]\rangle + 2\,\mathbb{E}[\|v\|^2_{\nabla^2\Phi_\lambda(\mu/t+v-1)}] \\ &= \Phi_\lambda(\mu/t-1) - \frac{\epsilon}{2}\|\nabla\Phi_\lambda(\mu/t-1)\|_2 + \frac{t}{t^{\text{new}}}\langle \nabla\Phi_\lambda(\mu/t-1), \mathbb{E}[t^{-1}\epsilon_\mu]\rangle + 2\,\mathbb{E}[\|v\|^2_{\nabla^2\Phi_\lambda(\mu/t-1)}] \\ &\le \Phi_\lambda(\mu/t-1) - \frac{\epsilon}{2}\|\nabla\Phi_\lambda(\mu/t-1)\|_2 + \frac{t}{t^{\text{new}}}\|\nabla\Phi_\lambda(\mu/t-1)\|_2 \cdot \|\,\mathbb{E}[t^{-1}\epsilon_\mu]\|_2 + 2\,\mathbb{E}[\|[v]\|^2_{\nabla^2\Phi_\lambda(\mu/t-1)}] \\ &\le \Phi_\lambda(\mu/t-1) - \frac{\epsilon}{2}\|\nabla\Phi_\lambda(\mu/t-1)\|_2 + 10\epsilon_{mp}\cdot\epsilon\|\nabla\Phi_\lambda(\mu/t-1)\|_2 + 2\,\mathbb{E}[\|v\|^2_{\nabla^2\Phi_\lambda(\mu/t-1)}],\end{aligned}$$

where the second step follows by substituting $\mathbb{E}[v]$ by (2.18), the third step follows by $\langle a,b\rangle \le \|a\|_2\cdot\|b\|_2$, the fourth step follows by $\|\,\mathbb{E}[t^{-1}\epsilon_\mu]\|_2 \le 10\epsilon_{mp}\cdot\epsilon$ (from Part 1 of Lemma 2.4.7 and $\mu \approx_{0.1} t$).

We still need to bound $\mathbb{E}[\|v\|^2_{\nabla^2\Phi_\lambda(\mu/t-1)}]$. Before bounding it, we first bound $\mathbb{E}[v_i^2]$,

$$\begin{aligned}\mathbb{E}[v_i^2] &\le 2\,\mathbb{E}\left[\left(\frac{\mu_i}{t}(\frac{t}{t^{\text{new}}}-1)\right)^2\right] + 2\,\mathbb{E}\left[\left(\frac{1}{t^{\text{new}}}(\widetilde{\delta}_{\mu,i}+\widehat{\delta}_{\mu,i})\right)^2\right] \\ &\le \epsilon^2/n + 2.5\,\mathbb{E}\left[((\mu_i^{\text{new}}-\mu_i)/\mu_i)^2\right] \\ &= \epsilon^2/n + 2.5\,\mathbb{V}[(\mu_i^{\text{new}}-\mu_i)/\mu_i] + 2.5(\mathbb{E}[(\mu_i^{\text{new}}-\mu_i)/\mu_i])^2 \\ &\le \epsilon^2/n + 125\epsilon^2/k + 2.5(\mathbb{E}[(\mu_i^{\text{new}}-\mu_i)/\mu_i])^2 \\ &\le 126\epsilon^2/k + 3(\mathbb{E}[(\mu_i^{\text{new}}-\mu_i)/\mu_i])^2, \qquad (2.19)\end{aligned}$$

where the first step follows by definition of v (see (2.17)), the second step follows by $\mu \approx_{0.1} t$ and $(t/t^{\text{new}} - 1)^2 \leq \epsilon^2/(4n)$, the third step follows by $\mathbb{E}[x^2] = \mathbb{V}[x] + (\mathbb{E}[x])^2$, the fourth step follows by Part 2 of Lemma 2.4.7, and the last step follows by $n \geq k$.

Now, we are ready to bound $\mathbb{E}[\|v\|^2_{\nabla^2\Phi_\lambda(\mu/t-1)}]$

$$
\begin{aligned}
&\mathbb{E}[\|v\|^2_{\nabla^2\Phi_\lambda(\mu/t-1)}] \\
&= \lambda^2 \sum_{i=1}^n \mathbb{E}[\Phi_\lambda(\mu/t-1)_i v_i^2] \\
&\leq \lambda^2 \sum_{i=1}^n \Phi_\lambda(\mu/t-1)_i \cdot (126\epsilon^2/k + 3(\mathbb{E}[(\mu_i^{\text{new}} - \mu_i)/\mu_i])^2) \\
&= 126\frac{\lambda^2\epsilon^2}{k}\Phi_\lambda(\mu/t-1) + 3\lambda^2 \sum_{i=1}^n \Phi_\lambda(\mu/t-1)_i \cdot (\mathbb{E}[(\mu_i^{\text{new}} - \mu_i)/\mu_i])^2 \\
&\leq 126\frac{\lambda^2\epsilon^2}{k}\Phi_\lambda(\mu/t-1) + 3\lambda \left(\sum_{i=1}^n \lambda^2 \Phi_\lambda(\mu/t-1)_i^2\right)^{1/2} \cdot \|\mathbb{E}[\mu^{-1}(\mu^{\text{new}} - \mu)]\|_4^2 \\
&\leq 126\frac{\lambda^2\epsilon^2}{k}\Phi_\lambda(\mu/t-1) + 3\lambda \left(\lambda\sqrt{n} + \|\nabla\Phi_\lambda(\mu/t-1)\|_2\right) \cdot (5\epsilon)^2,
\end{aligned}
$$

where the first step follows by defining $\Phi_\lambda(x)_i = \cosh(\lambda x_i)$, the second step follows from (2.19), the fourth step follows from Cauchy-Schwarz inequality, the fifth step follows from Part 3 of Lemma 2.4.11 and the fact that $\|\mathbb{E}[\mu^{-1}(\mu^{\text{new}} - \mu)]\|_4^2 \leq \|\mathbb{E}[\mu^{-1}(\mu^{\text{new}} - \mu)]\|_2^2 \leq (5\epsilon)^2$ (Lemma 2.4.7).

Then,

$$\begin{aligned}
&\mathbb{E}[\Phi_\lambda(\mu/t + v - 1)] \\
&\leq \Phi_\lambda(\mu/t - 1) - (\frac{\epsilon}{2} - 10\epsilon_{mp} \cdot \epsilon)\|\nabla\Phi_\lambda(\mu/t - 1)\|_2 + 252\frac{\lambda^2\epsilon^2}{k}\Phi_\lambda(\mu/t - 1) \\
&\quad + 150\lambda^2\epsilon^2\sqrt{n} + 150\lambda\epsilon^2\|\Phi_\lambda(\mu/t - 1)\|_2 \\
&\leq \Phi_\lambda(\mu/t - 1) - \frac{\epsilon}{3}\|\nabla\Phi_\lambda(\mu/t - 1)\|_2 + 252\frac{\lambda^2\epsilon^2}{k}\Phi_\lambda(\mu/t - 1) + 150\lambda^2\epsilon^2\sqrt{n} \\
&\leq \Phi_\lambda(\mu/t - 1) - \frac{\lambda\epsilon}{3\sqrt{n}}(\Phi_\lambda(\mu/t - 1) - n) + 252\frac{\lambda^2\epsilon^2}{k}\Phi_\lambda(\mu/t - 1) + 150\lambda^2\epsilon^2\sqrt{n} \\
&\leq \Phi_\lambda(\mu/t - 1) - \frac{\lambda\epsilon}{3\sqrt{n}}(\Phi_\lambda(\mu/t - 1)/5 - 2n),
\end{aligned}$$

where the second step follows from $1000\lambda\epsilon \leq 1$ and $1000\epsilon_{mp} \leq 1$, the third step follows from Part 2 of Lemma 2.4.11, and the last step follows from $1000\lambda\epsilon_{mp} \leq \log n$ and $k \geq \frac{\sqrt{n}\epsilon\log n}{\epsilon_{mp}}$. □

As a corollary, we have the following:

Lemma 2.4.13. *During the* Main *algorithm, Assumption 2.4.1 is always satisfied. Furthermore, the* ClassicalStep *happens with probability* $O(\frac{1}{n^2})$ *each step.*

Proof. The second and the fourth assumptions simply follow from the choice of ϵ_{mp} and k.

Let $\Phi^{(k)}$ be the potential at the k-th iteration of the Main. The ClassicalStep ensures that $\Phi^{(k)} \leq n^3$ at the end of each iteration. By the definition of Φ and the choice of λ in Main, we have that

$$\left\|\frac{xs}{t} - 1\right\|_\infty \leq \frac{\ln(2n^3)}{\lambda} \leq 0.1.$$

This proves the first assumption $xs \approx_{0.1} t$ with $t > 0$.

For the third assumption, we note that

$$\begin{aligned}\|\delta_\mu\|_2 &= \left\| \left(\frac{t^{\text{new}}}{t} - 1 \right) xs - \frac{\epsilon}{2} \cdot t^{\text{new}} \cdot \frac{\nabla \Phi_\lambda(\mu/t - 1)}{\|\nabla \Phi_\lambda(\mu/t - 1)\|_2} \right\|_2 \\ &\leq \left| \frac{t^{\text{new}}}{t} - 1 \right| \|xs\|_2 + \frac{\epsilon}{2} t^{\text{new}} \\ &\leq \frac{\epsilon}{3\sqrt{n}} \cdot 1.1\sqrt{n}t + 1.01 \cdot \frac{\epsilon}{2} t \leq \epsilon t,\end{aligned}$$

where we used $xs \approx_{0.1} t$ and the formula of t^{new}. Hence, we proved all assumptions in Assumption 2.4.1.

Now, we bound the probability that ClassicalStep happens. In the beginning of the Main, Lemma 3.8.2 is used to modify the linear program with parameter $\min(\frac{\delta}{2}, \frac{1}{\lambda})$. Hence, the initial point x and s satisfies $xs \approx_{1/\lambda} 1$. Therefore, we have $\Phi^{(0)} \leq 10n$. Lemma 2.4.12 shows $\mathbb{E}[\Phi^{(k+1)}] \leq (1 - \frac{\lambda\epsilon}{15\sqrt{n}}) \mathbb{E}[\Phi^{(k)}] + \frac{\lambda\epsilon}{15\sqrt{n}} 10n$. By induction, we have that $\mathbb{E}[\Phi^{(k)}] \leq 10n$ for all k. Since the potential is positive, Markov inequality shows that for any k, $\Phi^{(k)} \geq n^3$ with probability at most $O(\frac{1}{n^2})$. □

2.4.4 Analysis of cost per iteration

To apply the data structure for projection maintenance (Theorem 2.5.1), we need to first prove the input vector w does not change too much for each step.

Lemma 2.4.14. *Let $x^{\text{new}} = x + \widetilde{\delta}_x$ and $s^{\text{new}} = s + \widetilde{\delta}_s$. Let $w = \frac{x}{s}$ and $w^{\text{new}} = \frac{x^{\text{new}}}{s^{\text{new}}}$. Then we have*

$$\sum_{i=1}^n \left(\frac{\mathbb{E}[w_i^{\text{new}}] - w_i}{w_i} \right)^2 \leq 64\epsilon^2, \sum_{i=1}^n \left(\mathbb{E}\left[\left(\frac{w_i^{\text{new}} - w_i}{w_i} \right)^2 \right] \right)^2 \leq 1000\epsilon^2, \left| \frac{w_i^{\text{new}} - w_i}{w_i} \right| \leq 0.1.$$

Proof. From the definition, we know that

$$\frac{w_i^{\text{new}}}{w_i} = \frac{1}{s_i^{-1} x_i} \frac{x_i + \widetilde{\delta}_{x,i}}{s_i + \widetilde{\delta}_{s,i}} = \frac{1 + x_i^{-1}\widetilde{\delta}_{x,i}}{1 + s_i^{-1}\widetilde{\delta}_{s,i}}.$$

Part 1. For each $i \in [n]$, we have

$$\begin{aligned}
\frac{\mathbb{E}[w_i^{\text{new}}]}{w_i} - 1 = & \ \mathbb{E}\left[\frac{1 + x_i^{-1}\widetilde{\delta}_{x,i}}{1 + s_i^{-1}\widetilde{\delta}_{s,i}} \right] - 1 \\
= & \ \mathbb{E}\left[\frac{x_i^{-1}\widetilde{\delta}_{x,i} - s_i^{-1}\widetilde{\delta}_{s,i}}{1 + s_i^{-1}\widetilde{\delta}_{s,i}} \right] \\
\leq & \ 2|\,\mathbb{E}[x_i^{-1}\widetilde{\delta}_{x,i} - s_i^{-1}\widetilde{\delta}_{s,i}]| \qquad \text{by } |s_i^{-1}\widetilde{\delta}_{s,i}| \leq 0.2, \quad \text{part 3 of Lemma 2.4.3} \\
\leq & \ 2|\,\mathbb{E}[x_i^{-1}\widetilde{\delta}_{x,i}]| + 2|\,\mathbb{E}[s_i^{-1}\widetilde{\delta}_{s,i}]|. \qquad \text{by triangle inequality}
\end{aligned}$$

Thus, summing over all the coordinates gives

$$\sum_{i=1}^n \left(\frac{\mathbb{E}[w_i^{\text{new}}] - w_i}{w_i} \right)^2 \leq \sum_{i=1}^n 8(\mathbb{E}[x_i^{-1}\widetilde{\delta}_{x,i}])^2 + 8(\mathbb{E}[s_i^{-1}\widetilde{\delta}_{s,i}])^2 \leq 64\epsilon^2.$$

where the first step follows by triangle inequality, the last step follows by $\|\,\mathbb{E}[s^{-1}\widetilde{\delta}_s]\|_2^2, \|\,\mathbb{E}[x^{-1}\widetilde{\delta}_x]\|_2^2 \leq 4\epsilon^2$ (Part 1 of Lemma 2.4.3).

Part 2. For each $i \in [n]$, we have

$$\begin{aligned}
\mathbb{E}\left[\left(\frac{w_i^{\text{new}}}{w_i} - 1\right)^2\right] &= \mathbb{E}\left[\left(\frac{x_i^{-1}\widetilde{\delta}_{x,i} - s_i^{-1}\widetilde{\delta}_{s,i}}{1 + s_i^{-1}\widetilde{\delta}_{s,i}}\right)^2\right] \\
&\le 2\,\mathbb{E}[(x_i^{-1}\widetilde{\delta}_{x,i} - s_i^{-1}\widetilde{\delta}_{s,i})^2] \\
&\le 2\,\mathbb{E}[2(x_i^{-1}\widetilde{\delta}_{x,i})^2 + 2(s_i^{-1}\widetilde{\delta}_{s,i})^2] \\
&= 4\,\mathbb{E}[(x_i^{-1}\widetilde{\delta}_{x,i})^2] + 4\,\mathbb{E}[(s_i^{-1}\widetilde{\delta}_{s,i})^2] \\
&= 4\,\mathbb{V}[x_i^{-1}\widetilde{\delta}_{x,i}] + 4(\mathbb{E}[x_i^{-1}\widetilde{\delta}_{x,i}])^2 + 4\,\mathbb{V}[s_i^{-1}\widetilde{\delta}_{s,i}] + 4(\mathbb{E}[s_i^{-1}\widetilde{\delta}_{s,i}])^2 \\
&\le 16\epsilon^2/k + 4(\mathbb{E}[x_i^{-1}\widetilde{\delta}_{x,i}])^2 + 4(\mathbb{E}[s_i^{-1}\widetilde{\delta}_{s,i}])^2,
\end{aligned}$$

where the last step follows by $\mathbb{V}[x_i^{-1}\widetilde{\delta}_{x,i}], \mathbb{V}[s_i^{-1}\widetilde{\delta}_{s,i}] \le 2\epsilon^2/k$ (Part 2 of Lemma 2.4.3).

Thus summing over all the coordinates

$$\begin{aligned}
\sum_{i=1}^{n}\left(\mathbb{E}\left[\left(\frac{w_i^{\text{new}}}{w_i} - 1\right)^2\right]\right)^2 &\le \frac{512n\epsilon^4}{k^2} + 64\sum_{i=1}^{n}\left((\mathbb{E}[x_i^{-1}\widetilde{\delta}_{x,i}])^4 + (\mathbb{E}[s_i^{-1}\widetilde{\delta}_{s,i}])^4\right) \\
&\le \frac{512n\epsilon^4}{k^2} + 2048\epsilon^4 \\
&\le 1000\epsilon^2,
\end{aligned}$$

where the last step follows by $\|\,\mathbb{E}[s^{-1}\widetilde{\delta}_s]\|_2^2, \|\,\mathbb{E}[x^{-1}\widetilde{\delta}_x]\|_2^2 \le 4\epsilon^2$ and $k \ge \sqrt{n}\epsilon$.

Part 3. For each $i \in [n]$

$$\left|\frac{w_i^{\text{new}}}{w_i} - 1\right| = \left|\frac{1 + x_i^{-1}\widetilde{\delta}_{x,i}}{1 + s_i^{-1}\widetilde{\delta}_{s,i}} - 1\right| \le \left|\frac{1 + 0.02}{1 - 0.02} - 1\right| \le 0.1.$$

where the second step follows by $|x_i^{-1}\widetilde{\delta}_{x,i}| \le 0.02$ and $|s_i^{-1}\widetilde{\delta}_{s,i}| \le 0.02$ (Part 3 of Lemma 2.4.3).

□

Now, we analyze the cost per iteration in procedure Main. This is a direct application of our projection maintenance result.

Lemma 2.4.15. *For $\epsilon \geq \frac{1}{\sqrt{n}}$, each iteration of* Main *(Algorithm 3.6) takes*

$$n^{1+a+o(1)} + \epsilon \cdot (n^{\omega-1/2+o(1)} + n^{2-a/2+o(1)})$$

expected time per iteration in amortized where $0 \leq a \leq \alpha$ controls the batch size in the data structure and α is the dual exponent of matrix multiplication.

Proof. Lemma 2.4.13 shows that ClassicalStep happens with only $O(1/n^2)$ probability each step. Since the cost of each step only takes $\widetilde{O}(n^{2.5})$, the expected cost is only $\widetilde{O}(n^{0.5})$.

Lemma 2.4.14 shows that the conditions in Theorem 2.5.1 holds with the parameter $C_1 = O(\epsilon), C_2 = O(\epsilon), \epsilon_{mp} = \Theta(1)$.

In the procedure StochasticStep, Theorem 2.5.1 shows that the amortized time per iteration is mainly dominated by two steps:

1. mp.Update(w): $O(\epsilon \cdot (n^{\omega-1/2+o(1)} + n^{2-a/2+o(1)}))$.
2. mp.Query($\frac{1}{\sqrt{XS}}\widetilde{\delta}_\mu$): $O(n \cdot \|\widetilde{\delta}_\mu\|_0 + n^{1+a+o(1)})$.

Combining both running time and using $\mathbb{E}[\|\widetilde{\delta}_\mu\|_0] = O(1+k) = O(\epsilon\sqrt{n}\log^2 n)$ (according to the probability of success in Claim 2.4.6 and matching Assumption 2.4.1), we have the result.

□

2.4.5 Main result

Proof of Theorem 2.2.1. In the beginning of the MAIN algorithm, Lemma 3.8.2 is called to modify the linear program. Then, we run the stochastic central path method on this modified linear program.

When the algorithm stops, we obtain a vector x and s such that $xs \approx_{0.1} t$ with $t \leq \frac{\delta^2}{2n}$. Hence, the duality gap is bounded by $\sum_i x_i s_i \leq \delta^2$. Lemma 3.8.2 shows how to obtain an approximate solution of the original linear program with the guarantee needed using the x and s we just found.

Since t is decreased by $1 - \frac{\epsilon}{3\sqrt{n}}$ factor each iteration, it takes $O(\frac{\sqrt{n}}{\epsilon} \cdot \log(\frac{n}{\delta}))$ iterations in total. In Lemma 2.4.15, we proved that each iteration takes

$$n^{1+a+o(1)} + \epsilon \cdot (n^{\omega-1/2+o(1)} + n^{2-a/2+o(1)}).$$

and hence the total runtime is

$$O(n^{2.5-a/2+o(1)} + n^{\omega+o(1)} + \frac{n^{1.5+a+o(1)}}{\epsilon}) \cdot \log(\frac{n}{\delta}).$$

Since $\epsilon = \Theta(\frac{1}{\log n})$, the total runtime is

$$O(n^{2.5-a/2+o(1)} + n^{\omega+o(1)} + n^{1.5+a+o(1)}) \cdot \log(\frac{n}{\delta}).$$

Finally, we note that the optimal choice of a is $\min(\frac{2}{3}, \alpha)$, which gives the promised runtime.

□

Using the same proof, but different choice of the parameters, we can analyze the ultra short step stochastic central path method, where each step involves sampling only polylogarithmic coordinates. As we mentioned before, the runtime is still around n^ω.

Corollary 2.4.16. *Under the same assumption as Theorem 2.2.1, if we choose* $\epsilon = \Theta(1/\sqrt{n})$ *and* $a = \min(\frac{1}{3}, \alpha)$*, the expected time of* Main *(Algorithm 3.6) is*

$$\left(n^{\omega+o(1)} + n^{2.5-\alpha/2+o(1)} + n^{2+1/3+o(1)}\right) \cdot \log(\frac{n}{\delta}).$$

2.5 Projection Maintenance

The goal of this section is to prove the following theorem:

Theorem 2.5.1 (Projection maintenance)**.** *Given a full rank matrix $A \in \mathbb{R}^{d\times n}$ with $n \geq d$ and a tolerance parameter $0 < \epsilon_{mp} < 1/4$. Given any positive number a such that $a \leq \alpha$ where α is the dual exponent of matrix multiplication. There is a deterministic data structure (Algorithm 2.3, 2.4) that approximately maintains the projection matrices*

$$\sqrt{W}A^\top (AWA^\top)^{-1}A\sqrt{W}$$

for positive diagonal matrices W through the following two operations:

1. UPDATE(w)*: Output a vector $\widetilde{v}$ such that for all i,*

$$(1-\epsilon_{mp})\widetilde{v_i} \leq w_i \leq (1+\epsilon_{mp})\widetilde{v_i}.$$

2. QUERY(h)*: Output $\sqrt{\widetilde{V}}A^\top (A\widetilde{V}A^\top)^{-1}A\sqrt{\widetilde{V}}h$ for the $\widetilde{v}$ outputted by the last call to* UPDATE.

The data structure takes $n^2 d^{\omega-2}$ time to initialize and each call of QUERY(h) *takes time*

$$n \cdot \|h\|_0 + n^{1+a+o(1)}.$$

Furthermore, if the initial vector $w^{(0)}$ and the (random) update sequence $w^{(1)}, \cdots, w^{(T)} \in \mathbb{R}^n$ satisfies

$$\sum_{i=1}^n \left(\frac{\mathbb{E}[w_i^{(k+1)}] - w_i^{(k)}}{w_i^{(k)}}\right)^2 \leq C_1^2, \sum_{i=1}^n \left(\mathbb{E}\left[\left(\frac{w_i^{(k+1)} - w_i^{(k)}}{w_i^{(k)}}\right)^2\right]\right)^2 \leq C_2^2, \left|\frac{w_i^{(k+1)} - w_i^{(k)}}{w_i^{(k)}}\right| \leq \frac{1}{4}.$$

Algorithm 2.3 Projection Maintenance Data Structure - Initial, Query

datastructure MAINTAINPROJECTION ▷ Theorem 2.5.1

members
- $w \in \mathbb{R}^n$ ▷ Target vector
- $v, \widetilde{v} \in \mathbb{R}^n$ ▷ Approximate vector
- $A \in \mathbb{R}^{d \times n}$
- $M \in \mathbb{R}^{n \times n}$ ▷ Approximate Projection Matrix
- $\epsilon_{mp} \in (0, 1/4)$ ▷ Tolerance
- $a \in (0, \alpha]$ ▷ Batch Size for Update (n^a)

end members

procedure INITIALIZE(A, w, ϵ_{mp}, a) ▷ Lemma 2.5.2
- $w \leftarrow w$, $v \leftarrow w$, $\epsilon_{mp} \leftarrow \epsilon_{mp}$, $A \leftarrow A$, $a \leftarrow a$
- $M \leftarrow A^\top (AVA^\top)^{-1} A$

end procedure

procedure QUERY(h) ▷ Lemma 2.5.4
- Let $\widetilde{S}$ be the indices i such that $(1 - \epsilon_{mp})v_i \leq w_i \leq (1 + \epsilon_{mp})v_i$ is false.
- **return** $\sqrt{\widetilde{V}} \cdot (M \cdot (\sqrt{\widetilde{V}} \cdot h)) - \sqrt{\widetilde{V}} \cdot (M_{\widetilde{S}} \cdot ((\widetilde{\Delta}^{-1}_{\widetilde{S},\widetilde{S}} + M_{\widetilde{S},\widetilde{S}})^{-1} \cdot (M^\top_{\widetilde{S}} \sqrt{\widetilde{V}} h)))$

end procedure

end datastructure

with the expectation is conditional on $w_i^{(k)}$ *for all* $k = 0, 1, \cdots, T-1$. *Then, the amortized expected time*[5] *per call of* UPDATE(w) *is*

$$(C_1/\epsilon_{mp} + C_2/\epsilon_{mp}^2) \cdot (n^{\omega - 1/2 + o(1)} + n^{2 - a/2 + o(1)}).$$

Remark 2.5.1. For our linear program algorithm, we have $C_1 = O(1/\log n)$, $C_2 = O(1/\log n)$ and $\epsilon_{mp} = \Theta(1)$. See Lemma 2.4.14.

[5] If the input is deterministic, so is the output and the runtime.

Algorithm 2.4 Projection Maintenance Data Structure - Update

datastructure MAINTAINPROJECTION ▷ Theorem 2.5.1
procedure UPDATE(w^{new}) ▷ Lemma 2.5.3
  $y_i \leftarrow w_i^{\text{new}}/v_i - 1$, $\forall i \in [n]$
  $r \leftarrow$ the number of indices i such that $|y_i| \geq \epsilon_{mp}$.
  if $r < n^a$ **then**
    $v^{\text{new}} \leftarrow v$
    $M^{\text{new}} \leftarrow M$
  else
    Let $\pi : [n] \to [n]$ be a sorting permutation such that $|y_{\pi(i)}| \geq |y_{\pi(i+1)}|$
    while $1.5 \cdot r < n$ and $|y_{\pi(\lceil 1.5 \cdot r \rceil)}| \geq (1 - 1/\log n)|y_{\pi(r)}|$ **do**
      $r \leftarrow \min(\lceil 1.5 \cdot r \rceil, n)$
    end while
    $v^{\text{new}}_{\pi(i)} \leftarrow \begin{cases} w^{\text{new}}_{\pi(i)} & i \in \{1, 2, \cdots, r\} \\ v_{\pi(i)} & i \in \{r+1, \cdots, n\} \end{cases}$

    ▷ Compute $M^{\text{new}} = A^\top (A V^{\text{new}} A^\top)^{-1} A$ via Matrix Woodbury
    $\Delta \leftarrow \text{diag}(v^{\text{new}} - v)$ ▷ $\Delta \in \mathbb{R}^{n \times n}$ and $\|\Delta\|_0 = r$
    Let $S \leftarrow \pi([r])$ be the first r indices in the permutation.
    Let $M_S \in \mathbb{R}^{n \times r}$ be the r columns from S of M.
    Let $M_{S,S}, \Delta_{S,S} \in \mathbb{R}^{r \times r}$ be the r rows and columns from S of M and Δ.
    $M^{\text{new}} \leftarrow M - M_S \cdot (\Delta^{-1}_{S,S} + M_{S,S})^{-1} \cdot (M_S)^\top$
  end if
  $w \leftarrow w^{\text{new}}$, $v \leftarrow v^{\text{new}}$, $M \leftarrow M^{\text{new}}$
  $\widetilde{v}_i \leftarrow \begin{cases} v_i & \text{if } (1 - \epsilon_{mp}) v_i \leq w_i \leq (1 + \epsilon_{mp}) v_i \\ w_i & \text{otherwise} \end{cases}$
  return $\widetilde{v}$
end procedure
end datastructure

2.5.1 Proof outline

For intuition, we consider the case $C_1 = \Theta(1)$, $C_2 = \Theta(1)$, and $\epsilon_{mp} = \Theta(1)$ in this explanation. The correctness of the data structure directly follows from Woodbury matrix identity. The amortized time analysis is based on a potential function that measures the

distance of the approximate vector v and the target vector w. We will show that

- The cost to update the projection M is proportional to the decrease of the potential.
- Each call to query increase the potential by a fixed amount.

Combining both together gives the amortized runtime bound of our data structure.

Now, we explain the definition of the potential. Consider the k-th round of the algorithm. For all $i \in [n]$, we define $x_i^{(k)} = \frac{w_i^{(k)}}{v_i^{(k)}} - 1$. Note that $|x_i^{(k)}|$ measures the relative distance between $w_i^{(k)}$ and $v_i^{(k)}$. Our algorithm fixes the indices with largest error $x_i^{(k)}$. To capture the fact that updating in a larger batch is more efficient, we define the potential as a weighted combination of the error where we put more weight to higher $x_i^{(k)}$. Formally, we sort the coordinates of $x^{(k)}$ such that $|x_i^{(k)}| \geq |x_{i+1}^{(k)}|$ and define the potential by

$$\Psi_k = \sum_{i=1}^{n} g_i \cdot \psi(x_i^{(k)}).$$

where g_i are positive decreasing numbers to be chosen and ψ is a symmetric ($\psi(x) = \psi(-x)$) positive function that increases on both sides. For intuition, one can think $\psi(x)$ behaves roughly like $|x|$.

Each iteration we update the projection matrix such that the error of $|x_1|, \cdots, |x_r|$ drops from roughly ϵ_{mp} to 0. This decreases the potential of $\psi(x_i^{(k)})$ by $\Omega(\epsilon_{mp})$ from $i = 1, \cdots, r$. Therefore, the whole potential decreases by $\Omega(\epsilon_{mp} \sum_{i=1}^{r} g_i)$. To make the term $\sum_{i=1}^{r} g_i$ proportional to the time to update a rank r part of the projection matrix, we set

$$g_i = \begin{cases} n^{-a}, & \text{if } i < n^a; \\ i^{\frac{\omega-2}{1-a}-1} n^{-\frac{a(\omega-2)}{1-a}}, & \text{otherwise.} \end{cases} \tag{2.20}$$

where ω is the exponent of matrix multiplication and a is any positive number less than or equals to the dual exponent of matrix multiplication. Lemma 2.6.3 shows that g is indeed non-increasing and Lemma 2.5.3 shows that the update time of data-structure is indeed $O(rg_r n^{2+o(1)}) = O(\sum_{i=1}^{r} g_i n^{2+o(1)})$ for any $r \geq n^a$.

Each call to UPDATE, the expectation of the error vector $x^{(k)}$ moves roughly in an unit ℓ_2 ball. Therefore, the changes of the potential is roughly upper bounded $(\sum_{i=1}^{n} g_i^2)^{1/2} \approx n^{\omega-5/2}$. Since it takes us $n^{2+o(1)}$ time to decrease the potential by roughly 1 in the update step, the total time is roughly $n^{\omega-1/2}$.

For the case of stochastic central path, we note that the variance of the vector x is quite small. By choosing a smooth potential function ψ (see (2.21)), we can essentially give the same result as if there is no variance.

2.5.2 Proof of Theorem 2.5.1

Now, we give the proof of Theorem 2.5.1. We will defer some simple calculations into later sections.

Proof of Theorem 2.5.1.

Proof of Correctness. The definition of $\widetilde{v}$ in Line 20 ensures that $(1-\epsilon_{mp})\widetilde{v}_i \leq w_i \leq (1+\epsilon_{mp})\widetilde{v}_i$.

Using the Matrix Woodbury formula, one can verify that the update rule in Line 17 correctly maintains $M = A^\top (AVA^\top)^{-1}A$. See the deviation of the formula in Lemma 2.5.2. By the same reasoning, the Line 19 outputs the vector $\sqrt{\widetilde{V}}A^\top(A\widetilde{V}A^\top)^{-1}A\sqrt{\widetilde{V}}h$. This completes the proof of correctness.

Definition of x and y. Consider the k-th round of the algorithm. For all $i \in [n]$, we define $x_i^{(k)}$, $x_i^{(k+1)}$ and $y_i^{(k)}$ as follows:

$$x_i^{(k)} = \frac{w_i^{(k)}}{v_i^{(k)}} - 1, y_i^{(k)} = \frac{w_i^{(k+1)}}{v_i^{(k)}} - 1, x_i^{(k+1)} = \frac{w_i^{(k+1)}}{v_i^{(k+1)}} - 1.$$

Note that the difference between $x_i^{(k)}$ and $y_i^{(k)}$ is that w is changing. The difference between $y_i^{(k)}$ and $x_i^{(k+1)}$ is that v is changing. For simplicity, we define $\beta_i = (\mathbb{E}[w_i^{(k+1)}] - w_i^{(k)})/w_i^{(k)}$, then one of assumption becomes $\sum_{i=1}^n \beta_i^2 \leq C_1^2$.

Assume sorting. Assume the coordinates of vector $x^{(k)} \in \mathbb{R}^n$ are sorted such that $|x_i^{(k)}| \geq |x_{i+1}^{(k)}|$. Let τ and π are permutations such that $|x_{\tau(i)}^{(k+1)}| \geq |x_{\tau(i+1)}^{(k+1)}|$ and $|y_{\pi(i)}^{(k)}| \geq |y_{\pi(i+1)}^{(k)}|$.

Definition of Potential function. Let g be defined in (2.20). Let $\psi : \mathbb{R} \to \mathbb{R}$ be defined by

$$\psi(x) = \begin{cases} \frac{|x|^2}{2\epsilon_{mp}}, & |x| \in [0, \epsilon_{mp}] \\ \epsilon_{mp} - \frac{(2\epsilon_{mp} - |x|)^2}{2\epsilon_{mp}}, & |x| \in (\epsilon_{mp}, 2\epsilon_{mp}] \\ \epsilon_{mp}. & |x| \in (2\epsilon_{mp}, +\infty) \end{cases} \tag{2.21}$$

We define the potential at the k-th round by

$$\Psi_k = \sum_{i=1}^{n} g_i \cdot \psi(x^{(k)}_{\tau_k(i)}).$$

where $\tau_k(i)$ is the permutation such that $|x^{(k)}_{\tau_k(i)}| \geq |x^{(k)}_{\tau_k(i+1)}|$.

Bounding the potential.

We can express $\Psi_{k+1} - \Psi_k$ as follows:

$$\begin{aligned} \Psi_{k+1} - \Psi_k &= \sum_{i=1}^{n} g_i \cdot \left(\psi(x^{(k+1)}_{\tau(i)}) - \psi(x^{(k)}_i) \right) \\ &= \sum_{i=1}^{n} g_i \cdot \underbrace{\left(\psi(y^{(k)}_{\pi(i)}) - \psi(x^{(k)}_i) \right)}_{w \text{ move}} - \sum_{i=1}^{n} g_i \cdot \underbrace{\left(\psi(y^{(k)}_{\pi(i)}) - \psi(x^{(k+1)}_{\tau(i)}) \right)}_{v \text{ move}}. \end{aligned} \tag{2.22}$$

Now, using Lemma 2.5.5 and 3.6.12, and the fact that $\Psi_0 = 0$ and $\Psi_T \geq 0$, with (2.22), we get

$$\begin{aligned} 0 \leq \Psi_T - \Psi_0 &= \sum_{k=0}^{T-1} (\Psi_{k+1} - \Psi_k) \\ &\leq \sum_{k=0}^{T-1} \left(O(C_1 + C_2/\epsilon_{mp}) \cdot \sqrt{\log n} \cdot (n^{-a/2} + n^{\omega - 5/2}) - \Omega(\epsilon_{mp} r_k g_{r_k} / \log n) \right) \\ &= T \cdot O(C_1 + C_2/\epsilon_{mp}) \cdot \sqrt{\log n} \cdot (n^{-a/2} + n^{\omega - 5/2}) - \sum_{k=1}^{T} \Omega(\epsilon_{mp} r_k g_{r_k} / \log n), \end{aligned}$$

where the third step follows by Lemma 2.5.5 and Lemma 3.6.12 and r_k is the number of coordinates we update during that iteration.

Therefore, we get,

$$\sum_{k=1}^{T} r_k g_{r_k} = O\left(T \cdot (C_1/\epsilon_{mp} + C_2/\epsilon_{mp}^2) \cdot \log^{3/2} n \cdot (n^{\omega-5/2} + n^{-a/2})\right).$$

Proof of running time. See the Section 2.5.3. □

2.5.3 Initialization time, update time, query time

To formalize the amortized runtime proof, we first analyze the initialization time (Lemma 2.5.2), update time (Lemma 2.5.3), and query time (Lemma 2.5.4) of our projection maintenance data-structure.

Lemma 2.5.2 (Initialization time)**.** *The initialization time of data-structure* MAINTAINPROJECTION *(Algorithm 2.3) is* $O(n^2 d^{\omega-2})$.

Proof. Given matrix $A \in \mathbb{R}^{d\times n}$ and diagonal matrix $V \in \mathbb{R}^{n\times n}$, computing $A^\top (AVA^\top)^{-1}A$ takes $O(n^2 d^{\omega-2})$. □

Lemma 2.5.3 (Update time)**.** *The update time of data-structure* MAINTAINPROJECTION *(Algorithm 2.4) is* $O(rg_r n^{2+o(1)})$ *where* r *is the number of indices we updated in* v.

Proof. Let $A_S \in \mathbb{R}^{d\times r}$ be the r columns from S of A. From k-th query to $(k+1)$-th query, we have

$$\begin{aligned}
& A^\top (AV^{(k+1)}A^\top)^{-1}A \\
= {} & A^\top (A(V^{(k)}+\Delta)A^\top)^{-1}A \\
= {} & A^\top \left((AV^{(k)}A^\top)^{-1} - (AV^{(k)}A^\top)^{-1}A_S(\Delta_{S,S}^{-1} + A_S^\top (AV^{(k)}A^\top)^{-1}A_S)^{-1}A_S^\top (AV^{(k)}A^\top)^{-1}\right) A \\
= {} & A^\top (AV^{(k)}A^\top)^{-1}A - A^\top (AV^{(k)}A^\top)^{-1}A_S(\Delta_{S,S}^{-1} + A_S^\top (AV^{(k)}A^\top)^{-1}A_S)^{-1}A_S^\top (AV^{(k)}A^\top)^{-1}A \\
= {} & M^{(k)} - M_S^{(k)}(\Delta_{S,S}^{-1} + M_{S,S}^{(k)})^{-1}(M_S^{(k)})^\top,
\end{aligned}$$

where the second step follows by Matrix Woodbury Identity and the last step follows by definition of $M^{(k)} \in \mathbb{R}^{n\times n}$.

Thus the update rule of matrix $M^{(k+1)} \in \mathbb{R}^{n \times n}$ can be written as

$$M^{(k+1)} = M^{(k)} - M_S^{(k)} (\Delta_{S,S}^{-1} + (M^{(k)})_{S,S})^{-1} (M_S^{(k)})^\top.$$

The updates in round k can be splitted into four parts:

1. Adding two $r \times r$ matrices takes $O(r^2)$ time.
2. Computing the inverse of an $r \times r$ matrix takes $O(r^{\omega+o(1)})$ time.
3. Computing the matrix multiplication of a $n \times r$ and $r \times n$ matrix takes $O(rg_r \cdot n^{2+o(1)})$ time where we used that $r \geq n^a$ (Lemma 2.2.3).
4. Adding two $n \times n$ matrices together takes $O(n^2)$ time.

Hence, the total cost is

$$O(r^2 + r^{\omega+o(1)} + rg_r \cdot n^{2+o(1)} + n^2) = O(r^2 + r^{\omega+o(1)} + rg_r \cdot n^{2+o(1)}) = O(rg_r \cdot n^{2+o(1)}).$$

where the first step follows by $rg_r \geq 1$ for all $r \geq n^a$ and the last step follows by the calculations. □

Lemma 2.5.4 (Query time)**.** *The query time of data-structure* MaintainProjection *(Algorithm 2.3) is* $O(n \cdot \|h\|_0 + n^{1+a+o(1)})$.

Proof. Let $\widetilde{\Delta}$ satisfies $\widetilde{V} = V + \widetilde{\Delta}$. Let $\widetilde{S} \subset [n]$ denote the support of $\widetilde{\Delta}$ and then $|\widetilde{S}| \leq n^a$. Let $\widetilde{r}$ denote $|\widetilde{S}|$. We abuse the notation here, $\widetilde{\Delta}$ denotes both $n \times n$ diagonal matrix and a length n vector.

Using Matrix Woodbury Identity and definition of M, a same proof as Update time (Lemma 2.5.3) shows

$$A^\top (A\widetilde{V}A^\top)^{-1}A = M + M_{\widetilde{S}} \left(\widetilde{\Delta}^{-1}_{\widetilde{S},\widetilde{S}} + M_{\widetilde{S},\widetilde{S}}\right)^{-1} M^\top_{\widetilde{S}},$$

where $\widetilde{\Delta}_{\widetilde{S}\times\widetilde{S}}$ has size $\widetilde{r}\times\widetilde{r}$, $M_{\widetilde{S},\widetilde{S}}$ has size $\widetilde{r}\times\widetilde{r}$ and $M_{\widetilde{S}}$ has size $n\times\widetilde{r}$.

To compute $\sqrt{\widetilde{V}}A^\top(A\widetilde{V}A^\top)^{-1}A\sqrt{\widetilde{V}}h$, we just need to compute

$$\sqrt{\widetilde{V}}M\sqrt{\widetilde{V}}h + \sqrt{\widetilde{V}}M_{\widetilde{S}}(\widetilde{\Delta}^{-1}_{\widetilde{S},\widetilde{S}} + M_{\widetilde{S},\widetilde{S}})^{-1}M^\top_{\widetilde{S}}\sqrt{\widetilde{V}}h.$$

Note the running time of computing the first term of the above equation only takes $O(n\cdot\|h\|_0)$ time.

Next, we analyze the cost of computing the second term of the above equation. It contains several parts:

1. Computing $\widetilde{M^\top_{\widetilde{S}}}\cdot(\sqrt{\widetilde{V}}\cdot h)\in\mathbb{R}^{\widetilde{r}}$ takes $\widetilde{r}\|h\|_0$ time.
2. Computing $(\widetilde{\Delta}^{-1}_{\widetilde{S},\widetilde{S}} + M_{\widetilde{S},\widetilde{S}})^{-1}\in\mathbb{R}^{\widetilde{r}\times\widetilde{r}}$ that is the inverse of a $\widetilde{r}\times\widetilde{r}$ matrix takes $\widetilde{r}^{\omega+o(1)}$ time.
3. Computing matrix-vector multiplication between $\widetilde{r}\times\widetilde{r}$ matrix $((\widetilde{\Delta}^{-1}_{\widetilde{S},\widetilde{S}} + M_{\widetilde{S},\widetilde{S}})^{-1})$ and $\widetilde{r}\times 1$ vector $(\widetilde{M^\top_{\widetilde{S}}}\sqrt{\widetilde{V}}h)$ takes $O(\widetilde{r}^2)$ time.
4. Computing matrix-vector multiplication between $n\times\widetilde{r}$ matrix $(M_{\widetilde{S}})$ and $\widetilde{r}\times 1$ vector $((\widetilde{\Delta}^{-1}_{\widetilde{S},\widetilde{S}} + M_{\widetilde{S},\widetilde{S}})^{-1}M^\top_{\widetilde{S}}\sqrt{\widetilde{V}}h)$ takes $O(n\widetilde{r})$ time.
5. Computing the entry-wise product of two n vectors takes $O(n)$ time

Thus, overall the running time is

$$O(\widetilde{r}\|h\|_0 + \widetilde{r}^{\omega+o(1)} + \widetilde{r}^2 + n\widetilde{r} + n) = O(\widetilde{r}^{\omega+o(1)} + n\widetilde{r}) = O(n^{a\cdot\omega+o(1)} + n^{1+a}).$$

Finally, we note that $\omega \leq 3-\alpha \leq 3-a$ (Lemma 2.6.3) and hence $a\cdot\omega \leq a(3-a) \leq 1+a$. Therefore, the runtime is $n^{1+a+o(1)}$. □

2.5.4 Bounding w move

The goal of this section is to prove Lemma 2.5.5.

Lemma 2.5.5 (w move)**.** *We have*

$$\sum_{i=1}^{n} g_i \cdot \mathbb{E}\left[\psi(y^{(k)}_{\pi(i)}) - \psi(x^{(k)}_i)\right] \leq O(C_1 + C_2/\epsilon_{mp}) \cdot \sqrt{\log n} \cdot (n^{-a/2} + n^{\omega-5/2}).$$

Proof. Observe that since the errors $|x^{(k)}_i|$ are sorted in descending order, and $\psi(x)$ is symmetric and non-decreasing function for $x \geq 0$, thus $\psi(x^{(k)}_i)$ is also in decreasing order. In addition, note that g is decreasing, we have

$$\sum_{i=1}^{n} g_i \psi(x^{(k)}_{\pi(i)}) \leq \sum_{i=1}^{n} g_i \psi(x^{(k)}_i). \tag{2.23}$$

Hence the first term in (2.22) can be upper bounded as follows:

$$\begin{aligned}
&\mathbb{E}\left[\sum_{i=1}^{n} g_i \cdot \left(\psi(y^{(k)}_{\pi(i)}) - \psi(x^{(k)}_i)\right)\right] \\
\leq\ &\mathbb{E}\left[\sum_{i=1}^{n} g_i \cdot \left(\psi(y^{(k)}_{\pi(i)}) - \psi(x^{(k)}_{\pi(i)})\right)\right] && \text{by (2.23)} \\
=\ &\sum_{i=1}^{n} g_i \cdot \mathbb{E}[\psi(y^{(k)}_{\pi(i)}) - \psi(x^{(k)}_{\pi(i)})] \\
=\ &O(C_1 + C_2/\epsilon_{mp}) \cdot \sqrt{\log n} \cdot (n^{-a/2} + n^{\omega-5/2}). && \text{by Lemma 2.5.6}
\end{aligned}$$

Thus, we complete the proof of w move Lemma. □

It remains to prove the following Lemma,

Lemma 2.5.6.

$$\sum_{i=1}^{n} g_i \cdot \mathbb{E}[\psi(y^{(k)}_{\pi(i)}) - \psi(x^{(k)}_{\pi(i)})] = O(C_1 + C_2/\epsilon_{mp}) \cdot \sqrt{\log n} \cdot (n^{-a/2} + n^{\omega - 5/2}).$$

Proof. Let I be the set of indices such that $|x_i^{(k)}| \leq 1$. We separate the term into two:

$$\sum_{i=1}^{n} g_i \cdot \mathbb{E}[\psi(y^{(k)}_{\pi(i)}) - \psi(x^{(k)}_{\pi(i)})] = \sum_{i \in I} g_{\pi^{-1}(i)} \cdot \mathbb{E}[\psi(y_i^{(k)}) - \psi(x_i^{(k)})] + \sum_{i \in I^c} g_{\pi^{-1}(i)} \cdot \mathbb{E}[\psi(y_i^{(k)}) - \psi(x_i^{(k)})].$$

Case 1: Terms from I Mean value theorem shows that

$$\begin{aligned}
\psi(y_i^{(k)}) - \psi(x_i^{(k)}) &= \psi'(x_i^{(k)})(y_i^{(k)} - x_i^{(k)}) + \frac{1}{2}\psi''(\zeta)(y_i^{(k)} - x_i^{(k)})^2 \\
&\leq \psi'(x_i^{(k)}) \frac{w_i^{(k+1)} - w_i^{(k)}}{v_i^{(k)}} + \frac{L_2}{2} \left(\frac{w_i^{(k+1)} - w_i^{(k)}}{v_i^{(k)}} \right)^2,
\end{aligned}$$

where $L_2 = \max_x \psi''(x)$. Taking conditional expectation given $w^{(k)}$ on both sides

$$\begin{aligned}
\mathbb{E}[\psi(y_i^{(k)}) - \psi(x_i^{(k)})] &\leq \psi'(x_i^{(k)}) \cdot \frac{\mathbb{E}[w_i^{(k+1)}] - w_i^{(k)}}{v_i^{(k)}} + \frac{L_2}{2} \frac{1}{(v_i^{(k)})^2} \mathbb{E}[(w_i^{(k+1)} - w_i^{(k)})^2] \\
&= \psi'(x_i^{(k)}) \cdot \frac{w_i^{(k)}}{v_i^{(k)}} \beta_i + \frac{L_2}{2} \frac{(w_i^{(k)})^2}{(v_i^{(k)})^2} \gamma_i,
\end{aligned}$$

where $\beta_i = \frac{\mathbb{E}[w_i^{(k+1)}] - w_i^{(k)}}{w_i^{(k)}}$ and $\gamma_i = \mathbb{E}\left[\left(\frac{w_i^{(k+1)} - w_i^{(k)}}{w_i^{(k)}} \right)^2 \right]$.

To bound $\sum_{i \in I} g_{\pi^{-1}(i)} \mathbb{E}[\psi(y_i^{(k)}) - \psi(x_i^{(k)})]$, we need to bound the following two terms,

$$\sum_{i \in I} g_{\pi^{-1}(i)} \psi'(x_i^{(k)}) \frac{w_i^{(k)}}{v_i^{(k)}} \beta_i, \text{ and } \sum_{i \in I} g_{\pi^{-1}(i)} \frac{L_2}{2} \frac{(w_i^{(k)})^2}{(v_i^{(k)})^2} \gamma_i. \tag{2.24}$$

For the term $\frac{w_i^{(k)}}{v_i^{(k)}}$, we note that for $i \in I$, we have $\left| \frac{w_i^{(k)}}{v_i^{(k)}} \right| \leq \left| x_i^{(k)} \right| + 1 \leq 2$. Using this,

we can bound the first term by

$$\sum_{i\in I} g_{\pi^{-1}(i)}\psi'(x_i^{(k)})\frac{w_i^{(k)}}{v_i^{(k)}}\beta_i \le \left(\sum_{i\in I}\left(g_{\pi^{-1}(i)}\psi'(x_i^{(k)})\frac{w_i^{(k)}}{v_i^{(k)}}\right)^2\sum_{i\in I}\beta_i^2\right)^{1/2}$$
$$\le O(L_1)\left(\sum_{i=1}^n g_i^2\cdot C_1^2\right)^{1/2}$$
$$= O(C_1L_1\|g\|_2). \tag{2.25}$$

where $L_1 = \max_x|\psi'(x)|$, the first step follows by Cauchy-Schwarz inequality and the second step follows by $\left|\psi'(x_i^{(k)})\cdot w_i^{(k)}/v_i^{(k)}\right| \le 2L_1$ and $\sum_{i=1}^n\beta_i^2 \le C_1^2$.

For the second term, we have

$$\sum_{i\in I} g_{\pi^{-1}(i)}\frac{L_2}{2}\frac{(w_i^{(k)})^2}{(v_i^{(k)})^2}\gamma_i \le O(L_2)\cdot\sum_{i=1}^n g_i\cdot\gamma_i = O(C_2L_2\|g\|_2). \tag{2.26}$$

Now, combining (2.25) and (2.26) and using that $L_1 = O(1)$, $L_2 = O(1/\epsilon_{mp})$ (from part 4 of Lemma 2.5.9) and $\|g\|_2 \le \sqrt{\log n}\cdot O(n^{-a/2}+n^{\omega-5/2})$ (from Lemma 2.5.7), we have that

$$\sum_{i\in I} g_{\pi^{-1}(i)}\cdot\mathbb{E}[\psi(y_i^{(k)})-\psi(x_i^{(k)})] \le O(C_1+C_2/\epsilon_{mp})\cdot\sqrt{\log n}\cdot(n^{-a/2}+n^{\omega-5/2}).$$

Case 2: Terms from I^c

For all $i\in I^c$, we have $|x_i^{(k)}|\ge 1$. Note that $\psi(x)$ is a constant for $x\ge 2\epsilon_{mp}$ and that $\epsilon_{mp}\le 1/4$. Therefore, if $|y_i^{(k)}|\ge 1/2$, we have that $\psi(y_i^{(k)})-\psi(x_i^{(k)})=0$. Hence, we only need to consider the $i\in I^c$ such that $|y_i^{(k)}|<1/2$. For these i, we have that

$$\frac{1}{2} < |y_i^{(k)}-x_i^{(k)}| = \left|\frac{w_i^{(k+1)}-w_i^{(k)}}{v_i^{(k)}}\right| = \left|\frac{w_i^{(k+1)}}{v_i^{(k)}}\right|\left|\frac{w_i^{(k+1)}-w_i^{(k)}}{w_i^{(k+1)}}\right| \le \frac{3}{2}\left|\frac{w_i^{(k+1)}-w_i^{(k)}}{w_i^{(k+1)}}\right|,$$

where we used that $\left|y_i^{(k)}\right| = \left|\frac{w_i^{(k+1)}}{v_i^{(k)}} - 1\right| \le 1/2$. Hence, we have that $\left|\frac{w_i^{(k+1)} - w_i^{(k)}}{w_i^{(k+1)}}\right| > 1/3$ and hence $\left|\frac{w_i^{(k+1)} - w_i^{(k)}}{w_i^{(k)}}\right| > 1/4$, which is impossible.

Hence, we have

$$\sum_{i \in I^c} g_{\pi^{-1}(i)} \cdot \mathbb{E}[\psi(y_i^{(k)}) - \psi(x_i^{(k)})] = 0.$$

Combining both cases, we have the result. □

Lemma 2.5.7.

$$\left(\sum_{i=1}^{n} g_i^2\right)^{1/2} \le \sqrt{\log n} \cdot O(n^{-a/2} + n^{\omega - 5/2}).$$

Proof. Since function g behaves differently when $i \le n^a$ and $i > n^a$, we split the sum into two parts.

For the first part, we have

$$\sum_{i=1}^{n^a} g_i^2 = \sum_{i=1}^{n^a} n^{-2a} = n^{-a}.$$

For the second part, we have

$$\sum_{i=n^a}^{n} g_i^2 = \sum_{i=n^a}^{n} i^{\frac{2(\omega-2)}{1-a} - 2} n^{-\frac{2a(\omega-2)}{1-a}} = \sum_{i=n^a}^{n} \frac{1}{i} \cdot i^{\frac{2(\omega-2)}{1-a} - 1} n^{-\frac{2a(\omega-2)}{1-a}}.$$

Note that

$$\max_{i \in [n^a, n]} i^{\frac{2(\omega-2)}{1-a} - 1} n^{-\frac{2a(\omega-2)}{1-a}} = \max(n^{a\frac{2(\omega-2)}{1-a} - a} n^{-\frac{2a(\omega-2)}{1-a}}, n^{\frac{2(\omega-2)}{1-a} - 1} n^{-\frac{2a(\omega-2)}{1-a}}) = \max(n^{-a}, n^{2\omega-5}).$$

Thus, the second part is

$$\sum_{i=n^a}^{n} g_i^2 = \sum_{i=n^a}^{n} \frac{1}{i} \cdot \max(n^{-a}, n^{2\omega-5}) = O(\log n) \cdot \max(n^{-a}, n^{2\omega-5}).$$

Combining the first part and the second part completes the proof. □

2.5.5 Bounding v move

Lemma 2.5.8 (v move)**.** *We have,*

$$\sum_{i=1}^{n} g_i \cdot \left(\psi(y^{(k)}_{\pi(i)}) - \psi(x^{(k+1)}_{\tau(i)})\right) \geq \Omega(\epsilon_{mp} r_k g_{r_k} / \log n).$$

Proof. We split the proof into two cases.

We first understand some simple facts which are useful in the later proof. Note that from definition of $x^{(k+1)}_i$, we know that $x^{(k+1)}$ has r_k coordinates are 0. Basically, $\|y^{(k)} - x^{(k+1)}\|_0 = r_k$. The difference between those vectors is, for the largest r_k coordinates in $y^{(k)}$, we erase them in $x^{(k+1)}$. Then for each $i \in [n - r_k]$, $x^{(k+1)}_{\tau(i)} = y^{(k)}_{\pi(i+r_k)}$. For convenience, we define $y^{(k)}_{\pi(n+i)} = 0$, $\forall i \in [r_k]$.

Case 1. We exit the while loop when $1.5 r_k \geq n$.

Let u^* denote the largest u s.t. $|y^{(k)}_{\pi(u)}| \geq \epsilon_{mp}$. If $u^* = r_k$, we have that $|y^{(k)}_{\pi(r_k)}| \geq \epsilon_{mp} \geq \epsilon_{mp}/100$. Otherwise, the condition of the loop shows that

$$|y^{(k)}_{\pi(r_k)}| \geq (1 - 1/\log n)^{\log_{1.5} r_k - \log_{1.5} u^*} |y^{(k)}_{\pi(u^*)}| \geq (1 - 1/\log n)^{\log_{1.5} n} \epsilon_{mp} \geq \epsilon_{mp}/100.$$

where we used that $n \geq 4$.

According to definition of $x^{(k+1)}_{\tau(i)}$, we have

$$\begin{aligned}\sum_{i=1}^{n} g_i(\psi(y^{(k)}_{\pi(i)}) - \psi(x^{(k+1)}_{\tau(i)})) &= \sum_{i=1}^{n} g_i(\psi(y^{(k)}_{\pi(i)}) - \psi(y^{(k)}_{\pi(i+r_k)})) \geq \sum_{i=n/3+1}^{n} g_i(\psi(y^{(k)}_{\pi(i)}) - \psi(y^{(k)}_{\pi(i+r_k)})) \\ &\geq \sum_{i=n/3+1}^{n} g_i(\psi(y^{(k)}_{\pi(i)})) \geq \sum_{i=n/3+1}^{2n/3} g_i \psi(\epsilon_{mp}/100) \geq \Omega(r_k g_{r_k} \epsilon_{mp}),\end{aligned}$$

where the first step follows from $x^{(k+1)}_{\tau(i)} = y^{(k)}_{\pi(i+r_k)}$, the second step follows from $\psi(|x|)$ is non-decreasing (part 2 of Lemma 2.5.9) and $|y^{(k)}_{\pi(i)}|$ is non-increasing, the third step follows from $1.5r_k > n$ and hence $\psi(y^{(k)}_{\pi(i+r_k)}) = 0$ for $i \geq n/3 + 1$, the fourth step follows from ψ is non-decreasing and $|y^{(k)}_{\pi(i)}| \geq |y^{(k)}_{\pi(r_k)}| \geq \epsilon_{mp}/100$ for all $i < 2n/3$, and the last step follows by g is decreasing and part 3 of Lemma 2.5.9.

Case 2. We exit the while loop when $1.5r_k < n$ and $|y^{(k)}_{\pi(1.5r_k)}| < (1 - 1/\log n)|y^{(k)}_{\pi(r_k)}|$.

By the same argument as Case 1, we have that $|y^{(k)}_{\pi(r_k)}| \geq \epsilon_{mp}/100$. Part 3 of Lemma 2.5.9 together with the fact

$$|y^{(k)}_{\pi(1.5r)}| < \min(\epsilon_{mp}, |y^{(k)}_{\pi(r)}| \cdot (1 - 1/\log n)),$$

shows that

$$\psi(|y^{(k)}_{\pi(1.5r)}|) - \psi(|y^{(k)}_{\pi(r)}|) = \Omega(\epsilon_{mp}/\log n). \tag{2.27}$$

Putting it all together, we have

$$
\begin{aligned}
&\sum_{i=1}^{n} g_i \cdot (\psi(y^{(k)}_{\pi(i)}) - \psi(x^{(k+1)}_{\tau(i)})) \\
&= \sum_{i=1}^{n} g_i \cdot (\psi(y^{(k)}_{\pi(i)}) - \psi(y^{(k)}_{\pi(i+r_k)})) && \text{by } x^{(k+1)}_{\tau(i)} = y^{(k)}_{\pi(i+r_k)} \\
&\geq \sum_{i=r_k/2}^{r_k} g_i \cdot (\psi(y^{(k)}_{\pi(i)}) - \psi(y^{(k)}_{\pi(i+r_k)})) && \text{by } \psi(y^{(k)}_{\pi(i)}) - \psi(y^{(k)}_{\pi(i+r_k)}) \geq 0 \\
&\geq \sum_{i=r_k/2}^{r_k} g_i \cdot (\psi(y^{(k)}_{\pi(r_k)}) - \psi(y^{(k)}_{\pi(1.5r_k)})) \\
&\geq \sum_{i=r_k/2}^{r_k} g_i \cdot \Omega(\frac{\epsilon_{mp}}{\log n}) && \text{by (2.27)} \\
&\geq \sum_{i=r_k/2}^{r_k} g_{r_k} \cdot \Omega(\frac{\epsilon_{mp}}{\log n}) && \text{by } g_i \text{ is decreasing} \\
&= \Omega\left(\epsilon_{mp} r_k g_{r_k} / \log n\right),
\end{aligned}
$$

where the third step follows by $|y^{(k)}_{\pi(i)}|$ is decreasing and ψ is non-decreasing (from part 2 of Lemma 2.5.9). □

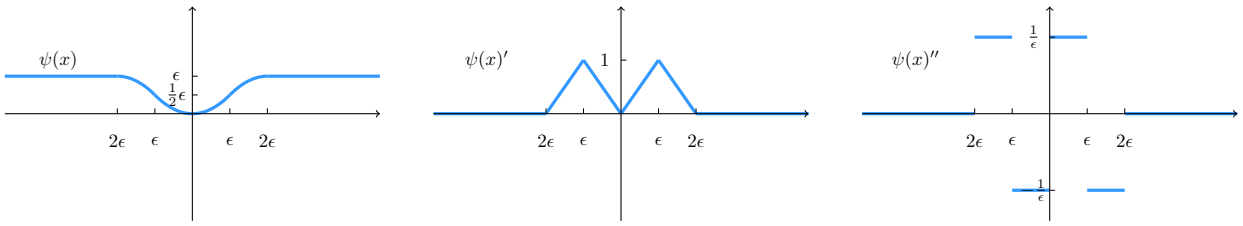


Figure 2.2: $\psi(x)$, $\psi(x)'$ and $\psi(x)''$. For $\epsilon_{mp} \in (0, 1)$.

2.5.6 Potential function ψ

Lemma 2.5.9 (Properties of function ψ)**.** *Let function ψ be defined in (2.21). Then function ψ satisfies the following properties:*

1. *Symmetric ($\psi(-x) = \psi(x)$) and $\psi(0) = 0$;*
2. *$\psi(|x|)$ is non-decreasing;*
3. *$|\psi'(x)| = \Omega(1), \forall |x| \in [0.01\epsilon_{mp}, \epsilon_{mp}]$;*
4. *$L_1 \stackrel{\text{def}}{=} \max_x \psi'(x) = 1$ and $L_2 \stackrel{\text{def}}{=} \max_x \psi''(x) = 1/\epsilon_{mp}$;*
5. *$\psi(x)$ is a constant for $|x| \geq 2\epsilon_{mp}$.*

Proof. We can see that

$$\psi(x)' = \begin{cases} \frac{|x|}{\epsilon_{mp}}, & |x| \in [0, \epsilon_{mp}] \\ \frac{2\epsilon_{mp} - |x|}{\epsilon_{mp}}, & |x| \in (\epsilon_{mp}, 2\epsilon_{mp}] \\ 0, & x \in (2\epsilon_{mp}, +\infty) \end{cases} \quad \text{and} \quad \psi(x)'' = \begin{cases} \frac{1}{\epsilon_{mp}}, & x \in [0, \epsilon_{mp}] \cup [-2\epsilon_{mp}, -\epsilon_{mp}] \\ -\frac{1}{\epsilon_{mp}}, & x \in (\epsilon_{mp}, 2\epsilon_{mp}] \cup [-\epsilon_{mp}, 0] \\ 0. & x \in (2\epsilon_{mp}, +\infty) \end{cases}$$

From the $\psi(x)'$ and $\psi(x)''$, it is not hard to see that ψ satisfies the properties needed. □

2.6 Omitted Proofs

Lemma 2.6.1. *Let x and y are (possibly dependent) random variables such that $|x| \le c_x$ and $|y| \le c_y$ almost surely. Then, we have*

$$\mathbb{V}[xy] \le 2c_x^2 \cdot \mathbb{V}[y] + 2c_y^2 \cdot \mathbb{V}[x].$$

Proof. Recall that $\mathbb{V}[xy] \le \mathbb{E}[(xy - t)^2]$ for any scalar t. Hence,

$$\begin{aligned}
\mathbb{V}[xy] &\le \mathbb{E}[(xy - \mathbb{E}[x]\,\mathbb{E}[y])^2] = \mathbb{E}[(xy - x\,\mathbb{E}[y] + x\,\mathbb{E}[y] - \mathbb{E}[x]\,\mathbb{E}[y])^2] \\
&\le 2\,\mathbb{E}[(xy - x\,\mathbb{E}[y])^2] + 2\,\mathbb{E}[(x\,\mathbb{E}[y] - \mathbb{E}[x]\,\mathbb{E}[y])^2] \\
&\le 2c_x^2 \cdot \mathbb{V}[y] + 2c_y^2 \cdot \mathbb{V}[x].
\end{aligned}$$

□

Lemma 2.6.2 ([Vai89b])**.** *Given a matrix $A \in \mathbb{R}^{d \times n}$, vectors $b \in \mathbb{R}^d, c \in \mathbb{R}^n$. Suppose $x, s, y \in \mathbb{R}^n$ satisfy that $xs \approx_{0.1} t$, $Ax = b$ and $A^\top y + s = c$ for some $t > 0$. For any $\epsilon \in (0, 1/2]$, in $\widetilde{O}(n^{2.5} \log(n/\epsilon))$ time, we can find vectors $x^{\mathrm{new}}, s^{\mathrm{new}} \in \mathbb{R}^n$ and $y^{\mathrm{new}} \in \mathbb{R}^d$ such that*

$$\begin{aligned}
\|x^{\mathrm{new}} s^{\mathrm{new}} - t\|_2 &\le \epsilon, \\
Ax^{\mathrm{new}} &= b, \\
A^\top y^{\mathrm{new}} + s &= c.
\end{aligned}$$

Remark 2.6.1. Instead of using this, one can also run our algorithm with $k = n$ for $O(\sqrt{n} \log n)$ iterations. Since $k = n$, there is no randomness involves and hence Φ will decrease deterministically to $O(n)$.

Lemma 2.6.3. $\omega \leq 3 - \alpha$.

Proof. We consider a $n \times n$ matrix A multiply another $n \times n$ B, we split A into $n^{1-\alpha}$ fat matrices where each of them has size $n^{\alpha} \times n$. Since ω is the best exponent of matrix multiplication, thus we know

$$n^{\omega+o(1)} \leq n^{1-\alpha} \cdot n^{2+o(1)},$$

which implies $\omega \leq 3 - \alpha$. □

Lemma 2.6.4 (Rectangular matrix multiplication)**.** *For any $n \geq r$, multiplying an $n \times r$ with an $r \times n$ matrix or $n \times n$ with $n \times r$ takes time*

$$n^{2+o(1)} + r^{\frac{\omega-2}{1-\alpha}} n^{2-\frac{\alpha(\omega-2)}{1-\alpha}+o(1)}.$$

Proof. The cost for multiplying a $n \times n$ and a $n \times r$ matrix is the same as multiplying a $n \times r$ and a $r \times n$ matrix. So, we focus on the later case.

For the case $r \leq n^{\alpha}$, it follows from the rectangular matrix multiplication result in [LGU18].

For the case $r \geq n^{\alpha}$, we let $k = (n/r)^{\frac{1}{1-\alpha}}$. We can view the problem as multiplying a $k \times k^{\alpha}$ and a $k^{\alpha} \times k$ block matrices and each block has size $\frac{n}{k} \times \frac{n}{k}$ size. Therefore, the total cost is

$$k^{2+o(1)} \times (\frac{n}{k})^{\omega+o(1)} = r^{\frac{\omega-2}{1-\alpha}} n^{2-\frac{\alpha(\omega-2)}{1-\alpha}+o(1)}.$$

□

For a matrix A, we define $\|A\|_1$ to be $\sum_{i,j} |A_{i,j}|$.

Lemma 2.6.5. *Consider a linear program* $\min_{Ax=b,x\geq 0} c^\top x$ *with* n *variables and* d *constraints. Assume that*

1. *Diameter of the polytope : For any* $x \geq 0$ *with* $Ax = b$*, we have that* $\|x\|_\infty \leq R$*.*
2. *Lipschitz constant of the linear program :* $\|c\|_\infty \leq L$*.*

For any $\delta \in (0, 1]$*, the modified linear program* $\min_{\overline{A}\overline{x}=\overline{b},\overline{x}\geq 0} \overline{c}^\top \overline{x}$ *with*

$$\overline{A} = \begin{bmatrix} A & 0 & \frac{1}{R}b - A1_n \\ 1_n^\top & 1 & 0 \end{bmatrix} \in \mathbb{R}^{(d+1)\times(n+2)}, \overline{b} = \begin{bmatrix} \frac{1}{R}b \\ n+1 \end{bmatrix} \in \mathbb{R}^{d+1} \text{ and, } \overline{c} = \begin{bmatrix} \frac{\delta}{L}\cdot c \\ 0 \\ 1 \end{bmatrix} \in \mathbb{R}^{n+2}$$

satisfies the following :

1. $\overline{x} = \begin{bmatrix} 1_n \\ 1 \\ 1 \end{bmatrix}, \overline{y} = \begin{bmatrix} 0_d \\ -1 \end{bmatrix}$ *and* $\overline{s} = \begin{bmatrix} 1_n + \frac{\delta}{L}\cdot c \\ 1 \\ 1 \end{bmatrix}$ *are feasible primal dual vectors.*
2. *For any feasible primal dual vectors* $(\overline{x}, \overline{y}, \overline{s})$ *with duality gap* $\leq \delta^2$*, consider the vector* $\widehat{x} = R \cdot \overline{x}_{1:n}$ *(*$\overline{x}_{1:n}$ *is the first* n *coordinates of* x*) is an approximate solution to the original linear program in the following sense*

$$\begin{aligned} c^\top \widehat{x} &\leq \min_{Ax=b,x\geq 0} c^\top x + LR\cdot\delta, \\ \|A\widehat{x} - b\|_1 &\leq 2n\delta\cdot(R\|A\|_1 + \|b\|_1), \\ \widehat{x} &\geq 0. \end{aligned}$$

Proof. For the first result, straightforward calculations show that $(\overline{x}, \overline{y}, \overline{s})$ are feasible, i.e.,

$$\overline{A}\overline{x} = \begin{bmatrix} A & 0 & \frac{1}{R}b - A1_n \\ 1_n^\top & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1_n \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{R}b \\ n+1 \end{bmatrix} = \overline{b}$$

and

$$\begin{aligned}
\overline{A}^\top \overline{y} + \overline{s} &= \begin{bmatrix} A^\top & 1_n \\ 0 & 1 \\ \frac{1}{R} b^\top - 1_n^\top A^\top & 0 \end{bmatrix} \cdot \begin{bmatrix} 0_d \\ -1 \end{bmatrix} + \begin{bmatrix} 1_n + \frac{\delta}{L} \cdot c \\ 1 \\ 1 \end{bmatrix} \\
&= \begin{bmatrix} -1_n \\ -1 \\ 0 \end{bmatrix} + \begin{bmatrix} 1_n + \frac{\delta}{L} \cdot c \\ 1 \\ 1 \end{bmatrix} \\
&= \begin{bmatrix} \frac{\delta}{L} \cdot c \\ 0 \\ 1 \end{bmatrix} \\
&= \overline{c}
\end{aligned}$$

For the second result, we let

$$\mathrm{OPT} = \min_{Ax=b, x \geq 0} c^\top x, \quad \text{and,} \quad \overline{\mathrm{OPT}} = \min_{\overline{A}\overline{x}=\overline{b}, \overline{x} \geq 0} \overline{c}^\top \overline{x}.$$

For any optimal x in the original LP, we consider the following $\overline{x}$

$$\overline{x} = \begin{bmatrix} \frac{1}{R} x \\ n + 1 - \frac{1}{R} \sum_i x_i \\ 0 \end{bmatrix} \tag{2.28}$$

We want to argue that $\overline{x}$ is feasible in the modified LP. It is obvious that $\overline{x} \geq 0$, it remains to show $\overline{A}\overline{x} = \overline{b}$. We have

$$\overline{A}\overline{x} = \begin{bmatrix} A & 0 & \frac{1}{R} b - A 1_n \\ 1_n^\top & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} \frac{1}{R} x \\ n + 1 - \frac{1}{R} \sum_i x_i \\ 0 \end{bmatrix} = \begin{bmatrix} \frac{1}{R} A x \\ n + 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{R} b \\ n + 1 \end{bmatrix} = \overline{b},$$

where the third step follows from $Ax = b$, and the last step follows from definition of $\overline{b}$.

Therefore, using the definition of $\overline{x}$ in (2.28) we have that

$$\overline{\mathrm{OPT}} \leq \overline{c}^\top \overline{x} = \begin{bmatrix} \frac{\delta}{L} \cdot c^\top & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \frac{1}{R} x \\ n + 1 - \frac{1}{R} \sum_i x_i \\ 0 \end{bmatrix} = \frac{\delta}{LR} \cdot c^\top x = \frac{\delta}{LR} \cdot \mathrm{OPT}\,. \tag{2.29}$$

where the first step follows from modified program is solving a minimization problem, the second step follows from definition of $\overline{c}$ and (2.28), the last step follows from x is an optimal solution in the original linear program.

Given a feasible $(\overline{x}, \overline{y}, \overline{s})$ with duality gap δ^2. Write $\overline{x} = \begin{bmatrix} \overline{x}_{1:n} \\ \tau \\ \theta \end{bmatrix}$ for some $\tau \geq 0$, $\theta \geq 0$. We can compute $\overline{c}^\top \overline{x}$ which is $\frac{\delta}{L} \cdot c^\top \overline{x}_{1:n} + \theta$. Then, we have

$$\frac{\delta}{L} \cdot c^\top \overline{x}_{1:n} + \theta \leq \overline{\mathrm{OPT}} + \delta^2 \leq \frac{\delta}{LR} \cdot \mathrm{OPT} + \delta^2, \tag{2.30}$$

where the first step follows from definition of duality gap, the last step follows from (2.29).

Hence, we can upper bound the OPT of the transformed program as follows:

$$c^\top \widehat{x} = R \cdot c^\top \overline{x}_{1:n} = \frac{RL}{\delta} \cdot \frac{\delta}{L} c^\top \overline{x}_{1:n} \leq \frac{RL}{\delta} (\frac{\delta}{LR} \cdot \mathrm{OPT} + \delta^2) = \mathrm{OPT} + LR \cdot \delta,$$

where the first step follows by $\widehat{x} = R \cdot \overline{x}_{1:n}$, the third step follows by (2.30).

We can upper bound the θ in the following sense,

$$\theta \leq \frac{\delta}{LR} \cdot \mathrm{OPT} + \delta^2 \leq n\delta + \delta^2 \leq 2n\delta$$

where the first step follows from (2.30) and $\frac{\delta}{L} c^\top \overline{x}_{1:n} \geq 0$, the second step follows by $\mathrm{OPT} = \min_{Ax=b, x \geq 0} c^\top x \leq nLR$ (because $\|c\|_\infty \leq L$ and $\|x\|_\infty \leq R$), and the last step follows from $\delta \leq 1 \leq n$.

The constraint in the new polytope shows that

$$A\overline{x}_{1:n} + (\frac{1}{R} b - A 1_n)\theta = \frac{1}{R} b.$$

Using $\widehat{x} = R x_{1:n}$, we have

$$A\frac{1}{R}\widehat{x} + (\frac{1}{R} b - A 1_n)\theta = \frac{1}{R} b.$$

Rewriting it, we have $A\widehat{x} - b = (RA1_n - b)\theta$ and hence

$$\begin{aligned}\|A\widehat{x} - b\|_1 = \|(RA1_n - b)\theta\|_1 &\leq \theta(\|RA1_n\|_1 + \|b\|_1) \leq \theta \cdot (R\|A\|_1 + \|b\|_1) \\ &\leq 2n\delta \cdot (R\|A\|_1 + \|b\|_1).\end{aligned}$$

□

Chapter 3

Empirical Risk Minimization

Many convex problems in machine learning and computer science share the same form:

$$\min_x \sum_i f_i(A_i x + b_i),$$

where f_i are convex functions on $\mathbb{R}^{n_i}$ with constant n_i, $A_i \in \mathbb{R}^{n_i \times d}$, $b_i \in \mathbb{R}^{n_i}$ and $\sum_i n_i = n$. This problem generalizes linear programming and includes many problems in empirical risk minimization.

In this chapter, we give an algorithm that runs in time

$$O^*((n^\omega + n^{2.5-\alpha/2} + n^{2+1/6}) \log(n/\delta))$$

where ω is the exponent of matrix multiplication, α is the dual exponent of matrix multiplication, and δ is the relative accuracy. Note that the runtime has only a log dependence on the condition numbers or other data dependent parameters and these are captured in δ. For the current bound $\omega \sim 2.38$ [Vassilevska Williams'12, Le Gall'14] and $\alpha \sim 0.31$ [Le Gall, Urrutia'18], our runtime $O^*(n^\omega \log(n/\delta))$ matches the current best for solving a dense least squares regression problem, a special case of the problem we consider. Very recently, [Alman'18] proved that all the current known techniques can not give a better ω below 2.168 which is larger than our $2 + 1/6$.

Our result generalizes the very recent result of solving linear programs in the current matrix multiplication time [Cohen, Lee, Song'19] to a more broad class of problems. Our algorithm proposes two concepts which are different from [Cohen, Lee, Song'19] :

• We give a robust deterministic central path method, whereas the previous one is a stochastic central path which updates weights by a random sparse vector.

• We propose an efficient data-structure to maintain the central path of interior point methods even when the weights update vector is dense.

3.1 Introduction

Empirical Risk Minimization (ERM) problem is a fundamental question in statistical machine learning. There are a huge number of papers that have considered this topic [Nes83, Vap92, PJ92, Nes04, BBM05, BB08, NJLS09, MB11, FGRW12, LRSB12, JZ13, Vap13, SSZ13, DB14, DBLJ14, FGKS15, DB16, SLC+17, ZYJ17, ZX17, ZWX+17, GSS17, MS17, NS17, AKK+17, Csi18, JLGJ18] as almost all convex optimization machine learning can be phrased in the ERM framework [SSBD14, Vap92]. While the statistical convergence properties and generalization bounds for ERM are well-understood, a general runtime bound for general ERM is not known although fast runtime bounds do exist for specific instances [AKPS19].

Examples of applications of ERM include linear regression, LASSO [Tib96], elastic net [ZH05], logistic regression [Cox58, HJLS13], support vector machines [CV95], ℓ_p regression [Cla05, DDH+09, BCLL18, AKPS19], quantile regression [Koe00, KH01, Koe05], AdaBoost [FS97], kernel regression [Nad64, Wat64], and mean-field variational inference [XJR02].

The classical Empirical Risk Minimization problem is defined as

$$\min_x \sum_{i=1}^{m} f_i(a_i^\top x + b_i)$$

where $f_i : \mathbb{R} \to \mathbb{R}$ is a convex function, $a_i \in \mathbb{R}^d$, and $b_i \in \mathbb{R}$, $\forall i \in [m]$. Note that this formulation also captures most standard forms of regularization as well.

Letting $y_i = a_i^\top x + b_i$, and $z_i = f_i(a_i^\top x + b_i)$ allows us to rewrite the original problem

in the following sense,

$$\begin{aligned} \min_{x,y,z} \quad & \sum_{i=1}^{m} z_i \\ \text{s.t.} \quad & Ax + b = y \\ & (y_i, z_i) \in K_i = \{(y_i, z_i) : f_i(y_i) \leq z_i\}, \forall i \in [m] \end{aligned} \tag{3.1}$$

We can consider a more general version where dimension of K_i can be arbitrary, e.g. n_i. Therefore, we come to study the general n-variable form

$$\min_{x \in \prod_{i=1}^{m} K_i, Ax=b} c^\top x$$

where $\sum_{i=1}^{m} n_i = n$. We state our main result for solving the general model.

Theorem 3.1.1 (Main result, informal version of Theorem 3.7.3)**.** *Given a matrix $A \in \mathbb{R}^{d \times n}$, two vectors $b \in \mathbb{R}^d$, $c \in \mathbb{R}^n$, and m compact convex sets $K_1, K_2, \cdots, K_m$. Assume that there is no redundant constraints and $n_i = O(1)$, $\forall i \in [m]$. There is an algorithm (procedure* Main *in Algorithm 3.6) that solves*

$$\min_{x \in \prod_{i=1}^{m} K_i, Ax=b} c^\top x$$

up to δ precision and runs in expected time

$$\widetilde{O}\left((n^{\omega+o(1)} + n^{2.5-\alpha/2+o(1)} + n^{2+1/6+o(1)}) \cdot \log(\frac{n}{\delta}) \right)$$

where ω is the exponent of matrix multiplication, α is the dual exponent of matrix multiplication.

For the current value of $\omega \sim 2.38$ [Wil12, LG14] and $\alpha \sim 0.31$ [LGU18], the expected time is simply $n^{\omega+o(1)}\widetilde{O}(\log(\frac{n}{\delta}))$.

Remark 3.1.1. More precisely, when n_i is super constant, our running time depends polynomially on $\max_{i \in [m]} n_i$ (but not exponential dependence).

Also note that our runtime depends on diameter, but logarithmically to the diameter. So, it can be applied to linear program by imposing an artificial bound on the solution.

3.1.1 Related Work

First-order algorithms for ERM are well-studied and a long series of accelerated stochastic gradient descent algorithms have been developed and optimized [Nes98, JZ13, XZ14, SSZ14, FGKS15, LMH15, MLF15, AY16, RHS+16, SS16, AH16, SLRB17, MS17, LMH17, LJCJ17, All17b, All17a, All18b, All18a]. However, these rates depend polynomially on the Lipschitz constant of ∇f_i and in order to achieve a $\log(1/\epsilon)$ dependence, the runtime will also have to depend on the strong convexity of the $\sum_i f_i$. In this chapter, we want to focus on algorithms that depend logarithmically on diameter/smoothness/strong convexity constants, as well as the error parameter ϵ. Note that gradient descent and a direct application of Newton's method do not belong to these class of algorithms, but for example, interior point method and ellipsoid method does.

Therefore, in order to achieve high-accuracy solutions for non-smooth and non strongly convex case, most convex optimization problems will rely on second-order methods, often under the general interior point method (IPM) or some sort of iterative refinement framework. So, we note that our algorithm is thus optimal in this general setting since second-order methods require at least n^ω runtime for general matrix inversion.

Our algorithm applies the interior point method framework to solve ERM. The most general interior point methods require $O(\sqrt{n})$-iterations of linear system solves [Nes98], requiring a naive runtime bound of $O(n^{\omega+1/2})$. Using the inverse maintenance technique [Vai89b, CLS19], one can improve the running time for LP to $O(n^\omega)$. This essentially implies that almost all convex optimization problems can be solved, up to subpolynomial factors, as fast as linear regression or matrix inversion!

The specific case of ℓ_2 regression can be solved in $O(n^\omega)$ time since the solution is explicitly given by solving a linear system. In the more general case of ℓ_p regression, [BCLL18] proposed a $\widetilde{O}_p(n^{|1/2-1/p|})$-iteration iterative solver with a naive $O(n^\omega)$ system solve at each step. Recently, [AKPS19] improved the runtime to $\widetilde{O}_p(n^{\max(\omega,7/3)})$, which is current matrix multiplication time as $\omega > 7/3$. However, both these results depend exponentially on p and fail to be impressive for large p. Otherwise, we are unaware of other ERM formulations that have have general runtime bounds for obtaining high-accuracy solutions.

Recently several works [AW18a, AW18b, Alm19] try to show the limitation of current known techniques for improving matrix multiplication time. Alman and Vassilevska Williams [AW18b] proved limitations of using the Galactic method applied to many tensors of interest (including Coppersmith-Winograd tensors [CW87]). More recently, Alman [Alm19] proved that by applying the Universal method on those tensors, we cannot hope to achieve any running time better than $n^{2.168}$ which is already above our $n^{2+1/6}$.

3.2 Overview of Techniques

In this section, we discuss the key ideas in this chapter. Generalizing the stochastic sparse update approach of [CLS19] to our setting is a natural first step to speeding up the matrix-vector multiplication that is needed in each iteration of the interior point method. In linear programs, maintaining approximate complementary slackness means that we maintain x, s to be close multiplicatively to the central path under some notion of distance. However, the generalized notion of complementary slackness requires a barrier-dependent notion of distance. Specifically, if $\phi(x)$ is a barrier function, then our distance is now defined as our function gradient being small in a norm depending on $\nabla^2\phi(x)$. One key fact of the stochastic sparse update is that the variance introduced does not perturb the approximation too much, which requires understanding the second derivative of the distance function. For our setting, this would require bounding the 4th derivative of $\phi(x)$, which may not exist for self-concordant functions. So, the stochastic approach may not work algorithmically (not just in the analysis) if $\phi(x)$ is assumed to be simply self-concordant. Even when assumptions on the 4th derivative of $\phi(x)$ are made, the analysis will become significantly more complicated due to the 4th derivative terms. To avoid these problems, the main contributions of this chapter is to 1) introduce a robust version of the central path and 2) exploit the robustness via sketching to apply the desired matrix-vector multiplication fast.

More generally, our main observation is that one can generally speed up an iterative method using sketching if the method is robust in a certain sense. To speed up interior point methods, in Section 3.4 and 3.5, we give a robust version of the interior point method; and in Section 3.6, we give a data structure to maintain the sketch; and in Section 3.7, we show how to combine them together. We provide several basic notations and definitions for

numerical linear algebra in Section 3.3. In Section 3.8, we provide some classical lemmas from the literature of interior point methods. In Section 3.9, we prove some basic properties of the sketching matrix. Now, we first begin with an overview of our robust central path and then proceed with an overview of sketching iterative methods.

3.2.1 Central Path Method

We consider the following optimization problem

$$\min_{x \in \prod_{i=1}^{m} K_i, Ax=b} c^\top x \tag{3.2}$$

where $\prod_{i=1}^{m} K_i$ is the direct product of m low-dimensional convex sets K_i. We let x_i be the i-th block of x corresponding to K_i. Interior point methods consider the path of solutions to the following optimization problem:

$$x(t) = \arg\min_{Ax=b} c^\top x + t \sum_{i=1}^{m} \phi_i(x_i) \tag{3.3}$$

where $\phi_i : K_i \to \mathbb{R}$ are self-concordant barrier functions. This parameterized path is commonly known as the *central path*. Many algorithms solve the original problem (3.2) by following the central path as the path parameter is decreased $t \to 0$. The rate at which we decrease t and subsequently the runtimes of these path-following algorithms are usually governed by the self-concordance properties of the barrier functions we use.

Definition 3.2.1. We call a function ϕ a ν self-concordant barrier for K if $\mathrm{dom}\phi = K$ and for any $x \in \mathrm{dom}\phi$ and for any $u \in \mathbb{R}^n$

$$|D^3\phi(x)[u,u,u]| \leq 2\|u\|_x^{3/2} \quad \text{and} \quad \|\nabla\phi(x)\|_x^* \leq \sqrt{\nu}$$

where $\|v\|_x := \|v\|_{\nabla^2\phi(x)}$ and $\|v\|_x^* := \|v\|_{\nabla^2\phi(x)^{-1}}$, for any vector v.

Remark 3.2.1. It is known that $\nu \geq 1$ for any self-concordant barrier function.

Nesterov and Nemirovsky showed that for any open convex set $K \subset \mathbb{R}^n$, there is a $O(n)$ self-concordant barrier function [Nes98]. In this chapter, the convex set K_i we

considered has $O(1)$ dimension. While Nesterov and Nemirovsky gave formulas for the universal barrier; in practice, most ERM problems lend themselves to explicit $O(1)$ self-concordant barriers for majority of the convex functions people use. For example, for the set $\{x : \|x\| < 1\}$, we use $-\log(1 - \|x\|^2)$; for the set $\{x : x > 0\}$, we use $-\log(x)$, and so on. That is the reason why we assume the gradient and hessian can be computed in $O(1)$ time. Therefore, in this chapter, we assume a ν_i self-concordant barrier ϕ_i is provided and that we can compute $\nabla\phi_i$ and $\nabla^2\phi_i$ in $O(1)$ time. The main result we will use about self-concordance is that the norm $\|\cdot\|_x$ is stable when we change x.

Theorem 3.2.1 (Theorem 4.1.6 in [Nes98]). *If ϕ is a self-concordant barrier and if $\|y-x\|_x < 1$, then we have :*

$$(1 - \|y - x\|_x)^2\nabla^2\phi(x) \preceq \nabla^2\phi(y) \preceq \frac{1}{(1 - \|y - x\|_x)^2}\nabla^2\phi(x).$$

In general, we can simply think of ϕ_i as a function penalizing any point $x_i \notin K_i$. It is known how to transform the original problem (3.2) by adding $O(n)$ many variables and constraints so that

- The minimizer $x(t)$ at $t = 1$ is explicitly given.
- One can obtain an approximate solution of the original problem using the minimizer at small t in linear time.

For completeness, we show how to do it in Lemma 3.8.2. Therefore, it suffices to study how we can move efficiently from $x(1)$ to $x(\epsilon)$ for some tiny ϵ where $x(t)$ is again the minimizer of the problem (3.3).

3.2.2 Robust Central Path

In the standard interior point method, we use a tight ℓ_2-bound to control how far we can deviate from $x(t)$ during the entirety of the algorithm. Specifically, if we denote $\gamma_i^t(x_i)$ as the appropriate measure of error (this will be specified later and is often called the Newton Decrement) in each block coordinate x_i at path parameter t, then as we let $t \to 0$, the old invariant that we are maintaining is,

$$\Phi_{\text{old}}^t(x) = \sum_{i=1}^{m} \gamma_i^t(x_i)^2 \leq O(1).$$

It can be shown that a Newton step in the standard direction will allow for us to maintain Φ_{old}^t to be small even as we decrease t by a multiplicative factor of $O(m^{-1/2})$ in each iteration, thereby giving a standard $O(\sqrt{m})$ iteration analysis. Therefore, the standard approach can be seen as trying to remain within a small ℓ_2 neighborhood of the central path by centering with Newton steps after making small decreases in the path parameter t. Note however that if each γ_i can be perturbed by an error that is $\Omega(m^{-1/2})$, $\Phi_{\text{old}}^t(x)$ can easily become too large for the potential argument to work.

To make our analysis more robust, we introduce a robust version that maintains the soft-max potential:

$$\Phi_{\text{new}}^t(x) = \sum_{i=1}^{m} \exp(\lambda \gamma_i^t(x_i)) \leq O(m)$$

for some $\lambda = \Theta(\log m)$. The robust central path is simply the region of all x that satisfies our potential inequality. We will specify the right constants later but we always make λ large enough to ensure that $\gamma_i \leq 1$ for all x in the robust central path. Now note that a ℓ_∞ perturbation of γ translates into a small multiplicative change in Φ^t, tolerating errors on each γ_i of up to $O(1/\text{poly}\log(n))$.

However, maintaining $\Phi^t_{\text{new}}(x) \leq O(m)$ is not obvious because the robust central path is a much wider region of x than the typical ℓ_2-neighborhood around the central path. We will show later how to modify the standard Newton direction to maintain $\Phi^t_{\text{new}}(x) \leq O(m)$ as we decrease t. Specifically, we will show that a variant of gradient descent of Φ^t_{new} in the Hessian norm suffices to provide the correct guarantees.

3.2.3 Speeding up via Sketching

To motivate our sketching algorithm, we consider an imaginary iterative method

$$z^{(k+1)} \leftarrow z^{(k)} + P \cdot F(z^{(k)})$$

where P is some dense matrix and $F(z)$ is some simple formula that can be computed efficiently in linear time. Note that the cost per iteration is dominated by multiplying P with a vector, which takes $O(n^2)$ time. To avoid the cost of multiplication, instead of storing the solution explicitly, we store it implicitly by $z^{(k)} = P \cdot u^{(k)}$. Now, the algorithm becomes

$$u^{(k+1)} \leftarrow u^{(k)} + F(P \cdot u^{(k)}).$$

This algorithm is as expensive as the previous one except that we switch the location of P. However, if we know the algorithm is robust under perturbation of the $z^{(k)}$ term in $F(z^{(k)})$, we can instead do

$$u^{(k+1)} \leftarrow u^{(k)} + F(R^\top RP \cdot u^{(k)})$$

for some random Gaussian matrix $R : \mathbb{R}^{b \times n}$. Note that the matrix RP is fixed throughout the whole algorithm and can be precomputed. Therefore, the cost of per iteration decreases from $O(n^2)$ to $O(nb)$.

For our problem, we need to make two adjustments. First, we need to sketch the change of z, that is $F(P \cdot u^{(k)})$, instead of $z^{(k)}$ directly because the change of z is smaller and this creates a smaller error. Second, we need to use a fresh random R every iteration to avoid the randomness dependence issue in the proof. For the imaginary iterative process, it

becomes

$$\overline{z}^{(k+1)} \leftarrow \overline{z}^{(k)} + R^{(k)\top} R^{(k)} P \cdot F(\overline{z}^{(k)}),$$
$$u^{(k+1)} \leftarrow u^{(k)} + F(\overline{z}^{(k)}).$$

After some iterations, $\overline{z}^{(k)}$ becomes too far from $z^{(k)}$ and hence we need to correct the error by setting $z^{(k)} = P \cdot u^{(k)}$, which zeros the error.

Note that the algorithm explicitly maintains the approximate vector $\overline{z}$ while implicitly maintaining the exact vector z by $Pu^{(k)}$. This is different from the classical way to sketch Newton method [PW16, PW17], which is to simply run $z^{(k+1)} \leftarrow z^{(k)} + R^\top RP \cdot F(z^{(k)})$ or use another way to subsample and approximate P. Such a scheme relies on the iteration method to fix the error accumulated in the sketch, while we are actively fixing the error by having both the approximate explicit vector $\overline{z}$ and the exact implicit vector z.

Without precomputation, the cost of computing $R^{(k)}P$ is in fact higher than that of $P \cdot F(z^{(k)})$. The first one involves multiplying multiple vectors with P and the second one involves multiplying 1 vector with P. However, we can precompute $[R^{(1)\top}; R^{(2)\top}; \cdots; R^{(T)\top}]^\top \cdot P$ by fast matrix multiplication. This decreases the cost of multiplying 1 vector with P to $n^{\omega-1}$ per vector. This is a huge saving from n^2. In our algorithm, we end up using only $\widetilde{O}(n)$ random vectors in total and hence the total cost is still roughly n^ω.

3.2.4 Maintaining the Sketch

The matrix P we use in interior point methods is of the form

$$P = \sqrt{W}A^\top (AWA^\top)^{-1} A\sqrt{W}$$

where W is some block diagonal matrix. [CLS19] showed one can approximately maintain the matrix P with total cost $\widetilde{O}(n^\omega)$ across all iterations of interior point method. However, the cost of applying the dense matrix P with a vector z is roughly $O(n\|z\|_0)$ which is $O(n^2)$ for dense vectors. Since interior point methods takes at least $\sqrt{n}$ iterations in general, this gives a total runtime of $O(n^{2.5})$. The key idea in [CLS19] is that one can design a stochastic interior point method such that each step only need to multiply P with a vector of density $\widetilde{O}(\sqrt{n})$. This bypasses the $n^{2.5}$ bottleneck.

In this chapter, we do not have this issue because we only need to compute RPz which is much cheaper than Pz. We summarize why it suffices to maintain RP throughout the algorithm. In general, for interior point method, the vector z is roughly an unit vector and since P is an orthogonal projection, we have $\|Pz\|_2 = O(1)$. One simple insight we have is that if we multiply a random $\sqrt{n} \times n$ matrix R with values $\pm\frac{1}{\sqrt{n}}$ by Pz, we have $\|RPz\|_\infty = \widetilde{O}(\frac{1}{\sqrt{n}})$ (Lemma 3.9.1). Since there are $\widetilde{O}(\sqrt{n})$ iterations in interior point method, the total error is roughly $\widetilde{O}(1)$ in a correctly reweighed ℓ_∞ norm. In Section 3.5, we showed that this is exactly what interior point method needs for convergence. Furthermore, we note that though each step needs to use a fresh random matrix R_l of size $\sqrt{n} \times n$, the random matrices $[R_1^\top; R_2^\top; \cdots; R_T^\top]^\top$ we need can all fit into $\widetilde{O}(n) \times n$ budget. Therefore, throughout the algorithm, we simply need to maintain the matrix $[R_1^\top; R_2^\top; \cdots; R_T^\top]^\top P$ which can be done with total cost $\widetilde{O}(n^\omega)$ across all iterations using idea similar to [CLS19].

The only reason the data structure looks complicated is that when the block matrix W changes in different location in $\sqrt{W}A^\top(AWA^\top)^{-1}A\sqrt{W}$, we need to update the matrix $[R_1; R_2; \cdots; R_T]P$ appropriately. This gives us few simple cases to handle in the algorithm and in the proof. For the intuition on how to maintain P under W change, see [CLS19, Section 2.2 and 5.1].

3.2.5 Fast rectangular matrix multiplication

Given two size $n \times n$ matrices, the time of multiplying them is $n^{2.81} < n^3$ by applying Strassen's original algorithm [Str69]. The current best running time takes n^ω time where $\omega < 2.373$ [Wil12, LG14]. One natural extension of multiplying two square matrices is multiplying two rectangular matrices. What is the running time of multiplying one $n \times n^a$ matrix with another $n^a \times n$ matrix? Let α denote the largest upper bound of a such that multiplying two rectangular matrices takes $n^{2+o(1)}$ time. The α is called the dual exponent of matrix multiplication, and the state-of-the-art result is $\alpha = 0.31$ [LGU18]. We use the similar idea as [CLS19] to delay the low-rank update when the rank is small.

3.3 Preliminaries

Given a vector $x \in \mathbb{R}^n$ and m compact convex sets $K_1 \subset \mathbb{R}^{n_1}, K_2 \subset \mathbb{R}^{n_2}, \cdots, K_m \subset \mathbb{R}^{n_m}$ with $\sum_{i=1}^m n_i = n$. We use x_i to denote the i-th block of x, then $x \in \prod_{i=1}^m K_i$ if $x_i \in K_i$, $\forall i \in [m]$.

We say a block diagonal matrix $A \in \oplus_{i=1}^m \mathbb{R}^{n_i \times n_i}$ if A can be written as

$$A = \begin{bmatrix} A_1 & & & \\ & A_2 & & \\ & & \ddots & \\ & & & A_m \end{bmatrix}$$

where $A_1 \in \mathbb{R}^{n_1 \times n_1}$, $A_2 \in \mathbb{R}^{n_2 \times n_2}$, and $A_m \in \mathbb{R}^{n_m \times n_m}$. For a matrix A, we use $\|A\|_F$ to denote its Frobenius norm and use $\|A\|$ to denote its operator norm. There are some trivial facts $\|AB\|_2 \leq \|A\|_2 \cdot \|B\|_2$ and $\|AB\|_F \leq \|A\|_F \cdot \|B\|_2$.

For notation convenience, we assume the number of variables $n \geq 10$ and there are no redundant constraints. In particular, this implies that the constraint matrix A is full rank.

For a positive integer n, let $[n]$ denote the set $\{1, 2, \cdots, n\}$.

For any function f, we define $\widetilde{O}(f)$ to be $f \cdot \log^{O(1)}(f)$. In addition to $O(\cdot)$ notation, for two functions f, g, we use the shorthand $f \lesssim g$ (resp. $\gtrsim$) to indicate that $f \leq Cg$ (resp. $\geq$) for some absolute constant C. For any function f, we use $\mathrm{dom} f$ to denote the domain of function f.

For a vector v, We denote $\|v\|$ as the standard Euclidean norm of v and for a symmetric PSD matrix A, we let $\|v\|_A = (v^\top A v)^{1/2}$. For a convex function $f(x)$ that is clear from context, we denote $\|v\|_x = \|v\|_{\nabla^2 f(x)}$ and $\|v\|_x^* = \|v\|_{\nabla^2 f(x)^{-1}}$.

3.4 Robust Central Path

In this section we show how to move move efficiently from $x(1)$ to $x(\epsilon)$ for some tiny ϵ by staying on a robust version of the central path. Because we are maintaining values that are slightly off-center, we show that our analysis still goes through despite ℓ_∞ perturbations on the order of $O(1/\text{poly}\log(n))$.

3.4.1 Newton Step

To follow the path $x(t)$, we consider the optimality condition of (3.3):

$$\begin{aligned} s/t + \nabla\phi(x) &= 0, \\ Ax &= b, \\ A^\top y + s &= c \end{aligned}$$

where $\nabla\phi(x) = (\nabla\phi_1(x_1), \nabla\phi_2(x_2), \cdots, \nabla\phi_m(x_m))$. To handle the error incurred in the progress, we consider the perturbed central path

$$\begin{aligned} s/t + \nabla\phi(x) &= \mu, \\ Ax &= b, \\ A^\top y + s &= c \end{aligned}$$

where μ represent the error between the original central path and our central path. Each iteration, we decrease t by a certain factor. It may increase the error term μ. Therefore, we need a step to decrease the norm of μ. The Newton method to move μ to $\mu + h$ is given by

$$\begin{aligned} \frac{1}{t} \cdot \delta_s^{\text{ideal}} + \nabla^2\phi(x) \cdot \delta_x^{\text{ideal}} &= h, \\ A\delta_x^{\text{ideal}} &= 0, \\ A^\top \delta_y^{\text{ideal}} + \delta_s^{\text{ideal}} &= 0 \end{aligned}$$

where $\nabla^2\phi(x)$ is a block diagonal matrix with the i-th block is given by $\nabla^2\phi_i(x_i)$. Letting $W = (\nabla^2\phi(x))^{-1}$, we can solve this:

$$\begin{aligned}
\delta_y^{\text{ideal}} &= -t \cdot \left(AWA^\top\right)^{-1} AWh, \\
\delta_s^{\text{ideal}} &= t \cdot A^\top \left(AWA^\top\right)^{-1} AWh, \\
\delta_x^{\text{ideal}} &= Wh - WA^\top \left(AWA^\top\right)^{-1} AWh.
\end{aligned}$$

We define projection matrix $P \in \mathbb{R}^{n\times n}$ as follows

$$P = W^{1/2}A^\top \left(AWA^\top\right)^{-1} AW^{1/2}$$

and then we rewrite them

$$\delta_x^{\text{ideal}} = W^{1/2}(I-P)W^{1/2}\delta_\mu, \tag{3.4}$$

$$\delta_s^{\text{ideal}} = tW^{-1/2}PW^{1/2}\delta_\mu. \tag{3.5}$$

One standard way to analyze the central path is to measure the error by $\|\mu\|_{\nabla^2\phi(x)^{-1}}$ and uses the step induced by $h = -\mu$. One can easily prove that if $\|\mu\|_{\nabla^2\phi(x)^{-1}} < \frac{1}{10}$, one step of Newton step decreases the norm by a constant factor. Therefore, one can alternatively decrease t and do a Newton step to follow the path.

3.4.2 Robust Central Path Method

In this section, we develop a central path method that is robust under certain ℓ_∞ perturbations. Due to the ℓ_∞ perturbation, we measure the error μ by a soft max instead of the ℓ_2 type potential:

Definition 3.4.1. For each $i \in [m]$, let $\mu_i^t(x,s) \in \mathbb{R}^{n_i}$ and $\gamma_i^t(x,s) \in \mathbb{R}$ be defined as follows:

$$\mu_i^t(x,s) = s_i/t + \nabla\phi_i(x_i), \tag{3.6}$$

$$\gamma_i^t(x,s) = \|\mu_i^t(x,s)\|_{\nabla^2\phi_i(x_i)^{-1}}, \tag{3.7}$$

and we define potential function Φ as follows:

$$\Phi^t(x,s) = \sum_{i=1}^{m} \exp(\lambda\gamma_i^t(x,s))$$

where $\lambda = O(\log m)$.

The *robust central path* is the region (x,s) that satisfies $\Phi^t(x,s) \leq O(m)$. To run our convergence argument, we will be setting λ appropriately so that staying on the robust central path will guarantee a ℓ_∞ bound on γ. Then, we will show how to maintain $\Phi^t(x,s)$ to be small throughout the algorithm while decreasing t, always staying on the robust central path. This is broken into a two step analysis: the progress step (decreasing t) and the centering step (moving x, s to decrease γ).

It is important to note that to follow the robust central path, we no longer pick the standard Newton direction by setting $h = -\mu$. To explain how we pick our centering step, suppose we can move $\mu \to \mu + h$ arbitrarily with the only restriction on the distance

$\|h\|_{\nabla^2\phi(x)^{-1}} = \alpha$. Then, the natural step would be

$$h = \arg \min_{\|h\|_{\nabla^2\phi(x)^{-1}}=\alpha} \langle \nabla f(\mu(x,s)), h\rangle$$

where $f(\mu) = \sum_{i=1}^m \exp(\lambda \|\mu\|_{\nabla^2\phi_i(x_i)^{-1}})$. Note that

$$\nabla f(\mu^t(x,s))_i = \lambda \exp(\lambda\gamma_i^t(x,s))/\gamma_i^t(x,s) \cdot \nabla^2\phi_i(x_i)^{-1}\mu_i^t(x,s).$$

Therefore, the solution for the minimization problem is

$$h_i^{\text{ideal}} = -\alpha \cdot c_i^t(x,s)^{\text{ideal}} \mu_i^t(x,s) \in \mathbb{R}^{n_i},$$

where $\mu_i^t(x,s) \in \mathbb{R}^{n_i}$ is defined as Eq. (3.6) and $c_i^t(x,s) \in \mathbb{R}$ is defined as

$$c_i^t(x,s)^{\text{ideal}} = \frac{\exp(\lambda\gamma_i^t(x,s))/\gamma_i^t(x,s)}{(\sum_{i=1}^m \exp(2\lambda\gamma_i^t(x,s)))^{1/2}}.$$

Eq. (3.4) and Eq. (3.5) gives the corresponding ideal step on x and s.

Now, we discuss the perturbed version of this algorithm. Instead of using the exact x and s in the formula of h, we use a $\overline{x}$ which is approximately close to x and a $\overline{s}$ which is close to s. Precisely, we have

$$h_i = -\alpha \cdot c_i^t(\overline{x},\overline{s})\mu_i^t(\overline{x},\overline{s}) \tag{3.8}$$

where

$$c_i^t(x,s) = \begin{cases} \frac{\exp(\lambda\gamma_i^t(x,s))/\gamma_i^t(x,s)}{(\sum_{i=1}^m \exp(2\lambda\gamma_i^t(x,s)))^{1/2}} & \text{if } \gamma_i^t(x,s) \geq 96\sqrt{\alpha} \\ 0 & \text{otherwise} \end{cases}. \tag{3.9}$$

Note that our definition of c_i^t ensures that $c_i^t(x,s) \leq \frac{1}{96\sqrt{\alpha}}$ regardless of the value of $\gamma_i^t(x,s)$. This makes sure we do not move too much in any coordinates and indeed when γ_i^t is small,

it is fine to set $c_i^t = 0$. Furthermore, for the formula on δ_x and δ_s, we use some matrix $\widetilde{V}$ that is close to $(\nabla^2\phi(x))^{-1}$. Precisely, we have

$$\delta_x = \widetilde{V}^{1/2}(I - \widetilde{P})\widetilde{V}^{1/2}h, \tag{3.10}$$

$$\delta_s = t \cdot \widetilde{V}^{-1/2}\widetilde{P}\ \widetilde{V}^{1/2}h. \tag{3.11}$$

where

$$\widetilde{P} = \widetilde{V}^{1/2}A^\top(A\widetilde{V}A^\top)^{-1}A\widetilde{V}^{1/2}.$$

Here we give a quick summary of our algorithm. (The more detailed of our algorithm can be found in Algorithm 3.5 and 3.6 in Section 3.7.)

- RobustIPM(A, b, c, ϕ, δ)
  - $\lambda = 2^{16}\log(m)$, $\alpha = 2^{-20}\lambda^{-2}$, $\kappa = 2^{-10}\alpha$.
  - $\delta = \min(\frac{1}{\lambda}, \delta)$.
  - $\nu = \sum_{i=1}^m \nu_i$ where ν_i are the self-concordant parameters of ϕ_i.
  - Modify the convex problem and obtain an initial x and s according to Lemma 3.8.2.
  - $t = 1$.
  - While $t > \frac{\delta^2}{4\nu}$
    - Find $\overline{x}$ and $\overline{s}$ such that $\|\overline{x}_i - x_i\|_{\overline{x}_i} < \alpha$ and $\|\overline{s}_i - s_i\|^*_{\overline{x}_i} < t\alpha$ for all i.
    - Find $\widetilde{V}_i$ such that $(1-\alpha)(\nabla^2\phi_i(\overline{x}_i))^{-1} \preceq \widetilde{V}_i \preceq (1+\alpha)(\nabla^2\phi_i(\overline{x}_i))^{-1}$ for all i.

* Compute $h = -\alpha \cdot c_i^t(\overline{x},\overline{s})\mu_i^t(\overline{x},\overline{s})$ where

$$c_i^t(\overline{x},\overline{s}) = \begin{cases} \frac{\exp(\lambda\gamma_i^t(\overline{x},\overline{s}))/\gamma_i^t(\overline{x},\overline{s})}{(\sum_{i=1}^m \exp(2\lambda\gamma_i^t(\overline{x},\overline{s})))^{1/2}} & \text{if } \gamma_i^t(\overline{x},\overline{s}) \geq 96\sqrt{\alpha} \\ 0 & \text{otherwise} \end{cases}.$$

  and $\mu_i^t(\overline{x},\overline{s}) = \overline{s}_i/t + \nabla\phi_i(\overline{x}_i)$ and $\gamma_i^t(\overline{x},\overline{s}) = \|\mu_i^t(\overline{x},\overline{s})\|_{\nabla^2\phi_i(\overline{x}_i)^{-1}}$

* Let $\widetilde{P} = \widetilde{V}^{1/2}A^\top(A\widetilde{V}A^\top)^{-1}A\widetilde{V}^{1/2}$.

* Compute $\delta_x = \widetilde{V}^{1/2}(I-\widetilde{P})\widetilde{V}^{1/2}h$ and $\delta_s = t\cdot\widetilde{V}^{-1/2}\widetilde{P}\,\widetilde{V}^{1/2}h$.

* Move $x \leftarrow x+\delta_x$, $s \leftarrow s+\delta_s$.

* $t^{\text{new}} = (1-\frac{\kappa}{\sqrt{\nu}})t$.

– Return an approximation solution of the convex problem according to Lemma 3.8.2.

Theorem 3.4.1 (Robust Interior Point Method)**.** *Consider a convex problem* $\min_{Ax=b, x\in\prod_{i=1}^m K_i} c^\top x$ *where* K_i *are compact convex sets. For each* $i\in[m]$*, we are given a* ν_i*-self concordant barrier function* ϕ_i *for* K_i*. Let* $\nu = \sum_{i=1}^m \nu_i$*. Also, we are given* $x^{(0)} = \arg\min_x \sum_{i=1}^m \phi_i(x_i)$*. Assume that*

1. *Diameter of the set: For any* $x\in\prod_{i=1}^m K_i$*, we have that* $\|x\|_2 \leq R$*.*

2. *Lipschitz constant of the program:* $\|c\|_2 \leq L$*.*

Then, the algorithm RobustIPM *finds a vector* x *such that*

$$\begin{aligned} c^\top x &\leq \min_{Ax=b, x\in\prod_{i=1}^m K_i} c^\top x + LR\cdot\delta, \\ \|Ax-b\|_1 &\leq 3\delta\cdot\left(R\sum_{i,j}|A_{i,j}| + \|b\|_1\right), \\ x &\in \prod_{i=1}^m K_i. \end{aligned}$$

in $O(\sqrt{\nu}\log^2 m \log(\frac{\nu}{\delta}))$ *iterations.*

Proof. Lemma 3.8.2 shows that the initial x and s satisfies

$$\|s + \nabla\phi(x)\|_x^* \leq \delta \leq \frac{1}{\lambda}$$

where the last inequality is due to our step $\delta \leftarrow \min(\frac{1}{\lambda}, \delta)$. This implies that $\gamma_i^1(x,s) = \|s_i + \nabla\phi_i(x_i)\|_{x_i}^* \leq \frac{1}{\lambda}$ and hence $\Phi^1(x,s) \leq e \cdot m \leq 80\frac{m}{\alpha}$ for the initial x and s. Apply Lemma 3.5.8 repetitively, we have that $\Phi^t(x,s) \leq 80\frac{m}{\alpha}$ during the whole algorithm. In particular, we have this at the end of the algorithm. This implies that

$$\|s_i + \nabla\phi_i(x_i)\|_{x_i}^* \leq \frac{\log(80\frac{m}{\alpha})}{\lambda} \leq 1$$

at the end. Therefore, we can apply Lemma 3.8.3 to show that

$$\langle c, x\rangle \leq \langle c, x^*\rangle + 4t\nu \leq \langle c, x^*\rangle + \delta^2$$

where we used the stop condition for t at the end. Note that this guarantee holds for the modified convex program. Since the error is δ^2, Lemma 3.8.2 shows how to get an approximate solution for the original convex program with error $LR \cdot \delta$.

The number of steps follows from the fact we decrease t by $1 - \frac{1}{\sqrt{\nu}\log^2 m}$ factor every iteration.

□

3.5 Robust Central Path

The goal of this section is to analyze robust central path. We provide an outline in Section 3.5.1. In Section 3.5.2, we bound the changes in μ and γ. In Section 3.5.3, we analyze the changes from (x, x, s) to $(x^{\text{new}}, x, s^{\text{new}})$. In Section 3.5.4, we analyze the changes from $(x^{\text{new}}, x, s^{\text{new}})$ to $(x^{\text{new}}, x^{\text{new}}, s^{\text{new}})$. We bound the changes in t in Section 3.5.5. Finally, we analyze entire changes of potential function in Section 3.5.6.

Statement	Section	Parameters
Lemma 3.5.2	Section 3.5.2	$\mu_i^t(x,s) \to \mu_i^t(x^{\text{new}}, s^{\text{new}})$
Lemma 3.5.4	Section 3.5.2	$\gamma_i^t(x,x,s) \to \gamma_i^t(x^{\text{new}}, x, s^{\text{new}})$
Lemma 3.5.5	Section 3.5.3	$\Phi(x,x,s) \to \Phi(x^{\text{new}}, x, s^{\text{new}})$
Lemma 3.5.6	Section 3.5.4	$\Phi(x^{\text{new}}, x, s^{\text{new}}) \to \Phi(x^{\text{new}}, x^{\text{new}}, s^{\text{new}})$
Lemma 3.5.7	Section 3.5.5	$\Phi^t \to \Phi^{t^{\text{new}}}$
Lemma 3.5.8	Section 3.5.6	$\Phi^t(x,s) \to \Phi^{t^{\text{new}}}(x^{\text{new}}, s^{\text{new}})$

Table 3.1: Bounding the changes of different variables

3.5.1 Outline of Analysis

Basically, the main proof is just a simple calculation on how $\Phi^t(x,s)$ changes during 1 iteration. It could be compared to the proof of ℓ_∞ potential reduction arguments for the convergence of long-step interior point methods, although the main difficulty arises from the perturbations from stepping using $\overline{x}, \overline{s}$ instead of x, s.

To organize the calculations, we note that the term $\gamma_i^t(x,s) = \|\mu_i^t(x,s)\|_{\nabla^2\phi_i(x_i)^{-1}}$ has two terms involving x, one in the μ term and one in the Hessian. Hence, we separate how different x affect the potential by defining

$$\gamma_i^t(x,z,s) = \|\mu_i^t(x,s)\|_{\nabla^2\phi_i(z_i)^{-1}},$$
$$\Phi^t(x,z,s) = \sum_{i=1}^m \exp(\lambda\gamma_i^t(x,z,s)).$$

One difference between our proof and standard ℓ_2 proofs of interior point is that we assume the barrier function is decomposable. We define $\alpha_i = \|\delta_{x,i}\|_{\overline{x}_i}$ is the "step" size of the coordinate i. One crucial fact we are using is that sum of squares of the step sizes is small.

Lemma 3.5.1. *Let α denote the parameter in* RobustIPM. *For all $i \in [m]$, let $\alpha_i = \|\delta_{x,i}\|_{\overline{x}_i}$.*

Then,

$$\sum_{i=1}^{m} \alpha_i^2 \leq 4\alpha^2.$$

Proof. Note that

$$\sum_{i=1}^{m} \alpha_i^2 = \|\delta_x\|_{\overline{x}}^2 = h^\top \widetilde{V}^{1/2}(I - \widetilde{P})\widetilde{V}^{1/2}\nabla^2\phi(\overline{x})\widetilde{V}^{1/2}(I - \widetilde{P})\widetilde{V}^{1/2}h.$$

Since $(1-\alpha)(\nabla^2\phi_i(\overline{x}_i))^{-1} \preceq \widetilde{V}_i \preceq (1+\alpha)(\nabla^2\phi_i(\overline{x}_i))^{-1}$, we have that

$$(1-\alpha)(\nabla^2\phi(\overline{x}))^{-1} \preceq \widetilde{V} \preceq (1+\alpha)(\nabla^2\phi(\overline{x}))^{-1}.$$

Using $\alpha \leq \frac{1}{10000}$, we have that

$$\sum_{i=1}^{m} \alpha_i^2 \leq 2h^\top \widetilde{V}^{1/2}(I - \widetilde{P})(I - \widetilde{P})\widetilde{V}^{1/2}h \leq 2h^\top \widetilde{V}h$$

where we used that $I - \widetilde{P}$ is an orthogonal projection at the end. Finally, we note that

$$\begin{aligned}
h^\top \widetilde{V}h &\leq 2\sum_{i=1}^{m} \|h_i\|_{\overline{x}_i}^{*2} \\
&= 2\alpha^2 \sum_{i=1}^{m} c_i^t(\overline{x},\overline{s})^2 \|\mu_i^t(\overline{x},\overline{s})\|_{\overline{x}_i}^{*2} \\
&\leq 2\alpha^2 \sum_{i=1}^{m} \frac{\exp(2\lambda\gamma_i^t(\overline{x},\overline{s}))/\gamma_i^t(\overline{x},\overline{s})^2}{\sum_{i=1}^m \exp(2\lambda\gamma_i^t(\overline{x},\overline{s}))^{1/2}} \|\mu_i^t(\overline{x},\overline{s})\|_{\overline{x}_i}^{*2} \\
&= 2\alpha^2 \frac{\sum_{i=1}^m \exp(2\lambda\gamma_i^t(\overline{x},\overline{s}))}{\sum_{i=1}^m \exp(2\lambda\gamma_i^t(\overline{x},\overline{s}))} \\
&= 2\alpha^2
\end{aligned}$$

where the second step follows from definition of h_i (3.8), the third step follows from definition c_i^t (3.9), the fourth step follows from definition of γ_i^t (3.7).

Therefore, putting it all together, we can show

$$\sum_{i=1}^{m} \alpha_i^2 \leq 4\alpha^2.$$

□

3.5.2 Changes in μ and γ

We provide basic lemmas that bound changes in μ, γ due to the centering steps.

Lemma 3.5.2 (Changes in μ)**.** *For all $i \in [m]$, let*

$$\mu_i^t(x^{\text{new}}, s^{\text{new}}) = \mu_i^t(x, s) + h_i + \epsilon_i^{(\mu)}.$$

Then, $\|\epsilon_i^{(\mu)}\|_{x_i}^* \leq 10\alpha \cdot \alpha_i$.

Proof. Let $x^{(u)} = ux^{\text{new}} + (1-u)x$ and $\mu_i^{\text{new}} = \mu_i^t(x^{\text{new}}, s^{\text{new}})$. The definition of μ (3.6) shows that

$$\begin{aligned}
\mu_i^{\text{new}} &= \mu_i + \frac{1}{t}\delta_{s,i} + \nabla\phi_i(x_i^{\text{new}}) - \nabla\phi_i(x_i) \\
&= \mu_i + \frac{1}{t}\delta_{s,i} + \int_0^1 \nabla^2\phi_i(x_i^{(u)})\delta_{x,i}\,\mathrm{d}u \\
&= \mu_i + \frac{1}{t}\delta_{s,i} + \nabla^2\phi_i(\overline{x}_i)\delta_{x,i} + \int_0^1 \left(\nabla^2\phi_i(x_i^{(u)}) - \nabla^2\phi_i(\overline{x}_i)\right)\delta_{x,i}\,\mathrm{d}u.
\end{aligned}$$

By the definition of δ_x and δ_s (3.10) and (3.11), we have that $\frac{1}{t}\delta_{s,i} + \widetilde{V}_i^{-1}\delta_{x,i} = h_i$. Hence, we have

$$\mu_i^{\text{new}} = \mu_i + h_i + \epsilon_i^{(\mu)}$$

where

$$\epsilon_i^{(\mu)} = \int_0^1 \left(\nabla^2\phi_i(x_i^{(u)}) - \nabla^2\phi_i(\overline{x}_i)\right)\delta_{x,i}\,\mathrm{d}u + (\nabla^2\phi_i(\overline{x}_i) - \widetilde{V}_i^{-1})\delta_{x,i}. \tag{3.12}$$

To bound $\epsilon_i^{(\mu)}$, we note that

$$\|x_i^{(t)} - \overline{x}_i\|_{\overline{x}_i} \leq \|x_i^{(t)} - x_i\|_{\overline{x}_i} + \|x_i - \overline{x}_i\|_{\overline{x}_i} \leq \|\delta_{x,i}\|_{\overline{x}_i} + \alpha = \alpha_i + \alpha \leq 3\alpha$$

where the first step follows from triangle inequality, the third step follows from definition of α_i (Lemma 3.5.1), and the last step follows from $\alpha_i \le 2\alpha$ (Lemma 3.5.1).

Using $\alpha \le \frac{1}{100}$, Theorem 3.2.1 shows that

$$-7\alpha \cdot \nabla^2\phi_i(\overline{x}_i) \preceq \nabla^2\phi_i(x_i^{(u)}) - \nabla^2\phi_i(\overline{x}_i) \preceq 7\alpha \cdot \nabla^2\phi_i(\overline{x}_i).$$

Equivalently, we have

$$(\nabla^2\phi_i(x_i^{(u)}) - \nabla^2\phi_i(\overline{x}_i)) \cdot (\nabla^2\phi_i(\overline{x}_i))^{-1} \cdot (\nabla^2\phi_i(x_i^{(u)}) - \nabla^2\phi_i(\overline{x}_i)) \preceq (7\alpha)^2 \cdot \nabla^2\phi_i(\overline{x}_i).$$

Using this, we have

$$\begin{aligned}\left\| \int_0^1 \left(\nabla^2\phi_i(x_i^{(u)}) - \nabla^2\phi_i(\overline{x}_i)\right)\delta_{x,i}\,\mathrm{d}u \right\|_{\overline{x}_i}^* &\le \int_0^1 \left\| \left(\nabla^2\phi_i(x_i^{(u)}) - \nabla^2\phi_i(\overline{x}_i)\right)\delta_{x,i} \right\|_{\overline{x}_i}^* \mathrm{d}u \\ &\le 7\alpha \|\delta_{x,i}\|_{\overline{x}_i} = 7\alpha \cdot \alpha_i, \end{aligned} \tag{3.13}$$

where the last step follows from definition of α_i (Lemma 3.5.1).

For the other term in $\epsilon_i^{(\mu)}$, we note that

$$(1-2\alpha) \cdot (\nabla^2\phi_i(\overline{x}_i)) \preceq \widetilde{V}_i^{-1} \preceq (1+2\alpha) \cdot (\nabla^2\phi_i(\overline{x}_i)).$$

Hence, we have

$$\left\| (\nabla^2\phi_i(\overline{x}_i) - \widetilde{V}_i^{-1})\delta_{x,i} \right\|_{\overline{x}_i}^* \le 2\alpha \|\delta_{x,i}\|_{\overline{x}_i} = 2\alpha \cdot \alpha_i. \tag{3.14}$$

Combining (3.12), (3.13) and (3.14), we have

$$\|\epsilon_i^{(\mu)}\|_{\overline{x}_i}^* \le 9\alpha \cdot \alpha_i.$$

Finally, we use the fact that x_i and $\overline{x}_i$ are α close and hence again by self-concordance, $\|\epsilon_i^{(\mu)}\|_{x_i}^* \le 10\alpha \cdot \alpha_i$.

□

Before bounding the change of γ, we first prove a helper lemma:

Lemma 3.5.3. *For all $i \in [m]$, we have*

$$\|\mu_i^t(x,s) - \mu_i^t(\overline{x},\overline{s})\|_{x_i}^* \le 4\alpha.$$

Proof. Note that

$$\|\mu_i^t(x,s) - \mu_i^t(\overline{x},\overline{s})\|_{\overline{x}_i}^* = \frac{1}{t}\|s_i - \overline{s}_i\|_{\overline{x}_i}^* + \|\nabla\phi_i(x_i) - \nabla\phi_i(\overline{x}_i)\|_{\overline{x}_i}^*.$$

For the first term, we have $\|s_i - \overline{s}_i\|_{\overline{x}_i}^* \le t\alpha$.

For the second term, let $x_i^{(u)} = ux_i + (1-u)\overline{x}_i$. Since x_i is close enough to $\overline{x}_i$, Theorem 3.2.1 shows that $\nabla^2\phi_i(x_i^{(u)}) \preceq 2 \cdot \nabla^2\phi_i(\overline{x}_i)$. Hence, we have

$$\|\nabla\phi_i(x_i) - \nabla\phi_i(\overline{x}_i)\|_{\overline{x}_i}^* = \left\|\int_0^1 \nabla^2\phi_i(x_i^{(u)}) \cdot (x_i - \overline{x}_i)du\right\|_{\overline{x}_i}^* \le 2\|x_i - \overline{x}_i\|_{\overline{x}_i} = 2\alpha.$$

Hence, we have $\|\mu_i^t(x,s) - \mu_i^t(\overline{x},\overline{s})\|_{\overline{x}_i}^* \le 3\alpha$ and using again x_i is close enough to $\overline{x}_i$ to get the final result.

□

Lemma 3.5.4 (Changes in γ)**.** *For all $i \in [m]$, let*

$$\gamma_i^t(x^{\text{new}}, x, s^{\text{new}}) \le (1 - \alpha \cdot c_i^t(\overline{x},\overline{s}))\gamma_i^t(x,x,s) + \epsilon_i^{(\gamma)}.$$

then $\epsilon_i^{(\gamma)} \le 10\alpha \cdot (\alpha c_i^t(\overline{x},\overline{s}) + \alpha_i)$. Furthermore, we have $|\gamma_i^t(x^{\text{new}}, x, s^{\text{new}}) - \gamma_i^t(x,x,s)| \le 3\alpha$.

Proof. For the first claim, Lemma 3.5.2, the definition of γ (3.7), h (3.8) and c (3.9) shows that

$$\begin{aligned}\gamma_i^t(x^{\text{new}}, x, s^{\text{new}}) &= \|\mu_i^t(x,s) + h_i + \epsilon_i^{(\mu)}\|_{x_i}^* \\ &= \|(1 - \alpha \cdot c_i^t(\overline{x},\overline{s}))\mu_i^t(x,s) + \epsilon_i\|_{x_i}^*\end{aligned}$$

where $\epsilon_i = \alpha \cdot c_i^t(\overline{x}, \overline{s})(\mu_i^t(x, s) - \mu_i^t(\overline{x}, \overline{s})) + \epsilon_i^{(\mu)}$.

From the definition of c_i^t, we have that $c_i^t \le \frac{1}{96\sqrt{\alpha}} \le \frac{1}{\alpha}$ and hence $0 \le 1 - \alpha \cdot c_i^t(\overline{x}, \overline{s}) \le 1$. Therefore, we have

$$\gamma_i^t(x^{\text{new}}, x, s^{\text{new}}) \le (1 - \alpha \cdot c_i^t(\overline{x}, \overline{s}))\gamma_i^t(x, x, s) + \|\epsilon_i\|_{x_i}^*. \tag{3.15}$$

Now, we bound $\|\epsilon_i\|_{x_i}^*$:

$$\begin{aligned} \|\epsilon_i\|_{x_i}^* &\le \alpha c_i^t(\overline{x}, \overline{s}) \cdot \|\mu_i^t(x, s) - \mu_i^t(\overline{x}, \overline{s})\|_{x_i}^* + \|\epsilon_i^{(\mu)}\|_{x_i}^* \\ &\le 4\alpha^2 c_i^t(\overline{x}, \overline{s}) + 10\alpha \cdot \alpha_i \end{aligned} \tag{3.16}$$

where we used Lemma 3.5.3 and Lemma 3.5.2 at the end.

For the second claim, we have

$$\left|\gamma_i^t(x^{\text{new}}, x, s^{\text{new}}) - \gamma_i^t(x, x, s)\right| \le \|h_i + \epsilon_i^{(\mu)}\|_{x_i}^* \le 2\alpha + 10\alpha \cdot \alpha_i$$

where we used (3.16) and that $\|h_i\|_{x_i}^* \le 2\|h\|_{\overline{x}}^* \le 2\alpha$. From Lemma 3.5.1 and that $\alpha \le \frac{1}{10000}$, we have $10\alpha \cdot \alpha_i \le 20\alpha^2 \le \alpha$.

□

3.5.3 Movement from (x, x, s) to $(x^{\text{new}}, x, s^{\text{new}})$

In the previous section, we see that γ_i will be expected to decrease by a factor of $\alpha \cdot c_i^t$ up to some small perturbations. We show that our potential Φ^t will therefore decrease significantly.

Lemma 3.5.5 (Movement along the first and third parameters)**.** *Assume that $\gamma_i^t(x, x, s) \leq 1$ for all i. We have*

$$\Phi^t(x^{\text{new}}, x, s^{\text{new}}) \leq \Phi^t(x, x, s) - \frac{\alpha\lambda}{5}\left(\sum_{i=1}^m \exp(2\lambda\gamma_i^t(\overline{x}, \overline{s}))\right)^{1/2} + \sqrt{m}\lambda \cdot \exp(192\lambda\sqrt{\alpha}).$$

Note that γ is a function that has three inputs. We use $\gamma(x, s)$ to denote $\gamma(x, x, s)$ for simplicity.

Proof. Let $\Phi^{\text{new}} = \Phi^t(x^{\text{new}}, x, s^{\text{new}})$, $\Phi = \Phi^t(x, x, s)$,

$$\gamma^{(u)} = u\gamma_i^t(x^{\text{new}}, x, s^{\text{new}}) + (1-u)\gamma_i^t(x, x, s).$$

Then, we have that

$$\Phi^{\text{new}} - \Phi = \sum_{i=1}^m (e^{\lambda\gamma_i^{(1)}} - e^{\lambda\gamma_i^{(0)}}) = \lambda\sum_{i=1}^m e^{\lambda\gamma_i^{(\zeta)}}(\gamma_i^{(1)} - \gamma_i^{(0)})$$

for some $0 \leq \zeta \leq 1$. Let $v_i = \gamma_i^{(1)} - \gamma_i^{(0)}$. Lemma 3.5.4 shows that

$$v_i \leq -\alpha \cdot c_i^t(\overline{x}, \overline{s}) \cdot \gamma_i^t(x, x, s) + \epsilon_i^{(\gamma)} = -\alpha \cdot c_i^t(\overline{x}, \overline{s}) \cdot \gamma_i^{(0)} + \epsilon_i^{(\gamma)}$$

and hence

$$\frac{\Phi^{\text{new}} - \Phi}{\lambda} \leq -\alpha\sum_{i=1}^m c_i^t(\overline{x}, \overline{s}) \cdot \gamma_i^{(0)} \exp(\lambda\gamma_i^{(\zeta)}) + \sum_{i=1}^m \epsilon_i^{(\gamma)} \exp(\lambda\gamma_i^{(\zeta)}). \tag{3.17}$$

To bound the first term in (3.17), we first relate $\gamma_i^{(0)}$, $\gamma_i^{(\zeta)}$ and $\gamma_i^t(\overline{x},\overline{s})$. Lemma 3.5.4 shows that

$$|\gamma_i^{(0)} - \gamma_i^{(\zeta)}| \leq |\gamma_i^{(0)} - \gamma_i^{(1)}| \leq 3\alpha. \tag{3.18}$$

Finally, we have

$$\begin{aligned}
\left|\gamma_i^t(\overline{x},\overline{s}) - \gamma_i^{(0)}\right| &= \left|\gamma_i^t(\overline{x},\overline{x},\overline{s}) - \gamma_i^t(x,x,s)\right| \\
&\leq \left|\gamma_i^t(\overline{x},\overline{x},\overline{s}) - \gamma_i^t(x,\overline{x},s)\right| + \left|\gamma_i^t(x,\overline{x},s) - \gamma_i^t(x,x,s)\right| \\
&\leq \|\mu_i^t(x,s) - \mu_i^t(\overline{x},\overline{s})\|_{\overline{x}_i}^* + \left|\|\mu_i^t(x,s)\|_{\overline{x}_i}^* - \|\mu_i^t(x,s)\|_{x_i}^*\right| \\
&\leq 2\|\mu_i^t(x,s) - \mu_i^t(\overline{x},\overline{s})\|_{x_i}^* + \left|\|\mu_i^t(x,s)\|_{\overline{x}_i}^* - \|\mu_i^t(x,s)\|_{x_i}^*\right| \\
&\leq 2\|\mu_i^t(x,s) - \mu_i^t(\overline{x},\overline{s})\|_{x_i}^* + 2\alpha\|\mu_i^t(x,s)\|_{x_i}^* \\
&\leq 8\alpha + 2\alpha = 10\alpha
\end{aligned} \tag{3.19}$$

where the first step follows from definition, the second and third step follows from triangle inequality, the fourth step follows from $\|\mu_i^t(x,s) - \mu_i^t(\overline{x},\overline{s})\|_{\overline{x}_i}^* \leq 2\|\mu_i^t(x,s) - \mu_i^t(\overline{x},\overline{s})\|_{x_i}^*$, the fifth step follows from self-concordance, the sixth step follows from Lemma 3.5.3 and that $\|\mu_i^t(x,s)\|_{x_i}^* = \gamma_i^t(x,x,s) \leq 1$ for all i

Using (3.18) and (3.19), we have

$$\begin{aligned}
&\sum_{i=1}^{m} c_i^t(\overline{x},\overline{s})\cdot\gamma_i^{(0)}\exp(\lambda\gamma_i^{(\zeta)})\\
=&\sum_{i=1}^{m} c_i^t(\overline{x},\overline{s})\cdot\gamma_i^{(0)}\exp(\lambda\gamma_i^t(\overline{x},\overline{s})-\lambda\gamma_i^t(\overline{x},\overline{s})+\lambda\gamma_i^{(0)}-\lambda\gamma_i^{(0)}+\lambda\gamma_i^{(\zeta)})\\
\geq&\sum_{i=1}^{m} c_i^t(\overline{x},\overline{s})\cdot\gamma_i^{(0)}\exp(\lambda\gamma_i^t(\overline{x},\overline{s})-13\lambda\alpha)\\
\geq&\frac{1}{2}\sum_{i=1}^{m} c_i^t(\overline{x},\overline{s})\cdot\gamma_i^{(0)}\exp(\lambda\gamma_i^t(\overline{x},\overline{s}))\\
\geq&\frac{1}{2}\sum_{i=1}^{m} c_i^t(\overline{x},\overline{s})\cdot\gamma_i^t(\overline{x},\overline{s})\exp(\lambda\gamma_i^t(\overline{x},\overline{s}))-3\alpha\sum_{i=1}^{m} c_i^t(\overline{x},\overline{s})\exp(\lambda\gamma_i^t(\overline{x},\overline{s})).
\end{aligned}\tag{3.20}$$

where the third step follows from $\exp(-13\lambda\alpha)\geq 1/2$, and the last step follows from (3.18).

For the first term in (3.20), we have

$$\begin{aligned}
&\sum_{i=1}^{m} c_i^t(\overline{x},\overline{s})\cdot\gamma_i^t(\overline{x},\overline{s})\exp(\lambda\gamma_i^t(\overline{x},\overline{s}))\\
=&\sum_{\gamma_i^t(\overline{x},\overline{s})\geq 96\sqrt{\alpha}}\frac{\exp(2\lambda\cdot\gamma_i^t(\overline{x},\overline{s}))}{(\sum_{i=1}^{m}\exp(2\lambda\gamma_i^t(\overline{x},\overline{s})))^{1/2}}\\
=&\sum_{i=1}^{m}\frac{\exp(2\lambda\cdot\gamma_i^t(\overline{x},\overline{s}))}{(\sum_{i=1}^{m}\exp(2\lambda\gamma_i^t(\overline{x},\overline{s})))^{1/2}}-\sum_{\gamma_i^t(\overline{x},\overline{s})<96\sqrt{\alpha}}\frac{\exp(2\lambda\cdot\gamma_i^t(\overline{x},\overline{s}))}{(\sum_{i=1}^{m}\exp(2\lambda\gamma_i^t(\overline{x},\overline{s})))^{1/2}}\\
\geq&\left(\sum_{i=1}^{m}\exp(2\lambda\gamma_i^t(\overline{x},\overline{s}))\right)^{1/2}-\frac{m\cdot\exp(192\lambda\cdot\sqrt{\alpha})}{(\sum_{i=1}^{m}\exp(2\lambda\gamma_i^t(\overline{x},\overline{s})))^{1/2}}.
\end{aligned}$$

So, if $\sum_{i=1}^{m}\exp(2\lambda\gamma_i^t(\overline{x},\overline{s}))\geq m\cdot\exp(192\lambda\cdot\sqrt{\alpha})$, we have

$$\sum_{i=1}^{m} c_i^t(\overline{x},\overline{s})\cdot\gamma_i^t(\overline{x},\overline{s})\exp(\lambda\gamma_i^t(\overline{x},\overline{s}))\geq\left(\sum_{i=1}^{m}\exp(2\lambda\gamma_i^t(\overline{x},\overline{s}))\right)^{1/2}-\sqrt{m}\cdot\exp(192\lambda\sqrt{\alpha}).$$

Note that if $\sum_{i=1}^{m}\exp(2\lambda\gamma_i^t(\overline{x},\overline{s}))\leq m\cdot\exp(192\lambda\cdot\sqrt{\alpha})$, this is still true because left hand

side is lower bounded by 0. For the second term in (3.20), we have

$$\begin{aligned}\sum_{i=1}^{m} c_i^t(\overline{x},\overline{s})\exp(\lambda\gamma_i^t(\overline{x},\overline{s})) &= \sum_{\gamma_i^t(\overline{x},\overline{s})\geq 96\sqrt{\alpha}} \frac{\exp(\lambda\cdot\gamma_i^t(\overline{x},\overline{s}))/\gamma_i^t(\overline{x},\overline{s})}{(\sum_{i=1}^m \exp(2\lambda\gamma_i^t(\overline{x},\overline{s})))^{1/2}}\exp(\lambda\gamma_i^t(\overline{x},\overline{s}))\\ &\leq \frac{1}{96\sqrt{\alpha}}\sum_{\gamma_i^t(\overline{x},\overline{s})\geq 96\sqrt{\alpha}} \frac{\exp(2\lambda\cdot\gamma_i^t(\overline{x},\overline{s}))}{(\sum_{i=1}^m \exp(2\lambda\gamma_i^t(\overline{x},\overline{s})))^{1/2}}\\ &\leq \frac{1}{96\sqrt{\alpha}}\sum_{i=1}^{m} \frac{\exp(2\lambda\cdot\gamma_i^t(\overline{x},\overline{s}))}{(\sum_{i=1}^m \exp(2\lambda\gamma_i^t(\overline{x},\overline{s})))^{1/2}}\\ &= \frac{1}{96\sqrt{\alpha}}\left(\sum_{i=1}^{m}\exp(2\lambda\gamma_i^t(\overline{x},\overline{s}))\right)^{1/2}.\end{aligned}$$

where the second step follows $\frac{1}{\gamma_i^t(\overline{x},\overline{s})} \leq \frac{1}{96\sqrt{\alpha}}$, and the third step follows from each term in the summation is non-negative.

Combining the bounds for both first and second term in (3.20), we have

$$\begin{aligned}\sum_{i=1}^{m} c_i^t(\overline{x},\overline{s})\cdot\gamma_i^{(0)}\exp(\lambda\gamma_i^{(\zeta)}) &\geq \frac{1}{2}\left(\left(\sum_{i=1}^{m}\exp(2\lambda\gamma_i^t(\overline{x},\overline{s}))\right)^{1/2} - \sqrt{m}\cdot\exp(192\lambda\sqrt{\alpha})\right)\\ &\quad - \frac{3\alpha}{96\sqrt{\alpha}}\left(\sum_{i=1}^{m}\exp(2\lambda\gamma_i^t(\overline{x},\overline{s}))\right)^{1/2}\\ &\geq \frac{2}{5}\left(\sum_{i=1}^{m}\exp(2\lambda\gamma_i^t(\overline{x},\overline{s}))\right)^{1/2} - \sqrt{m}\cdot\exp(192\lambda\sqrt{\alpha}). \qquad (3.21)\end{aligned}$$

where the last step follows from $\frac{1}{2} - \frac{3\alpha}{96\sqrt{\alpha}} \geq \frac{1}{2} - \frac{3}{96} = \frac{45}{96} \geq \frac{2}{5}$.

For the second term in (3.17), we note that $|\gamma_i^{(\zeta)} - \gamma_i^t(\overline{x},\overline{s})| \leq 13\alpha \leq \frac{1}{2\lambda}$ by (3.18) and (3.19). Hence,

$$\sum_{i=1}^{m}\epsilon_i^{(\gamma)}\exp(\lambda\gamma_i^{(\zeta)}) \leq 2\sum_{i=1}^{m}\epsilon_i^{(\gamma)}\exp(\lambda\gamma_i^t(\overline{x},\overline{s})).$$

Now, we use $\epsilon_i^{(\gamma)} \le 10\alpha \cdot (\alpha c_i^t(\overline{x},\overline{s}) + \alpha_i)$ (Lemma 3.5.4) to get

$$\begin{aligned}\sum_{i=1}^m \epsilon_i^{(\gamma)} \exp(\lambda\gamma_i^{(\zeta)}) &\le 20\alpha \sum_{i=1}^m (\alpha c_i^t(\overline{x},\overline{s}) + \alpha_i) \cdot \exp(\lambda\gamma_i^t(\overline{x},\overline{s})) \\ &\le 20\alpha \left(\sum_{i=1}^m (\alpha c_i^t(\overline{x},\overline{s}) + \alpha_i)^2\right)^{1/2} \left(\sum_{i=1}^m \exp(2\lambda\gamma_i^t(\overline{x},\overline{s}))\right)^{1/2}.\end{aligned}$$

where the last step follows from Cauchy-Schwarz inequality.

Note that by using Cauchy-Schwarz,

$$\begin{aligned}\left(\sum_{i=1}^m (\alpha c_i^t(\overline{x},\overline{s}) + \alpha_i)^2\right)^{1/2} &\le \alpha \left(\sum_{i=1}^m c_i^t(\overline{x},\overline{s})^2\right)^{1/2} + \left(\sum_{i=1}^m \alpha_i^2\right)^{1/2} \\ &\le \alpha \cdot \frac{1}{96\sqrt{\alpha}} + 2\alpha \le \frac{\sqrt{\alpha}}{90}.\end{aligned}$$

where we used the definition of c_i^t, Lemma 3.5.1 and $\alpha \le \frac{1}{2^{24}}$. Together, we conclude

$$\sum_{i=1}^m \epsilon_i^{(\gamma)} \exp(\lambda\gamma_i^{(\zeta)}) \le \frac{1}{5}\alpha \left(\sum_{i=1}^m \exp(2\lambda\gamma_i^t(\overline{x},\overline{s}))\right)^{1/2}. \tag{3.22}$$

Combining (3.21) and (3.22) to (3.17) gives

$$\begin{aligned}\frac{\Phi^{\text{new}} - \Phi}{\lambda} &\le -\frac{2}{5}\alpha \left(\sum_{i=1}^m \exp(2\lambda\gamma_i^t(\overline{x},\overline{s}))\right)^{1/2} + \sqrt{m} \cdot \exp(192\lambda\sqrt{\alpha}) + \frac{1}{5}\alpha \left(\sum_{i=1}^m \exp(2\lambda\gamma_i^t(\overline{x},\overline{s}))\right)^{1/2} \\ &= -\frac{1}{5}\alpha \left(\sum_{i=1}^m \exp(2\lambda\gamma_i^t(\overline{x},\overline{s}))\right)^{1/2} + \sqrt{m} \cdot \exp(192\lambda\sqrt{\alpha}).\end{aligned}$$

where the last step follows from merging the first term with the third term. □

3.5.4 Movement from $(x^{\text{new}}, x, s^{\text{new}})$ to $(x^{\text{new}}, x^{\text{new}}, s^{\text{new}})$

Next, we must analyze the potential change when we change the second term.

Lemma 3.5.6 (Movement along the second parameter)**.** *Assume that* $\|\gamma^t(x,x,s)\|_\infty \leq 1$. *Then we have*

$$\Phi^t(x^{\text{new}}, x^{\text{new}}, s^{\text{new}}) \leq \Phi^t(x^{\text{new}}, x, s^{\text{new}}) + 12\alpha(\|\gamma^t(x,x,s)\|_\infty + 3\alpha)\lambda \left(\sum_{i=1}^m \exp(2\lambda\gamma_i^t(x,x,s))\right)^{1/2}.$$

Proof. We can upper bound $\Phi^t(x^{\text{new}}, x^{\text{new}}, s^{\text{new}})$ as follows

$$\begin{aligned}\Phi^t(x^{\text{new}}, x^{\text{new}}, s^{\text{new}}) &= \sum_{i=1}^m \exp(\lambda\gamma_i^t(x^{\text{new}}, x^{\text{new}}, s^{\text{new}})) \\ &\leq \sum_{i=1}^m \exp(\lambda\gamma_i^t(x^{\text{new}}, x, s^{\text{new}})(1+2\alpha_i)).\end{aligned}$$

where the second step follows from $\gamma_i^t(x^{\text{new}}, x^{\text{new}}, s^{\text{new}}) \leq \gamma_i^t(x^{\text{new}}, x, s^{\text{new}}) \cdot (1+2\alpha_i)$ by self-concordance (Theorem 3.2.1) and $\|x_i^{\text{new}} - x_i\|_{x_i} \leq 2\|x_i^{\text{new}} - x_i\|_{\overline{x}_i} \leq 2\alpha_i$.

Now, by Lemma 3.5.4, we note that $\gamma_i^t(x^{\text{new}}, x, s^{\text{new}}) \leq \gamma_i^t(x,x,s) + 3\alpha \leq 1 + 3\alpha$ and that $\alpha \leq \frac{1}{100\lambda}$. Hence, by a simple taylor expansion, we have

$$\begin{aligned}&\Phi^t(x^{\text{new}}, x^{\text{new}}, s^{\text{new}}) \\ &\leq \sum_{i=1}^m \exp(\lambda\gamma_i^t(x^{\text{new}}, x, s^{\text{new}})) + 3\sum_{i=1}^m \alpha_i \exp(\lambda\gamma_i^t(x^{\text{new}}, x, s^{\text{new}}))\gamma_i^t(x^{\text{new}}, x, s^{\text{new}}).\end{aligned}$$

Finally, we bound the last term by

$$
\begin{aligned}
&\sum_{i=1}^{m} \exp(\lambda\gamma_i^t(x^{\text{new}}, x, s^{\text{new}}))\gamma_i^t(x^{\text{new}}, x, s^{\text{new}})\alpha_i \\
\leq &\sum_{i=1}^{m} \exp(\lambda\gamma_i^t(x, x, s) + 3\lambda\alpha)(\gamma_i^t(x, x, s) + 3\alpha)\alpha_i \\
\leq &\, 2(\|\gamma^t(x, x, s)\|_\infty + 3\alpha) \sum_{i=1}^{m} \exp(\lambda\gamma_i^t(x, x, s))\alpha_i \\
\leq &\, 2(\|\gamma^t(x, x, s)\|_\infty + 3\alpha) \left(\sum_{i=1}^{m} \exp(2\lambda\gamma_i^t(x, x, s))\right)^{1/2} \left(\sum_{i=1}^{m} \alpha_i^2\right)^{1/2} \\
\leq &\, 4\alpha(\|\gamma^t(x, x, s)\|_\infty + 3\alpha) \left(\sum_{i=1}^{m} \exp(2\lambda\gamma_i^t(x, x, s))\right)^{1/2},
\end{aligned}
$$

where the first step follows from $\lambda\gamma_i^t(x^{\text{new}}, x, s^{\text{new}}) \leq \exp(\lambda\gamma_i^t(x, x, s) + 3\lambda\alpha)$, the second step follows $\exp(3\lambda\alpha) \leq 2$, the third step follows from Cauchy-Schwarz inequality, the last step follows from $\sum_{i=1}^{m} \alpha_i^2 \leq 4\alpha^2$.

□

3.5.5 Movement of t

Lastly, we analyze the effect of setting $t \to t^{\text{new}}$.

Lemma 3.5.7 (Movement in t)**.** *For any x, s such that $\gamma_i^t(x,s) \leq 1$ for all i, let $t^{\text{new}} = \left(1 - \frac{\kappa}{\sqrt{\nu}}\right) t$ where $\nu = \sum_{i=1}^m \nu_i$, we have*

$$\Phi^{t^{\text{new}}}(x,s) \leq \Phi^t(x,s) + 10\kappa\lambda \left(\sum_{i=1}^m \exp(2\lambda \gamma_i^t(x,s)) \right)^{1/2}.$$

Proof. Note that

$$\begin{aligned}
\gamma_i^{t^{\text{new}}}(x,s) = & \left\| \frac{s}{t^{\text{new}}} + \nabla \phi_i(x_i) \right\|_{x_i}^* \\
= & \left\| \frac{s}{t(1-\kappa/\sqrt{\nu})} + \nabla \phi_i(x_i) \right\|_{x_i}^* \\
\leq & (1 + 2\kappa/\sqrt{\nu}) \gamma_i^t(x,s) + 2\|(\kappa/\sqrt{\nu}) \nabla \phi_i(x_i)\|_{x_i}^* \\
\leq & (1 + 2\kappa/\sqrt{\nu}) \gamma_i^t(x,s) + 3\kappa\sqrt{\nu_i}/\sqrt{\nu} \\
\leq & \gamma_i^t(x,s) + 5\kappa\sqrt{\nu_i}/\sqrt{\nu}
\end{aligned}$$

where the first step follows from definition, the second step follows from $t^{\text{new}} = t(1-\kappa/\sqrt{\nu})$, the second last step follows from the fact that our barriers are ν_i-self-concordant and the last step used $\gamma_i^t(x,s) \leq 1$ and $\nu_i \geq 1$. Using that $5\kappa \leq \frac{1}{10\lambda}$ and $\gamma_i^t(x,s) \leq 1$, we have by

simple taylor expansion,

$$\begin{aligned}
\Phi^{t^{\text{new}}}(x,s) &\leq \sum_{i=1}^{m} \exp(\lambda\gamma_i^t(x,s)) + 2\lambda \sum_{i=1}^{m} \exp(\lambda\gamma_i^t(x,s)) \left(5\kappa\sqrt{\nu_i/\nu}\right) \\
&= \sum_{i=1}^{m} \exp(\lambda\gamma_i^t(x,s)) + 10\kappa\lambda \sum_{i=1}^{m} \exp(\lambda\gamma_i^t(x,s)) \left(\sqrt{\nu_i/\nu}\right) \\
&\leq \sum_{i=1}^{m} \exp(\lambda\gamma_i^t(x,s)) + 10\kappa\lambda \left(\sum_{i=1}^{m} \exp(2\lambda\gamma_i^t(x,s))\right)^{1/2} \left(\sum_{i=1}^{m} \frac{\nu_i}{\nu}\right)^{1/2} \\
&= \sum_{i=1}^{m} \exp(\lambda\gamma_i^t(x,s)) + 10\kappa\lambda \left(\sum_{i=1}^{m} \exp(2\lambda\gamma_i^t(x,s))\right)^{1/2},
\end{aligned}$$

where the third step follows from Cauchy-Schwarz, and the last step follows from $\sum_{i=1}^{m} \nu_i = \nu$. □

3.5.6 Potential Maintenance

Putting it all together, we can show that our potential Φ^t can be maintained to be small throughout our algorithm.

Lemma 3.5.8 (Potential Maintenance)**.** *If* $\Phi^t(x,s) \leq 80\frac{m}{\alpha}$*, then*

$$\Phi^{t^{\text{new}}}(x^{\text{new}}, s^{\text{new}}) \leq \left(1 - \frac{\alpha\lambda}{40\sqrt{m}}\right)\Phi^t(x,s) + \sqrt{m}\lambda \cdot \exp(192\lambda\sqrt{\alpha}).$$

In particularly, we have $\Phi^{t^{\text{new}}}(x^{\text{new}}, s^{\text{new}}) \leq 80\frac{m}{\alpha}$.

Proof. Let

$$\zeta(x,s) = \left(\sum_{i=1}^{m} \exp(2\lambda\gamma_i^t(x,s))\right)^{1/2}.$$

By combining our previous lemmas,

$$\begin{aligned}
&\Phi^{t^{\text{new}}}(x^{\text{new}}, s^{\text{new}}) \\
\leq &\Phi^t(x^{\text{new}}, s^{\text{new}}) + 10\kappa\lambda \cdot \zeta(x^{\text{new}}, s^{\text{new}}) \\
\leq &\Phi^t(x^{\text{new}}, x, s^{\text{new}}) + 12\alpha\lambda(\|\gamma^t(x,s)\|_\infty + 3\alpha) \cdot \zeta(x,s) + 10\kappa\lambda \cdot \zeta(x^{\text{new}}, s^{\text{new}}) \\
\leq &\Phi^t(x,x,s) - \frac{\alpha\lambda}{5}\zeta(\overline{x}, \overline{s}) + \sqrt{m}\lambda \cdot \exp(192\lambda\sqrt{\alpha}) \\
&+ 12\alpha\lambda(\|\gamma^t(x,s)\|_\infty + 3\alpha) \cdot \zeta(x,s) + 10\kappa\lambda \cdot \zeta(x^{\text{new}}, s^{\text{new}})
\end{aligned} \tag{3.23}$$

where the first step follows from Lemma 3.5.7, the second step follows from Lemma 3.5.6, and the last step follows from Lemma 3.5.5. We note that in all lemma above, we used that fact that $\|\gamma^t\|_\infty \leq 1$ (for different combination of x, $\overline{x}$, x^{new}, s, $\overline{s}$, s^{new}) which we will show later.

We can upper bound $\gamma_i^t(x^{\text{new}}, s^{\text{new}})$ in the following sense,

$$\gamma_i^t(x^{\text{new}}, s^{\text{new}}) \leq \gamma_i^t(x^{\text{new}}, x, s^{\text{new}}) + 2\alpha \leq \gamma_i^t(x,x,s) + 5\alpha. \tag{3.24}$$

where the first step follows from self-concordance and $\gamma_i \le 1$, the second step follows from Lemma 3.5.4.

Hence, since ζ changes multiplicatively when γ changes additively, $\zeta(x^{\text{new}}, s^{\text{new}}) \le 2\zeta(x,s)$.

Lemma 3.5.3 shows that $\|\mu_i^t(x,s) - \mu_i^t(\overline{x},\overline{s})\|_{x_i}^* \le 4\alpha$ and hence

$$\begin{aligned}\zeta(\overline{x},\overline{s}) &\ge \frac{2}{3}\left(\sum_{i=1}^m \exp(2\lambda\gamma_i^t(\overline{x},x,\overline{s}))\right)^{1/2} \\ &\ge \frac{2}{3}\left(\sum_{i=1}^m \exp(2\lambda\gamma_i^t(x,x,s) - 8\alpha\lambda)\right)^{1/2} \\ &\ge \frac{1}{2}\zeta(x,s). \end{aligned} \tag{3.25}$$

Combining (3.24) and (3.25) into (3.23) gives

$$\begin{aligned}&\Phi^{t^{\text{new}}}(x^{\text{new}}, s^{\text{new}}) \\ &\ge \Phi^t(x,s) + \left(12\alpha\lambda(\|\gamma^t(x,s)\|_\infty + 3\alpha) + 20\kappa\lambda - \frac{\alpha\lambda}{10}\right)\cdot\zeta(x,s) + \sqrt{m}\lambda\cdot\exp(192\lambda\sqrt{\alpha}) \\ &\ge \Phi^t(x,s) + \left(12\alpha\lambda\|\gamma^t(x,s)\|_\infty - \frac{\alpha\lambda}{20}\right)\cdot\zeta(x,s) + \sqrt{m}\lambda\cdot\exp(192\lambda\sqrt{\alpha})\end{aligned}$$

where the last step follows from $\kappa \le \frac{\alpha}{1000}$ and $\alpha \le \frac{1}{10000}$.

Finally, we need to bound $\|\gamma^t(x,s)\|_\infty$. The bound for other $\|\gamma^t\|_\infty$, i.e. for different combination of x, $\overline{x}$, x^{new}, s, $\overline{s}$, s^{new}, are similar. We note that

$$\Phi^t(x,s) \le 80\frac{m}{\alpha}$$

implies that $\|\gamma^t(x,s)\|_\infty \le \frac{\log(80\frac{m}{\alpha})}{\lambda}$. Hence, by our choice of λ and α, we have that $\lambda \ge 480\log(80\frac{m}{\alpha})$ and hence

$$12\alpha\lambda\|\gamma^t(x,s)\|_\infty \le \frac{\alpha\lambda}{40}.$$

Finally, using $\Phi^t(x,s) \le \sqrt{m} \cdot \zeta(x,s)$, we have

$$\begin{aligned}\Phi^{t^{\text{new}}}(x^{\text{new}}, s^{\text{new}}) &\ge \Phi^t(x,s) - \frac{\alpha\lambda}{40}\zeta(x,s) + \sqrt{m}\lambda \cdot \exp(192\lambda\sqrt{\alpha}) \\ &\ge \left(1 - \frac{\alpha\lambda}{40\sqrt{m}}\right)\Phi^t(x,s) + \sqrt{m}\lambda \cdot \exp(192\lambda\sqrt{\alpha}).\end{aligned}$$

Since $\lambda \le \frac{1}{400\sqrt{\alpha}}$, we have $\Phi^t(x,s) \le 80\frac{m}{\alpha}$ implies $\Phi^{t^{\text{new}}}(x^{\text{new}}, s^{\text{new}}) \le 80\frac{m}{\alpha}$.

□

Name	Type	Statement	Algorithm	Input	Output
Initialize	public	Lemma 3.6.2	Alg. 3.1	$A, x, s, \overline{W}, \epsilon_{mp}, a, b$	$\emptyset$
Update	public	Lemma 3.6.3	Alg. 3.2	$\overline{W}$	$\emptyset$
FullUpdate	private	Lemma 3.6.5	Alg. 3.3	$\overline{W}$	$\emptyset$
PartialUpdate	private	Lemma 3.6.4	Alg. 3.2	$\overline{W}$	$\emptyset$
Query	public	Lemma 3.6.6	Alg. 3.1	$\emptyset$	$\overline{x}, \overline{s}$
MultiplyMove	public	Lemma 3.6.9	Alg. 3.4	h, t	$\emptyset$
Multiply	private	Lemma 3.6.8	Alg. 3.4	h, t	$\emptyset$
Move	private	Lemma 3.6.7	Alg. 3.4	$\emptyset$	$\emptyset$

Table 3.2: Summary of data structure CentralPathMaintenance

3.6 Central Path Maintenance

The goal of this section is to present a data-structure to perform our centering steps in $\widetilde{O}(n^{\omega-1/2})$ amortized time and prove a theoretical guarantee of it. The original idea of inverse maintenance is from Michael B. Cohen [Lee17], then [CLS19] used it to get faster running time for solving Linear Programs. Because a simple matrix vector product would require $O(n^2)$ time, our speedup comes via a low-rank embedding that provides ℓ_∞ guarantees, which is unlike the sparse vector approach of [CLS19]. In fact, we are unsure if moving in a sparse direction h can have sufficiently controlled noise to show convergence. Here, we give a stochastic version that is faster for dense direction h.

Theorem 3.6.1 (Central path maintenance)**.** *Given a full rank matrix $A \in \mathbb{R}^{d \times n}$ with $n \geq d$, a tolerance parameter $0 < \epsilon_{mp} < 1/4$ and a block diagonal structure $n = \sum_{i=1}^{m} n_i$. Given any positive number a such $a \leq \alpha$ where α is the dual exponent of matrix multiplication. Given any linear sketch of size b, there is a randomized data structure* CentralPathMainte-

NANCE *(in Algorithm 3.1, 3.2, 3.4) that approximately maintains the projection matrices*

$$\sqrt{W}A^\top(AWA^\top)^{-1}A\sqrt{W}$$

for positive block diagonal psd matrix $W \oplus_i \mathbb{R}^{n_i \times n_i}$*; exactly implicitly maintains central path parameters* (x, s) *and approximately explicitly maintains path parameters through the following five operations:*

1. INITIALIZE$(\overline{W}^{(0)}, \cdots)$ *: Assume* $\overline{W}^{(0)} \in \otimes_i \mathbb{R}^{n_i \times n_i}$*. Initialize all the parameters in* $O(n^\omega)$ *time.*

2. UPDATE$(\overline{W})$ *: Assume* $\overline{W} \in \oplus_i \mathbb{R}^{n_i \times n_i}$*. Output a block diagonal matrix* $\widetilde{V} \oplus_i \mathbb{R}^{n_i \times n_i}$ *such that*

$$(1 - \epsilon_{mp})\widetilde{v}_i \preceq \overline{w}_i \preceq (1 + \epsilon_{mp})\widetilde{v}_i.$$

3. QUERY() *: Output* $(\overline{x}, \overline{s})$ *such that* $\|\overline{x} - x\|_{\widetilde{V}^{-1}} \leq \epsilon_{mp}$ *and* $\|\overline{s} - s\|_{\widetilde{V}} \leq t\epsilon_{mp}$ *where* t *is the last* t *used in* MULTIPLYMOVE*, where* $\epsilon_{mp} = \alpha \log^2(nT)\frac{n^{1/4}}{\sqrt{b}}$ *and the success probability is* $1 - 1/\operatorname{poly}(nT)$*. This step takes* $O(n)$ *time.*

4. MULTIPLYMOVE(h, t) *: It outputs nothing. It implicitly maintains:*

$$x = x + \widetilde{V}^{1/2}(I - \widetilde{P})\widetilde{V}^{1/2}h, s = s + t\widetilde{V}^{-1/2}\widetilde{P}\widetilde{V}^{1/2}h.$$

where $\widetilde{P} = \widetilde{V}^{1/2}A^\top(A\widetilde{V}A^\top)^{-1}A\widetilde{V}^{1/2}$*. It also explicitly maintains* $\overline{x}, \overline{s}$*. Assuming* t *is decreasing, each call takes* $O(nb + n^{a\omega+o(1)} + n^a\|h\|_0 + n^{1.5})$ *amortized time.*

Let $\overline{W}^{(0)}$ *be the initial matrix and* $\overline{W}^{(1)}, \cdots, \overline{W}^{(T)}$ *be the (random) update sequence. Under the assumption that there is a sequence of matrix* $W^{(0)}, \cdots, W^{(T)} \in \oplus_{i=1}^m \mathbb{R}^{n_i \times n_i}$ *sat-*

isfies for all k

$$\left\| w_i^{-1/2}(\overline{w}_i - w_i)w_i^{-1/2} \right\|_F \leq \epsilon_{mp},$$
$$\sum_{i=1}^{m} \left\| (w_i^{(k)})^{-1/2}(\mathbb{E}[w_i^{(k+1)}] - w_i^{(k)})(w_i^{(k)})^{-1/2} \right\|_F^2 \leq C_1^2,$$
$$\sum_{i=1}^{m} \left(\mathbb{E}\left[\left\| (w_i^{(k)})^{-1/2}(w_i^{(k+1)} - w_i^{(k)})(w_i^{(k)})^{-1/2} \right\|_F^2 \right] \right)^2 \leq C_2^2,$$
$$\left\| (w_i^{(k)})^{-1/2}(w_i^{(k+1)} - w_i^{(k)})(w_i^{(k)})^{-1/2} \right\|_F \leq \frac{1}{4}.$$

where $w_i^{(k)}$ is the i-th block of $W^{(k)}$, $\forall i \in [m]$.

Then, the amortized expected time per call of UPDATE(w) *is*

$$(C_1/\epsilon_{mp} + C_2/\epsilon_{mp}^2) \cdot (n^{\omega - 1/2 + o(1)} + n^{2 - a/2 + o(1)}).$$

Remark 3.6.1. For our algorithm, we have $C_1 = O(1/\log^2 n)$, $C_2 = O(1/\log^4 n)$ and $\epsilon_{mp} = O(1/\log^2 n)$. Note that the input of UPDATE $\overline{W}$ can move a lot. It is working as long as $\overline{W}$ is close to some W that is slowly moving. In our application, our W satisfies C_1, C_2 deterministically. We keep it for possible future applications.

Algorithm 3.1 Central Path Maintenance Data Structure - Initial, Query, Move

datastructure CentralPathMaintenance ▷ Theorem 3.6.1

private : members
- $\overline{W} \in \otimes_{i\in[m]}\mathbb{R}^{n_i\times n_i}$ ▷ Target vector, $\overline{W}$ is ϵ_w-close to W
- $V, \widetilde{V} \in \otimes_{i\in[m]}\mathbb{R}^{n_i\times n_i}$ ▷ Approximate vector
- $A \in \mathbb{R}^{d\times n}$ ▷ Constraints matrix
- $M \in \mathbb{R}^{n\times n}$ ▷ Approximate Projection Matrix
- $\epsilon_{mp} \in (0, 1/4)$ ▷ Tolerance
- $a \in (0, \alpha]$ ▷ Batch Size for Update (n^a)
- $b \in \mathbb{Z}_+$ ▷ Sketch size of one sketching matrix
- $R \in \mathbb{R}^{n^{1+o(1)}\times n}$ ▷ A list of sketching matrices
- $Q \in \mathbb{R}^{b\times n}$ ▷ Sketched matrices
- $u_1 \in \mathbb{R}^n, F \in \mathbb{R}^{n\times n}, u_2 \in \mathbb{R}^n$ ▷ Implicit representation of x, $x = u_1 + F \cdot u_2$
- $u_3 \in \mathbb{R}^n, G \in \mathbb{R}^{n\times n}, u_4 \in \mathbb{R}^n$ ▷ Implicit representation of s, $s = u_3 + G \cdot u_4$
- $\overline{x}$, $\overline{s} \in \mathbb{R}^n$ ▷ Central path parameters, maintain explicitly
- $l \in \mathbb{Z}_+$ ▷ Randomness counter, $R_l \in \mathbb{R}^{b\times n}$
- $t^{\mathrm{pre}} \in \mathbb{R}_+$ ▷ Tracking the changes of t

end members

public : procedure Initialize($A, x, s, W, \epsilon_{mp}, a, b$) ▷ Lemma 3.6.2
- ▷ parameters will *never change* after initialization
- $A \leftarrow A$, $a \leftarrow a$, $b \leftarrow b$, $\epsilon_{mp} \leftarrow \epsilon_{mp}$
- ▷ parameters will *still change* after initialization
- $\overline{W} \leftarrow W$, $V \leftarrow W$, $\widetilde{V} \leftarrow V$
- Choose $R_l \in \mathbb{R}^{b\times n}$ to be sketching matrix, $\forall l \in [\sqrt{n}]$ ▷ Lemma 3.9.1
- $R \leftarrow [R_1^\top, R_2^\top, \cdots]^\top$ ▷ Batch them into one matrix R
- $M \leftarrow A^\top (AVA^\top)^{-1} A$, $Q \leftarrow R\sqrt{\widetilde{V}}M$ ▷ Initialize projection matrices
- $u_1 \leftarrow x$, $u_2 \leftarrow 0$, $u_3 \leftarrow s$, $u_4 \leftarrow 0$ ▷ Initialize x and s
- $\overline{x} \leftarrow x$, $\overline{s} \leftarrow s$
- $l \leftarrow 1$

end procedure

public : procedure Query() ▷ Lemma 3.6.6
- **return** $(\overline{x}, \overline{s})$

end procedure

end datastructure

Algorithm 3.2 Central Path Maintenance Data Structure - Update and PartialUpdate

datastructure CentralPathMaintenance ▷ Theorem 3.6.1

public : procedure Update($\overline{W}^{\mathrm{new}}$) ▷ Lemma 3.6.3, $\overline{W}^{\mathrm{new}}$ is close to W^{new}
  $\overline{y}_i \leftarrow v_i^{-1/2} \overline{w}_i^{\mathrm{new}} v_i^{-1/2} - 1, \forall i \in [m]$
  $r \leftarrow$ the number of indices i such that $\|\overline{y}_i\|_F \geq \epsilon_{mp}$
  if $r < n^a$ **then**
    PartialUpdate($\overline{W}^{\mathrm{new}}$)
  else
    FullUpdate($\overline{W}^{\mathrm{new}}$) ▷ Algorithm 3.3
  end if
procedure

private : procedure PartialUpdate($\overline{W}^{\mathrm{new}}$) ▷ Lemma 3.6.4
  $\overline{W} \leftarrow \overline{W}^{\mathrm{new}}$
  $\widetilde{v}_i^{\mathrm{new}} \leftarrow \begin{cases} v_i & \text{if } (1-\epsilon_{mp}) v_i \preceq \overline{w}_i \preceq (1+\epsilon_{mp}) v_i \\ w_i & \text{otherwise} \end{cases}$
  $F^{\mathrm{new}} \leftarrow F + ((\widetilde{V}^{\mathrm{new}})^{1/2} - (\widetilde{V})^{1/2}) M$ ▷ only takes n^{1+a} time, instead of n^2
  $G^{\mathrm{new}} \leftarrow G + ((\widetilde{V}^{\mathrm{new}})^{-1/2} - (\widetilde{V})^{-1/2}) M$
  $u_1 \leftarrow u_1 + (F - F^{\mathrm{new}}) u_2$, $u_3 \leftarrow u_3 + (G - G^{\mathrm{new}}) u_4$
  $F \leftarrow F^{\mathrm{new}}$, $G \leftarrow G^{\mathrm{new}}$
  Let $\widehat{S}$ denote the blocks where $\widetilde{V}$ and $\widetilde{V}^{\mathrm{new}}$ are different
  $\overline{x}_{\widehat{S}} \leftarrow (u_1)_{\widehat{S}} + (F u_2)_{\widehat{S}}$, $\overline{s}_{\widehat{S}} \leftarrow (u_3)_{\widehat{S}} + (G u_2)_{\widehat{S}}$ ▷ make sure x and $\overline{x}$ are close, similarly for s and $\overline{s}$
end procedure

end datastructure

3.6.1 Proof of Theorem 3.6.1

We follow the proof-sketch as [CLS19]. The proof contains four parts : 1) Definition of X and Y, 2) We need to assume sorting, 3) We provide the definition of potential function, 4) We write the potential function.

Algorithm 3.3 Central Path Maintenance Data Structure - Full Update

datastructure CentralPathMaintenance ▷ Theorem 3.6.1

private : procedure FullUpdate($\overline{W}^{\text{new}}$) ▷ Lemma 3.6.5

$\overline{y}_i \leftarrow v_i^{-1/2} \overline{w}_i^{\text{new}} v_i^{-1/2} - 1$, $\forall i \in [m]$

$r \leftarrow$ the number of indices i such that $\|\overline{y}_i\|_F \geq \epsilon_{mp}$

Let $\overline{\pi} : [m] \to [m]$ be a sorting permutation such that $\|\overline{y}_{\overline{\pi}(i)}\|_F \geq \|\overline{y}_{\overline{\pi}(i+1)}\|_F$

while $1.5 \cdot r < m$ and $\|\overline{y}_{\overline{\pi}(1.5r)}\|_F \geq (1 - 1/\log m)\|\overline{y}_{\overline{\pi}(r)}\|_F$

  $r \leftarrow \min(\lceil 1.5 \cdot r \rceil, m)$

end while

$$v^{\text{new}}_{\overline{\pi}(i)} \leftarrow \begin{cases} \overline{w}^{\text{new}}_{\overline{\pi}(i)} & i \in \{1, 2, \cdots, r\} \\ v_{\overline{\pi}(i)} & i \in \{r+1, \cdots, m\} \end{cases}$$

▷ Compute $M^{\text{new}} = A^\top (A V^{\text{new}} A^\top)^{-1} A$ via Matrix Woodbury

$\Delta \leftarrow V^{\text{new}} - V$ ▷ $\Delta \in \mathbb{R}^{n \times n}$ and $\|\Delta\|_0 = r$

$\Gamma \leftarrow \sqrt{V^{\text{new}}} - \sqrt{V}$

Let $S \leftarrow \overline{\pi}([r])$ be the first r indices in the permutation

Let $M_{*,S} \in \mathbb{R}^{n \times O(r)}$ be the r column-blocks from S of M

Let $M_{S,S}, \Delta_{S,S} \in \mathbb{R}^{O(r) \times O(r)}$ be the r row-blocks and column-blocks from S of M, Δ

$M^{\text{new}} \leftarrow M - M_{*,S} \cdot (\Delta_{S,S}^{-1} + M_{S,S})^{-1} \cdot (M_{*,S})^\top$ ▷ Update M

$Q^{\text{new}} \leftarrow Q + R \cdot (\Gamma \cdot M^{\text{new}}) + R \cdot \sqrt{V} \cdot (M^{\text{new}} - M)$ ▷ Update Q

$\overline{W} \leftarrow \overline{W}^{\text{new}}$, $V \leftarrow V^{\text{new}}$, $M \leftarrow M^{\text{new}}$, $Q \leftarrow Q^{\text{new}}$ ▷ Update in memory

$$\widetilde{v}_i \leftarrow \begin{cases} v_i & \text{if } (1 - \epsilon_{mp}) v_i \preceq \overline{w}_i \preceq (1 + \epsilon_{mp}) v_i \\ w_i & \text{otherwise} \end{cases}$$

$F^{\text{new}} \leftarrow \sqrt{\widetilde{V}} M$, $G^{\text{new}} \leftarrow \frac{1}{\sqrt{\widetilde{V}}} M$

$u_1 \leftarrow u_1 + (F - F^{\text{new}}) u_2$, $u_3 \leftarrow u_3 + (G - G^{\text{new}}) u_4$

$F \leftarrow F^{\text{new}}$, $G \leftarrow G^{\text{new}}$

Let $\widehat{S}$ denote the blocks where $\widetilde{V}$ and $\widetilde{V}^{\text{new}}$ are different

$\overline{x}_{\widehat{S}} \leftarrow (u_1)_{\widehat{S}} + (F u_2)_{\widehat{S}}$, $\overline{s}_{\widehat{S}} \leftarrow (u_3)_{\widehat{S}} + (G u_2)_{\widehat{S}}$ ▷ make sure x and $\overline{x}$ are close, similarly for s and $\overline{s}$

$t^{\text{pre}} \leftarrow t$

end procedure

end datastructure

Algorithm 3.4 Central Path Maintenance Data Structure - Multiply and Move

datastructure CentralPathMaintenance ▷ Theorem 3.6.1

public : procedure MultiplyAndMove(h, t) ▷ Lemma 3.6.9
  Multiply(h, t)
  Move()
end procedure

private : procedure Multiply (h, t) ▷ Lemma 3.6.8
  Let $\widetilde{S}$ be the indices i such that $(1-\epsilon_{mp})v_i \preceq \overline{w}_i \preceq (1+\epsilon_{mp})v_i$ is false.
  $\widetilde{\Delta} \leftarrow \widetilde{V} - V$
  $\widetilde{\Gamma} \leftarrow \sqrt{\widetilde{V}} - \sqrt{V}$
  $\delta_m \leftarrow ((\widetilde{\Delta}^{-1}_{\widetilde{S},\widetilde{S}} + M_{\widetilde{S},\widetilde{S}})^{-1} \cdot ((M_{\widetilde{S},*})^\top \sqrt{\widetilde{V}} h))$ ▷ $|\widetilde{S}| \leq n^a$
  ▷ Compute $\widetilde{\delta}_x = \widetilde{V}^{1/2}(I - R^\top R\widetilde{P})\widetilde{V}^{1/2}h$
  $\widetilde{\delta}_x \leftarrow \widetilde{V}h - \Big((R_l^\top \cdot ((Q_l + R_l \cdot \widetilde{\Gamma} \cdot M) \cdot \sqrt{\widetilde{V}} \cdot h)) - (R_l^\top \cdot ((Q_{l,\widetilde{S}} + R_l \cdot \widetilde{\Gamma} \cdot M_{\widetilde{S},*}) \cdot \delta_m))\Big)$
  ▷ Compute $\widetilde{\delta}_s = t\widetilde{V}^{-1/2}R^\top R\widetilde{P}\widetilde{V}^{1/2}h$
  $\widetilde{\delta}_s \leftarrow t \cdot \widetilde{V}^{-1} \cdot \Big((R_l^\top \cdot ((Q + R_l \cdot \widetilde{\Gamma} \cdot M) \cdot \sqrt{\widetilde{V}} \cdot h)) - (R_l^\top \cdot ((Q_{l,\widetilde{S}} + R_l \cdot \widetilde{\Gamma} \cdot M_{\widetilde{S},*}) \cdot \delta_m))\Big)$
  $l \leftarrow l + 1$ ▷ Increasing the randomness counter, and using the new randomness next time
  ▷ Implicitly maintain $x = x + \widetilde{V}^{1/2}(I - \widetilde{P})\widetilde{V}^{1/2}h$
  $u_1 \leftarrow u_1 + \widetilde{V}h$
  $u_2 \leftarrow u_2 - \sqrt{\widetilde{V}}h + \mathbf{1}_{\widetilde{S}}\delta_m$
  ▷ Implicitly maintain $s = s + t\widetilde{V}^{-1/2}\widetilde{P}\widetilde{V}^{1/2}h$
  $u_3 \leftarrow u_3 + 0$
  $u_4 \leftarrow u_4 - t\sqrt{\widetilde{V}}h + t\mathbf{1}_{\widetilde{S}}\delta_m$
end procedure

private : procedure Move() ▷ Lemma 3.6.7
  if $l > \sqrt{n}$ or $t \geq t^{\text{pre}}/2$ ▷ Variance is large enough
    $x \leftarrow u_1 + Fu_2$, $s \leftarrow u_3 + Fu_4$
    Initialize$(A, x, s, \overline{W}, \epsilon_{mp}, a, b)$ ▷ Algorithm 3.1
  else
    $\overline{x} \leftarrow \overline{x} + \widetilde{\delta}_x$, $\overline{s} \leftarrow \overline{s} + \widetilde{\delta}_s$ ▷ Update $\overline{x}, \overline{s}$
  end if
return $(\overline{x}, \overline{s})$
end procedure

end datastructure

Definition of matrices X and Y. Let us consider the k-th round of the algorithm. For all $i \in [m]$, matrix $\overline{y}_i^{(k)} \in \mathbb{R}^{n_i \times n_i}$ is constructed based on procedure Update (Algorithm 3.2)

:

$$\overline{y}_i^{(k)} = \frac{\overline{w}_i^{(k+1)}}{v_i^{(k)}} - I.$$

and $\overline{\pi}$ is a permutation such that $\|\overline{y}_{\overline{\pi}(i)}^{(k)}\|_F \geq \|\overline{y}_{\overline{\pi}(i+1)}^{(k)}\|_F$.

For the purpose of analysis : for all $i \in [m]$, we define $x_i^{(k)}$, $x_i^{(k)}$ and $y_i^{(k)} \in \mathbb{R}^{n_i \times n_i}$ as follows:

$$x_i^{(k)} = \frac{w_i^{(k)}}{v_i^{(k)}} - I, \quad y_i^{(k)} = \frac{w_i^{(k+1)}}{v_i^{(k)}} - I, \quad x_i^{(k+1)} = \frac{w_i^{(k+1)}}{v_i^{(k+1)}} - I,$$

where $\frac{w_i^{(k)}}{v_i^{(k)}}$ denotes $(v_i^{(k)})^{-1/2} w_i^{(k)} (v_i^{(k)})^{-1/2}$.

It is not hard to observe the difference between $x_i^{(k)}$ and $y_i^{(k)}$ is that w is changing. We call it "w move". Similarly, the difference between $y_i^{(k)}$ and $x_i^{(k+1)}$ is that v is changing. We call it "v move".

For each i, we define β_i as follows

$$\beta_i = \|(w_i^{(k)})^{-1/2} (\mathbb{E}[w_i^{(k+1)}] - w_i^{(k)}) (w_i^{(k)})^{-1/2}\|_F,$$

then one of assumption becomes

$$\sum_{i=1}^m \beta_i^2 \leq C_1^2.$$

Assume sorting for diagonal blocks. Without loss of generality, we can assume the diagonal blocks of matrix $x^{(k)} \in \oplus_{i=1}^m \mathbb{R}^{n_i \times n_i}$ are sorted such that $\|x_i^{(k)}\|_F \geq \|x_{i+1}^{(k)}\|_F$. In [CLS19], $x_i^{(k)}$ is a scalar. They sorted the sequence based on absolute value. In our situation, $x_i^{(k)}$ is a matrix. We sort the sequence based on Frobenius norm. Let τ permutation such that $\|x_{\tau(i)}^{(k+1)}\|_F \geq \|x_{\tau(i+1)}^{(k+1)}\|_F$. Let π denote the permutation such that $\|y_{\pi(i)}^{(k)}\|_F \geq \|y_{\pi(i+1)}^{(k)}\|_F$.

Definition of Potential function. We define three functions g, ψ and Φ_k here. The definition of ψ is different from [CLS19], since we need to handle matrix. The definitions of g and Φ_k are the same as [CLS19].

For the completeness, we still provide a definition of g. Let g be defined as

$$g_i = \begin{cases} n^{-a}, & \text{if } i < n^a; \\ i^{\frac{\omega-2}{1-a}} n^{-\frac{a(\omega-2)}{1-a}}, & \text{otherwise.} \end{cases}$$

In [CLS19], the input of function $\psi : \mathbb{R} \to \mathbb{R}$ has to be a number. We allow matrix here. Let ψ : square matrix $\to \mathbb{R}$ be defined by

$$\psi(x) = \begin{cases} \frac{\|x\|_F^2}{2\epsilon_{mp}}, & \|x\|_F \in [0, \epsilon_{mp}]; \\ \epsilon_{mp} - \frac{(4\epsilon_{mp}^2 - \|x\|_F^2)^2}{18\epsilon_{mp}^3}, & \|x\|_F \in (\epsilon_{mp}, 2\epsilon_{mp}]; \\ \epsilon_{mp}, & \|x\|_F \in (2\epsilon_{mp}, +\infty). \end{cases} \tag{3.26}$$

where $\|x\|_F$ denotes the Frobenius norm of square matrix x, and let $L_1 = \max_x D_x\psi[h]/\|H\|_F$, $L_2 = \max_x D_x^2\psi[h,h]/\|H\|_F^2$ where h is the vectorization of matrix H.

For the completeness, we define the potential at the k-th round by

$$\Phi_k = \sum_{i=1}^{m} g_i \cdot \psi(x_{\tau_k(i)}^{(k)})$$

where $\tau_k(i)$ is the permutation such that $\|x_{\tau_k(i)}^{(k)}\|_F \geq \|x_{\tau_k(i+1)}^{(k)}\|_F$. (Note that in [CLS19] $\|\cdot\|_F$ should be $|\cdot|$.)

Rewriting the potential, and bounding it. Following the ideas in [CLS19], we can rewrite $\Phi_{k+1} - \Phi_k$ into two terms: the first term is w move, and the second term is v move.

For the completeness, we still provide a proof.

$$\begin{aligned}\Phi_{k+1} - \Phi_k &= \sum_{i=1}^{m} g_i \cdot \left(\psi(x_{\tau(i)}^{(k+1)}) - \psi(x_i^{(k)})\right) \\ &= \sum_{i=1}^{m} g_i \cdot \underbrace{\left(\psi(y_{\pi(i)}^{(k)}) - \psi(x_i^{(k)})\right)}_{W \text{ move}} - \sum_{i=1}^{m} g_i \cdot \underbrace{\left(\psi(y_{\pi(i)}^{(k)}) - \psi(x_{\tau(i)}^{(k+1)})\right)}_{V \text{ move}}\end{aligned}$$

Using Lemma 3.6.10, we can bound the first term. Using Lemma 3.6.12, we can bound the second term.

3.6.2 Initialization time, update time, query time, move time, multiply time

Remark 3.6.2. In terms of implementing this data-structure, we only need three operations INITIALIZE, UPDATE, and QUERY. However, in order to make the proof more understoodable, we split UPDATE into many operations : FULLUPDATE, PARTIALUPDATE, MULTIPLY and MOVE. We give a list of operations in Table 3.2.

Lemma 3.6.2 (Initialization)**.** *The initialization time of data-structure* CENTRALPATHMAINTENANCE *(Algorithm 3.1) is* $O(n^{\omega+o(1)})$.

Proof. The running time is mainly dominated by two parts, the first part is computing $A^\top (AVA^\top)^{-1} A$, this takes $O(n^2 d^{\omega-2})$ time.

The second part is computing $R\sqrt{\widetilde{V}}M$. This takes $O(n^{\omega+o(1)})$ time.

□

Lemma 3.6.3 (Update time)**.** *The update time of data-structure* CENTRALPATHMAINTENANCE *(Algorithm 3.2) is* $O(rg_r n^{2+o(1)})$ *where* r *is the number of indices we updated in* V.

Proof. It is trivially follows from combining Lemma 3.6.4 and Lemma 3.6.5. □

Lemma 3.6.4 (Partial Update time)**.** *The partial update time of data-structure* CENTRALPATHMAINTENANCE *(Algorithm 3.2) is* $O(n^{1+a})$.

Proof. We first analyze the running time of F update, the update equation of F in algorithm

is

$$F^{\text{new}} \leftarrow F + ((\widetilde{V}^{\text{new}})^{1/2} - (\widetilde{V})^{1/2})M$$
$$F \leftarrow F^{\text{new}}$$

which can be implemented as

$$F \leftarrow F + ((\widetilde{V}^{\text{new}})^{1/2} - (\widetilde{V})^{1/2})M$$

where we only need to change n^a row-blocks of F. It takes $O(n^{1+a})$ time.

Similarly, for the update time of G.

Next we analyze the update time of u_1, the update equation of u_1 is

$$u_1 \leftarrow u_1 + (F - F^{\text{new}})u_2$$

Note that the difference between F and F^{new} is only n^a row-blocks, thus it takes n^{1+a} time to update.

Finally we analyze the update time of $\overline{x}$. Let $\widehat{S}$ denote the blocks where $\widetilde{V}$ and $\widetilde{V}^{\text{new}}$ are different.

$$\overline{x}_{\widehat{S}} \leftarrow (u_1)_{\widehat{S}} + (Fu_2)_{\widehat{S}}$$

This also can be done in n^{1+a} time, since $\widehat{S}$ indicates only n^a blocks.

Therefore, the overall running time is $O(n^{1+a})$.

□

Lemma 3.6.5 (Full Update time)**.** *The full update time of data-structure* CentralPathMaintenance *(Algorithm 3.3) is* $O(rg_r n^{2+o(1)})$ *where* r *is the number of indices we updated in* V*.*

Proof. The update equation we use for Q is

$$Q^{\text{new}} \leftarrow Q + R \cdot (\Gamma \cdot M^{\text{new}}) + R \cdot \sqrt{V} \cdot (M^{\text{new}} - M).$$

It can be re-written as

$$Q^{\text{new}} \leftarrow Q + R \cdot (\Gamma \cdot M^{\text{new}}) + R \cdot \sqrt{V} \cdot (-M_{*,S} \cdot (\Delta_{S,S}^{-1} + M_{S,S})^{-1} \cdot (M_{*,S})^\top)$$

The running time of computing second term is multiplying a $n \times r$ matrix with another $r \times n$ matrix. The running time of computing third term is also dominated by multiplying a $n \times r$ matrix with another $r \times n$ matrix.

Thus running time of processing Q update is the same as the processing M update.

For the running time of other parts, it is dominated by the time of updating M and Q.

Therefore, the rest of the proof is almost the same as Lemma 5.4 in [CLS19], we omitted here. □

Lemma 3.6.6 (Query time)**.** *The query time of data-structure* CentralPathMaintenance *(Algorithm 3.1) is $O(n)$ time.*

Proof. This takes only $O(n)$ time, since we stored $\overline{x}$ and $\overline{s}$. □

Lemma 3.6.7 (Move time)**.** *The move time of data-structure* CentralPathMaintenance *(Algorithm 3.4) is $O(n^{\omega+o(1)})$ time in the worst case, and is $O(n^{\omega-1/2+o(1)})$ amortized cost per iteration.*

Proof. In one case, it takes only $O(n)$ time. For the other case, the running time is dominated by INITIALIZE, which takes $n^{\omega+o(1)}$ by Lemma 3.6.5.

□

Lemma 3.6.8 (Multiply time)**.** *The multiply time of data-structure* CENTRALPATHMAINTENANCE *(Algorithm 3.4) is* $O(nb + n^{1+a+o(1)})$ *for dense vector* $\|h\|_0 = n$*, and is* $O(nb + n^{a\omega+o(1)} + n^a\|h\|_0)$ *for sparse vector* h*.*

Proof. We first analyze the running time of computing vector δ_m, the equation is

$$\delta_m \leftarrow \left(((\widetilde{\Delta}_{\widetilde{S},\widetilde{S}})^{-1} + M_{\widetilde{S},\widetilde{S}})^{-1} \cdot (M_{\widetilde{S},*})^\top \sqrt{\widetilde{V}} h\right)$$

where $\widetilde{\Delta} = \widetilde{V} - V$. Let $\widetilde{r} = \sum_{i \in \widetilde{S}} n_i = O(r)$ where r is the number of blocks are different in $\widetilde{V}$ and V.

It contains several parts:

1. Computing $\widetilde{M}_{\widetilde{S}}^\top \cdot (\sqrt{\widetilde{V}} h) \in \mathbb{R}^{\widetilde{r}}$ takes $O(\widetilde{r})\|h\|_0$.

2. Computing $(\widetilde{\Delta}^{-1}_{\widetilde{S},\widetilde{S}} + M_{\widetilde{S},\widetilde{S}})^{-1} \in \mathbb{R}^{O(\widetilde{r}) \times O(\widetilde{r})}$ that is the inverse of a $O(\widetilde{r}) \times O(\widetilde{r})$ matrix takes $O(\widetilde{r}^{\omega+o(1)})$ time.

3. Computing matrix-vector multiplication between $O(\widetilde{r}) \times O(\widetilde{r})$ matrix $((\widetilde{\Delta}_{\widetilde{S},\widetilde{S}} + M_{\widetilde{S},\widetilde{S}})^{-1})$ and $O(\widetilde{r}) \times 1$ vector $((\widetilde{M}_{\widetilde{S},*})^\top \sqrt{\widetilde{V}} h)$ takes $O(\widetilde{r}^2)$ time.

Thus, the running time of computing δ_m is

$$O(\widetilde{r}\|h\|_0 + \widetilde{r}^{\omega+o(1)} + \widetilde{r}^2) = O(\widetilde{r}\|h\|_0 + \widetilde{r}^{\omega+o(1)}).$$

Next, we want to analyze the update equation of $\widetilde{\delta}_x$

$$\widetilde{\delta}_x \leftarrow \widetilde{V} h - \left((R_l^\top \cdot ((Q_l + R_l \sqrt{\widetilde{\Delta}} M) \cdot \sqrt{\widetilde{V}} \cdot h)) - (R_l^\top \cdot ((Q_{l,\widetilde{S}} + R_l \widetilde{\Gamma} M_{\widetilde{S}}) \cdot \delta_m))\right)$$

where $\widetilde{\Gamma} = \sqrt{\widetilde{V}} - \sqrt{V}$ has $O(r)$ non-zero blocks.

It is clear that the running time is dominated by the second term in the equation. We only focus on that term.

1. Computing $R_l^\top Q_l \sqrt{\widetilde{V}} h$ takes $O(bn)$ time, because $Q_l, R_l \in \mathbb{R}^{b \times n}$.

2. Computing $R_l^\top R_l \sqrt{\widetilde{\Delta}} M \sqrt{\widetilde{V}} h$ takes $O(bn + b\widetilde{r} + \widetilde{r}\|h\|_0)$ time. The reason is, computing $\sqrt{\widetilde{\Delta}} M \sqrt{\widetilde{V}} h$ takes $\widetilde{r}\|h\|_0$ time, computing $R_l \cdot (\sqrt{\widetilde{\Delta}} M \sqrt{\widetilde{V}} h)$ takes $b\widetilde{r}$, then finally computing $R_l^\top \cdot (R_l \sqrt{\widetilde{\Delta}} M \sqrt{\widetilde{V}} h)$ takes nb.

Last, the update equation of u_1, u_2, u_3, u_4 only takes the $O(n)$ time.

Finally, we note that $r \leq O(n^a)$ due to the guarantee of FullUpdate and PartialUpdate.

Thus, overall the running time of the Multiply is

$$\begin{aligned} O(\widetilde{r}\|h\|_0 + \widetilde{r}^{\omega+o(1)} + \widetilde{r}^2 + b\widetilde{r} + nb) &= O(\widetilde{r}\|h\|_0 + \widetilde{r}^{\omega+o(1)} + nb) \\ &= O(r\|h\|_0 + r^{\omega+o(1)} + nb) \\ &= O(n^a\|h\|_0 + n^{a\omega+o(1)} + nb) \end{aligned}$$

where the first step follows from $b\widetilde{r} \leq nb$ and $\widetilde{r}^2 \leq \widetilde{r}^{\omega+o(1)}$, and the second step follows from $\widetilde{r} = O(r)$, and the last step follows from $r = O(n^a)$.

If h is the dense vector, then the overall time is

$$O(nb + n^{1+a} + n^{a\omega+o(1)}).$$

Based on Lemma 5.5 in [CLS19], we know that $a\omega \leq 1 + a$. Thus, it becomes $O(nb + n^{1+a+o(1)})$ time.

If h is a sparse vector, then the overall time is

$$O(nb + n^a \|h\|_0 + n^{a\omega+o(1)}).$$

□

Lemma 3.6.9 (MultiplyMove)**.** *The running time of* MultiplyMove *(Algorithm 3.6.9) is the* Multiply *time plus* Move *time.*

3.6.3 Bounding W move

The goal of this section is to analyze the movement of W. [CLS19] provided a scalar version of W move, here we provide a matrix version.

Lemma 3.6.10 (W move, matrix version of Lemma 5.7 in [CLS19])**.**

$$\sum_{i=1}^{m} g_i \cdot \mathbb{E}\left[\psi(y_{\pi(i)}^{(k)}) - \psi(x_{\pi(i)}^{(k)})\right] = O(C_1 + C_2/\epsilon_{mp}) \cdot \sqrt{\log n} \cdot (n^{-a/2} + n^{\omega-5/2}).$$

Proof. In scalar version, [CLS19] used absolute ($|\cdot|$) to measure each $x_i^{(k)}$. In matrix version, we use Frobenius norm ($\|\|_F$) to measure each $x_i^{(k)}$. Let $I \subseteq [m]$ be the set of indices such that $\|x_i^{(k)}\|_F \leq 1$. We separate the term into two :

$$\sum_{i=1}^{m} g_i \cdot \mathbb{E}[\psi(y_{\pi(i)}^{(k)}) - \psi(x_{\pi(i)}^{(k)})] = \sum_{i \in I} g_{\pi^{-1}(i)} \cdot \mathbb{E}[\psi(y_i^{(k)}) - \psi(x_i^{(k)})] + \sum_{i \in I^c} g_{\pi^{-1}(i)} \cdot \mathbb{E}[\psi(y_i^{(k)}) - \psi(x_i^{(k)})].$$

Case 1. Let us consider the terms from I.

Let $\mathrm{vec}(y_i^{(k)})$ denote the vectorization of matrix $y_i^{(k)}$. Similarly, $\mathrm{vec}(x_i^{(k)})$ denotes the vectorization of $x_i^{(k)}$. Mean value theorem shows that

$$\begin{aligned}
\psi(y_i^{(k)}) - \psi(x_i^{(k)}) &= \langle \psi'(x_i^{(k)}), y_i^{(k)} - x_i^{(k)} \rangle + \frac{1}{2} \mathrm{vec}(y_i^{(k)} - x_i^{(k)})^\top \psi''(\zeta) \mathrm{vec}(y_i^{(k)} - x_i^{(k)}) \\
&\leq \langle \psi'(x_i^{(k)}), y_i^{(k)} - x_i^{(k)} \rangle + \frac{L_2}{2} \|y_i^{(k)} - x_i^{(k)}\|_F^2 \\
&= \langle \psi'(x_i^{(k)}), (v_i^{(k)})^{-1/2} (w_i^{(k+1)} - w_i^{(k)}) (v_i^{(k)})^{-1/2} \rangle \\
&\quad + \frac{L_2}{2} \|(v_i^{(k)})^{-1/2} (w_i^{(k+1)} - w_i^{(k)}) (v_i^{(k)})^{-1/2}\|_F^2
\end{aligned}$$

where the second step follows from definition of L_2 (see Part 4 of Lemma 3.6.14).

Taking conditional expectation given $w^{(k)}$ on both sides

$$\begin{aligned}
\mathbb{E}[\psi(y_i^{(k)}) - \psi(x_i^{(k)})] &\leq \langle \psi'(x_i^{(k)}), (v_i^{(k)})^{-1/2}(\mathbb{E}[w_i^{(k+1)}] - w_i^{(k)})(v_i^{(k)})^{-1/2}\rangle \\
&\quad + \frac{L_2}{2}\,\mathbb{E}[\|(v_i^{(k)})^{-1/2}(w_i^{(k+1)} - w_i^{(k)})(v_i^{(k)})^{-1/2}\|_F^2] \\
&\leq L_1\|(v_i^{(k)})^{-1/2}(\mathbb{E}[w_i^{(k+1)}] - w_i^{(k)})(v_i^{(k)})^{-1/2}\|_F \\
&\quad + \frac{L_2}{2}\,\mathbb{E}[\|(v_i^{(k)})^{-1/2}(w_i^{(k+1)} - w_i^{(k)})(v_i^{(k)})^{-1/2}\|_F^2] \\
&\leq L_1\|(v_i^{(k)})^{-1/2}(w_i^{(k)})^{1/2}\|^2 \cdot \|(w_i^{(k)})^{-1/2}(\mathbb{E}[w_i^{(k+1)}] - w_i^{(k)})(w_i^{(k)})^{-1/2}\|_F \\
&\quad + \frac{L_2}{2}\|(v_i^{(k)})^{-1/2}(w_i^{(k)})^{1/2}\|^4 \cdot \mathbb{E}[\|(w_i^{(k)})^{-1/2}(w_i^{(k+1)} - w_i^{(k)})(w_i^{(k)})^{-1/2}\|_F^2] \\
&= L_1\|(v_i^{(k)})^{-1/2}(w_i^{(k)})^{1/2}\|^2 \cdot \beta_i + \frac{L_2}{2}\|(v_i^{(k)})^{-1/2}(w_i^{(k)})^{1/2}\|^4 \cdot \gamma_i \qquad (3.27)
\end{aligned}$$

where the second step follows from definition of L_2 (see Part 4 of Lemma 3.6.14), the third step follows from $\|AB\|_F \leq \|A\|_F \cdot \|B\|$, and the last step follows from defining β_i and γ_i as follows:

$$\begin{aligned}
\beta_i &= \left\|(w_i^{(k)})^{-1/2}(\mathbb{E}[w_i^{(k+1)}] - w_i^{(k)})(w_i^{(k)})^{-1/2}\right\|_F \\
\gamma_i &= \mathbb{E}\left[\left\|(w_i^{(k)})^{-1/2}(w_i^{(k+1)} - w_i^{(k)})(w_i^{(k)})^{-1/2}\right\|_F^2\right].
\end{aligned}$$

To upper bound $\sum_{i\in I} g_{\pi^{-1}(i)}\,\mathbb{E}[\psi(y_i^{(k)}) - \psi(x_i^{(k)})]$, we need to bound the following two terms,

$$\sum_{i\in I} g_{\pi^{-1}(i)} L_1 \left\|(v_i^{(k)})^{-1/2}(w_i^{(k)})^{1/2}\right\|^2 \beta_i, \text{ and } \sum_{i\in I} g_{\pi^{-1}(i)} \frac{L_2}{2} \left\|(v_i^{(k)})^{-1/2}(w_i^{(k)})^{1/2}\right\|^4 \gamma_i. \qquad (3.28)$$

For the first term (which is related to β) in Eq. (3.28), we have

$$\begin{aligned}\sum_{i\in I} g_{\pi^{-1}(i)} L_1 \|(v_i^{(k)})^{-1/2}(w_i^{(k)})^{1/2}\|^2 \beta_i &\leq \left(\sum_{i\in I}\left(g_{\pi^{-1}(i)} L_1 \|(v_i^{(k)})^{-1/2}(w_i^{(k)})^{1/2}\|^2\right)^2 \sum_{i\in I}\beta_i^2\right)^{1/2} \\ &\leq O(L_1)\left(\sum_{i=1}^n g_i^2 \cdot C_1^2\right)^{1/2} \\ &= O(C_1 L_1 \|g\|_2). \end{aligned} \tag{3.29}$$

where the first step follows from Cauchy-Schwarz inequality, the second step follows from $n_i = O(1)$ and $\|(v_i^{(k)})^{-1/2}(w_i^{(k)})^{1/2}\|^2 = O(1)$.

For the second term (which is related to γ) in Eq. (3.28), we have

$$\sum_{i\in I} g_{\pi^{-1}(i)} \frac{L_2}{2} \|(v_i^{(k)})^{-1/2}(w_i^{(k)})^{1/2}\|^4 n_i \gamma_i \leq O(L_2) \cdot \sum_{i=1}^m g_i \cdot \gamma_i = O(C_2 L_2 \|g\|_2). \tag{3.30}$$

Putting Eq. (3.27), Eq. (3.29) and Eq. (3.30) together, and using several facts $L_1 = O(1)$, $L_2 = O(1/\epsilon_{mp})$ (from part 4 of Lemma 3.6.14) and $\|g\|_2 \leq \sqrt{\log n} \cdot O(n^{-a/2} + n^{\omega-5/2})$ (from Lemma 3.6.11) gives us

$$\sum_{i\in I} g_{\pi^{-1}(i)} \cdot \mathbb{E}[\psi(y_i^k) - \psi(x_i^{(k)})] \leq O(C_1 + C_2/\epsilon_{mp}) \cdot \sqrt{\log n} \cdot (n^{-a/2} + n^{\omega-5/2}).$$

(Note that, the above Equation is the same as [CLS19].)

Case 2. Let us consider the terms from I^c.

For each $i \in I^c$, we know $\|x_i^{(k)}\|_F \geq 1$. We observe that $\psi(x)$ is constant for $\|x\|_F^2 \geq (2\epsilon_{mp})^2$, where $\epsilon_{mp} \leq 1/4$. If $\|y_i^{(k)}\|_F \geq 1/2$, then $\psi(y_i^{(k)}) - \psi(x_i^{(k)}) = 0$. Therefore, we only need to focus on the $i \in I^c$ such that $\|y_i^{(k)}\|_F < 1/2$.

For each $i \in I^c$ with $\|y_i^{(k)}\|_F < 1/2$, we have

$$\begin{aligned}
\frac{1}{2} &< \|y_i^{(k)} - x_i^{(k)}\|_F \\
&= \|(v_i^{(k)})^{-1/2}(w_i^{(k+1)} - w_i^{(k)})(v_i^{(k)})^{-1/2}\|_F \\
&= \|(v_i^{(k)})^{-1/2}(w_i^{(k)})^{1/2} \cdot (w_i^{(k)})^{-1/2}(w_i^{(k+1)} - w_i^{(k)})(w_i^{(k)})^{-1/2} \cdot (w_i^{(k)})^{1/2}(v_i^{(k)})^{-1/2}\|_F \\
&\leq \|(v_i^{(k)})^{-1/2}(w_i^{(k)})^{1/2}\| \cdot \|(w_i^{(k)})^{-1/2}(w_i^{(k+1)} - w_i^{(k)})(w_i^{(k)})^{-1/2}\|_F \cdot \|(w_i^{(k)})^{1/2}(v_i^{(k)})^{-1/2}\| \\
&= \|(v_i^{(k)})^{-1/2}w_i^{(k)}(v_i^{(k)})^{-1/2}\| \cdot \|(w_i^{(k)})^{-1/2}(w_i^{(k+1)} - w_i^{(k)})(w_i^{(k)})^{-1/2}\|_F \\
&\leq \frac{3}{2}\|(w_i^{(k)})^{-1/2}(w_i^{(k+1)} - w_i^{(k)})(w_i^{(k)})^{-1/2}\|_F \qquad (3.31)
\end{aligned}$$

where the last step follows from $\|y_i^{(k)}\|_F = \|\frac{w_i^{(k+1)}}{v_i^{(k)}} - I\|_F \leq 1/2$.

It is obvious that Eq. (3.31) implies

$$\|(w_i^{(k)})^{-1/2}(w_i^{(k+1)} - w_i^{(k)})(w_i^{(k)})^{-1/2}\|_F > 1/3 > 1/4.$$

But this is impossible, since we assume it is $\leq 1/4$.

Thus, we have

$$\sum_{i \in I^c} g_{\pi^{-1}(i)} \cdot \mathbb{E}[\psi(y_i^{(k)}) - \psi(x_i^{(k)})] = 0.$$

□

We state a Lemma that was proved in previous work [CLS19].

Lemma 3.6.11 (Lemma 5.8 in [CLS19])**.**

$$\left(\sum_{i=1}^{n} g_i^2\right)^{1/2} \leq \sqrt{\log n} \cdot O(n^{-a/2} + n^{\omega - 5/2})$$

3.6.4 Bounding V move

In previous work, [CLS19] only handled the movement of V in scalar version. Here, the goal of is to understand the movement of V in matrix version. We start to give some definitions about block diagonal matrices.

Definition 3.6.1. We define block diagonal matrices $X^{(k)}$, $Y^{(k)}$, $X^{(k+1)}$ and $\overline{Y}^{(k)} \otimes_{i\in[m]} \mathbb{R}^{n_i \times n_i}$ as follows

$$x_i^{(k)} = \frac{w_i^{(k)}}{v_i^{(k)}} - I, \quad y_i^{(k)} = \frac{w_i^{(k+1)}}{v_i^{(k)}} - I, \quad x_i^{(k+1)} = \frac{w_i^{(k+1)}}{v_i^{(k+1)}} - I, \quad \overline{y}_i^{(k)} = \frac{\overline{w}_i^{(k+1)}}{v_i^{(k)}} - I.$$

Let ϵ_w denote the error between W and $\overline{W}$

$$\|W_i^{-1/2}(\overline{W}_i - W_i)W_i^{-1/2}\|_F \leq \epsilon_w.$$

Lemma 3.6.12 (V move, matrix version of Lemma 5.9 in [CLS19])**.** *We have,*

$$\sum_{i=1}^{n} g_i \cdot \left(\psi(y_{\pi(i)}^{(k)}) - \psi(x_{\tau(i)}^{(k+1)})\right) \geq \Omega(\epsilon_{mp} r_k g_{r_k} / \log n).$$

Proof. To prove the Lemma, similarly as [CLS19], we will split the proof into two cases.

Before getting into the details of each case, let us first understand several simple facts which are useful in the later proof. Note that from the definition of the algorithm, we only change the block if $\|\overline{y}_i^{(k)}\|_F$ is larger than the error between w_i and $\overline{w}_i$. Hence, all the changes only decreases the norm, namely $\psi(y_i^{(k)}) \geq \psi(x_i^{(k+1)})$ for all i. So is their sorted version $\psi(y_{\pi(i)}^{(k)}) \geq \psi(x_{\tau(i)})^{(k+1)}$ for all i.

Case 1. The procedure exits the while loop when $1.5r_k \geq n$.

Let u^* denote the largest u such that $\|\overline{y}^{(k)}_{\overline{\pi}(u)}\|_F \geq \epsilon_{mp}$.

If $u^* = r_k$, we have that

$$\|\overline{y}^{(k)}_{\overline{\pi}(r_k)}\|_F \geq \epsilon_{mp} \geq \epsilon_{mp}/100.$$

If $u^* \neq r_k$, using the condition of the loop, we have that

$$\begin{aligned}\|\overline{y}^{(k)}_{\overline{\pi}(r_k)}\|_F &\geq (1 - 1/\log n)^{\log_{1.5} r_k - \log_{1.5} u^*} \cdot \|\overline{y}^{(k)}_{\overline{\pi}(u^*)}\|_F \\ &\geq (1 - 1/\log n)^{\log_{1.5} n} \cdot \epsilon_{mp} \\ &\geq \epsilon_{mp}/100.\end{aligned}$$

where the last step follows from $n \geq 4$.

Recall the definition of $x^{(k+1)}_{\tau(i)}$. We can lower bound the LHS in the Lemma statement in the following sense,

$$\begin{aligned}\sum_{i=1}^{n} g_i \cdot \left(\psi(y^{(k)}_{\pi(i)}) - \psi(x^{(k+1)}_{\tau(i)})\right) &\geq \sum_{i=n/3+1}^{2n/3} g_i \cdot \left(\psi(y^{(k)}_{\pi(i)}) - \psi(x^{(k+1)}_{\tau(i)})\right) \\ &\geq \sum_{i=n/3+1}^{2n/3} g_i \cdot (\Omega(\epsilon_{mp}) - O(\epsilon_w)) \\ &\geq \sum_{i=n/3+1}^{2n/3} g_i \cdot \Omega(\epsilon_{mp}) \\ &= \Omega(r_k g_{r_k} \epsilon_{mp}).\end{aligned}$$

where the second step follows from $\|y^{(k)}_{\pi(i)}\|_F \geq \|y^{(k)}_{\pi(r_k)}\|_F \geq (1 - O(\epsilon_w))\|\overline{y}^{(k)}_{\overline{\pi}(r_k)}\|_F \geq \epsilon_{mp}/200$ for all $i < 2n/3$.

Case 2. The procedure exits the while loop when $1.5r_k < n$ and $\|\overline{y}^{(k)}_{\overline{\pi}(1.5r_k)}\|_F < (1 - 1/\log n)\|\overline{y}^{(k)}_{\overline{\pi}(r_k)}\|_F$.

Using the same argument as Case 1, we have

$$\|\overline{y}^{(k)}_{\overline{\pi}(r_k)}\|_F \geq \epsilon_{mp}/100.$$

Using Part 3 of Lemma 3.6.14 and the following fact

$$\|\overline{y}^{(k)}_{\overline{\pi}(1.5r)}\|_F < \min\left(\epsilon_{mp}, \|\overline{y}^{(k)}_{\overline{\pi}(r)}\|_F \cdot (1 - 1/\log n)\right),$$

we can show that

$$\psi(\overline{y}^{(k)}_{\overline{\pi}(1.5r)}) - \psi(\overline{y}^{(k)}_{\overline{\pi}(r)}) = \Omega(\epsilon_{mp}/\log n). \tag{3.32}$$

Now the question is, how to relax $\psi(\overline{y}^{(k)}_{\overline{\pi}(1.5r)})$ to $\psi(y^{(k)}_{\pi(1.5r)})$ and how to relax $\psi(\overline{y}^{(k)}_{\overline{\pi}(r)})$ to $\psi(y^{(k)}_{\pi(r)})$

Note that $\|y^{(k)}_i\|_F \geq \|x^{(k+1)}_i\|_F$ for all i. Hence, we have $\psi(y^{(k)}_{\pi(i)}) \geq \psi(x^{(k+1)}_{\tau(i)})$ for all i.

Recall the definition of y, $\overline{y}$, π and $\overline{\pi}$,

$$y^{(k)}_i = \frac{w^{(k+1)}_i}{v^{(k)}_i} - I, \quad \overline{y}^{(k)}_i = \frac{\overline{w}^{(k+1)}_i}{v^{(k)}_i} - I.$$

and π and $\overline{\pi}$ denote the permutations such that $\|y^{(k)}_{\pi(i)}\|_F \geq \|y^{(k)}_{\pi(i+1)}\|_F$ and $\|\overline{y}^{(k)}_{\overline{\pi}(i)}\|_F \geq \|\overline{y}^{(k)}_{\overline{\pi}(i+1)}\|_F$.

Using Fact 3.6.13 and $\|\cdot\|_2 = \Theta(1)\|\cdot\|_F$ when the matrix has constant dimension

$$\|y^{(k)}_{\pi(i)} - \overline{y}^{(k)}_{\overline{\pi}(i)}\|_F \leq O(\epsilon_w).$$

where ϵ_w is the error between W and $\overline{W}$.

Next, $\forall i$, we have

$$\psi(y^{(k)}_{\pi(i)}) = \psi(\overline{y}^{(k)}_{\overline{\pi}(i)}) \pm O(\epsilon_w \epsilon_{mp}) \tag{3.33}$$

Next, we note that all the blocks the algorithm updated must lies in the range $i = 1, \cdots, \frac{3r_k}{2} - 1$. After the update, the error of r_k of these block becomes so small that its rank will much higher than r_k. Hence, $r_k/2$ of the unchanged blocks in the range $i = 1, \cdots, \frac{3r_k}{2}$ will move earlier in the rank. Therefore, the $r_k/2$-th element in $x^{(k+1)}$ must be larger than the $\frac{3}{2}r_k$-th element in $y^{(k)}$. In short, we have $\psi(x^{(k+1)}_{\tau(i)}) \leq \psi(y^{(k)}_{\pi(1.5r_k)})$ for all $i \geq r_k/2$.

Putting it all together, we have

$$\begin{aligned}
&\sum_{i=1}^{n} g_i \cdot \left(\psi(y^{(k)}_{\pi(i)}) - \psi(x^{(k+1)}_{\tau(i)})\right) \\
\geq &\sum_{i=r_k/2}^{r_k} g_i \cdot \left(\psi(y^{(k)}_{\pi(i)}) - \psi(x^{(k+1)}_{\tau(i)})\right) \\
\geq &\sum_{i=r_k/2}^{r_k} g_i \cdot \left(\psi(y^{(k)}_{\pi(i)}) - \psi(y^{(k+1)}_{\pi(1.5r_k)})\right) && \text{by } \psi(x^{(k+1)}_{\tau(i)}) \leq \psi(y^{(k)}_{\pi(1.5r_k)}), \forall i \geq r_k/2 \\
\geq &\sum_{i=r_k/2}^{r_k} g_i \cdot \left(\psi(\overline{y}^{(k)}_{\overline{\pi}(i)}) - \psi(\overline{y}^{(k+1)}_{\overline{\pi}(1.5r_k)}) - O(\epsilon_w \epsilon_{mp})\right) && \text{by (3.33)} \\
\geq &\sum_{i=r_k/2}^{r_k} g_i \cdot \left(\psi(\overline{y}^{(k)}_{\overline{\pi}(r_k)}) - \psi(\overline{y}^{(k+1)}_{\overline{\pi}(1.5r_k)}) - O(\epsilon_w \epsilon_{mp})\right) && \text{by } \psi(\overline{y}^{(k)}_{\overline{\pi}(i)}) \geq \psi(\overline{y}^{(k)}_{\overline{\pi}(r_k)}), \forall i \in [r_k/2, r_k] \\
\geq &\sum_{i=r_k/2}^{r_k} g_{r_k} \cdot \left(\Omega(\frac{\epsilon_{mp}}{\log n}) - O(\epsilon_w \epsilon_{mp})\right) && \text{by (3.32)} \\
\geq &\sum_{i=r_k/2}^{r_k} g_{r_k} \cdot \Omega(\frac{\epsilon_{mp}}{\log n}) && \text{by } \epsilon_w < O(1/\log n) \\
= &\, \Omega\left(\epsilon_{mp} r_k g_{r_k} / \log n\right).
\end{aligned}$$

Therefore, we complete the proof. □

Fact 3.6.13. *Given two length n positive vectors a, b. Let a be sorted such that $a_i \geq a_{i+1}$. Let π denote the permutation such that $b_{\pi(i)} \geq b_{\pi(i+1)}$. If for all $i \in [n]$, $|a_i - b_i| \leq \epsilon a_i$. Then for all $i \in [n]$, $|a_i - b_{\pi(i)}| \leq \epsilon a_i$.*

Proof. Case 1. $\pi(i) = i$. This is trivially true.

Case 2. $\pi(i) < i$. We have

$$b_{\pi(i)} \geq b_i \geq (1 - \epsilon)a_i$$

Since $\pi(i) < i$, we know that there exists a $j > i$ such that $\pi(j) < \pi(i)$. Then we have

$$b_{\pi(i)} \leq b_{\pi(j)} \leq (1 + \epsilon)a_j \leq (1 + \epsilon)a_i$$

Combining the above two inequalities, we have $(1 - \epsilon)a_i \leq b_{\pi(i)} \leq (1 + \epsilon)a_i$.

Case 3. $\pi(i) > i$. We have

$$b_{\pi(i)} \leq b_i \leq (1 + \epsilon)a_i$$

Since $\pi > i$, we know that there exists $j < i$ such that $\pi(j) > \pi(i)$. Then we have

$$b_{\pi(i)} \geq b_{\pi(j)} \geq (1 - \epsilon)a_j \geq (1 - \epsilon)a_i.$$

Combining the above two inequalities gives us $(1 - \epsilon)a_i \leq b_{\pi(i)} \leq (1 + \epsilon)a_i$.

Therefore, putting all the three cases together completes the proof.

□

3.6.5 Potential function ψ

[CLS19] used a scalar version potential function. Here, we generalize it to the matrix version.

Lemma 3.6.14 (Matrix version of Lemma 5.10 in [CLS19])**.** *Let function* ψ : square matrix $\to \mathbb{R}$ *(defined as Eq. (3.26)) satisfies the following properties :*

1. *Symmetric* $(\psi(x) = \psi(-x))$ *and* $\psi(0) = 0$
2. *If* $\|x\|_F \geq \|y\|_F$*, then* $\psi(x) \geq \psi(y)$
3. $|f'(x)| = \Omega(1/\epsilon_{mp}), \forall x \in [(0.01\epsilon_{mp})^2, \epsilon_{mp}^2]$
4. $L_1 \stackrel{\text{def}}{=} \max_x \frac{D_x\psi[H]}{\|H\|_F} = 2$ *and* $L_2 \stackrel{\text{def}}{=} \max_x \frac{D_x^2\psi[H,H]}{\|H\|_F^2} = 10/\epsilon_{mp}$
5. $\psi(x)$ *is a constant for* $\|x\|_F \geq 2\epsilon_{mp}$

Proof. Let $f : \mathbb{R}_+ \to \mathbb{R}$ be defined as

$$f(x) = \begin{cases} \frac{x^2}{2\epsilon_{mp}^3}, & x \in [0, \epsilon_{mp}^2]; \\ \epsilon_{mp} - \frac{(4\epsilon_{mp}^2 - x)^2}{18\epsilon_{mp}^3}, & x \in (\epsilon_{mp}^2, 4\epsilon_{mp}^2]; \\ \epsilon_{mp}, & x \in (4\epsilon_{mp}^2, +\infty). \end{cases}$$

We can see that

$$f(x)' = \begin{cases} \frac{x}{\epsilon_{mp}^3}, & x \in [0, \epsilon_{mp}^2]; \\ \frac{4\epsilon_{mp}^2 - x}{9\epsilon_{mp}^3}, & x \in (\epsilon_{mp}^2, 4\epsilon_{mp}^2]; \\ 0, & x \in (4\epsilon_{mp}^2, +\infty). \end{cases} \quad \text{and} \quad f(x)'' = \begin{cases} \frac{1}{\epsilon_{mp}^3}, & x \in [0, \epsilon_{mp}^2]; \\ -\frac{1}{9\epsilon_{mp}^3}, & x \in (\epsilon_{mp}^2, 4\epsilon_{mp}^2]; \\ 0, & |x| \in (4\epsilon_{mp}^2, +\infty). \end{cases}$$

It implies that $\max_x |f(x)'| \leq \frac{1}{\epsilon_{mp}}$ and $\max_x |f(x)''| \leq \frac{1}{\epsilon_{mp}^3}$. Let $\psi(x) = f(\|X\|_F^2)$.

Proof of Part 1,2 and 5. These proofs are pretty standard from definition of ψ.

Proof of Part 3. This is trivially following from definition of scalar function f.

Proof of Part 4. By chain rule, we have

$$D_x\psi[h] = 2f'(\|X\|_F^2) \cdot \mathrm{tr}[XH]$$
$$D_x^2\psi[h,h] = 2f''(\|X\|_F^2) \cdot (\mathrm{tr}[XH])^2 + 2f'(\|x\|_F^2) \cdot \mathrm{tr}[H^2]$$

where x is the vectorization of matrix X and h is the vectorization of matrix H. We can upper bound

$$|D_x\psi[h]| \leq 2|f'(\|X\|_F^2)| \cdot |\,\mathrm{tr}[XH]| \leq 2|f'(\|X\|_F^2)| \cdot \|X\|_F \cdot \|H\|_F$$

Then, we have

$$|f'(\|X\|_F^2)| \cdot \|X\|_F = \begin{cases} \|X\|_F^3/\epsilon_{mp}^3 \leq 1, & \|X\|_F \in [0, \epsilon_{mp}] \\ (4\epsilon_{mp}^2 - \|X\|_F^2)\|X\|_F/9\epsilon_{mp} \leq 2/3, & \|X\|_F \in (\epsilon_{mp}, 2\epsilon_{mp}] \\ 0, & \|X\|_F \in (2\epsilon_{mp}, +\infty) \end{cases}$$

It implies that $|D_x\psi[h]| \leq 2\|H\|_F$, $\forall x$.

By case analysis, we have

$$|f''(\|X\|_F^2)| \cdot \|X\|_F^2 \leq \begin{cases} \frac{1}{\epsilon_{mp}^3}\|X\|_F^2 \leq 4/\epsilon_{mp}, & \|X\|_F^2 \in [0, 4\epsilon_{mp}^2] \\ 0, & \|X\|_F^2 \in (4\epsilon_{mp}, +\infty) \end{cases}$$

We can also upper bound

$$\begin{aligned} |D_x^2\psi[h,h]| &\leq 2|f''(\|X\|_F^2)| \cdot (\mathrm{tr}[XH])^2 + 2|f'(\|X\|_F^2)| \cdot \mathrm{tr}[H^2] \\ &\leq 2|f''(\|X\|_F^2)| \cdot (\|X\|_F\|H\|_F)^2 + 2|f'(\|X\|_F^2)| \cdot \|H\|_F^2 \\ &\leq 2 \cdot \frac{4}{\epsilon_{mp}}\|H\|_F^2 + 2 \cdot \frac{1}{\epsilon_{mp}}\|H\|_F^2 \\ &= \frac{10}{\epsilon_{mp}}\|H\|_F^2. \end{aligned}$$

□

3.6.6 x and $\overline{x}$ are close

Lemma 3.6.15 (x and $\overline{x}$ are close in term of $\widetilde{V}^{-1}$)**.** *With probability* $1-\delta$ *over the randomness of sketching matrix* $R \in \mathbb{R}^{b\times n}$*, we have*

$$\|\overline{x}_i - x_i\|_{\widetilde{V}_i^{-1}} \leq \epsilon_x$$

$\epsilon_x = O(\alpha \log^2(n/\delta) \cdot \frac{n^{1/4}}{\sqrt{b}})$*,* b *is the size of sketching matrix.*

Proof. Recall the definition of $\widetilde{\delta}_x$ and δ_x, we have

$$\widetilde{\delta}_{x,i} - \delta_{x,i} = \widetilde{V}_i^{1/2}(I - R^\top R\widetilde{P})\widetilde{V}^{1/2}h - \widetilde{V}_i^{1/2}(I-\widetilde{P})\widetilde{V}^{1/2}h = \widetilde{V}_i^{1/2}(\widetilde{P} - R^\top R\widetilde{P})\widetilde{V}^{1/2}h$$

For iteration t, the definition should be

$$\widetilde{\delta}_{x,i}^{(t)} - \delta_{x,i}^{(t)} = (\widetilde{V}_i^{(t)})^{1/2}(\widetilde{P}^{(t)} - (R^{(t)})^\top R^{(t)}\widetilde{P}^{(t)})(\widetilde{V}^{(t)})^{1/2}h.$$

For any i, let k be the current iteration, k_i be the last when we changed the $\widetilde{V}_i$. Then, we have that

$$x_i^{(k)} - \overline{x}_i^{(k)} = \sum_{t=k_i}^{k} \widetilde{\delta}_{x,i}^{(t)} - \delta_{x,i}^{(t)}$$

because we have $x_i^{(k_i)} = \overline{x}_i^{(k_i)}$ (guaranteed by our algorithm). Since $\widetilde{V}_i^{(t)}$ did not change during iteration k_i to k for the block i. (However, the whole other parts of matrix $\widetilde{V}$ could change). We consider

$$\begin{aligned}
(x_i^{(k)} - \overline{x}_i^{(k)})^\top \cdot (\widetilde{V}_i^{(k)})^{-1} \cdot (x_i^{(k)} - \overline{x}_i^{(k)}) = & \left(\sum_{t=k_i}^{k} \widetilde{\delta}_{x,i}^{(t)} - \delta_{x,i}^{(t)}\right)^\top \cdot (\widetilde{V}_i^{(k)})^{-1} \cdot \left(\sum_{t=k_i}^{k} \widetilde{\delta}_{x,i}^{(t)} - \delta_{x,i}^{(t)}\right) \\
= & \left\| \sum_{t=k_i}^{k} \left((I - (R^{(t)})^\top R^{(t)})\widetilde{P}^{(t)} (\widetilde{V}^{(t)})^{1/2} h^{(t)} \right)_i \right\|_2^2 .
\end{aligned}$$

We consider block i and a coordinate $j \in$ block i. We define random vector $X_t \in \mathbb{R}^{n_i}$ as follows:

$$X_t = \left((I - R^{(t)\top} R^{(t)}) \widetilde{P}^{(t)} (\widetilde{V}^{(t)})^{1/2} h^{(t)}\right)_i.$$

Let $(X_t)_j$ denote the j-th coordinate of X_t, for each $j \in [n_i]$.

By Lemma 3.9.1 in Section 3.9, we have for each t,

$$\mathbb{E}[X_t] = 0, \quad \text{and} \quad \mathbb{E}[(X_t)_j^2] = \frac{1}{b} \|(\widetilde{P}^{(t)} (\widetilde{V}^{(t)})^{1/2} h^{(t)})_i\|_2^2$$

and with probability $1 - \delta$,

$$|(X_t)_j| \leq \|(\widetilde{P}^{(t)} (\widetilde{V}^{(t)})^{1/2} h^{(t)})_i\|_2 \frac{\log(n/\delta)}{\sqrt{b}} := M.$$

Now, we apply Bernstein inequality (Lemma A.1.2),

$$\Pr\left[\sum_t (X_t)_j > \tau\right] \leq \exp\left(-\frac{\tau^2/2}{\sum_t \mathbb{E}[(X_t)_j^2] + M\tau/3}\right)$$

Choosing $\tau = 10^3 \frac{\sqrt{T}}{\sqrt{b}} \log^2(n/\delta) \cdot \|(\widetilde{P}^{(t)} (\widetilde{V}^{(t)})^{1/2} h^{(t)})_i\|_2$

$$\begin{aligned}
&\Pr\left[\sum_t (X_t)_j > 10^3 \frac{\sqrt{T}}{\sqrt{b}} \log^2(n/\delta) \cdot \|(\widetilde{P}^{(t)} (\widetilde{V}^{(t)})^{1/2} h^{(t)})_i\|_2\right] \\
\leq\ &\exp\left(-\frac{10^6 \frac{T}{b} \log^4(n/\delta) \cdot \|(\widetilde{P}^{(t)} (\widetilde{V}^{(t)})^{1/2} h^{(t)})_i\|_2^2/2}{\frac{T}{b} \|(\widetilde{P}^{(t)} (\widetilde{V}^{(t)})^{1/2} h^{(t)})_i\|_2^2 + 10^3 \frac{\sqrt{T}}{b} \log^3(n/\delta) \|(\widetilde{P}^{(t)} (\widetilde{V}^{(t)})^{1/2} h^{(t)})_i\|_2^2/3}\right) \\
\leq\ &\exp(-100 \log(n/\delta))
\end{aligned}$$

Now, taking a union, we have

$$\begin{aligned}
\left\|\sum_{t=k_i}^{k} \left((I - (R^{(t)})^\top R^{(t)}) \widetilde{P}^{(t)} (\widetilde{V}^{(t)})^{1/2} h^{(t)}\right)_i\right\|_2 &= O\left(\frac{\sqrt{T}}{\sqrt{b}} \log^2(n/\delta) \left\|(\widetilde{P}^{(t)} (\widetilde{V}^{(t)})^{1/2} h^{(t)})_i\right\|_2\right) \\
&\leq O\left(\frac{\sqrt{T}}{\sqrt{b}} \log^2(n/\delta)\alpha\right)
\end{aligned}$$

where we use that $\|(\widetilde{P}^{(t)}(\widetilde{V}^{(t)})^{1/2}h^{(t)})_i\|_2 \leq \|((\widetilde{V}^{(t)})^{1/2}h^{(t)})_i\|_2 = O(\alpha)$, $n_i = O(1)$.

Finally, we use the fact that the algorithm reset $\overline{x} = x$, $\overline{s} = s$ in less than $\sqrt{n}$ iterations.

□

3.6.7 s and $\overline{s}$ are close

Lemma 3.6.16 (s and $\overline{s}$ are close)**.** *With probability $1-\delta$ over the randomness of sketching matrix $R \in \mathbb{R}^{b\times n}$, we have*

$$t^{-1}\|\overline{s}_i - s_i\|_{\widetilde{V}_i} \leq \epsilon_s,$$

$\epsilon_s = O(\alpha \log^2(n/\delta) \cdot \frac{n^{1/4}}{\sqrt{b}}$, and b is the size of sketching matrix.

Proof. Recall the definition of $\widetilde{\delta}_s$, δ_s, we have

$$\widetilde{\delta}_{s,i} - \delta_{s,i} = t\widetilde{V}_i^{-1/2}(R^\top R - I)\widetilde{P}\widetilde{V}^{1/2}h$$

The rest of the proof is identical to Lemma 3.6.15 except we use also the fact we make $\overline{s} = s$ whenever our t changed by a constant factor. We omitted the details here. □

3.6.8 Data structure is maintaining (x, s) implicitly over all the iterations

Lemma 3.6.17. *Over all the iterations, u_1+Fu_2 is always maintaining x implicitly, u_3+Gu_4 is always maintaining s implicitly.*

Proof. We only focus on the PartialUpdate. The FullUpdate is trivial, we ignore the proof.

For x.

Note that M is not changing. Let's assume that $u_1 + Fu_2 = x$, we want to show that

$$u_1^{\text{new}} + F^{\text{new}} u_2^{\text{new}} = x^{\text{new}}.$$

which is equivalent to prove

$$u_1^{\text{new}} + F^{\text{new}} u_2^{\text{new}} - (u_1 + Fu_2) = \delta_x$$

Let $\Delta u_1 = u_1^{\text{new}} - u_1$ be the change of u_1 over iteration t, then

$$\Delta u_1 = \widetilde{V}^{\text{new}} h + (F - F^{\text{new}}) u_2$$

Let $\Delta u_2 = u_2^{\text{new}} - u_2$ be the change of u_2 over iteration t, then

$$\Delta u_2 = -(\widetilde{V}^{\text{new}})^{1/2} h + \mathbf{1}_{\widetilde{S}} (\widetilde{\Delta}_{\widetilde{S},\widetilde{S}}^{-1} + M_{\widetilde{S},\widetilde{S}})^{-1} M_{\widetilde{S}}^\top (\widetilde{V}^{\text{new}})^{1/2} h.$$

By definition of δ_x at iteration t, we have

$$\delta_x = \widetilde{V}^{\text{new}} h - \left(\sqrt{\widetilde{V}^{\text{new}}} M \sqrt{\widetilde{V}^{\text{new}}} h - \sqrt{\widetilde{V}^{\text{new}}} M_{\widetilde{S}} (\widetilde{\Delta}_{\widetilde{S},\widetilde{S}}^{-1} + M_{\widetilde{S},\widetilde{S}})^{-1} (M_{\widetilde{S}})^\top \sqrt{\widetilde{V}^{\text{new}}} h \right).$$

We can compute

$$\begin{aligned}
&u_1^{\text{new}} + F^{\text{new}} u_2^{\text{new}} - (u_1 + F u_2) \\
&= \Delta u_1 + (F^{\text{new}} u_2^{\text{new}} - F u_2) \\
&= \widetilde{V}^{\text{new}} h + (F - F^{\text{new}}) u_2 + (F^{\text{new}} u_2^{\text{new}} - F u_2) \\
&= \widetilde{V}^{\text{new}} h + F^{\text{new}} (u_2^{\text{new}} - u_2) \\
&= \widetilde{V}^{\text{new}} h + F^{\text{new}} \Delta u_2 \\
&= \widetilde{V}^{\text{new}} h - F^{\text{new}} \sqrt{\widetilde{V}^{\text{new}}} h + F^{\text{new}} \mathbf{1}_{\widetilde{S}} (\widetilde{\Delta}_{\widetilde{S},\widetilde{S}}^{-1} + M_{\widetilde{S},\widetilde{S}})^{-1} (M_{\widetilde{S}})^{\top} \sqrt{\widetilde{V}^{\text{new}}} h \\
&= \delta_x
\end{aligned}$$

where we used $F^{\text{new}} = \sqrt{\widetilde{V}^{\text{new}}} M$ in the last step.

For s.

We have

$$G^{\text{new}} = \frac{1}{\sqrt{\widetilde{V}^{\text{new}}}} M, G = \frac{1}{\sqrt{\widetilde{V}}} M$$

Let $\Delta u_3 = u_3^{\text{new}} - u_3$ be the change of u_3 over iteration t, then

$$\Delta u_3 = (G - G^{\text{new}}) u_4$$

Let $\Delta u_4 = u_4^{\text{new}} - u_4$ be the change of u_4 over iteration t, then

$$\Delta u_4 = t \cdot \Delta u_2$$

By definition of δ_s in iteration t,

$$\delta_s = \left(\frac{1}{\sqrt{\widetilde{V}^{\text{new}}}} M \sqrt{\widetilde{V}^{\text{new}}}(th) - \frac{1}{\sqrt{\widetilde{V}^{\text{new}}}} M_{\widetilde{S}} (\widetilde{\Delta}^{-1}_{\widetilde{S},\widetilde{S}} + M_{\widetilde{S},\widetilde{S}})^{-1} (M_{\widetilde{S}})^\top \sqrt{\widetilde{V}^{\text{new}}}(th) \right)$$

We can compute

$$\begin{aligned}
(u_3^{\text{new}} + G^{\text{new}} u_4^{\text{new}}) - (u_3 + G u_4) &= \Delta u_3 + (G^{\text{new}} u_4^{\text{new}} - G u_4) \\
&= (G - G^{\text{new}}) u_4 + (G^{\text{new}} u_4^{\text{new}} - G u_4) \\
&= G^{\text{new}} (u_4^{\text{new}} - u_4) \\
&= G^{\text{new}} t \Delta u_2 \\
&= \delta_s
\end{aligned}$$

where the last step follows by definition of Δu_2.

□

3.7 Combining Robust Central Path with Data Structure

The goal of this section is to combine Section 3.5 and Section 3.6.

Ntn	Choice of Parameter	Statement	Comment
C_1	$\Theta(1/\log^2 n)$	Lem. 3.7.1, Thm. 3.6.1	ℓ_2 accuracy of W sequence
C_2	$\Theta(1/\log^4 n)$	Lem. 3.7.1, Thm. 3.6.1	ℓ_4 accuracy of W sequence
ϵ_{mp}	$\Theta(1/\log^2 n)$	RobustIPM Alg in Sec. 3.5	accuracy for data structure
T	$\Theta(\sqrt{n}\log^2 n \log(n/\delta))$	Thm. 3.4.1	#iterations
α	$\Theta(1/\log^2 n)$	RobustIPM Alg in Sec. 3.5	step size in Hessian norm
b	$\Theta(\sqrt{n}\log^6(nT)$	Lem. 3.6.15, 3.6.16, 3.7.2	sketch size
ϵ_x	$\Theta(1/\log^3 n)$	Lem. 3.6.15	accuracy of $\overline{x}$ (respect to x)
ϵ_s	$\Theta(1/\log^3 n)$	Lem. 3.6.16	accuracy of $\overline{s}$ (respect to s)
ϵ_w	$\Theta(1/\log^3 n)$	Lem. 3.7.2	accuracy of $\overline{W}$ (respect to W)
a	$\min(2/3, \alpha_m)$	α_m is the dual exponent of MM	batch size

Table 3.3: Summary of parameters. "Ntn" denotes Notation.

3.7.1 Guarantee for W matrices

Lemma 3.7.1 (Guarantee of a sequence of W)**.** *Let* $x^{\mathrm{new}} = x+\delta_x$. *Let* $W^{\mathrm{new}} = (\nabla^2\phi(x^{\mathrm{new}}))^{-1}$ *and* $W = (\nabla^2\phi(x))^{-1}$. *Then we have*

$$\sum_{i=1}^m \left\| w_i^{-1/2}(w_i^{\mathrm{new}} - w_i)w_i^{-1/2} \right\|_F^2 \leq C_1^2,$$
$$\sum_{i=1}^m \left\| w_i^{-1/2}(w_i^{\mathrm{new}} - w_i)w_i^{-1/2} \right\|_F^4 \leq C_2^2,$$
$$\left\| w_i^{-1/2}(w_i^{\mathrm{new}} - w_i)w_i^{-1/2} \right\|_F \leq \frac{1}{4}.$$

where $C_2 = \Theta(\alpha^2)$ *and* $C_1 = \Theta(\alpha)$.

Proof. For each $i \in [m]$, we have

$$\begin{aligned}
&\left\| W_i^{-1/2}(W_i^{\mathrm{new}} - W_i)W_i^{-1/2} \right\|_F^2 \\
&= n_i \left\| W_i^{-1/2}(W_i^{\mathrm{new}} - W_i)W_i^{-1/2} \right\|^2 \\
&= n_i \left\| (\nabla^2\phi(x_i))^{1/2}(\nabla^2\phi(x_i^{\mathrm{new}})^{-1} - \nabla^2\phi(x_i)^{-1})(\nabla^2\phi(x_i))^{1/2} \right\|^2 \\
&\leq \left(\frac{1}{(1 - \|x_i^{\mathrm{new}} - x_i\|_{\nabla^2\phi(x_i)})^2} - 1 \right)^2 \cdot \left\| (\nabla^2\phi(x_i))^{1/2}\nabla^2\phi(x_i)^{-1}(\nabla^2\phi(x_i))^{1/2} \right\|^2 \\
&= n_i \left(\frac{1}{(1 - \|x_i^{\mathrm{new}} - x_i\|_{\nabla^2\phi(x_i)})^2} - 1 \right)^2 \\
&\leq 100 n_i \|x_i^{\mathrm{new}} - x_i\|_{\nabla^2\phi(x_i)}^2,
\end{aligned}$$

where the second step follows by Theorem 3.2.1.

In our problem, we assume that $n_i = O(1)$. It remains to bound

$$\|x_i^{\mathrm{new}} - x_i\|_{\nabla^2\phi(x_i)}^2 = \|\delta_{x,i}\|_{\nabla^2\phi(x_i)}^2 \lesssim \|\delta_{x,i}\|_{\bar{x}_i}^2 = \alpha_i^2$$

where the last step follows from definition $\alpha_i = \|\delta_{x,i}\|_{\overline{x}_i}$.

Then, we have

$$\sum_{i=1}^{m} \|x_i^{\mathrm{new}} - x_i\|^2_{\nabla^2\phi(x_i)} \leq \sum_{i=1}^{m} O(\alpha_i^2) \leq O(\alpha^2).$$

where the last step follows by Lemma 3.5.1. □

Lemma 3.7.2 (Accuracy of $\overline{W}$)**.** *Let x and $\overline{x}$ be the vectors maintained by data-structure* StochasticProjectionMaintenance. *Let $W = (\nabla^2\phi(x))^{-1}$ and $\overline{W} = (\nabla^2\phi(\overline{x}))^{-1}$. Then we have*

$$\|w_i^{-1/2}(\overline{w}_i - w_i)w_i^{-1/2}\|_F \leq \epsilon_w,$$

where $\epsilon_w = O\Big(\alpha \log^2(nT) \cdot \frac{n^{1/4}}{\sqrt{b}}\Big)$, b is the size of sketching matrix.

Proof. By similar calculation, we have

$$\|w_i^{-1/2}(\overline{w}_i - w_i)w_i^{-1/2}\|_F = O(1) \cdot \|\overline{x}_i - x_i\|_{\nabla^2\phi(x_i)}.$$

Then, using Lemma 3.6.15 with $\delta = 1/T$

$$\|\overline{x}_i - x_i\|_{\nabla^2\phi(x_i)} \leq O\left(\alpha \log^2(nT) \cdot \frac{\sqrt{n^{1/4}}}{\sqrt{b}}\right).$$

□

Algorithm 3.5 Robust Central Path

procedure CentralPathStep($\overline{x}, \overline{s}, t, \lambda, \alpha$)
  for $i = 1 \to m$ **do** ▷ Figure out direction h
    $\mu_i^t \leftarrow \overline{s}_i / t + \nabla \phi_i(\overline{x}_i)$ ▷ According to Eq. (3.6)
    $\gamma_i^t \leftarrow \|\mu_i^t\|_{\nabla^2 \phi_i(\overline{x}_i)^{-1}}$ ▷ According to Eq. (3.7)
    $c_i^t \leftarrow \frac{\exp(\lambda \gamma_i^t)/\gamma_i^t}{(\sum_{i=1}^m \exp(2\lambda \gamma_i^t))^{1/2}}$ if $\gamma_i^t \geq 96\sqrt{\alpha}$ and $c_i^t \leftarrow 0$ otherwise ▷ According to Eq. (3.9)
    $h_i \leftarrow -\alpha \cdot c_i^t \cdot \mu_i^t$ ▷ According to Eq. (3.8)
  end for
  $\overline{W} \leftarrow (\nabla^2 \phi(\overline{x}))^{-1}$ ▷ Computing block-diagonal matrix $\overline{W}$
  return $h, \overline{W}$
end procedure

procedure RobustCentralPath(mp, t, λ, α) ▷ Lemma 3.5.8
  ▷ Standing at (x, s) implicitly via data-structure
  ▷ Standing at $(\overline{x}, \overline{s})$ explicitly via data-structure
  $(\overline{x}, \overline{s}) \leftarrow$ mp.Query() ▷ Algorithm 3.1, Lemma 3.6.6

  $h, \overline{W} \leftarrow$ CentralPathStep($\overline{x}, \overline{s}, t, \lambda, \alpha$)

  mp.Update($\overline{W}$) ▷ Algorithm 3.2, Lemma 3.6.3
  mp.MultiplyMove(h, t) ▷ Algorithm 3.4, Lemma 3.6.8, Lemma 3.6.7
    ▷ $x \leftarrow x + \delta_x$, $s \leftarrow s + \delta_s$, achieved by data-structure implicitly
    ▷ $\overline{x} \leftarrow \overline{x} + \widetilde{\delta}_x$, $\overline{s} \leftarrow \overline{s} + \widetilde{\delta}_s$, achieved by data-structure explicitly
    ▷ If x is far from $\overline{x}$, then $\overline{x} \leftarrow x$
end procedure

3.7.2 Main result

The goal of this section is to prove our main result.

Theorem 3.7.3 (Main result, formal version of Theorem 3.1.1)**.** *Consider a convex problem*

$$\min_{Ax=b, x \in \prod_{i=1}^m K_i} c^\top x$$

where K_i are compact convex set. For each $i \in [m]$, we are given a ν_i-self concordant barrier

Algorithm 3.6 Our main algorithm (More detailed version of RobustIPM in Section 3.4)

procedure Main(A, b, c, ϕ, δ) ▷ Theorem 3.1.1, Theorem 3.7.3
  $\lambda \leftarrow 2^{16}\log(m)$, $\alpha \leftarrow 2^{-20}\lambda^{-2}$, $\kappa \leftarrow 2^{-10}\alpha$
  $\delta \leftarrow \min(\frac{1}{\lambda}, \delta)$ ▷ Choose the target accuracy
  $a \leftarrow \min(2/3, \alpha_m)$ ▷ Choose the batch size
  $b_{\text{sketch}} \leftarrow 2^{10}\sqrt{\nu}\log^6(n/\delta) \cdot \log\log(1/\delta)$ ▷ Choose the size of sketching matrix
  Modify the ERM(A, b, c, ϕ) and obtain an initial x and s
  CentralPathMaintenance mp ▷ Algorithm 3.1, Theorem 3.6.1
  mp.Initialize$(A, x, s, \alpha, a, b_{\text{sketch}})$ ▷ Algorithm 3.1, Lemma 3.6.2
  $\nu \leftarrow \sum_{i=1}^m \nu_i$ ▷ ν_i are the self-concordant parameters of ϕ_i
  $t \leftarrow 1$
  while $t > \delta^2/(4\nu)$ **do**
    $t^{\text{new}} \leftarrow (1 - \frac{\kappa}{\sqrt{\nu}})t$
    RobustCentralPath$(\text{mp}, t, \lambda, \alpha)$ ▷ Algorithm 3.5
    $t \leftarrow t^{\text{new}}$
  end while
  Return an approximate solution of the original ERM according to Section 3.8
end procedure

function ϕ_i for K_i. Also, we are given $x^{(0)} = \arg\min_x \sum_i \phi_i(x_i)$. Assume that

1. *Diameter of the set: For any $x \in \prod_{i=1}^m K_i$, we have that $\|x\|_2 \leq R$.*

2. *Lipschitz constant of the program: $\|c\|_2 \leq L$.*

Then, the algorithm Main *finds a vector x such that*

$$c^\top x \leq \min_{Ax=b, x \in \prod_{i=1}^m K_i} c^\top x + LR \cdot \delta,$$
$$\|Ax - b\|_1 \leq 3\delta \cdot \left(R \sum_{i,j} |A_{i,j}| + \|b\|_1 \right),$$
$$x \in \prod_{i=1}^m K_i.$$

in time

$$O(n^{\omega+o(1)} + n^{2.5-\alpha/2+o(1)} + n^{2+1/6+o(1)}) \cdot \widetilde{O}(\log(n/\delta)).$$

where ω is the exponent of matrix multiplication [Wil12, LG14], and α is the dual exponent of matrix multiplication [LGU18].

Proof. The number of iterations is

$$O(\sqrt{\nu}\log^2(m)\log(\nu/\delta)) = O(\sqrt{n}\log^2(n)\log(n/\delta)).$$

For each iteration, the amortized cost per iteration is

$$\begin{aligned}
&O(nb + n^{1+a} + n^{1.5}) + O(C_1/\epsilon_{mp} + C_2/\epsilon_{mp}^2) \cdot (n^{\omega-1/2+o(1)} + n^{2-a/2+o(1)}) + O(n^{\omega-1/2+o(1)}) \\
&= O(nb + n^{1+a} + n^{1.5}) + O(\alpha + \alpha^2) \cdot (n^{\omega-1/2+o(1)} + n^{2-a/2+o(1)}) + O(n^{\omega-1/2+o(1)}) \\
&= O(nb + n^{1+a} + n^{1.5}) + O(1/\log^4 n) \cdot (n^{\omega-1/2+o(1)} + n^{2-a/2+o(1)}) + O(n^{\omega-1/2+o(1)}) \\
&= O(n^{1.5+o(1)}\log^6\log(1/\delta) + n^{1+a+o(1)}) + O(n^{\omega-1/2+o(1)} + n^{2-a/2+o(1)}).
\end{aligned}$$

where the last step follows from choice of b (see Table 3.3).

Finally, we have

$$\begin{aligned}
&\text{total time} \\
&= \#\text{iterations} \cdot \text{cost per iteration} \\
&= \underbrace{O\left(\sqrt{n}\log^2 n \log(n/\delta)\right)}_{\#\text{iterations}} \cdot \underbrace{O\left(n^{1.5+o(1)}\log^6\log(1/\delta) + n^{1+a+o(1)} + n^{\omega-1/2+o(1)} + n^{2-a/2+o(1)}\right)}_{\text{cost per iteration}} \\
&= O\left(n^{1.5+a+o(1)} + n^{\omega+o(1)} + n^{2.5-a/2+o(1)}\right) \cdot \log(n/\delta) \cdot \log^6\log(1/\delta) \\
&= O\left(n^{2+1/6+o(1)} + n^{\omega+o(1)} + n^{2.5-\alpha_m/2+o(1)}\right) \cdot \log(n/\delta) \cdot \log^6\log(1/\delta)
\end{aligned}$$

where we pick $a = \min(2/3, \alpha_m)$ and α_m is the dual exponent of matrix multiplication[LGU18].

Thus, we complete the proof. □

Corollary 3.7.4 (Empirical risk minimization)**.** *Given convex function $f_i(y) : \mathbb{R} \to \mathbb{R}$. Suppose the solution $x^* \in \mathbb{R}^d$ lies in ℓ_∞-Ball$(0, R)$. Suppose f_i is L-Lipschitz in region $\{y : |y| \leq 4\sqrt{n} \cdot M \cdot R\}$. Given a matrix $A \in \mathbb{R}^{d \times n}$ with $\|A\| \leq M$ and A has no redundant constraints, and a vector $b \in \mathbb{R}^d$ with $\|b\|_2 \leq M \cdot R$. We can find $x \in \mathbb{R}^d$ s.t.*

$$\sum_{i=1}^n f_i(a_i^\top x + b_i) \leq \min_{x \in \mathbb{R}^d} \sum_{i=1}^n f_i(a_i^\top x + b_i) + \delta MR$$

in time

$$O(n^{\omega + o(1)} + n^{2.5 - \alpha/2 + o(1)} + n^{2 + 1/6 + o(1)}) \cdot \widetilde{O}(\log(n/\delta)).$$

where ω is the exponent of matrix multiplication [Wil12, LG14], and α is the dual exponent of matrix multiplication [LGU18].

Proof. It follows from applying Theorem 3.7.3 on convex program (3.1) with an extra constraint x^* lies in ℓ_∞-Ball$(0, R)$. Note that in program (3.1), $n_i = 2$. Thus $m = O(n)$. □

3.8 Initial Point and Termination Condition

We first need some result about self concordance.

Lemma 3.8.1 (Theorem 4.1.7, Lemma 4.2.4 in [Nes98]). *Let ϕ be any ν-self-concordant barrier. Then, for any $x, y \in \mathrm{dom}\phi$, we have*

$$\langle \nabla\phi(x), y - x\rangle \leq \nu,$$
$$\langle \nabla\phi(y) - \nabla\phi(x), y - x\rangle \geq \frac{\|y-x\|_x^2}{1+\|y-x\|_x}.$$

Let $x^* = \arg\min_x \phi(x)$. For any $x \in \mathbb{R}^n$ such that $\|x - x^*\|_{x^*} \leq 1$, we have that $x \in \mathrm{dom}\phi$.

$$\|x^* - y\|_{x^*} \leq \nu + 2\sqrt{\nu}.$$

Lemma 3.8.2. *Consider a convex problem $\min_{Ax=b, x\in\prod_{i=1}^m K_i} c^\top x$ where K_i are compact convex set. For each $i \in [m]$, we are given a ν_i-self concordant barrier function ϕ_i for K_i. Also, we are given $x^{(0)} = \arg\min_x \sum_i \phi_i(x_i)$. Assume that*

1. *Diameter of the set: For any $x \in \prod_{i=1}^m K_i$, we have that $\|x\|_2 \leq R$.*
2. *Lipschitz constant of the program: $\|c\|_2 \leq L$.*

For any $\delta > 0$, the modified program $\min_{\overline{A}\overline{x}=\overline{b}, \overline{x}\in\prod_{i=1}^m K_i \times \mathbb{R}_+} \overline{c}^\top \overline{x}$ with

$$\overline{A} = [A \mid b - Ax^{(0)}], \overline{b} = b, \text{ and } \overline{c} = \begin{bmatrix} \frac{\delta}{LR}\cdot c \\ 1 \end{bmatrix}$$

satisfies the following:

1. *$\overline{x} = \begin{bmatrix} x^{(0)} \\ 1 \end{bmatrix}$, $\overline{y} = 0_d$ and $\overline{s} = \begin{bmatrix} \frac{\delta}{LR}\cdot c \\ 1 \end{bmatrix}$ are feasible primal dual vectors with $\|\overline{s} + \nabla\overline{\phi}(\overline{x})\|_{\overline{x}}^* \leq \delta$ where $\overline{\phi}(\overline{x}) = \sum_{i=1}^m \phi_i(\overline{x}_i) - \log(\overline{x}_{m+1})$.*

2. *For any $\overline{x}$ such that $\overline{A}\overline{x} = \overline{b}, \overline{x} \in \prod_{i=1}^m K_i \times \mathbb{R}_+$ and $\overline{c}^\top \overline{x} \leq \min_{\overline{A}\overline{x}=\overline{b}, \overline{x} \in \prod_{i=1}^m K_i \times \mathbb{R}_+} \overline{c}^\top \overline{x} + \delta^2$, the vector $\overline{x}_{1:n}$ ($\overline{x}_{1:n}$ is the first n coordinates of $\overline{x}$) is an approximate solution to the original convex program in the following sense*

$$\begin{aligned} c^\top \overline{x}_{1:n} &\leq \min_{Ax=b, x \in \prod_{i=1}^m K_i} c^\top x + LR \cdot \delta, \\ \|A\overline{x}_{1:n} - b\|_1 &\leq 3\delta \cdot \left(R \sum_{i,j} |A_{i,j}| + \|b\|_1 \right), \\ \overline{x}_{1:n} &\in \prod_{i=1}^m K_i. \end{aligned}$$

Proof. For the first result, straightforward calculations show that $(\overline{x}, \overline{y}, \overline{s})$ are feasible.

To compute $\|\overline{s} + \nabla\overline{\phi}(\overline{x})\|^*_{\overline{x}}$, note that

$$\|\overline{s} + \nabla\overline{\phi}(\overline{x})\|^*_{\overline{x}} = \|\frac{\delta}{LR} \cdot c\|_{\nabla^2\phi(x^{(0)})^{-1}}.$$

Lemma 3.8.1 shows that $x \in \mathbb{R}^n$ such that $\|x - x^{(0)}\|_{x^{(0)}} \leq 1$, we have that $x \in \prod_{i=1}^m K_i$ because $x^{(0)} = \arg\min_x \sum_i \phi_i(x_i)$. Hence, for any v such that $v^\top \nabla^2\phi(x^{(0)})v \leq 1$, we have that $x^{(0)} \pm v \in \prod_{i=1}^m K_i$ and hence $\|x^{(0)} \pm v\|_2 \leq R$. This implies $\|v\|_2 \leq R$ for any $v^\top \nabla^2\phi(x^{(0)})v \leq 1$. Hence, $(\nabla^2\phi(x^{(0)}))^{-1} \preceq R^2 \cdot I$. Hence, we have

$$\|\overline{s} + \nabla\overline{\phi}(\overline{x})\|^*_{\overline{x}} = \|\frac{\delta}{LR} \cdot c\|_{\nabla^2\phi(x^{(0)})^{-1}} \leq \|\frac{\delta}{L} \cdot c\|_2 \leq \delta.$$

For the second result, we let

$$\mathrm{OPT} = \min_{Ax=b, x \in \prod_{i=1}^m K_i} c^\top x \quad \text{and} \quad \overline{\mathrm{OPT}} = \min_{\overline{A}\overline{x}=b, \overline{x} \in \prod_{i=1}^m K_i \times \mathbb{R}_+} \overline{c}^\top \overline{x}.$$

For any feasible x in the original problem, $\overline{x} = \begin{bmatrix} x \\ 0 \end{bmatrix}$ is a feasible in the modified problem. Therefore, we have that

$$\overline{\mathrm{OPT}} \leq \frac{\delta}{LR} \cdot c^\top x = \frac{\delta}{LR} \cdot \mathrm{OPT}.$$

Given a feasible $\overline{x}$ with additive error δ^2. Write $\overline{x} = \begin{bmatrix} \overline{x}_{1:n} \\ \tau \end{bmatrix}$ for some $\tau \geq 0$. We can compute $\bar{c}^\top \overline{x}$ which is $\frac{\delta}{LR} \cdot c^\top \overline{x}_{1:n} + \tau$. Then, we have

$$\frac{\delta}{LR} \cdot c^\top \overline{x}_{1:n} + \tau \leq \overline{\mathrm{OPT}} + \delta^2 \leq \frac{\delta}{LR} \cdot \mathrm{OPT} + \delta^2. \tag{3.34}$$

Hence, we can upper bound the OPT of the transformed program as follows:

$$c^\top \overline{x}_{1:n} = \frac{LR}{\delta} \cdot \frac{\delta}{LR} c^\top \overline{x}_{1:n} \leq \frac{LR}{\delta} \left(\frac{\delta}{LR} \cdot \mathrm{OPT} + \delta^2 \right) = \mathrm{OPT} + LR \cdot \delta,$$

where the second step follows by (3.34).

For the feasibility, we have that $\tau \leq -\frac{\delta}{LR} \cdot c^\top \overline{x}_{1:n} + \frac{\delta}{LR} \cdot \mathrm{OPT} + \delta^2 \leq \delta + \delta + \delta$ because $\mathrm{OPT} = \min_{Ax=b, x \geq 0} c^\top x \leq LR$ and that $c^\top \overline{x}_{1:n} \leq LR$. The constraint in the new polytope shows that

$$A\overline{x}_{1:n} + (b - Ax^{(0)})\tau = b.$$

Rewriting it, we have $A\overline{x}_{1:n} - b = (Ax^{(0)} - b)\tau$ and hence

$$\|A\overline{x}_{1:n} - b\|_1 \leq \|Ax^{(0)} - b\|_1 \cdot \tau.$$

□

Lemma 3.8.3. *Let $\phi_i(x_i)$ be a ν_i-self-concordant barrier. Suppose we have $\frac{s_i}{t} + \nabla \phi_i(x_i) = \mu_i$ for all $i \in [m]$, $A^\top y + s = c$ and $Ax = b$. Suppose that $\|\mu_i\|^*_{x,i} \leq 1$ for all i, we have that*

$$\langle c, x \rangle \leq \langle c, x^* \rangle + 4t\nu$$

where $x^* = \arg\min_{Ax=b, x \in \prod_{i=1}^m K_i} c^\top x$ and $\nu = \sum_{i=1}^m \nu_i$.

Proof. Let $x_\alpha = (1-\alpha)x + \alpha x^*$ for some α to be chosen. By Lemma 3.8.1, we have that $\langle \nabla\phi(x_\alpha), x^* - x_\alpha\rangle \le \nu$. Hence, we have $\frac{\nu}{1-\alpha} \ge \langle \nabla\phi(x_\alpha), x^* - x\rangle$. Hence, we have

$$\begin{aligned}
\frac{\nu\alpha}{1-\alpha} &\ge \langle \nabla\phi(x_\alpha), x_\alpha - x\rangle \\
&= \langle \nabla\phi(x_\alpha) - \nabla\phi(x), x_\alpha - x\rangle + \left\langle \mu - \frac{s}{t}, x_\alpha - x\right\rangle \\
&\ge \sum_{i=1}^m \frac{\|x_{\alpha,i} - x_i\|_{x_i}^2}{1 + \|x_{\alpha,i} - x_i\|_{x_i}} + \langle \mu, x_\alpha - x\rangle - \frac{1}{t}\left\langle c - A^\top y, x_\alpha - x\right\rangle \\
&\ge \sum_{i=1}^m \frac{\alpha^2\|x_i^* - x_i\|_{x_i}^2}{1 + \alpha\|x_i^* - x_i\|_{x_i}} - \alpha\sum_{i=1}^m \|\mu_i\|_{x_i}^*\|x_i^* - x_i\|_{x_i} - \frac{\alpha}{t}\left\langle c, x^* - x\right\rangle.
\end{aligned}$$

where we used Lemma 3.8.1 on the second first, $Ax_\alpha = Ax$ on the second inequality. Hence, we have

$$\frac{\langle c, x\rangle}{t} \le \frac{\langle c, x^*\rangle}{t} + \frac{\nu}{1-\alpha} + \sum_{i=1}^m \|\mu_i\|_{x_i}^*\|x_i^* - x_i\|_{x_i} - \sum_{i=1}^m \frac{\alpha\|x_i^* - x_i\|_{x_i}^2}{1 + \alpha\|x_i^* - x_i\|_{x_i}}.$$

Using $\|\mu_i\|_{x_i}^* \le 1$ for all i, we have

$$\frac{\langle c, x\rangle}{t} \le \frac{\langle c, x^*\rangle}{t} + \frac{\nu}{1-\alpha} + \sum_{i=1}^m \frac{\|x_i^* - x_i\|_{x_i}}{1 + \alpha\|x_i^* - x_i\|_{x_i}} \le \frac{\langle c, x^*\rangle}{t} + \frac{\nu}{1-\alpha} + \frac{m}{\alpha}.$$

Setting $\alpha = \frac{1}{2}$, we have $\langle c, x\rangle \le \langle c, x^*\rangle + 2t(\nu + m) \le \langle c, x^*\rangle + 4t\nu$ because the self-concordance ν_i is always larger than 1.

□

3.9 Basic Properties of Subsampled Hadamard Transform Matrix

This section provides some standard calculations about sketching matrices, it can be found in previous literatures [PSW17]. Usually, the reason for using subsampled randomized Hadamard/Fourier transform [LDFU13] is multiplying the matrix with k vectors only takes $kn \log n$ time. Unfortunately, in our application, the best way to optimize the running is using matrix multiplication directly (without doing any fast Fourier transform [CT65], or more fancy sparse Fourier transform [HIKP12b, HIKP12a, Pri13, IKP14, IK14, PS15, CKPS16, Kap16, Kap17, NSW19a]). In order to have an easy analysis, we still use subsampled randomized Hadamard/Fourier matrix.

3.9.1 Properties obtained by random projection

Remark 3.9.1. The Subsampled Randomized Hadamard Transform [LDFU13] can be defined as $R = SH_n\Sigma \in \mathbb{R}^{b\times}$, where Σ is an $n \times n$ diagonal matrix with i.i.d. diagonal entries $\Sigma_{i,i}$ in which $\Sigma_{i,i} = 1$ with probability $1/2$, and $\Sigma_{i,i} = -1$ with probability $1/2$. H_n refers to the Hadamard matrix of size n, which we assume is a power of 2. The $b \times n$ matrix S samples b coordinates of n dimensional vector uniformly at random. If we replace the definition of sketching matrix in Lemma 3.9.1 by Subsampled Randomized Hadamard Transform and let $\overline{R} = SH_n$, then the same proof will go through.

Lemma 3.9.1 (Expectation, variance, absolute guarantees for sketching a fixed vector)**.** *Let $h \in \mathbb{R}^n$ be a fixed vector. Let $\overline{R} \in \mathbb{R}^{b\times n}$ denote a random matrix where each entry is i.i.d. sampled from $+1/\sqrt{b}$ with probability $1/2$ and $-1/\sqrt{b}$ with probability $1/2$. Let $\Sigma \in \mathbb{R}^{n\times n}$ denote a diagonal matrix where each entry is 1 with probability $1/2$ and -1 with probability $1/2$. Let $R = \overline{R}\Sigma$, then we have*

$$\mathbb{E}[R^\top Rh] = h, \quad \mathbb{E}[(R^\top Rh)_i^2] \leq h_i^2 + \frac{1}{b}\|h\|_2^2, \quad \Pr\left[|(R^\top Rh)_i - h_i| > \|h\|_2 \frac{\log(n/\delta)}{\sqrt{b}}\right] \leq \delta.$$

Proof. Let $R_{i,j}$ denote the entry at i-th row and j-th column in matrix $R \in \mathbb{R}^{b\times n}$. Let $R_{*,i} \in \mathbb{R}^b$ denote the vector in i-th column of R.

We first show expectation,

$$\begin{aligned}
\mathbb{E}[(R^\top Rh)_i] &= \mathbb{E}[\langle R, R_{*,i}h^\top\rangle] \\
&= \mathbb{E}\left[\sum_{j=1}^{b}\sum_{l=1}^{n} R_{j,l}R_{j,i}h_l\right] \\
&= \mathbb{E}\left[\sum_{j=1}^{b} R_{j,i}^2 h_i\right] + \mathbb{E}\left[\sum_{j=1}^{b}\sum_{l\in[n]\setminus i} R_{j,l}R_{j,i}h_l\right] \\
&= h_i + 0 \\
&= h_i
\end{aligned}$$

Secondly, we prove the variance is small

$$\begin{aligned}
\mathbb{E}[(R^\top Rh_i)^2] &= \mathbb{E}[\langle R, R_{*,i}h^\top\rangle^2] \\
&= \mathbb{E}\left[\left(\sum_{j=1}^{b}\sum_{l=1}^{n} R_{j,l}R_{j,i}h_l\right)^2\right] \\
&= \mathbb{E}\left[\left(\sum_{j=1}^{b} R_{j,i}^2 h_i + \sum_{j=1}^{b}\sum_{l\in[n]\setminus i} R_{j,l}R_{j,i}h_l\right)^2\right] \\
&= \mathbb{E}\left[\left(\sum_{j=1}^{b} R_{j,i}^2 h_i\right)^2\right] + 2\,\mathbb{E}\left[\sum_{j'=1}^{b} R_{j',i}^2 h_i \sum_{j=1}^{b}\sum_{l\in[n]\setminus i} R_{j,l}R_{j,i}h_l\right] \\
&\quad + \mathbb{E}\left[\left(\sum_{j=1}^{b}\sum_{l\in[n]\setminus i} R_{j,l}R_{j,i}h_l\right)^2\right] \\
&= C_1 + C_2 + C_3,
\end{aligned}$$

where the last step follows from defining those terms to be C_1, C_2 and C_3. For the term C_1,

we have

$$C_1 = h_i^2 \, \mathbb{E}\left[\left(\sum_{j=1}^{b} R_{j,i}^2\right)^2\right] = h_i^2 \, \mathbb{E}\left[\sum_{j=1}^{b} R_{j,i}^4 + \sum_{j' \neq j} R_{j,i}^2 R_{j',i}^2\right] = h_i^2 \left(b \cdot \frac{1}{b^2} + b(b-1) \cdot \frac{1}{b^2}\right) = h_i^2$$

For the second term C_2,

$$C_2 = 0.$$

For the third term C_3,

$$\begin{aligned} C_3 = & \ \mathbb{E}\left[\left(\sum_{j=1}^{b} \sum_{l \in [n] \setminus i} R_{j,l} R_{j,i} h_l\right)^2\right] \\ = & \ \mathbb{E}\left[\sum_{j=1}^{b} \sum_{l \in [n] \setminus i} R_{j,l}^2 R_{j,i}^2 h_l^2\right] + \mathbb{E}\left[\sum_{j=1}^{b} \sum_{l \in [n] \setminus i} R_{j,l} R_{j,i} h_l \sum_{j' \in [b] \setminus j} \sum_{l' \in [n] \setminus i \setminus l} R_{j',l'} R_{j',i} h_{l'}\right] \\ = & \sum_{j=1}^{b} \sum_{l \in [n] \setminus i} \frac{1}{b} \frac{1}{b} h_l^2 + 0 \leq \frac{1}{b} \|h\|_2^2 \end{aligned}$$

Therefore, we have

$$\mathbb{E}[(R^\top R h)_i^2] \leq C_1 + C_2 + C_3 \leq h_i^2 + \frac{1}{b} \|h\|_2^2.$$

Third, we prove the worst case bound with high probability. We can write $(R^\top R h)_i -$

h_i as follows

$$\begin{aligned}
(R^\top Rh)_i - h_i &= \langle R, R_{*,i}h^\top \rangle - h_i \\
&= \sum_{j=1}^{b}\sum_{l=1}^{n} R_{j,l}\cdot R_{j,i}\cdot h_l - h_i \\
&= \sum_{j=1}^{b} R_{j,i}^2 h_i - h_i + \sum_{j=1}^{b}\sum_{l\in[n]\backslash i} R_{j,l}R_{j,i}\cdot h_l \\
&= \sum_{j=1}^{b}\sum_{l\in[n]\backslash i} R_{j,l}R_{j,i}\cdot h_l && \text{by } R_{j,i}^2 = 1/b \\
&= \sum_{l\in[n]\backslash i} h_l \langle R_{*,l}, R_{*,i}\rangle \\
&= \sum_{l\in[n]\backslash i} h_l \cdot \langle \sigma_l \overline{R}_{*,l}, \sigma_i \overline{R}_{*,i}\rangle && \text{by } R_{*,l} = \sigma_l \overline{R}_{*,l}
\end{aligned}$$

First, we apply Khintchine's inequality, we have

$$\Pr\left[\left|\sum_{l\in[n]\backslash i} h_l\cdot\sigma_l\cdot\langle\overline{R}_{*,l},\sigma_i\overline{R}_{*,i}\rangle\right| \geq Ct\left(\sum_{l\in[n]\backslash i} h_l^2(\langle\overline{R}_{*,l},\sigma_i\overline{R}_{*,i}\rangle)^2\right)^{1/2}\right] \leq \exp(-C't^2)$$

and choose $t = \sqrt{\log(n/\delta)}$.

For each $l \neq i$, using [LDFU13] we have

$$\Pr\left[|\langle\overline{R}_{*,l},\overline{R}_{*,i}\rangle| \geq \frac{\sqrt{\log(n/\delta)}}{\sqrt{b}}\right] \leq \delta/n.$$

Taking a union bound over all $l \in [n]\backslash i$, we have

$$|(R^\top Rh)_i - h_i| \leq \|h\|_2 \frac{\log(n/\delta)}{\sqrt{b}}$$

with probability $1-\delta$. □

Chapter 4

Algorithmic Theory of ODEs

Sampling logconcave functions arising in statistics and machine learning has been a subject of intensive study. Recent developments include analyses for Langevin dynamics and Hamiltonian Monte Carlo (HMC). While both approaches have dimension-independent bounds for the underlying *continuous* processes under sufficiently strong smoothness conditions, the resulting discrete algorithms have complexity and number of function evaluations growing with the dimension. Motivated by this problem, in this chapter, we give a general algorithm for solving multivariate ordinary differential equations whose solution is close to the span of a known basis of functions (e.g., polynomials or piecewise polynomials). The resulting algorithm has polylogarithmic depth and essentially tight runtime — it is nearly linear in the size of the representation of the solution.

We apply this to the sampling problem to obtain a nearly linear implementation of HMC for a broad class of smooth, strongly logconcave densities, with the number of iterations (parallel depth) and gradient evaluations being *polylogarithmic* in the dimension (rather than polynomial as in previous work). This class includes the widely-used loss function for logistic regression with incoherent weight matrices and has been subject of much study recently. We also give a faster algorithm with *polylogarithmic depth* for the more general and standard class of strongly convex functions with Lipschitz gradient. These results are based on (1)

an improved contraction bound for the exact HMC process and (2) logarithmic bounds on the degree of polynomials that approximate solutions of the differential equations arising in implementing HMC.

4.1 Introduction

The complexity of sampling a high-dimensional density of the form $e^{-f(x)}$ where f is a convex function is a fundamental problem with many applications [LS90, LS92, LS93, LV06a, LV06b, Dal17, DK17, DRD18, DCWY18]. The focus of this chapter is to give very fast, i.e., nearly linear time algorithms, for a large subclass of such densities. A motivating and important case is the loss function for logistic regression, widely used in machine learning applications [Ber44, Paa00, NJ02, HJLS13, Bou14]:

$$\sum_{i=1}^{n} \phi_i(a_i^\top x)$$

where ϕ_i are convex functions; a popular choice is $\phi(t) = \log(1 + e^{-t})$. Sampling according to e^{-f} for this choice of f corresponds to sampling models according to their KL-divergence, a natural and effective choice for classification problems [HJLS13].

A general approach to sampling is by an ergodic Markov chain whose stationary distribution is designed to have the desired density. Traditionally, this is done via a Metropolis filter, which accepts a proposed (random) next step y from the current point x with probability $\min\{1, \frac{f(y)}{f(x)}\}$. While very general, one downside of this approach is the possibility of high rejection probabilities, which typically force local steps to be very small. Nevertheless, for arbitrary logconcave functions (including nonsmooth ones), this approach has the current best guarantees [LV06b].

Another family of algorithms is derived from an underlying *continuous* stochastic process with the desired stationary density. A classic example of such a continuous process is Brownian motion. To sample a convex body for example, one could use Brownian motion

with a boundary reflection condition. This is written as the stochastic equation:

$$\mathrm{d}X_t = \mathrm{d}W_t$$

with reflection at the boundary of the domain, and dW_t being infinitesimal Brownian motion. To sample from the density proportional to $e^{-f(x)}$, one can use the stochastic differential equation,

$$\mathrm{d}X_t = -\nabla f(X_t)\mathrm{d}t + \sqrt{2}\mathrm{d}W_t.$$

By the classical Fokker-Planck equation, under mild assumptions on f, the stationary density of this process is proportional to $e^{-f(x)}$.

How can we turn these continuous processes into algorithms? One approach is to take small rather than infinitesimal steps, and this leads to the Langevin dynamics, of which there are multiple flavors [Dal17, DK17, ZLC17, RRT17, DRD18, CCBJ18, CCAY+18, CFM+18]. Starting with Dalalyan [Dal17], it has been established that these dynamics converge in polynomial (in dimension) time for strongly logconcave functions, with the underdamped version converging in $O(\sqrt{d})$ iterations (and polynomial dependences on appropriate condition numbers) [CCBJ18]. The dependence on dimension seems unavoidable in the discretized algorithm, even though the continuous process has no such dependence.

Hamiltonian Monte Carlo. HMC is a random process that maintains a position x and velocity pair v. To sample according to e^{-f}, we define a Hamiltonian $H(x, v) = f(x) + \frac{1}{2}\|v\|^2$. At each step v is chosen randomly from $N(0, I)$ and x is updated using the following Ordinary Differential Equation (ODE) for a some fixed time interval.

$$\frac{\mathrm{d}x(t)}{\mathrm{d}t} = v(t), \quad \frac{\mathrm{d}v(t)}{\mathrm{d}t} = -\nabla f(x(t)).$$

This process has the particularly nice property that it conserves the value of H, and as a result there is no need to apply a Metropolis filter. HMC has been studied in many works [MS17, MV18, LV18]. Mangoubi and Smith [MS17] gave the following guarantee for strongly logconcave densities.

Theorem 4.1.1 ([MS17])**.** *Let f be a smooth, strongly convex function s.t. for all y*

$$m_2 \cdot I \preceq \nabla^2 f(y) \preceq M_2 \cdot I.$$

Then, HMC converges to the density proportional to e^{-f} in $\widetilde{O}((M_2/m_2)^2)$ iterations with each iteration being the exact solution of an ODE.

For the resulting algorithm presented in [MS17], which needs to approximate the solution of the ODE, the number of function evaluations and overall time grow as square-root of the dimension (and a higher polynomial of the condition number). Table 4.1 summarizes related work on sampling logconcave functions with various structural assumptions. In all these cases, even with higher-order smoothness and incoherence assumptions, the number of gradient/function evaluations grows as a polynomial in d. A special case of much interest is Bayesian logistic regression. To address this [MV18] define an incoherence parameter and achieve the previously best dependence on the dimension of $d^{1/4}$ for functions with bounded incoherence. They note that this is nearly optimal for the leapfrog implementation of HMC they use. Improving the complexity further, and in particular the dependence on the dimension d is an important open problem. This brings us to our main motivating question:

For what class of functions can we avoid polynomial dependence on dimension (in an algorithm)? Can we do this for the logistic loss function?

Year	Refs.	Method	Iters.	Gra/Iter	Total time
2006	[LV06b]	B./H.*	d^3, d^4	1	d^5, d^6
2017	[Dal17]	LMC *	$\kappa^2 d$, $\kappa^3 d^3$	1	$\kappa^2 d^2$, $\kappa^3 d^4$
2017	[Dal17]	LMCO *	$\kappa^2 d$, $\kappa^2 d^{2.5}$	1	$\kappa^2 d^4$, $\kappa^2 d^{5.5}$
2018	[CCBJ18]	DL	$\kappa^2 d^{0.5}$	1	$\kappa^2 d^{1.5}$
2018	[DCWY18]	MALA *	κd, κd^2	1	κd^2, κd^3
2017	[MS17]	HMC	$\kappa^{6.5} d^{0.5}$	1	$\kappa^{6.5} d^{1.5}$
2018	[MV18]	HMC *,†	$\kappa^{2.75} d^{0.25}$, $\kappa^{3.5} d^{0.25}$	1	$\kappa^{2.75} d^{1.25}$, $\kappa^{3.5} d^{1.25}$
2018	[LSV18]	HMC †	$\kappa^{1.5}$	1	$\kappa^{1.5} d$
	(thesis)	HMC	$\kappa^{1.5}$	$\kappa^{0.25} d^{0.5}$	$\kappa^{1.75} d^{1.5}$

* have different bounds for warm start and general (cold) start. We stated the runtime for cold start in green color.

† make smoothness and incoherence assumptions motivated by and applicable to Bayesian logistic regression.

Table 4.1: Summary of results, d is the dimension, κ is the condition number of $\nabla^2 f$. "Iters" denotes "the number of iterations / parallel depth". "Gra/Iter" denotes the "the number of gradients per iteration". We use the parallel depth of the algorithm as the number of iterations. We suppress polylogarithmic terms and dependence on the error parameter. Ball walk/hit-and-run apply to general logconcave distributions, the rest assume strongly logconcave with Lipschitz gradient and possibly more. "B./H." denotes "Ball Walk/Hit-and-run". "DL" denotes "Damped Langevin". In all previous work, for simplicity, we report the most favorable bounds by making various assumptions such as $\kappa \ll d$.

4.1.1 Results

We begin with an informal statement of our result for sampling from a class that includes the logistic loss function.

Theorem 4.1.2 (Informal version of Theorem 4.5.6)**.** *Let* $A = [a_1; a_2; \cdots; a_n] \in \mathbb{R}^{n \times d}$, $\phi_i : \mathbb{R} \to \mathbb{R}$ *with its* k*-th derivatives bounded by* $O(1)^k$ *and*

$$f(x) = \sum_{i=1}^{n} \phi_i(a_i^\top x) + \frac{m_2}{2}\|x\|^2.$$

Suppose that $\nabla^2 f$ *has condition number* κ *and* $\tau = \|AA^\top\|_{\infty\to\infty}$. *Then we can find a random point* X *whose Wasserstein distance to* Y *drawn from the density proportional to* e^{-f} *satisfies*

$$W_2(X, Y) \le \frac{\epsilon}{\sqrt{m_2}}$$

using $\widetilde{O}(\kappa^{1.5} + \frac{\tau}{m_2})$ *iterations, where each iteration takes* $\widetilde{O}(d)$ *time and* $\widetilde{O}(1)$ *evaluations of* ∇f.

Remark 4.1.1. The $\frac{1}{\sqrt{m_2}}$ term in the error is needed to make the statement invariant under scaling of f.

For the logistic loss[1] $\phi(t) = \log(1 + e^{-t})$, we have $\phi'(t) = -\frac{1}{1+e^t}$, and it has Cauchy estimate $M = 1$ with radius $r = 1$ (See Lemma 4.9.2). The above result has the following application,

Corollary 4.1.3 (Logistic loss sampling)**.** *Let* $f(x) = \sum_{i=1}^n \phi_i(a_i^\top x) + \frac{m_2}{2}\|x\|^2$ *with* $\phi(t) = \log(1 + e^{-t})$. *Let* $\tau = \|AA^\top\|_{\infty\to\infty}$ *and suppose that* $\nabla^2 f(x) \preceq M_2 \cdot I$ *for all* x. *Starting at the minimum* $x^{(0)}$ *of* f, *we can find a random point* X *whose Wasserstein distance to* Y *drawn from the density proportional to* e^{-f} *satisfies*

$$W_2(X, Y) \le \frac{\epsilon}{\sqrt{m_2}}$$

using $\widetilde{O}(\frac{\tau}{m_2} + \kappa^{1.5})$ *iterations with* $\kappa = \frac{M_2}{m_2}$. *Each iteration takes* $\widetilde{O}(d)$ *time and* $\widetilde{O}(1)$ *matrix-vector multiplications for* A *and* $A^\top$.

Remark 4.1.2. Lemma 4.6.1 shows that $\|AA^\top\|_{\infty\to\infty} = \Theta(\lambda_{\max}(AA^\top))$ for sparse enough matrix $AA^\top$. Since $\lambda_{\max}(AA^\top)$ usually has the same order as M_2, the number of iterations is dominated by the $\kappa^{1.5}$ term.

[1]The logistic function is $g(t) = \frac{1}{1+e^{-t}}$ and the logistic loss is $-\log(g(t)) = \log(1 + e^{-t})$.

The above results extend and improve previous work substantially. First, in all previous algorithms, the number of functions calls was polynomial in the dimension d, while the dependence here is polylogarithmic. Second, our incoherence assumption for logistic regression is simpler and milder. Third, due to the nature of how we implement each step, the parallel depth of the algorithm is just the number of iterations, i.e., polylogarithmic in the dimension and $\widetilde{O}(1)$ when the condition numbers are bounded. Fourth, the runtime and depth of our algorithm depends polynomially in $\log(1/\epsilon)$ while all previous (nearly) linear time algorithms depends polynomially in $1/\epsilon$.

We also give an improved bound on the complexity of sampling from e^{-f} when f is strongly convex and has a Lipschitz gradient (no further smoothness assumptions).

Theorem 4.1.4. (Strongly Convex)**.** Given a function f such that $0 \prec m_2 \cdot I \preceq \nabla^2 f(x) \preceq M_2 \cdot I$ for all $x \in \mathbb{R}^d$ and $0 < \epsilon < \sqrt{d}$. Starting $x^{(0)}$ at the minimum of f, we can find a random point X whose Wasserstein distance to Y drawn from the density proportional to e^{-f} satisfies

$$W_2(X, Y) \leq \frac{\epsilon}{\sqrt{m_2}}$$

using $O(\kappa^{1.5} \log(\frac{d}{\epsilon}))$ iterations where $\kappa = \frac{M_2}{m_2}$. Each iteration takes $O\left(\frac{\kappa^{\frac{1}{4}} d^{\frac{3}{2}}}{\epsilon} \log\left(\frac{\kappa d}{\epsilon}\right)\right)$ time and $O\left(\frac{\kappa^{\frac{1}{4}} d^{\frac{1}{2}}}{\epsilon} \log\left(\frac{\kappa d}{\epsilon}\right)\right)$ evaluations of ∇f, amortized over all iterations.

The previous best bound was $\kappa^2\sqrt{d}$ iterations [CCBJ18]. This result is one of the key surprises of this chapter. Although this problem has been studied extensively with specifically-designed algorithms and analysis, we show how to get a better result by a general ODE algorithm and a general analysis which works for any ODE. Furthermore, *our algorithm is the first to achieve polylogarithmic depth dependence on the dimension, which seemed impossible in prior work.*

The above results are based on three ingredients: (1) a new contraction rate for HMC of $\kappa^{1.5}$, improving on the previous best bound of κ^2 (2) a proof that a solution to ODE's arising from HMC applied to the above problem are approximated by (piecewise) low-degree polynomials and (3) a fast (nearly linear time and polylog parallel depth) algorithm for solving multivariate second-order ODEs.

We next present the multivariate high-order ODE guarantee. This generalizes and improves on the guarantee from [LV17]. While we state it below for the case of the piecewise polynomial basis of functions, it applies to any basis of functions. This is a general result about solving ODE efficiently, independent of the application to sampling. The only assumptions needed are that the ODE function is Lipschitz and that the solution is close to the span of small number of basis of functions. These natural assumptions suffice to get around the worst-case complexity lower bounds for solving such general ODEs [KF82, Ko83, Ko10, Kaw10, KC12].

Theorem 4.1.5 (Informal version of Theorem 4.2.4 for 1st order ODE)**.** *Let $x^*(t) \in \mathbb{R}^d$ be the solution of the ODE*

$$\frac{\mathrm{d}}{\mathrm{d}t}x(t) = F(x(t), t), x(0) = v$$

where $F : \mathbb{R}^{d+1} \to \mathbb{R}^d$, $x(t) \in \mathbb{R}^d$ and $v \in \mathbb{R}^d$. Given some L and $\epsilon > 0$ such that

1. There exists a piece-wise polynomial $q(t)$ such that $q(t)$ on $[T_{j-1}, T_j]$ is a degree D_j polynomial with

$$0 = T_0 < T_1 < \cdots < T_n = T$$

and that

$$\left\| q(t) - \frac{\mathrm{d}}{\mathrm{d}t} x^*(t) \right\| \leq \frac{\epsilon}{T}, \forall t \in [0, T]$$

2. *The algorithm knows about the intervals* $[T_{j-1}, T_j]$ *and the degree* D_j *for all* $j \in [n]$.

3. *For any* $y, z \in \mathbb{R}^d$,

$$\|F(y,t) - F(z,t)\| \leq L\|y - z\|, \forall t \in [0, T].$$

Assume $LT \leq 1/16000$. *Then, we can find a piece-wise polynomial* $x(t)$ *such that*

$$\max_{t \in [0,T]} \|x(t) - x^*(t)\| \lesssim \epsilon.$$

using $\widetilde{O}\left(\sum_{i=1}^n (1 + D_i)\right)$ *evaluations of* F *and* $\widetilde{O}\left(d \sum_{i=1}^n (1 + D_i)\right)$ *time.*

We suspect these methods will be useful in many other settings beyond the focus application of this chapter. Moreover, the result is nearly optimal. Roughly speaking, it says that the complexity of solving the ODE is nearly the same as the complexity of representing the solution. The assumption that $LT < 1$ is essential, as otherwise after longer time, the solution can blow up exponentially. Also, the assumption on the piece-wise polynomial approximation has a certain universality since this is how one implicitly represents a function using any iterative method. Finally, each iteration of the ODE algorithm, the collocation method, can be fully parallelized; as a result the parallel time complexity of the sampling algorithms in this chapter are polylogarithmic in the dimension.

4.1.2 HMC and improved contraction rate

We give an improved contraction rate for HMC, stated explicitly as Algorithm 4.1. We

Algorithm 4.1 Hamiltonian Monte Carlo Algorithm

procedure HMC($x^{(0)}, f, \epsilon, h$) ▷ Theorem 4.3.2

Suppose that f is m_2 strongly convex with M_2 Lipschitz gradient on $\mathbb{R}^d$.

Assume that the step size $h \leq \frac{m_2^{1/4}}{2M_2^{3/4}}$.

Let the number of iterations $N = \frac{1}{\theta} \cdot \log\left(\frac{4}{\epsilon^2}\left(\frac{\|\nabla f(x^{(0)})\|_2^2}{m_2} + d\right)\right)$ with $\theta = \frac{m_2 h^2}{8}$.

For $k = 1, 2, \cdots, N$ **do**

Generate a Gaussian random direction $v \sim \mathcal{N}(0, I_d)$.

Let $x(t)$ be the HMC defined by

$$\frac{\mathrm{d}^2 x}{\mathrm{d}t^2} = -\nabla f(x), \frac{\mathrm{d}x}{\mathrm{d}t}(0) = v, x(0) = x^{(k-1)}.$$

Find a point $x^{(k)}$ such that ▷ Theorem 4.2.4

$$\|x^{(k)} - x(h)\|_2 \leq \bar{\epsilon} := \frac{\theta \cdot \epsilon}{2\sqrt{m_2}}.$$

end for

return $x^{(N)}$.

end procedure

give two contraction bounds for the ideal HMC. The first bound is $(\frac{m_2}{M_2})^{1.5}$ using $T \sim \frac{m_2^{1/4}}{M_2^{3/4}}$. The second bound shows that there is a T that gives the optimal contraction bound $\frac{m_2}{M_2}$. However, as we will see this part cannot be used to bound the overall mixing time, because the time T depends on the point we use for coupling, which is unknown to the algorithm. We keep this as evidence for a possible $\frac{m_2}{M_2}$ bound. The improvement is from κ^2 in previous work to $\kappa^{1.5}$.

Lemma 4.1.6 (Contraction bound for HMC)**.** *Let $x(t)$ and $y(t)$ be the solution of HMC dynamics on e^{-f} starts at $x(0)$ and $y(0)$ with initial direction $x'(0) = y'(0) = v$ for some vector v. Suppose that f is m_2 strongly convex with M_2 Lipschitz gradient., i.e., $m_2 \cdot I \preceq \nabla^2 f(x) \preceq M_2 \cdot I$ for all x. Then, for $0 \leq t \leq \frac{m_2^{1/4}}{2M_2^{3/4}}$, we have that*

$$\|x(t) - y(t)\|_2^2 \leq \left(1 - \frac{m_2}{4}t^2\right) \|x(0) - y(0)\|_2^2.$$

Furthermore, there is $t \geq 0$ depending on $f, x(0), y(0)$, and v such that

$$\|x(t) - y(t)\|_2^2 \leq \left(1 - \frac{1}{16}\frac{m_2}{M_2}\right) \|x(0) - y(0)\|_2^2.$$

4.1.3 Techniques

In this chapter we give bounds on the Collocation Method for solving ODEs. To ensure the algorithm is applicable to the ODE's that arise in the sampling application, we need to show that the solution of the ODE is close to a low-rank basis. Given only bounds on the Hessian of a function, we do this by approximating the solution of the ODE with a piecewise degree two polynomial. For smooth functions, we can use low-degree polynomials.

The proofs of the degree bounds go via the Cauchy-Kowalevsky method, by showing bounds on all derivatives at the initial point. To do this for multivariate ODE's, we use

the method of majorants, and reduce the problem to bounding the radius of convergence of one-variable ODEs.

The improved convergence guarantees for exact HMC are also based on better analysis of the underlying ODE, showing that a larger step size than previously known is possible.

For many optimization and sampling methods, there are corresponding customized versions that deal with decomposable functions by sampling terms of the functions. These algorithms usually take nearly linear time with the number of iterations being polynomial in the dimension. Often, an improvement in the general case would lead to an improvement in the decomposable case. To limit the length of this chapter, we focus only on results with polylogarithmic depth. Therefore, in Table 4.1, we do not list algorithms that involve sampling each term in decomposable functions [BFR16, DSM+16, DRW+16, BFFN17, DK17, CWZ+17, NDH+17, CFM+18]. We expect our techniques can be further improved for decomposable functions by sampling individual terms.

Outline of chapter. Then we give the main ODE algorithm and guarantees in Section 4.2. We give the proof of the improved convergence bound for HMC in Section 4.3. We use both parts to obtain improved guarantees for sampling strongly logconcave functions with Lipschitz gradients in Section 4.4 and smooth functions, including logistic loss in Section 4.5.

Some preliminaries including standard definitions and well-known theorems about ODEs are in an appendix. Remaining proofs about ODEs are in Appendix 4.7. In Appendix 4.8, we present some useful tools for Cauchy estimates. Appendix 4.9 shows how to calculate the Cauchy estimates of some function which are extensively used in practice.

4.2 ODE Solver for any basis

In this section, we analyze the collocation method for solving ODEs [Ise09]. This method is classical in numerical analysis. The goal of this section is provide an introduction of this method and provide a non-asymptotic bounds for this method. In [LV17, LV18], we applied this method to obtain faster algorithms for sampling on polytopes. Unfortunately, the particular version of collocation method we used assume the solution can be approximated by a low-degree polynomial, which heavily restrict the set of functions we can sample.

To give an intuition for the collocation method, we first consider the following first-order ODE

$$\frac{\mathrm{d}}{\mathrm{d}t}x(t) = F(x(t), t), \quad \forall 0 \leq t \leq T,$$
$$x(0) = v.$$

where $F : \mathbb{R}^{d+1} \to \mathbb{R}$. The collocation method is partly inspired by the Picard-Lindelöf theorem, a constructive existence proof for a large class of ODE. Therefore, we will first revisit the proof of Picard-Lindelöf theorem for first-order ODE.

4.2.1 Picard-Lindelöf theorem

In general, we can rewritten the first-order ODE as an integral equation

$$x(t) = v + \int_0^t F(x(s), s)\mathrm{d}s \quad \text{for all } 0 \leq t \leq T.$$

To simplify the notation, we use $\mathcal{C}([0, T], \mathbb{R}^d)$ to denote $\mathbb{R}^d$-valued functions on $[0, T]$. We define the operator $\mathcal{T}$ from $\mathcal{C}([0, T], \mathbb{R}^d)$ to $\mathcal{C}([0, T], \mathbb{R}^d)$ by

$$\mathcal{T}(x)(t) = v + \int_0^t F(x(s), s)\mathrm{d}s \quad \text{for all } 0 \leq t \leq T. \tag{4.1}$$

Therefore, the integral equation is simply $x = \mathcal{T}(x)$.

Banach fixed point theorem shows that the integral equation $x = \mathcal{T}(x)$ has a unique solution if there is a norm, and $j \in \mathbb{N}$ such that the map $\mathcal{T}^{\circ j}$ has Lipschitz constant less than 1. Recall that $\mathcal{T}^{\circ j}$ is the composition of j many $\mathcal{T}$, i.e.,

$$\mathcal{T}^{\circ j}(x) = \underbrace{\mathcal{T}(\mathcal{T}(\cdots \mathcal{T}}_{j \text{ many } \mathcal{T}}(x)\cdots)).$$

Picard-Lindelöf theorem shows that if F is Lipschitz in x, then the map $\mathcal{T}^{\circ j}$ has Lipschitz constant less than 1 for some positive integer j.

Lemma 4.2.1. *Given any norm $\|\cdot\|$ on $\mathbb{R}^d$. Let L be the Lipschitz constant of F in x, i.e.*

$$\|F(x,s) - F(y,s)\| \le L\|x - y\| \quad \textit{for all } x, y \in \mathbb{R}^d, s \in [0,T].$$

For any $x \in \mathcal{C}([0,T], \mathbb{R}^d)$, we define the corresponding norm

$$\|x\| \stackrel{\text{def}}{=} \max_{0 \le t \le T} \|x(t)\|.$$

Then, the Lipschitz constant of $\mathcal{T}^{\circ j}$ in this norm is upper bounded by $(LT)^j/j!$.

Proof. We prove this lemma with a stronger induction statement

$$\|(\mathcal{T}^{\circ j}\, x)(h) - (\mathcal{T}^{\circ j}\, y)(h)\| \le \frac{L^j h^j}{j!}\|x - y\| \quad \text{for all } 0 \le h \le T.$$

The base case $j = 0$ is trivial. For the induction case j, we can upper bound the term

as follows

$$
\begin{aligned}
&\| \mathcal{T}^{\circ j} x(h) - \mathcal{T}^{\circ j} y(h)\| \\
&= \| \mathcal{T}\, \mathcal{T}^{\circ (j-1)} x(h) - \mathcal{T}\, \mathcal{T}^{\circ (j-1)} y(h)\| \\
&\leq \int_0^h \|F(\mathcal{T}^{\circ (j-1)} x(s), s) - F(\mathcal{T}^{\circ (j-1)} y(s), s)\| \mathrm{d}s \\
&\leq L \int_0^h \| \mathcal{T}^{\circ (j-1)} x(s) - \mathcal{T}^{\circ (j-1)} y(s)\| \mathrm{d}s && \text{by } f \text{ is } L \text{ Lipschitz} \\
&\leq L \int_0^h \frac{L^{j-1} s^{j-1}}{(j-1)!} \|x - y\| \mathrm{d}s && \text{by the induction statement} \\
&= \frac{L^j}{j!} h^j \|x - y\|.
\end{aligned}
$$

This completes the induction. □

4.2.2 Intuition of collocation method

To make the Picard-Lindelöf theorem algorithmic, we need to discuss how to represent a function in $\mathcal{C}([0, T], \mathbb{R}^d)$. One standard way is to use a polynomial $p_i(t)$ in t for each coordinate $i \in [d]$. In this section, we assume that there is a basis $\{\varphi_j\}_{j=1}^D \subset \mathcal{C}([0, T], \mathbb{R})$ such that for all $i \in [d]$, $\frac{\mathrm{d}x_i}{\mathrm{d}t}$ is approximated by some linear combination of $\varphi_j(t)$.

For example, if $\varphi_j(t) = t^{j-1}$ for $j \in [d]$, then our assumption is simply saying $\frac{\mathrm{d}x_i}{\mathrm{d}t}$ is approximated by a degree $D - 1$ polynomial. Other possible basis are piecewise-polynomial and Fourier series. By Gram-Schmidt orthogonalization, we can always pick nodes point $\{c_i\}_{j=1}^D$ such that

$$\varphi_j(c_i) = \delta_{i,j} \quad \text{for } i, j \in [D].$$

The benefit of such basis is that for any $f \in \mathrm{span}(\varphi_j)$, we have that $f(t) = \sum_{j=1}^D f(c_j)\varphi_j(t)$.

For polynomials, the resulting basis are the Lagrange polynomials

$$\varphi_j(t) = \prod_{i \in [D] \setminus \{j\}} \frac{t - c_i}{c_j - c_i} \quad \text{for } j \in [D].$$

The only assumption we make on the basis is that its integral is bounded.

Definition 4.2.1. Given a D dimensional subspace $\mathcal{V} \subset \mathcal{C}([0,T], \mathbb{R})$ and node points $\{c_j\}_{j=1}^D \subset [0,T]$. For any $\gamma_\varphi \geq 1$, we call a basis $\{\varphi_j\}_{j=1}^D \subset \mathcal{V}$ is γ_φ bounded if $\varphi_j(c_i) = \delta_{i,j}$ and we have

$$\sum_{j=1}^{D} \left| \int_0^t \varphi_j(s) \mathrm{d}s \right| \leq \gamma_\varphi T \quad \text{for } t \in [0,T].$$

Note that if the constant function $1 \in \mathcal{V}$, then we have

$$1 = \sum_{j=1}^{D} 1(c_j)\varphi_j(t) = \sum_{j=1}^{D} \varphi_j(t).$$

Hence, we have

$$T = \int_0^T 1 \mathrm{d}s \leq \sum_{j=1}^{D} \left| \int_0^T \varphi_j(s) \mathrm{d}s \right| \leq \gamma_\varphi T.$$

Therefore, $\gamma_\varphi \geq 1$ for most of the interesting basis. This is the reason why we simply put it as an assumption to shorten some formulas.

In the section 4.2.5, we prove that for the space of low degree polynomial and piecewise low degree polynomial, there is a basis on the Chebyshev nodes that is $O(1)$ bounded.

Assuming that $\frac{dx}{dt}$ can be approximated by some element in $\mathcal{V}$, we have that

$$\frac{\mathrm{d}x}{\mathrm{d}t}(t) \sim \sum_{j=1}^{D} \frac{\mathrm{d}x}{\mathrm{d}t}(c_j)\varphi_j(t) = \sum_{j=1}^{D} F(x(c_j), c_j)\varphi_j(t).$$

Integrating both sides, we have

$$x(t) \approx v + \int_0^t \sum_{j=1}^{D} F(x(c_j), c_j)\varphi_j(s)\mathrm{d}s. \tag{4.2}$$

This inspires us to consider the following operator from $\mathcal{C}([0,T], \mathbb{R}^d)$ to $\mathcal{C}([0,T], \mathbb{R}^d)$:

$$\mathcal{T}_\varphi(x)(t)_i = v_i + \int_0^t \sum_{j=1}^{D} F(x(c_j), c_j)_i\varphi_j(s)\mathrm{d}s \quad \text{for } i \in [d]. \tag{4.3}$$

Equation (4.2) can be written as $x \approx \mathcal{T}_\varphi(x)$. To find x to satisfies this, naturally, one can apply the fix point iteration and this is called the collocation method.

4.2.3 Collocation method

From the definition of (4.3), we note that $\mathcal{T}_\varphi(x)$ depends only on $x(t)$ at $t = c_j$. Therefore, we can only need to calculate $\mathcal{T}_\varphi(x)(t)$ at $t = c_j$. To simplify the notation, for any $x \in \mathcal{C}([0,T], \mathbb{R}^d)$, we define a corresponding matrix $[x] \in \mathbb{R}^{d\times D}$ by $[x]_{i,j} = x_i(c_j)$. For any $d \times D$ matrix X, we define $F(X, c)$ as an $d \times D$ matrix

$$F(X, c)_{i,j} = F(X_{*,j}, c_j)_i. \tag{4.4}$$

where $X_{*,j}$ is the j-th column of X. Finally, we define A_φ as a $D \times D$ matrix

$$(A_\varphi)_{i,j} = \int_0^{c_j} \varphi_i(s)\mathrm{d}s. \tag{4.5}$$

By inspecting the definition of (4.3), (4.4) and (4.5), we have that

$$[\mathcal{T}_\varphi(x)] = v \cdot 1_D^\top + F([x], c)A_\varphi$$

where 1_D is a column of all 1 vector of length D. Hence, we can apply the map $\mathcal{T}_\varphi$ by simply multiply $F([x], c)$ by a pre-compute $D \times D$ matrix A_φ. For the basis we considered

Algorithm 4.2 Collocation Method

procedure CollocationMethod(F, v, T, φ, c) ▷ Theorem 4.2.3
  Let $N = \left\lceil \log\left(\frac{T}{\epsilon}\max_{s\in[0,T]}\|F(v,s)\|\right)\right\rceil$ ▷ Choose number of iterations
  Let A_φ be the matrix defined by $(A_\varphi)_{i,j} = \int_0^{c_j}\varphi_i(s)\mathrm{d}s$.
  $X^{(0)} \leftarrow v \cdot 1_D^\top$. ▷ 1_D is a column of all 1 vector of length D
  for $j = 1, 2, \cdots, N-1$ **do**
  $X^{(j)} \leftarrow v \cdot 1_D^\top + F(X^{(j-1)}, c)A_\varphi$. ▷ Matrix $F(X,c)$ is defined in Eq. (4.4)
    ▷ Note that we evaluate D many F every iteration in this matrix notation.
  end for
  $x^{(N)}(t) \leftarrow v + \int_0^t \sum_{i=1}^D F(X_{*,i}^{(N)}, c_i)\varphi_i(s)\mathrm{d}s$
  return $x^{(N)}$
end procedure

in this chapter, each iteration takes only $\widetilde{O}(dD)$ which is nearly linear to the size of our representation of the solution.

We state our guarantee for a first-order ODE (Algorithm 4.2).

Theorem 4.2.2 (First order ODE)**.** *Let $x^*(t)$ be the solution of an d dimensional ODE*

$$x(0) = v, \frac{\mathrm{d}x(t)}{\mathrm{d}t} = F(x(t), t) \quad \textit{for all } 0 \le t \le T.$$

We are given a D dimensional subspace $\mathcal{V} \subset \mathcal{C}([0,T], \mathbb{R})$, node points $\{c_j\}_{j=1}^D \subset [0,T]$ and a γ_φ bounded basis $\{\varphi_j\}_{j=1}^D \subset \mathcal{V}$ (Definition 4.2.1). Given some L and $\epsilon > 0$ such that

1. *There exists a function $q \in \mathcal{V}$ such that*

$$\left\| q(t) - \frac{\mathrm{d}}{\mathrm{d}t}x^*(t) \right\| \le \frac{\epsilon}{T}, \forall t \in [0,T].$$

2. *For any $y, z \in \mathbb{R}^d$,*

$$\|F(y,t) - F(z,t)\| \le L\|y - z\|, \forall t \in [0,T].$$

Assume $\gamma_\varphi LT \le 1/2$. Then the algorithm CollocationMethod *(Algorithm 4.2) outputs a function $x^{(N)} \in \mathcal{V}$ such that*

$$\max_{t\in[0,T]} \|x^{(N)}(t) - x^*(t)\| \le 20\gamma_\varphi \epsilon.$$

The algorithm takes $O\left(D \log\left(\frac{T}{\epsilon} \max_{s\in[0,T]} \|F(v,s)\|\right)\right)$ evaluations of F.

Next we state the general result for a k-th order ODE. We prove this via a reduction from higher order ODE to first-order ODE. See the proof in Appendix 4.7.

Theorem 4.2.3 (k-th order ODE)**.** *Let $x^*(t) \in \mathbb{R}^d$ be the solution of the ODE*

$$\frac{\mathrm{d}^k}{\mathrm{d}t^k} x(t) = F\left(\frac{\mathrm{d}^{k-1}}{\mathrm{d}t^{k-1}} x(t), \cdots, x(t), t\right)$$
$$\frac{\mathrm{d}^i}{\mathrm{d}t^i} x(0) = v_i, \forall i \in \{k-1, \cdots, 1, 0\}.$$

where $F : \mathbb{R}^{kd+1} \to \mathbb{R}^d$, $x(t) \in \mathbb{R}^d$, and $v_0, v_1, \cdots, v_{k-1} \in \mathbb{R}^d$.

We are given a D dimensional subspace $\mathcal{V} \subset \mathcal{C}([0,T], \mathbb{R})$, node points $\{c_j\}_{j=1}^D \subset [0,T]$ and a γ_φ bounded basis $\{\varphi_j\}_{j=1}^D \subset \mathcal{V}$ (Definition 4.2.1). Given some L and $\epsilon > 0$ such that

1. *For $i \in [k]$, there exists a function $q^{(i)} \in \mathcal{V}$ such that*

$$\left\| q^{(i)}(t) - \frac{\mathrm{d}^i}{\mathrm{d}t^i} x^*(t) \right\| \le \frac{\epsilon}{T^i}, \forall t \in [0,T].$$

2. *For any $y, z \in \mathbb{R}^{kd}$,*

$$\|F(y,t) - F(z,t)\| \le \sum_{i=1}^k L_i \|y_i(t) - z_i(t)\|, \forall t \in [0,T].$$

Assume $\gamma_\varphi LT \le 1/8$ *with* $L = \sum_{i=1}^k L_i^{1/i}$. *Then, we can find functions* $\{q^{(i)}\}_{i\in\{0,1,\cdots,k-1\}} \subset \mathcal{V}$ *such that*

$$\max_{t\in[0,T]} \left\| q^{(i)}(t) - \frac{\mathrm{d}^i}{\mathrm{d}t^i} x^*(t) \right\|_p = 20(1+2k)\gamma_\varphi \frac{\epsilon}{T^i}, \forall i \in \{0,1,\cdots,k-1\}.$$

The algorithm takes $O(D\log(C/\epsilon))$ *evaluations of* F *where*

$$C = (4\gamma_\varphi T)^k \cdot \max_{s\in[0,T]} \|F(v_{k-1}, v_{k-2}, \cdots, v_0, s)\| + \sum_{i=1}^{k-1} (4\gamma_\varphi T)^i \|v_i\|.$$

Note that the statement is a bit awkward. Instead of finding a function whose derivatives are same as the derivatives of x^*, the algorithm approximates the derivatives of x^* individually. This is because we do not know if derivatives/integrals of functions in $\mathcal{V}$ remain in $\mathcal{V}$. For piece-wise polynomials, we can approximate the j-th derivative of the solution by taking $(k-j)$-th iterated integral of $q^{(k)}$, which is still a piece-wise polynomial.

In section 4.2.5, we give a basis for piece-wise polynomials (Lemma 4.2.8). Using this basis, we have the following Theorem.

Theorem 4.2.4. (k-th order ODE) Let $x^*(t) \in \mathbb{R}^d$ be the solution of the ODE

$$\frac{\mathrm{d}^k}{\mathrm{d}t^k} x(t) = F\left(\frac{\mathrm{d}^{k-1}}{\mathrm{d}t^{k-1}} x(t), \cdots, x(t), t\right)$$
$$\frac{\mathrm{d}^i}{\mathrm{d}t^i} x(0) = v_i, \forall i \in \{k-1, \cdots, 1, 0\}.$$

where $F : \mathbb{R}^{kd+1} \to \mathbb{R}^d$, $x(t) \in \mathbb{R}^d$, and $v_0, v_1, \cdots, v_{k-1} \in \mathbb{R}^d$. Given some L and $\epsilon > 0$ such that

1. There exists a piece-wise polynomial $q(t)$ such that $q(t)$ on $[T_{j-1}, T_j]$ is a degree D_j polynomial with

$$0 = T_0 < T_1 < \cdots < T_n = T$$

and that

$$\left\| q(t) - \frac{\mathrm{d}^k}{\mathrm{d}t^k} x^*(t) \right\| \leq \frac{\epsilon}{T^k}, \forall t \in [0, T]$$

2. The algorithm knows about the intervals $[T_{j-1}, T_j]$ and the degree D_j for all $j \in [n]$.

3. For any $y, z \in \mathbb{R}^{kd}$,

$$\|F(y,t) - F(z,t)\| \leq \sum_{i=1}^{k} L_i \|y_i - z_i\|, \forall t \in [0, T].$$

Assume $LT \leq 1/16000$ with $L = \sum_{i=1}^{k} L_i^{1/i}$. Then, we can find a piece-wise polynomial $x(t)$ such that

$$\max_{t \in [0,T]} \left\| \frac{\mathrm{d}^i}{\mathrm{d}t^i} x(t) - \frac{\mathrm{d}^i}{\mathrm{d}t^i} x^*(t) \right\|_p \lesssim \frac{\epsilon k}{T^i}, \forall i \in \{0, 1, \cdots, k-1\}.$$

using $O(D \log(C/\epsilon))$ evaluations of F with the size of basis $D = \sum_{i=1}^{n} (1 + D_i)$ and

$$O\left(d \min\left(\sum_{i=1}^{n} (1 + D_i)^2, D \log(CD/\epsilon) \right) \log(C/\epsilon) \right)$$

time where

$$C = O(T)^k \cdot \max_{s \in [0,T]} \|F(v_{k-1}, v_{k-2}, \cdots, v_0, s)\| + \sum_{i=1}^{k-1} O(T)^i \|v_i\|.$$

Remark 4.2.1. The two different runtime come from two different ways to the integrate of basis in Lemma 4.2.8. The first one is an navie method which is good enough for all our application. The second one follows from multipole method which gives an nearly linear time to the size of the basis with an extra log dependence on the accuracy.

In the rest of this section, we prove the first-order guarantee, Theorem 4.2.2

4.2.4 Proof of first order ODE

First, we bound the Lipschitz constant of the map $\mathcal{T}_\varphi$. Unlike the Picard-Lindelöf theorem, we are not able to get an improved bound of the Lipschitz constant of the composite of $\mathcal{T}_\varphi$. Fortunately, the Lipschitz constant of the map $\mathcal{T}_\varphi$ is good enough for all applications in this chapter.

Lemma 4.2.5. *Given any norm $\|\cdot\|$ on $\mathbb{R}^d$. Let L be the Lipschitz constant of F in x, i.e.*

$$\|F(x,s) - F(y,s)\| \le L\|x-y\| \quad \textit{for all } x,y \in \mathbb{R}^d, s \in [0,T].$$

Then, the Lipschitz constant of $\mathcal{T}_\varphi$ in this norm is upper bounded by $\gamma_\varphi LT$.

Proof. For any $0 \le t \le T$,

$$\begin{aligned}
\|\mathcal{T}_\varphi(x)(t) - \mathcal{T}_\varphi(y)(t)\| &= \left\| \int_0^t \sum_{j=1}^D F(x(c_j),c_j)\varphi_j(s)\mathrm{d}s - \int_0^t \sum_{j=1}^D F(y(c_j),c_j)\varphi_j(s)\mathrm{d}s \right\| \\
&\le \sum_{j=1}^D \left| \int_0^t \varphi_j(s)\mathrm{d}s \right| \cdot \max_{t\in[0,T]} \|F(x(t),t) - F(y(t),t)\| \\
&\le \gamma_\varphi LT \cdot \max_{s\in[0,t]} \|x(t) - y(t)\| \\
&\le \gamma_\varphi LT \|x-y\|.
\end{aligned}$$

where the third step follows by $\sum_{j=1}^D |\int_0^t \varphi_j(s)\mathrm{d}s| \le \gamma_\varphi T$ for all $0 \le t \le T$. □

For the rest of the proof, let x_φ^* denote the fixed point of $\mathcal{T}_\varphi$, i.e., $\mathcal{T}_\varphi(x_\varphi^*) = x_\varphi^*$. The Banach fixed point theorem and Lemma 4.2.5 shows that x_φ^* uniquely exists if $T \le \frac{1}{L\gamma_\varphi}$.

Let x^* denote the solution of the ODE, i.e., the fixed point of $\mathcal{T}$, with $\mathcal{T}(x^*) = x^*$. Let $x^{(0)}$ denote the initial solution given by $x^{(0)}(t) = v$ and $x^{(N)}$ denote the solution obtained by

applying operator $\mathcal{T}_\varphi$ for N times. Note that $x^{(N)}(t)$ is the output of CollocationMethod in Algorithm 4.2.

Let $q \in \mathcal{V}$ denote an approximation of $\frac{\mathrm{d}}{\mathrm{d}t}x^*$ such that

$$\left\| q(t) - \frac{\mathrm{d}}{\mathrm{d}t}x^*(t) \right\| \leq \frac{\epsilon}{T}. \forall t \in [0, T]$$

The next lemma summarizes how these objects are related and will allow us to prove the main guarantee for first-order ODEs.

Lemma 4.2.6. *Let $L^{(j)}$ be the Lipschitz constant of the map $\mathcal{T}_\varphi^{\circ j}$. Assume that $L^{(N)} \leq 1/2$. Then, we have*

$$\begin{aligned}
\|x^{(N)} - x^*\| &\leq L^{(N)}\|x^{(0)} - x^*\| + 2\|x^*_\varphi - x^*\|, && (4.6)\\
\|x^*_\varphi - x^*\| &\leq 2 \cdot \| \mathcal{T}_\varphi^{\circ N}(x^*) - x^*\|, && (4.7)\\
\|x^* - \mathcal{T}_\varphi^{\circ N}(x^*)\| &\leq \sum_{i=0}^{N-1} L^{(i)} \cdot \|x^* - \mathcal{T}_\varphi(x^*)\|, && (4.8)\\
\|x^* - \mathcal{T}_\varphi(x^*)\| &\leq 2\gamma_\varphi \cdot \epsilon. && (4.9)
\end{aligned}$$

Proof. We prove the claims in order.

For the first claim,

$$\begin{aligned}
\|x^{(N)} - x^*\| &\leq \|x^{(N)} - x^*_\varphi\| + \|x^*_\varphi - x^*\| && \text{by triangle inequality}\\
&= \| \mathcal{T}_\varphi^{\circ N}(x^{(0)}) - \mathcal{T}_\varphi^{\circ N}(x^*_\varphi)\| + \|x^*_\varphi - x^*\|\\
&\leq L^{(N)}\|x^{(0)} - x^*_\varphi\| + \|x^*_\varphi - x^*\|\\
&\leq L^{(N)}\|x^{(0)} - x^*\| + L^{(N)}\|x^* - x^*_\varphi\| + \|x^*_\varphi - x^*\|\\
&\leq L^{(N)}\|x^{(0)} - x^*\| + 2\|x^*_\varphi - x^*\|
\end{aligned}$$

where the last step follows by $L^{(N)} \leq 1$.

For the second claim,

$$\begin{aligned}
\|x_\varphi^* - x^*\| &= \| \mathcal{T}_\varphi^{\circ N}(x_\varphi^*) - x^*\| && \text{by } x_\varphi^* = \mathcal{T}_\varphi^{\circ N}(x_\varphi^*) \\
&\leq \| \mathcal{T}_\varphi^{\circ N}(x_\varphi^*) - \mathcal{T}_\varphi^{\circ N}(x^*)\| + \| \mathcal{T}_\varphi^{\circ N}(x^*) - x^*\| && \text{by triangle inequality} \\
&\leq L^{(N)} \cdot \|x_\varphi^* - x^*\| + \| \mathcal{T}_\varphi^{\circ N}(x^*) - x^*\| && \text{by the definition of } L^{(N)} \\
&\leq \frac{1}{2}\|x_\varphi^* - x^*\| + \| \mathcal{T}_\varphi^{\circ N}(x^*) - x^*\| && \text{by } L^{(N)} \leq 1/2
\end{aligned}$$

For the third claim,

$$\begin{aligned}
\|x^* - \mathcal{T}_\varphi^{\circ N}(x^*)\| &\leq \sum_{i=0}^{N-1} \| \mathcal{T}_\varphi^{\circ i}(x^*) - \mathcal{T}_\varphi^{\circ (i+1)}(x^*)\| \\
&\leq \sum_{i=0}^{N-1} L^{(i)} \cdot \|x^* - \mathcal{T}_\varphi(x^*)\|
\end{aligned}$$

For the last claim,

$$
\begin{aligned}
&\|x^*(t) - \mathcal{T}_\varphi(x^*)(t)\| \\
=& \|\mathcal{T}(x^*)(t) - \mathcal{T}_\varphi(x^*)(t)\| \\
=& \left\| \int_0^t F(x^*(s), s)\mathrm{d}s - \int_0^t \sum_{j=1}^D F(x^*(c_j), c_j)\varphi_j(s)\mathrm{d}s \right\| \\
=& \left\| \int_0^t \frac{\mathrm{d}}{\mathrm{d}t}x^*(s)\mathrm{d}s - \int_0^t \sum_{j=1}^D \frac{\mathrm{d}}{\mathrm{d}t}x^*(c_j)\varphi_j(s)\mathrm{d}s \right\| \\
\leq& \left\| \int_0^t (\frac{\mathrm{d}}{\mathrm{d}t}x^*(s) - q(s))\mathrm{d}s - \int_0^t \sum_{j=1}^D (\frac{\mathrm{d}}{\mathrm{d}t}x^*(c_j) - q(c_j))\varphi_j(s)\mathrm{d}s \right\| \\
+& \left\| \int_0^t q(s)\mathrm{d}s - \int_0^t \sum_{j=1}^D q(c_j)\varphi_j(s)\mathrm{d}s \right\| \\
\leq& \int_0^t \left\| \frac{\mathrm{d}}{\mathrm{d}t}x^*(s) - q(s) \right\| \mathrm{d}s + \sum_{j=1}^D \left\| \frac{\mathrm{d}}{\mathrm{d}t}x^*(c_j) - q(c_j) \right\| \left| \int_0^t \varphi_j(s)\mathrm{d}s \right| + 0 \\
\leq& (1 + \gamma_\varphi) \cdot \epsilon + 0
\end{aligned}
$$

where the first step follows by $\mathcal{T}(x^*) = x^*$, the second step follows by the definition of $\mathcal{T}$ and $\mathcal{T}_\varphi$, the third step follows by $x^*(t)$ is the solution of ODE, the fourth step follows by triangle inequality, the second last step follows by $q \in \mathcal{V}$, and the last step follows by $\|\frac{\mathrm{d}}{\mathrm{d}t}x^* - q\| \leq \frac{\epsilon}{T}$ and the definition of γ_φ. □

Now, we are ready to prove Theorem 4.2.2.

Proof. Using Lemma 4.2.6, we have

$$\begin{aligned}
\|x^{(N)} - x^*\| &\le L^{(N)}\|x^{(0)} - x^*\| + 2\|x^*_\varphi - x^*\| && \text{by Eq. (4.6)} \\
&\le L^{(N)}\|x^{(0)} - x^*\| + 4\|\mathfrak{T}^{\circ N}_\varphi(x^*) - x^*\| && \text{by Eq. (4.7)} \\
&\le L^{(N)}\|x^{(0)} - x^*\| + 4\sum_{i=0}^{N-1} L^{(i)} \cdot \|x^* - \mathfrak{T}_\varphi(x^*)\| && \text{by Eq. (4.8)} \\
&\le L^{(N)}\|x^{(0)} - x^*\| + 8\sum_{i=0}^{N-1} L^{(i)} \cdot \gamma_\varphi \cdot \epsilon. && \text{by Eq. (4.9)}
\end{aligned}$$

Using the assumption that $\gamma_\varphi LT \le \frac{1}{2}$, Lemma 4.2.5 shows that $L^{(1)} \le \frac{1}{2}$ and hence $L^{(j)} \le \frac{1}{2^j}$. Therefore, we have

$$\|x^{(N)} - x^*\| \le \frac{1}{2^N}\|x^{(0)} - x^*\| + 16\gamma_\varphi \cdot \epsilon = \frac{1}{2^N}\|x^* - x^*(0)\| + 16\gamma_\varphi \cdot \epsilon \tag{4.10}$$

To bound $\|x^* - x^*(0)\|$, for any $0 \le t \le T$

$$x^*(t) = x^*(0) + \int_0^t F(x^*(s), s)\mathrm{d}s.$$

Hence, we have that

$$\begin{aligned}
\|x^*(t) - x^*(0)\| &\le \left\|\int_0^T F(x^*(0), s)\mathrm{d}s\right\| + \left\|\int_0^t \left(F(x^*(s), s) - F(x^*(0), s)\right)\mathrm{d}s\right\| \\
&\le \left\|\int_0^T F(x^*(0), s)\mathrm{d}s\right\| + L\int_0^t \|x^*(s) - x^*(0)\|\mathrm{d}s.
\end{aligned}$$

Solving this integral inequality (see Lemma 4.6.3), we have that

$$\|x^*(t) - x^*(0)\| \le e^{Lt}\left\|\int_0^T F(x^*(0), s)\mathrm{d}s\right\|.$$

Now, we use $LT \le \frac{1}{2}$ and get

$$\|x^*(t) - x^*(0)\| \le 2\left\|\int_0^T F(x^*(0), s)\mathrm{d}s\right\|.$$

Picking $N = \left\lceil \log_2 \left(\frac{T}{\epsilon} \max_{s\in[0,T]} \|F(x^*(0), s)\| \right) \right\rceil$, (4.10) shows that the error is less than $20\gamma_\varphi \epsilon$.

□

4.2.5 A basis for piece-wise polynomials

In this section, we discuss how to construct a bounded basis for low-degree piece-wise polynomials. We are given n intervals $\{[I_{i-1}, I_i]\}_{i=1}^n$ where $I_0 = 0$ and $I_n = T$. In the i^{th} interval $[I_{i-1}, I_i]$, we represent the function by a degree D_i polynomial. Formally, we define the function subspace by

$$\mathcal{V} \stackrel{\text{def}}{=} \bigoplus_{i=1}^n \mathcal{V}_i \quad \text{with} \quad \mathcal{V}_i \stackrel{\text{def}}{=} \left\{ \left(\sum_{j=0}^{D_i} \alpha_j t^j \right) \cdot 1_{[I_{i-1}, I_i]} : \alpha_j \in \mathbb{R} \right\}. \tag{4.11}$$

The following Lemma shows we can construct the basis for $\mathcal{V}$ by concatenating the basis for $\mathcal{V}_i$.

Lemma 4.2.7. *For $i \in [n]$, we are given a γ_i bounded basis $\{\varphi_{j,i}\}_{j=0}^{D_i}$ for the subspace $\mathcal{V}_i \subset \mathcal{C}([I_{i-1}, I_i], \mathbb{R})$ on nodes point $\{c_{j,i}\}_{j=0}^{D_i}$. Then, $\{\varphi_{j,i}\}_{i,j}$ is a $\sum_{i=1}^n \gamma_i (I_i - I_{i-1})$ bounded basis for the subspace $\bigoplus_{i=1}^n \mathcal{V}_i \subset \mathcal{C}([I_0, I_n], \mathbb{R})$.*

Proof. For any $t \geq 0$, we have

$$\begin{aligned} \sum_{i=1}^n \sum_{j=0}^{D_i} \left| \int_{I_0}^t \varphi_{i,j}(s) \mathrm{d}s \right| &\leq \sum_{i=1}^n \left(\sum_{j=0}^{D_i} \left| \int_{I_{i-1}}^t \varphi_{i,j}(s) \mathrm{d}s \right| 1_{t \geq I_{i-1}} \right) \\ &\leq \sum_{i=1}^n \gamma_i (I_i - I_{i-1}) \end{aligned}$$

where we used that $\varphi_{i,j}$ is supported on $[I_{i-1}, I_i]$ in the first inequality. □

Next, we note that the boundedness for basis is shift and scale invariant. Hence, we will focus on obtaining a basis for $(t-1)$-degree polynomial on $[-1,1]$ for notation convenience.

For $[-1,1]$, we choose the node points $c_j = \cos(\frac{2j-1}{2t}\pi)$ and the basis are

$$\varphi_j(x) = \frac{\sqrt{1-c_j^2}\cos(t\cos^{-1}x)}{t(x-c_j)}.$$

It is easy to see that $\varphi_j(c_i) = \delta_{i,j}$. To bound the integral, Lemma 91 in [LV17] shows that

$$\left|\int_{-1}^{y}\varphi_j(x)dx\right| \le \frac{2000}{t} \text{ for all } y \in [-1,1].$$

Summing it over t basis functions, we have that $\gamma_\varphi \le 2000$. Together with Lemma 4.2.7, we have the following result:

Lemma 4.2.8. *Let $\mathcal{V}$ be a subspace of piecewise polynomials on $[0,T]$ with fixed nodes. Then, there is a 2000 bounded basis $\{\varphi\}$ for $\mathcal{V}$. Furthermore, for any vector v, it takes $O(\sum_{i=1}^{n}(1+D_i)^2)$ time to compute $v^\top A_\varphi$ where D_i is the maximum degree of the i-th piece.*

Alternatively, one can find u such that $\|u - v^\top A_\varphi\| \le \epsilon T\|v\|_\infty$ in time

$$O\left(\operatorname{rank}(\mathcal{V})\log(\frac{\operatorname{rank}(\mathcal{V})}{\epsilon})\right).$$

Proof. The bound follows from previous discussion. For the computation cost, note that

$$(v^\top A_\varphi)_{(i,j)} = \sum_{i',j'}\int_0^{c_{(i,j)}} v_{(i',j')}\varphi_{(i',j')}(s)\mathrm{d}s.$$

where (i,j) is the j-th node at the i-th piece. For any $i \neq i'$, the support of $\varphi_{(i',j')}(s)$ is either disjoint from $[0,c_{(i,j)}]$ (if $i' > i$) or included in $[0,c_{(i,j)}]$ (if $i' < i$). Hence, we have that

$$\int_0^{c_{(i,j)}}\varphi_{(i',j')}(s)\mathrm{d}s = \begin{cases} 0 & \text{if } i' > i \\ \int_{-\infty}^{\infty}\varphi_{(i',j')}(s)\mathrm{d}s & \text{if } i' < i \end{cases}.$$

Therefore, we have

$$(v^\top A_\varphi)_{(i,j)} = \sum_{i'<i,j'} v_{(i',j')} \cdot \int_{-\infty}^{\infty} \varphi_{(i',j')}(s)\mathrm{d}s + \sum_{j'} v_{(i,j')} \cdot \int_{0}^{c_{(i,j)}} \varphi_{(i,j')}(s)\mathrm{d}s.$$

Note that $\int_0^{c_{(i,j)}} \varphi_{(i',j')}(s)\mathrm{d}s$ can precomputed. Since there are $\sum_{i=1}^n (1+D_i)$ many pairs of (i,j), the first term can be computed in $\sum_{i=1}^n (1+D_i)$ time. Since there are $\sum_{i=1}^n (1+D_i)^2$ many pairs of (i,j,j'), the second term can be computed in $\sum_{i=1}^n (1+D_i)^2$ time.

Theorem 4.2.9 gives another way to compute the integration and its runtime is

$$O\left(\mathrm{rank}(\mathcal{V}) \log(\frac{\mathrm{rank}(\mathcal{V})}{\epsilon})\right).$$

□

Remark 4.2.2. Experiment seems to suggest the basis we proposed is 1 bounded.

Here is the theorem we used above to compute the Lagrange polynomials.

Theorem 4.2.9 ([DGR96, Section 5])**.** *Let ϕ_i be the Lagrange basis polynomials on the Chebyshev nodes $c_j = \cos(\frac{2j-1}{2t}\pi)$ for $j \in [t]$, namely, $\phi_i(s) = \prod_{j\neq i} \frac{s-c_j}{c_i-c_j}$. Given a polynomial $p(s) = \sum_{j=1}^t \alpha_j \phi_j(s)$ represented by $\{\alpha_j\}_{j=1}^t$, one can compute $\{\ell_i\}_{i=1}^t$ such that*

$$\left|\ell_i - \int_0^{c_i} p(s)ds\right| \leq \epsilon \|\alpha\|_\infty$$

in time $O(t\log(\frac{t}{\epsilon}))$.

4.3 Improved Contraction Bound for HMC

In this section, we give an improved contraction bound for HMC (Algorithm 4.1). Each iteration of Algorithm 4.1 solve the HMC dynamics approximately. In the later sections, we will discuss how to solve this ODE.

To give a contraction bound for the noisy HMC, we first analyze the contraction of the ideal HMC. We reduce the problem of bounding the contraction rate of the ideal HMC to the following lemma involving a matrix ODE.

Lemma 4.3.1. *Given a symmetric matrix $H(t)$ such that $0 \prec m_2 \cdot I \preceq H(t) \preceq M_2 \cdot I$ for all $t \geq 0$. Consider the ODE*

$$u''(t) = -H(t) \cdot u(t),$$
$$u'(0) = 0.$$

Let $\alpha(t) = \frac{1}{\|u(0)\|_2} \int_0^t (t-s) \cdot \|H(s)u(0)\|_2 \mathrm{d}s$. For any $0 \leq T \leq \frac{1}{2\sqrt{M_2}}$ such that $\alpha(T) \leq \frac{1}{8}\sqrt{\frac{m_2}{M_2}}$, we have that

$$\|u(T)\|_2^2 \leq \left(1 - \max\left(\frac{1}{4}m_2 T^2, \frac{1}{2}\sqrt{\frac{m_2}{M_2}} \cdot \alpha(T)\right)\right) \cdot \|u(0)\|_2^2.$$

Using this lemma, we prove both parts of the main contraction bound, Lemma 4.1.6.

Proof of Lemma 4.1.6. Let error function $e(t) = y(t) - x(t)$. The definition of HMC shows that

$$e''(t) = -(\nabla f(y(t)) - \nabla f(x(t))) = -H(t) \cdot e(t)$$

where $H(t) = \int_0^1 \nabla^2 f(x(t) + s(y(t) - x(t)))\mathrm{d}s$. By the strong convexity and the Lipschitz gradient of f, we have that

$$m_2 \cdot I \preceq H(t) \preceq M_2 \cdot I.$$

Hence, we can apply Lemma 4.3.1.

To get the first bound, we bound the $\alpha(t)$ defined in Lemma 4.3.1 as follows

$$\alpha(t) = \frac{1}{\|e(0)\|_2} \int_0^t (t-s) \cdot \|H(s)e(0)\|_2 \mathrm{d}s \leq M_2 \int_0^t (t-s)\mathrm{d}s = M_2 \frac{t^2}{2}.$$

Therefore, for $0 \leq t \leq \frac{m_2^{1/4}}{2M_2^{3/4}}$, we have that $\alpha(t) \leq \frac{1}{8}\sqrt{\frac{m_2}{M_2}}$ and hence Lemma 4.3.1 gives the first bound.

To get the second bound, we note that $\alpha(t)$ is increasing and hence there is t such that $\alpha(t) = \frac{1}{8}\sqrt{\frac{m_2}{M_2}}$. Using such t in Lemma 4.3.1 gives the second bound. □

Now, we prove the main technical lemma of this section, a contraction estimate for matrix ODE. We note that not all matrix ODEs come from some HMC and hence it might be possible to get a better bound by directly analyzing the HMC.

Proof of Lemma 4.3.1. Let e_1 denote the basis vector that it is 1 in the first coordinate and 0 everywhere else.

Without loss of generality, we can assume $\|u(0)\|_2 = 1$ and $u(0) = e_1$.

The proof involves first getting a crude bound on $\|u(t)\|_2$. Then, we boast the bound by splitting the movement of $u(t)$ into one parallel to e_1 and one orthogonal to e_1.

Crude bound on $\|u(t)\|_2$:

Integrating both sides of $u''(t) = -H(t) \cdot u(t)$ twices and using $u'(0) = 0$ gives

$$u(t) = u(0) - \int_0^t (t-s)H(s)u(s)\mathrm{d}s. \tag{4.12}$$

We take the norm on both sides and use $0 \preceq H(s) \preceq M_2 \cdot I$ to get

$$\|u(t)\|_2 \leq 1 + M_2 \cdot \int_0^t (t-s)\|u(s)\|_2 \mathrm{d}s.$$

Applying Lemma 4.6.2 to this equation and using $t \leq \frac{1}{2\sqrt{M_2}}$ gives

$$\|u(t)\|_2 \leq \cosh(\sqrt{M_2}t) \leq \frac{6}{5}$$

Putting it back to (4.12) gives

$$\|u(t) - e_1\|_2 \leq \int_0^t (t-s)\|H(s) \cdot u(s)\|_2 \mathrm{d}s = \int_0^t (t-s) \cdot M_2 \cdot \frac{6}{5} \mathrm{d}s = \frac{6}{10} M_2 t^2.$$

In particular, for any $0 \leq t \leq \frac{1}{2\sqrt{M_2}}$, we have that

$$\frac{5}{6} \leq u_1(t) \leq \frac{7}{6}. \tag{4.13}$$

Improved bound on $\|u(t)\|_2$:

Let P_1 be the orthogonal projection to the first coordinate and $P_{-1} = I - P_1$. We write $u(t) = u_1(t) + u_{-1}(t)$ with $u_1(t) = P_1 u(t)$ and $u_{-1}(t) = P_{-1} u(t)$, namely, $u_1(t)$ is parallel to e_1 and $u_{-1}(t)$ is orthogonal to e_1.

Fix any $0 \leq t \leq T$. Let $\beta(t) = e_1^\top u(t)$. By the definition of u, we have

$$u''(t) = -\beta(t) \cdot H(t) e_1 - H(t) u_{-1}(t). \tag{4.14}$$

Integrating both sides twice and using $u_{-1}(0) = 0$, we have

$$\begin{aligned} u_{-1}(t) &= \int_0^t (t-s) P_{-1} u''(s) \mathrm{d}s \\ &= -\int_0^t (t-s) \cdot \beta(s) \cdot P_{-1} H(s) e_1 \mathrm{d}s - \int_0^t (t-s) P_{-1} H(s) u_{-1}(s) \mathrm{d}s. \end{aligned}$$

Taking norm on both sides and using that $0 \preceq H(t) \preceq M_2 \cdot I$ and $\frac{5}{6} \leq \beta(s) \leq \frac{7}{6}$, we have that

$$\begin{aligned}
\|u_{-1}(t)\|_2 &\leq \int_0^t (t-s) \cdot \beta(s) \cdot \|H(s)e_1\|_2 \mathrm{d}s + M_2 \cdot \int_0^t (t-s) \cdot \|u_{-1}(s)\|_2 \mathrm{d}s \\
&\leq \frac{7}{6} \int_0^t (t-s) \cdot \|H(s)e_1\|_2 \mathrm{d}s + M_2 \cdot \int_0^t (t-s) \cdot \|u_{-1}(s)\|_2 \mathrm{d}s \\
&\leq \frac{7}{6}\alpha(T) + M_2 \cdot \int_0^t (t-s) \cdot \|u_{-1}(s)\|_2 \mathrm{d}s \qquad (4.15)
\end{aligned}$$

where $\alpha(T) \stackrel{\text{def}}{=} \int_0^T (T-s) \cdot \|H(s)e_1\|_2 \mathrm{d}s$. Solving this integral inequality (Lemma 4.6.2), we get

$$\|u_{-1}(t)\|_2 \leq \frac{7}{6}\alpha(T) \cdot \cosh(\sqrt{M_2}t) \leq \frac{7}{6}\alpha(T) \cdot \cosh(1/2) \leq \frac{4}{3}\alpha(T) \qquad (4.16)$$

where the second step follows from $t \leq \frac{1}{2\sqrt{M_2}}$, and the last step follows from $\cosh(1/2) \leq \frac{8}{7}$.

Next, we look at the first coordinate of (4.14) and get

$$\begin{aligned}
\beta''(t) &= -\beta(t) \cdot e_1^\top H(t) e_1 - e_1^\top H(t) u_{-1}(t) \\
&\leq -\frac{5}{6} e_1^\top H(t) e_1 + \|H(t)e_1\|_2 \cdot \|u_{-1}(t)\|_2. \qquad (4.17)
\end{aligned}$$

To bound the last term, we note that

$$\begin{aligned}
\|H(t)e_1\|_2 &= \sqrt{e_1^\top H^2(t) e_1} \\
&\leq \sqrt{M_2 \cdot e_1^\top H(t) e_1} && \text{by } H^2(t) \preceq M_2 \cdot H(t) \\
&= \sqrt{\frac{M_2}{m_2}} \sqrt{\frac{m_2}{e_1^\top H(t) e_1}} e_1^\top H(t) e_1 \\
&\leq \sqrt{\frac{M_2}{m_2}} \cdot e_1^\top H(t) e_1 && \text{by } m_2 \leq e_1^\top H(t) e_1. \qquad (4.18)
\end{aligned}$$

Using this into (4.17) and $\alpha(T) \leq \frac{1}{8}\sqrt{\frac{m_2}{M_2}}$, we have

$$\beta''(t) \leq -e_1^\top H(t) e_1 \cdot \left(\frac{5}{6} - \frac{4}{3}\sqrt{\frac{M_2}{m_2}}\alpha(T) \right) \leq -\frac{2}{3} e_1^\top H(t) e_1.$$

Hence, we have that

$$\beta(t) \leq 1 - \frac{2}{3}\int_0^t (t-s)\cdot e_1^\top H(s)e_1 \mathrm{d}s. \tag{4.19}$$

Using (4.19), $\beta(t) \geq \frac{5}{6}$ and (4.16) gives

$$\begin{aligned}
\|u(t)\|_2^2 &= \beta^2(t) + \|u_{-1}(t)\|_2^2 \\
&\leq 1 - \int_0^t (t-s)\cdot e_1^\top H(s)e_1 \mathrm{d}s + \left(\frac{4}{3}\int_0^T (T-s)\cdot \|H(s)e_1\|_2 \mathrm{d}s\right)^2 \\
&\leq 1 - \int_0^t (t-s)\cdot e_1^\top H(s)e_1 \mathrm{d}s + 2\sqrt{\frac{M_2}{m_2}}\int_0^T (T-s)\cdot e_1^\top H(s)e_1 \mathrm{d}s \cdot \alpha(T) \\
&= 1 - \int_0^t (t-s)\cdot e_1^\top H(s)e_1 \mathrm{d}s \left(1 - 2\sqrt{\frac{M_2}{m_2}}\alpha(T)\right) \\
&\leq 1 - \frac{1}{2}\int_0^t (t-s)\cdot e_1^\top H(s)e_1 \mathrm{d}s
\end{aligned}$$

where we used (4.18) at the second inequality and $\alpha(T) \leq \frac{1}{8}\sqrt{\frac{m_2}{M_2}}$ at the end.

Finally, we bound the last term in two way. One way simply uses $e_1^\top H(s)e_1 \geq m_2$ and get

$$\int_0^t (t-s)\cdot e_1^\top H(s)e_1 \mathrm{d}s \geq \frac{m_2}{2}t^2$$

which implies

$$\|u(t)\|_2^2 \leq 1 - \frac{m_2}{4}t^2.$$

For the other way, we apply (4.18) to get that

$$\int_0^t (t-s)\cdot e_1^\top H(s)e_1 \mathrm{d}s \geq \int_0^t (t-s)\cdot \|H(t)e_1\|_2 \sqrt{\frac{m_2}{M_2}}\mathrm{d}s = \sqrt{\frac{m_2}{M_2}}\cdot\alpha(T).$$

where the last step follows by the definition of $\alpha(T)$. Thus, we have

$$\|u(t)\|_2^2 \leq 1 - \frac{1}{2}\sqrt{\frac{m_2}{M_2}}\alpha(T).$$

□

Finally, we analyze the contraction of the noisy HMC.

Theorem 4.3.2 (Contraction of noisy HMC)**.** *Suppose f is m_2 strongly convexity with M_2 Lipschitz gradient. For any step-size $h \leq \frac{m_2^{1/4}}{2M_2^{3/4}}$, let $X \sim \mathrm{HMC}(x^{(0)}, f, \epsilon, h)$ and $Y \sim e^{-f}$. Then, we have that*

$$W_2(X, Y) \leq \frac{\epsilon}{\sqrt{m_2}}.$$

In addition, the number of iterations is

$$N = O\left(\frac{1}{m_2 h^2}\right) \cdot \log\left(\frac{\|\nabla f(x^{(0)})\|_2^2/m_2 + d}{\epsilon}\right).$$

Proof. To prove the W_2 distance, we let $x^{(k)}$ be the iterates of the algorithm HMC. Let $y^{(k)}$ be the k-th step of the ideal HMC starting from a random point $y^{(0)} \sim e^{-f}$ with the random initial direction identical to the algorithm HMC. Let $x^{*(k)}$ be the 1 step ideal HMC starting from $x^{(k-1)}$ with the same initial direction as $y^{(k)}$. Lemma 4.1.6 shows that

$$\|x^{*(k)} - y^{(k)}\|_2^2 \leq \left(1 - \frac{m_2 h^2}{4}\right)\|x^{(k-1)} - y^{(k-1)}\|_2^2.$$

Let $\theta = \frac{m_2 h^2}{8}$, then $\theta \leq \frac{1/(4\kappa^{3/4})}{8} \leq 1/32$. By the assumption of the noise, we have $\|x^{(k)} -$

$x^{*(k)}\|_2 \leq \frac{\epsilon\theta}{2\sqrt{m_2}}$. Hence, we have

$$\begin{aligned}
\|x^{(k)} - y^{(k)}\|_2^2 &= \|(x^{*(k)} - y^{(k)}) + (x^{(k)} - x^{*(k)})\|_2^2 \\
&\leq (1+\theta)\,\|x^{*(k)} - y^{(k)}\|_2^2 + (1+1/\theta)\|x^{(k)} - x^{*(k)}\|_2^2 \\
&\leq (1+\theta)\,(1-2\theta)\|x^{(k-1)} - y^{(k-1)}\|_2^2 + (1+1/\theta)\|x^{(k)} - x^{*(k)}\|_2^2 \\
&\leq (1-\theta)\,\|x^{(k-1)} - y^{(k-1)}\|_2^2 + (1+1/\theta)\|x^{(k)} - x^{*(k)}\|_2^2 \\
&\leq (1-\theta)\,\|x^{(k-1)} - y^{(k-1)}\|_2^2 + (2/\theta)\cdot\frac{\epsilon^2\theta^2}{4m_2}.
\end{aligned}$$

where the second step follows by $(a+b)^2 \leq (1+\theta)a^2 + (1+1/\theta)b^2$, the third step follows by $\|x^{*(k)} - y^{(k)}\|_2^2 \leq (1-2\theta)\|x^{(k-1)} - y^{(k-1)}\|_2^2$, the fourth step follows by $(1+\theta)(1-2\theta) \leq (1-\theta)$, the fifth step follow by $\theta \leq 1/4$ and $\|x^{(k)} - x^{*(k)}\|_2 \leq \frac{\epsilon\theta}{2\sqrt{m_2}}$.

Applying this bound iteratively gives

$$\|x^{(k)} - y^{(k)}\|_2^2 \leq (1-\theta)^k\,\|x^{(0)} - y^{(0)}\|_2^2 + \frac{\epsilon^2}{2m_2}. \tag{4.20}$$

Let $x^{(\min)}$ be the minimum of f. Then, we have

$$\|x^{(0)} - y^{(0)}\|_2^2 \leq 2\|x^{(0)} - x^{(\min)}\|_2^2 + 2\|y^{(0)} - x^{(\min)}\|_2^2. \tag{4.21}$$

For the first term, the strong convexity of f shows that

$$\|x^{(0)} - x^{(\min)}\|_2^2 \leq \frac{1}{m^2}\|\nabla f(x^{(0)})\|_2^2. \tag{4.22}$$

For the second term, Theorem 1 in [DM16] shows that

$$\mathbb{E}\left[\|y^{(0)} - x^{(\min)}\|_2^2\right] \leq \frac{d}{m_2}. \tag{4.23}$$

Combining (4.20), (4.21), (4.22) and (4.23), we have

$$\mathbb{E}\left[\|x^{(k)} - y^{(k)}\|_2^2\right] \leq (1-\theta)^k\left(\frac{2\|\nabla f(x^{(0)})\|_2^2}{m_2^2} + \frac{2d}{m_2}\right) + \frac{\epsilon^2}{2m_2}.$$

Picking

$$k = \frac{1}{\theta} \cdot \log\left(\frac{\frac{2\|\nabla f(x^{(0)})\|_2^2}{m_2^2} + \frac{2d}{m_2}}{\frac{\epsilon^2}{2m_2}}\right) = \frac{1}{\theta} \cdot \log\left(\frac{4}{\epsilon^2}\left(\frac{\|\nabla f(x^{(0)})\|_2^2}{m_2} + d\right)\right),$$

we have that

$$\mathbb{E}\left[\|x^{(k)} - y^{(k)}\|_2^2\right] \leq \frac{\epsilon^2}{m_2}.$$

This proves that $W_2(X, Y) \leq \frac{\epsilon}{\sqrt{m_2}}$. □

4.4 Strongly Convex functions with Lipschitz Gradient

In this section, we give a faster sampling algorithm for strongly convex functions with Lipschitz gradient. The purpose of this section is to illustrate that our contraction bound and ODE theorem are useful even for functions that are not infinitely differentiable. We believe that our bound can be beaten by algorithms designed for this specific setting.

4.4.1 Bounding the ODE solution

First, we prove that the HMC dynamic for these functions can be well approximated by piece-wise degree-2 polynomials. Note that this only requires that the Hessian has bounded eigenvalues.

Lemma 4.4.1 (Smoothness implies the existence of degree-2 polynomial approximation)**.** *Let f be a twice-differentiable function such that $-M_2 \cdot I \preceq \nabla^2 f(x) \preceq M_2 \cdot I$ for all $x \in \mathbb{R}^d$. Let $0 \leq h \leq \frac{1}{2\sqrt{M_2}}$. Consider the HMC dynamic*

$$\begin{aligned} \frac{\mathrm{d}^2 x}{\mathrm{d}t^2}(t) &= -\nabla f(x(t)) \text{ for } 0 \leq t \leq h, \\ \frac{\mathrm{d}x}{\mathrm{d}t}(0) &= v_1, \\ x(0) &= v_0. \end{aligned}$$

For any integer k, there is a continuously differentiable k-piece degree 2 polynomial q such that $q(0) = v_0$, $\frac{\mathrm{d}q}{\mathrm{d}t}(0) = v_1$ and $\left\| \frac{\mathrm{d}^2 q}{\mathrm{d}t^2}(t) - \frac{\mathrm{d}^2 x}{\mathrm{d}t^2}(t) \right\|_2 \leq \frac{\epsilon}{h^2}$ for $0 \leq t \leq h$ with

$$\epsilon = \frac{2M_2 h^3}{k} \left(\|v_1\|_2 + \|\nabla f(v_0)\|_2 \cdot h \right).$$

Proof. Let $x(t)$ be the solution of the ODE. We define a continuously differentiable k-piece

degree-2 polynomial q by

$$q(0) = v_0 \text{ and } q'(t) = \frac{\mathrm{d}x(t_{\text{pre}})}{\mathrm{d}t} \cdot \frac{t_{\text{next}} - t}{t_{\text{next}} - t_{\text{pre}}} + \frac{\mathrm{d}x(t_{\text{next}})}{\mathrm{d}t} \cdot \frac{t_{\text{pre}} - t}{t_{\text{next}} - t_{\text{pre}}}$$

with $t_{\text{pre}} = \left\lfloor \frac{t}{h/k} \right\rfloor \cdot \frac{h}{k}$ and $t_{\text{next}} = t_{\text{pre}} + \frac{h}{k}$. Clearly, we have that $q(0) = x(0) = v_0$ and $\frac{\mathrm{d}q}{\mathrm{d}t}(0) = \frac{\mathrm{d}x}{\mathrm{d}t}(0) = v_1$. Also, we have that

$$\left\| \frac{\mathrm{d}^2 q}{\mathrm{d}t^2}(t) - \frac{\mathrm{d}^2 x}{\mathrm{d}t^2}(t) \right\|_2 = \left\| \int_{t_{\text{pre}}}^{t_{\text{next}}} \frac{\mathrm{d}^2 x}{\mathrm{d}t^2}(s) ds - \frac{\mathrm{d}^2 x}{\mathrm{d}t^2}(t) \right\|_2 \leq \frac{h}{k} \max_{0 \leq t \leq h} \left\| \frac{\mathrm{d}^3 x}{\mathrm{d}t^3}(t) \right\|_2 \leq \frac{M_2 h}{k} \max_{0 \leq t \leq h} \left\| \frac{\mathrm{d}x}{\mathrm{d}t}(t) \right\| \tag{4.24}$$

where we used that $\frac{\mathrm{d}^3 x}{\mathrm{d}t^3}(t) = -\nabla^2 f(x(t)) \frac{\mathrm{d}x}{\mathrm{d}t}(t)$ at the end.

Therefore, it suffices to bound the term $\max_{0 \leq t \leq h} \left\| \frac{\mathrm{d}x}{\mathrm{d}t}(t) \right\|$. Using again that $\frac{\mathrm{d}^3 x}{\mathrm{d}t^2}(t) = -\nabla^2 f(x(t)) \frac{\mathrm{d}x}{\mathrm{d}t}(t)$, we have that

$$\begin{aligned} \frac{\mathrm{d}x}{\mathrm{d}t}(t) &= \frac{\mathrm{d}x}{\mathrm{d}t}(0) + \frac{\mathrm{d}^2 x}{\mathrm{d}t^2}(0) \cdot t + \int_0^t (t - s) \cdot \frac{\mathrm{d}^3 x}{\mathrm{d}t^3}(s) ds \\ &= v_1 - \nabla f(v_0) \cdot t - \int_0^t (t - s) \cdot \nabla^2 f(x(s)) \frac{\mathrm{d}x}{\mathrm{d}t}(s) ds. \end{aligned}$$

Hence, for $0 \leq t \leq h$, we have that

$$\left\| \frac{\mathrm{d}x}{\mathrm{d}t}(t) \right\|_2 \leq \|v_1\|_2 + \|\nabla f(v_0)\|_2 \cdot h + M_2 \cdot \int_0^t (t - s) \left\| \frac{\mathrm{d}x}{\mathrm{d}t}(t) \right\|_2 ds.$$

Solving this integral inequality, Lemma 4.6.2 shows that

$$\max_{0 \leq t \leq h} \left\| \frac{\mathrm{d}x}{\mathrm{d}t}(t) \right\| \leq (\|v_1\|_2 + \|\nabla f(v_0)\|_2 \cdot h) \cdot \cosh(\sqrt{M_2} \cdot h) \leq 2 (\|v_1\|_2 + \|\nabla f(v_0)\|_2 \cdot h)$$

where we used that $h \leq \frac{1}{2\sqrt{M_2}}$. Applying this in (4.24) gives this result. □

The precise runtime depends on how accurate we need to solve the ODE, namely, the parameter ϵ in Lemma 4.4.1. The term $\|v_1\|_2$ can be upper bounded by $O(\sqrt{d})$ with

high probability since v_1 is sampled from normal distribution. The term $\|\nabla f(v_0)\|_2$ is much harder to bound. Even for a random v_0 e^{-f}, the worst case bound we can give is

$$\|\nabla f(v_0)\|_2 = \|\nabla f(v_0) - \nabla f(x^*)\|_2 \leq M_2\|v_0 - x^*\|_2 \lesssim \sqrt{\kappa M_2 d}$$

where we used that $\|v_0 - x^*\|_2 \lesssim \sqrt{d/m_2}$ for random v_0 e^{-f} [DM16]. This is not enough for improving existing algorithms, as we would need $\sqrt{\kappa d}$ time per iteration. The crux of this section is to show that $\|\nabla f(v_0)\|_2 = O(\sqrt{M_2 d})$ for most of the iterations in the HMC walk if the process starts at the minimum of f. This is tight for quadratic f.

Lemma 4.4.2 (Smoothness implies expected gradient is upper bounded)**.** *Let f be a function such that $-M_2 \cdot I \preceq \nabla^2 f(x) \preceq M_2 \cdot I$ for all $x \in \mathbb{R}^d$. Let $x^{(k)}$ be the starting point of the k^{th} step in* $\mathrm{HMC}(x^{(0)}, f, \epsilon, h)$ *(Algorithm 4.1) with step size $h \leq \frac{1}{8\sqrt{M_2}}$. Then, we have that*

$$\frac{1}{N}\mathbb{E}\left[\sum_{k=0}^{N-1}\|\nabla f(x^{(k)})\|_2^2\right] \leq O\left(\frac{f(x^{(0)}) - \min_x f(x)}{h^2 N} + M_2 d + \frac{\bar{\epsilon}^2}{h^4}\right)$$

$\bar{\epsilon}$ is the error in solving the HMC defined in Algorithm 4.1.

Proof. Consider one step of the HMC dynamic. Note that

$$\frac{\mathrm{d}}{\mathrm{d}t} f(x(t)) = \nabla f(x(t))^\top \frac{\mathrm{d}x}{\mathrm{d}t}.$$

Hence, we have

$$\begin{aligned}
\frac{\mathrm{d}^2}{\mathrm{d}t^2} f(x(t)) &= \frac{\mathrm{d}x}{\mathrm{d}t}^\top \nabla^2 f(x(t)) \frac{\mathrm{d}x}{\mathrm{d}t} + \nabla f(x(t))^\top \frac{\mathrm{d}^2 x}{\mathrm{d}t^2} \\
&= \frac{\mathrm{d}x}{\mathrm{d}t}^\top \nabla^2 f(x(t)) \frac{\mathrm{d}x}{\mathrm{d}t} - \|\nabla f(x(t))\|^2 && \text{by } \frac{\mathrm{d}^2 x}{\mathrm{d}t^2} = -\nabla f(x(t)) \\
&\leq M_2 \cdot \|\frac{\mathrm{d}x}{\mathrm{d}t}\|_2^2 - \|\nabla f(x(t))\|^2. && \text{by } \nabla^2 f(x(t)) \preceq M_2 \cdot I \qquad (4.25)
\end{aligned}$$

In Lemma 4.4.1, we proved that $\|\frac{\mathrm{d}x}{\mathrm{d}t}\|_2 \leq 2\left(\|\frac{\mathrm{d}x}{\mathrm{d}t}(0)\|_2 + \|\nabla f(x(0))\|_2 \cdot h\right)$ for all $0 \leq t \leq h$. Using this, we have that

$$\begin{aligned}\|\nabla f(x(t)) - \nabla f(x(0))\| &\leq M_2\|x(t) - x(0)\|_2 \\ &\leq M_2 \int_0^t \|\frac{\mathrm{d}x}{\mathrm{d}t}(t)\|_2 \mathrm{d}t \\ &\leq 2M_2 h \cdot \left(\|\frac{\mathrm{d}x}{\mathrm{d}t}(0)\|_2 + \|\nabla f(x(0))\|_2 \cdot h\right)\end{aligned}$$

which implies

$$\|\nabla f(x(t))\| = \|\nabla f(x(0))\| \pm 2M_2 h \left(\|\frac{\mathrm{d}x}{\mathrm{d}t}(0)\| + \|\nabla f(x(0))\| \cdot h\right). \tag{4.26}$$

Using our choice of h, we have that $\|\nabla f(x(t))\| \geq \frac{1}{2}\|\nabla f(x(0))\| - 2M_2 h \cdot \|\frac{\mathrm{d}x}{\mathrm{d}t}(0)\|_2$. Putting these estimates into (4.25) gives

$$\begin{aligned}&\frac{\mathrm{d}^2}{\mathrm{d}t^2} f(x(t)) \\ &\leq 2M_2 \left(\|\frac{\mathrm{d}x}{\mathrm{d}t}(0)\|^2 + \|\nabla f(x(0))\|^2 h^2\right) - \frac{1}{4}\|\nabla f(x(0))\|^2 + 2M_2 h \cdot \|\nabla f(x(0))\| \cdot \|\frac{\mathrm{d}x}{\mathrm{d}t}(0)\| \\ &\leq 2M_2 \left(\|\frac{\mathrm{d}x}{\mathrm{d}t}(0)\|^2 + \|\nabla f(x(0))\|^2 h^2\right) - \frac{1}{4}\|\nabla f(x(0))\|^2 + 2 \cdot (\frac{1}{8}\|\nabla f(x(0))\|)^2 + 2 \cdot (16M_2 h\|\frac{\mathrm{d}x}{\mathrm{d}t}(0)\|)^2 \\ &= (2M_2 + 512M_2^2 h^2)\|\frac{\mathrm{d}x}{\mathrm{d}t}(0)\|^2 - (\frac{1}{4} - 2M_2 h^2 - \frac{1}{32})\|\nabla f(x(0))\|^2 \\ &\leq 10M_2 \cdot \|\frac{\mathrm{d}x}{\mathrm{d}t}(0)\|^2 - \frac{1}{8}\|\nabla f(x(0))\|^2\end{aligned}$$

where we used that $h \leq \frac{1}{8\sqrt{M_2}}$ at the last two equations.

Since $\frac{\mathrm{d}x}{\mathrm{d}t}(0)$ is sampled from normal distribution, we have that

$$\begin{aligned}\mathbb{E}[f(x(h))] &\leq f(x(0)) + \int_0^h (h-t)\frac{\mathrm{d}^2}{\mathrm{d}t^2} f(x(t))\mathrm{d}t \\ &= f(x(0)) + 5M_2 \cdot h^2 \cdot \mathbb{E}\left[\|\frac{\mathrm{d}x}{\mathrm{d}t}x(0)\|^2\right] - \frac{h^2}{16}\|\nabla f(x(0))\|^2 \\ &= f(x(0)) + 5M_2 \cdot h^2 \cdot d - \frac{h^2}{16}\|\nabla f(x(0))\|^2\end{aligned}$$

where the last step follows from $\mathbb{E}[\|\frac{\mathrm{d}x}{\mathrm{d}t}(0)\|^2] = d$.

For the ODE starting from $x^{(k)}$, we have that

$$\mathbb{E}[f(x(h))] \le f(x^{(k)}) - \frac{h^2}{16}\|\nabla f(x^{(k)})\|^2 + O(M_2 \cdot d \cdot h^2).$$

To compare $f(x(h))$ and $f(x^{(k+1)})$, we note that the distance between $x^{(k+1)}$ and $x(h)$ is less than $\bar{\epsilon}$ in ℓ_2 norm. Using (4.26) and the fact $\nabla^2 f \preceq M_2 I$, the function value changed by at most

$$\bar{\epsilon} \cdot \|\nabla f(x^{(k+1)})\| + M_2 \cdot \bar{\epsilon}^2 = 2\bar{\epsilon} \cdot \|\nabla f(x^{(k)})\| + O(\bar{\epsilon} M_2 \sqrt{d} h + \bar{\epsilon}^2 M_2)$$

with high probability. Hence, we have

$$\begin{aligned}
\mathbb{E}[f(x^{(k+1)})] &\le f(x^{(k)}) + 2\bar{\epsilon}\|\nabla f(x^{(k)})\| - \frac{h^2}{16}\|\nabla f(x^{(k)})\|^2 + O(\bar{\epsilon} M_2 \sqrt{d} h + \bar{\epsilon}^2 M_2 + M_2 d h^2) \\
&\le f(x^{(k)}) + 2\bar{\epsilon}\|\nabla f(x^{(k)})\| - \frac{h^2}{16}\|\nabla f(x^{(k)})\|^2 + O(\bar{\epsilon}^2 M_2 + M_2 d h^2) \\
&\le f(x^{(k)}) + 2\bar{\epsilon}\|\nabla f(x^{(k)})\| - \frac{h^2}{16}\|\nabla f(x^{(k)})\|^2 + O(\frac{\bar{\epsilon}^2}{h^2} + M_2 d h^2) \\
&\le f(x^{(k)}) - \frac{h^2}{32}\|\nabla f(x^{(k)})\|^2 + O(\frac{\bar{\epsilon}^2}{h^2} + M_2 d h^2)
\end{aligned}$$

where the step follows from $2ab \le a^2 + b^2$, the third step follows from our choice of h, the second last step follows by $2\bar{\epsilon}\|\nabla f(x^{(k)})\| \le \frac{h^2}{64}\|\nabla f(x^{(k)})\|^2 + 64\frac{\bar{\epsilon}^2}{h^2}$.

Summing k from 0 to $N-1$, we have

$$\sum_{k=0}^{N-1} \mathbb{E}\left[f(x^{(k+1)}) - f(x^{(k)}) + \frac{h^2}{32}\|\nabla f(x^{(k)})\|^2\right] \le N \cdot O\left(\frac{\bar{\epsilon}^2}{h^2} + M_2 d h^2\right)$$

Using $f(x^{(N)}) \ge \min_x f(x)$ and reorganizing the terms gives the desired result.

□

Remark 4.4.1. Both Lemma 4.4.1 and Lemma 4.4.2 do not need convexity.

4.4.2 Sampling

Now, we can apply our algorithm for second-order ODEs to the HMC dynamic. To control the gradient of the initial point in a simple way, we start at the algorithm at a local minimum of f.

Theorem 4.1.4. (Strongly Convex)**.** Given a function f such that $0 \prec m_2 \cdot I \preceq \nabla^2 f(x) \preceq M_2 \cdot I$ for all $x \in \mathbb{R}^d$ and $0 < \epsilon < \sqrt{d}$. Starting $x^{(0)}$ at the minimum of f, we can find a random point X whose Wasserstein distance to Y drawn from the density proportional to e^{-f} satisfies

$$W_2(X, Y) \leq \frac{\epsilon}{\sqrt{m_2}}$$

using $O(\kappa^{1.5} \log(\frac{d}{\epsilon}))$ iterations where $\kappa = \frac{M_2}{m_2}$. Each iteration takes $O\left(\frac{\kappa^{\frac{1}{4}} d^{\frac{3}{2}}}{\epsilon} \log\left(\frac{\kappa d}{\epsilon}\right)\right)$ time and $O\left(\frac{\kappa^{\frac{1}{4}} d^{\frac{1}{2}}}{\epsilon} \log\left(\frac{\kappa d}{\epsilon}\right)\right)$ evaluations of ∇f, amortized over all iterations.

Proof. The number of iterations follows from Theorem 4.3.2 with

$$h = \frac{m_2^{1/4}}{16000 M_2^{3/4}}. \tag{4.27}$$

To approximate the HMC dynamic, we apply the ODE algorithm (Theorem 4.2.4). Now, we estimate the parameters in Theorem 4.2.4. Note that $L_1 = 0$, $L_2 = M_2$, $L = \sqrt{M_2}$, $T = h$, $LT \leq 1/16000$, $\epsilon_{\text{ODE}} = \frac{\epsilon \cdot \sqrt{m_2} h^2}{16}$. Hence, if the solution is approximated by a k-piece degree 2 polynomial, we can find it in $O(dk) \cdot \log(\frac{C}{\epsilon_{\text{ODE}}})$ time and $O(k) \cdot \log(\frac{C}{\epsilon_{\text{ODE}}})$ evaluations where

$$C = O(h^2) \|\nabla f(v_0)\| + h \|v_1\|$$

with v_0 and v_1 are the initial point and initial velocity of the dynamic.

Finally, Lemma 4.4.1 shows that the ODE can be approximated using a

$$k \stackrel{\text{def}}{=} \frac{2 \cdot M_2 h^3 \left(\|v_1\|_2 + \|\nabla f(v_0)\|_2 \cdot h \right)}{\epsilon / (\sqrt{m_2} \cdot \kappa^{3/2})}$$

piece degree 2 polynomials where v_0 is the initial point and v_1 is the initial velocity. Since v_1 is sampled from normal distribution, we have that

$$k \lesssim \frac{1}{\epsilon} \left(\kappa^{\frac{1}{4}} \sqrt{d} + \frac{\|\nabla f(v_0)\|_2}{\sqrt{M_2}} \right).$$

in expectation.

Now, we apply Lemma 4.4.2 with $\bar{\epsilon} = \sqrt{m_2} h^2 \epsilon / 16$ and use the fact that $f(x^{(0)}) - \min_x f(x) \leq \frac{\|\nabla f(x^{(0)})\|_2^2}{m_2}$, we have

$$\begin{aligned}
\mathbb{E}\left[\frac{1}{N} \sum_{k=0}^{N-1} \|\nabla f(x^{(k)})\|_2^2 \right] &\lesssim \frac{\|\nabla f(x^{(0)})\|_2^2}{m_2 h^2 N} + M_2 d + \frac{\bar{\epsilon}^2}{h^4} \\
&\lesssim \frac{\|\nabla f(x^{(0)})\|_2^2}{m_2 h^2 N} + M_2 d + m_2 \epsilon^2 \\
&\lesssim \|\nabla f(x^{(0)})\|_2^2 + M_2 d + m_2 \epsilon^2 \\
&\lesssim \|\nabla f(x^{(0)})\|_2^2 + M_2 d
\end{aligned}$$

where the third step follows from $N \geq \frac{1}{m_2 h^2}$, our choice of h and ϵ (i.e., Eq. (4.27)), and we used that $\epsilon \leq \sqrt{d}$ at the end. Hence, the expected number of evaluations per each HMC iterations (amortized over all HMC iterations) is

$$\begin{aligned}
O(k) \cdot \log(\frac{C}{\epsilon_{\text{ODE}}}) &\lesssim \frac{1}{\epsilon} \left(\kappa^{\frac{1}{4}} \sqrt{d} + \frac{\|\nabla f(x^{(0)})\| + \sqrt{M_2 d}}{\sqrt{M_2}} \right) \log \left(\frac{h^2 (\|\nabla f(x^{(0)})\| + \sqrt{M_2 d}) + h\sqrt{d}}{\epsilon_{\text{ODE}}} \right) \\
&\lesssim \frac{1}{\epsilon} \left(\kappa^{\frac{1}{4}} \sqrt{d} + \frac{\|\nabla f(x^{(0)})\|}{\sqrt{M_2}} \right) \log \left(\frac{1}{\epsilon} \left(\frac{\|\nabla f(x^{(0)})\|}{\sqrt{m_2}} + \kappa^{3/4} \sqrt{d} \right) \right)
\end{aligned}$$

where the last step follows from our choice of $\epsilon_{\text{ODE}} = \frac{\epsilon \cdot \sqrt{m_2} h^2}{16}$ and our choice of $h = \frac{m_2^{1/4}}{16000 M_2^{3/4}}$. Since we start at the minimum of f, $\|\nabla f(x^{(0)})\| = 0$ and this gives the expected number of evaluations.

Similarly, we have the bound for expected time. This completes the proof. □

4.5 Sampling from Incoherent Logistic Loss Functions and More

In this section we prove Theorem 4.5.6. Since the function

$$f(x) = \sum_{i=1}^{n} \phi_i(a_i^\top x) + \frac{m_2}{2}\|x\|_2^2,$$

the HMC dynamic for sampling $e^{-f(x)}$ is given by

$$\frac{\mathrm{d}^2}{\mathrm{d}t^2}x(t) = -\nabla f(x(t)) = -A^\top \phi'(Ax) - m_2 x$$

where the i^{th} row of $A \in \mathbb{R}^{n\times d}$ is $a_i^\top$, $\forall i \in [n]$ and $\phi' : \mathbb{R}^n \to \mathbb{R}^n$ is defined by

$$\phi'(s) = \left(\frac{\mathrm{d}}{\mathrm{d}s_1}\phi_1(s_1), \frac{\mathrm{d}}{\mathrm{d}s_2}\phi_2(s_2), \cdots, \frac{\mathrm{d}}{\mathrm{d}s_n}\phi_n(s_n)\right).$$

To simplify the proof, we let $s(t) = Ax(t)$. Then, s satisfies the equation

$$\frac{\mathrm{d}^2}{\mathrm{d}t^2}s(t) = F(s(t)) \quad \text{where} \quad F(s) = -AA^\top \phi'(s) - m_2 s. \tag{4.28}$$

Ignoring the term $m_2 s$, F consists of two parts the first part $-AA^\top$ is linear and the second part is decoupled in each variable. This structure allows us to study the dynamic s easily. In this section, we discuss how to approximate the solution of (4.28) using the collocation method.

The proof consists of (a) bounding $\|s(t)\|_\infty$, (b) bounding Lipschitz constant of F and (c) showing that $s(t)$ can be approximated by a polynomial.

4.5.1 ℓ_∞ bound of the dynamic

Lemma 4.5.1 (ℓ_∞ bound of the dynamic)**.** *Let $x^{(j)}$ be the j^{th} iteration of the HMC dynamic defined in Algorithm 4.1 with*

$$f(x) = \sum_{i=1}^{n} \phi_i(a_i^\top x) + \frac{m_2}{2}\|x\|_2^2.$$

Assume that $m_2 \cdot I \preceq \nabla^2 f(x) \preceq M_2 \cdot I$ *for all* x, $|\phi'(s)| \leq M$ *for all* s, *and that* $\tau = \|AA^\top\|_{\infty\to\infty}$. *Let* $s^{(j)} = Ax^{(j)}$. *Suppose that the step size* $T \leq \frac{1}{2\sqrt{m_2}}$, *we have that*

$$\max_{j\in[N]} \|s^{(j)} - s^{(0)}\|_\infty = O\left(\sqrt{\frac{\tau}{m_2}} + \frac{\tau M}{m_2}\right) \cdot \log(dN/\eta)$$

with probability at least $1 - \eta$.

Proof. We ignore the index of iteration (j) and focus on how s changes within each ODE first.

For any $0 \leq t \leq T$ and for any i, we have that

$$s_i(t) = s_i(0) + s_i'(0)t - \int_0^t (t-\ell)(AA^\top \phi'(s(\ell)))_i \mathrm{d}\ell - m_2 \int_0^t (t-\ell) s_i(\ell) \mathrm{d}\ell. \tag{4.29}$$

Using $\tau = \|AA^\top\|_{\infty\to\infty}$ and $|\phi'(s)| \leq M$, we have

$$\begin{aligned}
|s_i(t) - s_i(0)| &\leq T \cdot |s_i'(0)| + \int_0^T (T-\ell)\cdot\tau\cdot M \mathrm{d}\ell + m_2 \int_0^t (t-\ell)|s_i(\ell) - s_i(0)|\mathrm{d}\ell + \frac{1}{2} m_2 T^2 |s_i(0)| \\
&= \frac{1}{2} m_2 T^2 \cdot |s_i(0)| + T \cdot |s_i'(0)| + \frac{1}{2} T^2 \cdot \tau M + m_2 \int_0^t (t-\ell)|s_i(\ell) - s_i(0)|\mathrm{d}\ell.
\end{aligned}$$

Solving this integral inequality (Lemma 4.6.2), we get

$$\begin{aligned}
|s_i(t) - s_i(0)| &\leq \left(\frac{1}{2} m_2 T^2 \cdot |s_i(0)| + T \cdot |s_i'(0)| + \frac{1}{2} T^2 \cdot \tau M\right) \cdot \cosh(\sqrt{m_2} t) \\
&\leq \frac{1}{4}|s_i(0)| + 2T \cdot |s_i'(0)| + T^2 \cdot \tau M
\end{aligned} \tag{4.30}$$

where we used that $t \leq T \leq \frac{1}{2\sqrt{m_2}}$.

Using this estimate back to (4.29), we have

$$\begin{aligned}
&\left|s_i(T) - s_i(0) - s_i'(0)T + \frac{1}{2}m_2T^2s_i(0)\right| \\
\leq\ & \int_0^T (T-\ell)|AA^\top\phi'(s(\ell))_i|\mathrm{d}\ell + m_2\int_0^T (T-\ell)|s_i(\ell)-s_i(0)|\mathrm{d}\ell \\
\leq\ & \int_0^T (T-\ell)\tau M\mathrm{d}\ell + m_2\int_0^T (T-\ell)\left(\frac{1}{4}|s_i(0)| + 2T\cdot|s_i'(0)| + T^2\cdot\tau M\right)\mathrm{d}\ell \\
=\ & \frac{1}{2}T^2\cdot\tau M + \frac{1}{2}m_2T^2\cdot\left(\frac{1}{4}|s_i(0)| + 2T\cdot|s_i'(0)| + T^2\cdot\tau M\right) \\
\leq\ & \frac{1}{2}T^2\cdot\tau M + \frac{1}{2}m_2T^2\cdot\left(\frac{1}{4}|s_i(0)| + \frac{1}{\sqrt{m_2}}\cdot|s_i'(0)| + \frac{1}{4m_2}\cdot\tau M\right) \\
=\ & \frac{5}{8}T^2\cdot\tau M + \frac{1}{8}m_2T^2|s_i(0)| + \frac{1}{2}\sqrt{m_2}T^2\cdot|s_i'(0)|
\end{aligned}$$

where the second step follows from $|AA^\top\phi'(s(\ell))_i| \leq \tau M$ and (4.30), the third step follows from $\int_0^T (T-\ell)\mathrm{d}\ell = \frac{1}{2}T^2$, the fourth step follows from $T \leq \frac{1}{2\sqrt{m_2}}$.

Note that we bounded how much each iteration of the HMC dynamic can change the solution. Writing it differently, we have

$$s_i^{(j+1)} = (1 - \frac{1}{2}m_2T^2)s_i^{(j)} + s_i^{(j)'}(0)\cdot T + \beta$$

where

$$|\beta| \leq \frac{1}{4}m_2T^2|s_i^{(j)}(0)| + \sqrt{m_2}T^2\cdot|s_i^{(j)'}(0)| + \tau M\cdot T^2.$$

Note that $s_i^{(j)'}(0) \sim \mathcal{N}(0, (A^\top A)_i)$ and that

$$\lambda_{\max}(A^\top A) = \lambda_{\max}(AA^\top) \leq \|AA^\top\|_{\infty\to\infty} = \tau.$$

Therefore, $s_i^{(j)'}(0)T \sim \alpha_i\cdot\mathcal{N}(0,1)$ with $0 \leq \alpha_i \leq \sqrt{\tau}T$. Now, we simplify the dynamic to

$$s_i^{(j+1)} = (1-\delta)s_i^{(j)} + \alpha_iN^{(j)} + \beta$$

where $\delta := \frac{1}{2}m_2T^2$, $\alpha_i \le a := \sqrt{\tau}T$, and

$$|\beta| \le \frac{\delta}{2}|s_i^{(j)}(0)| + b|N^{(j)}| + c$$

with $b := \sqrt{m_2\tau}T^2$ and $c := \tau M \cdot T^2$.

Applying Lemma 4.5.2, we have that

$$\Pr\left[\max_{j\in[N]}|s_i^{(j)} - s_i^{(0)}| \ge C \cdot \left(\frac{a}{\sqrt{\delta}} + \frac{c+b}{\delta}\right)\log(N/\epsilon)\right] \le \epsilon$$

for some constant C. Taking union bound over $i \in [d]$, the bound follows from the calculation

$$\frac{a}{\sqrt{\delta}} + \frac{c+b}{\delta} = \frac{\sqrt{2\tau}T}{\sqrt{m_2}T} + \frac{\tau M \cdot T^2 + \sqrt{m_2\tau}\cdot T^2}{2m_2T^2} \gtrsim \sqrt{\frac{\tau}{m_2}} + \frac{\tau M}{m_2}.$$

□

Lemma 4.5.2 (Bounding the Martingale)**.** *Let $X^{(i)}$ be a sequence of random variable such that*

$$X^{(i+1)} = (1-\delta)X^{(i)} + \alpha_i N^{(i)} + \beta$$

where $N^{(i)} \sim \mathcal{N}(0,1)$ are independent, $\alpha_i \le a$, $\beta \le \frac{\delta}{2}|X^{(i)}| + b|N^{(i)}| + c$ with positive a, b, c and $0 < \delta \le 1$. For some constant universal $C > 0$, we have that

$$\Pr\left[\max_{i\in[k]}|X^{(i)} - X^{(0)}| \ge C \cdot \left(\frac{a}{\sqrt{\delta}} + \frac{c+b}{\delta}\right)\log(k/\epsilon)\right] \le \epsilon$$

for any $0 < \epsilon < 1$.

Proof. We will first show that $X^{(i)}$ cannot grow to large. The proof of the other direction is similar. Consider the potential $\Phi^{(i)} = \mathbb{E}\left[e^{\lambda X^{(i)}}\right]$. Note that

$$\begin{aligned}\Phi^{(i+1)} &\le \mathbb{E}\left[e^{\lambda((1-\delta)X^{(i)} + \alpha_i N^{(i)} + b|N^{(i)}| + c + \frac{\delta}{2}|X^{(i)}|)}\right] \\ &\le e^{\lambda c}\,\mathbb{E}\left[e^{\lambda(1-\delta)X^{(i)} + \frac{\delta}{2}|X^{(i)}|}\right]\mathbb{E}\left[e^{\lambda(\alpha_i N^{(i)} + b|N^{(i)}|)}\right],\end{aligned}$$

where we used that $N^{(i)}$ are independent. Picking $\lambda \leq \frac{1}{a+b}$ and using $N^{(i)} \sim \mathcal{N}(0,1)$ and $|\alpha_i| \leq a$, we have that

$$\mathbb{E}\left[e^{\lambda(\alpha_i N^{(i)} + b|N^{(i)}|)}\right] \leq e^{O(\lambda b + \lambda^2 a^2)}.$$

Therefore, for any $\eta > 0$, we have

$$\begin{aligned}
\Phi^{(i+1)} &\leq e^{O(\lambda c + \lambda b + \lambda^2 a^2)} \cdot \left(\mathbb{E}\left[e^{\lambda(1-\delta)X^{(i)} + \frac{\delta}{2}|X^{(i)}|} 1_{X^{(i)} \leq \eta}\right] + \mathbb{E}\left[e^{\lambda(1-\frac{\delta}{2})X^{(i)}} 1_{X^{(i)} > \eta}\right]\right) \\
&\leq e^{O(\lambda c + \lambda b + \lambda^2 a^2)} \cdot \left(e^{\lambda\eta} + \mathbb{E}\left[e^{\lambda X^{(i)} - \frac{\lambda}{2}\delta\eta}\right]\right) \\
&\leq e^{O(\lambda c + \lambda b + \lambda^2 a^2 + \lambda\eta)} + e^{O(\lambda c + \lambda b + \lambda^2 a^2) - \frac{\lambda}{2}\delta\eta} \Phi^{(i)}.
\end{aligned}$$

Choose $\eta = \Theta(\frac{c+b+\lambda a^2}{\delta})$ such that $O(\lambda c + \lambda b + \lambda^2 a^2) - \frac{\lambda}{2}\delta\eta \leq 0$. Hence, we have

$$\Phi^{(i+1)} \leq e^{O(\lambda c + \lambda b + \lambda^2 a^2 + \lambda\eta)} + \Phi^{(i)} \leq e^{O(\frac{\lambda c + \lambda b + \lambda^2 a^2}{\delta})} + \Phi^{(i)}.$$

Picking an appropriate λ, we have the result. □

4.5.2 Lipschitz constant of F

Now, we bound the Lipschitz constant of function F.

Lemma 4.5.3 (Lipschitz bound of function F)**.** *Let $L_{\phi'}$ be the Lipschitz constant of ϕ', i.e.,*

$$\|\phi'(s_1) - \phi'(s_2)\|_\infty \leq L_{\phi'} \|s_1 - s_2\|_\infty, \forall s_1, s_2.$$

The function F defined in (4.28) has Lipschitz constant $(L_{\phi'}\tau + m_2)$ in ℓ_∞ norm where $\tau = \|AA^\top\|_{\infty\to\infty}$.

Proof. Note that

$$\begin{aligned}\|F(s_1) - F(s_2)\|_\infty &= \|AA^\top(\phi'(s_1) - \phi'(s_2))\|_\infty + m_2\|s_1 - s_2\|_\infty \\ &\le \tau \cdot \|\phi'(s_1) - \phi'(s_2)\|_\infty + m_2\|s_1 - s_2\|_\infty \\ &\le (\tau \cdot L_{\phi'} + m_2) \cdot \|s_1 - s_2\|_\infty,\end{aligned}$$

where the second step follows by $\|AA^\top\|_{\infty\to\infty} = \tau$, and second step follows by ϕ' is $L_{\phi'}$-Lipschitz function.

□

Remark 4.5.1. If we think of the role of F in our second order ODE (4.28), then F is in fact independent of $\frac{\mathrm{d}s}{\mathrm{d}t}$. Therefore $L_1 = 0$ and $L_2 = L_{\phi'}\tau + m_2$.

4.5.3 Existence of low-degree solutions

Next, we establish bounds on the radius up to which the solution to the ODE (4.28) have a low-degree polynomial approximation.

Lemma 4.5.4 (Low-degree polynomial approximation)**.** *Assume that $\|AA^\top\|_{\infty\to\infty} = \tau$ and that for all i, ϕ_i' has Cauchy estimate M with radius r, i.e.,*

$$\forall l \ge 0, \forall a \in \mathbb{R}, |(\phi_i)^{(l+1)}(a)| \le M \cdot l! \cdot r^{-l}.$$

Let $s^*(t) \in \mathbb{R}^n$ denote the solution of the ODE (4.28).

For any

$$0 \le T \le \frac{r}{4}\left((M\tau r + m_2 r(r + \|s(0)\|_\infty))^{1/2} + \|s'(0)\|_\infty\right)^{-1}$$

and any $0 < \epsilon < 1$*, there is a degree* $D = 2\lceil 4 + \log_2(1/\epsilon)\rceil$ *polynomial* $q : \mathbb{R} \to \mathbb{R}^n$ *such that*

$$q(0) = s^*(0), q'(0) = s^{*\prime}(0), \text{ and } \left\| \frac{\mathrm{d}^2}{\mathrm{d}t^2} q(t) - \frac{\mathrm{d}^2}{\mathrm{d}t^2} s^*(t) \right\|_\infty \leq \frac{\epsilon \cdot r}{T^2}, \forall t \in [0, T]$$

First, we verify the condition in Lemma 4.6.6.

Lemma 4.5.5. (Bounding derivatives of F)**.** Under the same assumptions in Lemma 4.5.4, we have

$$\|D^k F(s)[\Delta_1, \Delta_2, \cdots, \Delta_k]\|_\infty \leq g^{(k)}(0) \cdot \prod_{j=1}^{k} \|\Delta_j\|_\infty, \forall k \geq 0, \forall \Delta_1, \cdots, \Delta_k.$$

where

$$g(x) = \frac{\tau \cdot M + m_2 \cdot (r + \|s\|_\infty)}{1 - r^{-1}x}.$$

Proof. Recall that $\phi'(x) = (\phi_1'(x_1), \phi_2'(x_2), \cdots, \phi_n'(x_n))$, $\forall x \in \mathbb{R}^n$. We have

$$\begin{aligned}
&\|AA^\top D^k \phi'(s)[\Delta_1, \Delta_2, \cdots, \Delta_k]\|_\infty \\
\leq\ & \tau \cdot \|D^k \phi'(s)[\Delta_1, \Delta_2, \cdots, \Delta_k]\|_\infty && \text{by } \|AA^\top\|_{\infty\to\infty} = \tau \\
\leq\ & \tau \cdot \max_{i\in[n]} |\phi_i^{(k+1)}| \cdot |\Delta_{1,i}| \cdot |\Delta_{2,i}| \cdots |\Delta_{k,i}| \\
\leq\ & \tau \cdot \max_{i\in[n]} |\phi_i^{(k+1)}| \prod_{j=1}^{k} \|\Delta_j\|_\infty \\
\leq\ & \tau \cdot M \cdot k! \cdot r^{-k} \cdot \prod_{j=1}^{k} \|\Delta_j\|_\infty.
\end{aligned}$$

Next, the derivatives of the $m_2 s$ term in $F(s) = -AA^\top \phi'(s) - m_2 s$ can be bounded by the derivatives of $x \to m_2(\|s\|_\infty + x)$, which then can be bounded by the derivatives of $x \to m_2(r + \|s\|_\infty)/(1 - r^{-1}x)$. This explains the second part of the function g. □

Proof of Lemma 4.5.4. Theorem 4.6.4 and Lemma 4.5.5 shows that

$$\|s^{*(k)}(0)\|_\infty \le \frac{k!\alpha^k}{r^{-1}} = r \cdot k! \cdot \alpha^k \tag{4.31}$$

where

$$\alpha = \max\left(\left(\frac{4}{3} \cdot (M\tau + m_2(r + \|s(0)\|_\infty)) \cdot r^{-1}\right)^{1/2}, 2 \cdot \|s'(0)\|_\infty r^{-1}\right).$$

Since s^* is real analytic at 0 (Theorem 4.6.4), around $t = 0$, we have

$$s^*(t) = \sum_{k=0}^{\infty} \frac{s^{*(k)}(0)}{k!} t^k.$$

Apply Theorem 4.6.4 repeatedly at every t such that $s^*(t)$ is defined, we can show that the above equation holds as long as the right hand side converges.

Let $q(t) = \sum_{k=0}^{D} s^{*(k)}(0)t^k$. Then, we have that

$$\begin{aligned}
\left\|\frac{\mathrm{d}^2}{\mathrm{d}t^2}q(t) - \frac{\mathrm{d}^2}{\mathrm{d}t^2}s^*(t)\right\|_\infty &\le \left\|\sum_{k=D+1}^{\infty} \frac{s^{*(k)}(0)}{(k-2)!} t^{k-2}\right\|_\infty \\
&\le r \cdot \sum_{k=D+1}^{\infty} \frac{k!}{(k-2)!}\alpha^k t^{k-2} \\
&\le \frac{r}{T^2} \cdot \sum_{k=D+1}^{\infty} \frac{(k-1)k}{2^k} \\
&= \frac{r}{T^2} \cdot 2^{-D}(D^2 + 3D + 4) \\
&\le 16\frac{2^{-D/2}r}{T^2}
\end{aligned}$$

where we used (4.31) at the second inequality, $\alpha T \le \frac{1}{2}$ at the third inequality, and $D \ge 1$ at the fourth inequality. □

4.5.4 Main result

Theorem 4.5.6 (Formal version of Theorem 4.1.2)**.** *Let* $A = [a_1; a_2; \cdots; a_n] \in \mathbb{R}^{n \times d}$, $\phi_i : \mathbb{R} \to \mathbb{R}$ *be a function, and*

$$f(x) = \sum_{i=1}^{n} \phi_i(a_i^\top x) + \frac{m_2}{2}\|x\|^2.$$

Let $\tau = \|AA^\top\|_{\infty \to \infty}$ *and suppose that*

1. *f has M_2 Lipschitz gradient, i.e., $\nabla^2 f(x) \preceq M_2 \cdot I$ for all x,*

2. *ϕ_i' has Cauchy estimate M with radius r, i.e., $\forall i \in [n], \ell \geq 1, s \in \mathbb{R}$, $|\phi_i^{(l+1)}(s)| \leq M \cdot l! \cdot r^{-l}$.*

Starting at $x^{(0)}$, we can output a random point X such that

$$\mathop{\mathbb{E}}_{Y \propto e^{-f}} \left[\|X - Y\|_2^2\right] \leq \frac{\epsilon}{\sqrt{m_2}}$$

using $N \lesssim k \log\left(\frac{k}{\epsilon}\left(\frac{\|\nabla f(x^{(0)})\|^2}{m_2} + d\right)\right)$ iterations with

$$k \lesssim \kappa^{1.5} + \frac{M\tau}{m_2 r}\log(dN) + \frac{\tau}{m_2 r^2}\log^2(dN) \quad \text{with} \quad \kappa = \frac{M_2}{m_2}.$$

Each iteration takes $O(d \log^3(\frac{1}{\delta}))$ time and $O(\log^2(\frac{1}{\delta}))$ evaluations to the function ϕ' and the matrix vector multiplications for A and $A^\top$, with

$$\delta = \Omega(\frac{1}{r})\sqrt{\frac{\lambda_{\min}(A^\top A)}{n \cdot m_2}} \cdot \frac{\epsilon}{k}.$$

Proof. The proof consists of bounding the cost of each HMC step in Algorithm 4.1 and bounding the number of steps.

Cost per iteration:

Parameters	Value	Source
k	2	Eq. (4.28)
L_1	0	Lemma 4.5.3 and Remark 4.5.1
L_2	$\frac{M\tau}{r} + m_2$	Lemma 4.5.3 and Remark 4.5.1
D	$O(\log(1/\delta))$	Lemma 4.5.4
C	$O(r)$	Eq. (4.34)
ϵ_{ODE}	$O(\delta r)$	Eq. (4.33)

Table 4.2: Summary of parameters of Theorem 4.2.4

As we discussed in this section, we consider the ODE (4.28) instead. We will use Theorem 4.2.4 to solve the ODE (4.28). Hence, we need to bound the parameters of Theorem 4.2.4, which are summarized in Table 4.2.

Parameters D and ϵ_{ODE}: Let s denote its solution. Lemma 4.5.4 shows that if

$$h \le \frac{r}{4}\left((M\tau \cdot r + m_2 r(r + \|s(0)\|_\infty))^{1/2} + \|s'(0)\|_\infty\right)^{-1}, \tag{4.32}$$

for any $\delta > 0$, there is a degree $O(\log(1/\delta))$ polynomial q such that

$$q(0) = s(0), \quad q'(0) = s'(0), \quad \text{and} \quad \left\|\frac{\mathrm{d}^2}{\mathrm{d}t^2}q(t) - \frac{\mathrm{d}^2}{\mathrm{d}t^2}s(t)\right\|_\infty \le \frac{\delta \cdot r}{T^2} \text{ for } t \in [0, h]. \tag{4.33}$$

Parameter C: To apply Theorem 4.2.4, we first show parameter $C \le O(r)$ as follows:

$$\begin{aligned}
C &\lesssim h \cdot (h\|F(s(0))\|_\infty + \|s'(0)\|_\infty) \\
&\le h \cdot \left(h\left(\left\|AA^\top \phi'(s(0))\right\|_\infty + \|m_2 s(0)\|_\infty\right) + \|s'(0)\|_\infty\right) \\
&\le h^2\tau \|\phi'(s(0))\|_\infty + h^2 m_2 \|s(0)\|_\infty + h\|s'(0)\|_\infty \\
&\le h^2\tau M + h^2 m_2 \|s(0)\|_\infty + h\|s'(0)\|_\infty \\
&\le \frac{r}{16} + \frac{r}{16} + \frac{r}{4} \le r.
\end{aligned} \tag{4.34}$$

where the second step follows from (4.28) and triangle inequality, the third step follows from $\|AA^\top\|_{\infty\to\infty} = \tau$, the fourth step follows from ϕ' has Cauchy estimate M with radius r, and the last step follows from (4.32).

Now, Theorem 4.2.4 shows that if h satisfies (4.32) and that

$$h \le \frac{1}{16000}\frac{1}{L} = \frac{1}{16000}\frac{1}{L_1 + \sqrt{L_2}} = \frac{1}{16000}\sqrt{\frac{r}{M\tau + m_2 r}}, \tag{4.35}$$

then, we can find p such that

$$\|s(h) - p\|_\infty \le O(\delta \cdot r) \tag{4.36}$$

using $O(\log(\frac{1}{\delta})\log(\frac{C}{\delta\cdot r})) = O(\log^2(\frac{1}{\delta}))$ evaluations of ϕ_i' and $O(d\log^2(\frac{1}{\delta})\log(\frac{C}{\delta\cdot r})) = O(d\log^3(\frac{1}{\delta}))$ time.

To understand the condition in Eq. (4.32), we note that $s'(0) = Av$ where $v \sim \mathcal{N}(0, I)$. Hence, $s^{*\prime}(0) \sim \mathcal{N}(0, A^\top A)$. Note that $\lambda_{\max}(A^\top A) = \lambda_{\max}(AA^\top) \le \|AA^\top\|_{\infty\to\infty} = \tau$. Hence, we have that

$$\|s'(0)\|_\infty = O(1) \cdot \sqrt{\tau \cdot \log(dN/\eta)}$$

with probability at least $1 - \eta$ probability for all N iterations. In Lemma 4.5.1, we proved that

$$\|s(0)\|_\infty = O\left(\sqrt{\frac{\tau}{m_2}} + \frac{\tau M}{m_2}\right) \cdot \log(dN/\eta)$$

with probability at least $1 - \eta$ for all N iterations.

Putting the bound on $\|s'(0)\|$ and $\|s(0)\|$ into the right hand side of Eq. (4.32) gives

$$\begin{aligned}
&\frac{1}{r}\left(\sqrt{M\tau\cdot r + m_2 r(r+\|s(0)\|_\infty)} + \|s'(0)\|_\infty\right) \qquad (4.37)\\
&\lesssim \sqrt{\frac{M\tau}{r}} + \sqrt{m_2} + \sqrt{\frac{m_2}{r}\left(\sqrt{\frac{\tau}{m_2}} + \frac{\tau M}{m_2}\right)\log(dN/\eta)} + \frac{\sqrt{\tau}}{r}\sqrt{\log(dN/\eta)}\\
&\leq \frac{M_2^{3/4}}{m_2^{1/4}} + \left(\frac{(\tau m_2)^{1/4}}{r^{1/2}} + \sqrt{\frac{M\tau}{r}} + \frac{\sqrt{\tau}}{r}\right)\sqrt{\log(dN/\eta)}\\
&\leq \frac{M_2^{3/4}}{m_2^{1/4}} + \left(\frac{\sqrt{\tau}}{r}\sqrt{\log(dN/\eta)} + \frac{\sqrt{m_2}}{\sqrt{\log(dN/\eta)}} + \sqrt{\frac{M\tau}{r}} + \frac{\sqrt{\tau}}{r}\right)\sqrt{\log(dN/\eta)}\\
&\lesssim \frac{M_2^{3/4}}{m_2^{1/4}} + \frac{\sqrt{\tau}}{r}\log(dN/\eta) + \sqrt{\frac{M\tau}{r}}\sqrt{\log(dN/\eta)}
\end{aligned}$$

where the second step follows by Eq. (4.32), the third step follows by $\sqrt{m_2} \leq \frac{M_2^{3/4}}{m_2^{1/4}}$ and $\sqrt{\log(dN/\eta)} \geq 1$, the fourth step follows by $ab \leq a^2 + b^2$, the fifth step follows by $\sqrt{\log(dN/\eta)} \geq 1$ and $\sqrt{m_2} \leq \frac{M_2^{3/4}}{m_2^{1/4}}$.

Therefore,

$$h = \Theta\left(\sqrt{\frac{M\tau}{r}}\sqrt{\log(dN/\eta)} + \frac{\sqrt{\tau}}{r}\log(dN/\eta) + \frac{M_2^{3/4}}{m_2^{1/4}}\right)^{-1} \qquad (4.38)$$

satisfies the condition in Eq. (4.32) and Eq. (4.35). It also satisfies the condition in Theorem 4.3.2 ($h \leq \frac{m_2^{1/4}}{2M_2^{3/4}}$).

Next, we note that the corresponding HMC dynamic $x^*(h)$ is given by

$$x^*(h) = (A^\top A)^{-1}A^\top s^*(h).$$

Let p be the approximate of $s^*(h)$ we find using Theorem 4.3.2 and $q = (A^\top A)^{-1}A^\top p$, then,

we have

$$\begin{aligned}\|x^*(h) - q\|_2 &= \|(A^\top A)^{-1}A^\top s^*(h) - (A^\top A)^{-1}Ap\|_2 \\ &\leq \|(A^\top A)^{-1}A^\top\|_{2\to 2} \cdot \|s^*(h) - p\|_2 \\ &\leq \|(A^\top A)^{-1}A^\top\|_{2\to 2} \cdot O(\sqrt{n} \cdot r \cdot \delta) \\ &\leq (\lambda_{\min}(A^\top A))^{-\frac{1}{2}} \cdot O(\sqrt{n} \cdot r \cdot \delta),\end{aligned}$$

where the first step follows by definition of $x^*(h)$ and q, the second step follows by definition of $\|\|_{2\to 2}$ norm, the third step follows from $\|s^*(h) - p\|_2 \leq O(\sqrt{n}r\delta)$ (implied by (4.36) and $\|\|_2 \leq \sqrt{n}\|\|_\infty$).

Using such q as an approximation of $x^*(h)$ in Algorithm 4.1, Theorem 4.3.2 shows that the W_2 error of the sampled point is bounded by

$$O\left((\lambda_{\min}(A^\top A))^{-\frac{1}{2}} \cdot \sqrt{n} \cdot r \cdot \delta\right) \leq \frac{\epsilon \cdot \theta}{2\sqrt{m_2}}.$$

where $\theta = \frac{m_2 h^2}{8}$ and the last step follows from picking

$$\delta = c \cdot \frac{1}{r}\sqrt{\frac{\lambda_{\min}(A^\top A)}{n \cdot m_2}} \cdot \epsilon \cdot \theta$$

for some small enough c. The cost of each iteration follows from Theorem 4.2.4 and all parameters we pick.

Number of iterations:

Theorem 4.3.2 shows that the number of iterations is

$$O\left(\frac{1}{\theta}\right) \cdot \left(\log\left(\frac{1}{\theta \cdot \epsilon}\right) + \log\left(\frac{\|\nabla f(x^{(0)})\|^2}{m_2} + d\right)\right).$$

Finally, we bound the term $1/\theta$ as follows

$$\begin{aligned}\frac{1}{\theta} &\lesssim \frac{1}{m_2}\left(\frac{M\tau}{r}\log\left(\frac{dN}{\eta}\right)+\frac{\tau}{r^2}\log^2\left(\frac{dN}{\eta}\right)+\frac{M_2^{3/2}}{\sqrt{m_2}}\right)\\ &= \kappa^{1.5}+\frac{M\tau}{m_2 r}\log\left(\frac{dN}{\eta}\right)+\frac{\tau}{m_2 r^2}\log^2\left(\frac{dN}{\eta}\right).\end{aligned}$$

where we used $\kappa = M_2/m_2$ and (4.38). □

4.6 Preliminaries

4.6.1 Notation

For any function f, we define $\widetilde{O}(f)$ to be $f \cdot \log^{O(1)}(f)$. In addition to $O(\cdot)$ notation, for two functions f, g, we use the shorthand $f \lesssim g$ (resp. $\gtrsim$) to indicate that $f \leq Cg$ (resp. $\geq$) for an absolute constant C.

Definition 4.6.1 ($p \to q$ norm)**.** Given matrix $A \in \mathbb{R}^{n \times d}$, we define $\|\|_{p \to q}$ norm as follows

$$\|A\|_{p \to q} = \max_{x \in \mathbb{R}^d} \frac{\|Ax\|_q}{\|x\|_p}.$$

$\|A\|_{\infty \to \infty}$ is a special case where $p = \infty$ and $q = \infty$.

Definition 4.6.2 (Wasserstein distance)**.** The k-th Wasserstein distance between two probability measure μ and ν is

$$W_k(\mu, \nu) = \left(\inf_{(X,Y) \in \mathcal{C}(\mu,\nu)} \mathbb{E}\left[\|X - Y\|^k \right] \right)^{1/k},$$

where $\mathcal{C}(\mu, \nu)$ is the set of all couplings of μ and ν.

Definition 4.6.3 (Cauchy's estimates)**.** We say function ϕ has Cauchy estimate M and radius of convergence r, if for all $x \in \mathbb{R}$ and for all integers $l \geq 0$

$$|\phi^{(l)}(x)| \leq M \cdot l! \cdot r^{-l}.$$

Lemma 4.6.1. *If the number of non-zeros for each row of $AA^\top$ is bounded by s, then*

$$\lambda_{\max}(AA^\top) \leq \|AA^\top\|_{\infty \to \infty} \leq \sqrt{s} \cdot \lambda_{\max}(AA^\top).$$

Proof. Let v be the maximum eigenvalue of $AA^\top$. Then, we have that

$$AA^\top v = \lambda_{\max}(AA^\top) \cdot v.$$

Since $\|AA^\top\|_{\infty\to\infty} = \max_{\|v\|_\infty=1} \|AA^\top v\|_\infty$, we have that $\lambda_{\max}(AA^\top) \le \|AA^\top\|_{\infty\to\infty}$.

For the another direction,

$$\begin{aligned}\|AA^\top\|_{\infty\to\infty} &= \max_i \sum_j |AA^\top|_{ij} \le \max_i \sqrt{s} \cdot \sqrt{\sum_j (AA^\top)_{ij}^2} \\ &\le \sqrt{s} \cdot \max_i \max_{\|v\|_2=1} e_i^\top AA^\top v = \sqrt{s} \cdot \lambda_{\max}(AA^\top).\end{aligned}$$

□

4.6.2 Simple ODEs

We prove two helper lemmas (Lemma 4.6.2 and 4.6.3) for the later use.

Lemma 4.6.2. *Given a continuous function $v(t)$ and positive scalars β, γ such that*

$$0 \le v(t) \le \beta + \gamma \int_0^t (t-s)v(s)\mathrm{d}s.$$

We have that $v(t) \le \beta \cosh(\sqrt{\gamma}t)$ for all $t \ge 0$.

Proof. Let $\overline{v}(t)$ be the solution of the integral equation $\overline{v}(t) = \beta + \gamma \int_0^t (t-s)\overline{v}(s)\mathrm{d}s$. Note that $\overline{v}$ satisfies the ODE

$$\overline{v}''(t) = \gamma \overline{v}(t), \quad \overline{v}'(0) = 0, \quad \overline{v}(0) = \beta.$$

Solving it, we have $\overline{v}(t) = \beta \cosh(\sqrt{\gamma}t)$. Hence, it suffices to prove that $v(t) \le \overline{v}(t)$ for all $t \ge 0$.

Fix any $\epsilon > 0$. We let T be the supremum such that $(1+\epsilon)\overline{v}(t) \geq v(t)$ for all $0 \leq t \leq T$. Suppose $T < +\infty$. Then, we have that

$$v(T) \leq \beta + \gamma \int_0^T (T-s)v(s)\mathrm{d}s \leq \beta + (1+\epsilon)\gamma \int_0^T (T-s)\overline{v}(s)\mathrm{d}s < (1+\epsilon)\overline{v}(T).$$

By the continuity of v and $\overline{v}$, we show that T is not the supremum. This is a contradiction. Therefore, $T = +\infty$ for any $\epsilon > 0$. □

Lemma 4.6.3. *Given a continuous function $v(t)$ and positive scalars β, γ such that*

$$0 \leq v(t) \leq \beta + \gamma \int_0^t v(s)\mathrm{d}s.$$

We have that $v(t) \leq \beta e^{\gamma t}$ for all $t \geq 0$.

Proof. The proof is identical to Lemma 4.6.2.

Let $\overline{v}(t)$ be the solution of the integral equation $\overline{v}(t) = \beta + \gamma \int_0^t \overline{v}(s)\mathrm{d}s$. Note that $\overline{v}$ satisfies the ODE

$$\overline{v}''(t) = \gamma \overline{v}(t), \quad \overline{v}'(0) = 0, \quad \overline{v}(0) = \beta.$$

Solving it, we have $\overline{v}(t) = \beta \exp(\gamma t)$. Hence, it suffices to prove that $v(t) \leq \overline{v}(t)$ for all $t \geq 0$.

Fix any $\epsilon > 0$. We let T be the supremum such that $(1+\epsilon)\overline{v}(t) \geq v(t)$ for all $0 \leq t \leq T$. Suppose $T < +\infty$. Then, we have that

$$v(T) \leq \beta + \gamma \int_0^T v(s)\mathrm{d}s \leq \beta + (1+\epsilon)\gamma \int_0^T \overline{v}(s)\mathrm{d}s < (1+\epsilon)\overline{v}(T).$$

By the continuity of v and $\overline{v}$,we show that T is not the supremum. This is a contradiction. Therefore, $T = +\infty$ for any $\epsilon > 0$. □

4.6.3 Cauchy Estimates and Method of Majorants

In order to prove the solution of the HMC dynamic can be approximated by a low degree polynomial, we first give a general bound the k^{th} derivative of the second order ODE $u''(t) = F(u(t))$, by reducing it to bounding derivatives of a one-variable ODE. The low degree result then follows from: first, we take the Taylor expansion of the original function; second, truncate it at a certain degree; finally we can claim that the low-degree truncation provides a good approximation to the original function.

Theorem 4.6.4. *Given vectors $v_1, v_0 \in \mathbb{R}^d$ and any norm $\|\cdot\|$ on $\mathbb{R}^d$. Let $U \subset \mathbb{R}^d$ be a neighborhood of v_0 and that $F : U \to \mathbb{R}^d$ is real analytic near v_0. Suppose that*

$$\|D^{(k)}F(v_0)[\Delta_1, \Delta_2, \cdots, \Delta_k]\| \leq k! \cdot a \cdot c^k \prod_{j=1}^{k} \|\Delta_j\| \text{ for all } k \geq 0.$$

Then, the ODE

$$\frac{\mathrm{d}^2}{\mathrm{d}t^2}u(t) = F(u(t)), \frac{\mathrm{d}}{\mathrm{d}t}u(0) = v_1, u(0) = v_0 \tag{4.39}$$

has a unique real analytic solution around $t = 0$. Furthermore, we have

$$\|u^{(k)}(0)\| \leq \frac{k!\alpha^k}{c} \quad \forall k \geq 1$$

with $\alpha = \max(\sqrt{\frac{4}{3}ac}, 2\|v_1\|c)$.

Theorem 4.6.4 involves two steps. The first step (Lemma 4.6.6) involves bounding the derivatives of the solution of the multivariate ODE (4.39) by its scalar version. The second step (Lemma 4.6.7) involves bounding the scalar ODE directly.

The first step follows directly from the majorization proof of Cauchy–Kowalevski theorem. See [vdH03] for an introduction of the method of majorants. This theorem usually

stated qualitatively without an explicit bound. For completeness, we include a proof for the first step.

Theorem 4.6.5 (Cauchy–Kowalevski theorem [Cau42, Kow75, Kro83, Nak94])**.** *Given vectors* $v_{k-1}, v_{k-2}, \cdots, v_0 \in \mathbb{R}^d$*. Let* $U \subset \mathbb{R}^{kd+1}$ *be a neighborhood of* $z \stackrel{\text{def}}{=} (v_{k-1}, v_{k-2}, \cdots, v_0, 0)$ *and that* $F : U \to \mathbb{R}^d$ *is a real analytic near* z*. Then, the ODE*

$$\frac{d^k}{\mathrm{d}t^k} x(t) = F\left(\frac{\mathrm{d}^{k-1}}{\mathrm{d}t^{k-1}} x(t), \cdots, x(t), t\right),$$
$$\frac{d^i}{\mathrm{d}t^i} x(0) = v_i, \forall i \in \{k-1, \cdots, 1, 0\}$$

has a unique real analytic solution around $t = 0$*.*

Lemma 4.6.6. (Bounding multivariate ODE by scalar ODE)**.** Given vectors $v_1, v_0 \in \mathbb{R}^d$ and any norm $\|\cdot\|$ on $\mathbb{R}^d$. Let $U \subset \mathbb{R}^d$ be a neighborhood of v_0 and that $F : U \to \mathbb{R}^d$ is a real analytic near v_0. Suppose that

$$\|D^{(k)} F(v_0)[\Delta_1, \Delta_2, \cdots, \Delta_k]\| \leq f^{(k)}(0) \prod_{j=1}^{k} \|\Delta_j\| \text{ for all } k \geq 0$$

for some real analytic function f around 0. Then the ODE (4.39) has a unique real analytic solution around $t = 0$. Furthermore, for any $b \geq \|v_1\|$, we have

$$\|u^{(k)}(0)\| \leq \psi^{(k)}(0) \quad \forall k \geq 1$$

where ψ is the solution of the ODE

$$\psi''(t) = f(\psi(t)), \psi'(0) = b, \psi(0) = 0.$$

For many functions F, including the HMC dynamic or complex analytic F, we can pick bound the derivatives of F by the function $f(x) = \frac{a}{1-cx}$ for some a and c in Lemma 4.6.6. Therefore, we only need to give a bound on the scalar ODE for this function f.

Lemma 4.6.7. (Bounding scalar ODE) Let $f(x) = \frac{a}{1-cx}$ with positive a and c. Let $\psi(t)$ denote the solution of the ODE

$$\psi''(t) = f(\psi(t)), \psi'(0) = b, \psi(0) = 0$$

with $b \geq 0$. Then,

$$\psi^{(k)}(0) \leq \frac{k!\alpha^k}{c} \quad \forall k \geq 1$$

with $\alpha = \max(\sqrt{\frac{4}{3}ac}, 2bc)$.

Finally, we note that Theorem 4.6.4 follows from Lemma 4.6.6 and Lemma 4.6.7 with $f(x) = \frac{a}{1-cx}$.

4.7 Deferred Proof for ODE (Section 4.2)

4.7.1 Proof of general k-th order ODE

The goal of this section is to prove Theorem 4.2.3.

Proof. We define $\overline{x}(t) \in \mathbb{R}^{kd}$,

$$\overline{x}(t) = (\overline{x}_1(t), \overline{x}_2(t), \cdots, \overline{x}_k(t)) = \left(c_1 \frac{\mathrm{d}^{k-1}}{\mathrm{d}t^{k-1}} x(t), c_2 \frac{\mathrm{d}^{k-2}}{\mathrm{d}t^{k-2}} x(t), \cdots, c_k x(t) \right),$$

where $\overline{x}_1(t) \in \mathbb{R}^d$, $\overline{x}_2(t) \in \mathbb{R}^d$, $\cdots$, $\overline{x}_k(t) \in \mathbb{R}^d$. We define the norm on $\mathbb{R}^{kd}$ by $\|\overline{x}\| = \sum_{i=1}^k \|\overline{x}_i\|$.

Then we have

$$\frac{\mathrm{d}}{\mathrm{d}t}\overline{x}(t) = \left(c_1 \frac{\mathrm{d}^k}{\mathrm{d}t^k} x(t), c_2 \frac{\mathrm{d}^{k-1}}{\mathrm{d}t^{k-1}} x(t), \cdots, c_k \frac{\mathrm{d}}{\mathrm{d}t} x(t) \right).$$

In order to have $\overline{F}(\overline{x}(t), t) = \frac{\mathrm{d}}{\mathrm{d}t}\overline{x}(t)$, we let

$$\overline{F}(\overline{x}(t), t) = \left(c_1 F\left(c_1^{-1}\overline{x}_1(t), c_2^{-1}\overline{x}_2(t), \cdots, c_k^{-1}\overline{x}_k(t), t \right), c_2 c_1^{-1}\overline{x}_1(t), c_3 c_2^{-1}\overline{x}_2(t), \cdots, c_k c_{k-1}^{-1}\overline{x}_{k-1}(t) \right).$$

Now, we check that indeed

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}t}\overline{x}(t) = & \left(c_1 \frac{\mathrm{d}^k}{\mathrm{d}t^k} x(t), c_2 \frac{\mathrm{d}^{k-1}}{\mathrm{d}t^{k-1}} x(t), \cdots, c_k \frac{\mathrm{d}}{\mathrm{d}t} x(t) \right) \\
= & \left(c_1 F\left(\frac{\mathrm{d}^{k-1}}{\mathrm{d}t^{k-1}} x(t), \frac{\mathrm{d}^{k-2}}{\mathrm{d}t^{k-2}} x(t), \cdots, x(t), t \right), c_2 \frac{\mathrm{d}^{k-1}}{\mathrm{d}t^{k-1}} x(t), \cdots, c_k \frac{\mathrm{d}}{\mathrm{d}t} x(t) \right) \\
= & \left(c_1 F\left(c_1^{-1}\overline{x}_1(t), c_2^{-1}\overline{x}_2(t), \cdots, c_k^{-1}\overline{x}_k(t), t \right), c_2 c_1^{-1}\overline{x}_1(t), c_3 c_2^{-1}\overline{x}_2(t), \cdots, c_k c_{k-1}^{-1}\overline{x}_{k-1}(t) \right) \\
& \overline{F}(\overline{x}(t), t).
\end{aligned}$$

We bound the Lipschitz constant of function $\overline{F}$ by

$$\begin{aligned}\|\overline{F}(y(t),t)-\overline{F}(z(t),t)\| &\le c_1\sum_{i=1}^{k} L_i c_i^{-1}\|y_i(t)-z_i(t)\| + \sum_{i=1}^{k-1} c_{i+1}c_i^{-1}\|y_i(t)-z_i(t)\| \\ &= \sum_{i=1}^{k-1}(c_1 L_i c_i^{-1} + c_{i+1}c_i^{-1})\|y_i(t)-z_i(t)\| + c_1 L_k c_k^{-1}\|y_k(t)-z_k(t)\|.\end{aligned}$$

We choose $c_1 = 1$, $c_i = \sum_{j=i}^{k} L_j^{(i-1)/j} + \frac{1}{\overline{T}^{i-1}}$, $\forall i \in \{2,\cdots,k\}$ where $\overline{T} = 4\gamma_\varphi T$. Then we can calculate for each $i \in [k-1]$,

$$\begin{aligned}c_1 L_i c_i^{-1} + c_{i+1}c_i^{-1} &= \frac{L_i}{\sum_{j=i}^{k} L_j^{(i-1)/j} + \frac{1}{\overline{T}^{i-1}}} + \frac{\sum_{j=i+1}^{k} L_j^{i/j} + \frac{1}{\overline{T}^{i}}}{\sum_{j=i}^{k} L_j^{(i-1)/j} + \frac{1}{\overline{T}^{i-1}}} \\ &= \frac{L_i}{L_i^{(i-1)/i} + \sum_{j=i+1}^{k} L_j^{(i-1)/j} + \frac{1}{\overline{T}^{i-1}}} + \frac{\sum_{j=i+1}^{k} L_j^{i/j} + \frac{1}{\overline{T}^{i}}}{\sum_{j=i}^{k} L_j^{(i-1)/j} + \frac{1}{\overline{T}^{i-1}}} \\ &\le L_i^{1/i} + \sum_{j=i+1}^{k} L_j^{1/j} + \frac{1}{\overline{T}} \\ &= \sum_{j=i}^{k} L_j^{1/j} + \frac{1}{\overline{T}}\end{aligned}$$

For $i = k$, we have

$$c_1 L_k c_k^{-1} = \frac{L_k}{L_k^{(k-1)/k} + \frac{1}{\overline{T}^{k-1}}} \le L_k^{1/k}.$$

Thus,

$$\|\overline{F}(y(t),t)-\overline{F}(z(t),t)\| \le \left(\sum_{j=1}^{k} L_j^{1/j} + \frac{1}{\overline{T}}\right)\cdot\sum_{i=1}^{k}\|y_i(t)-z_i(t)\| = \left(\sum_{j=1}^{k} L_j^{1/j} + \frac{1}{\overline{T}}\right)\|y(t)-z(t)\|.$$

It gives the following Claim:

Claim 4.7.1. *Function $\overline{F}$ has Lipschitz constant $\overline{L} = \sum_{j=1}^{k} L_j^{1/j} + \frac{1}{\overline{T}}$.*

Then, we consider the following first order ODE,

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\overline{x}(t) &= \overline{F}(\overline{x}(t), t),\\ \overline{x}(0) &= (c_1 v_{k-1}, c_2 v_{k-2}, \cdots, c_k v_0)\end{aligned}$$

Let $\overline{x}^*(t) \in \mathbb{R}^{kd}$ denote the optimal solution, then

$$\overline{x}^*(t) = \left(c_1 \frac{\mathrm{d}^{k-1}}{\mathrm{d}t^{k-1}} x^*(t), c_2 \frac{\mathrm{d}^{k-2}}{\mathrm{d}t^{k-2}} x^*(t), \cdots, c_k x^*(t)\right).$$

Now, we prove that $\frac{\mathrm{d}}{\mathrm{d}t}\overline{x}^*(t)$ is approximate by some element in $\mathcal{V}^d$. Let $\overline{q} : \mathbb{R} \to \mathbb{R}^{kd}$ be defined as follows

$$\overline{q}(t) = \left(c_1 q^{(k)}(t), c_2 q^{(k-1)}(t), \cdots, c_k q^{(1)}(t)\right).$$

Then,

$$\begin{aligned}&\left\|\overline{q}(t) - \frac{\mathrm{d}}{\mathrm{d}t}\overline{x}^*(t)\right\|\\ =& \left\|\left(c_1 q^{(k)}(t), c_2 q^{(k-1)}(t), \cdots, c_k q^{(1)}(t)\right) - \left(c_1 \frac{\mathrm{d}^k}{\mathrm{d}t^k} x^*(t), c_2 \frac{\mathrm{d}^{k-1}}{\mathrm{d}t^{k-1}} x^*(t), \cdots, c_k \frac{\mathrm{d}}{\mathrm{d}t} x^*(t)\right)\right\|\\ =& \sum_{i=1}^{k} c_i \left\|q^{(k+1-i)}(t) - \frac{\mathrm{d}^{k+1-i}}{\mathrm{d}t^{k+1-i}} x^*(t)\right\|\\ \leq& \sum_{i=1}^{k} c_i \frac{\epsilon}{\overline{T}^{k+1-i}} = \frac{1}{\overline{T}} \underbrace{\epsilon \sum_{i=1}^{k} \frac{c_i}{\overline{T}^{k-i}}}_{\overline{\epsilon}}.\end{aligned}$$

By the assumption on T, we have that $\gamma_\varphi \overline{L} T \leq 1/2$ and hence theorem 4.2.2 finds

$$\overline{x}^{(N)}(t) = (\overline{x}_1^{(N)}(t), \overline{x}_2^{(N)}(t), \cdots, \overline{x}_k^{(N)}(t)) \in \mathbb{R}^{kd}$$

such that

$$\|\overline{x}^{(N)}(t) - \overline{x}^*(t)\| \leq 20\gamma_\varphi \overline{\epsilon} \tag{4.40}$$

This implies that

$$\left\| c_i^{-1}\overline{x}^{(N)}(t) - \frac{\mathrm{d}^{k-i}}{\mathrm{d}t^{k-i}} x^*(t) \right\| \leq 20 c_i^{-1}\gamma_\varphi \overline{\epsilon}, \forall i \in [k]. \tag{4.41}$$

To bound the last term, we show the following Claim:

Claim 4.7.2. *Let* $\overline{\epsilon} = \epsilon \sum_{i=1}^{k} \frac{c_i}{\overline{T}^{k-i}}$. *If we choose* $c_1 = 1$ *and* $c_i = \sum_{j=i}^{k} L_j^{(i-1)/j} + \frac{1}{\overline{T}^{i-1}}$, $\forall i \in \{2, \cdots, k\}$, *then*

$$c_i^{-1}\overline{\epsilon} \leq \frac{\epsilon}{\overline{T}^{k-i}}(2k+1).$$

Proof. For each $i \in [k]$,

$$c_i^{-1}\overline{\epsilon} = c_i^{-1}\epsilon \sum_{j=1}^{k} \frac{c_j}{\overline{T}^{k-j}} = \frac{\epsilon}{\overline{T}^{k-i}} \sum_{j=1}^{k} \frac{c_j c_i^{-1}}{\overline{T}^{i-j}}$$

We can lower bound the term $\frac{c_j c_i^{-1}}{\overline{T}^{i-j}}$ as follows,

$$\begin{aligned}
\frac{c_j c_i^{-1}}{\overline{T}^{i-j}} &= \frac{\sum_{l=j}^{k} L_l^{(j-1)/l} + \frac{1}{\overline{T}^{j-1}}}{\overline{T}^{i-j}(\sum_{l=i}^{k} L_l^{(i-1)/l} + \frac{1}{\overline{T}^{i-1}})} \\
&= \frac{\sum_{l=j}^{k} L_l^{(j-1)/l} + \frac{1}{\overline{T}^{j-1}}}{\sum_{l=i}^{k} \overline{T}^{i-j} L_l^{(i-1)/l} + \frac{1}{\overline{T}^{j-1}}} \\
&\leq \frac{\sum_{l=j}^{k} L_l^{(j-1)/l} + \frac{1}{\overline{T}^{j-1}}}{\frac{1}{\overline{T}^{j-1}}} \\
&= 1 + \sum_{l=j}^{k} (L_l^{1/l}\overline{T})^{j-1}
\end{aligned}$$

Therefore, we have

$$\begin{aligned} c_i^{-1}\overline{\epsilon} &= \frac{\epsilon}{\overline{T}^{k-i}} \sum_{j=1}^{k} \frac{c_j c_i^{-1}}{\overline{T}^{i-j}} \\ &\leq \frac{\epsilon}{\overline{T}^{k-i}} \sum_{j=1}^{k} \left(1 + \sum_{l=j}^{k} (L_l^{1/l}\overline{T})^{j-1} \right) \\ &\leq \frac{\epsilon}{\overline{T}^{k-i}} \left(2k + \sum_{j=1}^{k} \left(\sum_{l=1}^{k} L_l^{1/l}\overline{T} \right)^j \right) \\ &\leq \frac{\epsilon}{\overline{T}^{k-i}} \left(2k + \sum_{j=1}^{k} (1/2)^j \right) \end{aligned}$$

where we used $\gamma_\varphi LT \leq 1/8$ at the end. Thus we complete the proof of Claim. □

Now, using the claim to (4.41), we have the error is

$$20(2k+1)\gamma_\varphi \frac{\epsilon}{\overline{T}^{k-i}} \leq 20(2k+1)\gamma_\varphi \frac{\epsilon}{T^{k-i}}.$$

To bound the number of iterations needed in the log term in Theorem 4.2.2, we note that

$$\begin{aligned} \int_0^T \|\overline{F}(\overline{x}(0), s)\| ds &= c_1 \left\| \int_0^T F(c_1^{-1}\overline{x}_1(0), c_2^{-1}\overline{x}_2(0), \cdots, c_k^{-1}\overline{x}_k(0), s) ds \right\| + T \cdot \sum_{i=1}^{k-1} c_{i+1} c_i^{-1} \|\overline{x}_i(t)\| \\ &= \left\| \int_0^T F(\frac{d^{k-1}}{dt^{k-1}}x(0), \frac{d^{k-2}}{dt^{k-2}}x(0), \cdots, x(0), s) ds \right\| + T \cdot \sum_{i=1}^{k-1} c_{i+1} \left\| \frac{d^{k-i}}{dt^{k-i}}x(0) \right\|. \end{aligned}$$

Note that

$$c_{i+1} \leq \sum_{j=1}^{k} L_j^{i/j} + \frac{1}{\overline{T}^i} \leq L^i + \frac{1}{\overline{T}^i} \leq \frac{2}{\overline{T}^i}$$

where we used $L\overline{T} \leq \frac{1}{2}$. Hence, the number of iterations we need is

$$O\left(D\log\left(\frac{1}{\overline{\epsilon}}\left(\left\|\int_0^T F(v_{k-1}, v_{k-2}, \cdots, v_0, s)ds\right\| + \sum_{i=1}^{k-1}\frac{\|v_i\|}{\overline{T}^{k-i-1}}\right)\right)\right)$$
$$= O\left(D\log\left(\frac{1}{\epsilon}\left(\overline{T}^{k-1}\cdot\left\|\int_0^T F(v_{k-1}, v_{k-2}, \cdots, v_0, s)ds\right\| + \sum_{i=1}^{k-1}\overline{T}^{i}\|v_i\|\right)\right)\right)$$

where we used $\overline{\epsilon} \geq \epsilon \cdot c_k \geq \frac{\epsilon}{\overline{T}^{k-1}}$. □

4.8 Deferred Proof for Cauchy Estimates (Section 4.6.3)

In this section, we provide the proofs of some core Lemmas/Claims used for proving Theorem 4.6.4.

Lemma 4.6.6. (Bounding multivariate ODE by scalar ODE). Given vectors $v_1, v_0 \in \mathbb{R}^d$ and any norm $\|\cdot\|$ on $\mathbb{R}^d$. Let $U \subset \mathbb{R}^d$ be a neighborhood of v_0 and that $F : U \to \mathbb{R}^d$ is a real analytic near v_0. Suppose that

$$\|D^{(k)}F(v_0)[\Delta_1, \Delta_2, \cdots, \Delta_k]\| \leq f^{(k)}(0) \prod_{j=1}^{k} \|\Delta_j\| \text{ for all } k \geq 0$$

for some real analytic function f around 0. Then the ODE (4.39) has a unique real analytic solution around $t = 0$. Furthermore, for any $b \geq \|v_1\|$, we have

$$\|u^{(k)}(0)\| \leq \psi^{(k)}(0) \quad \forall k \geq 1$$

where ψ is the solution of the ODE

$$\psi''(t) = f(\psi(t)), \psi'(0) = b, \psi(0) = 0.$$

Proof. Theorem 4.6.5 shows that the solution u uniquely exists and is real analytic around 0. Therefore, we can take derivatives on both sides of $u''(t) = F(u(t))$ and get

$$\begin{aligned}
u^{(3)}(t) &= DF(u(t))[u^{(1)}(t)], \\
u^{(4)}(t) &= DF(u(t))[u^{(2)}(t)] + D^2F(u(t))[u^{(1)}(t), u^{(1)}(t)] \\
u^{(5)}(t) &= DF(u(t))[u^{(3)}(t)] + 2D^2F(u(t))[u^{(2)}(t), u^{(1)}(t)] + D^3F(u(t))[u^{(1)}(t), u^{(1)}(t), u^{(1)}(t)] \\
&\vdots = \vdots
\end{aligned}$$

Therefore, we have

$$\begin{aligned}
\|u^{(3)}(0)\| &= \|DF(u(0))[u^{(1)}(0)]\| \\
&\leq f^{(1)}(0)\|u^{(1)}(0)\| \\
\|u^{(4)}(0)\| &= \|DF(u(0))[u^{(2)}(0)] + D^2F(u(0))[u^{(1)}(0), u^{(1)}(0)]\| \\
&\leq \|DF(u(0))[u^{(2)}(0)]\| + \|D^2F(u(0))[u^{(1)}(0), u^{(1)}(0)]\| \\
&\leq f^{(1)}(0)\|u^{(2)}(0)\| + f^{(2)}(0)\|u^{(1)}(0)\|^2 \\
\|u^{(5)}(0)\| &= \|DF(u(0))[u^{(3)}(0)] + 2D^2F(u(0))[u^{(2)}(0), u^{(1)}(0)] \\
&\quad + D^3F(u(0))[u^{(1)}(0), u^{(1)}(0), u^{(1)}(0)]\| \\
&\leq \|DF(u(0))[u^{(3)}(0)]\| + \|2D^2F(u(0))[u^{(2)}(0), u^{(1)}(0)]\| \\
&\quad + \|D^3F(u(0))[u^{(1)}(0), u^{(1)}(0), u^{(1)}(0)]\| \\
&\leq f^{(1)}(0)\|u^{(3)}(0)\| + 2f^{(2)}(0)\|u^{(2)}(0)\|\|u^{(1)}(0)\| + f^{(3)}(0)\|u^{(1)}(0)\|^3 \\
&\vdots = \vdots
\end{aligned}$$

Similarly, Theorem 4.6.5 shows that the solution ψ uniquely exists and is real analytic around 0. By expanding $\psi''(t) = f(\psi(t))$ at $t = 0$, we see that

$$\begin{aligned}
\psi^{(3)}(0) &= f^{(1)}(0)\psi^{(1)}(0), \\
\psi^{(4)}(0) &= f^{(1)}(0)\psi^{(2)}(0) + f^{(2)}(0)(\psi^{(1)}(0))^2, \\
\psi^{(5)}(0) &= f^{(1)}(0)\psi^{(3)}(0) + 2f^{(2)}\psi^{(2)}(0)\psi^{(1)}(0) + f^{(3)}(0)(\psi^{(1)}(0))^3, \\
&\vdots = \vdots
\end{aligned}$$

Since $\|u^{(1)}(0)\| \leq b = \psi^{(1)}(0)$, $\|u^{(2)}(0)\| = \|F(u(0))\| \leq f(0) = \psi^{(2)}(0)$.

For $k = 3, 4, 5, \cdots$, we have

$$\begin{aligned}
\|u^{(3)}(0)\| &\leq f^{(1)}(0)\|u^{(1)}(0)\| = f^{(1)}(0)\psi^{(1)}(0) = \psi^{(3)}(0) \\
\|u^{(4)}(0)\| &\leq \psi^{(4)}(0) \\
\|u^{(5)}(0)\| &\leq \psi^{(5)}(0) \\
\vdots &\leq \vdots
\end{aligned}$$

Thus, we have that $\|u^{(k)}(0)\| \leq \psi^{(k)}(0)$ for all $k \geq 1$.

□

Lemma 4.6.7. (Bounding scalar ODE) Let $f(x) = \frac{a}{1-cx}$ with positive a and c. Let $\psi(t)$ denote the solution of the ODE

$$\psi''(t) = f(\psi(t)), \psi'(0) = b, \psi(0) = 0$$

with $b \geq 0$. Then,

$$\psi^{(k)}(0) \leq \frac{k!\alpha^k}{c} \quad \forall k \geq 1$$

with $\alpha = \max(\sqrt{\frac{4}{3}ac}, 2bc)$.

Proof. Let $\widetilde{\psi}(t) = \frac{1}{c}\left(1 - \sqrt{1 - \alpha t}\right)$ with $\alpha = \max(\sqrt{\frac{4}{3}ac}, 2bc)$. Note that

$$\widetilde{\psi}''(t) = \frac{\alpha^2}{4c(1-\alpha t)^{3/2}} = \widetilde{f}(\widetilde{\psi}(t)) \quad \text{with} \quad \widetilde{f}(x) = \frac{\alpha^2}{4c(1-cx)^3}.$$

Since $\alpha^2 \geq \frac{4}{3}ac$, we have that

$$\widetilde{f}^{(k)}(0) = \frac{a}{3} \cdot \frac{(k+2)!}{2} c^k \geq k! \cdot a \cdot c^k = f^{(k)}(0).$$

Also, we have that $\widetilde{\psi}(0) = 0$ and $\widetilde{\psi}'(0) = \frac{\alpha}{2c} \geq b$. Hence, Lemma 4.6.6 shows that

$$\psi^{(k)}(0) \leq \widetilde{\psi}^{(k)}(0) = \frac{\alpha^k}{c} \prod_{i=1}^{k} \frac{|2i-3|}{2} \leq \frac{k!\alpha^k}{c}$$

for all $k \geq 1$. □

4.9 Cauchy Estimates of Some Functions

We first state a useful tool,

Lemma 4.9.1. *Let $f : U \to \mathbb{C}$ be holomorphic and suppose that $D_R = \{z \ : \ |z - z_0| \leq R\} \subset U$. Let $\gamma_R = \{z \ : \ |z - z_0| = R\}$ denote the boundary of D_R. For all $n \geq 0$*

$$|f^{(n)}(z_0)| \leq \frac{n!}{R^n} \max_{z \in \gamma_R} |f(z)|.$$

4.9.1 Logistic loss function

Lemma 4.9.2 (Property of Logistic loss function)**.** *Let $\phi(t) = \log(1 + e^{-t})$, then we know that $\phi'(t)$ has Cauchy estimate $M = 1$ with radius $r = 1$.*

Proof. Given the definition of $\phi(t)$, it is easy to see that

$$\phi'(t) = \frac{-1}{1 + e^t}.$$

Let $f(z) = \frac{1}{1+e^z}$. Let $z = a + b\mathbf{i}$. We have

$$|f(z)| = \left| \frac{1}{1 + e^a \cos b + \mathbf{i} e^a \sin b} \right| = \frac{1}{\sqrt{(1 + e^a \cos b)^2 + (e^a \sin b)^2}} = \frac{1}{\sqrt{1 + 2e^a \cos b + e^{2a}}}$$

Let $z_0 = a_0 + \mathbf{i} b_0$. We choose $R = 1$, then $(a - a_0)^2 + (b - b_0)^2 = 1$. Since we only care about real numbers, we have $b_0 = 0$. Then we know that $b \in [-1, 1]$, which means $\cos b \in [0.54, 1]$. Thus, we have,

$$\sqrt{1 + 2e^a \cos b + e^{2a}} \geq \sqrt{1 + e^a + e^{2a}} \geq 1.$$

Thus, $\left|\frac{1}{1+e^z}\right| \leq 1$. Therefore, using Lemma 4.9.1, for all real z_0,

$$|f^{(n)}(z_0)| \leq n!$$

□

4.9.2 Pseudo-Huber loss function

Huber function [Hub64] has been extensively studied in a large number of algorithmic questions, e.g. regression [CW15b, CW15a], low-rank approximation [SWZ19b], clustering [BFL16], sparse recovery [NSW19a]. Formally speaking, the Huber loss function can be defined as

$$f(x) = \begin{cases} \frac{x^2}{2\delta}, & \text{if } |x| \leq \delta; \\ |x| - \delta/2, & \text{otherwise .} \end{cases}$$

where $\delta > 0$ is a parameter. For many applications, the Pseudo-Huber loss function can be used as a smooth alternative for the Huber loss function.

Lemma 4.9.3 (Property of Pseudo-Huber function)**.** *Fix any $\delta > 0$. Let $\phi(x) = \sqrt{x^2 + \delta^2} - \delta$, then we know that $\phi'(x)$ has Cauchy estimate $M = 1$ with radius $r = \delta/2$.*

Proof. Given the definition of function $\phi(x)$, it is easy to see that

$$\phi'(x) = \frac{x}{\sqrt{x^2 + \delta^2}}$$

Let $f(z) = \frac{z}{\sqrt{z^2+\delta^2}}$. Let $z = a + b\mathbf{i}$. We have

$$|f(z)| = \left| \frac{(a+b\mathbf{i})}{\sqrt{(a+b\mathbf{i})^2 + \delta^2}} \right| = \left| \frac{(a+b\mathbf{i})}{\sqrt{a^2 - b^2 + \delta^2 + 2ab\mathbf{i}}} \right| = \frac{|(a+b\mathbf{i})|}{|\sqrt{a^2 - b^2 + \delta^2 + 2ab\mathbf{i}}|}$$

For the numerator, we have $|(a+b\mathbf{i})| = \sqrt{a^2 + b^2}$. For the denominator, we have

$$|\sqrt{a^2 - b^2 + \delta^2 + 2ab\mathbf{i}}| = \left((a^2 - b^2 + \delta^2)^2 + 4(ab)^2\right)^{1/4}$$

Let $z_0 = a_0 + \mathbf{i}b_0$. We choose $R = \frac{\delta}{2}$, then $(a - a_0)^2 + (b - b_0)^2 = (\delta/2)^2$. Since we are only real z_0, we have $b_0 = 0$. Then we know that $b \in [-\frac{\delta}{2}, \frac{\delta}{2}]$. Thus

$$|f(z)| = \frac{\sqrt{a^2 + b^2}}{((a^2 - b^2 + \delta^2)^2 + 4(ab)^4)^{1/4}} \leq \frac{\sqrt{a^2 + \frac{\delta^2}{4}}}{\left((a^2 - \frac{\delta^2}{4} + \delta^2)^2 + 0\right)^{1/4}} = \frac{\sqrt{a^2 + \frac{\delta^2}{4}}}{\sqrt{a^2 + \frac{3\delta^2}{4}}} \leq 1.$$

Therefore, using Lemma 4.9.1, for all real z_0,

$$|f^{(n)}(z_0)| \leq n! \cdot (2/\delta)^n.$$

□

Chapter 5

Convergence result of Deep Neural Network

Deep neural networks (DNNs) have demonstrated dominating performance in many fields; since AlexNet, networks used in practice are going wider and deeper. On the theoretical side, a long line of works has been focusing on training neural networks with one hidden layer. The theory of multi-layer networks remains largely unsettled.

In this work, we prove why stochastic gradient descent (SGD) can find *global minima* on the training objective of DNNs in *polynomial time*. We only make two assumptions: the inputs are non-degenerate and the network is over-parameterized. The latter means the network width is sufficiently large: *polynomial* in L, the number of layers and in n, the number of samples.

Our key technique is to derive that, in a sufficiently large neighborhood of the random initialization, the optimization landscape is almost-convex and semi-smooth even with ReLU activations. This implies an equivalence between over-parameterized neural networks and neural tangent kernel (NTK) in the finite (and polynomial) width setting.

As concrete examples, starting from randomly initialized weights, we prove that SGD can attain 100% training accuracy in classification tasks, or minimize regression loss in linear convergence speed, with running time polynomial in n, L. Our theory applies to the widely-used but non-smooth ReLU activation, and to any smooth and possibly non-

convex loss functions. In terms of network architectures, our theory at least applies to fully-connected neural networks, convolutional neural networks (CNN), and residual neural networks (ResNet).

5.1 Introduction

Neural networks have demonstrated a great success in numerous machine-learning tasks [KSH12, GMH13, LHP+15, AAA+16, HZRS16, SHM+16, SSS+17]. One of the empirical findings is that neural networks, trained by first-order methods from random initialization, have a remarkable ability to fit training data [ZBH+17].

From an *expressibility* perspective, this may not be surprising since modern neural networks are often over-parameterized: they have much more parameters than the number of training samples. There certainly *exist* parameter choices with zero training error as long as data is non-degenerate.

Yet, from an optimization perspective, the fact that randomly-initialized first-order methods can find global minima on the training data is *quite non-trivial*: neural networks are often equipped with the ReLU activation, making the training objective not only non-convex, but even non-smooth. Even the general convergence for finding approximate critical points of a non-convex, non-smooth function is not fully-understood [BLO05] and appears to be a challenging question on its own. This is in direct contrast to practice, in which ReLU networks trained by stochastic gradient descent (SGD) from random initialization *almost never* suffer from non-smoothness or non-convexity, and can avoid local minima for a variety of network architectures (see [GVS15]). A theoretical justification was missing to explain this phenomenon.

There are quite a few papers trying to understand the success of neural networks from optimization perspective. Many of them focus on the case when the inputs are random Gaussian, and work only for two-layer neural networks [BG17, Son17, Tia17, LY17, DLT+18,

GLM17, PRSZ18, ZSJ+17, ZSD17]. [LL18] show that for a two-layer network with ReLU activation, SGD finds nearly-global optimal (say, 99% classification accuracy) solutions on the training data, as long as the network is *over-parameterized*, meaning that the number of neurons is polynomially large comparing to the input size. Moreover, if the data is sufficiently structured (say, coming from mixtures of separable distributions), this accuracy extends also to *test data*. As a separate note, over-parameterization is suggested as the possible key to avoid bad local minima by [SS18] even for two-layer networks.

There are also results that go beyond two-layer networks with limitations. Some consider deep *linear* neural networks without any activation functions [HM17, ACGH18, BHL18, Kaw16]. [Dal17] studies multi-layer neural networks but essentially only with respect to the *convex* task of training the last layer.[1] [SC16] show that under over-parameterization and under random input perturbation, there is bad local minima for multi-layer neural networks. [JGH18] derive global convergence using neural tangent kernel for *infinite-width* neural networks.

In this paper, we study the following fundamental questions

Can DNN be trained close to zero training error efficiently under mild assumptions?

If so, can the running time depend only polynomially *in the network depth and input size?*

Motivation In 2012, AlexNet was born with 5 convolutional layers [KSH12]. The later VGG network uses 19 layers [SSZ14], and GoogleNet uses 22 layers [SLJ+15]. In practice, we

[1] [Dal17] works in a parameter regime where the weight changes of all layers except the last one make negligible contribution to the final output.

cannot go deeper by naively stacking layers together, due to the so-called vanishing/exploding gradient problem. To deal with this issue, networks with residual links (ResNet) were proposed with the capability of handling at least 152 layers [HZRS16]. Compared with practical networks that go much deeper, existing theory has been mostly around two-layer (thus one-hidden-layer) neural networks, even just for the training process alone. Thus,

Can we theoretically justify how the training process has worked for multi-layer neural networks?

In this paper, we extend the over-parameterization theory to *multi-layer* neural networks.

5.1.1 Our Result

We show that over-parameterized neural networks can be trained by vanilla first-order methods such as gradient descent (GD) or stochastic gradient descent (SGD) to global minima (e.g. *zero* training error), as long as the data is non-degenerate.

We say that the data is non-degenerate if every pairs of samples are distinct. This is a minimal requirement since a dataset with two identical data points of different labels cannot be trained to zero error. We denote by δ the minimum (relative) distance between two data points, and by n the number of training samples. Now, consider an L-layer fully-connected feedforward neural network, each hidden layer consisting of m neurons equipped with ReLU activation. We show that,

- As long as $m \geq \text{poly}(n, L, \delta^{-1})$, starting from random Gaussian initialization, GD/SGD finds an ϵ-error global minimum in ℓ_2 regression using at most $T = \text{poly}(n, L, \delta^{-1}) \log \frac{1}{\epsilon}$ iterations.

- If the task is multi-label classification, then GD/SGD finds an 100% accuracy classifier on the training set in $T = \text{poly}(n, L, \delta^{-1})$ iterations.

- Our result also applies to other Lipschitz-smooth loss functions, and some other network architectures including convolutional neural networks (CNNs) and residual networks (ResNet).

In contrast, prior work on this task either requires m and T to grow in $e^{O(L)}$ (and essentially only the last layer is trained) [Dal17]; or requires $m = \infty$ [JGH18].

Our Contributions We summarize our technical contributions below.

- For a sufficiently large neighborhood of the random initialization, we prove that the training landscape is almost convex and semi-smooth. This somewhat explains the empirical finding by [GVS15] that GD/SGD will not be trapped in local minima. (See Section 5.4.1.)

- For a sufficiently large neighborhood of the random initialization, we derive an equivalence between neural networks and the neural tangent kernel (NTK) introduced by [JGH18]. Unlike the prior work in which they show the equivalence only for infinite-width networks (i.e., $m = \infty$), here we only need $m = \text{poly}(L)$ for such an equivalence to hold. (See Section 5.4.2.)

- We show that equipped with ReLU activation, neural networks do not suffer from exponential gradient explosion or vanishing. This is the key reason we can avoid exponential dependency on L. If one is okay with $e^{O(L)}$ dependency, many proofs shall become trivial. (See Section 5.5.)

- We derive a stability theory of neural networks against small but adversarial perturbations that may be of independent interests. Previous results on this topic either have exponential blowup in L [Dal17] or requires the width to go to infinity [JGH18]. (See Section 5.5.)

- We derive our results by training only hidden layers. This can be more meaningful than training all the layers together, in which if one is not careful with parameter choices, the training process can degenerate as if only the last layer is trained [Dal17].

That is a convex task and may not reflect the true power of deep learning. (Of course, as a simple corollary, our results also apply to training all the layers together.)

Finally, we emphasize that this present paper as a deeply-simplified version of the recurrent neural network (RNN) paper [AZLS18] by the same set of authors. To some extent, DNN is a "special case" of RNN,[2] thus most of the technical tools were already developed in [AZLS18]. We write this DNN result as a separate paper because: (1) not all the readers can easily derive the DNN result from [AZLS18]; (2) the convergence of DNN can be important on its own; (3) the proof in this paper is much simpler (30 vs 80 pages) and could reach out to a wider audience; (4) the simplicity of this paper allows us to tighten parameters in some non-trivial ways; and (5) the simplicity of this paper allows us to also study convolutional networks, residual networks, as well as different loss functions (all of them were missing from [AZLS18]). We also note that the techniques of this paper can be combined with [AZLS18] to show the global convergence of training over-parameterized *deep* RNN. We ignore the details so as not to complicate this paper.

Towards Generalization In practice, deeper and wider neural networks generalize better [SGS15, ZK16], so what can we say in theory? Although this paper does not explicitly cover generalization to test data, since a neural network in our parameter regime simulates

[2] A recurrent neural network executed on input sequences with time horizon L is very similar to a feedforward neural network with L layers. The main difference is that in a feedforward network, weight matrices are different across layers, and thus independently randomly initialized; in contrast, in an RNN, the same weight matrix is applied across the entire time horizon, so we do not have fresh new randomness for proofs that involve in induction. In other words, the over-parameterized convergence theory of DNN is *much simpler* than that of RNN.

its neural tangent kernel (NTK), it is clear that neural networks provide generalization at least *as good as its NTK*.

In the PAC-learning language, one may study generalization with respect to *concept classes*. Follow-up work [ALL18c] shows that three-layer over-parameterized ReLU networks can efficiently (in polynomial time and sample complexity) learn the concept class of three-layer neural networks with smooth activations [ALL18c], and the follow-up work [AL19b] shows stronger results for three-layer ResNet.

It is worth pointing out that the three-layer result [ALL18c] goes beyond the almost-convex regime and thus is not captured by its NTK; more interestingly, the three-layer ResNet result [AL19b] is not achievable (in a provable sense) by any kernel method including any NTK.

A concurrent, independent and beautiful result We acknowledge a concurrent work [DLL+19] by Du, Lee, Li, Wang and Zhai. They proved a similar result.

5.1.2 Other Related Works

[LL18] originally prove their result for the cross-entropy loss, together with some test accuracy guarantee. (If data is "well-structured", they prove two-layer over-parameterized neural networks can learn it using SGD with polynomially many samples [LL18].) Later, the "training accuracy" part of [LL18] was extended to the ℓ_2 loss [DZPS19]. From another perspective, [DZPS19] focus on gradient descent.

Linear networks without activation functions are important subjects on its own. Besides the already cited references [HM17, ACGH18, BHL18, Kaw16], there are a number of works that study *linear dynamical systems*, which can be viewed as the linear version of recurrent neural networks or reinforcement learning. Recent works in this line of research include [HMR18, HSZ17, HLS+18, DMM+17, OO18, AAMM18, SMT+18, MT18, DTMR18, AHL+18].

There is sequence of work about one-hidden-layer (multiple neurons) CNN [BG17, ZSD17, DLT+18, GKM18, Oym18]. Whether the patches overlap or not plays a crucial role in analyzing algorithms for such CNN. One category of the results have required the patches to be disjoint [BG17, ZSD17, DLT+18]. The other category [GKM18, Oym18] have figured out a weaker assumption or even removed that patch-disjoint assumption. On input data distribution, most relied on inputs being Gaussian [BG17, ZSD17, DLT+18, Oym18], and some assumed inputs to be symmetrically distributed with identity covariance and boundedness [GKM18].

As for ResNet, [LY17] proved that SGD learns one-hidden-layer residual neural networks under Gaussian input assumption. The techniques in [ZSJ+17, ZSD17] can also be

generalized to one-hidden-layer ResNet under the Gaussian input assumption; they can show that GD starting from good initialization point (via tensor initialization) learns ResNet. [HM17] deep *linear* residual networks have no spurious local optima.

If no assumption is allowed, neural networks have been shown hard in several different perspectives. Thirty years ago, [BR93] first proved that learning the neural network is NP-complete. Stronger hardness results have been proved over the last decade [KS09, LSSS14, Dan16, DSS16, GKKT17, SVWX17, MR18].

5.2 Preliminaries

We use $\mathcal{N}(\mu, \sigma)$ to denote the Gaussian distribution of mean μ and variance σ; and $\mathcal{B}(m, \frac{1}{2})$ to denote the binomial distribution with m trials and $1/2$ success rate. We use $\|v\|_2$ or $\|v\|$ to denote Euclidean norms of vectors v, and $\|M\|_2, \|M\|_F$ to denote spectral and Frobenius norms of matrices M. For a tuple $\overrightarrow{W} = (W_1, \ldots, W_L)$ of matrices, we let $\|\overrightarrow{W}\|_2 = \max_{\ell \in [L]} \|W_\ell\|_2$ and $\|\overrightarrow{W}\|_F = (\sum_{\ell=1}^{L} \|W_\ell\|_F^2)^{1/2}$.

We use $\phi(x) = \max\{0, x\}$ to denote the ReLU function, and extend it to vectors $v \in \mathbb{R}^m$ by letting $\phi(v) = (\phi(v_1), \ldots, \phi(v_m))$. We use $\mathbb{1}_{event}$ to denote the indicator function for *event*.

The training data consist of vector pairs $\{(x_i, y_i^*)\}_{i \in [n]}$, where each $x_i \in \mathbb{R}^{\mathfrak{d}}$ is the feature vector and y_i^* is the label of the i-th training sample. We assume *without loss of generality* that data are normalized so that $\|x_i\| = 1$ and its last coordinate $(x_i)_{\mathfrak{d}} = \frac{1}{\sqrt{2}}$.[3] We also assume $\|y_i^*\| \leq O(1)$ for notation simplicity.[4]

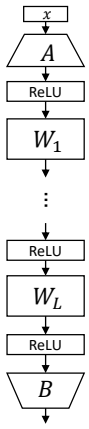
We make the following separable assumption on the training data (motivated by [LL18]):

Assumption 5.2.1. *For every pair $i, j \in [n]$, we have $\|x_i - x_j\| \geq \delta$.*

[3] Without loss of generality, one can re-scale and assume $\|x_i\| \leq 1/\sqrt{2}$ for every $i \in [n]$. Again, without loss of generality, one can pad each x_i by an additional coordinate to ensure $\|x_i\| = 1/\sqrt{2}$. Finally, without loss of generality, one can pad each x_i by an additional coordinate $\frac{1}{\sqrt{2}}$ to ensure $\|x_i\| = 1$. This last coordinate $\frac{1}{\sqrt{2}}$ is equivalent to introducing a (random) bias term, because $A(\frac{y}{\sqrt{2}}, \frac{1}{\sqrt{2}}) = \frac{A}{\sqrt{2}}(y, 0) + b$ where $b \sim \mathcal{N}(0, \frac{1}{m} I)$. In our proofs, the specific constant $\frac{1}{\sqrt{2}}$ does not matter.

[4] If $\|y_i^*\| \leq \Omega$ for some parameter Ω, our complexities shall also grow in poly(Ω).

To present the simplest possible proof, the main body of this paper only focuses on depth-L feedforward fully-connected neural networks with an ℓ_2-regression task. Therefore, each $y_i^* \in \mathbb{R}^d$ is a target vector for the regression task. We explain how to extend it to more general settings in Section 5.6 and the Appendix. For notational simplicity, we assume all the hidden layers have the same number of neurons, and our results trivially generalize to each layer having different number of neurons. Specifically, we focus on the following network



$$
\begin{aligned}
g_{i,0} &= Ax_i & h_{i,0} &= \phi(Ax_i) & &\text{for } i \in [n] \\
g_{i,\ell} &= W_\ell h_{i,\ell-1} & h_{i,\ell} &= \phi(W_\ell h_{i,\ell-1}) & &\text{for } i \in [n], \ell \in [L] \\
y_i &= Bh_{i,L} & & & &\text{for } i \in [n]
\end{aligned}
$$

where $A \in \mathbb{R}^{m \times \mathfrak{d}}$ is the weight matrix for the input layer, $W_\ell \in \mathbb{R}^{m \times m}$ is the weight matrix for the ℓ-th hidden layer, and $B \in \mathbb{R}^{d \times m}$ is the weight matrix for the output layer. For notational convenience in the proofs, we may also use $h_{i,-1}$ to denote x_i and W_0 to denote A.

Definition 5.2.1 (diagonal sign matrix)**.** For each $i \in [n]$ and $\ell \in \{0, 1, \ldots, L\}$, we denote by $D_{i,\ell}$ the diagonal sign matrix where $(D_{i,\ell})_{k,k} = \mathbb{1}_{(W_\ell h_{i,\ell-1})_k \geq 0}$ for each $k \in [m]$.

As a result, we have $h_{i,\ell} = D_{i,\ell} W_\ell h_{i,\ell-1} = D_{i,\ell} g_{i,\ell}$ and $(D_{i,\ell})_{k,k} = \mathbb{1}_{(g_{i,\ell})_k \geq 0}$. We make the following standard choices of random initialization:

Definition 5.2.2. We say that $\overrightarrow{W} = (W_1, \ldots, W_L)$, A and B are at random initialization if

- $[W_\ell]_{i,j} \sim \mathcal{N}(0, \frac{2}{m})$ for every $i, j \in [m]$ and $\ell \in [L]$;

- $A_{i,j} \sim \mathcal{N}(0, \frac{2}{m})$ for every $(i,j) \in [m] \times [\mathfrak{d}]$; and
- $B_{i,j} \sim \mathcal{N}(0, \frac{1}{d})$ for every $(i,j) \in [d] \times [m]$.

Assumption 5.2.2. *Throughout this paper we assume $m \geq \Omega\big(\operatorname{poly}(n, L, \delta^{-1}) \cdot d\big)$ for some sufficiently large polynomial. To present the simplest proof, we did not try to improve such polynomial factors. We will also assume $\delta \leq O(\frac{1}{L})$ for notation simplicity.*

5.2.1 Objective and Gradient

Our regression objective is

$$F(\overrightarrow{W}) \stackrel{\text{def}}{=} \sum_{i=1}^{n} F_i(\overrightarrow{W}) \quad \text{where} \quad F_i(\overrightarrow{W}) \stackrel{\text{def}}{=} \frac{1}{2}\|Bh_{i,L} - y_i^*\|^2 \quad \text{for each } i \in [n]$$

We also denote by $\mathsf{loss}_i \stackrel{\text{def}}{=} Bh_{i,L} - y_i^*$ the *loss vector* for sample i. For simplicity, we focus on training only hidden weights $\overrightarrow{W}$ in this paper and leave A and B at random initialization. Our result naturally extends to the case when A, B and $\overrightarrow{W}$ are jointly trained.[5]

Definition 5.2.3. For each $\ell \in \{1, 2, \cdots, L\}$, we define $\mathsf{Back}_{i,\ell} \stackrel{\text{def}}{=} BD_{i,L}W_L \cdots D_{i,\ell}W_\ell \in \mathbb{R}^{d\times m}$ and for $\ell = L+1$, we define $\mathsf{Back}_{i,\ell} = B \in \mathbb{R}^{d\times m}$.

Using this notation, one can calculate the gradient of $F(\overrightarrow{W})$ as follows.

Fact 5.2.3. *The gradient with respect to the k-th row of $W_\ell \in \mathbb{R}^{m\times m}$ is*

$$\nabla_{[W_\ell]_k} F(\overrightarrow{W}) = \textstyle\sum_{i=1}^n (\mathsf{Back}_{i,\ell+1}^\top \mathsf{loss}_i)_k \cdot h_{i,\ell-1} \cdot \mathbb{1}_{\langle [W_\ell]_k, h_{i,\ell-1}\rangle \geq 0}$$

The gradient with respect to W_ℓ is

$$\nabla_{W_\ell} F(\overrightarrow{W}) = \textstyle\sum_{i=1}^n D_{i,\ell}(\mathsf{Back}_{i,\ell+1}^\top \mathsf{loss}_i) h_{i,\ell-1}^\top$$

We denote by $\nabla F(\overrightarrow{W}) = \big(\nabla_{W_1} F(\overrightarrow{W}), \ldots, \nabla_{W_L} F(\overrightarrow{W})\big)$.

[5] We note that if one jointly trains all the layers, in certain parameter regimes, it may be equivalent to as if only the last layer is trained [Dal17]. We therefore choose to fix the last layer B to avoid such confusion.

5.3 Our Results and Techniques

To present our result in the simplest possible way, we choose to mainly focus on fully-connected L-layer neural networks with the ℓ_2 regression loss. We shall extend it to more general settings (such as convolutional and residual networks and other losses) in Section 5.6. Our main results can be stated as follows:

Theorem 5.3.1 (gradient descent)**.** *Suppose* $m \geq \widetilde{\Omega}\big(\operatorname{poly}(n, L, \delta^{-1}) \cdot d\big)$*. Starting from random initialization, with probability at least* $1 - e^{-\Omega(\log^2 m)}$*, gradient descent with learning rate* $\eta = \Theta\big(\frac{d\delta}{\operatorname{poly}(n,L)\cdot m}\big)$ *finds a point* $F(\overrightarrow{W}) \leq \epsilon$ *in* $T = \Theta\big(\frac{\operatorname{poly}(n,L)}{\delta^2} \cdot \log \epsilon^{-1}\big)$ *iterations.*

This is known as the linear convergence rate because ϵ drops exponentially fast in T. We have not tried to improve the polynomial factors in m and T, and are aware of several ways to improve these factors (but at the expense of complicating the proof). We note that $\mathfrak{d}$ is the data input dimension and our result is independent of $\mathfrak{d}$.

Theorem 5.3.2 (SGD)**.** *Suppose* $b \in [n]$ *and* $m \geq \widetilde{\Omega}\big(\frac{\operatorname{poly}(n,L,\delta^{-1})\cdot d}{b}\big)$*. Starting from random initialization, with probability at least* $1 - e^{-\Omega(\log^2 m)}$*, SGD with learning rate* $\eta = \Theta\big(\frac{b\delta d}{\operatorname{poly}(n,L) m \log^2 m}\big)$ *and mini-batch size* b *finds* $F(\overrightarrow{W}) \leq \epsilon$ *in* $T = \Theta\big(\frac{\operatorname{poly}(n,L)\cdot \log^2 m}{\delta^2 b} \cdot \log \epsilon^{-1}\big)$ *iterations.*

This is again a linear convergence rate because $T \propto \log \frac{1}{\epsilon}$. The reason for the additional $\log^2 m$ factor comparing to Theorem 5.13.1 is because we have a $1-e^{-\Omega(\log^2 m)}$ high confidence bound.

Remark 5.3.1. For experts in optimization theory, one may immediately question the accuracy of Theorem 5.14.1, because SGD is known to converge at a slower rate $T \propto \frac{1}{\operatorname{poly}(\epsilon)}$ even

for convex functions. There is no contradiction here. Imaging a strongly convex function $f(x) = \sum_{i=1}^{n} f_i(x)$ that has a common minimizer $x^* \in \text{argmin}_x\{f_i(x)\}$ for every $i \in [n]$, then SGD is known to converge in a linear convergence rate.

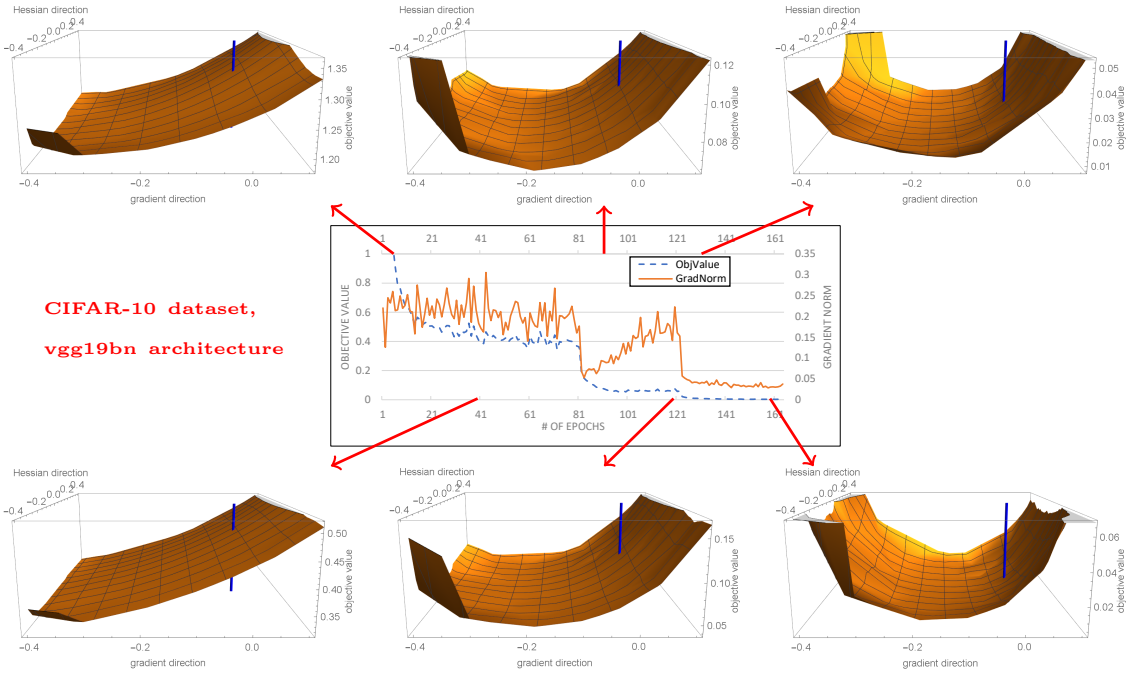


Figure 5.1: Landscapes of the CIFAR10 image-classification training objective $F(W)$ near the SGD training trajectory. The blue vertical stick marks the current point $W = W_t$ at the current iteration t. The x and y axes represent the gradient direction $\nabla F(W_t)$ and the most negatively curved direction of the Hessian after smoothing (approximately found by Oja's method [AL17, AL18]). The z axis represents the objective value. **Observation.** As far as minimizing objective is concerned, the (negative) gradient direction sufficiently decreases the training objective, and it is not needed to use second-order method to find negative curvature. This is consistent with our findings Theorem 5.11.1 and 5.12.1. Remark 1. Gradient norm does not tend to zero because cross-entropy loss is not strongly convex (see Section 5.6). Remark 2. The task is CIFAR10 (for CIFAR100 or CIFAR10 with noisy label, see Figure 5.2 through 5.7 in appendix). Remark 3. Architecture is VGG19 (for Resnet-32 or ResNet-110, see Figure 5.2 through 5.7 in the later sections). Remark 4. The six plots correspond to epochs 5, 40, 90, 120, 130 and 160. We start with learning rate 0.1, and decrease it to 0.01 at epoch 81, and to 0.001 at epoch 122. SGD with momentum 0.9 is used. The training code is unchanged from [Yan18] and we only write new code for plotting such landscapes.

5.4 Conceptual Messages and Technical Theorems

We highlight two conceptual messages that arise from the proofs of Theorem 5.13.1 and 5.14.1.

5.4.1 Objective is Almost Convex and Semi-Smooth

The first message is about the optimization landscape for points that are sufficiently close to the random initialization. It consists of two theorems, Theorem 5.11.1 says that the objective is "almost convex" and Theorem 5.12.1 says that the objective is "semi-smooth."

Theorem 5.4.1 (no critical point)**.** *With probability* $\geq 1 - e^{-\Omega(m/\operatorname{poly}(n,L,\delta^{-1}))}$ *over randomness* $\overrightarrow{W}^{(0)}, A, B$*, it satisfies for every* $\ell \in [L]$*, every* $i \in [n]$*, and every* $\overrightarrow{W}$ *with* $\|\overrightarrow{W} - \overrightarrow{W}^{(0)}\|_2 \leq \frac{1}{\operatorname{poly}(n,L,\delta^{-1})}$*,*

$$\|\nabla F(\overrightarrow{W})\|_F^2 \leq O\Big(F(\overrightarrow{W}) \times \frac{Lnm}{d}\Big) \quad \textit{and} \quad \|\nabla F(\overrightarrow{W})\|_F^2 \geq \Omega\Big(F(\overrightarrow{W}) \times \frac{\delta m}{dn^2}\Big) \ .$$

The first property above is easy to prove, while the second property above says that as long as the objective is large, the gradient norm is also large. (See also Figure 5.1.) This means, when we are sufficiently close to the random initialization, there is no saddle point or critical point of any order.

Theorem 5.11.1 gives us hope to find *global minima* of the objective $F(\overrightarrow{W})$, but is not enough. If we follow the negative gradient direction of $F(\overrightarrow{W})$, how can we guarantee that the objective truly decreases? Classical optimization theory usually relies on objective's (Lipscthiz) smoothness [Nes04] to derive an objective-decrease guarantee. Unfortunately, smoothness property at least requires the objective to be twice differentiable, but ReLU activation is not. To deal with this issue, we prove the following.

Theorem 5.4.2 (semi-smoothness)**.** *With probability at least* $1 - e^{-\Omega(m/\operatorname{poly}(L,\log m))}$ *over the randomness of* $\overrightarrow{W}^{(0)}, A, B$*, we have: for every* $\breve{\overrightarrow{W}} \in (\mathbb{R}^{m\times m})^L$ *with* $\|\breve{\overrightarrow{W}} - \overrightarrow{W}^{(0)}\|_2 \leq \frac{1}{\operatorname{poly}(L,\log m)}$*,*

and for every $\overrightarrow{W}' \in (\mathbb{R}^{m\times m})^L$ *with* $\|\overrightarrow{W}'\|_2 \le \frac{1}{\mathrm{poly}(L,\log m)}$*, the following inequality holds*

$$F(\overrightarrow{\breve{W}}+\overrightarrow{W}') \le F(\overrightarrow{\breve{W}}) + \langle \nabla F(\overrightarrow{\breve{W}}), \overrightarrow{W}'\rangle + \frac{\mathrm{poly}(L)\sqrt{nm\log m}}{\sqrt{d}} \cdot \|\overrightarrow{W}'\|_2 \big(F(\overrightarrow{\breve{W}})\big)^{1/2} + O\big(\frac{nL^2m}{d}\big)\|\overrightarrow{W}'\|_2^2$$

Different from classical smoothness, we still have a first-order term $\|\overrightarrow{W}'\|_2$ on the right hand side, while classical smoothness only has a second-order term $\|\overrightarrow{W}'\|_2^2$. As one can see in our final proofs, as m goes larger, the effect of the first-order term becomes smaller comparing to the second-order term. This brings Theorem 5.12.1 closer, but still not identical, to the classical Lipschitz smoothness.

Back to Theorem 5.13.1 and 5.14.1 The derivation of Theorem 5.13.1+5.14.1 from Theorem 5.11.1+5.12.1 is quite straightforward, and can be found in Section 5.13 and 5.14. At a high level, we show that GD/SGD can converge fast enough so that the weights stay close to random initialization by spectral norm bound $\frac{1}{\mathrm{poly}(n,L,\delta^{-1})}$. This ensures Theorem 5.11.1 and 5.12.1 both apply.[6]

In practice, one often goes beyond this theory-predicted spectral-norm boundary. However, quite interestingly, we still observe Theorem 5.11.1 and 5.12.1 hold in practice (see Figure 5.1). The gradient is sufficiently large and going in its negative direction can indeed decrease the objective.

[6]This spectral norm bound seems small, but is in fact quite large: it can totally change the outputs and fit the training data, because weights are randomly initialized (per entry) at around $\frac{1}{\sqrt{m}}$ for m being large.

5.4.2 Equivalence to Neural Tangent Kernel

Recall on input $x \in \mathbb{R}^{\mathfrak{d}}$, the network output $y(\overrightarrow{W}; x) \stackrel{\text{def}}{=} y = Bh_L \in \mathbb{R}^d$ is a function of the weights $\overrightarrow{W}$. Let us here focus on $d = 1$ for notational simplicity and leave $d > 1$ to the appendix. The *neural tangent kernel (NTK)* [JGH18] is usually referred to as the feature space defined by the network gradient at random initialization. In other words,

- Given two inputs $x, \widetilde{x} \in \mathbb{R}^{\mathfrak{d}}$, the NTK kernel function is given as

$$K^{\mathsf{ntk}}(x, \widetilde{x}) \stackrel{\text{def}}{=} \langle \nabla y(\overrightarrow{W}^{(0)}; x), \nabla y(\overrightarrow{W}^{(0)}; \widetilde{x}) \rangle$$

- Given weight matrix tuple $\overrightarrow{W}'$, the NTK model computes (we call the NTK objective)

$$y^{\mathsf{ntk}}(\overrightarrow{W}'; x) \stackrel{\text{def}}{=} \langle \nabla y(\overrightarrow{W}^{(0)}; x), \overrightarrow{W}' \rangle = \textstyle\sum_{\ell=1}^{L} \langle \nabla_{W_\ell} y(\overrightarrow{W}^{(0)}; x), W'_\ell \rangle \ .$$

In contrast, the dynamic NTK is given by arbitrary weight tuple $\overrightarrow{W} = \overrightarrow{W}^{(0)} + \overrightarrow{W}'$ that may not be at random initialization. [JGH18] proved in their original paper that, when m is infinite, dynamic NTK and NTK are identical because during the training process $\lim_{m\to\infty} \|\overrightarrow{W}'\|_2 = 0$ if $\overrightarrow{W} = \overrightarrow{W}^{(0)} + \overrightarrow{W}'$ is output of gradient descent.

In this paper, we complement [JGH18] by showing a *polynomial* bound on this equivalence, for *any* point that is within a certain ball of $\overrightarrow{W}^{(0)}$. It is a simple corollary of Theorem 5.11.1 and Theorem 5.12.1, but we state it independently here since it may be of additional interest.[7]

[7] Although Theorem 5.15.1 was not explicitly stated until version 5 of this paper, its proof was fully contained in the proofs of Theorem 5.13.1 and 5.14.1. Since some readers cannot find it, we state it here as a separate theorem.

Theorem 5.4.3. *Let $\overrightarrow{W}^{(0)}, A, B$ be at random initialization. For fixed unit vectors $x, \widetilde{x} \in \mathbb{R}^{\mathfrak{d}}$, every (small) parameter $\omega \leq \frac{1}{\mathrm{poly}(L, \log m)}$, with probability at least $1 - e^{-\Omega(m\omega^{2/3}L)}$ over $\overrightarrow{W}^{(0)}, A, B$, we have for all $\overrightarrow{W}'$ with $\|\overrightarrow{W}'\|_2 \leq \omega$,*

(a) $\|\nabla y(\overrightarrow{W}^{(0)} + \overrightarrow{W}'; x) - \nabla y^{\mathsf{ntk}}(\overrightarrow{W}'; x)\|_F \leq \widetilde{O}\big(\omega^{1/3} L^3\big) \cdot \|\nabla y^{\mathsf{ntk}}(\overrightarrow{W}'; x)\|_F$;

(b) $y(\overrightarrow{W}^{(0)} + \overrightarrow{W}'; x) = y(\overrightarrow{W}^{(0)}; x) + y^{\mathsf{ntk}}(\overrightarrow{W}'; x) \pm \widetilde{O}\big(L^3 \omega^{4/3} \sqrt{m}\big)$; *and*

(c) $\big\langle \nabla y(\overrightarrow{W}^{(0)} + \overrightarrow{W}'; x), \nabla y(\overrightarrow{W}^{(0)} + \overrightarrow{W}'; \widetilde{x}) \big\rangle = K^{\mathsf{ntk}}(x, \widetilde{x}) \pm \widetilde{O}\big(\omega^{1/3} L^3\big) \cdot \sqrt{K^{\mathsf{ntk}}(x, x) K^{\mathsf{ntk}}(\widetilde{x}, \widetilde{x})}$.

Theorem 5.15.1a and 5.15.1c says that dynamic NTK and NTK are almost equivalent up to a small multiplicative factor as long as $\omega < \frac{1}{\mathrm{poly}(L, \log m)}$; while Theorem 5.15.1b says that the NTK objective is almost exactly the first-order approximation of the neural network output as long as $\omega < \frac{1}{m^{3/8}\,\mathrm{poly}(L)}$.

In comparison, in Theorem 5.13.1 and 5.14.1, GD/SGD outputs $\overrightarrow{W}'$ satisfying $\omega \leq \frac{\mathrm{poly}(n, \delta^{-1})}{\sqrt{m}} \ll \frac{1}{m^{3/8}\,\mathrm{poly}(L)}$ under the assumption $m \geq \mathrm{poly}(n, L, \delta^{-1})$.[8] Thus, Theorem 5.15.1b implies $\overrightarrow{W}'$ is also a solution to the NTK regression objective.

Remark 5.4.1. If one wishes to have $y(\overrightarrow{W}^{(0)} + \overrightarrow{W}'; x) \approx y^{\mathsf{ntk}}(\overrightarrow{W}'; x)$ without the zero-order term $y(\overrightarrow{W}^{(0)}; x)$, this can be achieved by properly scaling down the random initialization by a factor of the target error $\epsilon < 1$. This was used in the follow-up [ALL18c] to achieve small generalization error on over-parameterized neural networks.

[8] See (5.19) and (5.21) in the proofs.

5.5 Proof Overview

Our proof to the Theorem 5.11.1 and 5.12.1 mostly consist of the following steps.

Step 1: properties at random initialization Let $\overrightarrow{W} = \overrightarrow{W}^{(0)}$ be at random initialization and $h_{i,\ell}$ and $D_{i,\ell}$ be defined with respect to $\overrightarrow{W}$. We first show that forward propagation neither explode or vanish. That is,

$$\|h_{i,\ell}\| \approx 1 \text{ for all } i \in [n] \text{ and } \ell \in [L].$$

This is basically because for a fixed y, we have $\|Wy\|^2$ is around 2, and if its signs are sufficiently random, then ReLU activation kills half of the norm, that is $\|\phi(Wy)\| \approx 1$. Then applying induction finishes the proof.

Analyzing forward propagation is not enough. We also need spectral norm bounds on the backward matrix and on the intermediate matrix

$$\|BD_{i,L}W_L \cdots D_{i,a}W_a\|_2 \leq O(\sqrt{m/d}) \quad \text{and} \quad \|D_{i,a}W_a \cdots D_{i,b}W_b\|_2 \leq O(\sqrt{L}) \tag{5.1}$$

for every $a, b \in [L]$. Note that if one naively bounds the spectral norm by induction, then $\|D_{i,a}W_a\|_2 \approx 2$ and it will *exponentially blow up!* Our careful analysis ensures that even when L layers are stacked together, there is no exponential blow up in L.

The final lemma in this step proves that, as long as $\|x_i - x_j\| \geq \delta$, then

$$\|h_{i,\ell} - h_{j,\ell}\| \geq \Omega(\delta) \text{ for each layer } \ell \in [L].$$

Again, if one is willing to sacrifice an exponential factor and prove a lower bound $\delta \cdot 2^{-\Omega(L)}$, this will be easy. What is hard is to derive such lower bound without sacrificing more than a constant factor, but under the condition of $\delta \leq \frac{1}{CL}$. Details are in Section 5.8.

Step 2: stability after adversarial perturbation We show that for every $\overrightarrow{W}$ that is "close" to initialization, meaning $\|\overrightarrow{W} - \overrightarrow{W}^{(0)}\|_2 \leq \omega$ for some $\omega \leq \frac{1}{\mathrm{poly}(L)}$, then

(a) the number of sign changes $\|D_{i,\ell} - \widetilde{D}_{i,\ell}\|_0$ is at most $O(m\omega^{2/3}L) \ll m$, and

(b) the perturbation amount $\|h_{i,\ell} - \widetilde{h}_{i,\ell}\| \leq O(\omega L^{5/2}) \ll 1$.

Since $\omega \leq \frac{1}{\mathrm{poly}(L)}$, both changes above become negligible. We call this "forward stability", and it is the most technical proof of this paper. Intuitively, both "(a) implies (b)" and "(b) implies (a)" are trivial to prove by matrix concentration.[9] Unfortunately, one cannot apply such derivation by induction, because constants will blow up exponentially in the number of layers. We need some careful double induction introduced by [AZLS18], and details in Section 5.9.1. Another main result in this step is to derive stability for the backward matrix and the intermediate matrix. We show that when $w \leq \mathrm{poly}(L)$, (5.1) remains to hold. Details are in Section 5.9.2 and 5.9.3.

Remark 5.5.1. In the final proof, $\overrightarrow{W}$ is a point obtained by GD/SGD starting from $\overrightarrow{W}^{(0)}$, and thus $\overrightarrow{W}$ may depend on the randomness of $\overrightarrow{W}^{(0)}$. Since we cannot control how such randomness correlates, we argue for the above stability properties against *all possible* $\overrightarrow{W}$. This is why we call it "stability against adversarial perturbation."

Step 3: gradient bound The hard part of Theorem 5.11.1 is to show gradient lower bound. For this purpose, recall from Fact 5.2.3 that each sample $i \in [n]$ contributes to the

[9]Namely, if the number of sign changes is bounded in all layers, then $h_{i,\ell}$ and $\widetilde{h}_{i,\ell}$ cannot be too far away by applying matrix concentration; and reversely, if $h_{i,\ell}$ is not far from $\widetilde{h}_{i,\ell}$ in all layers, then the number of sign changes per layer must be small.

full gradient matrix by $D_{i,\ell}(\mathsf{Back}_{i,\ell+1}^\top \mathsf{loss}_i)h_{i,\ell-1}^\top$, where the backward matrix is applied to a loss vector loss_i. To show this is large, intuitively, one wishes to show $(\mathsf{Back}_{i,\ell+1}^\top \mathsf{loss}_i)$ and $h_{i,\ell-1}$ are both vectors with large Euclidean norm.

Thanks to Step 1 and 2, this is not hard for a *single* sample $i \in [n]$. For instance, $\|\widetilde{h}_{i,\ell-1}\| \approx 1$ by Step 1 and we know $\|h_{i,\ell-1} - \widetilde{h}_{i,\ell-1}\| \leq o(1)$ from Step 2. One can also argue for $\mathsf{Back}_{i,\ell+1}^\top \mathsf{loss}_i$ but this is a bit harder. Indeed, when moving from random initialization $\overrightarrow{W}^{(0)}$ to $\overrightarrow{W}$, the loss vector loss_i can change completely. Fortunately, $\mathsf{loss}_i \in \mathbb{R}^d$ is a low-dimensional vector, so one can calculate $\|\mathsf{Back}_{i,\ell+1}^\top u\|$ for every fixed u and then apply ϵ-net.

Finally, how to combine the above argument with multiple samples $i \in [n]$? These matrices are clearly not independent and may (in principle) sum up to zero. To deal with this, we use $\|h_{i,\ell} - h_{j,\ell}\| \geq \Omega(\delta)$ from Step 1. In other words, even if the contribution matrix $D_{i,\ell}(\mathsf{Back}_{i,\ell+1}^\top \mathsf{loss}_i)h_{i,\ell-1}^\top$ with respect to one sample i is fixed, the contribution matrix with respect to other samples $j \in [n] \setminus \{i\}$ are still sufficiently random. Thus, the final gradient matrix will still be large. This idea comes from the prior work [LL18], [10] and helps us prove Theorem 5.11.1. Details in Section 5.10 and 5.11.

Step 4: semi-smoothness In order to prove Theorem 5.12.1, one needs to argue, if we are currently at $\breve{\overrightarrow{W}}$ and perturb it by $\overrightarrow{W}'$, then how much does the objective change in second and higher order terms. This is different from our stability theory in Step 2, because Step 2 is regarding having a perturbation on $\overrightarrow{W}^{(0)}$; in contrast, in Theorem 5.12.1 we need a (small) perturbation $\overrightarrow{W}'$ on top of $\breve{\overrightarrow{W}}$, which may already be a point perturbed from $\overrightarrow{W}^{(0)}$.

[10] This is the only technical idea that we borrowed from [LL18], which is the over-parameterization theory for 2-layer neural networks.

Nevertheless, we still manage to show that, if $\hbar_{i,\ell}$ is calculated on $\overrightarrow{\breve{W}}$ and $h_{i,\ell}$ is calculated on $\overrightarrow{\breve{W}} + \overrightarrow{W'}$, then $\|h_{i,\ell} - \hbar_{i,\ell}\| \leq O(L^{1.5})\|W'\|_2$. This, along with other properties to prove, ensures semi-smoothness. This explains Theorem 5.12.1 and details are in Section 5.12.

Remark 5.5.2. In other words, the amount of changes to each hidden layer (i.e., $h_{i,\ell} - \hbar_{i,\ell}$) is proportional to the amount of perturbation $\|W'\|_2$. This may sound familiar to some readers: a ReLU function is Lipschitz continuous $|\phi(a) - \phi(b)| \leq |a - b|$, and composing Lipschitz functions still yield Lipschitz functions. What is perhaps surprising here is that this "composition" does not create exponential blow-up in the Lipschitz continuity parameter, as long as the amount of over-parameterization is sufficient and $\overrightarrow{\breve{W}}$ is close to initialization.

5.6 Notable Extensions

Our Step 1 through Step 4 in Section 5.5 in fact give rise to a general plan for proving the training convergence of any neural network (at least with respect to the ReLU activation). Thus, it is expected that it can be generalized to many other settings. Not only we can have different number of neurons each layer, our theorems can be extended at least in the following three major directions.[11]

Different loss functions There is absolutely no need to restrict only to ℓ_2 regression loss. We prove in Section 5.16 that, for any Lipschitz-smooth loss function f:

Theorem 5.6.1 (arbitrary loss)**.** *From random initialization, with probability at least* $1 - e^{-\Omega(\log^2 m)}$*, gradient descent with appropriate learning rate satisfy the following.*

- *If f is nonconvex but σ-gradient dominant (a.k.a. Polyak-Łojasiewicz), GD finds ϵ-error minimizer in*[12]

$$T = \widetilde{O}\big(\tfrac{\mathrm{poly}(n,L)}{\sigma\delta^2} \cdot \log \tfrac{1}{\epsilon}\big) \text{ iterations}$$

  as long as $m \geq \widetilde{\Omega}\big(\mathrm{poly}(n, L, \delta^{-1}) \cdot d\sigma^{-2}\big)$.

- *If f is convex, then GD finds ϵ-error minimizer in*

$$T = \widetilde{O}\big(\tfrac{\mathrm{poly}(n,L)}{\delta^2} \cdot \tfrac{1}{\epsilon}\big) \text{ iterations}$$

[11] In principle, each such proof may require a careful rewriting of the main body of this paper. We choose to sketch only the proof difference (in the appendix) in order to keep this paper short. If there is sufficient interest from the readers, we can consider adding the full proofs in the future revision of this paper.

[12] Note that the loss function when combined with the neural network together $f(Bh_{i,L})$ is *not* gradient dominant. Therefore, one cannot apply classical theory on gradient dominant functions to derive our same result.

as long as $m \geq \widetilde{\Omega}\big(\operatorname{poly}(n, L, \delta^{-1}) \cdot d \log \epsilon^{-1}\big)$.

- *If* f *is non-convex, then SGD finds a point with* $\|\nabla f\| \leq \epsilon$ *in at most*[13]

$$T = \widetilde{O}\big(\tfrac{\operatorname{poly}(n,L)}{\delta^2} \cdot \tfrac{1}{\epsilon^2}\big) \text{ iterations}$$

as long as $m \geq \widetilde{\Omega}\big(\operatorname{poly}(n, L, \delta^{-1}) \cdot d\epsilon^{-1}\big)$.

- *If* f *is cross-entropy for multi-label classification, then GD attains* 100% *training accuracy in at most*[14].

$$T = \widetilde{O}\big(\tfrac{\operatorname{poly}(n,L)}{\delta^2}\big) \text{ iterations}$$

as long as $m \geq \widetilde{\Omega}\big(\operatorname{poly}(n, L, \delta^{-1}) \cdot d\big)$.

We remark here that the ℓ_2 loss is 1-gradient dominant so it falls into the above general Theorem 5.6.1. One can also derive similar bounds for (mini-batch) SGD so we do not repeat the statements here.

Convolutional neural networks (CNN) There are lots of different ways to design CNN and each of them may require somewhat different proofs. In Section 5.17, we study the case when $A, W_1, \ldots, W_{L-1}$ are convolutional while W_L and B are fully connected. We assume for notational simplicity that each hidden layer has $\mathfrak{d}$ points each with m channels. (In vision tasks, a point is a pixel). In the most general setting, these values $\mathfrak{d}$ and m can vary across layers. We prove the following theorem:

[13] Again, this cannot be derived from classical theory of finding approximate saddle points for non-convex functions, because weights $\overrightarrow{W}$ with small $\|\nabla f(Bh_{i,L})\|$ is a very different (usually much harder) task comparing to having small gradient with respect to $\overrightarrow{W}$ for the entire composite function $f(Bh_{i,L})$.

[14] This is because attaining constant objective error $\epsilon = 1/4$ for the cross-entropy loss suffices to imply perfect training accuracy.

Theorem 5.6.2 (CNN). *As long as* $m \geq \widetilde{\Omega}\big(\operatorname{poly}(n, L, \mathfrak{d}, \delta^{-1}) \cdot d\big)$, *with high probability, GD and SGD find an* ϵ*-error solution for* ℓ_2 *regression in* $T = \widetilde{O}\big(\frac{\operatorname{poly}(n,L,\mathfrak{d})}{\delta^2} \cdot \log \epsilon^{-1}\big)$ *iterations for CNN.*

Of course, one can replace ℓ_2 loss with other loss functions in Theorem 5.6.1 to get different types of convergence rates. We do not repeat them here.

Residual neural networks (ResNet) There are lots of different ways to design ResNet and each of them may require somewhat different proofs. In symbols, between two layers, one may study $h_\ell = \phi(h_{\ell-1} + Wh_{\ell-1})$, $h_\ell = \phi(h_{\ell-1} + W_2\phi(W_1 h_{\ell-1}))$, or even $h_\ell = \phi(h_{\ell-1} + W_3\phi(W_2\phi(W_1 h_{\ell-1})))$. Since the main purpose here is to illustrate the generality of our techniques but not to attack each specific setting, in Section 5.18, we choose to consider the simplest residual setting $h_\ell = \phi(h_{\ell-1} + Wh_{\ell-1})$ (that was also studied for instance by theoretical work [HM17]). With appropriately chosen random initialization, we prove the following theorem:

Theorem 5.6.3 (ResNet). *As long as* $m \geq \widetilde{\Omega}\big(\operatorname{poly}(n, L, \delta^{-1}) \cdot d\big)$, *with high probability, GD and SGD find an* ϵ*-error solution for* ℓ_2 *regression in* $T = \widetilde{O}\big(\frac{\operatorname{poly}(n,L)}{\delta^2} \cdot \log \epsilon^{-1}\big)$ *iterations for ResNet.*

Of course, one can replace ℓ_2 loss with other loss functions in Theorem 5.6.1 to get different types of convergence rates. We do not repeat them here.

5.7 Detailed Proofs

- In Section 5.8, we derive network properties at random initialization.
- In Section 5.9, we derive the stability theory against adversarial perturbation.
- In Section 5.10, we gradient upper and lower bounds at random initialization.
- In Section 5.11, we prove Theorem 5.11.1.
- In Section 5.12, we prove Theorem 5.12.1.
- In Section 5.13, we prove Theorem 5.13.1.
- In Section 5.14, we prove Theorem 5.14.1.
- In Section 5.15, we prove Theorem 5.15.1.

5.8 Properties at Random Initialization

Throughout this section we assume $\overrightarrow{W}, A$ and B are randomly generated according to Definition 5.2.2. The diagonal sign matrices $D_{i,\ell}$ are also determined according to this random initialization.

5.8.1 Forward Propagation

Lemma 5.8.1 (forward propagation)**.** *If $\epsilon \in (0,1]$, with probability at least $1 - O(nL) \cdot e^{-\Omega(m\epsilon^2/L)}$ over the randomness of $A \in \mathbb{R}^{m \times \mathfrak{d}}$ and $\overrightarrow{W} \in (\mathbb{R}^{m\times m})^L$, we have*

$$\forall i \in [n], \ell \in \{0,1,\ldots,L\} \quad : \quad \|h_{i,\ell}\| \in [1-\epsilon, 1+\epsilon] \ .$$

Remark 5.8.1. Lemma 5.8.1 is in fact trivial to prove if the allowed failure probability is instead $e^{-\Omega(m\epsilon^2/L^2)}$ (by applying concentration inequality layer by layer).

Before proving Lemma 5.8.1 we note a simple mathematical fact:

Fact 5.8.2. *Let $h, q \in \mathbb{R}^p$ be fixed vectors and $h \neq 0$, $W \in \mathbb{R}^{m\times p}$ be random matrix with i.i.d. entries $W_{i,j} \sim \mathcal{N}(0, \frac{2}{m})$, and vector $v \in \mathbb{R}^m$ defined as $v_i = \phi((Wh)_i) = \mathbb{1}_{(W(h+q))_i \geq 0}(Wh)_i$. Then,*

- *$|v_i|$ follows i.i.d. from the following distribution: with half probability $|v_i| = 0$, and with the other half probability $|v_i|$ follows from folded Gaussian distributions $|\mathcal{N}(0, \frac{2\|h\|^2}{m})|$.*
- *$\frac{m\|v\|^2}{2\|h\|^2}$ is in distribution identical to χ^2_ω (chi-square distribution of order ω) where ω follows from binomial distribution $\mathcal{B}(m, 1/2)$.*

Proof of Fact 5.8.2. We assume each vector W_i is generated by first generating a gaussian vector $g \sim \mathcal{N}(0, \frac{2I}{m})$ and then setting $W_i = \pm g$ where the sign is chosen with half-half probability. Now, $|\langle W_i, h\rangle| = |\langle g, h\rangle|$ only depends on g, and is in distribution identical to $|\mathcal{N}(0, \frac{2\|h\|^2}{m})|$. Next, after the sign is determined, the indicator $\mathbb{1}_{\langle W_i, h+q\rangle \geq 0}$ is 1 with half probability and 0 with another half. Therefore, $|v_i|$ satisfies the aforementioned distribution. As for $\|v\|^2$, letting $\omega \in \{0,1,\ldots,m\}$ be the variable indicator how many indicators are 1, then $\omega \sim \mathcal{B}(m, 1/2)$ and $\frac{m\|v\|^2}{2\|h\|^2} \sim \chi^2_\omega$. □

Proof of Lemma 5.8.1. We only prove Lemma 5.8.1 for a fixed $i \in [n]$ and $\ell \in \{0, 1, 2, \ldots, L\}$ because we can apply union bound at the end. Below, we drop the subscript i for notational convenience, and write $h_{i,\ell}$ and x_i as h_ℓ and x respectively.

Letting $\Delta_\ell \stackrel{\text{def}}{=} \frac{\|h_\ell\|^2}{\|h_{\ell-1}\|^2}$, we can write

$$\log \|h_{b-1}\|^2 = \log \|x\|^2 + \sum_{\ell=0}^{b-1} \log \Delta_\ell = \sum_{\ell=0}^{b-1} \log \Delta_\ell \ .$$

According to Fact 5.8.2, fixing any $h_{\ell-1} \neq 0$ and letting W_ℓ be the only source of randomness, we have $\frac{m}{2}\Delta_\ell \sim \chi^2_\omega$ where $\omega \sim \mathcal{B}(m, 1/2)$. For such reason, for each Δ_ℓ, we can write $\Delta_\ell = \Delta_{\ell,\omega}$ where $\frac{m}{2}\Delta_{\ell,\omega} \sim \chi^2_\omega$ and $\omega \sim \mathcal{B}(m, 1/2)$. In the analysis below, we condition on the event that $\omega \in [0.4m, 0.6m]$; this happens with probability $\geq 1 - e^{-\Omega(m)}$ for each layer $\ell \in [L]$. To simplify our notations, if this event does not hold, we set $\Delta_\ell = 1$.

Expectation One can verify that $\mathbb{E}[\log \Delta_{\ell,\omega} \mid \omega] = \log \frac{4}{m} + \psi(\frac{\omega}{2})$ where $\psi(h) = \frac{\Gamma'(h)}{\Gamma(h)}$ is the digamma function. Using the bound $\log h - \frac{1}{h} \leq \psi(h) \leq \log h - \frac{1}{2h}$ of digamma function, we have

$$\log \frac{2\omega}{m} - \frac{2}{\omega} \leq \mathbb{E}[\log \Delta_{\ell,\omega} \mid \omega] \leq \log \frac{2\omega}{m} - \frac{1}{\omega}.$$

Whenever $\omega \in [0.4m, 0.6m]$, we can write

$$\log \frac{2\omega}{m} = \log \left(1 + \frac{2\omega - m}{m}\right) \geq \frac{2\omega - m}{m} - \left(\frac{2\omega - m}{m}\right)^2$$

It is easy to verify $\mathbb{E}_\omega \left[\frac{2\omega - m}{m}\right] = 0$ and $\mathbb{E}_\omega \left[\left(\frac{2\omega - m}{m}\right)^2\right] = \frac{1}{m}$. Therefore,

$$\mathop{\mathbb{E}}_{\omega} \left[\log \frac{2\omega}{m}\right] \geq -\frac{1}{m} - \Pr\left[\omega \notin [0.4m, 0.6m]\right] \cdot \log \frac{2}{m} \geq -\frac{2}{m}$$

Combining everything together, along with the fact that $\mathbb{E}_\omega[\log \frac{2\omega}{m}] \le \log \frac{\mathbb{E}[2\omega]}{m} = 0$, we have (when m is sufficiently larger than a constant)

$$-\frac{4}{m} \le \mathbb{E}[\log \Delta_\ell] \le 0. \tag{5.2}$$

Subgaussian Tail By standard tail bound for chi-square distribution, we know that

$$\forall t \in [0,\infty): \quad \Pr\left[\left|\frac{m}{2}\Delta_{\ell,\omega} - \omega\right| \le t \,\middle|\, \omega\right] \ge 1 - 2e^{-\Omega(t^2/\omega)} - e^{-\Omega(t)} \ .$$

Since we only need to focus on $\omega \ge 0.4m$, this means

$$\forall t \in [0,m]: \quad \Pr\left[\left|\frac{m}{2}\Delta_{\ell,\omega} - \omega\right| \le t \,\middle|\, \omega \ge 0.4m\right] \ge 1 - O(e^{-\Omega(t^2/m)}) \ .$$

On the other hand, by Chernoff-Hoeffding bound, we also have

$$\Pr_\omega\left[\left|\omega - \frac{m}{2}\right| \le t\right] \ge 1 - O(e^{-\Omega(t^2/m)})$$

Together, using the definition $\Delta_\ell = \Delta_{\ell,\omega}$ (or $\Delta_\ell = 1$ if $\omega \notin [0.4m, 0.6m]$), we obtain

$$\forall t \in [0,m]: \quad \Pr\left[\left|\frac{m}{2}\Delta_{\ell} - \frac{m}{2}\right| \le t\right] \ge 1 - O(e^{-\Omega(t^2/m)}) \ .$$

This implies,

$$\forall t \in \left[0,\frac{m}{4}\right]: \quad \Pr\left[|\log \Delta_\ell| \le \frac{t}{m}\right] \ge 1 - O(e^{-\Omega(t^2/m)}) \ . \tag{5.3}$$

Now, let us make another simplification: define $\widehat{\Delta}_\ell = \Delta_\ell$ if $|\log \Delta_\ell| \le \frac{1}{4}$ and $\widehat{\Delta}_\ell = 1$ otherwise. In this way, (5.3) implies that $X = \log \widehat{\Delta}_\ell$ is an $O(m)$-subgaussian random variable.

Concentration Using martingale concentration on subgaussian variables (see for instance [Sha11]), we have for $\epsilon \in (0, 1]$,

$$\Pr\left[\left|\sum_{\ell=0}^{b-1} \log \widehat{\Delta}_\ell - \mathbb{E}[\log \widehat{\Delta}_\ell]\right| > \epsilon\right] \leq O\big(e^{-\Omega(\epsilon^2 m/L)}\big).$$

Since with probability $\geq 1 - Le^{-\Omega(m)}$ it satisfies $\widehat{\Delta}_\ell = \Delta_\ell$ for all $\ell \in [L]$, combining this with (5.2), we have

$$\Pr\left[\left|\sum_{\ell=0}^{b-1} \log \Delta_\ell\right| > \epsilon\right] \leq O\big(e^{-\Omega(\epsilon^2 m/L)}\big).$$

In other words, $\|h_{b-1}\|^2 \in \big[1 - \epsilon, 1 + \epsilon\big]$ with probability at least $1 - O\big(e^{-\Omega(\epsilon^2 m/L)}\big)$. □

5.8.2 Intermediate Layers

Lemma 5.8.3 (intermediate layers)**.** *Suppose $m \geq \Omega(nL\log(nL))$. With probability at least $\geq 1 - e^{-\Omega(m/L)}$ over the randomness of $\overrightarrow{W} \in (\mathbb{R}^{m\times m})^L$, for all $i \in [n], 1 \leq a \leq b \leq L$,*

(a) $\|W_b D_{i,b-1} W_{b-1} \cdots D_{i,a} W_a\|_2 \leq O(\sqrt{L})$.

(b) $\|W_b D_{i,b-1} W_{b-1} \cdots D_{i,a} W_a v\| \leq 2\|v\|$ *for all vectors v with* $\|v\|_0 \leq O\big(\frac{m}{L\log m}\big)$.

(c) $\|u^\top W_b D_{i,b-1} W_{b-1} \cdots D_{i,a} W_a\| \leq O(1)\|u\|$ *for all vectors u with* $\|u\|_0 \leq O\big(\frac{m}{L\log m}\big)$.

For any integer s with $1 \leq s \leq O\big(\frac{m}{L\log m}\big)$, with probability at least $1 - e^{-\Omega(s\log m)}$ over the randomness of $\overrightarrow{W} \in (\mathbb{R}^{m\times m})^L$:

(d) $|u^\top W_b D_{i,b-1} W_{b-1} \cdots D_{i,a} W_a v| \leq \|u\|\|v\| \cdot O\big(\frac{\sqrt{s\log m}}{\sqrt{m}}\big)$ *for all vectors u, v with* $\|u\|_0, \|v\|_0 \leq s$.

Proof. Again we prove the lemma for fixed i, a and b because we can take a union bound at the end. We drop the subscript i for notational convenience.

(a) Let z_{a-1} be any fixed unit vector, and define $z_\ell = D_\ell W_\ell \cdots D_a W_a z_{a-1}$. According to Fact 5.8.2 again, fixing any $z_{\ell-1}$ and letting W_ℓ be the only source of randomness, defining $\Delta_\ell \stackrel{\text{def}}{=} \frac{\|z_\ell\|^2}{\|z_{\ell-1}\|^2}$, we have that $\frac{m}{2}\Delta_\ell$ is distributed according to a χ^2_ω where $\omega \sim \mathcal{B}(m, \frac{1}{2})$. Therefore, we have

$$\log \|z_{b-1}\|^2 = \log \|z_{a-1}\|^2 + \sum_{\ell=a}^{b-1} \log \Delta_\ell = \sum_{\ell=a}^{b-1} \log \Delta_\ell \ .$$

Using exactly the same proof as Lemma 5.8.1, we have

$$\|z_{b-1}\|^2 = \|W_b D_{b-1} W_{b-1} \cdots D_a W_a z_{a-1}\|^2 \in \big[1 - 1/3, 1 + 1/3\big]$$

with probability at least $1 - e^{-\Omega(m/L)}$. As a result, if we fix a subset $M \subseteq [m]$ of cardinality $|M| \leq O(m/L)$, taking ϵ-net, we know that with probability at least $e^{-\Omega(m/L)}$, it satisfies

$$\|W_b D_{b-1} W_{b-1} \cdots D_a W_a u\| \leq 2\|u\| \tag{5.4}$$

for all vectors u whose coordinates are zeros outside M. Now, for an arbitrary unit vector $v \in \mathbb{R}^m$, we can decompose it as $v = u_1 + \cdots + u_N$ where $N = O(L)$, each u_j is non-zero only at $O(m/L)$ coordinates, and the vectors $u_1, \ldots, u_N$ are non-zeros on different coordinates. We can apply (5.4) for each each such u_j and triangle inequality. This gives

$$\|W_b D_{b-1} W_{b-1} \cdots D_a W_a v\| \leq 2 \sum_{j=1}^{N} \|u_j\| \leq 2\sqrt{N} \Big(\sum_{j=1}^{N} \|u_j\|^2 \Big)^{1/2} \leq O(\sqrt{L}) \cdot \|v\|.$$

(b) The proof of Lemma 5.8.3b is the same as Lemma 5.8.3a, except to take ϵ-net over all $O\big(\frac{m}{L \log m}\big)$-sparse vectors u and then applying union bound.

(c) Similar to the proof of Lemma 5.8.3a, for any fixed vector v, we have that with probability at least $1 - e^{-\Omega(m/L)}$ (over the randomness of $W_{b-1}, \ldots, W_1, A$),

$$\|D_{b-1} W_{b-1} \cdots D_a W_a v\| \leq 2\|v\|.$$

Conditioning on this event happens, using the randomness of W_b, we have for each fixed vector $u \in \mathbb{R}^m$, we have

$$\Pr_{W_b} \left[\Big| u^\top W_b \big(D_{b-1} W_{b-1} \cdots D_a W_a v \big) \Big| \geq \frac{4}{\sqrt{L}} \|u\| \|v\| \right] \leq e^{-\Omega(m/L)}.$$

Now consider the case that v is a sparse vector that is only non-zero over some fixed index set $M \subseteq [m]$ (with $|M| \leq O(m/L)$), and that u is of sparsity $s = O\big(\frac{m}{L \log m}\big)$. Taking ϵ-net over all such possible vectors u and v, we have with probability at least $1 - e^{-\Omega(m/L)}$, for all vectors $u \in \mathbb{R}^m$ with $\|u\|_0 \leq s$ and all vectors $v \in \mathbb{R}^m$ that have non-zeros only in M,

$$\Big|u^\top W_b\big(D_{b-1}W_{b-1}\cdots D_a W_a v\big)\Big| \leq \frac{8}{\sqrt{L}}\|u\|\|v\| \ . \tag{5.5}$$

Back to the case when v is an arbitrary vector, we can partition $[m]$ into N index sets $[m] = M_1 \cup M_2 \cup \cdots \cup M_N$ and write $v = v_1 + v_2 + \cdots + v_N$, where $N = O(L)$ and each v_j is non-zero only in M_j. By applying (5.5) for N times and using triangle inequality, we have

$$\begin{aligned}\Big|u^\top W_b\big(D_{b-1}W_{b-1}\cdots D_a W_a v\big)\Big| &\leq \sum_{j=1}^{N}\Big|u^\top W_b\big(D_{b-1}W_{b-1}\cdots D_a W_a v_j\big)\Big| \\ &\leq \frac{8}{\sqrt{L}}\|u\| \times \sum_{j=1}^{N}\|v_j\| \leq O(1) \times \|u\|\|v\| \ .\end{aligned}$$

(d) We apply the same proof as Lemma 5.8.3c with minor changes to the parameters. We can show with probability at least $1-e^{-\Omega(m/L)}$ (over the randomness of $W_{b-1}, \ldots, W_1, A$), for a fixed vector $v \in \mathbb{R}^m$:

$$\|D_{b-1}W_{b-1}\cdots D_a W_a v\| \leq 2\|v\| \ .$$

Further using the randomness of W_b, we have that conditioning on the above event, fixing any $u \in \mathbb{R}^m$, with probability at least $1 - e^{-\Omega(s\log m)}$ over the randomness of W_b:

$$\big|uW_b\big(D_{b-1}W_{b-1}\cdots D_a W_a v\big)\big| \leq \big(\frac{s\log m}{m}\big)^{1/2} \times O\big(\|v\|\|u\|\big) \ .$$

Finally, taking ϵ-net over all possible vectors u, v that are s sparse, we have the desired result.

□

5.8.3 Backward Propagation

Lemma 5.8.4 (backward propagation)**.** *Suppose $m \geq \Omega(nL\log(nL))$. If $s \geq \Omega\big(\frac{d}{\log m}\big)$ and $s \leq O\big(\frac{m}{L\log m}\big)$, then with probability at least $1-e^{-\Omega(s\log m)}$, for all $i \in [n]$, $a = 1, 2, \ldots, L+1$,*

(a) $|v^\top B D_{i,L} W_L \cdots D_{i,a} W_a u| \leq O\big(\frac{\sqrt{s\log m}}{\sqrt{d}}\big)\|v\|\|u\|$ *for all $v \in \mathbb{R}^d$ and all $u \in \mathbb{R}^m$ with $\|u\|_0 \leq s$.*

With probability at least $\geq 1 - e^{-\Omega(m/L)}$, for all $i \in [n], 1 \leq a \leq L$,

(b) $\|v^\top B D_{i,L} W_L \cdots D_{i,a} W_a\| \leq O(\sqrt{m/d})\|v\|$ *for all vectors $u \in \mathbb{R}^d$ if $d \leq O\big(\frac{m}{L\log m}\big)$.*

Proof. (a) The proof follows the same idea of Lemma 5.8.3 (but choosing $b = L$). Given any fixed vector u, we have with probability at least $1-e^{-\Omega(m/L)}$ (over the randomness of $W_L, \ldots, W_1, A$),

$$\|D_L W_L \cdots D_a W_a u\| \leq 2\|u\| \ .$$

Conditioning on this event happens, using the randomness of B (recall each entry of B follows from $\mathcal{N}(0, \frac{1}{d})$), we have for each fixed vector $u \in \mathbb{R}^m$,

$$\Pr_B\left[\left|v^\top B\big(D_L W_L \cdots D_a W_a u\big)\right| \geq \frac{\sqrt{s\log m}}{\sqrt{d}} \cdot O(\|u\|\|v\|)\right] \leq e^{-\Omega(s\log m)} \ .$$

Finally, one can take ϵ-net over all s-sparse vectors $u \in \mathbb{R}^m$ and all vectors $v \in \mathbb{R}^d$ and apply union bound.

(b) The proof is identical to Lemma 5.8.3c, except the fact that each entry of B follows from $\mathcal{N}(0, \frac{1}{d})$ instead of $\mathcal{N}(0, \frac{2}{m})$.

□

5.8.4 δ-Separateness

Lemma 5.8.5 (δ-separateness)**.** *Let $m \geq \Omega\big(\frac{L\log(nL)}{\delta^6}\big)$. There exists some constant $C > 1$ so that, if $\delta \leq \frac{1}{CL}$, $\|x_1\| = \cdots = \|x_n\| = 1$ and $\|x_i - x_j\| \geq \delta$ for every pair $i, j \in [n]$, then with probability at least $1 - e^{-\Omega(\delta^6 m/L)}$, we have :*

$$\forall i \neq j \in [n], \quad \forall \ell \in \{0, 1, \ldots, L\}\colon \|(I - \frac{h_{i,\ell}h_{i,\ell}^\top}{\|h_{i,\ell}\|^2})h_{j,\ell}\| \geq \frac{\delta}{2}.$$

Proof of Lemma 5.8.5. We first apply Lemma 5.8.1 to show that $\|h_{i,\ell}\| \in [1 - \delta^3/10, 1 + \delta^3/10]$. Next we prove Lemma 5.8.5 by induction.

In the base case of $\ell = -1$, since $\|x_i - x_j\| \geq \delta$ by our Assumption 5.2.1 and without loss of generality $\|x_i\| = 1$ and $(x_i)_\mathfrak{d} = \frac{1}{\sqrt{2}}$, we already have

$$\|(I - \frac{h_{i,\ell}h_{i,\ell}^\top}{\|h_{i,\ell}\|^2})h_{j,\ell}\|^2 = \|(I - \frac{x_i x_i^\top}{\|x_i\|^2})x_j\|^2 = \|x_j - x_i \cdot \langle x_i, x_j\rangle\|^2 = 1 - \big(\langle x_i, x_j\rangle\big)^2 \geq \frac{3}{4}\delta^2 \ .$$

Suppose $h_{i,\ell-1}$ and $h_{j,\ell-1}$ are fixed and satisfies $\|(I - \frac{h_{i,\ell-1}h_{i,\ell-1}^\top}{\|h_{i,\ell-1}\|^2})h_{j,\ell-1}\|^2 \geq \delta_{\ell-1}^2$ for some $\delta_{\ell-1} \geq \delta/2$. We write $W_\ell h_{i,\ell-1} = \vec{g}_1$ where $\vec{g}_1 \sim N(0, \frac{2\|h_{i,\ell-1}\|^2}{m} I)$.

Denoting by $\widehat{h} = h_{i,\ell-1}/\|h_{i,\ell-1}\|$, we can write $W_\ell h_{j,\ell-1} = W_\ell \widehat{h}\widehat{h}^\top h_{j,\ell-1} + W_\ell\big(I - \widehat{h}\widehat{h}^\top\big)h_{j,\ell-1}$ and the randomness of the two terms are independent. In particular, we can write

$$W_\ell h_{j,\ell-1} = \frac{\langle h_{i,\ell-1}, h_{j,\ell-1}\rangle}{\|h_{i,\ell-1}\|^2} \cdot \vec{g}_1 + \|\big(I - \widehat{h}\widehat{h}^\top\big)h_{j,\ell-1}\| \cdot \vec{g}_2 \tag{5.6}$$

where $\vec{g}_2 \sim \mathcal{N}(0, \frac{2}{m}I)$ is independent of g_1. Applying Claim 5.8.6 for each coordinate $k \in [m]$ (and re-scaling by $\frac{m}{\|h_{i,\ell-1}\|^2}$, we have

$$\mathbb{E}[(\phi(W_\ell h_{i,\ell-1}) - \phi(W_\ell h_{j,\ell-1}))_k^2] \geq \left(\frac{\delta_{\ell-1}}{\|h_{i,\ell-1}\|}\right)^2 \left(1 - \frac{\delta_{\ell-1}}{\|h_{i,\ell-1}\|}\right) \cdot \frac{\|h_{i,\ell-1}\|^2}{m} \geq \frac{\delta_{\ell-1}^2\,(1 - O(\delta_{\ell-1}))}{m}$$

Applying Chernoff bound (on independent subgaussian random variables), we have with probability at least $1 - e^{-\Omega(\delta_{\ell-1}^4 m)}$,[15]

$$\|h_{i,\ell} - h_{j,\ell}\|^2 = \|\phi(W_\ell h_{i,\ell-1}) - \phi(W_\ell h_{j,\ell-1})\|^2 \geq \delta_{\ell-1}^2 \left(1 - O(\delta_{\ell-1})\right) \ .$$

Since $\|h_{i,\ell}\|$ and $\|h_{j,\ell}\|$ are close to 1, we have

$$\begin{aligned}\left\| \left(I - \frac{h_{i,\ell} h_{i,\ell}^\top}{\|h_{i,\ell}\|^2} \right) h_{j,\ell} \right\|^2 &= \|h_{j,\ell}\|^2 - \frac{\langle h_{i,\ell}, h_{j,\ell} \rangle^2}{\|h_{i,\ell}\|^2} \\ &= \|h_{j,\ell}\|^2 + \frac{\|h_{i,\ell} - h_{j,\ell}\|^2 - \|h_{i,\ell}\|^2 - \|h_{j,\ell}\|^2}{2\|h_{i,\ell}\|^2} \geq \delta_{\ell-1}^2 (1 - O(\delta_{\ell-1})) \ .\end{aligned}$$

□

5.8.4.1 Auxiliary Claim

The following mathematical fact is needed in the proof of Lemma 5.8.5. Its proof is by carefully integrating the PDF of Gaussian distribution.

Claim 5.8.6. *Given $g_1, g_2 \sim \mathcal{N}(0, 2)$, constant $\alpha \in \mathbb{R}$ and $\delta \in [0, \frac{1}{6}]$, we have*

$$\mathop{\mathbb{E}}_{g_1, g_2} \left[(\phi(g_1) - \phi(\alpha g_1 + \delta g_2))^2 \right] \geq \delta^2 (1 - \delta) \ .$$

Proof of Claim 5.8.6. We first tackle two easy cases.

Suppose $a < \frac{3}{4}$. If so, then with probability at least 0.3 we have $g_1 > 1$. If this happens, then with probability at least $1/2$ we have $g_2 < 0$. If both happens, we have

$$\phi(g_1) - \phi(\alpha g_1 + \delta g_2) = g_1 - \phi(\alpha g_1 + \delta g_2) \geq g_1 - \alpha g_1 \geq \frac{1}{4} \ .$$

[15] More specifically, we can let $X_k = m \left(\phi(W_\ell h_{i,\ell-1}) - \phi(W_\ell h_{j,\ell-1})\right)_k^2$ which is $O(1)$-subgaussian and let $X = X_1 + \cdots + X_m$. We have $\Pr[X \geq \mathbb{E}[X](1 - \delta_{\ell-1})] \geq 1 - e^{-\Omega(\delta_{\ell-1}^2 \mathbb{E}[X])}$.

Therefore, we have if $a < \frac{3}{4}$ then the expectation is at least 0.03. For similar reason, if $a > \frac{5}{4}$ we also have the expectation is at least 0.03. In the remainder of the proof, we assume $\alpha \in \left[\frac{3}{4}, \frac{5}{4}\right]$.

If $g_1 \geq 0$, we have

$$\begin{aligned} f(g_1) &\stackrel{\text{def}}{=} \mathop{\mathbb{E}}_{g_2}\left[(\phi(g_1) - \phi(\alpha g_1 + \delta g_2))^2 \,|\, g_1 \geq 0\right] \\ &= \int_0^\infty \frac{(x-g_1)^2 \exp\left(-\frac{(x-\alpha g_1)^2}{4\delta^2}\right)}{\sqrt{4\pi\delta^2}} \,\mathrm{d}x \\ &= \frac{(\alpha-2)\delta g_1 e^{-\frac{\alpha^2 g_1^2}{4\delta^2}}}{\sqrt{\pi}} + \frac{1}{2}\left((\alpha-1)^2 g_1^2 + 2\delta^2\right)\left(\mathrm{erf}\left(\frac{\alpha g_1}{2\delta}\right) + 1\right). \end{aligned}$$

If $g_1 < 0$, we have

$$\begin{aligned} f(g_1) &\stackrel{\text{def}}{=} \mathop{\mathbb{E}}_{g_2}\left[(\phi(g_1) - \phi(\alpha g_1 + \delta g_2))^2 \,|\, g_1 < 0\right] \\ &= \int_0^\infty \frac{x^2 \exp\left(-\frac{(x-\alpha g_1)^2}{4\delta^2}\right)}{\sqrt{4\pi\delta^2}} \,\mathrm{d}x \\ &= \frac{1}{2}\left(\alpha^2 g_1^2 + 2\delta^2\right)\left(\mathrm{erf}\left(\frac{\alpha g_1}{2\delta}\right) + 1\right) + \frac{\alpha\delta g_1 e^{-\frac{\alpha^2 g_1^2}{4\delta^2}}}{\sqrt{\pi}}. \end{aligned}$$

Overall, we have

$$
\begin{aligned}
&\mathop{\mathbb{E}}_{g_1,g_2}\left[(\phi(g_1)-\phi(\alpha g_1+\delta g_2))^2\right] \\
&= \int_0^\infty \frac{f(g)\exp\left(-\frac{g^2}{4}\right)}{\sqrt{4\pi}}\,\mathrm{d}g + \int_{-\infty}^0 \frac{f(g)\exp\left(-\frac{g^2}{4}\right)}{\sqrt{4\pi}}\,\mathrm{d}g \\
&= \left(\frac{(\alpha-1)^2\alpha\delta}{\pi\left(\alpha^2+\delta^2\right)} + \frac{(\alpha-2)\delta^3}{\pi\left(\alpha^2+\delta^2\right)} + \frac{1}{2}\left((\alpha-1)^2+\delta^2\right) + \frac{1}{\pi}\left((\alpha-1)^2+\delta^2\right)\arctan\left(\frac{\alpha}{\delta}\right)\right) \\
&\quad + \frac{1}{2\pi}\left(\pi\left(\alpha^2+\delta^2\right) - 2\left(\alpha^2+\delta^2\right)\arctan\left(\frac{\alpha}{\delta}\right) - 2\alpha\delta\right) \\
&= \frac{\delta\left(-2\alpha^2+\alpha-2\delta^2\right)}{\pi\left(\alpha^2+\delta^2\right)} + \frac{(1-2\alpha)\arctan\left(\frac{\alpha}{\delta}\right)}{\pi} + (\alpha-1)\alpha+\delta^2+\frac{1}{2} \\
&= \left(\alpha^2-2\alpha+1\right)+\delta^2+\frac{2}{\pi}\sum_{k=1}^{\infty}(-1)^k\frac{(\alpha+k)\delta^{2k+1}}{(2k+1)\alpha^{2k+1}}.
\end{aligned}
$$

It is easy to see that, as long as $\delta \le \alpha$, we always have $\frac{(\alpha+k)\delta^{2k+1}}{(2k+1)\alpha^{2k+1}} \ge \frac{(\alpha+k+1)\delta^{2k+3}}{(2k+3)\alpha^{2k+3}}$. Therefore

$$
\mathop{\mathbb{E}}_{g_1,g_2}\left[(\phi(g_1)-\phi(\alpha g_1+\delta g_2))^2\right] \ge \left(\alpha^2-2\alpha+1\right)+\delta^2-\frac{2}{\pi}\frac{(\alpha+1)\delta^3}{3\alpha^3} \ge \delta^2(1-\delta)\ . \quad \square
$$

5.9 Stability against Adversarial Weight Perturbations

Let A, B and $\overrightarrow{W}^{(0)} = (\widetilde{W}_1, \ldots, \widetilde{W}_L)$ be matrices at random initialization (see Definition 5.2.2), and throughout this section, we consider (adversarially) perturbing $\overrightarrow{W}$ by $\overrightarrow{W}' = (W'_1, \ldots, W'_L)$ satisfying $\|\overrightarrow{W}'\|_2 \leq \omega$ (meaning, $\|W'_\ell\|_2 \leq \omega$ for every $\ell \in [L]$). We stick to the following notations in this section

Definition 5.9.1.

$$\begin{aligned}
&\widetilde{g}_{i,0} = Ax_i && g_{i,0} = Ax_i && \text{for } i \in [n] \\
&\widetilde{h}_{i,0} = \phi(Ax_i) && h_{i,0} = \phi(Ax_i) && \text{for } i \in [n] \\
&\widetilde{g}_{i,\ell} = \widetilde{W}_\ell h_{i,\ell-1} && g_{i,\ell} = (\widetilde{W}_\ell + W'_\ell)h_{i,\ell-1} && \text{for } i \in [n] \text{ and } \ell \in [L] \\
&\widetilde{h}_{i,\ell} = \phi(\widetilde{W}_\ell h_{i,\ell-1}) && h_{i,\ell} = \phi((\widetilde{W}_\ell + W'_\ell)h_{i,\ell-1}) && \text{for } i \in [n] \text{ and } \ell \in [L]
\end{aligned}$$

Define diagonal matrices $\widetilde{D}_{i,\ell} \in \mathbb{R}^{m\times m}$ and $D_{i,\ell} \in \mathbb{R}^{m\times m}$ by letting $(\widetilde{D}_{i,\ell})_{k,k} = \mathbb{1}_{(\widetilde{g}_{i,\ell})_k \geq 0}$ and $(D_{i,\ell})_{k,k} = \mathbb{1}_{(g_{i,\ell})_k \geq 0}, \forall k \in [m]$. Accordingly, we let $g'_{i,\ell} = g_{i,\ell} - \widetilde{g}_{i,\ell}$, $h'_{i,\ell} = h_{i,\ell} - \widetilde{h}_{i,\ell}$, and diagonal matrix $D'_{i,\ell} = D_{i,\ell} - \widetilde{D}_{i,\ell}$.

5.9.1 Forward Perturbation

Lemma 5.9.1 (forward perturbation)**.** *Suppose $\omega \le \frac{1}{CL^{9/2}\log^3 m}$ for some sufficiently large constant $C > 1$. With probability at least $1 - e^{-\Omega(m\omega^{2/3}L)}$, for every $\overrightarrow{W}'$ satisfying $\|\overrightarrow{W}'\|_2 \le \omega$,*

(a) $g'_{i,\ell}$ can be written as $g'_{i,\ell} = g'_{i,\ell,1} + g'_{i,\ell,2}$ where $\|g'_{i,\ell,1}\| \le O(\omega L^{3/2})$ and $\|g'_{i,\ell,2}\|_\infty \le O\left(\frac{\omega L^{5/2}\sqrt{\log m}}{\sqrt{m}}\right)$

(b) $\|D'_{i,\ell}\|_0 \le O(m\omega^{2/3}L)$ and $\|D'_{i,\ell}g_{i,\ell}\| \le O(\omega L^{3/2})$.

(c) $\|g'_{i,\ell}\|, \|h'_{i,\ell}\| \le O(\omega L^{5/2}\sqrt{\log m})$.

Proof of Lemma 5.9.1. In our proof below, we drop the subscript with respect to i for notational simplicity, and one can always take a union bound over all possible indices i at the end.

Using Lemma 5.8.1, we can first assume that $\|\widetilde{h}_\ell\|, \|\widetilde{g}_\ell\| \in [\frac{2}{3}, \frac{4}{3}]$ for all ℓ. This happens with probability at least $1 - e^{-\Omega(m/L)}$. We also assume $\left\|\prod_{b=\ell}^{a+1} \widetilde{W}_b \widetilde{D}_{b-1}\right\|_2 \le c_1\sqrt{L}$ where $c_1 > 0$ is the hidden constant in Lemma 5.8.3a.

We shall inductively prove Lemma 5.9.1. In the base case $\ell = 0$, we have $g'_\ell = 0$ so all the statements holds. In the remainder of the proof, we assume that Lemma 5.9.1 holds for $\ell - 1$ and we shall prove the three statements for layer ℓ. To help the readers understand how the constants propagate without blowing up, we shall prove $\|g'_{i,\ell,1}\| \le 4c_1 L^{1.5}\omega$ in Lemma 5.9.1a without the big-O notation, while for all other terms we use big-O to hide polynomial dependency on c_1.[16]

[16] Alternatively, one can fully specify all the constants without using the big-O notation. This was done in our prior work [AZLS18] but is notation-heavy. We refrain from doing so in this simplified paper.

We first carefully rewrite:

$$\begin{aligned}
g'_\ell &= (\widetilde{W}_\ell + W'_\ell)(\widetilde{D}_{\ell-1} + D'_{\ell-1})(\widetilde{g}_{\ell-1} + g'_{\ell-1}) - \widetilde{W}_\ell \widetilde{D}_{\ell-1}\widetilde{g}_{\ell-1} \\
&= W'_\ell(\widetilde{D}_{\ell-1} + D'_{\ell-1})(\widetilde{g}_{\ell-1} + g'_{\ell-1}) + \widetilde{W}_\ell D'_{\ell-1}(\widetilde{g}_{\ell-1} + g'_{\ell-1}) + \widetilde{W}_\ell \widetilde{D}_{\ell-1} g'_{\ell-1} \\
&= \cdots \\
&= \sum_{a=1}^{\ell} \Big(\prod_{b=\ell}^{a+1} \widetilde{W}_b \widetilde{D}_{b-1} \Big) \Big(\underbrace{W'_a(\widetilde{D}_{a-1} + D'_{a-1})(\widetilde{g}_{a-1} + g'_{a-1})}_{(\diamondsuit)} + \underbrace{\widetilde{W}_a D'_{a-1}(\widetilde{g}_{a-1} + g'_{a-1})}_{(\heartsuit)} \Big)
\end{aligned}$$

For each term in $(\diamondsuit)$, we have

$$\begin{aligned}
&\Big\| \Big(\prod_{b=\ell}^{a+1} \widetilde{W}_b \widetilde{D}_{b-1} \Big) \Big(W'_a(\widetilde{D}_{a-1} + D'_{a-1})(\widetilde{g}_{a-1} + g'_{a-1}) \Big) \Big\| \\
\le &\Big\| \prod_{b=\ell}^{a+1} \widetilde{W}_b \widetilde{D}_{b-1} \Big\|_2 \cdot \Big\| W'_a \Big\|_2 \cdot \Big\| \widetilde{D}_{a-1} + D'_{a-1} \Big\|_2 \cdot \Big\| \widetilde{g}_{a-1} + g'_{a-1} \Big\| \\
\overset{①}{\le} &\, c_1 \cdot \omega \cdot 1 \cdot \Big\| \widetilde{g}_{a-1} + g'_{a-1} \Big\| \overset{②}{\le} 2c_1\sqrt{L}\omega + O\big(\omega^2 L^3 \sqrt{\log m}\big) \ .
\end{aligned}$$

Above, inequality ① uses Lemma 5.8.3a and $\|\widetilde{D}_{a-1} + D'_{a-1}\|_2 = \|D_{a-1}\|_2 \le 1$; and inequality ② has used $\|\widetilde{g}_\ell\| \le 2$ and our inductive assumption Lemma 5.9.1c. By triangle inequality, we have

$$g'_\ell = \overrightarrow{\mathsf{err}}_1 + \sum_{a=1}^{\ell} \Big(\prod_{b=\ell}^{a+1} \widetilde{W}_b \widetilde{D}_{b-1} \Big) \Big(\underbrace{\widetilde{W}_a D'_{a-1}(\widetilde{g}_{a-1} + g'_{a-1})}_{(\heartsuit)} \Big)$$

where $\|\overrightarrow{\mathsf{err}}_1\| \le 2c_1 L^{1.5}\omega + O\big(\omega^2 L^4 \sqrt{\log m}\big)$. We next look at each term in $(\heartsuit)$. For each $a = 2, 3, \ldots, \ell$, we let

$$x \overset{\text{def}}{=} D'_{a-1}(\widetilde{g}_{a-1} + g'_{a-1}) = D'_{a-1}(\widetilde{W}_{a-1}\widetilde{h}_{a-1} + g'_{a-1}) \ .$$

If we re-scale x by $\frac{1}{\|\widetilde{h}_{a-1}\|}$ (which is a constant in $[0.75, 1.5]$), we can apply Claim 5.9.2 (with parameter choices in Corollary 5.9.3) on x and this tells us, with probability at least

$1 - e^{-\Omega(m\omega^{2/3}L)}$:

$$\|x\|_0 \le O(m\omega^{2/3}L) \quad \text{and} \quad \|x\| \le O(\omega L^{3/2}). \tag{5.7}$$

Next, each term in (♡) contributes to g'_ℓ by

$$y = \Big(\prod_{b=\ell}^{a+1} \widetilde{W}_b \widetilde{D}_{b-1}\Big)\widetilde{W}_a\Big(D'_{a-1}(\widetilde{g}_{a-1} + g'_{a-1})\Big)$$

using (5.7) and Claim 5.9.4 (with $s = O(m\omega^{2/3}L)$), we have with probability at least $1 - e^{-\Omega(s\log m)}$, one can write $y = y_1 + y_2$ for

$$\|y_1\| \le O\big(\omega L^{3/2} \cdot L^{1/2}\omega^{1/3}\log m\big) \quad \text{and} \quad \|y_2\|_\infty \le O\Big(\omega L^{3/2} \cdot \frac{\sqrt{\log m}}{\sqrt{m}}\Big) \ .$$

And therefore by triangle inequality we can write

$$g'_\ell = \overrightarrow{\mathsf{err}}_1 + \overrightarrow{\mathsf{err}}_2 + \overrightarrow{\mathsf{err}}_3$$

where $\|\overrightarrow{\mathsf{err}}_2\| \le O\big(L \cdot \omega L^{3/2} \cdot L^{1/2}\omega^{1/3}\log m\big) = O\big(\omega^{4/3}L^3 \log m\big)$ and $\|\overrightarrow{\mathsf{err}}_3\|_\infty \le O\Big(L \cdot \omega L^{3/2} \cdot \frac{\sqrt{\log m}}{\sqrt{m}}\Big)$. Together with the upper bound on $\overrightarrow{\mathsf{err}}_1$, we have

$$\|\overrightarrow{\mathsf{err}}_1 + \overrightarrow{\mathsf{err}}_2\| \le 2c_1 L^{1.5}\omega + O\big(\omega^2 L^4\sqrt{\log m} + \omega^{4/3}L^3\log m\big) \ .$$

We emphasize that the above big-O notion can hide polynomial dependency on c_1. Nevertheless, when ω is sufficiently small, the above term is at most $4c_1L^{1.5}\omega$. This finishes the proof of Lemma 5.9.1a for layer ℓ without blowing up the constant. Finally,

- Lemma 5.9.1b is due to (5.7),
- g'_ℓ part of Lemma 5.9.1c is a simple corollary of Lemma 5.9.1a, and
- h'_ℓ part of Lemma 5.9.1c is due to $h'_\ell = D_\ell g'_\ell + D'_\ell g_\ell$ together with the bound on $\|g'_\ell\|$ and the bound on $D'_\ell g_\ell$ from Lemma 5.9.1b. □

5.9.1.1 Auxiliary Claim

Claim 5.9.2. *Suppose $\delta_2 \in [0, O(1)]$ and $\delta_\infty \in [0, \frac{1}{4\sqrt{m}}]$. Suppose $\widetilde{W} \in \mathbb{R}^{m\times m}$ is a random matrix with entries drawn i.i.d. from $\mathcal{N}\big(0, \frac{2}{m}\big)$. With probability at least $1 - e^{-\Omega(m^{3/2}\delta_\infty)}$, the following holds. Fix any unit vector $\widetilde{h} \in \mathbb{R}^m$, and for all $g' \in \mathbb{R}^m$ that can be written as*

$$g' = g'_1 + g'_2 \text{ where } \|g'_1\| \le \delta_2 \text{ and } \|g'_2\|_\infty \le \delta_\infty.$$

Let $D' \in \mathbb{R}^{m\times m}$ be the diagonal matrix where $(D')_{k,k} = \mathbb{1}_{(\widetilde{W}\widetilde{h}+g')_k \ge 0} - \mathbb{1}_{(\widetilde{W}\widetilde{h})_k \ge 0}, \forall k \in [m]$. Then, letting $x = D'(\widetilde{W}\widetilde{h} + g') \in \mathbb{R}^m$, we have

$$\|x\|_0 \le \|D'\|_0 \le O(m(\delta_2)^{2/3} + \delta_\infty m^{3/2}) \quad \text{and} \quad \|x\| \le O(\delta_2 + (\delta_\infty)^{3/2} m^{3/4}) \ .$$

Corollary 5.9.3. *In particular, if $\omega L^{3/2} \le O(1)$, then with probability at least $1 - e^{-\Omega(m\omega^{2/3}L)}$, for every $g' = g'_1 + g'_2$ with $\|g'_1\| \le O(\omega L^{3/2})$ and $\|g'_2\|_\infty \le O\big(\frac{\omega^{2/3}L}{m^{1/2}}\big)$, it satisfies*

$$\|D'\|_0 \le O(m\omega^{2/3}L) \quad \text{and} \quad \|x\| \le O(\omega L^{3/2}) \ .$$

Proof of Claim 5.9.2. We first observe $\widetilde{g} = \widetilde{W}\widetilde{h}$ follows from $\mathcal{N}\big(0, \frac{2I}{m}\big)$ regardless of the choice of $\widetilde{h}$. Therefore, in the remainder of the proof, we just focus on the randomness of $\widetilde{g}$.

We also observe that $(D')_{j,j}$ is non-zero for some diagonal $j \in [m]$ only if

$$|(g'_1 + g'_2)_j| > |(\widetilde{g})_j| \ . \tag{5.8}$$

Let $\xi \le \frac{1}{2\sqrt{m}}$ be a parameter to be chosen later. We shall make sure that $\|g'_2\|_\infty \le \xi/2$.

- We denote by $S_1 \subseteq [m]$ the index sets where j satisfies $|(\widetilde{g})_j| \le \xi$. Since we know $(\widetilde{g})_j \sim \mathcal{N}(0, 2/m)$, we have $\Pr[|(\widetilde{g})_j| \le \xi] \le O\left(\xi\sqrt{m}\right)$ for each $j \in [m]$. Using Chernoff bound for all $j \in [m]$, we have with probability at least $1 - e^{-\Omega(m^{3/2}\xi)}$,

$$|S_1| = |\{i \in [m] \colon |(\widetilde{g})_j| \le \xi\}| \le O(\xi m^{3/2}) \ .$$

Now, for each $j \in S_1$ such that $x_j \neq 0$, we must have $|x_j| = |(\widetilde{g}+g_1'+g_2')_j| \leq |(g_1')_j|+2\xi$ so we can calculate the ℓ_2 norm of x on S_1:

$$\sum_{i\in S_1} x_j^2 \leq O(\|g_1'\|^2 + \xi^2|S_1|) \leq O(\|g_1'\|^2 + \xi^3 m^{3/2}) \ .$$

- We denote by $S_2 \subseteq [m] \setminus S_1$ the index set of all $j \in [m] \setminus S_1$ where $x_j \neq 0$. Using (5.8), we have for each $j \in S_2$:

$$|(g_1')_j| \geq |(\widetilde{g})_j| - |(g_2')_j| \geq \xi - \|g_2'\|_\infty \geq \xi/2 \ .$$

This means

$$|S_2| \leq \frac{4\|g_1'\|^2}{\xi^2} \ .$$

Now, for each $j \in S_2$ where $x_j \neq 0$, we know that the signs of $(\widetilde{g} + g_1' + g_2')_j$ and $(\widetilde{g})_j$ are opposite. Therefore, we must have

$$|x_j| = |(\widetilde{g} + g_1' + g_2')_j| \leq |(g_1' + g_2')_j| \leq |(g_1')_j| + \xi/2 \leq 2|(g_1')_j|$$

and therefore

$$\sum_{j\in S_2} x_j^2 \leq 4\sum_{j\in S_2} (g_1')_j^2 \leq 4\|g_1'\|^2 \ .$$

From above, we have $\|x\|_0 \leq |S_1| + |S_2| \leq O\big(\xi m^{3/2} + \frac{(\delta_2)^2}{\xi^2}\big)$ and $\|x\|^2 \leq O\big((\delta_2)^2 + \xi^3 m^{3/2}\big)$. Choosing $\xi = \max\{2\delta_\infty, \Theta(\frac{(\delta_2)^{2/3}}{m^{1/2}})\}$ for the former, and choosing $\xi = 2\delta_\infty$ for the latter, we have the desired result. □

Claim 5.9.4. *For any $2 \leq a \leq b \leq L$ and any positive integer $s \leq O\big(\frac{m}{L\log m}\big)$, with probability at least $1 - e^{-\Omega(s\log m)}$, for all $x \in \mathbb{R}^m$ with $\|x\| \leq 1$ and $\|x\|_0 \leq s$, letting $y = \widetilde{W}_b\widetilde{D}_{b-1}\widetilde{W}_{b-1}\cdots\widetilde{D}_a\widetilde{W}_a x$, we can write $y = y_1 + y_2$ with*

$$\|y_1\| \leq O\big(\sqrt{s/m}\log m\big) \quad \textit{and} \quad \|y_2\|_\infty \leq \frac{2\sqrt{\log m}}{\sqrt{m}} \ .$$

Proof of Claim 5.9.4. First of all, fix any x, we can let $u = \widetilde{D}_{b-1}\widetilde{W}_{b-1}\cdots\widetilde{D}_a\widetilde{W}_a x$ and the same proof of Lemma 5.8.3 implies that with probability at least $1 - e^{-\Omega(m/L)}$ we have $\|u\| \le O(\|x\|)$. We next condition on this event happens.

Let $\beta = \sqrt{\log m}/\sqrt{m}$. If u is fixed and using only the randomness of W_b, we have $y_i \sim \mathcal{N}\big(0, \frac{2\|u\|^2}{m}\big)$ so for every $p \ge 1$, by Gaussian tail bound

$$\Pr[|y_i| \ge \beta p] \le e^{-\Omega(\beta^2 p^2 m/\|x\|^2)} \le e^{-\Omega(\beta^2 p^2 m)} \ .$$

As long as $\beta^2 p^2 m \ge \beta^2 m \ge \Omega(\log m)$, we know that if $|y_i| \ge \beta p$ occurs for q/p^2 indices i out of $[m]$, this cannot happen with probability more than

$$\binom{m}{q/p^2} \times \left(e^{-\Omega(\beta^2 p^2 m)}\right)^{q/p^2} \le e^{\frac{q}{p^2}\left(O(\log m) - \Omega(\beta^2 p^2 m)\right)} \le e^{-\Omega(\beta^2 q m)} \ .$$

In other words,

$$\Pr\left[\left|\{i \in [m] \colon |y_i| \ge \beta p\}\right| > q/p^2\right] \le e^{-\Omega(\beta^2 q m)} \ .$$

Finally, by applying union bound over $p = 1, 2, 4, 8, 16, \ldots$ we have with probability $\ge 1 - e^{-\Omega(\beta^2 q m)} \cdot \log q$,

$$\sum_{i\colon |y_i| \ge \beta} y_i^2 \le \sum_{k=0}^{\lceil \log q \rceil} (2^{k+1}\beta)^2 \left|\left\{i \in [m] \colon |y_i| \ge 2^k \beta\right\}\right| \le \sum_{k=0}^{\lceil \log q \rceil} (2^{k+1}\beta)^2 \cdot \frac{q}{2^{2k}} \le O(q\beta^2 \log q) \quad (5.9)$$

In other words, vector y can be written as $y = y_1 + y_2$ where $\|y_2\|_\infty \le \beta$ and $\|y_1\|^2 \le O(q\beta^2 \log q)$.

Finally, we want to take ϵ-net over all s-sparse inputs x. This requires $\beta^2 q m \ge \Omega(s \log m)$, so we can choose $q = \Theta\big(\frac{s \log m}{m\beta^2}\big) = \Theta(s)$. □

5.9.2 Intermediate Layers

Lemma 5.9.5 (intermediate perturbation)**.** *For any integer s with $1 \le s \le O\big(\frac{m}{L^3 \log m}\big)$, with probability at least $1 - e^{-\Omega(s \log m)}$ over the randomness of $\overrightarrow{W}^{(0)}, A$,*

- *for every $i \in [n], 1 \le a \le b \le L$,*
- *for every diagonal matrices $D''_{i,0}, \ldots, D''_{i,L} \in [-3,3]^{m \times m}$ with at most s non-zero entries.*
- *for every perturbation matrices $W'_1, \ldots, W'_L \in \mathbb{R}^{m \times m}$ with $\|\overrightarrow{W}'\|_2 \le \omega \in [0,1]$.*

we have

(a) $\|\widetilde{W}_b(\widetilde{D}_{i,b-1} + D''_{i,b-1}) \cdots (\widetilde{D}_{i,a} + D''_{i,a})\widetilde{W}_a\|_2 \le O(\sqrt{L})$.

(b) $\|(\widetilde{W}_b + W'_b)(\widetilde{D}_{i,b-1} + D''_{i,b-1}) \cdots (\widetilde{D}_{i,a} + D''_{i,a})(\widetilde{W}_a + W'_a)\|_2 \le O(\sqrt{L})$ *if* $\omega \le O(\frac{1}{L^{1.5}})$.

Proof. For notational simplicity we ignore subscripts in i in the proofs.

(a) Note that each D''_ℓ can be written as $D''_\ell = D_\ell^{0/1} D''_\ell D_\ell^{0/1}$, where each $D_\ell^{0/1}$ is a diagonal matrix satisfying

$$(D_\ell^{0/1})_{k,k} = \begin{cases} 1, & (D''_\ell)_{k,k} \neq 0; \\ 0, & (D''_\ell)_{k,k} = 0. \end{cases} \quad \text{and} \quad \|D_\ell^{0/1}\|_0 \le s \ .$$

In order to bound the spectral norm of $\widetilde{W}_b(\widetilde{D}_{b-1} + D''_{b-1})\widetilde{W}_{b-1} \cdots (\widetilde{D}_a + D''_a)\widetilde{W}_a$, by triangle inequality, we can expend it into 2^{b-a} matrices and bound their spectral norms individually. Each such matrix can be written as (ignoring the subscripts)

$$(\widetilde{W}\widetilde{D} \cdots \widetilde{W} D^{0/1}) D'' (D^{0/1} \widetilde{W}\widetilde{D} \cdots \widetilde{W} D^{0/1}) D'' \cdots D'' (D^{0/1} \widetilde{W}\widetilde{D} \cdots \widetilde{W}) \tag{5.10}$$

Therefore, it suffices for us to bound the spectral norm of the following four types of matrices:

- $\widetilde{W}\widetilde{D}\cdots\widetilde{W}D^{0/1}$, such matrix has spectral norm at most 2 owing to Lemma 5.8.3b;
- $D^{0/1}\widetilde{W}\widetilde{D}\cdots\widetilde{W}$, such matrix has spectral norm at most $O(1)$ owing to Lemma 5.8.3c;
- $D^{0/1}\widetilde{W}\widetilde{D}\cdots\widetilde{W}D^{0/1}$, such matrix has spectral norm at most $\frac{1}{100L^{1.5}}$ owing to Lemma 5.8.3d and our choice $s \le O(\frac{m}{L^3 \log m})$;
- D'', such matrix has spectral norm at most 3.

Together, we have

$$\begin{aligned}&\left\|\widetilde{W}_b(\widetilde{D}_{b-1}+D''_{b-1})\widetilde{W}_{b-1}\cdots(\widetilde{D}_a+D''_a)\widetilde{W}_a\right\| \\ \le O(\sqrt{L}) + &\sum_{j=1}^{b-a}\binom{b-a}{j}\cdot O(1)\cdot\left(\frac{1}{100L^{1.5}}\right)^{j-1}\cdot 3^j\cdot O(1) \le O(\sqrt{L})\ .\end{aligned}$$

(b) In order to bound the spectral norm of $(\widetilde{W}_b+W'_b)(\widetilde{D}_{b-1}+D''_{b-1})\cdots(\widetilde{D}_a+D''_a)(\widetilde{W}_a+W'_a)$, by triangle inequality, we can expend it into 2^{b-a+1} matrices in terms of W' and bound their spectral norms individually. Each such matrix can be written as (ignoring the subscripts, and denoting $\breve{D}=\widetilde{D}+D'$)

$$(\widetilde{W}\breve{D}\cdots\widetilde{W}\breve{D})W'(\breve{D}\widetilde{W}\cdots\widetilde{W}\breve{D})\cdots W'(\breve{D}\widetilde{W}\cdots\breve{D}\widetilde{W})$$

Moreover, from Lemma 5.9.5a, we know the following three types of matrices

- $\widetilde{W}\breve{D}\cdots\widetilde{W}\breve{D}$,
- $\breve{D}\widetilde{W}\cdots\widetilde{W}\breve{D}$, and
- $\breve{D}\widetilde{W}\cdots\breve{D}\widetilde{W}$

all have spectral norm at most $O(\sqrt{L})$. Together, using $\|W'_\ell\|_2 \leq O(\frac{1}{L^{1.5}})$, we have

$$\begin{aligned}
&\left\|(\widetilde{W}_b + W'_b)(\widetilde{D}_{b-1} + D''_{b-1})(\widetilde{W}_{b-1} + W'_{b-1}) \cdots (\widetilde{D}_a + D''_a)(\widetilde{W}_a + W'_a)\right\| \\
&\leq \sum_{j=0}^{b-a+1} \binom{b-a+1}{j} \cdot \Big(O(\sqrt{L})\Big)^{j+1} \cdot \Big(O(\frac{1}{L^{1.5}})\Big)^{j} \leq O(\sqrt{L}) \ . \qquad \square
\end{aligned}$$

5.9.3 Backward

Lemma 5.9.6 (backward perturbation)**.** *For any integer* $s \in \left[\Omega\left(\frac{d}{\log m}\right), O\left(\frac{m}{L^3 \log m}\right)\right]$*, for* $d \leq O\left(\frac{m}{L \log m}\right)$*, with probability at least* $1 - e^{-\Omega(s \log m)}$ *over the randomness of* $\overrightarrow{W}^{(0)}, A, B$*,*

- *for all* $i \in [n]$*,* $a = 1, 2, \ldots, L+1$*,*
- *for every diagonal matrices* $D''_{i,0}, \ldots, D''_{i,L} \in [-3, 3]^{m \times m}$ *with at most* s *non-zero entries,*
- *for every perturbation matrices* $W'_{i,1}, \ldots, W'_{i,L} \in \mathbb{R}^{m \times m}$ *with* $\|\overrightarrow{W}'\|_2 \leq \omega = O(\frac{1}{L^{1.5}})$*,*

it satisfies $\|B(\widetilde{D}_{i,L}+D''_{i,L})(\widetilde{W}_L+W'_L)\cdots(\widetilde{W}_{a+1}+W'_{a+1})(\widetilde{D}_{i,a}+D''_{i,a})-B\widetilde{D}_{i,L}\widetilde{W}_L\cdots\widetilde{W}_{a+1}\widetilde{D}_{i,a}\|_2 \leq O\left(\frac{\sqrt{L^3 s \log m+\omega^2 L^3 m}}{\sqrt{d}}\right)$*. Note that if* $s = O(m\omega^{2/3}L)$*, this upper bound becomes* $O\left(\frac{\omega^{1/3}L^2\sqrt{m \log m}}{\sqrt{d}}\right)$*.*

Proof. For notational simplicity we ignore subscripts in i in the proofs.

Ignoring the subscripts for cleanness, we have

$$
\begin{aligned}
&\left\|B(\widetilde{D}_{i,L} + D''_{i,L})(\widetilde{W}_L + W'_L)\cdots(\widetilde{W}_{a+1} + W'_{a+1})(\widetilde{D}_{i,a} + D''_{i,a}) - B\widetilde{D}_{i,L}\widetilde{W}_L\cdots\widetilde{W}_{a+1}\widetilde{D}_{i,a}\right\|_2 \\
&\leq \sum_{\ell=a}^{L} \underbrace{\left\|B\widetilde{D}_{i,L}\widetilde{W}_L\cdots\widetilde{W}_{\ell+1}D_\ell^{0/1}\right\|_2}_{\text{Lemma 5.8.4a}} \|D''_\ell\|_2 \underbrace{\left\|D_\ell^{0/1}(\widetilde{W}_\ell + W'_\ell)\cdots(\widetilde{D}_{i,a} + D''_{i,a})\right\|_2}_{\text{Lemma 5.9.5b}} \\
&\quad + \sum_{\ell=a+1}^{L} \underbrace{\left\|B\widetilde{D}_{i,L}\widetilde{W}_L\cdots\widetilde{W}_{\ell+1}\widetilde{D}_\ell\right\|_2}_{\text{Lemma 5.8.4b}} \|W'_\ell\|_2 \underbrace{\left\|(\widetilde{D}_{\ell-1} + D''_{\ell-1})(\widetilde{W}_{\ell-1} + W'_{\ell-1})\cdots(\widetilde{D}_{i,a} + D''_{i,a})\right\|_2}_{\text{Lemma 5.9.5b}} \\
&\leq L \cdot O\left(\frac{\sqrt{s \log m}}{\sqrt{d}}\right) \cdot O(\sqrt{L}) + L \cdot O(\sqrt{m/d}) \cdot \omega \cdot O(\sqrt{L})
\end{aligned}
$$

□

5.10 Gradient Bound at Random Initialization

Throughout this section we assume $\overrightarrow{W}, A$ and B are randomly generated according to Definition 5.2.2. The diagonal sign matrices $D_{i,\ell}$ are also determined according to this random initialization.

Recall we have defined $\mathsf{Back}_{i,\ell} \stackrel{\text{def}}{=} BD_{i,L}W_L \cdots D_{i,\ell}W_\ell \in \mathbb{R}^{d\times m}$. In this section, we introduce the following notion

Definition 5.10.1. For any vector tuple $\vec{\mathsf{v}} = (\mathsf{v}_1, \ldots, \mathsf{v}_n) \in (\mathbb{R}^d)^n$ (viewed as a fake loss vector), for each $\ell \in [L]$, we define

$$\widehat{\nabla}^{\vec{\mathsf{v}}}_{[W_\ell]_k} F(\overrightarrow{W}) \stackrel{\text{def}}{=} \sum_{i=1}^n (\mathsf{Back}_{i,\ell+1}^\top \mathsf{v}_i)_k \cdot h_{i,\ell-1} \cdot \mathbb{1}_{\langle [W_\ell]_k, h_{i,\ell-1}\rangle \geq 0}, \forall k \in [m]$$

$$\widehat{\nabla}^{\vec{\mathsf{v}}}_{W_\ell} F(\overrightarrow{W}) \stackrel{\text{def}}{=} \sum_{i=1}^n \widehat{\nabla}^{\vec{\mathsf{v}}}_{W_\ell} F_i(\overrightarrow{W}) \quad \text{where} \quad \widehat{\nabla}^{\vec{\mathsf{v}}}_{W_\ell} F_i(\overrightarrow{W}) \stackrel{\text{def}}{=} D_{i,\ell}(\mathsf{Back}_{i,\ell+1}^\top \mathsf{v}_i) h_{i,\ell-1}^\top$$

Remark 5.10.1. It is an easy exercise to check that, if letting $\vec{\mathsf{v}} = (\mathsf{v}_1, \ldots, \mathsf{v}_n)$ where $\mathsf{v}_i = Bh_{i,L} - y_i^*$, then $\widehat{\nabla}^{\vec{\mathsf{v}}}_{[W_\ell]_k} F(\overrightarrow{W}) = \nabla_{[W_\ell]_k} F(\overrightarrow{W})$ and $\widehat{\nabla}^{\vec{\mathsf{v}}}_{W_\ell} F_i(\overrightarrow{W}) = \nabla_{W_\ell} F_i(\overrightarrow{W})$.

Our main lemma of this section is the following.

Lemma 5.10.1 (gradient bound at random initialization)**.** *Fix any $\vec{\mathsf{v}} \in (\mathbb{R}^d)^n$, with probability at least $1 - e^{-\Omega(\delta m/n)}$ over the randomness of $A, \overrightarrow{W}, B$, it satisfies for every $\ell \in [L]$:*

$$\|\widehat{\nabla}^{\vec{\mathsf{v}}}_{W_\ell} F_i(\overrightarrow{W})\|_F^2 \leq O\Big(\frac{\|\mathsf{v}_i\|^2}{d} \times m\Big) \qquad \|\widehat{\nabla}^{\vec{\mathsf{v}}}_{W_\ell} F(\overrightarrow{W})\|_F^2 \leq O\Big(\frac{\|\vec{\mathsf{v}}\|^2}{d} \times mn\Big)$$

$$\|\widehat{\nabla}^{\vec{\mathsf{v}}}_{W_L} F(\overrightarrow{W})\|_F^2 \geq \Omega\Big(\frac{\max_{i\in[n]} \|\mathsf{v}_i\|^2}{dn/\delta} \times m\Big)$$

5.10.1 Proof of Lemma 5.10.1: Upper Bound

For each $i \in [n], \ell \in [L]$, we can calculate that

$$\begin{aligned}
\left\| \widehat{\nabla}^{\vec{\mathsf{v}}}_{W_\ell} F_i(\overrightarrow{W}) \right\|_F &= \left\| D_{i,\ell}(\mathsf{Back}_{i,\ell+1}^\top \cdot \mathsf{v}_i) \cdot h_{i,\ell-1}^\top \right\|_F \\
&= \left\| D_{i,\ell}(\mathsf{Back}_{i,\ell+1}^\top \cdot \mathsf{v}_i) \right\|_2 \cdot \|h_{i,\ell-1}\|_2 \\
&\leq \|\mathsf{Back}_{i,\ell+1}\|_2 \cdot \|\mathsf{v}_i\|_2 \cdot \|h_{i,\ell-1}\|_2 \\
&\leq \|BW_L D_{L-1} \cdots D_{i,\ell+1} W_{\ell+1}\|_2 \cdot \|\mathsf{v}_i\|_2 \cdot \|h_{i,\ell-1}\|_2 \\
&\overset{①}{\leq} O(\sqrt{m/d}) \cdot O(1) \cdot \|\mathsf{v}_i\|_2 \ .
\end{aligned}$$

where inequality ① uses Lemma 5.8.4b and Lemma 5.8.1 with high probability. Applying triangle inequality with respect to all $\ell \in [L]$, taking square on both sides, and summing up over all $i \in [n]$ finish the proof.

5.10.2 Proof of Lemma 5.10.1: Lower Bound

Let $i^* = \operatorname{argmax}_{i\in[n]}\{\|\mathsf{v}_i\|\}$. Recall

$$\widehat{\nabla}^{\vec{\mathsf{v}}}_{[W_L]_k} F(\overrightarrow{W}) = \sum_{i=1}^{n} \langle B_k, \mathsf{v}_i\rangle \cdot h_{i,L-1} \cdot \mathbb{1}_{(W_L h_{i,L-1})_k \geq 0}$$

Let $\widehat{h} \stackrel{\text{def}}{=} \frac{h_{i^*,L-1}}{\|h_{i^*,L-1}\|}$. For analysis purpose, after $\widehat{h}$ is fixed (so after fixing the randomness of $A, W_1, \ldots, W_{L-1}$), we redefine $W_L\widehat{h} = \sqrt{1-\theta^2}\widehat{g}_1 + \theta\widehat{g}_2$ where $\widehat{g}_1$ and $\widehat{g}_2$ are generated independently from $\mathcal{N}(0, \frac{2I}{m})$. We can do so because the two sides are equal in distribution. In other words, we can set

$$W'_L \stackrel{\text{def}}{=} W_L\big(I - \widehat{h}\widehat{h}^\top\big) - \sqrt{1-\theta^2}\widehat{g}_1\widehat{h}^\top \text{ and } W''_L \stackrel{\text{def}}{=} \theta\widehat{g}_2\widehat{h}^\top,$$

then we have $W_L = W'_L + W''_L$. In particular, the randomness of W'_L and W''_L are *independent*.

In the remainder of the proof, let us choose $\theta \stackrel{\text{def}}{=} \frac{\delta}{5n} \leq \frac{1}{5}$.

We first make two technical claims, and the proof of the first one can be found in Section 5.10.2.1.

Claim 5.10.2. *We have* $\Pr_{W'_L, W_{L-1}, \ldots, W_1, A}\left[|N_2| \geq \frac{\delta}{40n}m\right] \geq 1 - e^{\Omega(\delta m/n)}$

$$N_2 \stackrel{\text{def}}{=} \left\{k \in [m]\colon \left(|(W'_L h_{i^*,L-1})_k| \leq \frac{\delta}{10n\sqrt{m}}\right) \bigwedge \left(\forall i \in [n]\setminus\{i^*\}, \quad \big|\big(W'_L h_{i,L-1}\big)_k\big| \geq \frac{\delta}{4n\sqrt{m}}\right)\right\}$$

Claim 5.10.3. *Given set* $N_2 \subset [m]$ *and* $\vec{\mathsf{v}}$*, we have*

$$\Pr_{B_k}\left[\left|\left\{k \in N_2\colon \big|\langle B_k, \mathsf{v}_{i^*}\rangle\big| \geq \frac{\|\mathsf{v}_{i^*}\|}{\sqrt{d}}\right\}\right| \geq \frac{|N_2|}{2}\right] \geq 1 - e^{-\Omega(|N_2|)}$$

Proof of Claim 5.10.3. Observe that each $\langle B_k, \mathsf{v}_{i^*}\rangle$ follows from $\mathcal{N}(0, \|\mathsf{v}_{i^*}\|^2/d)$, so with probability at least 0.68 it satisfies $|\langle B_k, \mathsf{v}_{i^*}\rangle| \geq \frac{\|\mathsf{v}_{i^*}\|}{\sqrt{d}}$. Using Chernoff bound we have the desired claim. □

Combining Claim 5.10.4 and Claim 5.10.3, we can obtain a set $N \subseteq [m]$ satisfying

$$N \stackrel{\text{def}}{=} \left\{ k \in [m] \colon \left(|(W_L' h_{i^*,L-1})_k| \le \frac{\delta}{10n\sqrt{m}} \right) \bigwedge \left(\forall i \in [n] \setminus \{i^*\}, \quad |\left(W_L' h_{i,L-1}\right)_k| \ge \frac{\delta}{4n\sqrt{m}} \right) \bigwedge |\langle B_k, \mathsf{v}_{i^*} \rangle| \ge \frac{\|\mathsf{v}_{i^*}\|}{\sqrt{d}} \right\}$$

of cardinality $|N| \ge \frac{\delta}{100n} m$. Let us fix the randomness of W_L' so that N is fixed. Let k be any index in N. We can write

$$\widehat{\nabla}^{\vec{\mathsf{v}}}_{[W_L]_k} F(\overrightarrow{W}) = \sum_{i=1}^{n} \langle B_k, \mathsf{v}_i \rangle \cdot h_{i,L-1} \cdot \mathbb{1}_{(W_L' h_{i,L-1})_k + (W_L'' h_{i,L-1})_k \ge 0}.$$

The only remaining source of randomness comes from $W_L'' = \theta \widehat{g}_2 \widehat{h}^\top$.

Recalling that $\theta = \frac{1}{5n}$ and $\widehat{g}_2 \sim \mathcal{N}(0, \frac{2}{m} I)$, so since $\theta(\widehat{g}_2)_k \sim \mathcal{N}(0, \frac{2\theta^2}{m})$, using numerical values of Gaussian CDF, one can verify that

$$\Pr_{\widehat{g}_2} \left[|\theta(\widehat{g}_2)_k| \in \left(\frac{\delta}{9n\sqrt{m}}, \frac{\delta}{5n\sqrt{m}} \right) \right] \ge 0.2 \ .$$

Let us denote this event of $\widehat{g}_2$ as $\mathfrak{E}_k$. Conditioning on $\mathfrak{E}_k$ happens, recalling $\|h_{i,L-1}\| \in [0.9, 1.1]$ from Lemma 5.8.1,

- For every $i \in [n] \setminus \{i^*\}$, we have

  $$|(W_L'' h_{i,L-1})_k| = |(\theta \widehat{g}_2 \widehat{h}^\top h_{i,L-1})_k| \le |(\theta \widehat{g}_2)_k| \cdot \|h_{i,L-1}\| < \frac{\delta}{5n\sqrt{m}} \cdot 1.1 < |(W_L' h_{i,L-1})_k|$$

  and this means $\mathbb{1}_{(W_L h_{i,L-1})_k \ge 0} = \mathbb{1}_{(W_L' h_{i,L-1})_k \ge 0}$.

- For $i = i^*$, we have

  $$|(W_L'' h_{i^*,L-1})_k| = |(\theta \widehat{g}_2 \widehat{h}^\top h_{i^*,L-1})_k| = |(\theta \widehat{g}_2)_k| \cdot \|h_{i^*,L-1}\| > \frac{\delta}{9n\sqrt{m}} \cdot 0.9 > |(W_L' h_{i^*,L-1})_k|$$

  and this means $\mathbb{1}_{(W_L h_{i^*,L-1})_k \ge 0} \ne \mathbb{1}_{(W_L' h_{i^*,L-1})_k \ge 0}$ with probability exactly $\frac{1}{2}$ — this is because, conditioning on event $\mathfrak{E}_k$, the sign of $(\theta \widehat{g}_2)_k$ is ± 1 each with half probability.

Recall that for every $k \in N$,

$$\widehat{\nabla}^{\vec{\mathsf{v}}}_{[W_L]_k} F(\overrightarrow{W}) = \underbrace{\langle B_k, \mathsf{v}_{i^*} \rangle \cdot h_{i^*,L-1} \cdot \mathbb{1}_{(W_L h_{i^*,L-1})_k \geq 0}}_{\spadesuit} + \sum_{i \in [n] \setminus \{i^*\}} \underbrace{\langle B_k, \mathsf{v}_i \rangle \cdot h_{i,L-1} \cdot \mathbb{1}_{(W_L h_{i,L-1})_k \geq 0}}_{\clubsuit}$$

Now, fix the randomness of $A, B, W_1, \ldots, W_{L-1}, W'_L$ and let $\widehat{g}_2$ be the only randomness. Conditioning on $\mathfrak{E}_k$, we have that each term in $\clubsuit$ is fixed (i.e., independent of $\widehat{g}_2$) because $\mathbb{1}_{(W_L h_{i,L-1})_k \geq 0} = \mathbb{1}_{(W'_L h_{i,L-1})_k \geq 0}$. In contrast, conditioning on $\mathfrak{E}_k$, the indicator $\mathbb{1}_{(W_L h_{i^*,L-1})_k \geq 0}$ of the $\spadesuit$ term may be 1 or 0 each with half probability. This means,

$$\Pr_{(\widehat{g}_2)_k} \left[\|\widehat{\nabla}^{\vec{\mathsf{v}}}_{[W_L]_k} F(\overrightarrow{W})\|^2 \geq |\langle B_k, \mathsf{v}_{i^*} \rangle|^2 \cdot \|h_{i^*,L-1}\|^2 \,\middle|\, k \in N \wedge \mathfrak{E}_k \right] \geq \frac{1}{2} \ .$$

Taking into account the fact that $|\langle B_k, \mathsf{v}_{i^*} \rangle| \geq \frac{\|\mathsf{v}_{i^*}\|}{\sqrt{d}}$ (by definition of N), the fact that $\|h_{i,L-1}\| \geq 0.9$, and the fact that $\Pr_{(\widehat{g}_2)_k}[\mathfrak{E}] \geq 0.2$, we have

$$\Pr_{(\widehat{g}_2)_k} \left[\|\widehat{\nabla}^{\vec{\mathsf{v}}}_{[W_L]_k} F(\overrightarrow{W})\|^2 \geq 0.8 \frac{\|\mathsf{v}_{i^*}\|^2}{d} \,\middle|\, k \in N \right] \geq \frac{1}{10} \ .$$

Using the independence of $(\widehat{g}_2)_k$ with respect to different $k \in N$, we can apply Chernoff bound and derive:

$$\Pr_{\widehat{g}_2} \left[\sum_{k \in N} \|\widehat{\nabla}^{\vec{\mathsf{v}}}_{[W_L]_k} F(\overrightarrow{W})\|^2 \geq 0.8 \frac{\|\mathsf{v}_{i^*}\|^2}{d} \cdot \frac{|N|}{15} \,\middle|\, N \right] \geq 1 - e^{-\Omega(|N|)} \ .$$

Finally, using and $|N| \geq \frac{\delta}{100n} m$, we have

$$\Pr \left[\|\widehat{\nabla}^{\vec{\mathsf{v}}}_{W_L} F(\overrightarrow{W})\|_F^2 \geq \frac{\|\mathsf{v}_{i^*}\|^2}{d} \frac{\delta}{2000n} m \right] \geq 1 - e^{-\Omega(\delta m/n)} \ .$$

We finish the upper bound proof of Lemma 5.10.1. ■

5.10.2.1 Proof of Claim 5.10.4

Claim 5.10.4. *We have* $\Pr_{W_L', W_{L-1}, \ldots, W_1, A}\left[|N_2| \geq \frac{\delta}{40n} m\right] \geq 1 - e^{\Omega(\delta m/n)}$

$$N_2 \stackrel{\text{def}}{=} \left\{ k \in [m] \colon \left(|(W_L' h_{i^*, L-1})_k| \leq \frac{\delta}{10n\sqrt{m}} \right) \bigwedge \left(\forall i \in [n] \setminus \{i^*\}, \quad |\left(W_L' h_{i, L-1}\right)_k| \geq \frac{\delta}{4n\sqrt{m}} \right) \right\}$$

Proof of Claim 5.10.4. Throughout the proof we assume $W_{L-1}, \ldots, A$ are good enough so that Lemma 5.8.1 holds (for $\epsilon = 0.01$) and we fix their randomness. Define

$$N_1 \stackrel{\text{def}}{=} \left\{ k \in [m] \colon |(W_L' h_{i^*, L-1})_k| \leq \frac{\delta}{10n\sqrt{m}} \right\}$$

Since $\|h_{i^*, L-1}\|^2 \leq 1.1$ by Lemma 5.8.1, and since by definition of W_L' we have $(W_L' h_{i^*, L-1})_k \sim \mathcal{N}(0, \frac{2(1-\theta^2)\|h_{i^*, L-1}\|^2}{m})$. By standard properties of Gaussian CDF (see Fact 5.10.5), we know $|(W_L' h_{i^*, L-1})_k| \leq \frac{\delta}{10n\sqrt{m}}$ with probability at least $\frac{\delta}{25n}$ for each $k \in [m]$. By Chernoff bound,

$$\Pr_{W_L'}\left[|N_1| \geq \frac{\delta}{30n} m \right] \geq 1 - e^{-\Omega(\delta m/n)}$$

Next, suppose we fix the randomness of $W_L' \widehat{h}$. Define

$$N_2 \stackrel{\text{def}}{=} \left\{ k \in N_1 \colon \forall i \in [n] \setminus \{i^*\}, \quad |\left(W_L' h_{i, L-1}\right)_k| \geq \frac{\delta}{4n\sqrt{m}} \right\}$$

For each $k \in N_1$ and $i \in [n] \setminus \{i^*\}$, we can write

$$W_L' h_{i, L-1} = W_L' \widehat{h} (\widehat{h}^\top h_{i, L-1}) + W_L' (I - \widehat{h}\widehat{h}^\top) h_{i, L-1} \ .$$

Above, the first term on the right hand side is fixed (because we have fixed the randomness of $W_L' \widehat{h}$); however, $W_L'(I - \widehat{h}\widehat{h}^\top) h_{i, L-1}$ is still fresh new random Gaussian. In symbols,

$$W_L' h_{i, L-1} \sim \mathcal{N}\left(W_L' \widehat{h}\widehat{h}^\top h_{i, L-1}, \frac{2\|(I - \widehat{h}\widehat{h}^\top) h_{i, L-1}\|^2}{m} I \right) \ .$$

According to Lemma 5.8.5, the variance here is at least $\frac{2}{m}\|(I - \widehat{h}\widehat{h}^\top)h_{i,L-1}\|^2 \geq \frac{\delta^2}{2m}$. Using standard properties of Gaussian CDF (see Fact 5.10.5), we know $|(W_L' h_{i,L-1})_k| \geq \frac{\delta}{4n\sqrt{m}}$ with probability at least $1 - \frac{1}{8n}$ for each $k \in [m]$. By union bound, for this $k \in [m]$, with probability at least $\frac{7}{8}$ we know $|(W_L' h_{i,L-1})_k| \geq \frac{\delta}{4n\sqrt{m}}$ for all $i \in [n] \setminus \{i^*\}$. By Chernoff bound (over all $k \in N_1$), we conclude that

$$\Pr_{W_L'}\left[|N_2| \geq \frac{3}{4}|N_1| \,\middle|\, N_1\right] \geq 1 - e^{-\Omega(|N_1|)} = 1 - e^{-\Omega(\delta m/n)} \ .$$

Combining the two bounds we finish the proof. □

Fact 5.10.5. *Suppose $x \sim \mathcal{N}(0, \sigma^2)$ is a Gaussian random variable. For any $t \in (0, \sigma)$ we have*

$$\Pr[x \geq t] \in \left[\tfrac{1}{2}(1 - \tfrac{4}{5}\tfrac{t}{\sigma}), \tfrac{1}{2}(1 - \tfrac{2}{3}\tfrac{t}{\sigma})\right].$$

Similarly, if $x \sim \mathcal{N}(\mu, \sigma^2)$, for any $t \in (0, \sigma)$, we have

$$\Pr[|x| \geq t] \in \left[1 - \tfrac{4}{5}\tfrac{t}{\sigma}, 1 - \tfrac{2}{3}\tfrac{t}{\sigma}\right].$$

5.11 Theorem 5.11.1: Gradient Bound at After Perturbation

In this section we prove our main theorem on the gradient upper and lower bounds.

Theorem 5.11.1 (gradient bound, restated)**.** *Let* $\omega \stackrel{\text{def}}{=} O\big(\frac{\delta^{3/2}}{n^{9/2}L^6\log^3 m}\big)$*. With probability at least* $1-e^{-\Omega(m\omega^{2/3}L)}$ *over the randomness of* $\overrightarrow{W}^{(0)}, A, B$*, it satisfies for every* $\ell\in[L]$*, every* $i\in[n]$*, and every* $\overrightarrow{W}$ *with* $\|\overrightarrow{W}-\overrightarrow{W}^{(0)}\|_2\le\omega$*,*

$$\|\nabla_{W_\ell}F_i(\overrightarrow{W})\|_F^2 \le O\Big(\frac{F_i(\overrightarrow{W})}{d}\times m\Big) \qquad \|\nabla_{W_\ell}F(\overrightarrow{W})\|_F^2 \le O\Big(\frac{F(\overrightarrow{W})}{d}\times mn\Big)$$
$$\|\nabla_{W_L}F(\overrightarrow{W})\|_F^2 \ge \Omega\Big(\frac{\max_{i\in[n]}F_i(\overrightarrow{W})}{dn/\delta}\times m\Big)\ .$$

Remark 5.11.1. Our Theorem 5.11.1 only gives gradient lower bound on $\|\nabla_{W_L}F(\overrightarrow{W})\|_F$. In principle, one can derive similar lower bounds on $\|\nabla_{W_\ell}F(\overrightarrow{W})\|_F$ for all $\ell=1,2,\dots,L-1$. However, the proof will be significantly more involved. We choose not to derive those bounds at the expense of losing a polynomial factor in L in the final running time. For readers interested in the techniques for obtaining those bounds, we refer to them to the "randomness decomposition" part of [AZLS18].

Proof of Theorem 5.11.1. Again we denote by $\widetilde{D}_{i,\ell}$ and $D_{i,\ell}$ respectively the sign matrix at the initialization $\overrightarrow{W}^{(0)}$ and at the current point $\overrightarrow{W}$; and by $\widetilde{h}_{i,\ell}$ and $h_{i,\ell}$ respectively the forward vector at $\overrightarrow{W}^{(0)}$ and at $\overrightarrow{W}$. Let us choose $s=O(m\omega^{2/3}L)$ which bounds the sparsity of $\|D_{i,\ell}-\widetilde{D}_{i,\ell}\|_0$ by Lemma 5.9.1b. Recall

$$\begin{aligned}&\widehat{\nabla}^{\mathsf{v}}_{W_\ell}F(\overrightarrow{W}^{(0)})-\widehat{\nabla}^{\mathsf{v}}_{W_\ell}F(\overrightarrow{W})\\ =&\sum_{i=1}^n\Big(\big(\mathsf{v}_i^\top B\widetilde{D}_{i,L}\widetilde{W}_L\cdots\widetilde{W}_{\ell+1}\widetilde{D}_{i,\ell}\big)^\top(\widetilde{h}_{i,\ell-1})^\top-\big(\mathsf{v}_i^\top BD_{i,L}W_L\cdots W_{\ell+1}D_{i,\ell}\big)^\top(h_{i,\ell-1})^\top\Big)\end{aligned} \tag{5.11}$$

By Lemma 5.9.6, we know that

$$\|\mathsf{v}_i^\top B\widetilde{D}_{i,L}\widetilde{W}_L\cdots\widetilde{D}_{i,a}\widetilde{W}_a\widetilde{D}_{i,a-1}-\mathsf{v}_i^\top BD_{i,L}W_L\cdots D_{i,a}W_aD_{i,a-1}\|\le O(\omega^{1/3}L^2\sqrt{m\log m}/\sqrt{d})\cdot\|\mathsf{v}_i\|$$

By Lemma 5.8.4b we know

$$\|\mathsf{v}_i^\top B\widetilde{D}_{i,L}\widetilde{W}_L\cdots\widetilde{D}_{i,a}\widetilde{W}_a\widetilde{D}_{i,a-1}\|\le O(\sqrt{m/d})\cdot\|\mathsf{v}_i\|$$

By Lemma 5.8.1 and Lemma 5.9.1c, we have

$$\|h_{i,\ell-1}\|\le 1.1\quad\text{and}\quad\|h_{i,\ell-1}-\widetilde{h}_{i,\ell-1}\|\le O(\omega L^{5/2}\sqrt{\log m})$$

Together, they imply

$$\begin{aligned}\Big\|\widehat{\nabla}^{\vec{\mathsf{v}}}_{W_\ell}F(\overrightarrow{W}^{(0)})-\widehat{\nabla}^{\vec{\mathsf{v}}}_{W_\ell}F(\overrightarrow{W})\Big\|_F^2&\le n\|\vec{\mathsf{v}}\|^2\cdot O\left(\omega^{1/3}L^2\sqrt{m\log m}/\sqrt{d}+\sqrt{m/d}\times\omega L^{5/2}\sqrt{\log m}\right)^2\\&\le n\|\vec{\mathsf{v}}\|^2\cdot O\left(\frac{m\log m}{d}\cdot\omega^{2/3}L^4\right)\quad.\end{aligned}\tag{5.12}$$

With our parameter assumption on ω, this together with Lemma 5.10.1 implies the same upper and lower bounds at point $\overrightarrow{W}=\overrightarrow{W}^{(0)}+\overrightarrow{W}'$:

$$\|\widehat{\nabla}^{\vec{\mathsf{v}}}_{W_\ell}F_i(\overrightarrow{W}^{(0)}+\overrightarrow{W}')\|_F^2\le O\Big(\frac{\|\mathsf{v}_i\|^2}{d}\times m\Big)\quad\|\widehat{\nabla}^{\vec{\mathsf{v}}}_{W_\ell}F(\overrightarrow{W}^{(0)}+\overrightarrow{W}')\|_F^2\le O\Big(\frac{\|\vec{\mathsf{v}}\|^2}{d}\times mn\Big)$$
$$\|\widehat{\nabla}^{\vec{\mathsf{v}}}_{W_L}F(\overrightarrow{W}^{(0)}+\overrightarrow{W}')\|_F^2\ge\Omega\Big(\frac{\max_{i\in[n]}\|\mathsf{v}_i\|^2}{dn/\delta}\times m\Big)\quad.$$

Finally, taking ϵ-net over all possible vectors $\vec{\mathsf{v}}=(\mathsf{v}_1,\dots,\mathsf{v}_n)\in(\mathbb{R}^d)^n$, we know that the above bounds hold not only for fixed $\vec{\mathsf{v}}$ but for all $\vec{\mathsf{v}}$. In particular, we can now plug in the choice of $\mathsf{v}_i=\mathsf{loss}_i=Bh_{i,L}-y_i^*$ and it implies our desired bounds on the true gradients. □

5.12 Theorem 5.12.1: Objective Semi-Smoothness

The purpose of this section is to prove

Theorem 5.12.1 (objective semi-smoothness, restated). *Let* $\omega \in \big[\Omega(\frac{d^{3/2}}{m^{3/2}L^{3/2}\log^{3/2} m}), O(\frac{1}{L^{4.5}\log^3 m})\big]$ *and* $\overrightarrow{W}^{(0)}, A, B$ *be at random initialization. With probability at least* $1 - e^{-\Omega(m\omega^{2/3}L)}$ *over the randomness of* $\overrightarrow{W}^{(0)}, A, B$*, we have for every* $\breve{\overrightarrow{W}} \in (\mathbb{R}^{m\times m})^L$ *with* $\|\breve{\overrightarrow{W}} - \overrightarrow{W}^{(0)}\|_2 \le \omega$*, and for every* $\overrightarrow{W}' \in (\mathbb{R}^{m\times m})^L$ *with* $\|\overrightarrow{W}'\|_2 \le \omega$*, we have*

$$F(\breve{\overrightarrow{W}} + \overrightarrow{W}') \le F(\breve{\overrightarrow{W}}) + \langle \nabla F(\breve{\overrightarrow{W}}), \overrightarrow{W}'\rangle + \sqrt{nF(\breve{\overrightarrow{W}})} \cdot \frac{\omega^{1/3}L^2\sqrt{m\log m}}{\sqrt{d}} \cdot O(\|\overrightarrow{W}'\|_2) + O\big(\frac{nL^2m}{d}\big)\|\overrightarrow{W}'\|_2^2$$

We introduce the following notations before we go to proofs.

Definition 5.12.1. For $i \in [n]$ and $\ell \in [L]$:

$$\begin{array}{lll}
\widetilde{g}_{i,0} = Ax_i & \breve{g}_{i,0} = Ax_i & g_{i,0} = Ax_i \\
\widetilde{h}_{i,0} = \phi(Ax_i) & \hbar_{i,0} = \phi(Ax_i) & h_{i,0} = \phi(Ax_i) \\
\widetilde{g}_{i,\ell} = \widetilde{W}_\ell \widetilde{h}_{i,\ell-1} & \breve{g}_{i,\ell} = \breve{W}_\ell \hbar_{i,\ell-1} & g_{i,\ell} = (\breve{W}_\ell + W'_\ell)h_{i,\ell-1} \\
\widetilde{h}_{i,\ell} = \phi(\widetilde{W}_\ell \widetilde{h}_{i,\ell-1}) & \hbar_{i,\ell} = \phi(\breve{W}_\ell \hbar_{i,\ell-1}) & h_{i,\ell} = \phi((\breve{W}_\ell + W'_\ell)h_{i,\ell-1}) \\
 & \breve{\mathsf{loss}}_i = B\hbar_{i,L} - y_i^* &
\end{array}$$

Define diagonal matrices $\widetilde{D}_{i,\ell} \in \mathbb{R}^{m\times m}$ and $\breve{D}_{i,\ell} \in \mathbb{R}^{m\times m}$ respectively by letting

$$(\widetilde{D}_{i,\ell})_{k,k} = \mathbb{1}_{(\widetilde{g}_{i,\ell})_k \ge 0} \text{ and } (\breve{D}_{i,\ell})_{k,k} = \mathbb{1}_{(\breve{g}_{i,\ell})_k \ge 0}, \forall k \in [m].$$

The following claim gives rise to a new recursive formula to calculate $h_{i,\ell} - \hbar_{i,\ell}$.

Claim 5.12.2. *There exist diagonal matrices $D''_{i,\ell} \in \mathbb{R}^{m\times m}$ with entries in $[-1,1]$ such that,*

$$\forall i \in [n], \forall \ell \in [L]: \quad h_{i,\ell} - \hbar_{i,\ell} = \sum_{a=1}^{\ell} (\breve{D}_{i,\ell} + D''_{i,\ell}) \breve{W}_\ell \cdots \breve{W}_{a+1} (\breve{D}_{i,a} + D''_{i,a}) W'_a h_{i,a-1} \tag{5.13}$$

Furthermore, we have $\|h_{i,\ell} - \hbar_{i,\ell}\| \le O(L^{1.5})\|W'\|_2$, $\|Bh_{i,\ell} - B\hbar_{i,\ell}\| \le O(L\sqrt{m/d})\|W'\|_2$ and $\|D''_{i,\ell}\|_0 \le O(m\omega^{2/3}L)$.

Proof of Theorem 5.12.1. First of all, since

$$\frac{1}{2}\|Bh_{i,L} - y_i^*\|^2 = \frac{1}{2}\|\breve{\mathsf{loss}}_i + B(h_{i,L} - \hbar_{i,L})\|^2 = \frac{1}{2}\|\breve{\mathsf{loss}}_i\|^2 + \breve{\mathsf{loss}}_i^\top B(h_{i,L} - \hbar_{i,L}) + \frac{1}{2}\|B(h_{i,L} - \hbar_{i,L})\|^2 \tag{5.14}$$

we can write

$$\begin{aligned}
&F(\breve{\overrightarrow{W}} + \overrightarrow{W}') - F(\overrightarrow{W}) - \langle \nabla F(\overrightarrow{W}), \overrightarrow{W}' \rangle \\
&\overset{①}{=} -\langle \nabla F(\breve{\overrightarrow{W}}), \overrightarrow{W}' \rangle + \frac{1}{2}\sum_{i=1}^n \|Bh_{i,L} - y^*_{i,L}\|^2 - \|B\hbar_{i,L} - y^*_{i,L}\|^2 \\
&\overset{②}{=} -\langle \nabla F(\breve{\overrightarrow{W}}), \overrightarrow{W}' \rangle + \sum_{i=1}^n \breve{\mathsf{loss}}_i^\top B(h_{i,L} - \hbar_{i,L}) + \frac{1}{2}\|B(h_{i,L} - \hbar_{i,L})\|^2 \\
&\overset{③}{=} \sum_{i=1}^n \breve{\mathsf{loss}}_i^\top B \left((h_{i,L} - \hbar_{i,L}) - \sum_{\ell=1}^L \breve{D}_{i,L}\breve{W}_L \cdots \breve{W}_{\ell+1}\breve{D}_{i,\ell} W'_\ell \hbar_{i,\ell-1} \right) + \frac{1}{2}\|B(h_{i,L} - \hbar_{i,L})\|^2 \\
&\overset{④}{=} \sum_{i=1}^n \breve{\mathsf{loss}}_i^\top B \left(\sum_{\ell=1}^L (\breve{D}_{i,L} + D''_{i,L})\breve{W}_L \cdots \breve{W}_{\ell+1}(\breve{D}_{i,\ell} + D''_{i,\ell}) W'_\ell h_{i,\ell-1} - \breve{D}_{i,L}\breve{W}_L \cdots \breve{W}_{\ell+1}\breve{D}_{i,\ell} W'_\ell \hbar_{i,\ell-1} \right) \\
&\quad + \frac{1}{2}\sum_{i=1}^n \|B(h_{i,L} - \hbar_{i,L})\|^2
\end{aligned} \tag{5.15}$$

Above, ① is by the definition of $F(\cdot)$; ② is by (5.14); ③ is by the definition of $\nabla F(\cdot)$ (see Fact 5.2.3 for an explicit form of the gradient).

We next bound the RHS of (5.15). We first note that by Lemma 5.9.1b, we have $\|\breve{D}_{i,\ell} + D''_{i,\ell} - \widetilde{D}_{i,\ell}\|_0 \le s$ and $\|\breve{D}_{i,\ell} - \widetilde{D}_{i,\ell}\|_0 \le s$ for $s = O(m\omega^{2/3}L)$.

We ignore subscripts in i for notational convenience. We first use Claim 5.12.2 to get

$$\|B(h_L - \hbar_L)\| \le O(L\sqrt{m/d}) \cdot \|\overrightarrow{W}'\|_2 \ . \tag{5.16}$$

Next we calculate that

$$\begin{aligned}
&\left|\breve{\mathsf{loss}}_i^\top B(\breve{D}_L + D_L'')\breve{W}_L \cdots (\breve{D}_\ell + D_\ell'')W_\ell' h_{\ell-1} - \breve{\mathsf{loss}}_i^\top B\breve{D}_L\breve{W}_L \cdots \breve{D}_\ell W_\ell' h_{\ell-1}\right| \\
&\le \|\breve{\mathsf{loss}}_i\| \cdot \underbrace{\left\|B(\breve{D}_L + D_L'')\breve{W}_L \cdots \breve{W}_{\ell-1}(\breve{D}_\ell + D_\ell'') - B\breve{D}_L\breve{W}_L \cdots \breve{W}_{\ell-1}\breve{D}_\ell\right\|_2}_{\text{Lemma 5.9.6 with } s = O(m\omega^{2/3}L)} \cdot \|W_\ell' h_{\ell-1}\| \\
&\le \|\breve{\mathsf{loss}}_i\| \cdot O\left(\frac{\sqrt{L^3\omega^{2/3}Lm\log m}}{\sqrt{d}}\right) \cdot O(\|W_\ell'\|_2) \ .
\end{aligned} \tag{5.17}$$

Finally, we also have

$$\begin{aligned}
&\left|\breve{\mathsf{loss}}_i^\top B\breve{D}_L\breve{W}_L \cdots \breve{D}_\ell W_\ell'(h_{\ell-1} - \hbar_{\ell-1})\right| \\
&\overset{①}{\le} \|\breve{\mathsf{loss}}_i\| \cdot O\left(\sqrt{m/d} + \frac{\omega^{1/3}L^2\sqrt{m\log m}}{\sqrt{d}}\right) \cdot \|W_\ell'\|_2 \cdot \|h_\ell - \hbar_\ell\|_2 \\
&\overset{②}{\le} O(L^{0.5}\sqrt{m/d}) \cdot \|\breve{\mathsf{loss}}_i\|_2 \cdot L^{1.5}\|W_\ell'\|^2
\end{aligned} \tag{5.18}$$

where ① uses Lemma 5.8.4b (and Lemma 5.9.6 for bounding the perturbation) and ② uses Claim 5.12.2 to bound $\|h_\ell - \hbar_\ell\|_2$ and our choice of ω.

Putting (5.16), (5.17) and (5.18) back to (5.15), and using triangle inequality, we have the desired result. □

5.12.1 Proof of Claim 5.12.2

We first present a simple proposition about the ReLU function.

Proposition 5.12.3. *Given vectors $a, b \in \mathbb{R}^m$ and $D \in \mathbb{R}^{m \times m}$ the diagonal matrix where $D_{k,k} = \mathbb{1}_{a_k \geq 0}$. Then, then there exists a diagonal matrix $D'' \in \mathbb{R}^{m \times m}$ with*

- $|D_{k,k} + D''_{k,k}| \leq 1$ *and* $|D''_{k,k}| \leq 1$ *for every* $k \in [m]$,
- $D''_{k,k} \neq 0$ *only when* $\mathbb{1}_{a_k \geq 0} \neq \mathbb{1}_{b_k \geq 0}$, *and*
- $\phi(a) - \phi(b) = (D + D'')(a - b)$

Proof. We verify coordinate by coordinate for each $k \in [m]$.

- If $a_k \geq 0$ and $b_k \geq 0$, then $(\phi(a) - \phi(b))_k = a_k - b_k = \big(D(a-b)\big)_k$.
- If $a_k < 0$ and $b_k < 0$, then $(\phi(a) - \phi(b))_k = 0 - 0 = \big(D(a-b)\big)_k$.
- If $a_k \geq 0$ and $b_k < 0$, then $(\phi(a) - \phi(b))_k = a_k = (a_k - b_k) + \frac{b_k}{a_k - b_k}(a_k - b_k) = \big(D(a-b) + D''(a-b)\big)_k$, if we define $(D'')_{k,k} = \frac{b_k}{a_k - b_k} \in [-1, 0]$.
- If $a_k < 0$ and $b_k \geq 0$, then $(\phi(a) - \phi(b))_k = -b_k = 0 \cdot (a_k - b_k) - \frac{b_k}{b_k - a_k}(a_k - b_k) = \big(D(a-b) + D''(a-b)\big)_k$, if we define $(D'')_{k,k} = \frac{b_k}{b_k - a_k} \in [0, 1]$. □

Proof of Claim 5.12.2. We ignore the subscript in i for cleanness, and calculate that

$$\begin{aligned} h_\ell - \hbar_\ell &\overset{①}{=} \phi((\breve{W}_\ell + W'_\ell)h_{\ell-1}) - \phi(\breve{W}_\ell \hbar_{\ell-1}) \\ &\overset{②}{=} (\breve{D}_\ell + D''_\ell)\left((\breve{W}_\ell + W'_\ell)h_{\ell-1} - \breve{W}_\ell \hbar_{\ell-1}\right) \\ &= (\breve{D}_\ell + D''_\ell)\breve{W}_\ell(h_{\ell-1} - \hbar_{\ell-1}) + (\breve{D}_\ell + D''_\ell)W'_\ell h_{\ell-1} \\ &\overset{③}{=} \sum_{a=1}^{\ell} (\breve{D}_\ell + D''_\ell)\breve{W}_\ell \cdots \breve{W}_{a+1}(\breve{D}_a + D''_a)W'_a h_{a-1} \end{aligned}$$

Above, ① is by the recursive definition of h_ℓ and $\hbar_\ell$; ② is by Proposition 5.12.3 and D''_ℓ is defined according to Proposition 5.12.3; and inequality ③ is by recursively computing $h_{\ell-1} - \hbar_{\ell-1}$. As for the remaining properties:

- We have $\|D''_\ell\|_0 \le O(m\omega^{2/3}L)$.

  This is because, $(D''_\ell)_{k,k}$ is non-zero only at the coordinates $k \in [m]$ where the signs of $\breve{g}_\ell$ and g_ℓ are opposite (by Proposition 5.12.3). Such a coordinate k must satisfy either $(\widetilde{D}_\ell)_{k,k} \neq (\breve{D}_\ell)_{k,k}$ or $(\widetilde{D}_\ell)_{k,k} \neq (D_\ell)_{k,k}$, and therefore by Lemma 5.9.1b there are at most $O(m\omega^{2/3}L)$ such coordinates k.

- We have $\|h_\ell - \hbar_\ell\| \le O(L^{1.5})\|\overrightarrow{W}'\|_2$.

  This is because we have $\left\|(\breve{D}_\ell + D''_\ell)\breve{W}_\ell \cdots \breve{W}_{a+1}(\breve{D}_a + D''_a)\right\|_2 \le O(\sqrt{L})$ from Lemma 5.9.5b, we have $\|h_{a-1}\| \le O(1)$ (by $\|\widetilde{h}_{a-1}\| \le O(1)$ from Lemma 5.8.1 and $\|\widetilde{h}_{a-1} - h_{a-1}\| \le o(1)$ from Lemma 5.9.1c); and and $\|W'_a h_{a-1}\| \le \|W'_a\|_2 \|h_{a-1}\| \le O(\|\overrightarrow{W}'\|_2)$.

- We have $\|Bh_\ell - B\hbar_\ell\| \le O(L\sqrt{m/d})\|\overrightarrow{W}'\|_2$.

  This is because we have $\left\|B(\breve{D}_\ell + D''_\ell)\breve{W}_\ell \cdots \breve{W}_{a+1}(\breve{D}_a + D''_a)\right\|_2 \le O(\sqrt{m/d})$ from Lemma 5.8.4b (along with perturbation bound Lemma 5.9.6), we have $\|h_{a-1}\| \le O(1)$

(by $\|\widetilde{h}_{a-1}\| \leq O(1)$ from Lemma 5.8.1 and $\|\widetilde{h}_{a-1} - h_{a-1}\| \leq o(1)$ from Lemma 5.9.1c); and and $\|W'_a h_{a-1}\| \leq \|W'_a\|_2 \|h_{a-1}\| \leq O(\|\overrightarrow{W}'\|_2)$.

□

5.13 Theorem 5.13.1: Convergence Rate of GD

Theorem 5.13.1 (gradient descent, restated)**.** *For any $\epsilon \in (0,1]$, $\delta \in \big(0, O(\frac{1}{L})\big]$. Let $m \geq \widetilde{\Omega}\big(\operatorname{poly}(n, L, \delta^{-1})d\big)$, $\eta = \Theta\big(\frac{d\delta}{\operatorname{poly}(n,L)m}\big)$, and $\overrightarrow{W}^{(0)}, A, B$ are at random initialization. Then, with probability at least $1-e^{-\Omega(\log^2 m)}$, suppose we start at $\overrightarrow{W}^{(0)}$ and for each $t = 0, 1, \ldots, T-1$,*

$$\overrightarrow{W}^{(t+1)} = \overrightarrow{W}^{(t)} - \eta \nabla F(\overrightarrow{W}^{(t)}) \ .$$

Then, it satisfies

$$F(\overrightarrow{W}^{(T)}) \leq \epsilon \quad \textit{for} \quad T = \Theta\left(\frac{\operatorname{poly}(n,L)}{\delta^2} \log \frac{1}{\epsilon}\right) \ .$$

In other words, the training loss drops to ϵ in a linear convergence speed.

Proof of Theorem 5.13.1. Using Lemma 5.8.1 we have $\|h_{i,L}\|_2 \leq 1.1$ and then using the randomness of B, it is easy to show that $\|B\widetilde{h}_{i,L} - y_i^*\|^2 \leq O(\log^2 m)$ with at least $1-e^{-\Omega(\log^2 m)}$ (where $\widetilde{h}_{i,L}$ is defined with respect to the random initialization $\overrightarrow{W}^{(0)}$), and therefore

$$F(\overrightarrow{W}^{(0)}) \leq O(n \log^2 m) \ .$$

Let us assume for every $t = 0, 1, \ldots, T-1$, the following holds

$$\|\overrightarrow{W}^{(t)} - \overrightarrow{W}^{(0)}\|_F \leq \omega \stackrel{\text{def}}{=} O\left(\frac{n^3\sqrt{d}}{\delta\sqrt{m}} \log m\right) \ . \tag{5.19}$$

We shall prove the convergence of GD assuming (5.19) holds, so that previous statements such as Theorem 5.12.1 and Theorem 5.11.1 can be applied. At the end of the proof, we shall verify that (5.19) is satisfied.

To make the proof simple, we choose

$$m \geq \Omega\big(\frac{n^{24}L^{12}d\log^5 m}{\delta^8}\big),$$
$$\eta = \Theta\big(\frac{d\delta}{n^4L^2m}\big),$$
$$T = \Theta\left(\frac{n^6L^2}{\delta^2}\log\frac{1}{\epsilon}\right)$$

We emphasize that

- Most of the polynomial dependency in n, L, δ^{-1} come from the non-smoothness of the ReLU activation; if one instead studies smooth activations, their power can be significantly reduced. For instance, for smooth activation functions, one does not need the semi-smoothness Theorem 5.12.1.
- We have not tried to tighten the polynomial dependency on n, L, δ^{-1}. We are aware of many ways to improve the constant in the exponents at the expense of complicating the proofs. Since the main focus of this paper is to derive the first *polynomial* running time, we do not include such improvements.

Letting $\nabla_t = \nabla F(\overrightarrow{W}^{(t)})$, we calculate that

$$\begin{aligned}
&F(\overrightarrow{W}^{(t+1)}) \\
&\overset{①}{\leq} F(\overrightarrow{W}^{(t)}) - \eta\|\nabla F(\overrightarrow{W}^{(t)})\|_F^2 + \eta\sqrt{nF(\overrightarrow{W}^{(t)})}\cdot O\left(\frac{\omega^{1/3}L^2\sqrt{m\log m}}{\sqrt{d}}\right)\cdot\|\nabla_t\|_2 + O\big(\eta^2\frac{nL^2m}{d}\big)\|\nabla_t\|_2^2 \\
&\overset{②}{\leq} F(\overrightarrow{W}^{(t)}) - \eta\|\nabla F(\overrightarrow{W}^{(t)})\|_F^2 + O\left(\frac{\eta nL^2m\omega^{1/3}\sqrt{\log m}}{d} + \frac{\eta^2n^2L^2m^2}{d^2}\right)\cdot F(\overrightarrow{W}^{(t)}) \\
&\overset{③}{\leq} \left(1 - \Omega\Big(\frac{\eta\delta m}{dn^2}\Big)\right)F(\overrightarrow{W}^{(t)}) \ . \qquad (5.20)
\end{aligned}$$

Above, ① uses Theorem 5.12.1; ② uses Theorem 5.11.1 (which gives $\|\nabla_t\|_2^2 \le \max_{\ell\in[L]} \|\nabla_{W_\ell} F(\overrightarrow{W}^{(t)})\|_F^2 \le O\big(\frac{F(\overrightarrow{W}^{(t)})}{d} \times mn\big)$); ③ use gradient lower bound from Theorem 5.11.1 and our choice of η. In other words, after $T = \Theta(\frac{dn^2}{\eta\delta m}) \log \frac{n \log m}{\epsilon}$ iterations we have $F(\overrightarrow{W}^{(T)}) \le \epsilon$.

We need to verify for each t, $\|\overrightarrow{W}^{(t)} - \overrightarrow{W}^{(0)}\|_F$ is small so that (5.19) holds. By Theorem 5.11.1,

$$\begin{aligned}\|W_\ell^{(t)} - W_\ell^{(0)}\|_F &\le \sum_{i=0}^{t-1} \|\eta \nabla_{W_\ell} F(\overrightarrow{W}^{(i)})\|_F \le O(\eta\sqrt{nm/d}) \cdot \sum_{i=0}^{t-1} \sqrt{F(\overrightarrow{W}^{(i)})} \\ &\le O(\eta\sqrt{nm/d}) \cdot \Theta(\frac{dn^2}{\eta\delta m}) \cdot O(\sqrt{n\log^2 m}) \le O\left(\frac{n^3\sqrt{d}}{\delta\sqrt{m}} \log m\right) .\end{aligned}$$

where the last step follows by our choice of T. □

5.14 Theorem 5.14.1: Convergence Rate of SGD

Theorem 5.14.1 (stochastic gradient descent, stated)**.** *For any* $\epsilon \in (0,1]$, $\delta \in \big(0, O(\frac{1}{L})\big]$, $b \in [n]$. *Let* $m \geq \widetilde{\Omega}\big(\frac{\mathrm{poly}(n,L,\delta^{-1})\cdot d}{b}\big)$, $\eta \stackrel{\text{def}}{=} \Theta(\frac{b\delta d}{\mathrm{poly}(n,L)m\log^2 m})$, *and* $\overrightarrow{W}^{(0)}, A, B$ *are at random initialization. Suppose we start at* $W^{(0)}$ *and for each* $t = 0, 1, \ldots, T-1$,

$$W^{(t+1)} = W^{(t)} - \eta \cdot \frac{n}{|S_t|} \sum_{i \in S_t} \nabla F(W^{(t)})$$

(for a random subset $S_t \subseteq [n]$ of fixed cardinality b.)

Then, it satisfies with probability at least $1 - e^{-\Omega(\log^2 m)}$ *over the randomness of* $S_1, \ldots, S_T$*:*

$$F(W^{(T)}) \leq \epsilon \quad \textit{for all} \quad T = \Theta\Big(\frac{\mathrm{poly}(n,L)\log^2 m}{b\delta^2} \log \frac{n \log m}{\epsilon}\Big) \ .$$

The proof of Theorem 5.14.1 is the same as Theorem 5.13.1 plus the careful use of martingale concentration.

Proof of Theorem 5.14.1. Using similar argument as the proof of Theorem 5.13.1, we have with at least $1 - e^{-\Omega(\log^2 m)}$ probability

$$F(\overrightarrow{W}^{(0)}) \leq O(n \log^2 m) \ .$$

Let us assume for every $t = 0, 1, \ldots, T-1$, the following holds

$$\|\overrightarrow{W}^{(t)} - \overrightarrow{W}^{(0)}\|_F \leq \omega \stackrel{\text{def}}{=} O\left(\frac{n^{3.5}\sqrt{d}}{\delta\sqrt{bm}} \log m\right) \ . \tag{5.21}$$

We shall prove the convergence of SGD assuming (5.21) holds, so that previous statements such as Theorem 5.12.1 and Theorem 5.11.1 can be applied. At the end of the proof, we shall verify that (5.21) is satisfied throughout the SGD with high probability.

To make the proof simple, we choose

$$m \geq \Omega\big(\frac{n^{24}L^{12}bd\log^5 m}{\delta^8}\big),$$
$$\eta = \Theta\big(\frac{b\delta d}{n^5L^2m\log^2 m}\big),$$

$$T = \Theta\Big(\frac{dn^2}{\eta\delta m}\log\frac{n\log m}{\epsilon}\Big)$$
$$= \Theta\Big(\frac{n^7L^2\log^2 m}{b\delta^2}\log\frac{n\log m}{\epsilon}\Big)$$

We emphasize that

- Most of the polynomial dependency in n, L, δ^{-1} come from the non-smoothness of the ReLU activation; if one instead studies smooth activations, their power can be significantly reduced. For instance, for smooth activation functions, one does not need the semi-smoothness Theorem 5.12.1.
- We have not tried to tighten the polynomial dependency on n, L, δ^{-1}. We are aware of many ways to improve the constant in the exponents at the expense of complicating the proofs. Since the main focus of this paper is to derive the first *polynomial* running time, we do not include such improvements.

For each $t = 0, 1, \ldots, T-1$, using the same notation as Theorem 5.13.1, except that

we choose $\nabla_t = \frac{n}{|S_t|}\sum_{i\in S_t}\nabla F_i(\overrightarrow{W}^{(t)})$. We have $\mathbb{E}_{S_t}[\nabla_t] = \nabla F(\overrightarrow{W}^{(t)})$ and therefore

$$\begin{aligned}
&\mathop{\mathbb{E}}_{S_t}[F(\overrightarrow{W}^{(t+1)})] \\
&\overset{①}{\le} F(\overrightarrow{W}^{(t)}) - \eta\|\nabla F(\overrightarrow{W}^{(t)})\|_F^2 + \eta\sqrt{nF(\overrightarrow{W}^{(t)})}\cdot O\left(\frac{\omega^{1/3}L^2\sqrt{m\log m}}{\sqrt{d}}\right)\cdot \mathop{\mathbb{E}}_{S_t}[\|\nabla_t\|_2] \\
&\quad + O\big(\eta^2\frac{nL^2m}{d}\big)\mathop{\mathbb{E}}_{S_t}[\|\nabla_t\|_2^2] \\
&\overset{②}{\le} F(\overrightarrow{W}^{(t)}) - \eta\|\nabla_t\|_F^2 + O\left(\frac{\eta nL^2m\omega^{1/3}\sqrt{\log m}}{d} + \frac{\eta^2n^2L^2m^2}{d^2}\right)\cdot F(\overrightarrow{W}^{(t)}) \\
&\overset{③}{\le} \left(1 - \Omega\Big(\frac{\eta\delta m}{dn^2}\Big)\right)F(\overrightarrow{W}^{(t)}) \ . \qquad (5.22)
\end{aligned}$$

Above, ① uses Theorem 5.12.1 and $\mathbb{E}_{S_t}[\nabla_t] = \nabla F(\overrightarrow{W}^{(t)})$; ② uses Theorem 5.11.1 which give

$$\begin{aligned}
&\mathop{\mathbb{E}}_{S_t}\left[\|\nabla_t\|_2^2\right] \le \frac{n^2}{b}\mathop{\mathbb{E}}_{S_t}\left[\sum_{i\in S_t}\max_{\ell\in[L]}\left\|\nabla_{W_\ell}F_i(\overrightarrow{W}^{(t)})\right\|_F^2\right] \le O\Big(\frac{nmF(\overrightarrow{W}^{(t)})}{d}\Big) \\
&\mathop{\mathbb{E}}_{S_t}\left[\|\nabla_t\|_2\right] \le \left(\mathop{\mathbb{E}}_{S_t}\left[\|\nabla_t\|_2^2\right]\right)^{1/2} \le O\Big(\big(\frac{nmF(\overrightarrow{W}^{(t)})}{d}\big)^{1/2}\Big) \ ;
\end{aligned}$$

③ use gradient lower bound from Theorem 5.11.1 and our choice of η.

At the same time, we also have the following absolute value bound:

$$\begin{aligned}
F(\overrightarrow{W}^{(t+1)}) &\overset{①}{\le} F(\overrightarrow{W}^{(t)}) + \eta\|\nabla F(\overrightarrow{W}^{(t)})\|_F \cdot \|\nabla_t\|_F \\
&\quad + \eta\sqrt{nF(\overrightarrow{W}^{(t)})} \cdot O\left(\frac{\omega^{1/3}L^2\sqrt{m\log m}}{\sqrt{d}}\right) \cdot \|\nabla_t\|_2 + O\Big(\eta^2\frac{nL^2m}{d}\Big) \cdot \|\nabla_t\|_2^2 \\
&\overset{②}{\le} F(\overrightarrow{W}^{(t)}) + \eta \cdot O\left(\sqrt{\frac{LF(\overrightarrow{W}^{(t)})mn}{d}}\right) \cdot O\left(\sqrt{\frac{n^2mLF(\overrightarrow{W}^{(t)})}{bd}}\right) \\
&\quad + \eta\sqrt{nF(\overrightarrow{W}^{(t)})} \cdot O\left(\frac{\omega^{1/3}L^2\sqrt{m\log m}}{\sqrt{d}}\right) \cdot \frac{\sqrt{n^2mF(\overrightarrow{W}^{(t)})}}{\sqrt{bd}} \\
&\quad + O\Big(\eta^2\frac{nL^2m}{d}\Big) \cdot \frac{n^2}{b}O\Big(\frac{mF(\overrightarrow{W}^{(t)})}{d}\Big) \\
&\overset{③}{\le} \Big(1 + O\Big(\frac{\eta Lmn^{1.5}}{\sqrt{bd}} + \frac{\eta n^{1.5}\omega^{1/3}L^2m\sqrt{\log m}}{\sqrt{bd}} + \frac{\eta^2n^3L^2m^2}{d^2b}\Big)\Big)F(\overrightarrow{W}^{(t)}) \ . \qquad (5.23)
\end{aligned}$$

Above, ① uses Theorem 5.12.1 and Cauchy-Schwarz $\langle A, B\rangle \le \|A\|_F\|B\|_F$, and ② uses Theorem 5.11.1 which give

$$\|\nabla_t\|_2^2 \le \frac{n^2}{b}\left[\sum_{i\in S_t}\max_{\ell\in[L]}\Big\|\nabla_{W_\ell}F_i(\overrightarrow{W}^{(t)})\Big\|_F^2\right] \le \frac{n^2}{b}O\Big(\frac{mF(\overrightarrow{W}^{(t)})}{d}\Big)$$

$$\|\nabla_t\|_F^2 \le \frac{n^2}{b}\left[\sum_{i\in S_t}\sum_{\ell=1}^{L}\Big\|\nabla_{W_\ell}F_i(\overrightarrow{W}^{(t)})\Big\|_F^2\right] \le \frac{Ln^2}{b}O\Big(\frac{mF(\overrightarrow{W}^{(t)})}{d}\Big)$$

and the derivation from (5.22).

Next, taking logarithm on both sides of (5.22) and (5.23), and using Jensen's inequality $\mathbb{E}[\log X] \le \log\mathbb{E}[X]$, we have

$$\mathbb{E}[\log F(\overrightarrow{W}^{(t+1)})] \le \log F(\overrightarrow{W}^{(t)}) - \Omega\Big(\frac{\eta\delta m}{dn^2}\Big) \quad \text{and} \quad \log F(\overrightarrow{W}^{(t+1)}) \le \log F(\overrightarrow{W}^{(t)}) + O\Big(\frac{\eta Lmn^{1.5}}{\sqrt{bd}}\Big)$$

By (one-sided) martingale concentration, we have with probability at least $1 - e^{-\Omega(\log^2 m)}$, for every $t = 1, 2, \ldots, T$:

$$\log F(\overrightarrow{W}^{(t)}) - \mathbb{E}[\log F(\overrightarrow{W}^{(t)})] \le \sqrt{t} \cdot O\Big(\frac{\eta Lmn^{1.5}}{\sqrt{bd}}\Big) \cdot \log m \ .$$

This implies for every $t = 1, 2, \ldots, T$, we have

$$\begin{aligned}
\log F(\overrightarrow{W}^{(t)}) &\le \sqrt{t} \cdot O\Big(\frac{\eta L m n^{1.5}}{\sqrt{b} d}\Big) \cdot \log m + \log F(\overrightarrow{W}^{(0)}) - \Omega\Big(\frac{\eta \delta m}{d n^2}\Big) t \\
&\overset{①}{=} \log F(\overrightarrow{W}^{(0)}) - \left(\sqrt{\frac{\eta\delta m}{dn^2}} \cdot \Omega(\sqrt{t}) - \sqrt{\frac{dn^2}{\eta\delta m}} \cdot O\Big(\frac{\eta L m n^{1.5}}{\sqrt{b} d} \log m\Big)\right)^2 \\
&\quad + O\Big(\frac{\eta L^2 m n^5}{b\delta d} \log^2 m\Big) \\
&\overset{②}{\le} \log F(\overrightarrow{W}^{(0)}) + 1 - \left(\sqrt{\frac{\eta\delta m}{dn^2}} \cdot \Omega(\sqrt{t}) - \sqrt{\frac{dn^2}{\eta\delta m}} \cdot O\Big(\frac{\eta L m n^{1.5}}{\sqrt{b} d} \log m\Big)\right)^2 \\
&\overset{③}{\le} \log F(\overrightarrow{W}^{(0)}) + 1 - \mathbb{1}\left[t \ge \Theta\Big(\frac{L^2 n^7}{b\delta^2} \log^2 m\Big)\right] \cdot \Omega\left(\frac{\eta\delta m}{dn^2} t\right) \\
&\overset{④}{\le} \log F(\overrightarrow{W}^{(0)}) + 1 - \mathbb{1}\left[t \ge \Theta\Big(\frac{L^2 n^7}{b\delta^2} \log^2 m\Big)\right] \cdot \Omega\left(\frac{b\delta^2}{L^2 n^7 \log^2 m} t\right) \quad .
\end{aligned}$$

Above, in ① we have used $2a\sqrt{t} - b^2 t = -(b\sqrt{t} - a/b)^2 + a^2/b^2$; in ② we have used our choice of η; in ③ we have used $-(a\sqrt{t} - b)^2 \le -\mathbb{1}[t \ge 2b^2/a^2] \cdot \frac{a^2 t}{4}$; and in ④ we have used our choice of η again. We can read two things from the above formula:

- If $T \ge \Omega\big(\frac{L^2 n^7}{b\delta^2} \log^2 m \log \frac{n \log m}{\epsilon}\big)$ then we have

  $$\log F(\overrightarrow{W}^{(T)}) \le \log O(n \log^2 m) - \Omega\Big(\log \frac{n \log^2 m}{\epsilon}\Big) \le \log \epsilon \quad .$$

  so $F(\overrightarrow{W}^{(T)}) \le \epsilon$.

- Letting $T_0 = \Omega\big(\frac{L^2 n^7}{b\delta^2} \log^2 m\big)$, we have

  $$\sum_{i=0}^{t-1} \sqrt{F(\overrightarrow{W}^{(i)})} \le \sqrt{n \log^2 m} \cdot 2T_0 + \frac{\sqrt{n \log^2 m}}{2} \cdot 2T_0 + \frac{\sqrt{n \log^2 m}}{4} \cdot 2T_0 + \cdots \le O\big(\sqrt{n \log^2 m} T_0\big)$$

and therefore one can verify that $\|\overrightarrow{W}^{(t)} - \overrightarrow{W}^{(0)}\|_F$ is small and (5.21) holds: by Theorem 5.11.1,

$$\|W_\ell^{(t)} - W_\ell^{(0)}\|_F \le \sum_{i=0}^{t-1} \left\| \eta \frac{n}{|S_t|} \sum_{i \in S_t} \nabla_{W_\ell} F_i(\overrightarrow{W}^{(t)}) \right\|_F \le O\left(\eta \sqrt{\frac{n^2 m}{bd}}\right) \cdot \sum_{i=0}^{t-1} \sqrt{F(\overrightarrow{W}^{(i)})}$$
$$\le O\left(\eta \sqrt{\frac{n^2 m}{bd}}\right) \cdot O(T_0 \sqrt{n} \log m) \le O\left(\frac{n^{3.5}\sqrt{d}}{\delta \sqrt{bm}} \log m\right) \quad . \qquad \square$$

5.15 Theorem 5.15.1: Equivalence to Neural Tangent Kernel

Recall on input $x \in \mathbb{R}^{\mathfrak{d}}$, the network output $y(\overrightarrow{W}; x) \stackrel{\text{def}}{=} y = Bh_L \in \mathbb{R}^d$ is a function of the weights $\overrightarrow{W}$. The *neural tangent kernel (NTK)* [JGH18] is usually referred to as the feature space defined by the network gradient at random initialization. In other words, for the j-th output dimension,

- the NTK kernel function $K_j^{\mathsf{ntk}}(x, \widetilde{x}) \stackrel{\text{def}}{=} \langle \nabla y_j(\overrightarrow{W}^{(0)}; x), \nabla y_j(\overrightarrow{W}^{(0)}; \widetilde{x}) \rangle$
- the NTK objective $y_j^{\mathsf{ntk}}(\overrightarrow{W}'; x) \stackrel{\text{def}}{=} \langle \nabla y_j(\overrightarrow{W}^{(0)}; x), \overrightarrow{W}' \rangle$.

We have the following theorem whose proof is subsumed by the proofs of Theorem 5.11.1 and 5.12.1. We prove it here for completeness' sake.

Theorem 5.15.1. *Let $\overrightarrow{W}^{(0)}, A, B$ be at random initialization. For every fixed unit vector $x \in \mathbb{R}^{\mathfrak{d}}$, every (small) parameter $\omega \in \big[\Omega(\frac{d^{3/2}}{m^{3/2}L^{3/2}\log^{3/2} m}), O(\frac{1}{L^{4.5}\log^3 m})\big]$, with probability at least $1 - e^{-\Omega(m\omega^{2/3}L)}$ over $\overrightarrow{W}^{(0)}, A, B$, we have for all $\overrightarrow{W}' \in (\mathbb{R}^{m\times m})^L$ with $\|\overrightarrow{W}'\|_2 \le \omega$, for all $j \in [d]$,*

(a) $\|\nabla y_j(\overrightarrow{W}^{(0)} + \overrightarrow{W}'; x) - \nabla y_j^{\mathsf{ntk}}(\overrightarrow{W}'; x)\|_F \le O\big(\sqrt{\log m} \cdot \omega^{1/3} L^3\big) \cdot \|\nabla y_j^{\mathsf{ntk}}(\overrightarrow{W}'; x)\|_F$ *; and*

(b) $y_j(\overrightarrow{W}^{(0)} + \overrightarrow{W}'; x) = y_j(\overrightarrow{W}^{(0)}; x) + y_j^{\mathsf{ntk}}(\overrightarrow{W}'; x) + O\big(\frac{L^3\omega^{4/3}\sqrt{m\log m}}{\sqrt{d}}\big)$.

(c) If $x, \widetilde{x} \in \mathbb{R}^{\mathfrak{d}}$ are two fixed unit vectors, and $\omega \le O(\frac{1}{L^9 \log^{3/2} m})$, then

$$\big|\big\langle \nabla y_j(\overrightarrow{W}^{(0)} + \overrightarrow{W}'; x), \nabla y_j(\overrightarrow{W}^{(0)} + \overrightarrow{W}'; \widetilde{x})\big\rangle - K_j^{\mathsf{ntk}}(x, \widetilde{x})\big| \le O\big(\sqrt{\log m} \cdot \omega^{1/3} L^3\big) \cdot \sqrt{K_j^{\mathsf{ntk}}(x,x) K_j^{\mathsf{ntk}}(\widetilde{x}, \widetilde{x})} \ .$$

Proof of Theorem 5.15.1. As before we denote by $\widetilde{D}_1, \dots \widetilde{D}_L$ and $\widetilde{h}_1, \dots, \widetilde{h}_L$ the diagonal sign matrices and forward vectors determined at random initialization $\overrightarrow{W}^{(0)}$ and by $D_1, \dots, D_L$ and $h_1, \dots, h_L$ those determined at $\overrightarrow{W} \stackrel{\text{def}}{=} \overrightarrow{W}^{(0)} + \overrightarrow{W}'$. Recall $\|D'_\ell\|_0 = \|D_\ell - \widetilde{D}_\ell\|_0 \le s \stackrel{\text{def}}{=} O(m\omega^{2/3}L)$ from Lemma 5.9.1b.

(a) Let $e_j \in \mathbb{R}^d$ be the j-th basis vector and ℓ be in $[L]$. We have

$$\begin{aligned}&\nabla_{W_\ell} y_j^{\mathsf{ntk}}(\overrightarrow{W}'; x) - \nabla_{W_\ell} y_j(\overrightarrow{W}^{(0)} + \overrightarrow{W}'; x) \\ &= \left(e_j^\top B \widetilde{D}_L \widetilde{W}_L \cdots \widetilde{W}_{\ell+1} \widetilde{D}_\ell\right)^\top (\widetilde{h}_{\ell-1})^\top - \left(e_j^\top B D_L W_L \cdots W_{\ell+1} D_\ell\right)^\top (h_{\ell-1})^\top\end{aligned}$$

This difference matrix is precisely (5.11) (by setting $n = 1$ and $\mathsf{v} = e_j$). Using the bound (5.12) we have its Frobenius norm is at most $O\left(\sqrt{m \log m/d} \cdot \omega^{1/3} L^2\right)$. On the other hand, one can calculate for every $k \in [m]$,

$$\nabla_{[W_L]_k} y_j^{\mathsf{ntk}}(\overrightarrow{W}'; x) = (B^\top e_j)_k \cdot \widetilde{h}_{L-1} \cdot \mathbb{1}_{\langle [W_L]_k, \widetilde{h}_{L-1} \rangle \ge 0} \ .$$

We already know $\|\widetilde{h}_{L-1}\| \ge \Omega(1)$ from Lemma 5.8.1. Now, regardless of the randomness of $\widetilde{h}_{L-1}$, we have $\mathbb{1}_{\langle [W_L]_k, \widetilde{h}_{L-1} \rangle \ge 0} = 1$ with exactly half probability; also, regardless of the randomness of $\overrightarrow{W}^{(0)}$ and A, we have $(B^\top e_j)_k \sim \mathcal{N}(0, \frac{1}{d})$. Therefore, we conclude that with probability at least $1 - e^{-\Omega(m)}$ it satisfies $\|\nabla_{W_L} y_j^{\mathsf{ntk}}(\overrightarrow{W}'; x)\|_F^2 \ge \Omega(m/d)$. Putting the two abounds together we finish the proof.

(b) This statement can be derived from (5.15), (5.17) and (5.18). For completeness' sake,

below we provide a direct proof without invoking them. We first calculate that

$$\Big|y_j(\overrightarrow{W}^{(0)}+\overrightarrow{W}';x)-e_j^\top B\widetilde{D}_L W_L\cdots\widetilde{D}_1W_1x\Big| = \Big|\sum_{\ell=1}^{L} e_j^\top B\widetilde{D}_L W_L\cdots\widetilde{D}_{\ell+1}W_{\ell+1}D'_\ell g_{\ell-1}\Big|$$

$$\le \sum_{\ell=1}^{L}\Big(\underbrace{\Big\|B\widetilde{D}_L\widetilde{W}_L\cdots\widetilde{D}_{\ell+1}\widetilde{W}_{\ell+1}D'_\ell\Big\|_2}_{\text{Lemma 5.8.4a}} + \underbrace{\Big\|B\widetilde{D}_L\widetilde{W}_L\cdots\widetilde{D}_{\ell+1}\widetilde{W}_{\ell+1}-B\widetilde{D}_L W_L\cdots\widetilde{D}_{\ell+1}W_{\ell+1}\Big\|_2}_{\text{Lemma 5.9.6}}\Big)\cdot\underbrace{\|D'_\ell g_{\ell-1}\|}_{\text{Lemma 5.9.1b}}$$

$$\le L\cdot\Big(O\big(\frac{\sqrt{s\log m}}{\sqrt{d}}\big)+O\big(\omega L^{1.5}\frac{\sqrt{m}}{\sqrt{d}}\big)\Big)\cdot O(\omega L^{3/2}) \le O\big(\frac{L^3\omega^{4/3}\sqrt{m\log m}}{\sqrt{d}}\big) \tag{5.24}$$

We next calculate that

$$\Big|e_j^\top B\widetilde{D}_L W_L\cdots\widetilde{D}_1W_1x-y_j(\overrightarrow{W}^{(0)})-y^{\mathsf{ntk}}(\overrightarrow{W}';x)\Big|$$

$$=\Big|\sum_{\ell=1}^{L} e_j^\top B\widetilde{D}_L W_L\cdots W_{\ell+1}\widetilde{D}_\ell W'_\ell\widetilde{h}_{\ell-1}-e_j^\top B\widetilde{D}_L\widetilde{W}_L\cdots\widetilde{W}_{\ell+1}\widetilde{D}_\ell W'_\ell\widetilde{h}_{\ell-1}\Big|$$

$$\le\sum_{\ell=1}^{L}\underbrace{\Big\|B\Big(\widetilde{D}_L W_L\cdots W_{\ell+1}\widetilde{D}_\ell-\widetilde{D}_L\widetilde{W}_L\cdots\widetilde{W}_{\ell+1}\widetilde{D}_\ell\Big)\Big\|_2}_{\text{Lemma 5.9.6}}\cdot\|W'_\ell\|_2\cdot\underbrace{\|\widetilde{h}_{\ell-1}\|}_{\text{Lemma 5.8.1}}$$

$$\le L\cdot O\big(\omega L^{1.5}\frac{\sqrt{m}}{\sqrt{d}}\big)\cdot\omega\cdot O(1)\le O\big(\omega^2L^{2.5}\frac{\sqrt{m}}{\sqrt{d}}\big) \tag{5.25}$$

Putting (5.24) and (5.25) together finishes the proof.

(c) This is a direct corollary of (a). □

5.16 Extension to Other Loss Functions

For simplicity, in them main body of this paper we have used the ℓ_2 regression loss. Our results generalize easily to other Lipschitz smooth (but possibly nonconvex) loss functions.

Suppose we are given loss function $f(z;y)$ that takes as input a neural-network output $z \in \mathbb{R}^d$ and a label y. Then, our training objective for the i-th training sample becomes $F_i(W) = f(Bh_{i,L}; y_i^*)$. We redefine the loss vector $\mathsf{loss}_i \stackrel{\text{def}}{=} \nabla f(Bh_{i,L}; y_i^*) \in \mathbb{R}^d$ (where the gradient is with respect to z). Note that if $f(z;y) = \frac{1}{2}\|z-y\|^2$ is the ℓ_2 loss, then this notion coincides with Section 5.2. We assume that $f(z;y)$ is 1-Lipscthiz (upper) smooth with respect to z.[17]

All the results in Section 5.8, 5.9 and 5.10 remain unchanged. Section 5.11 also remains unchanged, except we need to restate Theorem 5.11.1 with respect to this new notation:

$$\|\nabla_{W_\ell} F_i(\overrightarrow{W})\|_F^2 \le O\Big(\frac{\|\,\mathsf{loss}_i\,\|^2}{d} \times m\Big) \qquad \|\nabla_{W_\ell} F(\overrightarrow{W})\|_F^2 \le O\Big(\frac{\|\,\mathsf{loss}\,\|^2}{d} \times mn\Big)$$
$$\|\nabla_{W_L} F(\overrightarrow{W})\|_F^2 \ge \Omega\Big(\frac{\max_{i\in[n]}\|\,\mathsf{loss}_i\,\|^2}{dn/\delta} \times m\Big)\ .$$

Section 5.12 also remains unchanged, except that we need to replace the precise definition of ℓ_2 loss in (5.14) with the semi-smoothness condition:

$$\begin{aligned} F_i(\overrightarrow{W}) = f(Bh_{i,L}; y_i^*) &\le f(B\hbar_{i,L}; y_i^*) + \langle \nabla f(B\hbar_{i,L}, y_i^*), B(h_{i,L} - \hbar_{i,L})\rangle + \frac{1}{2}\|B(h_{i,L} - \hbar_{i,L})\|^2 \\ &= F_i(\overrightarrow{\breve{W}}) + \langle \breve{\mathsf{loss}}_i, B(h_{i,L} - \hbar_{i,L})\rangle + \frac{1}{2}\|B(h_{i,L} - \hbar_{i,L})\|^2 \end{aligned} \tag{5.26}$$

[17] That is, $f(z+z';y) \le f(z) + \langle \nabla f(z;y), z'\rangle + \frac{1}{2}\|z'\|^2$.

and the rest of the proof remains unchanged.

As for the final convergence theorem of gradient descent, we can replace (5.20) with

$$F(\overrightarrow{W}^{(t+1)}) \le F(\overrightarrow{W}^{(t)}) - \Omega\Big(\frac{\eta\delta m}{dn^2}\Big) \cdot \| \mathsf{loss}^{(t)} \|^2 \ . \tag{5.27}$$

This means many things:

- If the loss is nonconvex but satisfies the Polyak-Łojasiewicz condition $\|\nabla f(z;y)\|^2 \ge \sigma(f(z;y) - f(z^*;y))$, then in $T = \Omega\Big(\frac{dn^2}{\eta\delta m\sigma}\Big) = O\big(\frac{n^6L^2}{\delta^2\sigma}\log\frac{1}{\epsilon}\big)$ iterations, GD can find a point $\overrightarrow{W}^{(T)}$ with $\| \mathsf{loss}^{(T)} \| \le \epsilon$. It suffices to choose $m \ge \widetilde{\Omega}\big(\operatorname{poly}(n, L, \delta^{-1}) \cdot d\sigma^{-2}\big)$ for same reason as before.

- If the loss is nonconvex but bounded (say, $|f(z;y)| \le O(1)$), then in $T = O\Big(\frac{dn^2}{\eta\delta m\epsilon^2}\Big) = O\big(\frac{n^6L^2}{\delta^2\epsilon^2}\big)$ iterations, we can find a point $\overrightarrow{W}^{(T)}$ with $\| \mathsf{loss}^{(T)} \| \le \epsilon$. If suffices to choose $m \ge \widetilde{\Omega}\big(\operatorname{poly}(n, L, \delta^{-1}) \cdot d\epsilon^{-1}\big)$.)

- If the loss is convex and its minimizer has bounded norm, meaning there exists z^* so that $f(z^*;y) = \min_z f(z;y)$ and $\|z - z^*\| \le D$. Then, by convexity

  $$f(z;y) - f(z^*;y) \le \langle \nabla f(z;y), z - z^* \rangle \le D\|\nabla f(z;y)\|$$

  Putting this into (5.27), we have (here $\overrightarrow{W}^* = \operatorname{argmin}_{\overrightarrow{W}} F_i(\overrightarrow{W})$ for all $i \in [n]$)

  $$\begin{aligned} F(\overrightarrow{W}^{(t+1)}) - F(\overrightarrow{W}^*) &\le F(\overrightarrow{W}^{(t)}) - F(\overrightarrow{W}^*) - \Omega\Big(\frac{\eta\delta m}{dn^2D^2}\Big) \cdot \sum_{i\in[n]} \big(F_i(\overrightarrow{W}^{(t)}) - F_i(\overrightarrow{W}^*)\big)^2 \\ &\le F(\overrightarrow{W}^{(t)}) - F(\overrightarrow{W}^*) - \Omega\Big(\frac{\eta\delta m}{dn^3D^2}\Big) \cdot \big(F(\overrightarrow{W}^{(t)}) - F(\overrightarrow{W}^*)\big)^2 \ . \end{aligned}$$

  This implies (see for instance the classical calculation steps in [Nes04]) that after $T = O\Big(\frac{dn^3D^2}{\eta\delta m\epsilon}\Big) = O\big(\frac{n^7L^2D^2}{\delta^2\epsilon}\big)$ iterations, we can have $F(\overrightarrow{W}^{(T)}) - F(\overrightarrow{W}^*) \le \epsilon$. The amount of over-parameterization needed is $m \ge \widetilde{\Omega}\big(\operatorname{poly}(n, L, \delta^{-1}) \cdot d\log\epsilon^{-1}\big)$.

- If the loss is cross entropy $f(z;y) = \frac{e^{z_y}}{\sum_{i=1}^{d} e^{z_i}}$ for classification, then $\|\nabla f(z;y)\| < 1/4$ implies perfect classification.[18] Thus, we have 100% training accuracy in $T = O\big(\frac{n^6 L^2}{\delta^2}\big)$ iterations. If suffices to choose $m \geq \widetilde{\Omega}\big(\operatorname{poly}(n, L, \delta^{-1}) \cdot d\big)$

[18] Recall $\frac{\partial f(z;y)}{z_y} = p_y(1-p_y)$ where $p_j = \frac{e^{z_j}}{\sum_{i=1}^{d} e^{z_i}}$. If $p_y > 1/2$, then z correctly predicts the target label y because $p_y > p_j$ for $j \neq z$.

5.17 Extension to Convolutional Neural Networks

There are numerous versions of convolutional neural networks (CNNs) that are used in practice. To demonstrate the capability of applying our techniques to such convolutional settings, in this section, we study a simple enough CNN for the ℓ_2 regression task.

A Simple CNN Model We assume that for the input layer (corresponding to A) and for each hidden layer $\ell = 1, 2, \ldots, L-1$ (corresponding to $W_1, \ldots, W_{L-1}$), there are $\mathfrak{d}$ positions each consisting of m channels. (Each position can be thought as a pixel of an image in computer vision tasks.) We assume the last hidden layer $\ell = L$ (corresponding to W_L) and the output layer (corresponding to B) are fully connected. We assume for each $j \in [\mathfrak{d}]$, there exists a set $Q_j \subseteq [\mathfrak{d}]$ of fixed cardinality $q \in [\mathfrak{d}]$ so that the value at position j in any convolutional layer is completely determined by positions $k \in Q_j$ of the previous layer.

Assumption 5.17.1. *We assume that $(Q_1, \ldots, Q_{\mathfrak{d}})$ give rise to a q-regular bipartite graph: each Q_j has exactly q entries and each $k \in [\mathfrak{d}]$ appears in exactly q different sets Q_j.*

(In vision tasks, if 3×3 kernels are used then $|Q_j| = 9$. We ignore the padding issue for simplicity.)

The output of each convolutional layer $\ell = 0, 1, 2, \ldots, L-1$ is represented by a $\mathfrak{d}m$-dimensional vector $h_\ell = (h_{\ell,1}, \ldots, h_{\ell,\mathfrak{d}})$ where each $h_{\ell,j} \in \mathbb{R}^m, \forall j \in [\mathfrak{d}]$. In the input layer and each $j \in [\mathfrak{d}]$, we assume

$$h_{0,j} = \phi\big(A_j x_{Q_j}\big) \in \mathbb{R}^m$$

where $x_{Q_j} \in \mathbb{R}^q$ denotes the concatenation of x_k for all $k \in Q_j$ given input $x \in \mathbb{R}^{\mathfrak{d}}$, and $A_j \in \mathbb{R}^{m \times q}$ is randomly initialized at $\mathcal{N}(0, \frac{2}{\sqrt{qm}})$ per entry. For notational simplicity, we

define matrix $A \in \mathbb{R}^{\mathfrak{d}m \times \mathfrak{d}}$ so that it satisfies $h_1 = \phi\big(Ax\big)$. Each row of A has q non-zero entries.

For each layer $\ell = 1, \ldots, L-1$ and each $j \in [\mathfrak{d}]$, we assume

$$h_{\ell,j} = \phi\big(W_{\ell,j} h_{\ell-1,Q_j} + \tau \cdot \mathbf{b}_{\ell,j}\big) \in \mathbb{R}^m$$

where $h_{\ell-1,Q_j} \in \mathbb{R}^{qm}$ denotes the concatenation of $h_{\ell-1,k}$ for all $k \in Q_j$, the weights $W_{\ell,j} \in \mathbb{R}^{m \times (qm)}$ and the bias the $\mathbf{b}_{\ell,j} \in \mathbb{R}^m$ are randomly initialized at $\mathcal{N}(0, \frac{2}{qm})$ per entry, and τ is a small parameter (say, $\tau = \frac{\delta^2}{10\mathfrak{d}L}$) for bias. For notational simplicity, we define matrix $W_\ell \in \mathbb{R}^{\mathfrak{d}m \times \mathfrak{d}m}$ and vector $\mathbf{b}_\ell \in \mathbb{R}^{\mathfrak{d}m}$ so that it satisfies $h_\ell = \phi\big(W_\ell h_{\ell-1} + \tau \mathbf{b}_\ell\big)$, and define vector $g_\ell \stackrel{\text{def}}{=} W_\ell h_{\ell-1} + \tau \mathbf{b}_\ell \in \mathbb{R}^{\mathfrak{d}m}$. Note that each row of W_ℓ has qm non-zero entries.

We assume the last layer W_L and the output layer B are simply fully connected (say without bias). That is, each entry of $W_L \in \mathbb{R}^{\mathfrak{d}m \times \mathfrak{d}m}$ is from $\mathcal{N}(0, \frac{2}{qm})$, and of $B \in \mathbb{R}^{d \times \mathfrak{d}m}$ is from $\mathcal{N}(0, \frac{1}{d})$.

We denote by $h_{i,\ell}$ the value of h_ℓ when the input vector is x_i, and define $g_{i,\ell}$, $D_{i,\ell}$ in the same way as before.

5.17.1 Changes in the Proofs

If one is willing to loose polynomial factors in L and $\mathfrak{d}$ in the final complexity, then changes to each of the lemmas of this paper is very little.[19]

Changes to Section 5.8 The first main result is Lemma 5.8.1: $\|h_{i,\ell}\|$ is in $[1-\epsilon, 1+\epsilon]$ with high probability. In the CNN case, for every $j \in [\mathfrak{d}]$, recalling that $h_{i,\ell,j} = \phi(W_{\ell,j} h_{i,\ell-1,Q_j} + \tau \mathbf{b}_\ell)$. Applying Fact 5.8.2, we have that $\frac{qm\|h_{i,\ell,j}\|^2}{2(\|h_{i,\ell-1,Q_j}\|^2+\tau^2)}$ is distributed as χ^2_ω distribution with $\omega \sim \mathcal{B}(m, \frac{1}{2})$. Due to the concentration of χ^2 distribution and the concentration of binomial distribution, $\|h_{i,\ell,j}\|^2$ is extremely close to $\frac{\|h_{i,\ell-1,Q_j}\|^2+\tau^2}{q}$ (a careful argument of this can be found in the proof of Lemma 5.8.1). Summing this up over all $j \in [\mathfrak{d}]$, and using Assumption 5.17.1, we have $\|h_{i,\ell,j}\|^2$ is concentrated at $\|h_{i,\ell-1}\|^2 + \frac{\tau^2 \mathfrak{d}}{q}$. Applying induction, we have $\|h_{i,\ell}\|$ is in $[1-\epsilon, 1+\epsilon]$ with probability at least $1 - e^{-\Omega(m\epsilon^2/L^2)}$, as long as $\tau^2 \le \frac{\epsilon q}{10\mathfrak{d}L}$.[20]

The changes to Lemma 5.8.3 and Lemma 5.8.4 are the same as above, but we loose some polynomial factors in L (because we are not careful in the argument above). For instance, the intermediate bound in Lemma 5.8.3a becomes $\|W_b D_{i,b-1} W_{b-1} \cdots D_{i,a} W_a\|_2 \le O(L)$.

As for the δ-separateness Lemma 5.8.5, we need to redefine the notion of δ-separateness between $h_{i,\ell}$ and $h_{j,\ell}$:

$$\sum_{k\in[\mathfrak{d}]} \left\| (I - \tfrac{h_{i,\ell,k} h_{i,\ell,k}^\top}{\|h_{i,\ell,k}\|^2}) h_{j,\ell,k} \right\|^2 \ge \Omega(\delta^2) \tag{5.28}$$

[19] We acknowledge the existence of more careful modifications to avoid loosing too many such factors, but do not present such result for the simplicity of this paper.

[20] We note that in all of our applications of Lemma 5.8.1, the minimal choice of ϵ is around δ^3 from the proof of δ-separateness. Therefore, choosing $\tau = \frac{\delta^2}{10\mathfrak{d}L}$ is safe. We are aware of slightly more involved proofs that are capable of handling much larger values of τ.

Then, denoting by $\widehat{h}_k = h_{i,\ell-1,k}/\|h_{i,\ell-1,k}\|$, we have

$$h_{j,\ell,k} = \phi(W_{\ell,k} h_{j,\ell-1,Q_j} + \tau \mathbf{b}_{\ell,k}) = \phi\Big(\vec{g}_1 + \Big(\sum_{z\in Q_k} \|(I - \widehat{h}_z \widehat{h}_z^\top) h_{j,\ell-1,z}\|^2\Big)^{1/2} \vec{g}_2\Big)$$

where $\vec{g}_2 \sim \mathcal{N}(0, \frac{2}{qm} I)$ is independent of the randomness of $h_{i,\ell,k}$ once $A, W_1, \ldots, W_{\ell-1}$ are fixed. One can use this to replace (5.6) and the rest of the proof follows.

Changes to Section 5.9 The first main result is Lemma 5.9.1, and we discuss necessary changes here to make it work for CNN. The first change in the proof is to replace $2c_1 L^{1.5}$ with $2c_1 L^2$ due to the above additional factor from Lemma 5.8.3a. Next, call that the proof of Lemma 5.9.1 relied on Claim 5.9.2 and Claim 5.9.4:

- For Claim 5.9.2, we can replace the definition of x with $x = D'(\widetilde{W}\widetilde{h} + \tau\mathbf{b} + g')$ for $\mathbf{b} \in \mathcal{N}(0, \frac{2}{qm} I)$. This time, instead of using the randomness of $\widetilde{W}$ like in the old proof (because $\widetilde{W}$ is no longer a full matrix), we use the randomness of $\tau\mathbf{b}$. The new statement becomes

  $$\|x\|_0 \le O\Big(\frac{\mathfrak{d}m}{\tau^{2/3}} \|g_1'\|^{2/3} + \frac{1}{\tau} \|g_2'\|_\infty (\mathfrak{d}m)^{3/2}\Big) \quad \text{and} \quad \|x\| \le O\Big(\|g_1'\| + \frac{1}{\sqrt{\tau}} \|g_2'\|_\infty^{3/2} (\mathfrak{d}m)^{3/4}\Big) \ .$$

  and its proof is by re-scaling x by $\frac{1}{\tau}$ and then applying the old proof (with dimension m replaced with $\mathfrak{d}m$).

- For Claim 5.9.4, it becomes $\|y_1\| \le O\big(\sqrt{qs/m} \log m\big)$ and $\|y_2\|_\infty \le \frac{2\sqrt{\log m}}{\sqrt{qm}}$.

After making all of these changes, we loose at most some polynomial factors in L and $\mathfrak{d}$ for the new statement of Lemma 5.9.1:

(a) $\|D'_{i,\ell}\|_0 \le m\omega^{2/3} \operatorname{poly}(L, \mathfrak{d})$ and $\|D'_{i,\ell} g_{i,\ell}\| \le \omega \operatorname{poly}(L, \mathfrak{d})$.

(b) $\|g'_{i,\ell}\|, \|h'_{i,\ell}\| \leq \omega \operatorname{poly}(L, \mathfrak{d})\sqrt{\log m}$.

Finally, the statements of Lemma 5.9.5 and Lemma 5.9.6 only loose polynomial factors in L and $\mathfrak{d}$.

Changes to Section 5.10 The norm upper bound part is trivial to modify so we only focus on the gradient norm lower bound. Since we have assumed W_L to be fully connected, the gradient on W_L is the same as before:

$$\widehat{\nabla}^{\vec{\mathsf{v}}}_{[W_L]_k} F(\overrightarrow{W}) = \sum_{i=1}^{n} \langle B_k, \mathsf{v}_i \rangle \cdot h_{i,L-1} \cdot \mathbb{1}_{(W_L h_{i,L-1})_k \geq 0}$$

Since we still have δ-separateness (5.28), one can verify for $\ell = L-1$,

$$\|h_{i,\ell} - h_{j,\ell}\|^2 = \sum_{k\in[\mathfrak{d}]} \|h_{i,\ell,k} - h_{j,\ell,k}\|^2 \geq \sum_{k\in[\mathfrak{d}]} \left\| (I - \frac{h_{i,\ell,k} h_{i,\ell,k}^\top}{\|h_{i,\ell,k}\|^2}) h_{j,\ell,k} \right\|^2 \geq \Omega(\delta^2) \ .$$

Since $\|h_{i,\ell}\| \approx 1$ and $\|h_{j,\ell}\| \approx 1$, this gives back the old definition of δ-separateness: $(I - h_{i,\ell} h_{i,\ell}^\top / \|h_{i,\ell}\|^2) h_{j,\ell}$ has norm at least $\Omega(\delta)$. Therefore, the entire rest of Section 5.10 follows as before.

Final Theorem Since Section 5.11 and 5.12 rely on previous sections, they do not need to be changed (besides some polynomial factor blowup in L and $\mathfrak{d}$). Our final theorem becomes

Theorem 5.17.2 (CNN). *Let $m \geq \widetilde{\Omega}\big(\operatorname{poly}(n, L, \mathfrak{d}, \delta^{-1}) \cdot d\big)$. For the convolutional neural network defined in this section, with probability at least $1 - e^{-\Omega(\log^2 m)}$ over the random initialization, GD and SGD respectively need at most $T = \frac{\operatorname{poly}(n,L,\mathfrak{d})}{\delta^2} \log \frac{1}{\epsilon}$ and $T = \frac{\operatorname{poly}(n,L,\mathfrak{d}) \cdot \log^2 m}{\delta^2} \log \frac{1}{\epsilon}$ iterations to find a point $F(\overrightarrow{W}) \leq \epsilon$.*

5.18 Extension to Residual Neural Networks

Again as we have discussed in Section 5.18, there are numerous versions of residual neural networks that are used in practice. To demonstrate the capability of applying our techniques to residual settings, in this section, we study a simple enough residual network for the ℓ_2 regression task (without convolutional layers).

A Simple Residual Model We consider an input layer $h_0 = \phi(Ax)$, $L-1$ residual layers $h_\ell = \phi(h_{\ell-1} + \tau W_\ell h_{\ell-1})$ for $\ell = 1, 2, \ldots, L-1$, a fully-connected layer $h_L = \phi(W_L h_{L-1})$ and an output layer $y = Bh_L$. We assume that $h_0, \ldots, h_L \in \mathbb{R}^m$ and the entries of $W_\ell \in \mathbb{R}^{m \times m}$ are from $\mathcal{N}(0, \frac{2}{m})$ as before. We choose $\tau = \frac{1}{\Omega(L \log m)}$ which is similar as previous work [ZLSD18].

We denote by $g_0 = Ax$, $g_\ell = h_{\ell-1} + \tau W_\ell h_{\ell-1}$ for $\ell = 1, 2, \ldots, L-1$ and $g_L = W_L h_{L-1}$. For analysis, we use $h_{i,\ell}$ and $g_{i,\ell}$ to denote the value of h_ℓ when the input vector is x_i, and $D_{i,\ell}$ the diagonal sign matrix so that $[D_{i,\ell}]_{k,k} = \mathbb{1}_{(g_{i,\ell})_k \geq 0}$.

5.18.1 Changes in the Proofs

Conceptually, we need to replace all the occurrences of W_ℓ with $(I + \tau W_\ell)$ for $\ell = 1, 2, \ldots, L-1$. Many of the proofs in the residual setting becomes much simpler when residual links are present. The main property we shall use is that the spectral norm

$$\|(I + \tau W_a) D_{i,a+1} \cdots D_{i,b} (I + \tau W_b)\|_2 \le 1.01 \tag{5.29}$$

for any $L - 1 \ge a \ge b \ge 1$ with our choice of τ.

Changes to Section 5.8 For Lemma 5.8.1, ignoring subscripts in i for simplicity, we can combine the old proof with (5.29) to derive that $\|h_\ell\| \le 1.02$ for every i and ℓ. We also have $\|h_\ell\| \ge \frac{1}{\sqrt{20}}$ by the following argument.

- Fact 5.8.2 says each coordinate of h_0 follows i.i.d. from a distribution which is 0 with half probability, and $|\mathcal{N}(0, \frac{2}{m})|$ with half probability. Therefore, with high probability, at least $m/4$ of the coordinates $k \in [m]$ will satisfy $|(h_0)_k| \ge \frac{0.6}{\sqrt{m}}$. Denote this set as $M_0 \subseteq [m]$.

- In the following layer $\ell = 1$, $(h_\ell)_k \ge (h_{\ell-1})_k - \tau |(W_\ell h_{\ell-1})_k|$. Since $W_\ell h_{\ell-1} \sim \mathcal{N}(0, \frac{2\|h_{\ell-1}\|^2}{m} I)$ and $\|h_{\ell-1}\| \le 1.02$, we know with high probability, at least $1 - \frac{1}{10L}$ fraction of the coordinates in M_0 will satisfy $|(W_\ell h_{\ell-1})_k| \le O(\frac{\log L}{\sqrt{m}})$. Therefore, for each of these $(1 - \frac{1}{10L})|M_0|$ coordinates, we have $(h_\ell)_k \ge (h_{\ell-1})_k - \frac{1}{10L}$ by our choice of τ. Denote this set as $M_1 \subseteq M_0$, then we have $(h_\ell)_k \ge \frac{0.6}{\sqrt{m}} - \frac{1}{10L\sqrt{m}}$ for each $k \in M_1$.

- Continuing this argument for $\ell = 2, 3, \ldots, L-1$, we know that every time we move from $M_{\ell-1}$ to M_ℓ, its size shrinks by a factor $1 - \frac{1}{10L}$, and the magnitude of $(h_\ell)_k$ for

$k \in M_\ell$ decreases by $\frac{1}{10L\sqrt{m}}$. Putting this together, we know $\|h_\ell\|^2 \geq (\frac{0.6}{\sqrt{m}} - \frac{1}{10\sqrt{m}})^2 \cdot (1 - \frac{1}{10L})^L \cdot \frac{m}{4} \geq \frac{1}{20}$ for all $\ell = 1, 2, \ldots, L-1$. The proof of the last layer h_L is the same as the old proof.

Lemma 5.8.3 is not needed anymore because of (5.29). Lemma 5.8.4 becomes trivial to prove using (5.29): for instance for Lemma 5.8.4a, we have $\|D_{i,L}W_L D_{i,L-1}(I + \tau W_{L-1}) \cdots D_{i,a}(I + \tau W_a)u\| \leq O(\|u\|)$ and thus $\|BD_{i,L}W_L D_{i,L-1}(I+\tau W_{L-1}) \cdots D_{i,a}(I+\tau W_a)u\| \leq O\big(\frac{\sqrt{s \log m}}{\sqrt{d}}\big)\|u\|$ for all s-sparse vectors u.

Lemma 5.8.5 needs the following changes in the same spirit as our changes to Lemma 5.8.1. With probability at least $1 - e^{-\Omega(\log^2 m)}$ it satisfies $\|W_\ell h_{i,\ell}\|_\infty \leq O(\frac{\log m}{\sqrt{m}})$ for all $i \in [n]$ and $\ell \in L$. In the following proof we condition on this event happens.[21] Consider $i, j \in [n]$ with $i \neq j$.

- In the input layer, since $\|x_i - x_j\| \geq \delta$, the same Claim 5.8.6 shows that, with high probability, there are at least $\frac{3}{4}m$ coordinates $k \in [m]$ with $|(h_{i,0} - h_{j,0})_k| \geq \frac{\delta}{10\sqrt{m}}$. At the same time, at least $\frac{3}{4}m$ coordinates $k \in [m]$ will satisfy $(h_{i,0})_k \geq \frac{1}{10\sqrt{m}}$ and $(h_{j,0})_k \geq \frac{1}{10\sqrt{m}}$. Denote $M_0 \subseteq [m]$ as the set of coordinates k satisfying both properties. We have $|M_0| \geq \frac{m}{2}$ and $\sum_{k \in M_0} |(h_{i,0} - h_{j,0})_k| \geq \frac{\delta}{20}\sqrt{m}$.

- In the following layer $\ell = 1$, we have

$$(h_{i,\ell} - h_{j,\ell})_k = \phi((h_{i,\ell-1})_k + \tau(W_\ell h_{i,\ell-1})_k) - \phi((h_{j,\ell-1})_k + \tau(W_\ell h_{j,\ell-1})_k)$$

[21] For simplicity, we only show how to modify Lemma 5.8.5 with success probability $1 - e^{-\Omega(\log^2 m)}$ because that is all we need to the downstream application of Lemma 5.8.5. If one is willing to be more careful, the success probability can be much higher.

Using $\|W_\ell h_{i,\ell}\|_\infty \le O(\frac{\log m}{\sqrt{m}})$ and our choice of τ, we know for every $k \in M_0$, it satisfies $(h_{i,\ell})_k \ge \frac{1}{10\sqrt{m}} - \frac{1}{100L\sqrt{m}}$ and $(h_{j,\ell})_k \ge \frac{1}{10\sqrt{m}} - \frac{1}{100L\sqrt{m}}$. Therefore, the ReLU activation becomes identity for such coordinates $k \in M_0$ and

$$\Delta_k \stackrel{\text{def}}{=} (h_{i,\ell} - h_{j,\ell})_k = (h_{i,\ell-1} - h_{j,\ell-1})_k + \tau(W_\ell(h_{i,\ell-1} - h_{j,\ell-1}))_k \ .$$

Let $s_k = 1$ if $(h_{i,\ell-1} - h_{j,\ell-1})_k \ge 0$ and $s_k = -1$ otherwise. Then,

$$\sum_{k\in M_0} |\Delta_k| \ge \sum_{k\in M_0} s_k \cdot \Delta_k = \sum_{k\in M_0} |(h_{i,\ell-1} - h_{j,\ell-1})_k| + \tau \cdot s_k(W_\ell(h_{i,\ell-1} - h_{j,\ell-1}))_k$$

Note that when $h_{i,\ell-1}$ and $h_{j,\ell-1}$ are fixed, the values $s_k(W_\ell(h_{i,\ell-1} - h_{j,\ell-1}))_k$ are independent Gaussian with mean zero. This means, with probability at least $1 - e^{-\Omega(\log^2 m)}$, the summation $\sum_{k\in M_0} s_k(W_\ell(h_{i,\ell-1} - h_{j,\ell-1}))_k$ is at most $O(\log m)$ in absolute value. Putting this into the above equation, we have

$$\sum_{k\in M_0} |\Delta_k| \ge \sum_{k\in M_0} |(h_{i,\ell-1} - h_{j,\ell-1})_k| - O(\tau \log m) \ge \frac{\delta}{20}\sqrt{m} - O(\tau \log m) \ .$$

- Continuing this process for $\ell = 2, 3, \ldots, L-1$, we can conclude that $\sum_{k\in M_0} |(h_{i,L-1} - h_{j,L-1})_k| \ge \frac{\delta}{30}\sqrt{m}$ and therefore $\|h_{i,L-1} - h_{j,L-1}\| \ge \Omega(\delta^2)$. This is the same statement as before that we shall need for the downstream application of Lemma 5.8.5.

Changes to Section 5.9 Lemma 5.9.1 becomes easy to prove with all the L factors disappear for the following reason. Fixing i and ignoring the subscript in i, we have for $\ell = 1, 2, \ldots, L-1$:

$$\begin{aligned} h'_\ell &= D''_\ell\big((I + \tau W_\ell + \tau W'_\ell)h_{\ell-1} - (I + \tau W_\ell)\widetilde{h}_{\ell-1}\big) \\ &= D''_\ell\big((I + \tau W_\ell)h'_{\ell-1} + \tau W'_\ell h_{\ell-1}\big) \end{aligned}$$

For some diagonal matrix $D''_\ell \in \mathbb{R}^{m\times m}$ with diagonal entries in $[-1,1]$ (see Proposition 5.12.3). By simple spectral norm of matrices bound we have

$$\|h'_\ell\| \le (1+\tau\|W_\ell\|_2+\tau\|W'_\ell\|_2)\|h'_{\ell-1}\|+\tau\|W'_\ell\|_2\|\widetilde{h}_{\ell-1}\| \le (1+\frac{1}{10L})\|h'_{\ell-1}\|+O(\tau\omega) \le \cdots \le O(\tau\omega)$$

This implies $\|h'_\ell\|, \|g'_\ell\| \le O(\tau\omega)$ for all $\ell \in [L-1]$, and combining with the old proof we have $\|h'_L\|, \|g'_L\| \le O(\omega)$.

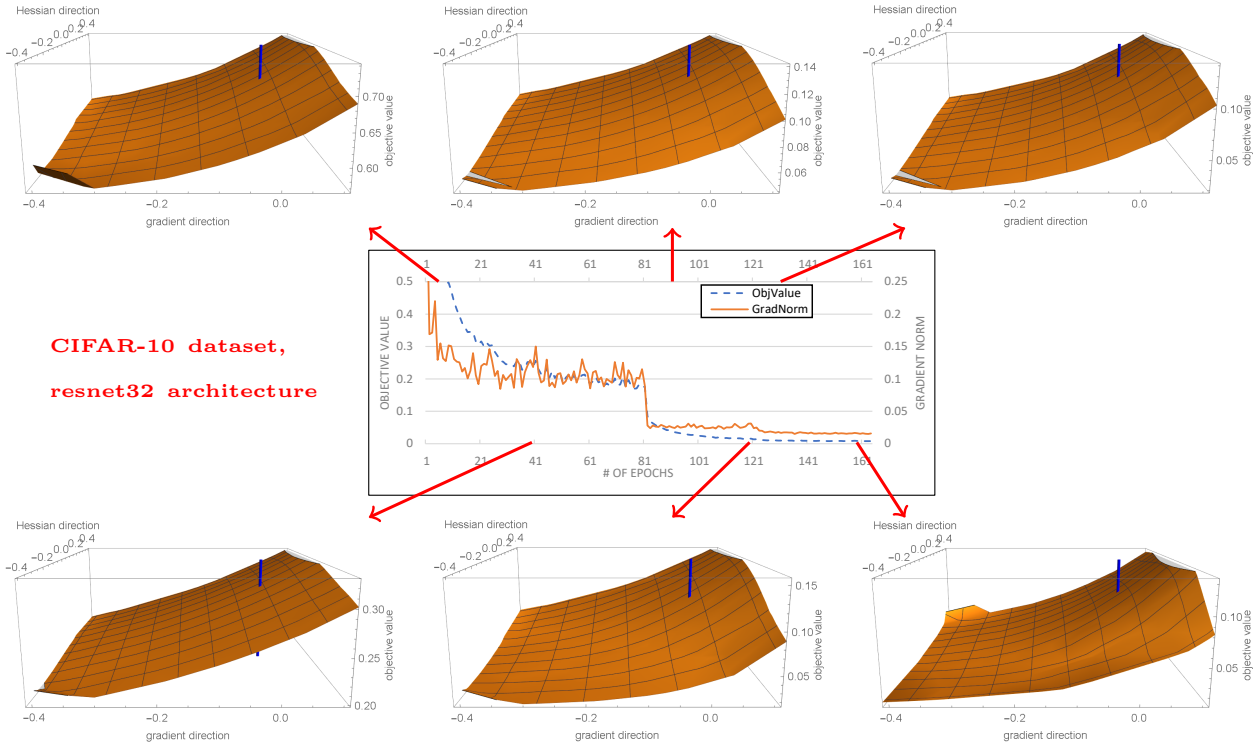
As for the sparsity $\|D'_\ell\|_0$, because $\widetilde{g}_\ell = \widetilde{h}_{\ell-1} + \tau\widetilde{W}_\ell\widetilde{h}_{\ell-1} \sim \mathcal{N}\big(\widetilde{h}_{\ell-1}, \frac{2\tau^2\|\widetilde{h}_{\ell-1}\|^2}{m}\big)$ and $\|g'_\ell\| \le O(\tau\omega)$, applying essentially the same Claim 5.9.2, we have $\|D'_\ell\|_0 \le O(m\omega^{2/3})$ for every $\ell = 1, 2, \ldots, L-1$. One can similarly argue that $\|D'_L\|_0 \le O(m\omega^{2/3})$.

Next, Lemma 5.9.5 and Lemma 5.9.6 become trivial to prove (recall we have to change $\widetilde{W}_\ell$ with $I + \tau\widetilde{W}_\ell$ for $\ell < L$) and the L factor also gets improved.

Changes to Section 5.10 The proofs of this section require only notational changes.

Final Theorem Since Section 5.11 and 5.12 rely on previous sections, they do not need to be changed (besides improving polynomial factors in L). Our final theorem becomes

Theorem 5.18.1 (ResNet). *Let $m \ge \widetilde{\Omega}\big(\operatorname{poly}(n, L, \delta^{-1}) \cdot d\big)$. For the residual neural network defined in this section, with probability at least $1 - e^{-\Omega(\log^2 m)}$ over the random initialization, GD needs at most $T = O\big(\frac{n^6L^2}{\delta^2}\log\frac{1}{\epsilon}\big)$ iterations and SGD needs at most $T = O\big(\frac{n^7L^2\log^2 m}{b\delta^2}\log\frac{1}{\epsilon}\big)$ iterations to find a point $F(\overrightarrow{W}) \le \epsilon$.*



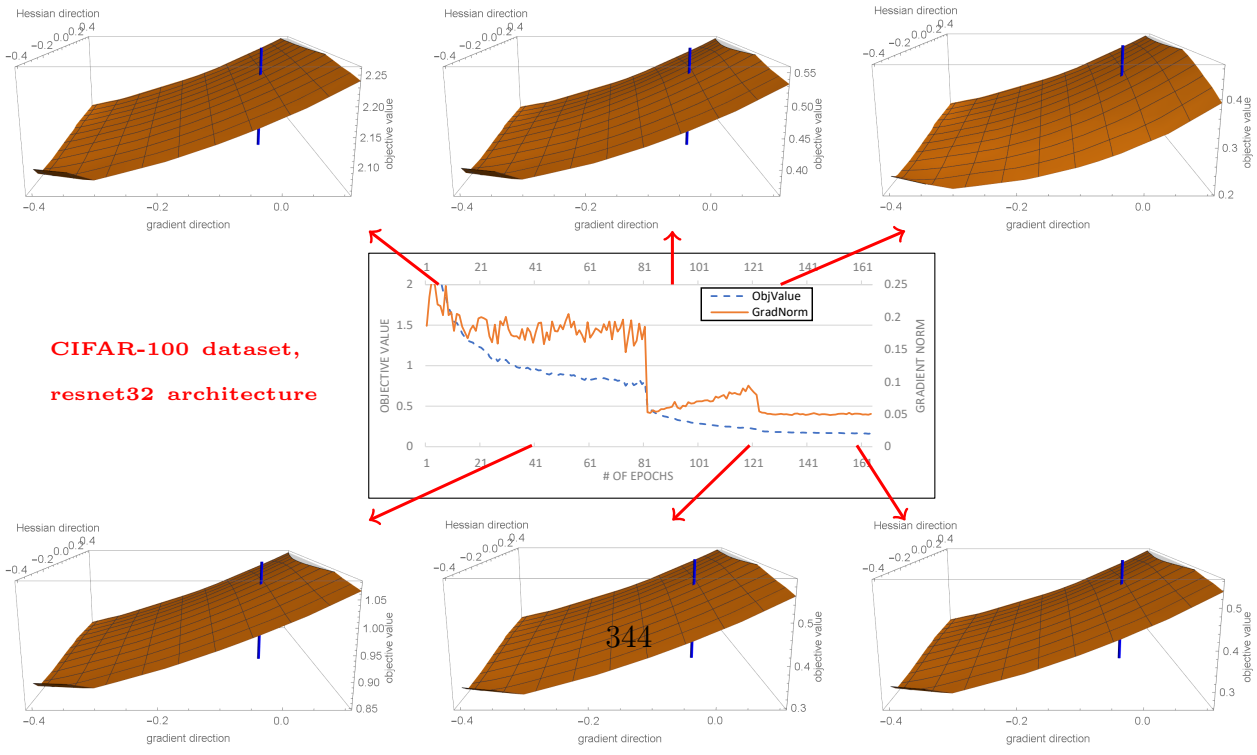


Figure 5.2: ResNet-32 architecture [Yan18] landscape on CIFAR10 vs CIFAR100.

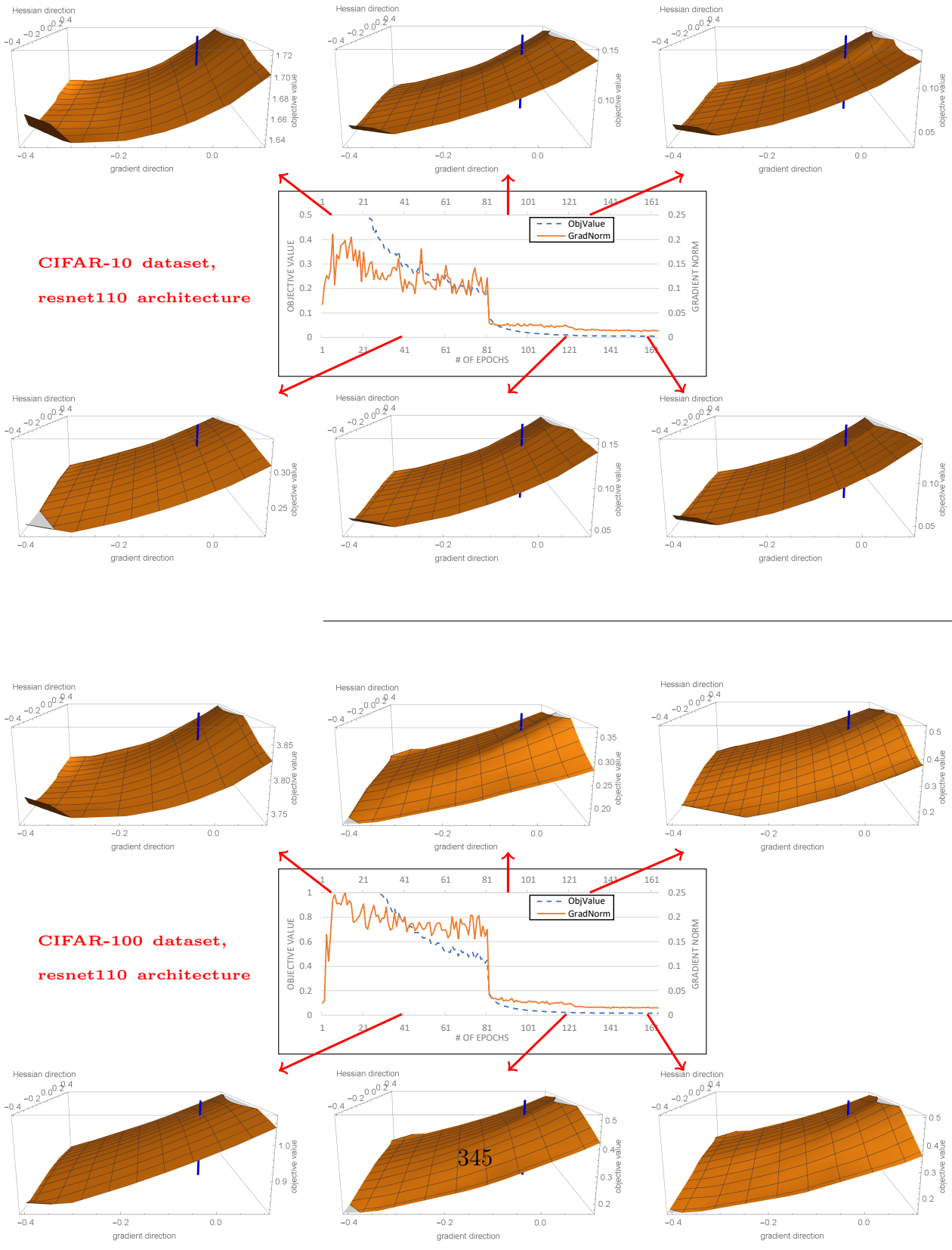


Figure 5.3: ResNet-110 architecture [Yan18] landscape on CIFAR10 vs CIFAR100.

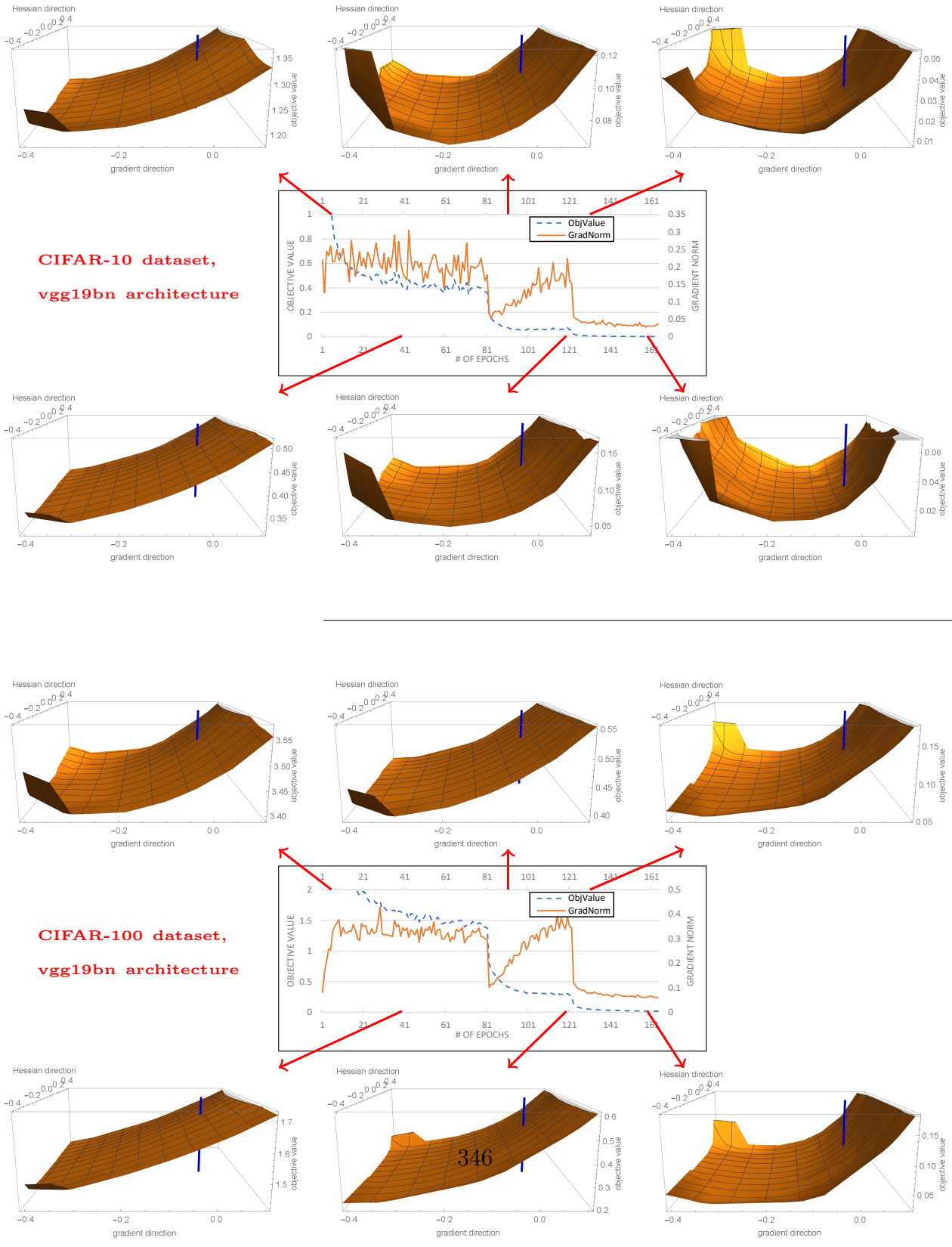
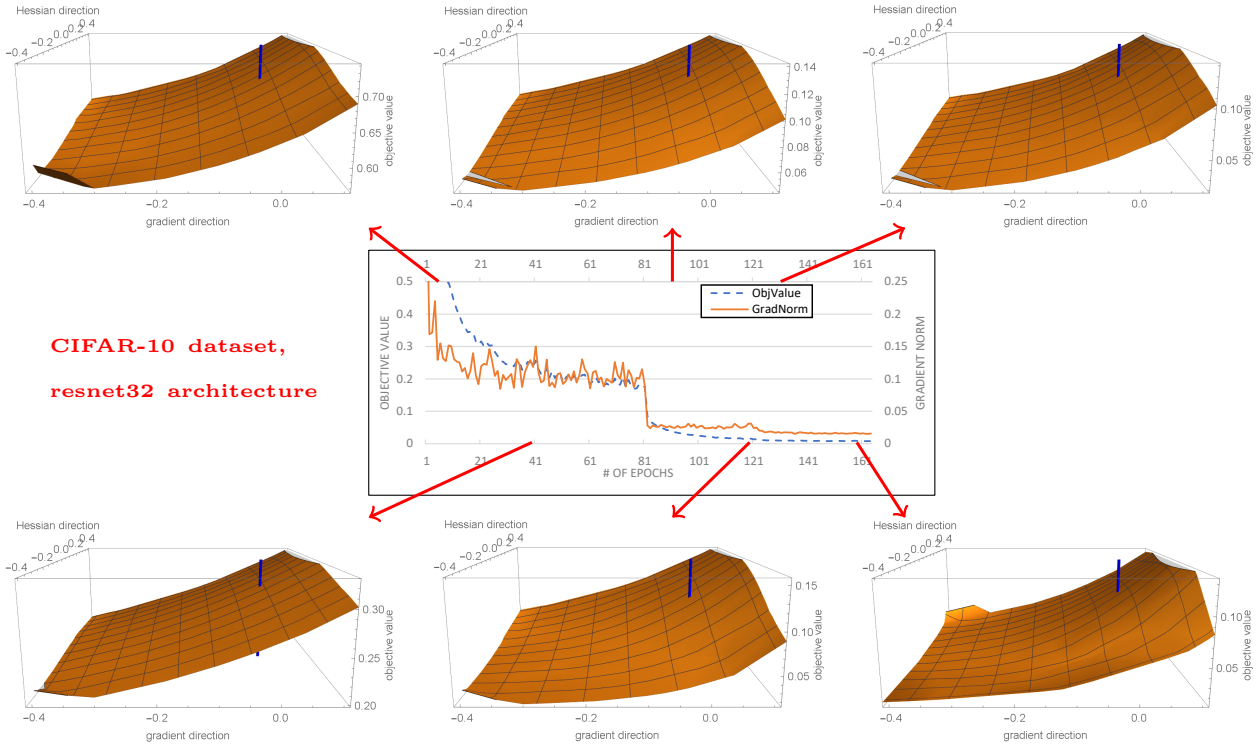


Figure 5.4: VGG19 architecture (with BN) [Yan18] landscape on CIFAR10 vs CIFAR100.



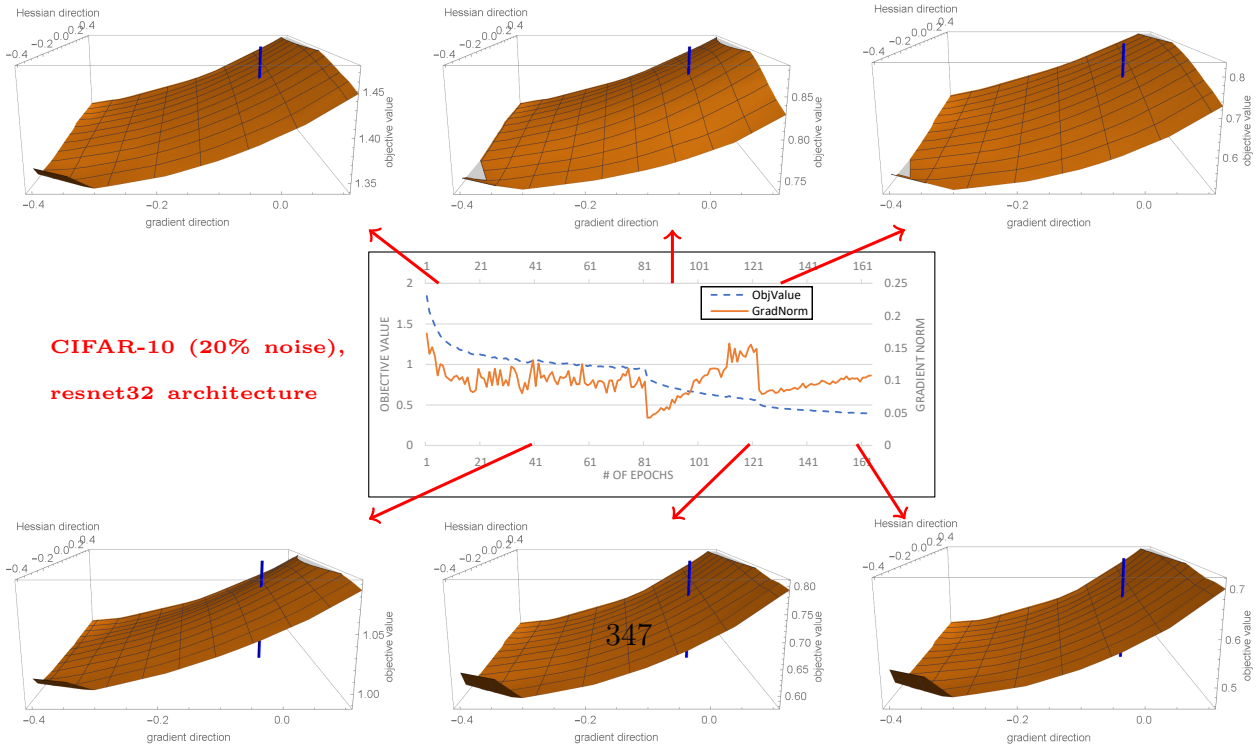
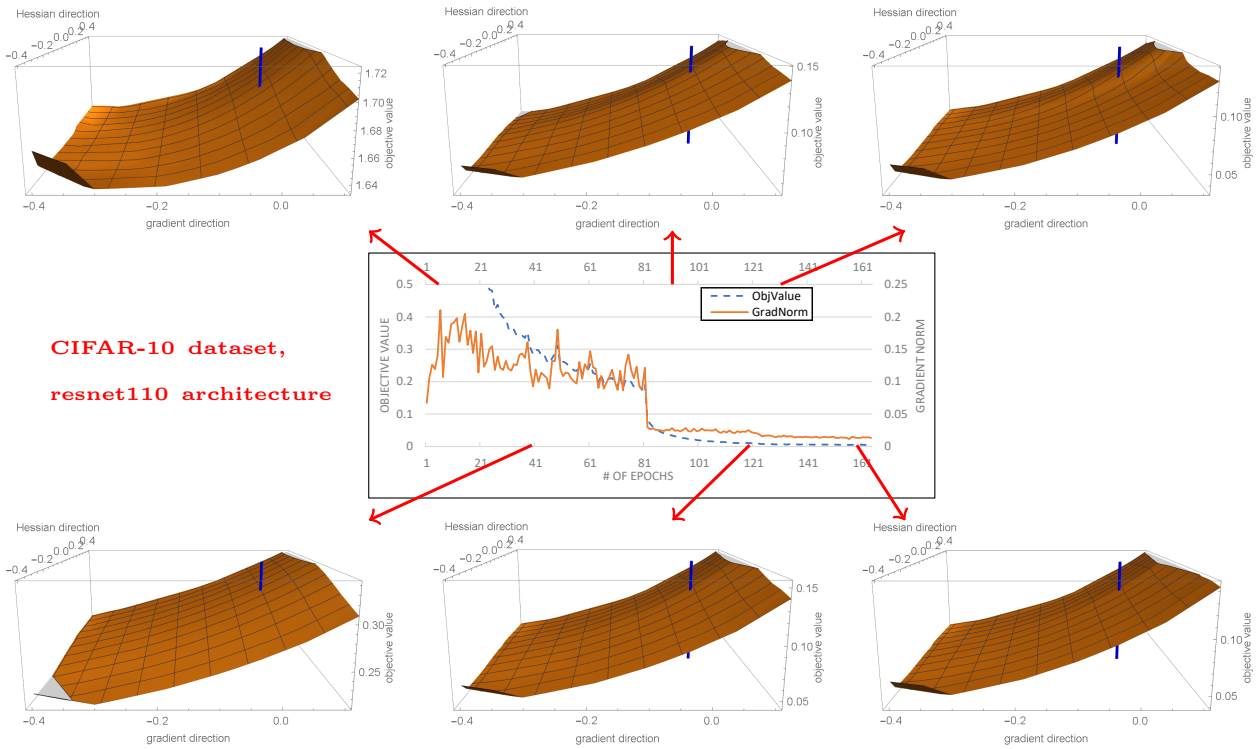


Figure 5.5: ResNet-32 architecture [Yan18] landscape on CIFAR10 vs CIFAR10 (20% noise), means we have randomly perturbed 20% of the true labels in the training set.



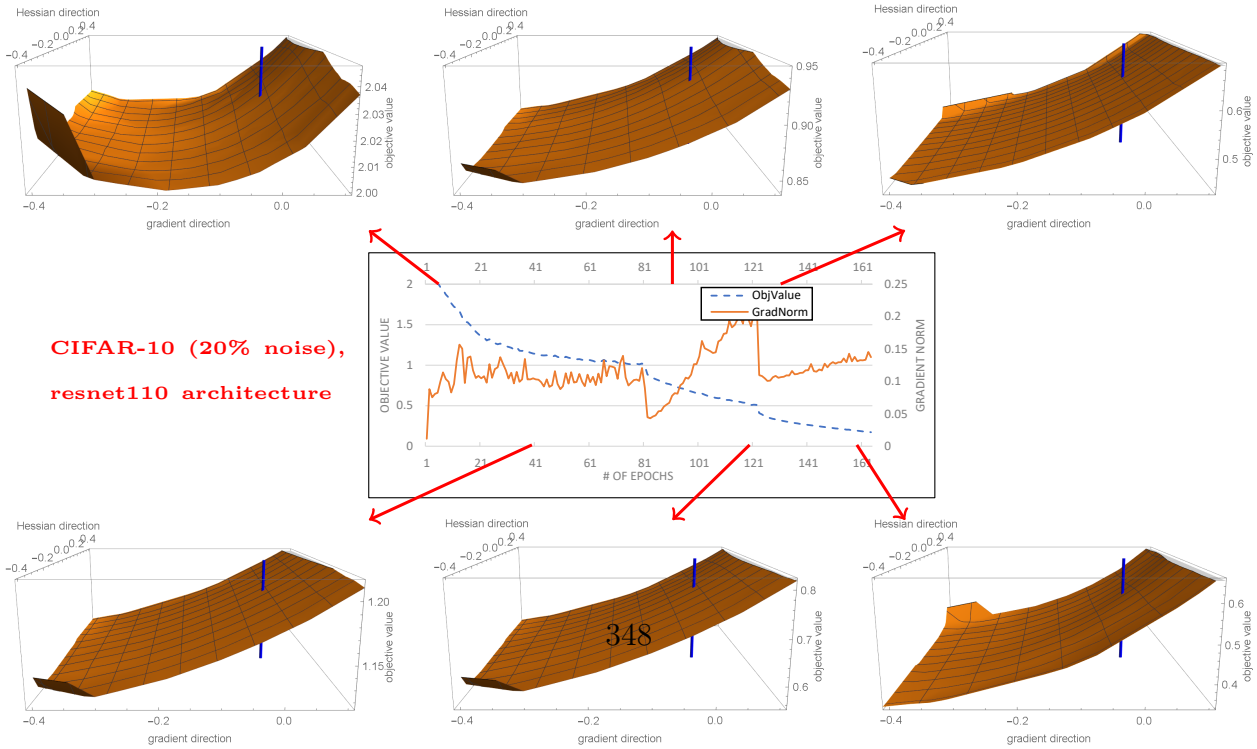


Figure 5.6: ResNet-110 architecture [Yan18] landscape on CIFAR10 vs CIFAR10 (20% noise), means we have randomly perturbed 20% of the true labels in the training set.

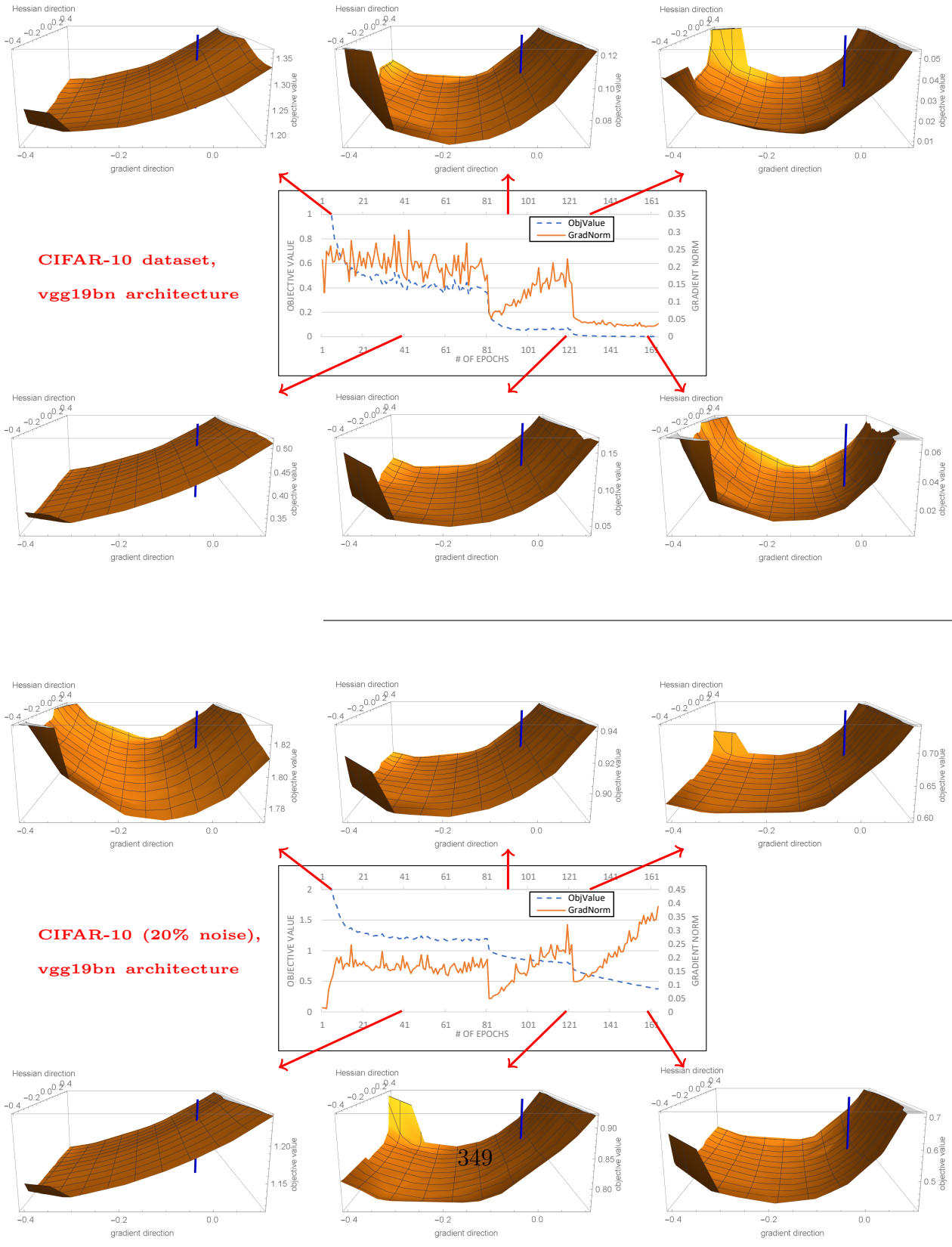


Figure 5.7: VGG19 architecture (with BN) [Yan18] landscape on CIFAR10 vs CIFAR10 (20% noise), means we have randomly perturbed 20% of the true labels in the training set.

Chapter 6

Convergence result of Recurrent Neural Network

In this chapter, we focus on recurrent neural networks (RNNs) which are multi-layer networks widely used in natural language processing. They are harder to analyze than feedforward neural networks, because the *same* recurrent unit is repeatedly applied across the entire time horizon of length L, which is analogous to feedforward networks of depth L. We show when the number of neurons is sufficiently large, meaning polynomial in the training data size and in L, then SGD is capable of minimizing the regression loss in the linear convergence rate. This gives theoretical evidence of how RNNs can memorize data.

More importantly, in this paper we build general toolkits to analyze multi-layer networks with ReLU activations. For instance, we prove why ReLU activations can prevent exponential gradient explosion or vanishing, and build a perturbation theory to analyze first-order approximation of multi-layer networks.

6.1 Introduction

Neural networks have been one of the most powerful tools in machine learning over the past a few decades [KSH12, GMH13, LHP+15, AAA+16, HZRS16, SHM+16, SSS+17]. The multi-layer structure of neural network gives it supreme power in expressibility and learning performance. However, it raises complexity concerns: the training objective is generally *non-convex* and *non-smooth*. In practice, local-search algorithms such as stochastic gradient descent (SGD) are capable of finding global optima, at least on the training data [ZBH+17, GVS15]. How SGD avoids local minima for such objectives remains an open theoretical question since [GVS15].

In recent years, there have been a number of theoretical results aiming at a better understanding of this phenomenon. Many of them focus on two-layer (thus one-hidden-layer) neural networks and assume that the inputs are random Gaussian or sufficiently close to Gaussian [BG17, Son17, Tia17, LY17, DLT+18, GLM17, PRSZ18, ZSJ+17, ZSD17]. Some study deep neural networks but assuming the activation function is linear [HM17, ACGH18, BHL18]. Some study the convex task of training essentially only the last layer of the network [Dal17]. On the technique side, some of these results try to understand the gradient dynamics [Son17, Tia17, BG17, LY17, Dal17, BGMSS18, DLT+18, PRSZ18, ZSJ+17, ZSD17, VW18], while others focus on the geometry properties of the training objective [GLM17, SS18, FB17, ZL17, HM17].

More recently, [SS18] provided evidence that, even when inputs are standard Gaussians, two-layer neural networks can indeed have spurious local minima, and suggested that *over-parameterization* (i.e., increasing the number of neurons) may be the key in avoiding spurious local minima. [LL18] showed that, for two-layer networks with the cross-entropy

loss, in the over-parametrization regime, gradient descent (GD) is capable of finding nearly-global optimal solutions on the training data. This result was later extended to the ℓ_2 loss by [DZPS19].

In this paper, we show GD and SGD are capable of training multi-layer neural networks (with ReLU activation) to global minima on any non-degenerate training data set. Furthermore, the running time is polynomial in the number of layers and the number of data points. Since there are many different types of multi-layer networks (convolutional, feedforward, recurrent, etc.), in this present paper, we focus on recurrent neural networks (RNN) as our choice of multi-layer networks, and feedforward networks are only its "special case" (see for instance a follow-up work [AZLS19]).

Recurrent Neural Networks Among different architectures of neural networks, one of the *least* theoretically-understood structure is the recurrent one [Elm90]. A recurrent neural network recurrently applies the same network unit to a sequence of input tokens, such as a sequence of words in a language sentence. RNN is particularly useful when there are long-term, non-linear interactions between input tokens in the same sequence. These networks are widely used in practice for natural language processing, language generation, machine translation, speech recognition, video and music processing, and many other tasks [MKB+10, MKB+11, SSN12, KB13, SSB14, SVL14, CVMBB14, CGCB14]. On the theory side, while there are some attempts to show that an RNN is more expressive than a feedforward neural network [KNO18], when and how an RNN can be efficiently learned has little theoretical explanation.

In practice, RNN is usually trained by simple local-search algorithms such as SGD.

However, unlike shallow networks, the training process of RNN often runs into the trouble of vanishing or exploding gradient [SMH11]. That is, the value of the gradient becomes exponentially small or large in the time horizon, even when the training objective is still constant. [1] In practice, one of the popular ways to resolve this is by the long short term memory (LSTM) structure [HS97]. However, one can also use rectified linear units (ReLUs) as activation functions to avoid vanishing or exploding gradient [SBS+17]. In fact, one of the earliest adoptions of ReLUs was on applications of RNNs for this purpose twenty years ago [Hah98, SA96].

Since the RNN structure was proposed, a large number of variations have been designed over past decades, including LSTM [HS97], Bidirectional RNN [SP97], Bidirectional LSTM [GS05], gate recurrent unit [CVMG+14], statistical recurrent unit [OPS17], Fourier recurrent unit [ZLSD18]. For a more detailed survey, we refer the readers to [SBS+17].

[1] Intuitively, an RNN recurrently applies the same network unit for L times if the input sequence is of length L. When this unit has "operator norm" larger than one or smaller than one, the final output can possibly exponentially explode or vanish in L. More importantly, when one back propagates through time —which intuitively corresponds to applying the reverse unit multiple times— the gradient can also vanish or explode. Controlling the operator norm of a non-linear operator can be quite challenging.

6.1.1 Our Question

In this paper, we study the following general question

- *Can ReLU provably stabilize the training process and avoid vanishing/exploding gradient?*
- *Can RNN be trained close to zero training error* efficiently *under mild assumptions?*

Remark 6.1.1. When there is no activation function, RNN is known as *linear dynamical system*. Hardt, Ma and Recht [HMR18] first proved the convergence of finding global minima for such linear dynamical systems. Followups in this line of research include [HSZ17, HLS+18].

Motivations One may also want to study whether RNN can be trained close to zero test error. However, unlike feedforward networks, the training error, or the ability to memorize examples, may actually be desirable for RNN. After all, many tasks involving RNN are related to memories, and certain RNN units are even referred to memory cells. Since RNN applies the same network unit to all input tokens in a sequence, the following question can possibly of its own interest:

- *How does RNN learn mappings (say from token 3 to token 7) without destroying others?*

Another motivation is the following. An RNN can be viewed as a space constraint, differentiable Turing machine, except that the input is only allowed to be read in a fixed order. It was shown in [SS91] that all Turing machines can be simulated by fully-connected recurrent networks built of neurons with non-linear activations. In practice, RNN is also used as a tool to build neural Turing machines [GWD14], equipped with a grand goal of

automatically learning an algorithm based on the observation of the inputs and outputs. To this extent, we believe the task of understanding the *trainability* as a first step towards understanding RNN can be meaningful on its own.

Our Result To present the simplest result, we focus on the classical Elman network with ReLU activation:

$$h_\ell = \phi(W \cdot h_{\ell-1} + Ax_\ell) \in \mathbb{R}^m \qquad \text{where } W \in \mathbb{R}^{m\times m},\ A \in \mathbb{R}^{m\times d_x}$$
$$y_\ell = B \cdot h_\ell \in \mathbb{R}^d \qquad \text{where } B \in \mathbb{R}^{d\times m}$$

We denote by ϕ the ReLU activation function: $\phi(x) = \max(x, 0)$. We note that (fully-connected) feedforward networks are only "special cases" to this by replacing W with W_ℓ for each layer.[2]

We consider a regression task where each sequence of inputs consists of vectors $x_1, \ldots, x_L \in \mathbb{R}^{d_x}$ and we perform least-square regression with respect to $y_1^*, \ldots, y_L^* \in \mathbb{R}^d$. We assume there are n training sequences, each of length L. We assume the training sequences are δ-separable (say vectors x_1 are different by relative distance $\delta > 0$ for every pairs of training sequences). Our main theorem can be stated as follows

Theorem 6.1.1. *If the number of neurons $m \geq \mathrm{poly}(n, d, L, \delta^{-1}, \log \epsilon^{-1})$ is polynomially large, we can find weight matrices W, A, B where the RNN gives ϵ training error*

- *if gradient descent (GD) is applied for $T = \Omega\big(\frac{\mathrm{poly}(n,d,L)}{\delta^2} \log \frac{1}{\epsilon}\big)$ iterations, starting from random Gaussian initializations; or*

[2]Most of the technical lemmas of this paper remain to hold (and become much simpler) once W is replaced with W_ℓ. This is carefully treated by [AZLS19].

- *if (mini-batch or regular) stochastic gradient descent (SGD) is applied for*

$$T = \Omega\big(\frac{\text{poly}(n, d, L)}{\delta^2} \log \frac{1}{\epsilon}\big)$$

iterations, starting from random Gaussian initializations.[3]

(To present the simplest possible result, we have not tried to tighten the polynomial dependency with respect to n, d and L. We only tightened the dependency with respect to δ and ϵ.)

Our Contribution We summarize our contributions as follows.

- We believe this is the first proof of convergence of GD/SGD for training the *hidden layers* of recurrent neural networks (or even for any multi-layer networks of more than two layers) when activation functions are present.[4]

- Our results provide arguably the first theoretical evidence towards the empirical finding of [GVS15] on multi-layer networks, regarding the ability of SGD to avoid (spurious) local minima. Our theorem does not exclude the existence of bad local minima

[3] At a first glance, one may question how it is possible for SGD to enjoy a logarithmic time dependency in ϵ^{-1}; after all, even when minimizing strongly-convex and Lipschitz-smooth functions, the typical convergence rate of SGD is $T \propto 1/\epsilon$ as opposed to $T \propto \log(1/\epsilon)$. We quickly point out there is no contradiction here if the stochastic pieces of the objective enjoy a *common* global minimizer. In math terms, suppose we want to minimize some function $f(x) = \frac{1}{n}\sum_{i=1}^n f_i(x)$, and suppose x^* is the global minimizer of convex functions $f_1(x), \ldots, f_n(x)$. Then, if $f(x)$ is σ-strongly convex, and each each $f_i(x)$ is L-Lipschitz smooth, then SGD —moving in negative direction of $\nabla f_i(x)$ for a random $i \in [n]$ per step— can find ϵ-minimizer of this function in $O\big(\frac{L^2}{\sigma^2} \log \frac{1}{\epsilon}\big)$ iterations.

[4] Our theorem holds even when A, B are at random initialization and only the hidden weight matrix W is trained. This is much more difficult to analyze than the convex task of training only the last layer B [Dal17]. Training only the last layer can significantly reduce the learning power of (recurrent or not) neural networks in practice.

- We build new technical toolkits to analyze multi-layer networks with ReLU activation, which have now found many applications [AZLS19, ALL18c, AL19a, CG19]. For instance, combining this paper with new techniques, one can derive guarantees on testing error for RNN in the PAC-learning language [AL19a].

Extension: Deep RNN Elman RNN is also referred to as three-layer RNN, and one may also study the convergence of RNNs with more hidden layers. This is referred to as deep RNN [SBS+17]. Our theorem also applies to deep RNNs (by combining this paper together with [AZLS19]).

Extension: Loss functions For simplicity, in this paper we have adopted the ℓ_2 regression loss. Our results generalize to other Lipschitz smooth (but possibly nonconvex) loss functions, by combining with the techniques of [AZLS19].

6.1.2 Other Related Works

Another relevant work is [BGMSS18] where the authors studied over-paramterization in the case of two-layer neural network under a linear-separable assumption.

Instead of using randomly initialized weights like this paper, there is a line of work proposing algorithms using weights generated from some "tensor initialization" process [ABGM14, SA15, JSA15, Tia17, ZSJ+17].

There is huge literature on using the mean-field theory to study neural networks [MMN18, YS17, DFS16, XBSD+18, LBN+17, YS18, CPS18, PW17, PB17, PSG17, PLR+16, SGGSD17]. At a high level, they study the network dynamics at random initialization when the number of hidden neurons grow to infinity, and use such initialization theory to predict performance after training. However, they do not provide theoretical convergence rate for the training process (at least when the number of neurons is finite).

6.2 Notations and Preliminaries

We denote by $\|\cdot\|_2$ (or sometimes $\|\cdot\|$) the Euclidean norm of vectors, and by $\|\cdot\|_2$ the spectral norm of matrices. We denote by $\|\cdot\|_\infty$ the infinite norm of vectors, $\|\cdot\|_0$ the sparsity of vectors or diagonal matrices, and $\|\cdot\|_F$ the Frobenius norm of matrices. Given matrix W, we denote by W_k or w_k the k-th row vector of W. We denote the row ℓ_p norm for $W \in \mathbb{R}^{m\times d}$ as

$$\|W\|_{2,p} \stackrel{\text{def}}{=} \left(\sum_{i\in[m]} \|w_i\|_2^p\right)^{1/p}. \tag{6.1}$$

By definition, $\|W\|_{2,2} = \|W\|_F$.

We use $\mathcal{N}(\mu,\sigma)$ to denote Gaussian distribution with mean μ and variance σ; or $\mathcal{N}(\mu,\Sigma)$ to denote Gaussian vector with mean μ and covariance Σ. We use $\mathbb{1}_{event}$ to denote the indicator function of whether *event* is true. We denote by e_k the k-th standard basis vector. We use $\phi(\cdot)$ to denote the ReLU function, namely $\phi(x) = \max\{x,0\} = \mathbb{1}_{x\geq 0}\cdot x$. Given univariate function $f\colon \mathbb{R}\to\mathbb{R}$, we also use f to denote the same function over vectors: $f(x) = (f(x_1),\dots,f(x_m))$ if $x\in\mathbb{R}^m$.

Given vectors $v_1,\dots,v_n\in\mathbb{R}^m$, we define $U = \mathsf{GS}(v_1,\dots,v_n)$ as their Gram-Schmidt orthonormalization. Namely, $U = [\widehat{v}_1,\dots,\widehat{v}_n]\in\mathbb{R}^{m\times n}$ where

$$\widehat{v}_1 = \frac{v_1}{\|v_1\|} \quad \text{and} \quad \text{for } i\geq 2\text{:} \quad \widehat{v}_i = \frac{\prod_{j=1}^{i-1}(I-\widehat{v}_j\widehat{v}_j^\top)v_i}{\left\|\prod_{j=1}^{i-1}(I-\widehat{v}_j\widehat{v}_j^\top)v_i\right\|}.$$

Note that in the occasion that $\prod_{j=1}^{i-1}(I-\widehat{v}_j\widehat{v}_j^\top)v_i$ is the zero vector, we let $\widehat{v}_i$ be an arbitrary unit vector that is orthogonal to $\widehat{v}_1,\dots,\widehat{v}_{i-1}$.

6.2.1 Elman Recurrent Neural Network

We assume n training inputs are given: $(x_{i,1}, x_{i,2}, \ldots, x_{i,L}) \in \left(\mathbb{R}^{d_x}\right)^L$ for each input $i \in [n]$. We assume n training labels are given: $(y^*_{i,1}, y^*_{i,2}, \ldots, y^*_{i,L}) \in \left(\mathbb{R}^{d}\right)^L$ for each input $i \in [n]$. Without loss of generality, we assume $\|x_{i,\ell}\| \leq 1$ for every $i \in [n]$ and $\ell \in [L]$. Also without loss of generality, we assume $\|x_{i,1}\| = 1$ and its last coordinate $[x_{i,1}]_{d_x} = \frac{1}{\sqrt{2}}$ for every $i \in [n]$.[5]

We make the following assumption on the input data :

Assumption 6.2.1. *$\|x_{i,1} - x_{j,1}\| \geq \delta$ for some parameter $\delta \in (0, 1]$ and every pair of $i \neq j \in [n]$.*

Given weight matrices $W \in \mathbb{R}^{m \times m}$, $A \in \mathbb{R}^{m \times d_x}$, $B \in \mathbb{R}^{d \times m}$, we introduce the following notations to describe the evaluation of RNN on the input sequences. For each $i \in [n]$ and $j \in [L]$:

$$h_{i,0} = 0 \in \mathbb{R}^m \qquad g_{i,\ell} = W \cdot h_{i,\ell-1} + Ax_{i,\ell} \in \mathbb{R}^m$$
$$y_{i,\ell} = B \cdot h_{i,\ell} \in \mathbb{R}^d \qquad h_{i,\ell} = \phi(W \cdot h_{i,\ell-1} + Ax_{i,\ell}) \in \mathbb{R}^m$$

A very important notion that this entire paper relies on is the following:

Definition 6.2.1. For each $i \in [n]$ and $\ell \in [L]$, let $D_{i,\ell} \in \mathbb{R}^{m \times m}$ be the diagonal matrix where

$$(D_{i,\ell})_{k,k} = \mathbf{1}_{(W \cdot h_{i,\ell-1} + Ax_{i,\ell})_k \geq 0} = \mathbf{1}_{(g_{i,\ell})_k \geq 0} \ .$$

As a result, we can write $h_{i,\ell} = D_{i,\ell} W h_{i,\ell-1}$.

[5] If it only satisfies $\|x_{i,1}\| \leq 1$ one can pad it with an additional coordinate to make $\|x_{i,1}\| = 1$ hold. As for the assumption $[x_{i,1}]_{d_x} = \frac{1}{\sqrt{2}}$, this is equivalent to adding a bias term $\mathcal{N}(0, \frac{1}{m})$ for the first layer.

We consider the following random initialization distributions for W, A and B.

Definition 6.2.2. We say that W, A, B are at random initialization, if the entries of W and A are i.i.d. generated from $\mathcal{N}(0, \frac{2}{m})$, and the entries of $B_{i,j}$ are i.i.d. generated from $\mathcal{N}(0, \frac{1}{d})$.

Throughout this paper, for notational simplicity, we refer to index ℓ as the ℓ-th *layer* of RNN, and $h_{i,\ell}$, $x_{i,\ell}$, $y_{i,\ell}$ respectively as the hidden neurons, input, output on the ℓ-th layer. We acknowledge that in certain literatures, one may regard Elman network as a three-layer RNN.

Assumption 6.2.2. *We assume $m \geq \mathrm{poly}(n, d, L, \frac{1}{\delta}, \log \frac{1}{\epsilon})$ for some sufficiently large polynomial.*

Without loss of generality, we assume $\delta \leq \frac{1}{CL^2 \log^3 m}$ for some sufficiently large constant C (if this is not satisfied one can decrease δ). Throughout the paper except the detailed appendix, we use $\widetilde{O}$, $\widetilde{\Omega}$ and $\widetilde{\Theta}$ notions to hide polylogarithmic dependency in m. To simplify notations, we denote by

$$\rho \stackrel{\text{def}}{=} nLd \log m \quad \text{and} \quad \varrho \stackrel{\text{def}}{=} nLd\delta^{-1} \log(m/\epsilon) \ .$$

6.2.2 Objective and Gradient

For simplicity, we only optimize over the weight matrix $W \in \mathbb{R}^{m \times m}$ and let A and B be at random initialization. As a result, our ℓ_2-regression objective is a function over W:[6]

$$f(W) \stackrel{\text{def}}{=} \sum_{i=1}^{n} f_i(W) \quad \text{and} \quad f_i(W) \stackrel{\text{def}}{=} \frac{1}{2} \sum_{\ell=2}^{L} \| \mathsf{loss}_{i,\ell} \|_2^2 \quad \text{where} \quad \mathsf{loss}_{i,\ell} \stackrel{\text{def}}{=} B h_{i,\ell} - y^*_{i,\ell} \ .$$

Using chain rule, one can write down a closed form of the (sub-)gradient:

Fact 6.2.3. *For $k \in [m]$, the gradient with respect to W_k (denoted by ∇_k) and the full gradient are*

$$\nabla_k f(W) = \sum_{i=1}^{n} \sum_{a=2}^{L} \sum_{\ell=1}^{a-1} (\mathsf{Back}_{i,\ell+1\to a}^\top \cdot \mathsf{loss}_{i,a})_k \cdot h_{i,\ell} \cdot \mathbf{1}_{\langle W_k, h_{i,\ell}\rangle + \langle A_k, x_{i,\ell+1}\rangle \geq 0}$$

$$\nabla f(W) = \sum_{i=1}^{n} \sum_{a=2}^{L} \sum_{\ell=1}^{a-1} D_{i,\ell+1} \big(\mathsf{Back}_{i,\ell+1\to a}^\top \cdot \mathsf{loss}_{i,a} \big) \cdot h_{i,\ell}^\top$$

where for every $i \in [n]$, $\ell \in [L]$, and $a = \ell+1, \ell+2, \ldots, L$:

$$\mathsf{Back}_{i,\ell\to\ell} \stackrel{\text{def}}{=} B \in \mathbb{R}^{d \times m} \quad \textit{and} \quad \mathsf{Back}_{i,\ell\to a} \stackrel{\text{def}}{=} B D_{i,a} W \cdots D_{i,\ell+1} W \in \mathbb{R}^{d \times m} \ .$$

[6] The index ℓ starts from 2, because $B h_{i,1} = B\phi(A x_{i,1})$ remains constant if we are not optimizing over A and B.

6.3 Our Results

Our main results can be formally stated as follows.

Theorem 6.3.1 (GD). *Suppose* $\eta = \widetilde{\Theta}\big(\frac{\delta}{m}\operatorname{poly}(n,d,L)\big)$ *and* $m \geq \operatorname{poly}(n,d,L,\delta^{-1},\log\epsilon^{-1})$. *Let* $W^{(0)}, A, B$ *be at random initialization. With high probability over the randomness of* $W^{(0)}, A, B$, *if we apply gradient descent for* T *steps* $W^{(t+1)} = W^{(t)} - \eta\nabla f(W^{(t)})$, *then it satisfies*

$$f(W^{(T)}) \leq \epsilon \quad \text{for} \quad T = \widetilde{\Omega}\big(\frac{\operatorname{poly}(n,d,L)}{\delta^2}\log\frac{1}{\epsilon}\big).$$

Theorem 6.3.2 (SGD). *Suppose* $\eta = \widetilde{\Theta}\big(\frac{\delta}{m}\operatorname{poly}(n,d,L)\big)$ *and* $m \geq \operatorname{poly}(n,d,L,\delta^{-1},\log\epsilon^{-1})$. *Let* $W^{(0)}, A, B$ *be at random initialization. If we apply stochastic gradient descent for* T *steps* $W^{(t+1)} = W^{(t)} - \eta\nabla f_i(W^{(t)})$ *for a random index* $i \in [n]$ *per step, then with high probability (over* $W^{(0)}, A, B$ *and the randomness of SGD), it satisfies*

$$f(W^{(T)}) \leq \epsilon \quad \text{for} \quad T = \widetilde{\Omega}\big(\frac{\operatorname{poly}(n,d,L)}{\delta^2}\log\frac{1}{\epsilon}\big).$$

In both cases, we essentially have *linear convergence rates.* [7] Notably, our results show that the dependency of the number of layers L, is *polynomial.* Thus, even when RNN is applied to sequences of long input data, it does not suffer from exponential gradient explosion or vanishing (e.g., $2^{\Omega(L)}$ or $2^{-\Omega(L)}$) through the entire training process.

Main Technical Theorems Our main Theorem 6.17.1 and Theorem 6.18.1 are in fact natural consequences of the following two technical theorems. They both talk about the

[7] We remark here that the $\widetilde{O}$ notation may hide additional polynomial dependency in $\log\log\epsilon^{-1}$. This is not necessary, at the expense of slightly complicating the proofs, as shown by follow up [AZLS19].

first-order behavior of RNNs when the weight matrix W is sufficiently close to some random initialization.

The first theorem is similar to the classical Polyak-Łojasiewicz condition [Pol63, Loj63], and says that $\|\nabla f(W)\|_F^2$ is at least as large as the objective value.

Theorem 6.3.3. *With high probability over random initialization $\widetilde{W}, A, B$, it satisfies*

$$\forall W \in \mathbb{R}^{m\times m} \text{ with } \|W - \widetilde{W}\|_2 \le \frac{\mathrm{poly}(\varrho)}{\sqrt{m}}:$$

$$\begin{aligned} \|\nabla f(W)\|_F^2 &\ge \frac{\delta}{\mathrm{poly}(\rho)} \times m \times f(W) \ , \\ \|\nabla f(W)\|_F^2, \|\nabla f_i(W)\|_F^2 &\le \mathrm{poly}(\rho) \times m \times f(W) \ . \end{aligned}$$

The second theorem shows a special "semi-smoothness" property of the objective.

Theorem 6.3.4. *With high probability over random initialization $\widetilde{W}, A, B$, it satisfies for every $\breve{W} \in \mathbb{R}^{m\times m}$ with $\|\breve{W} - \widetilde{W}\| \le \frac{\mathrm{poly}(\varrho)}{\sqrt{m}}$, and for every $W' \in \mathbb{R}^{m\times m}$ with $\|W'\| \le \frac{\tau_0}{\sqrt{m}}$,*

$$f(\breve{W} + W') \le f(\breve{W}) + \langle \nabla f(\breve{W}), W' \rangle + \mathrm{poly}(\varrho) m^{1/3} \cdot \sqrt{f(W)} \cdot \|W'\|_2 + \mathrm{poly}(\rho) m \|W'\|_2^2 \ .$$

At a high level, the convergence of GD and SGD are careful applications of the two technical theorems above: indeed, Theorem 6.15.2 shows that as long as the objective value is high, the gradient is large; and Theorem 6.16.1 shows that if one moves in the (negative) gradient direction, then the objective value can be sufficiently decreased. These two technical theorems together ensure that GD/SGD does not hit any saddle point or (bad) local minima along its training trajectory. This was practically observed by [GVS15] and a theoretical justification was open since then.

An Open Question We did not try to tighten the polynomial dependencies of (n, d, L) in the proofs. When m is sufficiently large, we make use of the randomness at initialization to argue that, *for all the points* within a certain radius from initialization, for instance Theorem 6.15.2 holds. In practice, however, the SGD can create additional randomness as time goes; also, in practice, it suffices for those points on the SGD trajectory to satisfy Theorem 6.15.2. Unfortunately, such randomness can — in principle — be correlated with the SGD trajectory, so we do not know how to use that in the proofs. Analyzing such correlated randomness is certainly beyond the scope of this paper, but can possibly explain why in practice, the size of m needed is not that large.

6.3.1 Conclusion

Overall, we provide the first proof of convergence of GD/SGD for non-linear neural networks that have more two layers. We show with overparameterization GD/SGD can avoid hitting any (bad) local minima along its training trajectory. This was practically observed by [GVS15] and a theoretical justification was open since then. We present our result using recurrent neural networks (as opposed to the simpler feedforward networks [AZLS19]) in this very first paper, because memorization in RNN could be of independent interest. Also, our result proves that RNN can learn mappings from different input tokens to different output tokens *simultaneously* using the same recurrent unit.

Last but not least, we build new tools to analyze multi-layer networks with ReLU activations that could facilitate many new research on deep learning. For instance, our techniques in Section 6.5 provide a general theory for why ReLU activations avoid exponential exploding (see e.g. (6.2), (6.5)) or exponential vanishing (see e.g. (6.2), (6.4)); and our techniques in Section 6.6 give a general theory for the stability of multi-layer networks against adversarial weight perturbations, which is at the heart of showing the semi-smoothness Theorem 6.16.1, and used by all the follow-up works [AZLS19, ALL18c, AL19a, CG19].

6.4 Proof Sketch

The main difficulty of this paper is to prove Theorem 6.15.2 and 6.16.1, and we shall sketch the proof ideas in Section 6.5 through 6.8. In such high-level discussions, we shall put our emphasize on

- how to avoid exponential blow up in L, and
- how to deal with the issue of randomness dependence across layers.

We genuinely hope that this high-level sketch can (1) give readers a clear overview of the proof without the necessity of going to the appendix, and (2) appreciate our proof and understand why it is necessarily long.[8]

[8]For instance, proving gradient norm lower bound in Theorem 6.15.2 for a single neuron $k \in [m]$ is easy, but how to apply concentration across neurons? Crucially, due to the recurrent structure these quantities are never independent, so we have to build necessary probabilistic tools to tackle this. If one is willing to ignore such subtleties, then our sketched proof is sufficiently short and gives a good overview.

6.5 Basic Properties at Random Initialization

In this section we derive basic properties of the RNN when the weight matrices W, A, B are all *at random initialization*. The corresponding precise statements and proofs are in Section 6.11.

The first one says that the forward propagation neither explodes or vanishes, that is,

$$\frac{1}{2} \leq \|h_{i,\ell}\|_2, \|g_{i,\ell}\|_2 \leq O(L) \ . \tag{6.2}$$

Intuitively, (6.2) very reasonable. Since the weight matrix W is randomly initialized with entries i.i.d. from $\mathcal{N}\left(0, \frac{2}{m}\right)$, the norm $\|Wz\|_2$ is around $\sqrt{2}$ for any fixed vector z. Equipped with ReLU activation, it "shuts down" roughly half of the coordinates of Wz and reduces the norm $\|\phi(Wz)\|$ to one. Since in each layer ℓ, there is an additional unit-norm signal $x_{i,\ell}$ coming in, we should expect the final norm of hidden neurons to be at most $O(L)$.

Unfortunately, the above argument cannot be directly applied since the weight matrix W is reused for L times so there is no fresh new randomness across layers. Let us explain how we deal with this issue carefully, because it is at the heart of *all* of our proofs in this paper. Recall, each time W is applied to some vector $h_{i,\ell}$, it only uses "one column of randomness" of W. Mathematically, letting $U_\ell \in \mathbb{R}^{m \times n\ell}$ denote the column orthonormal matrix using Gram-Schmidt

$$U_\ell \stackrel{\text{def}}{=} \mathsf{GS}\left(h_{1,1}, \ldots, h_{n,1}, h_{1,2}, \ldots, h_{n,2}, \ldots, h_{1,\ell}, \ldots, h_{n,\ell}\right) \ ,$$

we have $Wh_{i,\ell} = WU_{\ell-1}U_{\ell-1}^\top h_{i,\ell} + W(I - U_{\ell-1}U_{\ell-1}^\top)h_{i,\ell}$.

- The second term $W(I-U_{\ell-1}U_{\ell-1}^\top)h_{i,\ell}$ has new randomness independent of the previous layers.[9]

- The first term $WU_{\ell-1}U_{\ell-1}^\top h_{i,\ell}$ relies on the randomness of W in the directions of $h_{i,a}$ for $a<\ell$ of the previous layers. We cannot rely on the randomness of this term, because when applying inductive argument till layer ℓ, the randomness of $WU_{\ell-1}$ is already used.

  Fortunately, $WU_{\ell-1}\in\mathbb{R}^{m\times n(\ell-1)}$ is a rectangular matrix with $m\gg n(\ell-1)$ (thanks to overparameterization!) so one can bound its spectral norm by roughly $\sqrt{2}$. This ensures that no matter how $h_{i,\ell}$ behaves (even arbitrarily correlated with $WU_{\ell-1}$), the norm of the first term cannot be too large. It is crucial here that $WU_{\ell-1}$ is a *rectangular* matrix, because for a square random matrix such as W, its spectral norm is 2 and using that, the forward propagation bound will exponentially blow up.

This summarizes the main idea for proving $\|h_{i,\ell}\|\le O(L)$ in (6.2); the lower bound $\frac{1}{2}$ is similar. Our next property says in each layer, the amount of "fresh new randomness" is non-negligible:

$$\|(I-U_{\ell-1}U_{\ell-1}^\top)h_{i,\ell}\|_2\ge\widetilde{\Omega}(\frac{1}{L^2})\ . \tag{6.3}$$

This relies on a more involved inductive argument than (6.2). At high level, one needs to show that in each layer, the amount of "fresh new randomness" reduces only by a factor at most $1-\frac{1}{10L}$.

[9] More precisely, letting $v=(I-U_{\ell-1}U_{\ell-1}^\top)h_{i,\ell}$, we have $W(I-U_{\ell-1}U_{\ell-1}^\top)h_{i,\ell}=\big(W\frac{v}{\|v\|}\big)\|v\|$. Here, $W\frac{v}{\|v\|}$ is a random Gaussian vector in $\mathcal{N}(0,\frac{2}{m}I)$ and is *independent* of all $\{h_{i,a}\mid i\in[n],a<\ell\}$.

Using (6.2) and (6.3), we obtain the following property about the data separability:

$$(I - U_\ell U_\ell^\top)h_{i,\ell+1} \text{ and } (I - U_\ell U_\ell^\top)h_{j,\ell+1} \text{ are } (\delta/2)\text{-separable, } \forall i, j \in [n] \text{ with } i \neq j \quad (6.4)$$

Here, we say two vectors x and y are δ-separable if $\left\|(I - yy^\top/\|y\|_2^2)x\right\| \geq \delta$ and vice versa. Property (6.4) shows that the separability information (say on input token 1) *does not diminish* by more than a polynomial factor even if the information is propagated for L layers.

We prove (6.4) by induction. In the first layer $\ell = 1$ we have $h_{i,1}$ and $h_{j,1}$ are δ-separable which is a consequence of Assumption 6.2.1. If having fresh new randomness, given two δ separable vectors x, y, one can show that $\phi(Wx)$ and $\phi(Wy)$ are also $\delta(1-o(\frac{1}{L}))$-separable. Again, in RNN, we do not have fresh new randomness, so we rely on (6.3) to give us reasonably large fresh new randomness. Applying a careful induction helps us to derive that (6.4) holds for all layers.[10]

Intermediate Layers and Backward Propagation Training neural network is not only about forward propagation. We also have to bound intermediate layers and backward propagation.

The first two results we derive are the following. For every $\ell_1 \geq \ell_2$ and diagonal

[10]This is the only place that we rely on Assumption 6.2.1. This assumption is somewhat necessary in the following sense. If $x_{i,\ell} = x_{j,\ell}$ for some pair $i \neq j$ for all the first ten layers $\ell = 1, 2, \ldots, 10$, and if $y^*_{i,\ell} \neq y^*_{i,\ell}$ for even just one of these layers, then there is no hope in having the training objective decrease to zero. Of course, one can make more relaxed assumption on the input data, involving both $x_{i,\ell}$ and $y^*_{i,\ell}$. While this is possible, it complicates the statements so we do not present such results in this paper.

matrices D' of sparsity $s \in [\rho^2, m^{0.49}]$:

$$\|WD_{i,\ell_1} \cdots D_{i,\ell_2} W\|_2 \le O(L^3) \tag{6.5}$$

$$\|D'WD_{i,\ell_1} \cdots D_{i,\ell_2} WD'\|_2 \le \widetilde{O}(\sqrt{s}/\sqrt{m}) \tag{6.6}$$

Intuitively, one cannot use spectral bound argument to derive (6.5) or (6.6): the spectral norm of W is 2, and even if ReLU activations cancel half of its mass, the spectral norm $\|DW\|_2$ remains to be $\sqrt{2}$. When stacked together, this grows exponential in L.

Instead, we use an analogous argument to (6.2) to show that, for each fixed vector z, the norm of $\|WD_{i,\ell_1} \cdots D_{i,\ell_2} Wz\|_2$ is at most $O(1)$ with extremely high probability $1 - e^{-\Omega(m/L^2)}$. By standard ϵ-net argument, $\|WD_{i,\ell_1} \cdots D_{i,\ell_2} Wz\|_2$ is at most $O(1)$ for all $\frac{m}{L^3}$-sparse vectors z. Finally, for a possible dense vector z, we can divide it into L^3 chunks each of sparsity $\frac{m}{L^3}$. Finally, we apply the upper bound for L^3 times. This proves (6.5). One can use similar argument to prove (6.6).

Remark 6.5.1. We did not try to tighten the polynomial factor here in L. We conjecture that proving an $O(1)$ bound may be possible, but that question itself may be a sufficiently interesting random matrix theory problem on its own.

The next result is for back propagation. For every $\ell_1 \ge \ell_2$ and diagonal matrices D' of sparsity $s \in [\rho^2, m^{0.49}]$:

$$\|BD_{i,\ell_1} \cdots D_{i,\ell_2} WD'\|_2 \le \widetilde{O}(\sqrt{s}) \tag{6.7}$$

Its proof is in the same spirit as (6.6), with the only difference being the spectral norm of B is around $\sqrt{m/d}$ as opposed to $O(1)$.

6.6 Stability After Adversarial Perturbation

In this section we study the behavior of RNN after adversarial perturbation. The corresponding precise statements and proofs are in Section 6.12.

Letting $\widetilde{W}, A, B$ be at random initialization, we consider some matrix $W = \widetilde{W} + W'$ for $\|W'\|_2 \leq \frac{\text{poly}(\varrho)}{\sqrt{m}}$. Here, W' may depend on the randomness of $\widetilde{W}, A$ and B, so we say it can be *adversarially* chosen. The results of this section will later be applied essentially twice:

- Once for those updates generated by GD or SGD, where W' is how much the algorithm has moved away from the random initialization.
- The other time (see Section 6.7.3) for a technique that we call "randomness decomposition" where we decompose the true random initialization W into $W = \widetilde{W} + W'$, where $\widetilde{W}$ is a "fake" random initialization but identically distributed as W. Such technique at least traces back to smooth analysis [ST04].

To illustrate our high-level idea, from this section on (so in Section 6.6, 6.7 and 6.8)

we ignore the polynomial dependency in ϱ and *hide it in the big-O notion.*

We denote by $\widetilde{D}_{i,\ell}$, $\widetilde{g}_{i,\ell}$, $\widetilde{h}_{i,\ell}$ respectively the values of $D_{i,\ell}$, $g_{i,\ell}$ and $h_{i,\ell}$ determined by $\widetilde{W}$ and A at random initialization; and by $D_{i,\ell} = \widetilde{D}_{i,\ell} + D'_{i,\ell}$, $g_{i,\ell} = \widetilde{g}_{i,\ell} + g'_{i,\ell}$ and $h_{i,\ell} = \widetilde{h}_{i,\ell} + h'_{i,\ell}$ respectively those determined by $W = \widetilde{W} + W'$ after the adversarial perturbation.

Forward Stability Our first, and most technical result is the following:

$$\|g'_{i,\ell}\|_2, \|h'_{i,\ell}\|_2 \leq O(m^{-1/2}) \ , \quad \|D'_{i,\ell}\|_0 \leq O(m^{2/3}) \quad \text{and} \quad \|D'_{i,\ell} g_{i,\ell}\|_2 \leq O(m^{-1/2}) \ . \tag{6.8}$$

Intuitively, one may hope to prove (6.8) by induction, because we have (ignoring subscripts in i)

$$g'_{\ell'} = \underbrace{W' D_{\ell'-1} g_{\ell'-1}}_{①} + \underbrace{\widetilde{W} D'_{\ell'-1} g_{\ell'-1}}_{②} + \underbrace{\widetilde{W} \widetilde{D}_{\ell'-1} g'_{\ell'-1}}_{③} \ .$$

The main issue here is that, the spectral norm of $\widetilde{W}\widetilde{D}_{\ell'-1}$ in ③ is greater than 1, so we cannot apply naive induction due to exponential blow up in L. Neither can we apply techniques from Section 6.5, because the changes such as $g_{\ell'-1}$ can be *adversarial*.

In our actual proof of (6.8), instead of applying induction on ③, we recursively expand ③ by the above formula. This results in a total of L terms of ① type and L terms of ② type. The main difficulty is to bound a term of ② type, that is:

$$\big\|\widetilde{W}\widetilde{D}_{\ell_1} \cdots \widetilde{D}_{\ell_2+1} \widetilde{W} D'_{\ell_2} g_{\ell_2}\big\|_2$$

Our argument consists of two conceptual steps.

1. Suppose $g_{\ell_2} = \widetilde{g}_{\ell_2} + g'_{\ell_2} = \widetilde{g}_{\ell_2} + g'_{\ell_2,1} + g'_{\ell_2,2}$ where $\|g'_{\ell_2,1}\|_2 \le m^{-1/2}$ and $\|g'_{\ell_2,2}\|_\infty \le m^{-1}$, then we argue that $\|D'_{\ell_2} g_{\ell_2}\|_2 \le O(m^{-1/2})$ and $\|D'_{\ell_2} g_{\ell_2}\|_0 \le O(m^{2/3})$.

2. Suppose $x \in \mathbb{R}^m$ with $\|x\|_2 \le m^{-1/2}$ and $\|x\|_0 \le m^{2/3}$, then we show that $y = \widetilde{W}\widetilde{D}_{\ell_1} \cdots \widetilde{D}_{\ell_2+1}\widetilde{W}x$ can be written as $y = y_1 + y_2$ with $\|y_1\|_2 \le O(m^{-2/3})$ and $\|y_2\|_\infty \le O(m^{-1})$.

The two steps above enable us to perform induction without exponential blow up. Indeed, they together enable us to go through the following logic chain:

$$\left.\begin{array}{l} \|\cdot\|_2 \le m^{-1/2} \text{ and } \|\cdot\|_\infty \le m^{-1} \\ \|\cdot\|_2 \le m^{-2/3} \text{ and } \|\cdot\|_\infty \le m^{-1} \end{array}\right. \begin{array}{c} \overset{(1)}{\Longrightarrow} \\ \overset{}{\Longleftarrow}_{(2)} \end{array} \left.\right\} \|\cdot\|_2 \le m^{-1/2} \text{ and } \|\cdot\|_0 \le m^{2/3}$$

Since there is a gap between $m^{-1/2}$ and $m^{-2/3}$, we can make sure that all blow-up factors are absorbed into this gap, using the property that m is polynomially large. This enables us to perform induction to prove (6.8) without exponential blow-up.

Intermediate Layers and Backward Stability Using (6.8), and especially using the sparsity $\|D'\|_0 \le m^{2/3}$ from (6.8), one can apply the results in Section 6.5 to derive the following stability bounds for intermediate layers and backward propagation:

$$\left\|D_{i,\ell_1}W\cdots D_{i,\ell_2}W - \widetilde{D}_{i,\ell_1}\widetilde{W}\cdots\widetilde{D}_{i,\ell_2}\widetilde{W}\right\|_2 \le O(L^7) \tag{6.9}$$

$$\left\|BD_{i,\ell_1}W\cdots D_{i,\ell_2}W - B\widetilde{D}_{i,\ell_1}\widetilde{W}\cdots\widetilde{D}_{i,\ell_2}\widetilde{W}\right\|_2 \le O\big(m^{1/3}\big)\ . \tag{6.10}$$

Special Rank-1 Perturbation For technical reasons, we also need two bounds in the special case of $W' = yz^\top$ for some unit vector z and sparse y with $\|y\|_0 \le \mathrm{poly}(\varrho)$. We prove that, for this type of rank-one adversarial perturbation, it satisfies for every $k \in [m]$:

$$|((\widetilde{W}+W')h'_{i,\ell})_k| \le O\big(m^{-2/3}\big) \tag{6.11}$$

$$\left\|BD_{i,\ell_1}W\cdots D_{i,\ell_2}We_k - B\widetilde{D}_{i,\ell_1}\widetilde{W}\cdots\widetilde{D}_{i,\ell_2}\widetilde{W}e_k\right\|_2 \le O\big(m^{-1/6}\big) \tag{6.12}$$

6.7 Proof Sketch of Theorem 6.15.2: Polyak-Łojasiewicz Condition

The upper bound in Theorem 6.15.2 is easy to prove (based on Section 6.5 and 6.6), but the lower bound (a.k.a. the Polyak-Łojasiewicz condition) is the most technically involved result to prove in this paper. We introduce the notion of "fake gradient". Given *fixed* vectors $\{\mathsf{loss}_{i,a}\}_{i\in[n],a\in\{2,\dots,L\}}$, we define

$$\widehat{\nabla}_k f(W) \stackrel{\text{def}}{=} \sum_{i=1}^{n}\sum_{\ell=1}^{L-1}(u_{i,\ell})_k \cdot h_{i,\ell} \cdot \mathbf{1}_{(g_{i,\ell+1})_k\geq 0} \tag{6.13}$$

where $u_{i,\ell} \stackrel{\text{def}}{=} \sum_{a=\ell+1}^{L} \mathsf{Back}_{i,\ell+1\to a}^\top \cdot \mathsf{loss}_{i,a}$. Note that if $\mathsf{loss}_{i,a} = Bh_{i,a} - y^*_{i,a}$ is the *true* loss vector, then $\widehat{\nabla}_k f(W)$ will be identical to $\nabla_k f(W)$ by Fact 6.2.3. Our main technical theorem is the following:

Theorem 6.7.1. *For every fixed vectors* $\{\mathsf{loss}_{i,a}\}_{i\in[n],a\in\{2,\dots,L\}}$*, if* W, A, B *are at* random initialization*, then with high probability*

$$\|\widehat{\nabla} f(W)\|_F^2 \geq \widetilde{\Omega}\left(\frac{\delta m}{\mathrm{poly}(\rho)}\right) \times \max_{i,\ell}\{\|\mathsf{loss}_{i,\ell}\|^2\} \ .$$

There are only two conceptually simple steps from Theorem 6.14.2 to Theorem 6.15.2 (see Section 6.15).

- First, one can use the stability lemmas in Section 6.6 to show that, the fake gradient $\|\widehat{\nabla} f(W+W')\|_F$ after adversarial perturbation W' (with $\|W'\|_2 \leq \frac{1}{\sqrt{m}}$) is also large.
- Second, one can apply ϵ-net and union bound to turn "fixed loss" into "for all loss". This allows us to turn the lower bound on the fake gradient into a lower bound on the true gradient $\|\nabla_k f(W+W')\|_F$.

Therefore, in the rest of this section, we only sketch the ideas behind proving Theorem 6.14.2.

Let $(i^*, \ell^*) = \text{argmax}_{i,\ell}\{\| \mathsf{loss}_{i,\ell} \|_2\}$ be the sample and layer corresponding to the largest loss. Recall W, A, B are at random initialization.

6.7.1 Indicator and Backward Coordinate Bounds

There are three factors in the notion of fake gradient (6.13): $(u_{i,\ell})_k \in \mathbb{R}$ the backward coordinate, $h_{i,\ell} \in \mathbb{R}^m$ the forward vector, and $\mathbf{1}_{(g_{i,\ell+1})_k \geq 0} \in \{0,1\}$ the indicator coordinate. We already know very well how the forward vector $h_{i,\ell}$ behaves from the previous sections. Let us provide bounds on the other two factors at the random initialization. (Details in Section 6.13.)

Our "backward coordinate bound" controls the value of $(u_{i,\ell})_k$: at random initialization,

$$\left|(u_{i^*,\ell^*})_k\right| \geq \frac{\|\mathsf{loss}_{i^*,\ell^*}\|_2}{\rho} \quad \text{for at least } 1 - o(1) \text{ fraction of } k \in [m] \tag{6.14}$$

The main idea behind proving (6.14) is to use the randomness of B. For a fixed $k \in [m]$, it is in fact not hard to show that $\left|(u_{i^*,\ell^*})_k\right|$ is large with high probability. Unfortunately, the randomness of B are shared for different coordinates k. We need to also bound the correlation between pairs of coordinates $k_1, k_2 \in [m]$, and resort to MiDiarmid inequality to provide a high concentration bound with respect to all the coordinates.

Our indicator coordinate bound controls the value $(g_{i,\ell+1})_k$ inside the indicator functions $\mathbf{1}_{(g_{i,\ell+1})_k \geq 0}$. It says, letting $\beta_+ \stackrel{\text{def}}{=} \frac{\delta}{\rho^2}$ and $\beta_- \stackrel{\text{def}}{=} \frac{\delta}{\rho^{10}}$, then at random initialization, for at least $\frac{\delta}{\mathrm{poly}(\rho)}$ *fraction* of the coordinates $k \in [m]$,

$$|(g_{i^*,\ell^*+1})_k| \leq \frac{\beta_-}{\sqrt{m}} \quad \text{and} \quad |(g_{i,\ell+1})_k| \geq \frac{\beta_+}{\sqrt{m}} \qquad \begin{array}{l} \text{for } i \neq i^* \text{ and } \ell = \ell^*, \text{ or} \\ \text{for } i \in [n] \text{ and } \ell > \ell^*. \end{array} \tag{6.15}$$

This should be quite intuitive to prove, in the following two steps.

- First, there are $\frac{\delta}{\mathrm{poly}(\rho)} m$ coordinates $k \in [m]$ with $|(g_{i^*,\ell^*+1})_k| \leq \frac{\beta_-}{\sqrt{m}}$.

To show this, we write $(g_{i^*,\ell^*+1})_k = (WU_{\ell^*}U_{\ell^*}h_{i^*,\ell^*} + Ax_{i^*,\ell^*+1})_k$, and prove that for every $z \in \mathbb{R}^{n\ell^*}$ with bounded norms (by ϵ-net), there are at least $\frac{\delta}{\mathrm{poly}(\rho)}m$ coordinates $k \in [m]$ with $|(WU_{\ell^*}z + Ax_{i^*,\ell^*+1})_k| \leq \frac{\beta_-}{\sqrt{m}}$. This is possible using the independence between WU_{ℓ^*} and A.

- Then, *conditioning* on the first event happens, we look at $|(g_{i,\ell+1})_k|$ for (1) each $i \neq i^*$ and $\ell = \ell^*$, or (2) each $i \in [n]$ and $\ell > \ell^*$. In both cases, even though the value of g_{i^*,ℓ^*+1} is fixed, we still have sufficient fresh new randomness (by invoking (6.4) for case (1) and (6.3) for case (2)). Such additional randomness can make sure that, with high probability (over the fresh new randomness), the value of $(g_{i,\ell+1})_k$ is larger than $\frac{\beta_+}{\sqrt{m}}$.

6.7.2 Thought Experiment: Adding Small Rank-One Perturbation

We now focus on a fixed coordinate $k \in [m]$ satisfying (6.15) and (6.14) in the Section 6.7.1, and denote by $v_{i^*,\ell^*} \stackrel{\text{def}}{=} \frac{(I-U_{\ell^*-1}U_{\ell^*-1}^\top)h_{i^*,\ell^*}}{\|(I-U_{\ell^*-1}U_{\ell^*-1}^\top)h_{i^*,\ell^*}\|}$.

For analysis purpose, imagine that we apply a small *random* perturbation W'_k to the already-randomly initialized matrix W, in the rank-one direction $e_k v_{i^*,\ell^*}^\top$. Namely, we set $W'_k = g \cdot e_k v_{i^*,\ell^*}^\top$ where $g \sim \mathcal{N}(0,\theta^2)$ and θ is a parameter satisfying $\frac{\beta_-}{\sqrt{m}} \ll \theta \ll \frac{\beta_+}{\sqrt{m}}$. Using the fact that k satisfies (6.15), one can show that[11]

(a) $\mathbf{1}_{(g_{i,\ell+1})_k \geq 0}$ stays the same with respect to perturbation W'_k, except for $i = i^*$ and $\ell = \ell^*$; and

(b) $\mathbf{1}_{(g_{i^*,\ell^*+1})_k \geq 0}$ can be 0 or 1 each with at least constant probability over W'_k.

At the same time, using the fact that k satisfies (6.14), one can show that

(c) $h_{i,\ell}$ and $(u_{i,\ell})_k$ does not change by much (owing to Section 6.6); and

(d) $|(u_{i^*,\ell^*+1})_k| \geq \frac{\|\mathsf{loss}_{i^*,\ell^*}\|_2}{\rho}$ is large (owing to (6.14)).

[11] Specifically,

- For every $\ell < \ell^*$, we have $(g_{i,\ell+1})_k$ is unchanged with respect to W'_k. This is because $g_{i,\ell+1}$ only depends on the randomness of WU_ℓ, but v_{i^*,ℓ^*} is orthogonal to the columns of U_ℓ.
- For $i = i^*$ and $i = \ell^*$, we have $(g_{i,\ell+1})_k$ will change sign with constant probability with respect to W'_k. This is because $\|W'_k\|$ is above $\frac{\beta_-}{\sqrt{m}}$ — the original $(g_{i,\ell+1})_k$ before perturbation.
- For any other i and ℓ, we have $(g_{i,\ell+1})_k$ will not change sign with high probability with respect to W'_k. This is because $\|W'_k\|$ is below $\frac{\beta_+}{\sqrt{m}}$ — the original $(g_{i,\ell+1})_k$ before perturbation.

Putting (a), (b), (c), and (d) together, we know for such specially chosen k, at least with constant probability over the random perturbation of W'_k,

$$\|\widehat{\nabla}_k f(W + W'_k)\|_F \geq \Omega\Big(\frac{\| \mathsf{loss}_{i^*,\ell^*} \|_2}{\rho}\Big) \ . \tag{6.16}$$

6.7.3 Real Proof: Randomness Decomposition and McDiarmid's Inequality

There are only two main differences between (6.16) and our desired Theorem 6.14.2. First, (6.16) gives a gradient lower bound at $W+W'_k$, while in Theorem 6.14.2 we need a gradient lower bound at random initialization W. Second, (6.16) gives a lower bound on $\widehat{\nabla}_k f(\cdot)$ with *constant probability* for a small fraction of good coordinates k, but in Theorem 6.14.2 we need a lower bound for the entire $\widehat{\nabla} f(\cdot)$.

Randomness Decomposition To fix the first issue, we resort to a randomness decomposition technique at least tracing back to the smooth analysis of [ST04]:

Proposition 6.7.2. *Given small constant $\theta \in (0,1)$ and m-dimensional random $g \sim \mathcal{N}(0, \frac{1}{m}I)$, we can rewrite $g = g_1 + g_3$ where g_1 follows from $\mathcal{N}(0, \frac{1}{m}I)$ and g_3 is very close to $\mathcal{N}(0, \frac{\theta^2}{m}I)$.*

(Note that there is no contradiction here because g_1 and g_3 shall be correlated.)

Proof. Let $g_1, g_2 \in \mathbb{R}^m$ be two independent random vectors sampled from $\mathcal{N}(0, \frac{1}{m}I)$. We can couple them and make sure $g = \sqrt{1-\theta^2}g_1 + \theta g_2$. We now choose $g_3 = \theta g_2 - (1-\sqrt{1-\theta^2})g_1$. Since $(1-\sqrt{1-\theta^2}) \le O(\theta^2)$, when θ is sufficiently small, we know that g_3 is close to being generated from distribution $\mathcal{N}(0, \frac{\theta^2}{m}I)$. □

Using Proposition 6.7.2, for each good coordinate k, instead of "adding" perturbation W'_k to W, we can instead decompose W into $W = W_0 + W'_k$, where W_0 is distributed in the same way as W. In other words, W_0 is also at random initialization. If this idea is carefully implemented, one can immediately turn (6.16) into

$$\|\widehat{\nabla}_k f(W)\|_F = \|\widehat{\nabla}_k f(W_0 + W'_k)\|_F \ge \Omega\Big(\frac{\|\,\mathsf{loss}_{i^*,\ell^*}\,\|_2}{\rho}\Big)\ . \tag{6.17}$$

Extended McDiarmid's Inequality To fix the second issue, one may wish to consider all the indices $k \in [m]$ satisfying (6.15) and (6.14). Since there are at least $\frac{\delta}{\mathrm{poly}(\rho)}$ fraction of such coordinates, if all of them satisfied (6.17), then we would have already proved Theorem 6.14.2. Unfortunately, neither can we apply Chernoff bound (because the events with different $k \in [m]$ are correlated), nor can we apply union bound (because the event occurs only with constant probability).

Our technique is to resort to (an extended probabilistic variant of) McDiarmid's inequality (see Section 6.10.6) in a very non-trivial way to boost the confidence.

Give any fixed subset N of cardinality $|N| = (\mathrm{poly}(\rho)/\delta)^2$, one can show that there again exists $\frac{\delta}{\mathrm{poly}(\rho)}$ fraction of coordinate $k \in N$ satisfying (6.15) and (6.14).[12] Now, instead of decomposing $W = W_0 + W'_k$, we decompose it as $W = W_1 + W'_N$ for $W'_N = u_N v_{i^*,\ell^*}^\top$ where u_N is only supported on coordinates $k \in N$. In other words, we *simultaneously* perturb in the directions of W'_k for all $k \in N$. Since this perturbation is small enough —i.e., $\|W'_N\|_2 \leq \frac{\mathrm{poly}(\rho)}{\sqrt{m}}$— one can show that (6.17) remains true, that is, for a large fraction of $k \in N$, with at least constant probability over W'_N:

$$\|\widehat{\nabla}_k f(W_1 + W'_N)\|_F \geq \Omega\Big(\frac{\|\,\mathsf{loss}_{i^*,\ell^*}\,\|_2}{\rho}\Big) \ . \tag{6.18}$$

In order to apply McDiarmid's, we next need to bound on the difference between $\|\widehat{\nabla}_k f(W_1 + W'_N)\|_F$ and $\|\widehat{\nabla}_k f(W_1 + W'_N + W''_j)\|_F$ for an arbitrary (but small) perturbation W''_j (in the direction of $e_j v_{i^*,\ell^*}^\top$). We show that,

$$\sum_{k\in[N]} \Big| \|\widehat{\nabla}_k(W_1 + W'_N)\|_2^2 - \|\widehat{\nabla}_k(W_1 + W'_N + W''_j)\|_2^2 \Big| \leq O\left(\rho^8 + \frac{|N|}{m^{1/6}}\right) \ . \tag{6.19}$$

[12] We use $(\mathrm{poly}(\rho))^2$ to emphasize that the first polynomial needs to be bigger than the second $\mathrm{poly}(\rho)$.

In other words, although there are $|N|$ difference terms, their total summation only grows in rate $\frac{|N|}{m^{1/6}}$ according to (6.19). After applying a variant of McDiarmid's inequality, we derive that with *high probability* over W'_N, it satisfies

$$\sum_{k\in N} \|\widehat{\nabla}_k f(W)\|_F^2 = \sum_{k\in N} \|\widehat{\nabla}_k f(W_1 + W'_N)\|_F^2 \geq \Omega\Big(\frac{\|\,\mathsf{loss}_{i^*,\ell^*}\,\|_2^2}{\mathrm{poly}(\rho)}|N|\Big)\ . \tag{6.20}$$

Finally, by sampling sufficiently many random sets N to cover the entire space $[m]$, we can show

$$\|\widehat{\nabla} f(W)\|_F^2 = \sum_{k\in[m]} \|\widehat{\nabla}_k f(W)\|_F^2 \geq \Omega\Big(\frac{\|\,\mathsf{loss}_{i^*,\ell^*}\,\|_2^2}{\mathrm{poly}(\rho)}m\Big)\ .$$

and therefore Theorem 6.14.2 holds.

6.8 Proof Sketch of Theorem 6.16.1: Objective Semi-Smoothness

The objective semi-smoothness Theorem 6.16.1 turns out to be much simpler to prove than Theorem 6.15.2. It only relies on Section 6.5 and 6.6, and does not need randomness decomposition or McDiarmid's inequality. (Details in Section 6.16.)

Recall that in Theorem 6.16.1, $\widetilde{W}, A, B$ are at random initialization. $\breve{W}$ is an adversarially chosen matrix with $\|\breve{W}-\widetilde{W}\| \le \frac{\mathrm{poly}(\varrho)}{\sqrt{m}}$, and W' is some other adversarial perturbation on top of $\breve{W}$, satisfying $\|W'\| \le \frac{\tau_0}{\sqrt{m}}$. We denote by

- $\widetilde{D}_{i,\ell}$, $\widetilde{g}_{i,\ell}$, $\widetilde{h}_{i,\ell}$ respectively the values of $D_{i,\ell}$, $g_{i,\ell}$, $h_{i,\ell}$ determined by weight matrix $\widetilde{W}$;
- $\breve{D}_{i,\ell}$, $\breve{g}_{i,\ell}$, $\hbar_{i,\ell}$, $\breve{\mathsf{loss}}_{i,\ell}$ respectively those of $D_{i,\ell}$, $g_{i,\ell}$, $h_{i,\ell}$ and $\mathsf{loss}_{i,\ell}$ at weight matrix $\breve{W}$; and
- $D_{i,\ell}$, $g_{i,\ell}$, $h_{i,\ell}$ respectively the values of $D_{i,\ell}$, $g_{i,\ell}$, $h_{i,\ell}$ at weight matrix $W = \breve{W} + W'$.

Our main tool is to derive the following strong formula for $h_{i,\ell} - \hbar_{i,\ell}$: there exist diagonal matrices $D''_{i,\ell} \in \mathbb{R}^{m\times m}$ with entries in $[-1,1]$ and sparsity $\|D''_{i,\ell}\|_0 \le O(m^{2/3})$ such that,

$$h_{i,\ell} - \hbar_{i,\ell} = \sum_{a=1}^{\ell-1} (\breve{D}_{i,\ell} + D''_{i,\ell})\breve{W}\cdots\breve{W}(\breve{D}_{i,a+1} + D''_{i,a+1})W'h_{i,a} \tag{6.21}$$

In particular, (6.21) implies $\|h_{i,\ell} - \hbar_{i,\ell}\| \le O(L^9)\|W'\|_2$ after careful linear-algebraic manipulations (esp. using (6.6)). The main take-away message from (6.21) is that, this difference $h_{i,\ell} - \hbar_{i,\ell}$ is proportional to the norm of the perturbation, $\|W'\|_2$, no matter how small it is. In contrast, in (6.2), we only derived a weak upper bound of the form $\|h_{i,\ell} - \widetilde{h}_{i,\ell}\| \le O(m^{-1/2})$. Nevertheless, the proof of (6.21) relies on (6.2), so we are not duplicating proofs.

Finally, we carefully derive that

$$\begin{aligned}
&f(\breve{W}+W')-f(W)-\langle\nabla f(W),W'\rangle \\
&=\sum_{i=1}^{n}\sum_{\ell=2}^{L}\breve{\mathsf{loss}}_{i,\ell}^{\top}B\left((h_{i,\ell}-\hbar_{i,\ell})-\sum_{a=1}^{\ell-1}\breve{D}_{i,\ell}\breve{W}\cdots\breve{W}\breve{D}_{i,a+1}W'\hbar_{i,a}\right)+\frac{1}{2}\|B(h_{i,\ell}-\hbar_{i,\ell})\|^2
\end{aligned} \tag{6.22}$$

and plug (6.21) into (6.22) to derive our final Theorem 6.16.1.

6.9 Roadmap of Later Proof Details

- Section 6.10 recalls some old lemmas and derives some new lemmas in probability theory.
- Section 6.11 serves for Section 6.5, the basic properties at random initialization.
- Section 6.12 serves for Section 6.6, the stability after adversarial perturbation.
- Section 6.13, 6.14 and 6.15 together serve for Section 6.7 and prove Theorem 6.15.2, the Polyak-Łojasiewicz condition and gradient upper bound. In particular:
  - Section 6.13 serves for Section 6.7.1 (the indicator and backward coordinate bounds).
  - Section 6.14 serves for Section 6.7.3 (the randomness decomposition and McDiarmid's inequality) and proves Theorem 6.14.2.
  - Section 6.15 shows how to go from Theorem 6.14.2 to Theorem 6.15.2.
- Section 6.16 serves for Section 6.8, the proof of Theorem 6.16.1, the objective semi-smoothness.
- Section 6.17 gives the final proof for Theorem 6.17.1, the GD convergence theorem.
- Section 6.18 gives the final proof for Theorem 6.18.1, the SGD convergence theorem.

Parameters We also summarize a few parameters we shall use in the proofs.

- In Definition 6.13.1, we shall introduce two parameters

$$\beta_+ \stackrel{\text{def}}{=} \frac{\delta}{\rho^2} \text{ and } \beta_- \stackrel{\text{def}}{=} \frac{\delta}{\rho^{10}}$$

to control the thresholds of indicator functions (recall Section 6.7.1).

- In Definition 6.14.2, we shall introduce parameter

$$\theta \in \left[\rho^4 \cdot \beta_-, \rho^{-3} \cdot \beta_+\right]$$

to describe how much randomness we want to decompose out of W (recall Section 6.7.2).

- In (6.60) of Section 6.14.4, we shall choose

$$N = \frac{\rho^{22}}{\beta_-^2}$$

which controls the size of the set N where we apply McDiarmid's inequality (recall Section 6.7.3).

6.10 Preliminaries on Probability Theory

The goal of this section is to present a list of probability tools.

- In Section 6.10.1, we recall how to swap randomness.
- In Section 6.10.2, we recall concentration bounds for the chi-square distribution.
- In Section 6.10.3, we proved a concentration bound of sum of squares of ReLU of Gaussians.
- In Section 6.10.4 and Section 6.10.5, we show some properties for random Gaussian vectors.
- In Section 6.10.6, we recall the classical McDiarmid's inequality and then prove a general version of it.

6.10.1 Swapping Randomness

Fact 6.10.1 (probability splitting)**.** *If $f(X,Y)$ holds with probability at least $1-\epsilon$, then*

- *with probability at least $1-\sqrt{\epsilon}$ (over randomness of X), the following event holds,*
  - *$f(X,Y)$ holds with probability at least $1-\sqrt{\epsilon}$ (over randomness of Y).*

In other words,

$$\Pr_X\left[\Pr_Y[f(X,Y)] \geq 1-\sqrt{\epsilon}\right] \geq 1-\sqrt{\epsilon}.$$

Fact 6.10.2 (swapping probability and expectation)**.** *If $\Pr_{X,Y}[f(X,Y) \geq a] \geq \epsilon$, then*

$$\Pr_X\left[\mathop{\mathbb{E}}_Y[f(X,Y)] \geq a\epsilon/2\right] \geq \epsilon/2.$$

Proof. We prove it by making a contradiction,

Suppose

$$\Pr_X\left[\mathop{\mathbb{E}}_Y[f(X,Y)] \leq a\epsilon/2\right] \geq 1-\epsilon/2,$$

which implies that

$$\Pr_{X,Y}[f(X,Y) \geq a|X] \leq \epsilon.$$

□

6.10.2 Concentration of Chi-Square Distribution

Lemma 6.10.3 (Lemma 1 on page 1325 of [LM00]). *Let $X \sim \mathcal{X}_k^2$ be a chi-squared distributed random variable with k degrees of freedom. Each one has zero mean and σ^2 variance. Then*

$$\Pr[X - k\sigma^2 \geq (2\sqrt{kt} + 2t)\sigma^2] \leq \exp(-t)$$
$$\Pr[k\sigma^2 - X \geq 2\sqrt{kt}\sigma^2] \leq \exp(-t)$$

One straightforward application is

Lemma 6.10.4. *Let $x_1, x_2, \cdots, x_n$ denote i.i.d. samples from $\mathcal{N}(0, \sigma^2)$. For any $b \geq 1$, we have*

$$\Pr\left[|\|x\|_2^2 - n\sigma^2| \geq \frac{n}{b}\sigma^2\right] \leq 2\exp(-n/(8b^2)).$$

Proof. We choose $t = k/(8b^2)$ in Lemma 6.10.3,

$$\Pr\left[\left|\|x\|_2^2 - n\sigma^2\right| \geq (\frac{2n}{\sqrt{8}b} + \frac{2n}{8b^2})\sigma^2\right] \leq 2\exp(-n/(8b^2)).$$

Since $\frac{2n}{\sqrt{8}b} + \frac{2n}{8b^2} \leq \frac{2n}{\sqrt{8}b} + \frac{2n}{8b} \leq n/b$. Thus,

$$\Pr\left[\left|\|x\|_2^2 - n\sigma^2\right| \geq \frac{n}{b}\sigma^2\right] \leq 2\exp(-n/(8b^2)),$$

which completes the proof. □

Lemma 6.10.5. *Let $x_1, x_2, \cdots, x_m$ denote i.i.d. samples from $\mathcal{N}(0, 1)$, and $y_i = \max\{x_i^2 - \log m, 0\}$. We have*

$$\Pr\left[\sum_{i=1}^{m} y_i \geq 2\sqrt{m}\right] \leq e^{-\Omega(\sqrt{m})} .$$

Proof. First of all, letting $g = \sqrt{\log m}$,

$$\mathbb{E}[y_i] = \int_g^\infty \frac{\exp\left(-\frac{x^2}{2}\right)(x-g)^2}{\sqrt{2\pi}}\,dx = \frac{1}{2}\left(g^2+1\right)\operatorname{erfc}\left(\frac{g}{\sqrt{2}}\right) - \frac{e^{-\frac{g^2}{2}}g}{\sqrt{2\pi}} \leq \frac{1}{e^{\frac{g^2}{2}}} = \frac{1}{\sqrt{m}}\ .$$

On the other hand, each random variable y_i is $O(1)$-subgaussian. By subgaussian concentration,

$$\Pr\left[\sum_{i=1}^m y_i \geq 2\sqrt{m}\right] \leq e^{-\Omega(\sqrt{m})}$$

□

6.10.3 Concentration of Sum of Squares of ReLU of Gaussians

Lemma 6.10.6 (Upper bound)**.** *Given n i.i.d. Gaussian random variables $x_1, x_2, \cdots, x_n \sim \mathcal{N}(0, \sigma^2)$, we have*

$$\Pr\left[\left(\sum_{i=1}^{n} \max(x_i, 0)^2\right)^{1/2} < (1+\epsilon)\sqrt{n/2}\sigma\right] \geq 1 - \exp(-\epsilon^2 n/100).$$

Proof. Using Chernoff bound, we know that with probability $1-\exp(-\epsilon^2 n/6)$, $\sum_{i=1}^{n} \max(x_i, 0)^2$ is a at most degree-$(1+\epsilon)\frac{n}{2}$ Chi-square random variable. Let us say this is the first event.

Using Lemma 6.10.3, we have

$$\begin{aligned}
& \Pr[X \geq k\sigma^2 + (2\sqrt{kt} + 2t)\sigma^2] \leq \exp(-t) \\
\implies & \Pr[X \geq k\sigma^2 + (2\epsilon k + 2\epsilon^2 k)\sigma^2] \leq \exp(-\epsilon^2 k) && \text{by choosing } t = \epsilon^2 k \\
\implies & \Pr[X \geq k(1+4\epsilon)\sigma^2] \leq \exp(-\epsilon^2 k) \\
\implies & \Pr\left[X \geq (1+\epsilon)(1+4\epsilon)\frac{n}{2}\sigma^2\right] \leq \exp(-\epsilon^2(1+\epsilon)n/2) && \text{by } k = (1+\epsilon)n/2
\end{aligned}$$

Thus, we have with probability at least $1 - \exp(-\epsilon^2 n/2)$,

$$X \leq (1+4\epsilon)^2 \frac{n}{2}\sigma^2.$$

Let the above event denote the second event.

By taking the union bound of two events, we have with probability $1 - 2\exp(-\epsilon^2 n/6)$

$$\left(\sum_{i=1}^{n} \max(x_i, 0)^2\right)^{1/2} \leq X \leq (1+4\epsilon)\frac{n}{2}\sigma$$

Then rescaling the ϵ, we get the desired result. □

Lemma 6.10.7 (Lower bound). *Given n i.i.d. Gaussian random variables $x_1, x_2, \cdots, x_n \sim \mathcal{N}(0, \sigma^2)$, we have*

$$\Pr\left[\left(\sum_{i=1}^{n} \max(x_i, 0)^2\right)^{1/2} > (1-\epsilon)\sqrt{n/2}\sigma\right] \geq 1 - \exp(\epsilon^2 n/100)$$

Proof. Using Chernoff bound, we know that with probability $1-\exp(-\epsilon^2 n/6)$, $\sum_{i=1}^{n} \max(x_i, 0)^2$ is a at most degree-$(1-\epsilon)\frac{n}{2}$ Chi-square random variable. Let us say this is the first event.

Using Lemma 6.10.3, we have

$$\begin{aligned}
& \Pr[X \leq k\sigma^2 - 2\sqrt{kt}\sigma^2] \leq \exp(-t) \\
\implies & \Pr[X \leq k\sigma^2 - 2\epsilon k\sigma^2] \leq \exp(-\epsilon^2 k) && \text{by choosing } t = \epsilon^2 k \\
\implies & \Pr[X \leq k\sigma^2(1-2\epsilon)] \leq \exp(-\epsilon^2 k) \\
\implies & \Pr\left[X \leq (1-\epsilon)(1-2\epsilon)\frac{n}{2}\sigma^2\right] \leq \exp(-\epsilon^2(1-\epsilon)n/2) && \text{by } k = (1-\epsilon)n/2
\end{aligned}$$

Thus, we have with probability at least $1 - \exp(-\epsilon^2 n/4)$,

$$X \geq (1-2\epsilon)^2 \frac{n}{2}\sigma^2.$$

Let the above event denote the second event.

By taking the union bound of two events, we have with probability at least $1 - 2\exp(-\epsilon^2 n/6)$,

$$\left(\sum_{i=1}^{n} \max(x_i, 0)^2\right)^{1/2} \geq X \geq (1-2\epsilon)\frac{n}{2}\sigma.$$

Then rescaling the ϵ, we get the desired result. □

Combining Lemma 6.10.7 and Lemma 6.10.6, we have

Lemma 6.10.8 (Two sides bound)**.** *Given n i.i.d. Gaussian random variables $x_1, x_2, \cdots, x_n \sim \mathcal{N}(0, \sigma^2)$, let $\phi(a) = \max(a, 0)^2$. We have*

$$\Pr_x \left[\|\phi(x)\|_2 \in ((1-\epsilon)\sqrt{n/2}\sigma, (1+\epsilon)\sqrt{n/2}\sigma) \right] \geq 1 - 2\exp(-\epsilon^2 n/100)$$

Corollary 6.10.9 (Two sides bound for single matrix)**.** *Let $x \in \mathbb{R}^n$ denote a fixed vector. Given a random Gaussian matrix $A \in \mathbb{R}^{m \times n}$ where each entry is i.i.d. sampled from $\mathcal{N}(0, 2\sigma^2/m)$.*

$$\Pr_A [\|\phi(Ax)\|_2 \in ((1-\epsilon)\|x\|_2\sigma, (1+\epsilon)\|x\|_2\sigma)] \geq 1 - 2\exp(-\epsilon^2 m/100).$$

Proof. For each $i \in [m]$, let $y_i = (Ax)_i$. Then $y_i \sim \mathcal{N}(0, \widetilde{\sigma}^2)$, where $\widetilde{\sigma}^2 = 2\sigma^2/m \cdot \|x\|_2^2$. Using Corollary 6.10.8, we have

$$\Pr_y \left[\|\phi(y)\|_2 \in ((1-\epsilon)\sqrt{m/2}\widetilde{\sigma}, (1+\epsilon)\sqrt{m/2}\widetilde{\sigma}) \right] \geq 1 - 2\exp(-\epsilon m/100)$$

which implies

$$\Pr_y [\|\phi(y)\|_2 \in ((1-\epsilon)\|x\|_2\sigma, (1+\epsilon)\|x\|_2\sigma)] \geq 1 - 2\exp(-\epsilon^2 m/100).$$

Since $y = Ax$, thus we complete the proof. □

Corollary 6.10.10 (Two sides bound for multiple matrices)**.** *Let $x_1, x_2, \cdots, x_k$ denote k fixed vectors where $x_i \in \mathbb{R}^{n_i}$. Let $x = [x_1^\top \ x_2^\top \ \cdots \ x_k^\top]^\top \in \mathbb{R}^n$ where $n = \sum_{i=1}^k n_i$. Let $A_1, A_2, \cdots A_k$ denote k independent random Gaussian matrices where each entry of $A_i \in \mathbb{R}^{m \times n_i}$ is i.i.d. sampled from $\mathcal{N}(0, 2\sigma_i^2/m)$ for each $i \in [k]$. Let $A = [A_1 \ A_2 \ \cdots \ A_k]$. We have*

$$\Pr_A \left[\|\phi(Ax)\|_2 \in \left((1-\epsilon) \left(\sum_{i=1}^k \|x_i\|_2^2 \sigma_i^2 \right)^{1/2}, (1+\epsilon) \left(\sum_{i=1}^k \|x_i\|_2^2 \sigma_i^2 \right)^{1/2} \right) \right] \geq 1 - 2\exp(-\epsilon^2 m/100).$$

Proof. It is similar as Corollary 6.10.9. □

Fact 6.10.11 (see e.g. [AZLS19])**.** *Let $h, q \in \mathbb{R}^p$ be fixed vectors and $h \neq 0$, $W \in \mathbb{R}^{m \times p}$ be random matrix with i.i.d. entries $W_{i,j} \sim \mathcal{N}(0, \frac{2}{m})$, and vector $v \in \mathbb{R}^m$ defined as $v_i = \mathbf{1}_{\langle W_i, h+q \rangle \geq 0} \langle W_i, h \rangle$. Then,*

- *$|v_i|$ follows i.i.d. from the following distribution: with half probability $|v_i| = 0$, and with the other half probability $|v_i|$ follows from folded Gaussian distributions $|\mathcal{N}(0, \frac{2\|h\|^2}{m})|$.*
- *$\frac{m\|v\|^2}{2\|h\|^2}$ is in distribution identical to χ^2_ω (chi-square distribution of order ω) where ω follows from binomial distribution $\mathcal{B}(m, 1/2)$ (m trials each with success rate $1/2$).*

6.10.4 Gaussian Vector Percentile: Center

Fact 6.10.12. *Suppose $x \sim \mathcal{N}(0, \sigma^2)$ is a Gaussian random variable. For any $t \in (0, \sigma]$ we have*

$$\Pr[x \geq t] \in \left[\frac{1}{2}(1 - \frac{4}{5}\frac{t}{\sigma}), \frac{1}{2}(1 - \frac{2}{3}\frac{t}{\sigma})\right].$$

Similarly, if $x \sim \mathcal{N}(\mu, \sigma^2)$, for any $t \in (0, \sigma]$, we have

$$\Pr[|x| \geq t] \in \left[1 - \frac{4}{5}\frac{t}{\sigma}, 1 - \frac{2}{3}\frac{t}{\sigma}\right].$$

Lemma 6.10.13. *Let $x \sim \mathcal{N}(0, \sigma^2 I)$. For any $\alpha \in (0, 1/2)$, we have with probability at least $1 - \exp(-\alpha^2 m/100)$,*

- *there exists at least $\frac{1}{2}(1 - \alpha)$ fraction of i such that $x_i \geq 5\alpha\sigma/16$, and*
- *there exists at least $\frac{1}{2}(1 - \alpha)$ fraction of i such that $x_i \leq -5\alpha\sigma/16$.*

Proof. Let $c_1 = 4/5$. For each $i \in [m]$, we define random variable y_i as

$$y_i = \begin{cases} 1, & \text{if } x_i \geq \alpha\sigma/(4c_1); \\ 0, & \text{otherwise.} \end{cases}$$

Let $p = \Pr[y_i = 1]$. Using Fact 6.10.12 with $t = \alpha\sigma/(4c_1)$, we know that $p \geq \frac{1}{2}(1 - \alpha/4)$. Letting $Y = \sum_{i=1}^m y_i$, we have $\mu = \mathbb{E}[Y] = mp \geq \frac{1}{2}(1 - \alpha/4)m$. Using Chernoff bound, we have

$$\Pr[Y \leq (1 - \delta)\mu] \leq \exp(-\delta^2\mu/2).$$

Choosing $\delta = \frac{1}{4}\alpha$, we have

$$\Pr[Y \leq (1 - \alpha/4)\mu] \leq \exp(-\alpha^2\mu/32)$$

Further we have

$$\begin{aligned}
\Pr[Y \le \frac{1}{2}(1-\alpha)m] &\le \Pr[Y \le (1-\alpha/4)\frac{1}{2}(1-\alpha/4)m] && \text{by } (1-\alpha) \le (1-\alpha/4)^2 \\
&\le \Pr[Y \le (1-\alpha/4)\mu] && \text{by } \mu \ge \frac{1}{2}(1-\alpha/4)m \\
&= \Pr[Y \le (1-\delta)\mu] && \text{by } \delta = \alpha/4 \\
&\le \exp(-\alpha^2\mu/32) \\
&\le \exp(-\alpha^2\frac{1}{2}(1-\alpha/4)m/32) && \text{by } \mu \ge \frac{1}{2}(1-\alpha/4)m \\
&= \exp(-\alpha^2(1-\alpha/4)m/64)\ . && \square
\end{aligned}$$

We provide the definition of (α, σ)-good. Note that this definition will be used often in the later proof.

Definition 6.10.1 ((α, σ)-good)**.** Given $w \in \mathbb{R}^m$, we say w is (α, σ)-good if the following two conditions holding:

- there are at least $\frac{1}{2}(1-\alpha)$ fraction coordinates satisfy that $w_i \ge \alpha\sigma$; and
- there are at least $\frac{1}{2}(1-\alpha)$ fraction coordinates satisfy that $w_i \le -\alpha\sigma$.

Lemma 6.10.13 gives the following immediate corollary:

Corollary 6.10.14 (random Gaussian is $(\alpha, \sigma/4)$-good)**.** *Let $x \sim \mathcal{N}(0, \sigma^2 I)$. For any $\alpha \in (0, 1/2)$, we have with probability at least $1 - \exp(-\alpha^2 m/100)$ that x is $(\alpha, \sigma/4)$-good.*

Corollary 6.10.15. *Let $x_1, x_2, \cdots, x_k$ be k fixed vectors where $x_i \in \mathbb{R}^{n_i}$, and $A_1, A_2, \cdots, A_k$ be k independent random Gaussian matrices where each entry of $A_i \in \mathbb{R}^{m \times n_i}$ is i.i.d. sampled from $\mathcal{N}(0, \sigma_i^2)$ for each $i \in [k]$. Denote by $x = (x_1, \ldots, x_k) \in \mathbb{R}^n$ for $n = \sum_{i=1}^k n_i$, $A =$*

$[A_1, A_2, \cdots, A_k] \in \mathbb{R}^{m\times n}$, *and* $\sigma = \left(\sum_{i=1}^k \sigma_i^2 \|x_i\|_2^2\right)^{1/2}$. *For any fixed parameter* $\alpha \in (0, 1/2)$, *we have*

$$Ax = A_1x_1 + \cdots A_kx_k \text{ is } (\alpha, \sigma/4)\text{-good with probability at least } 1 - \exp(-\alpha^2 m/100).$$

Proof. It is clear that Ax follows from a Gaussian distribution $\mathcal{N}\big(0, \big(\sum_{i=1}^k \sigma_i^2 \|x_i\|_2^2\big) I\big)$, so we can directly apply Corollary 6.10.14. □

6.10.5 Gaussian Vector Percentile: Tail

Lemma 6.10.16. *Suppose $W \in \mathbb{R}^{m\times n}$ is a random matrix with entries drawn i.i.d. from $\mathcal{N}\big(0, \frac{2}{m}\big)$. Given $d \le s \le m$, with probability at least $1 - e^{-\Omega(s\log^2 m)}$, for all $x \in \mathbb{R}^n$, letting $y = Wx$, we can write $y = y_1 + y_2$ with*

$$\|y_1\| \le \frac{\sqrt{s}\log^2 m}{\sqrt{m}} \cdot \|x\| \quad \textit{and} \quad \|y_2\|_\infty \le \frac{\log m}{\sqrt{m}} \cdot \|x\| \ .$$

Proof of Lemma 6.10.16. Without loss of generality we only prove the result for $\|x\| = 1$.

Fixing any such x and letting $\beta = \frac{\log m}{2\sqrt{m}}$, we have $y_i \sim \mathcal{N}\big(0, \frac{2}{m}\big)$ so for every $p \ge 1$, by Gaussian tail bound

$$\Pr[|y_i| \ge \beta p] \le e^{-\Omega(\beta^2 p^2 m)} \ .$$

Since $\beta^2 p^2 m \ge \beta^2 m \gg \Omega(\log m)$, we know that if $|y_i| \ge \beta p$ occurs for q/p^2 indices i out of $[m]$, this cannot happen with probability more than

$$\binom{m}{q/p^2} \times \left(e^{-\Omega(\beta^2 p^2 m)}\right)^{q/p^2} \le e^{\frac{q}{p^2}\left(O(\log m) - \Omega(\beta^2 p^2 m)\right)} \le e^{-\Omega(\beta^2 q m)} \ .$$

In other words,

$$\Pr\left[\left|\{i \in [m] \colon |y_i| \ge \beta p\}\right| > q/p^2\right] \le e^{-\Omega(\beta^2 q m)} \ .$$

Finally, by applying union bound over $p = 1, 2, 4, 8, 16, \ldots$ we have with probability $\ge 1 - e^{-\Omega(\beta^2 q m)} \cdot \log q$,

$$\sum_{i\colon |y_i| \ge \beta} y_i^2 \le \sum_{k=0}^{\lceil \log q \rceil} (2^{k+1}\beta)^2 \left|\left\{i \in [m] \colon |y_i| \ge 2^k \beta\right\}\right| \le \sum_{k=0}^{\lceil \log q \rceil} (2^{k+1}\beta)^2 \cdot \frac{q}{2^{2k}} \le 4q\beta^2 \log q \ . \tag{6.23}$$

In other words, vector y can be written as $y = y_1 + y_2$ where $\|y_2\|_\infty \le \beta$ and $\|y_1\|^2 \le 4q\beta^2 \log q$.

At this point, we can choose $q = \frac{s \log^2 m}{m\beta^2} = 4s$ so the above event happens with probability at least $1 - e^{-\Omega(\beta^2 qm)} \geq 1 - e^{-\Omega(s \log^2 m)}$. Finally, applying standard ϵ-net argument over all unit vectors $x \in \mathbb{R}^n$, we have for each such $x \in \mathbb{R}^n$, we can decompose $y = Wx$ into $y = y_1 + y_2$ where

$$\|y_2\|_\infty \leq 2\beta = \frac{\log m}{\sqrt{m}} \text{ and } \|y_1\|^2 \leq \frac{8s \log^3 m}{m} \ . \qquad \square$$

6.10.6 McDiarmid's Inequality and An Extension

We state the standard McDiarmid's inequality,

Lemma 6.10.17 (McDiarmid's inequality)**.** *Consider independent random variables $x_1, \cdots, x_n \in \mathcal{X}$ and a mapping $f : \mathcal{X}^n \to \mathbb{R}$. If for all $i \in [n]$ and for all $y_1, \cdots, y_n, y_i' \in \mathcal{X}$, the function f satisfies*

$$|f(y_1, \cdots, y_{i-1}, y_i, y_{i+1}, \cdots, y_n) - f(y_1, \cdots, y_{i-1}, y_i', y_{i+1}, \cdots, y_n)| \leq c_i.$$

Then

$$\Pr[f(x_1, \cdots, x_n) - \mathbb{E}\, f \geq t] \geq \exp(\frac{-2t^2}{\sum_{i=1}^n c_i^2}),$$
$$\Pr[f(x_1, \cdots, x_n) - \mathbb{E}\, f \leq -t] \geq \exp(\frac{2t^2}{\sum_{i=1}^n c_i^2}).$$

We prove a more general version of McDiarmid's inequality,

Lemma 6.10.18 (McDiarmid extension)**.** *Let $w_1, \ldots, w_N$ be independent random variables and*

$f \colon (w_1, \ldots, w_N) \mapsto [0, 1]$. Suppose it satisfies:

- $\mathbb{E}_{w_1,\ldots,w_N}[f(w_1, \ldots, w_N)] \geq \mu$, *and*
- *With probability at least $1 - p$ over $w_1, \ldots, w_N$, it satisfies*

$$\forall k \in [N], \forall w_k'' \colon\ |f(w_{-k}, w_k) - f(w_{-k}, w_k'')| \leq c \tag{6.24}$$

Then, $\Pr[f(w_1, \ldots, w_N) \geq \mu/2] \geq 1 - N^2\sqrt{p} - e^{\Omega(\frac{-\mu^2}{N(c^2+p)})}$.

Proof of Lemma 6.10.18. For each $t \in [N]$, we have with probability at least $1 - \sqrt{p}$ over $w_1, \ldots, w_t$, it satisfies

$$\Pr_{w_{t+1},\ldots,w_N}[\forall k \in [N], \forall w_k'' \colon |f(w_{-k}, w_k) - f(w_{-k}, w_k'')| \le c] \ge 1 - \sqrt{p} \ .$$

Define those $(w_1, \ldots, w_t)$ satisfying the above event to be $K_{\le t}$.

Define random variable X_t (which depends only on $w_1, \ldots, w_t$) as

$$X_t := \mathop{\mathbb{E}}_{w_{>t}}[f(\vec{w}) \mid w_{\le t}] \mathbb{1}_{(w_{\le 1},\ldots,w_{\le t}) \in K_{\le 1} \times \cdots \times K_{\le t}} + N(1 - \mathbb{1}_{(w_{\le 1},\ldots,w_{\le t}) \in K_{\le 1} \times \cdots \times K_{\le t}})$$

For every t and fixed $w_1, \ldots, w_{t-1}$.

- If $(w_{\le 1}, \ldots, w_{<t}) \notin K_{\le 1} \times \cdots \times K_{<t}$, then $X_t = X_{t-1} = N$.
- If $(w_{\le 1}, \ldots, w_{<t}) \in K_{\le 1} \times \cdots \times K_{<t}$,
  - If $w_{\le t} \notin K_{\le t}$, then $X_t - X_{t-1} = N - \cdots \ge 0$.
  - If $w_{\le t} \in K_{\le t}$, then

    $$X_t - X_{t-1} = \mathop{\mathbb{E}}_{w_{>t}}[f(w_{<t}, w_t, w_{>t}) \mid w_{\le t}] - \mathop{\mathbb{E}}_{w_{\ge t}}[f(w_{<t}, w_t, w_{>t}) \mid w_{<t}]$$

    Recall from our assumption that, with probability at least $1 - \sqrt{p}$ over w_t and $w_{>t}$, it satisfies

    $$\forall w_t'' \colon |f(w_{<t}, w_t'', w_{>t}) - f(w_{<t}, w_t, w_{>t})| \le c$$

    Taking expectation over w_t and $w_{>t}$, we have

    $$\forall w_t'' \colon \mathop{\mathbb{E}}_{w_{>t}}[f(w_{<t}, w_t'', w_{>t})] - \mathop{\mathbb{E}}_{w_{\ge t}}[f(w_{<t}, w_t, w_{>t})] \ge -(c + \sqrt{p})$$

    This precisely means $X_t - X_{t-1} \ge c + \sqrt{p}$.

In sum, we have just shown that $X_t - X_{t-1} \geq -(c + \sqrt{p})$ always holds. By applying martingale concentration (with one-sided bound),

$$\Pr[X_N - X_0 \leq -t] \leq \exp\left(\frac{-t^2}{N(c+\sqrt{p})^2}\right)$$

Notice that $X_0 = \mu$ so if we choose $t = \mu/2$, we have

$$\Pr[X_N \geq \mu/2] \geq 1 - \exp\left(\frac{-\mu^2}{N(c+\sqrt{p})^2}\right)$$

Recalling

$$X_N := f(\vec{w})\mathbb{1}_{(w_{\leq 1},\ldots,w_{\leq t}) \in K_{\leq 1} \times \cdots \times K_{\leq t}} + N(1 - \mathbb{1}_{(w_{\leq 1},\ldots,w_{\leq t}) \in K_{\leq 1} \times \cdots \times K_{\leq t}})$$

and we have $X_N = f(w_1, \ldots, w_N)$ with probability at least $1 - N\sqrt{p}$ (and $X_N = N$ with the remaining probabilities). Together, we have the desired theorem.

□

6.11 Basic Properties at Random Initialization

Recall that the recursive update equation of RNN can be described as follows

$$\begin{aligned} h_{i,0} &= 0 && \forall i \in [n] \\ h_{i,\ell} &= \phi(W \cdot h_{i,\ell-1} + Ax_{i,\ell}) && \forall i \in [n], \forall \ell \in [L] \\ y_{i,\ell} &= B \cdot h_{i,\ell} && \forall i \in [n], \forall \ell \in [L] \end{aligned}$$

Throughout this section, we assume that matrices $W \in \mathbb{R}^{m \times m}$, $A \in \mathbb{R}^{m \times d_x}$, $B \in \mathbb{R}^{d \times m}$ are at their random initialization position: each entry of W and A is sampled i.i.d. from $\mathcal{N}(0, \frac{2}{m})$ and each entry of B is sampled i.i.d. from $\mathcal{N}(0, \frac{1}{d})$. We recall

Definition 6.11.1. For each $i \in [n]$ and $\ell \in [L]$, let $D_{i,\ell} \in \mathbb{R}^{m \times m}$ be the diagonal matrix where

$$(D_{i,\ell})_{k,k} = \mathbf{1}_{(W \cdot h_{i,\ell-1} + Ax_{i,\ell})_k \geq 0} = \mathbf{1}_{(g_{i,\ell})_k \geq 0} \ .$$

As a result, we can write $h_{i,\ell} = D_{i,\ell} W h_{i,\ell-1}$.

We introduce two notations that shall repeatedly appear in our proofs.

Definition 6.11.2 (U_ℓ)**.** Let $U_\ell \in \mathbb{R}^{m \times n\ell}$ denote the column orthonormal matrix using Gram-Schmidt

$$U_\ell \stackrel{\text{def}}{=} \mathsf{GS}\left(h_{1,1}, \ldots, h_{n,1}, h_{1,2}, \ldots, h_{n,2}, \ldots, h_{1,\ell}, \ldots, h_{n,\ell}\right)$$

Definition 6.11.3 ($v_{i,\ell}$)**.** For each $i \in [n]$, $\ell \in [L]$, we define vector $v_{i,\ell} \in \mathbb{R}^m$ as

$$v_{i,\ell} = \frac{(I - U_{\ell-1}U_{\ell-1}^\top)h_{i,\ell}}{\|(I - U_{\ell-1}U_{\ell-1}^\top)h_{i,\ell}\|_2}$$

Roadmap

- Section 6.11.1 proves that the forward propagation $\|h_{i,\ell}\|_2$ neither vanishes nor explodes.
- Section 6.11.2 gives a lower bound on the projected forward propagation $\|(I-U_\ell U_\ell^\top)h_{i,\ell+1}\|_2$.
- Section 6.11.3 proves that for two data points $i, j \in [n]$, the projected forward propagation $(I - U_\ell U_\ell^\top)h_{i,\ell+1}$ and $(I - U_\ell U_\ell^\top)h_{j,\ell+1}$ are separable from either other.
- Section 6.11.4 and Section 6.11.5 prove that the consecutive intermediate layers, in terms of spectral norm, do not explode (for full and sparse vectors respectively).
- Section 6.11.6 proves that the backward propagation does not explode.

6.11.1 Forward Propagation

Our first result of this section is on showing upper and lower bounds on the forward propagation.

Lemma 6.11.1 (c.f. (6.2))**.** *With probability at least* $1 - \exp(-\Omega(m/L^2))$ *over the random initialization* W, A *(see Definition 6.2.2), it satisfies*

$$\forall i \in [n], \quad \forall \ell \in \{0, 1, \ldots, L-1\} \quad : \quad (1 - 1/(4L))^\ell \leq \|h_{i,\ell+1}\|_2 \leq 2\ell + 4.$$
$$\|g_{i,\ell+1}\|_2 \leq 4\ell + 8.$$

We prove Lemma 6.11.1 by induction on $\ell \in [L]$. We only prove the $h_{i,\ell+1}$ part and the $g_{i,\ell+1}$ part is completely analogous.

For the base case $\ell = 0$, we have $\|x_{i,1}\|_2 = 1$ and $h_{i,1} = \phi(Ax_{i,1})$. Using Corollary 6.10.10 we have $1-1/4L \leq \|h_{i,1}\|_2 \leq 1+1/4L$ with probability at least $1-\exp(-\Omega(-m/L^2))$. We proceed the proof for the case of $\ell \geq 1$, assuming that Lemma 6.11.1 already holds for $0, 1, \ldots, \ell - 1$. We first show

Claim 6.11.2. *With probability at least* $1 - \exp(-\Omega(m/L^2))$ *over* W *and* A*,*

$$\forall i \in [n] \colon \|h_{i,\ell+1}\|_2 \leq (1 + 1/L)(\|h_{i,\ell}\|_2 + 1)$$

Note if Claim 6.11.2 holds for all $\ell = 0, 1, \ldots, \ell - 1$, then we must have $\|h_{i,\ell}\|_2 \leq (1 + 1/L)^\ell \|h_{i,0}\|_2 + \sum_{i=1}^{\ell} (1 + 1/L)^i = (1 + 1/L)^\ell + \sum_{i=1}^{\ell} (1 + 1/L)^i \leq 2\ell + 4$.

Proof of Claim 6.11.2. Recall the definition of $h_{i,\ell} \in \mathbb{R}^m$, we have

$$\|h_{i,\ell+1}\|_2 = \|\phi(W \cdot h_{i,\ell} + Ax_{i,\ell+1})\|_2$$

We can rewrite vector $Wh_{i,\ell} \in \mathbb{R}^m$ as follows

$$\begin{aligned} Wh_{i,\ell} &= WU_{\ell-1}U_{\ell-1}^\top h_{i,\ell} + W(I - U_{\ell-1}U_{\ell-1}^\top)h_{i,\ell} \\ &= WU_{\ell-1}U_{\ell-1}^\top h_{i,\ell} + W\frac{(I - U_{\ell-1}U_{\ell-1}^\top)h_{i,\ell}}{\|(I - U_{\ell-1}U_{\ell-1}^\top)h_{i,\ell}\|_2} \cdot \|(I - U_{\ell-1}U_{\ell-1}^\top)h_{i,\ell}\|_2 \\ &= WU_{\ell-1}U_{\ell-1}^\top h_{i,\ell} + Wv_{i,\ell} \cdot \|(I - U_{\ell-1}U_{\ell-1}^\top)h_{i,\ell}\|, \end{aligned}$$

where the last step follows by definition of $v_{i,\ell}$ (See Definition 6.11.3). Define $z_1 \in \mathbb{R}^{n(\ell-1)}$, $z_2 \in \mathbb{R}$, $z_3 \in \mathbb{R}^d$ as follows

$$z_1 = U_{\ell-1}^\top h_{i,\ell}, \quad z_2 = \|(I - U_{\ell-1}U_{\ell-1}^\top)h_{i,\ell}\|_2, \quad z_3 = x_{i,\ell+1}. \tag{6.25}$$

Then

$$\begin{aligned} \|z_1\|_2^2 + z_2^2 + \|z_3\|_2^2 &= \|U_{\ell-1}^\top h_{i,\ell}\|_2^2 + \|(I - U_{\ell-1}U_{\ell-1}^\top)h_{i,\ell}\|_2^2 + \|x_{i,\ell+1}\|_2^2 \\ &= \|h_{i,\ell}\|_2^2 + \|x_{i,\ell+1}\|_2^2 \leq \|h_{i,\ell}\|_2^2 + 1 \ . \end{aligned} \tag{6.26}$$

We can thus rewrite

$$\begin{aligned} Wh_{i,\ell} + Ax_{i,\ell+1} &= WU_{\ell-1}z_1 + Wv_{i,\ell}z_2 + Az_3 \\ &= \begin{bmatrix} M_1 & M_2 & M_3 \end{bmatrix} \cdot \begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix} \\ &= M \cdot z, \end{aligned}$$

where the last step follows by defining $M \in \mathbb{R}^{m \times (n(\ell-1)+1+d)}$ as follows,

$$M = \begin{bmatrix} M_1 & M_2 & M_3 \end{bmatrix} = \begin{bmatrix} WU_{\ell-1} & Wv_{i,\ell} & A \end{bmatrix} \tag{6.27}$$

We stress here that the entries of M_1, M_2, M_3 are i.i.d. from $\mathcal{N}(0, \frac{2}{m})$.[13] We have $\|h_{i,\ell+1}\| = \|\phi(Wh_{i,\ell} + Ax_{i,\ell+1})\| = \|\phi(M \cdot z)\|$.

Next, applying Corollary 6.10.10, we know that if z_1, z_2, z_3 are fixed (instead of defined as in (6.25)), then, letting Choosing $\epsilon = 1/2L$, we have

$$\Pr_M \left[\|h_{i,\ell+1}\| \leq (1+\epsilon) \cdot \sqrt{\|z_1\|_2^2 + z_2^2 + \|z_3\|_2^2} \right] \geq 1 - \exp(-\Omega(\epsilon^2 m)) \ .$$

To move from fixed choices of z_1 and z_2 to *all* choices of z_1 and z_2, we perform a standard ϵ-net argument. Since the dimension of z_1 and z_2 are respectively $n(\ell-1)$ and 1, the size of ϵ-net for z_1 and z_2 is at most $e^{O(nL \log L)}$. Thus, with probability at least $1 - e^{O(nL \log L + \log n)} \cdot \exp(-\Omega(\epsilon^2 m)) \geq 1 - \exp(-\Omega(\epsilon^2 m))$ we have: for all $z_1 \in \mathbb{R}^{n(\ell-1)}$ and $z_2 \in \mathbb{R}$ satisfying $\|z_1\| \leq 2L + 4$ and $0 \leq z_2 \leq 2L + 4$, and fixed $z_3 = x_{i,\ell+1}$,

$$\|h_{i,\ell+1}\| \leq (1+2\epsilon) \cdot \sqrt{\|z_1\|_2^2 + z_2^2 + \|z_3\|_2^2}.$$

In particular, since we have "for all" quantifies on z_1 and z_2 above, we can substitute the choice of z_1 and z_2 in (6.25) (which may depend on the randomness of W and A). This, together with (6.26), gives

$$\Pr \left[\|h_{\ell+1}\|_2 \leq (1+\epsilon) \cdot \sqrt{\|h_{i,\ell}\|_2^2 + 1} \right] \geq 1 - \exp(-\Omega(\epsilon^2 m)) \ . \qquad \square$$

Similarly, we can prove a lower bound

[13] Indeed, after Gram-Schmidt we can write $U_{\ell-1} = [\widehat{h}_1, \ldots, \widehat{h}_{n(\ell-1)}]$, where each $\widehat{h}_j$ only depends on the randomness of A and $W[\widehat{h}_1, \ldots, \widehat{h}_{j-1}]$. In other words, conditioning on any choice of A and $W[\widehat{h}_1, \ldots, \widehat{h}_{j-1}]$, we still have $W\widehat{h}_j$ is an independent Gaussian vector from $\mathcal{N}(0, \frac{2}{m}I)$. Similarly, $v_{i,\ell}$ may depend on the randomness of A and $WU_{\ell-1}$, but conditioning on any choice of A and $WU_{\ell-1}$, we still have $Wv_{i,\ell}$ follows from $\mathcal{N}(0, \frac{2}{m}I)$. This proves that the entries of M_1, M_2, M_3 are independent.

Claim 6.11.3. *With probability at least* $1 - \exp(-\Omega(m/L^2))$ *over* W *and* A,

$$\|h_{i,\ell+1}\|_2 \geq (1 - \frac{1}{4L})\|h_{i,\ell}\|_2.$$

Proof. We can define z_1, z_2, z_3 in the same way as (6.25). This time, we show a lower bound

$$\|z_1\|_2^2 + z_2^2 + \|z_3\|_2^2 \geq \|z_1\|_2^2 + z_2^2 = \|h_{i,\ell}\|_2^2. \tag{6.28}$$

Applying Corollary 6.10.10, we know if z_1, z_2, z_3 are fixed (instead of defined as in (6.25)), then choosing $\epsilon = 1/8L$,

$$\Pr\left[\|h_{i,\ell+1}\| \geq (1-\epsilon) \cdot \sqrt{\|z_1\|_2^2 + z_2^2 + \|z_3\|_2^2}\right] \geq 1 - \exp(-\Omega(\epsilon^2 m)) \ ..$$

Again, after applying ϵ-net, we know with probability at least $1 - e^{O(nL\log L + \log n)} \cdot \exp(-\Omega(\epsilon^2 m)) \geq 1 - \exp(-\Omega(\epsilon^2 m))$, for all $z_1 \in \mathbb{R}^{n(\ell-1)}$ and $z_2 \in \mathbb{R}$ satisfying $\|z_1\| \leq 2L + 4$ and $0 \leq z_2 \leq 2L + 4$, and fixed $z_3 = x_{i,\ell+1}$,

$$\|h_{i,\ell+1}\|_2 \geq (1-\epsilon) \cdot \sqrt{\|z_1\|_2^2 + z_2^2 + \|z_3\|_2^2}.$$

Substituting the choice of z_1, z_2, z_3 in (6.25), and the lower bound (6.28), we have

$$\Pr\left[\|h_{i,\ell+1}\| \geq (1-\epsilon) \cdot \|h_{i,\ell}\|_2\right] \geq 1 - \exp(-\Omega(\epsilon^2 m)).$$

Choosing $\epsilon = 1/(4L)$ gives the desired statement. □

Finally, recursively applying Claim 6.11.2 and Claim 6.11.3 for all $\ell = 0, 1, \ldots, L-1$, we have with probability at least $1 - L^2 \exp(-\Omega(m/L^2)) \geq 1 - \exp(-\Omega(m/L^2))$, we have

$$\|h_{i,\ell}\|_2 \geq (1 - \frac{1}{4L})^\ell, \qquad \|h_{i,\ell}\|_2 \leq 2\ell + 4 \ .$$

This finishes the proof of Lemma 6.11.1. ■

6.11.2 Forward Correlation

This subsection proves the following lemma which, as discussed in Section 6.5, bounds how much "fresh new randomness" is left after propagating to layer ℓ.

Lemma 6.11.4 (c.f. (6.3))**.** *With probability at least $1 - e^{-\Omega(\sqrt{m})}$ over the random initialization W, A in Definition 6.2.2, letting U_ℓ be defined in Definition 6.11.2, we have*

$$\forall i \in [n], \forall \ell \in \{0, 1, \cdots, L-1\} \quad : \quad \|(I - U_\ell U_\ell^\top) h_{i,\ell+1}\|_2 \geq \frac{1}{2 \cdot 10^6 L^2 \log^3 m} \ .$$

To prove Lemma 6.11.4, we inductively (with the increasing order of ℓ) show for each $i \in [n]$, for each $\ell \in \{0, 1, \cdots, L-1\}$, we have

$$\|(I - U_\ell U_\ell^\top) h_{i,\ell+1}\|_2 \geq \xi_\ell \stackrel{\text{def}}{=} \frac{1}{10^6 L^2 \log^3 m}(1 - 2\alpha - \frac{1}{4L})^\ell \quad \text{where} \quad \alpha \stackrel{\text{def}}{=} \frac{1}{2 \cdot 10^4 L^2 \log^2 m} \ . \tag{6.29}$$

We first show (6.29) in the base case $\ell = 0$. Since U_0 is an empty matrix, we have $(I - U_0 U_0^\top) h_{i,1} = h_{i,1}$ so according to Lemma 6.11.1 we have $\|h_{i,1}\|_2 \geq (1 - 1/4L)$.

The remainder of the proof assumes (6.29) already holds for $\ell - 1$. We can write

$$h_{i,\ell+1} = \phi(W h_{i,\ell} + A x_{i,\ell+1}) = \phi(M_1 z_1 + M_2 z_2 + M_3 z_3).$$

where $M_1 \in \mathbb{R}^{m \times n(\ell-1)}$, $M_2 \in \mathbb{R}^{m \times 1}$, $M_3 \in \mathbb{R}^{m \times d}$ and $z_1 \in \mathbb{R}^{n(\ell-1)}$, $z_2 \in \mathbb{R}$, $z_3 \in \mathbb{R}^d$ are defined as

$$\begin{aligned} &M_1 = W U_{\ell-1} && M_2 = W v_{i,\ell} && M_3 = A \\ &z_1 = U_{\ell-1}^\top h_{i,\ell} && z_2 = \|(I - U_{\ell-1} U_{\ell-1}^\top) h_{i,\ell}\|_2 && z_3 = x_{i,\ell+1} \ . \end{aligned} \tag{6.30}$$

in the same way as (6.25) and (6.27) as in the proof of Lemma 6.11.1. We again have the entries of M_1, M_2, M_3 are i.i.d. from $\mathcal{N}(0, \frac{2}{m})$ (recall Footnote 13). For the quantity $M_2 z_2$, we can further decompose it as follows

$$Mz_2 = \nu \cdot (z_2 - c_5\alpha)_+ + \nu' z_2' \tag{6.31}$$

where $c_5 = \frac{1}{16 \log m}$ is some fixed parameter, ν and ν' denote two vectors that are independently generated from $\mathcal{N}(0, \frac{2I}{m})$, and

$$(z_2 - c_5\alpha)_+ = \begin{cases} 0, & \text{if } z_2 < c_5\alpha; \\ \sqrt{z_2^2 - c_5^2\alpha^2}, & \text{if } z_2 \geq c_5\alpha. \end{cases} \qquad z_2' = \begin{cases} z_2, & \text{if } z_2 < c_5\alpha; \\ c_5\alpha, & \text{if } z_2 \geq c_5\alpha. \end{cases}$$

It is clear that the two sides of (6.31) are identical in distribution (because $M_2 \sim \mathcal{N}(0, \frac{2I}{m})$ and $(z_2 - c_5\alpha)_+^2 + (z_2')^2 = z_2^2$).

Next, suppose z_1 and z_2 are fixed (instead of depending on the randomness of W and A) and satisfies[14]

$$\frac{1}{\sqrt{2}} \leq \|z_1\| \leq 2L + 6 \text{ and } |z_2| \leq 2L + 6. \tag{6.32}$$

We can apply Corollary 6.10.15 to obtain the following statement: with probability at least $1 - \exp(-\Omega(\alpha^2 m))$,

$$w \stackrel{\text{def}}{=} M_1 z_1 + \nu(z_2 - c_5\alpha)_+ + M_3 z_3$$

is $(\alpha, \sigma/4)$-good where $\sigma = \left(\frac{2}{m}\|z_1\|_2^2 + \frac{2}{m}((z_2 - c_5\alpha)_+)^2 + \frac{2}{m}\|z_3\|_2^2\right)^{1/2}$. Using $\|z_1\|_2^2 \geq 1/2$, we can lower bound σ^2 as

$$\sigma^2 \geq \frac{2}{m}(\|z_1\|_2^2 + z_2^2 - c_5^2\alpha^2) \geq \frac{2}{m}(\frac{1}{2} - c_5^2\alpha^2) \geq \frac{1}{2m} \ .$$

[14]Note that if z_1 and z_2 are random, then they satisfy such constraints by Lemma 6.11.1.

In other words, w is $(\alpha, \frac{1}{4\sqrt{2}\sqrt{m}})$-good. Next, applying standard ϵ-net argument, we have with probability at least $1 - \exp(-\Omega(\alpha^2 m))$, for all vectors $z_1 \in \mathbb{R}^{n(\ell-1)}$ and $z_2 \in \mathbb{R}$ satisfying (6.32), it satisfies w is $(\alpha, \frac{1}{8\sqrt{m}})$-good. This allows us to plug in the random choice of z_1 and z_2 in (6.30).

We next apply Lemma 6.11.5 with

$$w = M_1 z_1 + \nu(z_2 - c_5\alpha)_+ + M_3 z_3, \quad r = z_2', \quad v = \nu, \quad U = U_\ell.$$

(We can do so because the randomness of v is independent of the randomness of U_ℓ and w.) Lemma 6.11.5 tells us that, with probability at least $1 - \exp(-\Omega(\sqrt{m}))$ over the randomness of v,

$$\|(I - U_\ell U_\ell^\top) \cdot \phi(w + rv)\| \geq r(1 - 2\alpha) - \frac{\alpha^{1.5}}{4} \ .$$

By induction hypothesis, we know

$$r = z_2' \geq \min\{z_2, c_5\alpha\} \geq \min\{\xi_{\ell-1}, c_5\alpha\} \geq \xi_{\ell-1} \ ,$$

where the last step follows by $\xi_{\ell-1} \leq c_5\alpha$. Thus, we have

$$\|(I - U_\ell U_\ell^\top) \cdot \phi(w + rv)\| \geq r(1 - 2\alpha) - \frac{\alpha^{1.5}}{4} \geq \xi_{\ell-1}(1 - 2\alpha) - \frac{\alpha^{1.5}}{4} \geq \xi_{\ell-1}(1 - 2\alpha - \frac{1}{4L}) = \xi_\ell \ ,$$

where the last inequality uses $\alpha^{1.5} \leq \frac{\xi_{\ell-1}}{L}$.

■

6.11.2.1 Tools

Lemma 6.11.5. *Suppose $\alpha \in [\frac{1}{100L^4}, 1/2)$, $m \geq 4\widetilde{d}/\alpha$ and $r \in (0, \frac{\alpha}{16 \log m}]$. Suppose w is a fixed vector that is $(\alpha, \frac{1}{8\sqrt{m}})$-good (see Definition 6.10.1), and $U \in \mathbb{R}^{m \times \widetilde{d}}$ is a fixed column*

orthonormal matrix. Then, if $v \sim \mathcal{N}(0, \frac{2}{m}I)$*, with probability at least* $1 - \exp(-\Omega(\sqrt{m}))$*, it satisfies*

$$\|(I - UU^\top) \cdot \phi(w + rv)\| \geq r(1 - 2\alpha) - \frac{\alpha^{1.5}}{4} \ .$$

Proof. We split $w \in \mathbb{R}^m$ into three pieces $w = (w_1, w_2, w_3) \in \mathbb{R}^m$ with disjoint support such that:

- w_1 corresponds to the coordinates that are $\geq \alpha/8\sqrt{m}$,
- w_2 is the remaining, which corresponds to the coordinates that are within $(-\alpha/8\sqrt{m}, \alpha/8\sqrt{m})$.
- w_3 corresponds to the coordinates that are $\leq -\alpha/8\sqrt{m}$, and

We write $v = (v_1, v_2, v_3)$ according to the same partition. We consider the following three cases.

- For each index k in the first block, we have $\big(\phi(w_1 + rv_1)\big)_k \neq \big(w_1 + rv_1\big)_k$ only if $(rv_1)_k \leq -\alpha/8\sqrt{m}$. However, if this happens, we have

  $$0 \leq \big(\phi(w_1 + rv_1)\big)_k - \big(w_1 + rv_1\big)_k \leq \max\big\{\big(rv_1\big)_k - \alpha/8\sqrt{m}, 0\big\} \ .$$

  Applying Lemma 6.10.5, we know with probability at least $1 - e^{-\Omega(\sqrt{m})}$,

  $$\delta_1 \stackrel{\text{def}}{=} \phi(w_1 + rv_1) - \big(w_1 + rv_1\big) \quad \text{satisfies} \quad \|\delta_1\|^2 \leq 2\sqrt{m}\frac{2r^2}{m} \leq 4\alpha^2/\sqrt{m} \ .$$

- Similarly, for each index k in the third block, we have $\big(\phi(w_1 + rv_1)\big)_k \neq 0$ only if $(rv_1)_k \geq \alpha/8\sqrt{m}$. Therefore, we can similarly derive that

  $$\delta_3 \stackrel{\text{def}}{=} \phi(w_3 + rv_3) \quad \text{satisfies} \quad \|\delta_3\|^2 \leq 4\alpha^2/\sqrt{m} \ .$$

- For the second block, we claim that

$$\delta_2 \stackrel{\text{def}}{=} \phi(w_2 + rv_2) \quad \text{satisfies} \quad \|\delta_2\|_2 \leq \alpha^{3/2}/4 \ . \tag{6.33}$$

To prove (6.33), we use triangle inequality,

$$\|\phi(w_2 + rv_2)\|_2 \leq \|\phi(w_2)\|_2 + \|\phi(rv_2)\|_2$$

Since $\|\phi(w_2)\|_\infty \leq \alpha/8\sqrt{m}$ and the size of support of $\phi(w_2)$ is at most αm, we have

$$\|\phi(w_2)\|_2 \leq ((\alpha/8\sqrt{m})^2 \alpha m)^{1/2} \leq \frac{\alpha^{3/2}}{8}.$$

Since $v \sim \mathcal{N}(0, \frac{2}{m}I)$, and since the size of support of v_2 is at most αm, we have $\|v_2\| \leq 2\sqrt{\alpha}$ with probability at least $1 - e^{-\Omega(\alpha m)}$ (due to chi-square distribution concentration). Thus

$$\|\phi(rv_2)\|_2 = r \cdot \|\phi(v_2)\|_2 \leq \frac{\alpha}{16} \cdot 2\sqrt{\alpha} = \frac{\alpha^{3/2}}{8} \ .$$

Together, by triangle inequality we have $\|\phi(w_2 + rv_2)\| \leq \frac{\alpha^{3/2}}{4}$. This finishes the proof of (6.33).

Denoting by $\delta = (\delta_1, \delta_2, \delta_3)$, we have

$$\phi(w + rv) = w_1 + rv_1 + \delta.$$

Taking the norm on both sides,

$$\begin{aligned}
\|(I-UU^\top)\phi(w+rv)\| &= \|(I-UU^\top)(w_1+rv_1+\delta)\| \\
&\geq \|(I-UU^\top)(w_1+rv_1)\| - \|(I-UU^\top)\delta\| \\
&\qquad\qquad \text{(by triangle inequality)} \\
&\geq \|(I-UU^\top)(w_1+rv_1)\| - \|\delta\| \qquad (\text{by } \|(I-UU^\top)\delta\| \leq \|\delta\|) \\
&\geq \|(I-UU^\top)(w_1+rv_1)\| - \frac{\alpha^{3/2}}{4} - \frac{4\alpha}{m^{1/4}} \\
&\geq \|(I-UU^\top)(w_1+rv_1)\| - \frac{\alpha^{3/2}}{2} \\
&\geq \|(I-UU^\top)w_1+rv_1\| - r\|U^\top v_1\| - \frac{\alpha^{3/2}}{2} \ .
\end{aligned}$$

Now, since $U \in \mathbb{R}^{m\times\widetilde{d}}$, it is not hard to show that $\|U^\top v_1\|_2^2 \leq \frac{2\widetilde{d}}{m}$ with probability at least $1-\exp(-\Omega(m))$. Using $\frac{2\widetilde{d}}{m} \leq \frac{\alpha}{2}$ (owing to our assumption $m \geq 4\widetilde{d}/\alpha$), we have $r\|U^\top v_1\| \leq \frac{r\alpha}{2}$.

On the other hand, the random vector $z = (I-UU^\top)w_1+rv_1$ follows from distribution $\mathcal{N}(\mu, \frac{2r^2}{m})$ for some fixed vector $\mu = (I-UU^\top)w_1$ and has at least $\frac{m}{2}(1-\alpha)$ dimensions. By chi-square concentration, we have $\|z\| \geq r(1-3\alpha/2)$ with probability at least $1-e^{-\Omega(\alpha^2 m)}$. Putting these together, we have

$$\|(I-UU^\top)\phi(w+rv)\| \geq r(1-3\alpha/2) - \frac{r\alpha}{2} - \frac{\alpha^{3/2}}{2} = r(1-2\alpha) - \frac{\alpha^{3/2}}{2} \ .$$

□

6.11.3 Forward δ-Separateness

We first give the definition of δ-Separable,

Definition 6.11.4 (δ-separable vectors)**.** For any two vectors x, y, we say x and y are δ-separable if

$$\left\|(I - \frac{yy^\top}{\|y\|_2^2})x\right\| \geq \delta \quad \text{and} \quad \left\|(I - \frac{xx^\top}{\|x\|_2^2})y\right\| \geq \delta \ ..$$

We say a finite set X is δ-separable if for any two vectors $x, y \in X$, x and y are δ-separable.

The goal of this subsection is to prove the δ-separateness over all layers ℓ.

Lemma 6.11.6 (c.f. (6.4))**.** *Let $\{x_{i,1}\}_{i\in[n]}$ be δ-separable with $\delta \leq \frac{1}{10^6 L^2 \log^3 m}$. With probability at least $1 - e^{-\Omega(\sqrt{m})}$, for all $\ell = 0, 1, \ldots, L-1$, for all $i, j \in [n]$ with $i \neq j$, we have*

$$(I - U_\ell U_\ell^\top)h_{i,\ell+1} \text{ and } (I - U_\ell U_\ell^\top)h_{j,\ell+1} \text{ are } \frac{\delta}{2}\text{-separable.}$$

(Note that since we have assumed $\|x_{i,1} - x_{j,1}\| \geq \delta$ in Assumption 6.2.1 and assumed without loss of generality that $(x_{i,1})_{d_x} = \frac{1}{\sqrt{2}}$, it automatically satisfies that $\{x_{i,1}\}_{i\in[n]}$ is $O(\delta)$-separable.)

To prove Lemma 6.11.6, we inductively (with the increasing order of ℓ) show for each $i \in [n]$, for each $\ell \in \{0, 1, \cdots, L-1\}$, we have

$$(I - U_\ell U_\ell^\top)h_{i,\ell+1} \text{ and } (I - U_\ell U_\ell^\top)h_{j,\ell+1} \text{ are } \delta_\ell \text{ separable for } \delta_\ell \stackrel{\text{def}}{=} \delta(1 - 2\alpha - \frac{1}{4L})^\ell \quad (6.34)$$

where $\alpha \stackrel{\text{def}}{=} 16\delta \log m$.

We skip the base case and only prove (6.34) for $\ell \geq 1$ by assuming (6.34) already holds for $\ell - 1$.[15]

Claim 6.11.7. *For $i \neq j$, if $(I - U_{\ell-1}U_{\ell-1}^\top)h_{i,\ell}$ and $(I - U_{\ell-1}U_{\ell-1}^\top)h_{j,\ell}$ are $\delta_{\ell-1}$-separable, then letting $U = [U_\ell, \widehat{h}]$ where $\widehat{h} = \frac{(I-U_\ell U_\ell^\top)h_{j,\ell+1}}{\|(I-U_\ell U_\ell^\top)h_{j,\ell+1}\|_2}$, we have*

$$\|(I - UU^\top)h_{i,\ell+1}\| \geq \delta_\ell = \delta_{\ell-1}(1 - 3\alpha) \ .$$

holds with probability at least $1 - \exp(-\Omega(\sqrt{m}))$.

Since it is easy to verify that

$$\left\| \left(I - \widehat{h}\widehat{h}^\top\right)(I - U_\ell U_\ell^\top)h_{i,\ell+1} \right\| = \|(I - UU^\top)h_{i,\ell+1}\|_2 \ ,$$

Claim 6.11.7 immediately implies (6.34) for layer ℓ, and thus finishes the proof of Lemma 6.11.6. Therefore, we only need to prove Claim 6.11.7 below.

Proof of Claim 6.11.7. Let $x = (I - U_{\ell-1}U_{\ell-1}^\top)h_{j,\ell}$ and $y = (I - U_{\ell-1}U_{\ell-1}^\top)h_{i,\ell}$. If x and y are $\delta_{\ell-1}$-separable, we split y into two parts where y_1 is parallel to x and y_2 is orthogonal to x

$$y = y_1 + y_2, \quad y_1 = \frac{\langle x, y\rangle x}{\|x\|_2^2}, \quad y_2 = (I - xx^\top/\|x\|_2^2)y$$

This also implies that the randomness in y_1 is independent of the randomness in y_2.

It is easy to see that $\|y_1\|_2 = \frac{\langle x,y\rangle}{\|x\|_2}$, so we can rewrite Wy as follows

$$Wy = Wy_1 + Wy_2 = W\frac{\langle x, y\rangle x}{\|x\|_2^2} + Wy_2 = (\|y_1\|_2/\|x\|_2)Wx + Wy_2.$$

[15] The proof of the base case is a replication of Claim 6.11.7 and only simpler. Indeed, for the base case of $\ell = 0$, one can view $h_{i,\ell} = x_{i,1}$ and $h_{j,\ell} = x_{j,1}$, and view U_ℓ as an empty matrix. Then, the same analysis of Claim 6.11.7 but with slightly different notations will apply.

Therefore,

$$\begin{aligned} Wh_{i,\ell} + Ax_{i,\ell+1} &= WU_{\ell-1}U_{\ell-1}^\top h_{i,\ell} + W(I - U_{\ell-1}U_{\ell-1}^\top)h_{i,\ell} + Ax_{i,\ell+1} \\ &= WU_{\ell-1}U_{\ell-1}^\top h_{i,\ell} + Wy + Ax_{i,\ell+1} && \text{(by definition of } y\text{)} \\ &= WU_{\ell-1}U_{\ell-1}^\top h_{i,\ell} + (\|y_1\|_2/\|x\|_2)Wx + Ax_{i,\ell+1} + Wy_2 \\ & && \text{(by rewritting } Wy\text{)} \\ &= M_1 z_1 + M_2 z_2 + M_3 z_3 + M_4 z_4 \end{aligned}$$

where

$$\begin{aligned} M_1 &= WU_{\ell-1} & M_2 &= W\frac{x}{\|x\|_2} & M_3 &= A & M_4 &= W\frac{y_2}{\|y_2\|_2} \\ z_1 &= U_{\ell-1}^\top h_{i,\ell} & z_2 &= \|y_1\|_2 & z_3 &= x_{i,\ell+1} & z_4 &= \|y_2\|_2 \ , \end{aligned} \tag{6.35}$$

and we know the entries of M_1, M_2, M_3, M_4 are i.i.d. from $\mathcal{N}(0, \frac{2}{m})$, owing to a similar treatment as Footnote 13. For the vector $M_4 z_4$, we further rewrite it as

$$M_4 z_4 = \nu \cdot (z_4 - c_5\alpha)_+ + \nu' z_4'$$

where $c_5 = \frac{1}{16 \log m}$ is a fixed parameter, and ν and ν' denote two vectors that are independently generated from the same distribution $\mathcal{N}(0, \frac{2}{m}I)$ as vector $M_4 \in \mathbb{R}^m$, and

$$(z_4 - c_5\alpha)_+ = \begin{cases} 0, & \text{if } z_4 < c_5\alpha; \\ \sqrt{z_4^2 - c_5^2\alpha^2}, & \text{if } z_4 \geq c_5\alpha. \end{cases} \qquad z_4' = \begin{cases} z_4, & \text{if } z_4 < c_5\alpha; \\ c_5\alpha, & \text{if } z_4 \geq c_5\alpha. \end{cases}$$

Together, we can write

$$\begin{aligned} (I - UU^\top)h_{i,\ell+1} &= (I - UU^\top)\phi(Wh_{i,\ell} + Ax_{i,\ell+1}) \\ &= (I - UU^\top)\phi(M_1 z_1 + M_2 z_2 + M_3 z_3 + M_4 z_4) \\ &= (I - UU^\top)\phi(M_1 z_1 + M_2 z_2 + M_3 z_3 + \nu \cdot (z_4 - c_5\alpha)_+ + \nu' z_4') \\ &= (I - UU^\top)\phi(w + \nu' z_4') \end{aligned}$$

where the last step follows by defining

$$w = M_1 z_1 + M_2 z_2 + M_3 z_3 + \nu \cdot (z_4 - c_5\alpha)_+ \ .$$

Our plan is to first use the randomness in w to argue that w is $(\alpha, \gamma/4\sqrt{m})$-good. Then we conditioned w is good, and prove that the norm of $(I - UU^\top)\phi(w + \nu' z_4')$ is lower bounded (using (6.11.5)).

Now, suppose that $z_1 \in \mathbb{R}^{n(\ell-1)}$, $z_2 \in \mathbb{R}$, and $z_4 \in \mathbb{R}$ are fixed (instead of computed based on the randomness of W and A), and satisfies[16]

$$\frac{1}{\sqrt{2}} \le \|z_1\| \le 2L + 6 \text{ and } |z_2|, |z_4| \le 2L + 6. \tag{6.36}$$

we can use Corollary 6.10.15 to obtain the following statement: w is $(\alpha, \sigma/4)$-good with probability at least $1 - \exp(-\Omega(\alpha^2 m))$ where $\sigma = \left(\frac{2}{m}\|z_1\|_2^2 + \frac{2}{m}z_2^2 + \frac{2}{m}\|z_3\|_2^2 + \frac{2}{m}(z_4^2 - c_5^2\alpha^2)\right)^{1/2}$. Using $\|z_1\|_2^2 \ge 1/2$, we can lower bound σ^2 as

$$\sigma^2 \ge \frac{2}{m}(\|z_1\|_2^2 + z_2^2 + z_4^2 - c_5^2\alpha^2) \ge \frac{2}{m}(\frac{1}{2} - c_5^2\alpha^2)$$

In other words, w is $(\alpha, \frac{1}{4\sqrt{2}\sqrt{m}})$-good. Next, applying standard ϵ-net argument, we have with probability at least $1 - \exp(-\Omega(\alpha^2 m))$, for all z_1, z_2, z_4 satisfying (6.36), it satisfies w is $(\alpha, \frac{1}{8\sqrt{m}})$-good. This allows us to plug in the random choice of z_1, z_2, z_4 in (6.35).

We apply Lemma 6.11.5 with the following setting

$$U = \left[U_\ell, \frac{(I - U_\ell U_\ell^\top)h_{j,\ell+1}}{\|(I - U_\ell U_\ell^\top)h_{j,\ell+1}\|_2}\right], \quad w = w, \quad v = \nu', \quad r = z_4' \in [0, c_5\alpha]$$

[16]Note that if z_1, z_2 and z_4 are random, then they satisfy such constraints by Lemma 6.11.1.

where w is $(\alpha, \gamma/8\sqrt{m})$-good. (We can do so because the randomness of v is independent of the randomness of U and w.) Lemma 6.11.5 tells us that, with probability at least $1 - \exp(-\Omega(\sqrt{m}))$ over the randomness of v,

$$\|(I - UU^\top) \cdot \phi(w + rv)\| \geq r(1 - 2\alpha) - \frac{\alpha^{1.5}}{4} \ .$$

By induction hypothesis, we know

$$r = z'_4 \geq \min\{z_4, c_5\alpha\} \geq \min\{\delta_{\ell-1}, c_5\alpha\} \geq \delta_{\ell-1} \ ,$$

where the last step follows by $\delta_{\ell-1} \leq c_5\alpha$. Thus, we have

$$\|(I - UU^\top) \cdot \phi(w + rv)\| \geq r(1 - 2\alpha) - \frac{\alpha^{1.5}}{4} \geq \delta_{\ell-1}(1 - 2\alpha) - \frac{\alpha^{1.5}}{4} \geq \delta_{\ell-1}(1 - 2\alpha - \frac{1}{4L}) = \delta_\ell \ ,$$

where the last inequality uses $\alpha^{1.5} \leq \frac{\delta_{\ell-1}}{L}$. □

6.11.4 Intermediate Layers: Spectral Norm

The following lemma bounds the spectral norm of (consecutive) intermediate layers.

Lemma 6.11.8 (c.f. (6.5))**.** *With probability at least* $1-\exp(-\Omega(m/L^2))$*, we have for all* $L \geq \ell_2 \geq \ell_1 \geq 0$ *and* $i \in [n]$

$$\left\| \prod_{\ell=\ell_2}^{\ell_1} D_{i,\ell} W \right\|_2 \leq O(L^3),$$

We start with an important claim whose proof is almost identical to Lemma 6.11.1.

Claim 6.11.9. *Given* $\ell > b \geq 1$*, given* $i \in [n]$*, and given* $z_{b-1} \in \mathbb{R}^m$ *a fixed vector, letting*

$$z_\ell = D_{i,\ell} W D_{i,\ell-1} \cdots D_{i,b} W z_{b-1} \ ,$$

we have with probability at least $1-\exp(-\Omega(m/L^2))$*, it satisfies* $\|z_\ell\|_2 \leq (1+1/L)^{\ell-b+1}\|z_{b-1}\|_2$.

Proof. Let U denote the following column orthonormal matrix using Gram-Schmidt (its first $n(\ell-1)$ columns coincide with $U_{\ell-1}$):

$$U \stackrel{\text{def}}{=} \mathsf{GS}\left(h_{1,1}, \ldots, h_{n,1}, h_{1,2}, \ldots, h_{n,2}, \ldots, h_{1,\ell-1}, \ldots, h_{n,\ell-1}, z_{b-1}, \cdots z_{\ell-1}\right) \ .$$

We can rewrite $\|z_\ell\|_2$ as follows:

$$\begin{aligned}
\|z_\ell\|_2 &= \|D_{i,\ell} W z_{\ell-1}\|_2 \\
&= \left\|\mathbf{1}_{W h_{i,\ell-1} + A x_{i,\ell} \geq 0} \cdot W z_{\ell-1}\right\|_2 \\
&= \left\|\mathbf{1}_{W U U^\top h_{i,\ell-1} + A x_{i,\ell} \geq 0} \cdot W U U^\top z_{\ell-1}\right\|_2 \\
&= \left\|\mathbf{1}_{My + Ax \geq 0} M z\right\|_2 .
\end{aligned}$$

where in the last step we have defined $M = WU$, $y = U^\top h_{i,\ell-1}$, $x = x_{i,\ell}$ and $z = U^\top z_{\ell-1}$. We stress here that the entries of M and A are i.i.d. from $\mathcal{N}(0, \frac{2}{m})$.[17]

Now, Claim 6.11.10 tells us for fixed $x \in \mathbb{R}^d$ and fixed $y, z \in \mathbb{R}^{n\ell+\ell-b}$, we have with probability $1 - \exp(-\Omega(m/L^2))$ (over the randomness of M, A),

$$\|\mathbf{1}_{My+Ax\geq 0} Mz\|_2 \leq \|z\|_2(1 + 1/2L) \ .$$

After taking ϵ-net over all possible y, z, we have that for fixed $x \in \mathbb{R}^d$ but all $y, z \in \mathbb{R}^{n\ell+\ell-b}$:

$$\|\mathbf{1}_{My+Ax\geq 0} Mz\|_2 \leq \|z\|_2(1 + 1/L) \ .$$

We can thus plug in the choice $y = U^\top h_{i,\ell-1}$ and $z = U^\top z_{\ell-1}$ (both of which may depend on the randomness of W and A). Using $\|z\|_2 \leq \|z_{\ell-1}\|_2$, we have

$$\|z_\ell\|_2 \leq \|z_{\ell-1}\|_2(1 + 1/L) \ .$$

Finally, taking union bound over all possible ℓ and applying induction, we have

$$\|z_\ell\|_2 \leq (1 + 1/L)^{\ell-b+1} \|z_{b-1}\|_2. \qquad \square$$

Now, to prove the spectral norm bound in Lemma 6.11.8, we need to go from "for each z_{b-1} (see Claim 6.11.9)" to "for all z_{b-1}". Since z_{b-1} has m dimensions, we cannot afford taking union bound over all possible z_{b-1} (or its ϵ-net).

To bypass this issue, we partition the coordinates of z_{b-1} into L^3 trunk (each of length m/L^3). We write $z_{b-1} = \sum_{j=1}^{L^3} (z_{b-1})_j$ where each $(z_{b-1})_j \in \mathbb{R}^m$ denotes a vector that only

[17] This follows from a similar argument as Footnote 13, taking into account the additional fact that each z_j may only depend on the randomness of A, $WU_{\ell-1}$, and $Wz_{b-1}, \ldots Wz_{j-1}$.

has non-zero entries on m/L^3 coordinates. For each $j \in [L^3]$, we have with probability $1 - \exp(-m/L^2)$,

$$\|(z_\ell)_j\|_2 \leq (1 + 1/L)^{\ell - b} \cdot \|(z_{b-1})_j\|_2 \leq 2\|(z_{b-1})_j\|_2 \ .$$

By applying an ϵ-net argument over all such possible (but sparse) $(z_{b-1})_j$, we have with probability at least $1 - 2^{O(m/L^3)} \exp(-\Omega(m/L^2)) \geq 1 - \exp(-\Omega(m/L^2))$, the above equation holds for all possible $(z_{b-1})_j$.

Next, taking a union bound over all $j \in [L^3]$, we have with probability at least $1 - \exp(-\Omega(m/L^2))$:

$$\forall z_{b-1} \in \mathbb{R}^m \colon \|z_\ell\|_2 \leq O(L^3)\|z_{b-1}\|_2 \ .$$

Taking a union bound over all ℓ, b, we complete the proof of Lemma 6.11.8. ■

6.11.4.1 Tools

Claim 6.11.10. *For fixed $x \in \mathbb{R}^d, y, z \in \mathbb{R}^k$. Let $M \in \mathbb{R}^{m \times k}$ and $A \in \mathbb{R}^{m \times d}$ denote random Gaussian matrices where each entry is i.i.d. sampled from $\mathcal{N}(0, 2/m)$. We have with probability at least $1 - \exp(-\Omega(m/L^2))$*

$$\|\mathbf{1}_{My + Ax \geq 0} \cdot Mz\|_2 \leq \|z\|_2(1 + 1/2L) \ .$$

Proof. Without loss of generality we assume $\|z\|_2 = 1$. We can rewrite My as follows

$$\begin{aligned} My &= M(zz^\top)y + M(I - zz^\top)y \\ &= Mz \cdot z^\top y + \frac{M(I - zz^\top)y}{\|(I - zz^\top)y\|_2} \cdot \|(I - zz^\top)y\|_2 \\ &= M_1 z_1 + M_2 z_2, \end{aligned}$$

where $M_1, M_2 \in \mathbb{R}^m$ and $z_1, z_2 \in \mathbb{R}$ are defined as follows

$$M_1 = Mz, \quad M_2 = \frac{M(I - zz^\top)y}{\|(I - zz^\top)y\|_2}$$
$$z_1 = z^\top y, \quad z_2 = \|(I - zz^\top)y\|_2$$

It is easy to see that M_1 is independent of M_2. We can rewrite

$$\|\mathbf{1}_{My+Ax\geq 0} \cdot Mz\|_2 = \|\mathbf{1}_{M_1 z_1 + M_2 z_2 + Ax \geq 0} \cdot M_1\|_2$$

Using Fact 6.10.11 together with concentration bounds (for binomial distribution and for chi-square distribution), we have with probability at least $1 - \exp(-\Omega(m/L^2))$,

$$\|\mathbf{1}_{M_1 z_1 + M_2 z_2 + Ax \geq 0} \cdot M_1\|_2 \leq \|z\|_2 (1 + 1/2L).$$

Thus, we complete the proof. □

6.11.5 Intermediate Layers: Sparse Spectral Norm

This section proves two results corresponding to the spectral norm of intermediate layers with respect to *sparse* vectors. We first show Lemma 6.11.11 and our Corollary 6.11.12 and 6.11.13 shall be direct applications of Lemma 6.11.11.

Lemma 6.11.11. *For every $k \in [m]$ and $t \geq 2$, with probability at least*

$$1 - nL^2 e^{O(k \log(m))}(e^{-\Omega(m/L^2)} + e^{-\Omega(nLt^2)})$$

it satisfies, for all $i \in [n]$, for all $L > \ell_2 \geq \ell_1 \geq 1$, for all k-sparse vectors $z, y \in \mathbb{R}^m$

$$\left| y^\top W \left(\prod_{\ell=\ell_2}^{\ell_1} D_{i,\ell} W \right) z \right| \leq \frac{5t\sqrt{nL}}{m^{1/2}} \cdot \|y\|_2 \cdot \|z\|_2.$$

Proof. Without loss of generality we assume $\|y\| = \|z\| = 1$. Fixing $\ell_2 \geq \ell_1$, fixing i, and fixing $z_{\ell_1 - 1} = z$, we have according to Claim 6.11.9, letting $z_{\ell_2} = D_{i,\ell_2} W D_{i,\ell_2 - 1} W \cdots D_{i,\ell_1} z_{\ell_1 - 1}$, then with probability at least $1 - e^{-\Omega(m/L^2)}$

$$\|z_{\ell_2}\|_2 \leq (1 + 1/L)^{\ell_2 - \ell_1 - 1} \|z_{\ell_1 - 1}\|_2 \leq 2 \ .$$

Applying ϵ-net over all k-sparse vectors z, we have with probability at least $1 - e^{O(k \log m)} e^{-\Omega(m/L^2)}$, it satisfies $\|z_{\ell_2}\|_2 \leq 3$ for all k-sparse z:

Similar to the proof of Lemma 6.11.8, we let U denote the following column orthonormal matrix using Gram-Schmidt:

$$U \stackrel{\text{def}}{=} \mathsf{GS}\left(h_{1,1}, \ldots, h_{n,1}, h_{1,2}, \ldots, h_{n,2}, \ldots, h_{1,\ell_2 - 1}, \ldots, h_{n,\ell_2 - 1}, z_{\ell_1 - 1}, \cdots z_{\ell_2}\right) \ ,$$

and we have

$$\left| y^\top W \left(\prod_{\ell=\ell_2}^{\ell_1} D_{i,\ell} W \right) z \right| = \left| y^\top W U U^\top z_{\ell_2} \right| \leq \left\| y^\top W U \right\| \cdot \|U^\top z_{\ell_2}\| \leq 3 \left\| y^\top W U \right\| \ .$$

Next, observe the entries of $WU \in \mathbb{R}^{m \times n(\ell_2-1)+(\ell_2-\ell_1+2)}$ are i.i.d. drawn from $\mathcal{N}(0, \frac{2}{m})$ (following a similar argument as Footnote 17). Therefore, if y is a fixed vector, then $y^\top WU$ is in distribution identical to a Gaussian vector $\mathcal{N}(0, \frac{2}{m}I)$ of $n(\ell_2-1)+(\ell_2-\ell_1+2) \le 2nL$ dimensions. By chi-square distribution tail bound (see Lemma 6.10.3), we have for $t \ge 2$:

$$\Pr[\|y^\top WU\|^2 \ge \frac{nLt^2}{m}] \le e^{-\Omega(nLt^2)} \ .$$

Applying ϵ-net over all k-sparse vectors y, we have with probability at least $1-e^{O(k\log m)}e^{-\Omega(nLt^2)}$, it satisfies $\|y^\top WU\|^2 \le \frac{2nLt^2}{m}$ for all k-sparse vectors y.

Conditioning on both events happen, we have

$$\left| y^\top W \left(\prod_{\ell=\ell_2}^{\ell_1} D_{i,\ell} W \right) z \right| \le 3\frac{\sqrt{2nL}t}{\sqrt{m}} \ .$$

Taking union bound over all possible i, ℓ_1, ℓ_2 we finish the proof. □

Choosing $k = s^2 m^{2/3}$ and $t = \frac{sm^{1/3}\log m}{5\sqrt{nL}}$ in Lemma 6.11.11, we have

Corollary 6.11.12 (c.f. (6.6)). *Let $s \in [m^{-1/4}, m^{1/6}]$ be a fixed real. With probability at least $1 - e^{-\Omega(s^2 m^{2/3}\log^2 m)}$, we have for all $i \in [n]$, for all $L > \ell_2 \ge \ell_1 \ge 1$, for all $s^2m^{2/3}$-sparse vectors $y, z \in \mathbb{R}^m$,*

$$\left| z^\top W \left(\prod_{\ell=\ell_2}^{\ell_1} D_{i,\ell} W \right) y \right| \le \frac{s\log m}{m^{1/6}} \cdot \|z\|_2 \cdot \|y\|_2.$$

Choosing $k = 1$ and $t = \frac{\sqrt{nLd}\log m}{5}$ in Lemma 6.11.11, we have

Corollary 6.11.13 (c.f. (6.6)). *Let $\rho = nLd\log m$. With probability at least $1-\exp(-\Omega(\rho^2))$, it satisfies for all $i \in [n]$, for all $L > \ell_2 \ge \ell_1 \ge 1$, and for all 1-sparse vectors $y, z \in \mathbb{R}^m$,*

$$\left| z^\top W \left(\prod_{\ell=\ell_2}^{\ell_1} D_{i,\ell} W \right) y \right| \le \frac{\rho}{m^{1/2}} \cdot \|z\|_2 \cdot \|y\|_2 \ .$$

6.11.6 Backward Propagation

This section proves upper bound on the backward propagation against *sparse* vectors. We first show Lemma 6.11.14 and our Corollary 6.11.15 and 6.11.16 shall be immediate corollaries.

Lemma 6.11.14. *For any $k \in [m]$, $t \geq 2$ and any $a \in \mathbb{R}^d$, with probability at least*

$$1 - nL^2 e^{O(k \log m)}(e^{-\Omega(m/L^2)} + e^{-\Omega(t^2)}),$$

over the randomness of W, A, B, we have for all $i \in [n]$, for all $L \geq \ell_2 \geq \ell_1 \geq 1$, and for all k-sparse $y \in \mathbb{R}^m$,

$$\left| a^\top B \left(\prod_{\ell=\ell_2}^{\ell_1} D_{i,\ell} W \right) y \right| \leq \frac{t}{\sqrt{d}} \cdot \|a\|_2 \cdot \|y\|_2.$$

Proof of Lemma 6.11.14. Using Claim 6.11.9, we know that for fixed $z_{\ell_1 - 1} = y$, letting

$$z_{\ell_2} = D_{i,\ell_2} W D_{i,\ell_2 - 1} W \cdots D_{i,\ell_1} W z_{\ell_1 - 1} \ ,$$

with probability at least $1 - \exp(-\Omega(m/L^2))$ we have

$$\|z_{\ell_2}\|_2 \leq (1 + 1/L)^{\ell_2 - \ell_1 + 1} \|z_{\ell_1 - 1}\|_2 \leq 3\|z_{\ell_1 - 1}\|_2.$$

Next, fixing vectors a and z_{ℓ_2} and letting B be the only source of randomness, we know $a^\top B z_{\ell_2}$ follows from $\mathcal{N}(0, \|a\|_2^2 \|z_{\ell_2}\|_2^2 / d)$. Then with probability at least $1 - \exp(-\Omega(t^2))$ over B,

$$|a^\top B z_{\ell_2}| \leq t \cdot \|a\|_2 \|z_{\ell_2}\|_2 / \sqrt{d}.$$

Taking a union of the above two events, we get with probability $1 - \exp(-\Omega(m/L^2)) - \exp(-\Omega(t^2))$,

$$|a^\top B z_{\ell_2}| \leq \frac{3t}{\sqrt{d}} \cdot \|a\|_2 \cdot \|y\|_2.$$

At this point, we apply ϵ-net argument for all k-sparse vectors $y \in \mathbb{R}^m$ (the size of which is at most $e^{O(k \log(m/k))}$). Taking a union bound over all such vectors in the ϵ-net, we have with probability at least

$$1 - \exp(O(d + k \log(m/k)))(\exp(-m/L^2) + \exp(-\Omega(t^2))) \ ,$$

for all k-sparse vector $y \in \mathbb{R}^m$,

$$|a^\top B z_{\ell_2}| \leq \frac{3t}{\sqrt{d}} \cdot \|a\|_2 \cdot \|y\|_2 \ .$$

Finally, we also take a union bound over all ℓ_2, ℓ_1 and i, and there are at most $O(nL^2)$ choices. □

Using Lemma 6.11.14 with $k = s^2 m^{2/3}$ and $t = s m^{1/3} \log m$, and taking union bound over all $a \in \mathbb{R}^d$, give

Corollary 6.11.15 (c.f. (6.7))**.** *Let $s \in [m^{-1/4}, m^{1/6}]$ be a fixed real. With probability at least $1 - \exp(-\Omega(s^2 m^{2/3} \log^2 m))$, we have for all $i \in [n]$, for all $L \geq \ell_2 \geq \ell_1 \geq 1$, for all $(s^2 \cdot m^{2/3})$-sparse $y \in \mathbb{R}^m$, for all $a \in \mathbb{R}^d$,*

$$\left| a^\top B \left(\prod_{\ell=\ell_2}^{\ell_1} D_{i,\ell} W \right) y \right| \leq (s m^{1/3} \log m) \cdot \|a\|_2 \cdot \|y\|_2 \ .$$

Using Lemma 6.11.14 with $k = 1$ and $t = nLd \log m$, and taking union bound over all $a \in \mathbb{R}^d$, give

Corollary 6.11.16 (c.f. (6.7))**.** *Let* $\rho = nLd \log m$*. With probability at least* $1 - \exp(-\Omega(\rho^2))$*, we have for all* $i \in [n]$*, for all* $L \geq \ell_2 \geq \ell_1 \geq 1$*, for all* 1*-sparse vector* $y \in \mathbb{R}^m$*, and for all* $a \in \mathbb{R}^d$*,*

$$\left| a^\top B \left(\prod_{\ell=\ell_2}^{\ell_1} D_{i,\ell} W \right) y \right| \leq \rho \cdot \|a\|_2 \cdot \|y\|_2.$$

6.12 Stability After Adversarial Perturbation

Throughout this section, we consider some random initialization $\widetilde{W}, A, B$, and some adversarially chosen perturbation $W' \in \mathbb{R}^{m\times m}$ which may depend on the randomness of $\widetilde{W}, A, B$. We introduce the following notations in this section

Definition 6.12.1.

$$\begin{aligned}
&\widetilde{g}_{i,0} = \widetilde{h}_{i,0} = 0 && g_{i,0} = h_{i,0} = 0 && \text{for } i \in [n]\\
&\widetilde{g}_{i,\ell} = \widetilde{W}\widetilde{h}_{i,\ell-1} + Ax_{i,\ell} && g_{i,\ell} = (\widetilde{W} + W')h_{i,\ell-1} + Ax_{i,\ell} && \text{for } i \in [n] \text{ and } \ell \in [L]\\
&\widetilde{h}_{i,\ell} = \phi(\widetilde{W}\widetilde{h}_{i,\ell-1} + Ax_{i,\ell}) && h_{i,\ell} = \phi((\widetilde{W} + W')h_{i,\ell-1} + Ax_{i,\ell}) && \text{for } i \in [n] \text{ and } \ell \in [L]\\
&h'_{i,\ell} = h_{i,\ell} - \widetilde{h}_{i,\ell} && g'_{i,\ell} = g_{i,\ell} - \widetilde{g}_{i,\ell} && \text{for } i \in [n] \text{ and } \ell \in [L]
\end{aligned}$$

Define diagonal matrices $\widetilde{D}_{i,\ell}$ and $D_{i,\ell}$ by letting

$$(\widetilde{D}_{i,\ell})_{k,k} = \mathbb{1}_{(\widetilde{g}_{i,\ell})_k \geq 0} \text{ and } (D_{i,\ell})_{k,k} = \mathbb{1}_{(g_{i,\ell})_k \geq 0}.$$

Accordingly, we let diagonal matrix $D'_{i,\ell} = D_{i,\ell} - \widetilde{D}_{i,\ell}$.

Roadmap

- Section 6.12.1 proves the stability at forward when the perturbation matrix W' has small spectral norm. It gives bounds on $\|g'_{i,\ell}\|_2$, $\|h'_{i,\ell}\|_2$, $\|D'_{i,\ell}\|_0$, and $\|D'_{i,\ell}g_{i,\ell}\|_2$.
- Section 6.12.2 proves the stability of intermediate layers. It gives bound on $\|\prod_{\ell=\ell_2}^{\ell_1} D_{i,\ell}\widetilde{W}\|_2$ and $\|\prod_{\ell=\ell_2}^{\ell_1} D_{i,\ell}W\|_2$.
- Section 6.12.3 analyzes the stability for backward. It bounds the difference $\|a^\top B(\prod_{\ell=\ell_2}^{\ell_1} \widetilde{D}_{i,\ell}\widetilde{W} - \prod_{\ell=\ell_2}^{\ell_1} D_{i,\ell}\widetilde{W})\|_2$.

- Section 6.12.4 considers a special type of rank-one perturbation matrix W', and provides stability bounds on the forward and backward propagation.

As discussed in Section 6.6, the results of Section 6.12.1, 6.12.2 and 6.12.3 shall be used twice, once for the final training updates (see Section 6.16), and once for the randomness decomposition (see Section 6.14). In contrast, the results of Section 6.12.4 shall only be used once in Section 6.14.

6.12.1 Forward

The goal of this section is to prove Lemma 6.12.1,

Lemma 6.12.1 (forward stability, c.f. (6.8))**.** *Letting* $\varrho = nLd\delta^{-1}\log(m/\epsilon)$*, for any* $\tau_0 \in \left[\varrho^{-100}, \varrho^{100}\right]$*, with probability at least* $1 - e^{-\Omega(L^6\tau_0^{4/3}m^{1/3})}$ *over the randomness of* $\widetilde{W}, A, B$*, for every* $W' \in \mathbb{R}^{m\times m}$ *with* $\|W'\|_2 \le \frac{\tau_0}{\sqrt{m}}$*, for every* $i \in [n]$ *and* $\ell \in [L]$*, we have*

(a) $\|g'_{i,\ell}\|_2, \|h'_{i,\ell}\|_2 \le O(L^6\tau_0)/m^{1/2}$ *,*

(b) $\|D'_{i,\ell}\|_0 \le O(L^{10/3}\tau_0^{2/3})\cdot m^{2/3}$ *, and*

(c) $\|D'_{i,\ell}g_{i,\ell}\|_2 \le O(L^5\tau_0)/m^{1/2}$ *.*

Proof of Lemma 6.12.1. Suppose $C > 1$ is a large enough constant so that the hidden constant in Lemma 6.11.8 can be C. We inductively prove that one can write $g'_{i,\ell} = g'_{i,\ell,1} + g'_{i,\ell,2}$ where

$$\begin{aligned}
&\text{I:}\quad \|g'_{i,\ell,1}\|_2 \le \tau_1\cdot\frac{1}{m^{1/2}} \quad \text{II:}\quad \|g'_{i,\ell,2}\|_\infty \le \tau_2\cdot\frac{1}{m} \quad \text{III:}\quad \|g'_{i,\ell}\|_2 \le \tau_3\cdot\frac{1}{m^{1/2}}\\
&\text{IV:}\quad \|D'_{i,\ell}\|_0 \le \tau_4\cdot m^{2/3} \quad \text{V:}\quad \|D'_{i,\ell}g_{i,\ell}\|_2 \le \tau_5\cdot\frac{1}{m^{1/2}} \quad \text{VI:}\quad \|h'_{i,\ell}\|_2 \le (\tau_3+\tau_5)\frac{1}{m^{1/2}}\ .
\end{aligned} \tag{6.37}$$

Above, we choose parameters

$$\tau_1 = 5CL^4(L+2)\tau_0 \qquad \tau_2 = 4L\tau_5\log m \qquad \tau_3 = \tau_1+\tau_2 \qquad \tau_4 = 10(\tau_1)^{2/3} \qquad \tau_5 = 3\tau_1\ .$$

We emphasize that all these parameters are polynomial in ϱ so negligible when comparing to m.

Throughout the proof, we focus on some fixed $i \in [n]$ without loss of generality, and one can always take a union bound at the end. We drop the subscript i for notational

simplicity. In order to prove (6.37), we first assume that it holds for all $1, 2, \ldots, \ell - 1$. In particular, we assume for all $\ell' \leq \ell - 1$,

$$\|g'_{\ell',1}\|_2 \leq \tau_1 \cdot \frac{1}{m^{1/2}} \quad \text{and} \quad \|g'_{\ell',2}\|_\infty \leq \tau_2 \cdot \frac{1}{m} \ .$$

A useful observation is $g'_{\ell'}$ can be split into three terms

$$g'_{\ell'} = \underbrace{W' D_{\ell'-1} g_{\ell'-1}}_{z_{\ell'-1,1}} + \underbrace{\widetilde{W} D'_{\ell'-1} g_{\ell'-1}}_{z_{\ell'-1,2}} + \underbrace{\widetilde{W} \widetilde{D}_{\ell'-1} g'_{\ell'-1}}_{z_{\ell'-1,3}} \ . \tag{6.38}$$

After recursively applying (6.38), we can write

$$g'_\ell = g'_{\ell,1} = \sum_{\ell_a=1}^{\ell-1} \left(\widetilde{W} \widetilde{D}_{\ell-1} \cdots \widetilde{W} \widetilde{D}_{\ell-\ell_a+1} \right) z_{\ell-\ell_a,1} + \left(\widetilde{W} \widetilde{D}_{\ell-1} \cdots \widetilde{W} \widetilde{D}_{\ell-\ell_a+1} \right) z_{\ell-\ell_a,2} \ . \tag{6.39}$$

Applying Claim 6.12.2, with probability at least $1 - e^{-\Omega(\tau_1^{4/3} m^{1/3})}$, we have for all $\ell' \leq \ell - 1$,

$$\|z_{\ell',1}\|_2 \leq \frac{\tau_0}{\sqrt{m}} \left(4\ell' + 8 + \frac{\tau_1 + \tau_2}{\sqrt{m}} \right) \tag{6.40}$$

Applying Claim 6.12.5, we have with probability at least $1 - e^{-\Omega(\tau_1^{4/3} m^{1/3})}$, one can write

$$\left(\widetilde{W} \widetilde{D}_{\ell-1} \cdots \widetilde{W} \widetilde{D}_{\ell-\ell_a+1} \right) z_{\ell-\ell_a,2} = z_{\ell-\ell_a,2^\sharp} + z_{\ell-\ell_a,2^\flat}$$

where

$$\|z_{\ell-\ell_a,2^\sharp}\|_2 \leq \frac{3\tau_5 \sqrt{\tau_4} \log^2 m}{m^{2/3}}, \quad \|z_{\ell-\ell_a,2^\flat}\|_\infty \leq \frac{4\tau_5 \log m}{m} \ . \tag{6.41}$$

As a result, we can define $g'_\ell = g'_{\ell,1} + g'_{\ell,2}$ for

$$g'_{\ell,1} = \sum_{\ell_a=1}^{\ell-1} \left(\widetilde{W} \widetilde{D}_{\ell-1} \cdots \widetilde{W} \widetilde{D}_{\ell-\ell_a+1} \right) z_{\ell-\ell_a,1} + z_{\ell-\ell_a,2^\sharp} \quad \text{and} \quad g'_{\ell,2} = \sum_{\ell_a=1}^{\ell-1} z_{\ell-\ell_a,2^\flat} \ .$$

We first bound $\|g'_{\ell,1}\|_2$,

$$\begin{aligned}\|g'_{\ell,1}\|_2 &\le \sum_{\ell_a=1}^{\ell-1} \|\widetilde{W}\widetilde{D}_{i,\ell-1}\cdots\widetilde{W}\widetilde{D}_{i,\ell-\ell_a+1}\|_2 \cdot \|z_{\ell-\ell_a,1}\|_2 + \|z_{\ell-\ell_a,2^\sharp}\|_2 \\ &\overset{①}{\le} \sum_{\ell_a=1}^{\ell} (CL^3\|z_{\ell-\ell_a,1}\|_2 + \|z_{\ell-\ell_a,2^\sharp}\|_2) \\ &\overset{②}{\le} CL^4\frac{\tau_0}{\sqrt{m}}\left(4L+8+\frac{\tau_1+\tau_2}{\sqrt{m}}\right) + L\frac{3\tau_5\sqrt{\tau_4}\log^2 m}{m^{2/3}} \\ &\le 5CL^4\cdot\frac{\tau_0(L+2)}{\sqrt{m}} \overset{③}{\le} \tau_1\frac{1}{\sqrt{m}} \ .\end{aligned}$$

Above, inequality ① follows from Lemma 6.11.8, inequality ② follows from (6.40) and (6.41), and ③ follows from our choice of $\tau_1 = 5CL^4(L+2)\tau_0$. We next bound $\|g'_{\ell,2}\|_\infty$,

$$\|g'_{\ell,2}\|_\infty \le \sum_{\ell_a=1}^{\ell-1} \|z_{\ell-\ell_a,2^\flat}\|_\infty \overset{①}{\le} L\cdot\frac{4\tau_5\log m}{m} \overset{②}{\le} \tau_2\frac{1}{m}$$

where inequality ① is due to (6.41), and ② follows from our choice of $\tau_2 = 4L\tau_5\log m$. Thus, we have showed I and II of (6.37):

$$\|g'_{\ell,1}\|_2 \le \tau_1\cdot\frac{1}{m^{1/2}} \quad \text{and} \quad \|g'_{\ell,2}\|_\infty \le \tau_2\cdot\frac{1}{m} \ .$$

They together further imply

$$\|g'_\ell\|_2 \le (\tau_1+\tau_2)\frac{1}{m^{1/2}} = \tau_3\frac{1}{m^{1/2}}$$

so III of (6.37) holds. IV and V of (6.37) are implied by Claim 6.12.3, and VI is implied because

$$\|h'_\ell\|_2 = \|\phi(g_\ell)-\phi(\widetilde{g}_\ell)\|_2 \le \|g_\ell-\widetilde{g}_\ell\|_2 \le \|g'_\ell\|_2 \ . \qquad \square$$

6.12.1.1 Tools

Claim 6.12.2. *Suppose* $g_{\ell-1} = \widetilde{g}_{\ell-1} + g'_{\ell-1} = \widetilde{g}_{\ell-1} + g'_{\ell-1,1} + g'_{\ell-1,2}$ *where*

$$\|g'_{\ell-1,1}\|_2 \le \frac{\tau_1}{\sqrt{m}} \quad \text{and} \quad \|g'_{\ell-1,2}\|_\infty \le \frac{\tau_2}{m} \ .$$

Then, we have with probability at least $1 - e^{-\Omega(m/L^2)}$

$$\|W' D_{\ell-1} g_{\ell-1}\|_2 \le \frac{\tau_0}{\sqrt{m}} \left(4\ell + 8 + \frac{\tau_1 + \tau_2}{\sqrt{m}} \right).$$

Proof of Claim 6.12.2. Using triangle inequality, we can calculate

$$\|W' D_{\ell-1} g_{\ell-1}\|_2 \le \|W'\|_2 \cdot \|D_{\ell-1}\|_2 \cdot (\|\widetilde{g}_{\ell-1}\|_2 + \|g'_{\ell-1,1}\|_2 + \|g'_{\ell-1,2}\|_2)$$

Using Lemma 6.11.1, we have $\|\widetilde{g}_{\ell-1}\| \le 4\ell + 8$. Using elementary calculation, we have

$$\|g'_{\ell-1,1}\|_2 + \|g'_{\ell-1,2}\|_2 \le \frac{\tau_1}{\sqrt{m}} + \sqrt{m}\frac{\tau_2}{m} \le \frac{\tau_1 + \tau_2}{\sqrt{m}}$$

Together we finish the proof. □

Claim 6.12.3. *With probability at least* $1 - e^{-\Omega(\tau_1^{4/3} m^{1/3})}$ *the following holds. Whenever*

$$\|g'_{\ell-1,1}\|_2 \le \tau_1 \cdot \frac{1}{m^{1/2}}, \quad \|g'_{\ell-1,2}\|_\infty \le \tau_2 \cdot \frac{1}{m} \ ,$$

then letting $\tau_4 = 10(\tau_1)^{2/3}$ *and* $\tau_5 = 3\tau_1$*, we have*

$$\|D'_{\ell-1} g_{\ell-1}\|_2 \le \tau_5 \cdot \frac{1}{m^{1/2}}, \quad \|D'_{\ell-1}\|_0 \le \tau_4 \cdot m^{2/3}$$

Proof of Claim 6.12.3. We choose parameters $\xi = \frac{(\tau_1)^{2/3}}{10^{1/3} m^{5/6}}$ and $\alpha = 10\xi\sqrt{m}$ in the proof. We have $\|g'_{\ell-1,2}\|_\infty \le \xi$.

First of all, using similar (but simpler) proof as Lemma 6.11.4, one can show with probability at least $1 - \exp(-\Omega(\alpha^2 m))$, the vector $\widetilde{g}_{\ell-1}$ is $(\alpha, \frac{1}{5\sqrt{m}})$-good (recall Definition 6.10.1).[18] This implies that $\widetilde{g}_{\ell-1}$ has at most $\alpha m = 10\xi m^{3/2}$ coordinates j satisfying $|(\widetilde{g}_{\ell-1})_j| \le \frac{\alpha}{5\sqrt{m}} = 2\xi$.

For each $j \in [m]$, if it satisfies $(D'_{\ell-1})_{j,j} \neq 0$, then the sign of $\widetilde{g}_{\ell-1}$ and $\widetilde{g}_{\ell-1} + g'_{\ell-1,1} + g'_{\ell-1,2}$ must differ on coordinate j. As a result:

$$|(g'_{\ell-1,1} + g'_{\ell-1,2})_j| > |(\widetilde{g}_{\ell-1})_j| \ .$$

Define $y = D'_{\ell-1} g_{\ell-1}$. There are two possibilities for such j with $(D'_{\ell-1})_{j,j} \neq 0$.

- Case 1: $|(\widetilde{g}_{\ell-1})_j| \le 2\xi$. Let such coordinates be $S_1 \subset [m]$, and we have $|S_1| \le 10\xi m^{3/2}$ using the above argument.

  Next, for each such $j \in S_1$, we must have $|y_j| = |(\widetilde{g}_{\ell-1} + g'_{\ell-1,1} + g'_{\ell-1,2})_j| \le |(g'_{\ell-1,1} +$

[18] Indeed,

$$\widetilde{g}_{\ell-1} = \widetilde{W}\widetilde{h}_{\ell-2} + Ax_{\ell-1} = M_1 z_1 + M_2 z_2.$$

where $M_1 \in \mathbb{R}^{m \times n(\ell-2)}$, $M_3 \in \mathbb{R}^{m \times d}$ and $z_1 \in \mathbb{R}^{n(\ell-1)}$, $z_3 \in \mathbb{R}^d$ are defined as

$$\begin{aligned} M_1 &= \widetilde{W} U_{\ell-2} & M_2 &= A \\ z_1 &= U_{\ell-2}^\top \widetilde{h}_{\ell-2} & z_2 &= x_{\ell-1} \ . \end{aligned} \tag{6.42}$$

The entries of M_1 and M_2 are i.i.d. from $\mathcal{N}(0, \frac{2}{m})$ (recall Footnote 13). Suppose z_1 and z_2 are fixed (instead of depending on the randomness of $\widetilde{W}$ and A) and satisfies $\frac{1}{\sqrt{2}} \le \|z_1\| \le 2L+6$. We can apply Corollary 6.10.15 to obtain the following statement: with probability at least $1 - \exp(-\Omega(\alpha^2 m))$, $\widetilde{g}_{\ell-1}$ is $(\alpha, \sigma/4)$-good where $\sigma^2 = \frac{2}{m}\|z_1\|_2^2 + \frac{2}{m}\|z_3\|_2^2 \ge \frac{1}{m}$. In other words, w is $(\alpha, \frac{1}{4\sqrt{m}})$-good. Applying standard ϵ-net argument, we have with probability at least $1 - \exp(-\Omega(\alpha^2 m))$, for all vectors $z_1 \in \mathbb{R}^{n(\ell-2)}$ satisfying $\frac{1}{\sqrt{2}} \le \|z_1\| \le 2L+6$, it satisfies $\widetilde{g}_{\ell-1}$ is $(\alpha, \frac{1}{5\sqrt{m}})$-good. This allows us to plug in the random choice of z_1 in (6.42), to conclude that $\widetilde{g}_{\ell-1}$ is $(\alpha, \frac{1}{5\sqrt{m}})$-good.

$g'_{\ell-1,2})_j| \le |(g'_{\ell-1,1})_j| + \xi$ so we can calculate the ℓ_2 norm of y on S_1:

$$\sum_{i\in S_1} y_j^2 \le 2\|g'_{\ell-1,1}\|^2 + 2\xi^2|S_1| \le \frac{2\tau_1^2}{m} + 20\xi^3 m^{3/2} = \frac{4\tau_1^2}{m} \ .$$

- Case 2: $|(\widetilde{g}_{\ell-1})_j| > 2\xi$. Let such coordinates be $S_2 \subset [m] \setminus S_1$. In this case we must have $|(g'_{\ell-1,1})_j| \ge |(\widetilde{g}_{\ell-1})_j| - |(g'_{\ell-1,2})_j| > 2\xi - \xi = \xi$. Therefore, $|S_2| \le \frac{\|g'_{\ell-1,1}\|_2^2}{\xi^2} \le \frac{\tau_1^2}{m\xi^2}$.

  Next, for each $j \in S_2$, we must have

$$|y_j| = |(\widetilde{g}_{\ell-1} + g'_{\ell-1,1} + g'_{\ell-1,2})_j| \le |(g'_{\ell-1,1} + g'_{\ell-1,2})_j| \le |(g'_{\ell-1,1})_j| + \xi/2 \le \frac{5}{4}|(g'_{\ell-1,1})_j| \ .$$

  and therefore

$$\sum_{j\in S_2} y_j^2 \le 2\sum_{j\in S_2} (g'_{\ell-1,1})_j^2 \le \frac{2\tau_1^2}{m} \ .$$

  In sum, we conclude that

$$\|D'_{\ell-1}\|_0 \le |S_1| + |S_2| \le 10\xi m^{3/2} + \frac{\tau_1^2}{m\xi^2} < 10(\tau_1)^{2/3} m^{2/3} = \tau_4 \cdot m^{2/3}$$

and

$$\|y\|_2 \le \frac{\sqrt{4\tau_1^2 + 2\tau_1^2}}{\sqrt{m}} \le \frac{3\tau_1}{\sqrt{m}} = \tau_5 \cdot \frac{1}{m^{1/2}} \ . \qquad \square$$

Claim 6.12.4. *With probability at least $1 - e^{-\Omega(\tau_4 m^{2/3}\log^2 m)}$, for all $L \ge \ell + 1 \ge b \ge 1$, for all*

$$x \in \mathbb{R}^m \quad \textit{with} \quad \|x\|_2 \le \frac{\tau_5}{\sqrt{m}} \quad \textit{and} \quad \|x\|_0 \le \tau_4 \cdot m^{2/3}$$

we have that the vector $y = \widetilde{W}\widetilde{D}_\ell\widetilde{W}\cdots\widetilde{D}_b\widetilde{W}x$ can be written as

$$y = y_1 + y_2 \quad \textit{where} \quad \|y_1\|_2 \le \frac{3\tau_5\sqrt{\tau_4}\log^2 m}{m^{2/3}} \quad \textit{and} \quad \|y_2\|_\infty \le \frac{4\tau_5 \log m}{m}$$

Proof of Claim 6.12.4. Let $s = \tau_4 \cdot m^{2/3}$ for notational simplicity. Let $z_{b-1} = x$ and assume for now that x is a fixed vector. Letting $z_\ell = \widetilde{D}_\ell \widetilde{W} \cdots \widetilde{D}_b z_{b-1}$, we have according to Claim 6.11.9, with probability at least $1 - \exp(-\Omega(m/L^2))$,

$$\|z_\ell\|_2 \le (1 + 1/L)^{\ell - b - 1} \|z_{b-1}\|_2 \le 3\|z_{b-1}\|_2.$$

Similar to the proof of Lemma 6.11.8, we let U denote the following column orthonormal matrix using Gram-Schmidt:

$$U \stackrel{\text{def}}{=} \mathsf{GS}\left(h_{1,1}, \ldots, h_{n,1}, h_{1,2}, \ldots, h_{n,2}, \ldots, h_{1,\ell-1}, \ldots, h_{n,\ell-1}, z_{b-1}, \cdots z_\ell\right) \ ,$$

and we have

$$\widetilde{W}\widetilde{D}_\ell\widetilde{W} \cdots \widetilde{D}_b\widetilde{W}x = \widetilde{W}UU^\top z_{\ell_2} \ .$$

Observe the entries of $\widetilde{W}U \in \mathbb{R}^{m \times n(\ell-1)+(\ell-b+2)}$ are i.i.d. drawn from $\mathcal{N}(0, \frac{2}{m})$ (following a similar argument as Footnote 17). Therefore, according to Lemma 6.10.16, we have can write $y = \widetilde{W}\widetilde{D}_\ell\widetilde{W} \cdots \widetilde{D}_b\widetilde{W}x$ as $y = y_1 + y_2$ with

$$\|y_1\| \le \frac{\sqrt{s}\log^2 m}{\sqrt{m}} \cdot \|U^\top z_{\ell_2}\| \quad \text{and} \quad \|y_2\|_\infty \le \frac{\log m}{\sqrt{m}} \cdot \|U^\top z_{\ell_2}\| \ .$$

Plugging in $\|U^\top z_\ell\|_2 \le \|z_\ell\|_2 \le 3\|z_{b-1}\|_2 = 3\|x\|_2 \le \frac{3\tau_5}{\sqrt{m}}$, we have

$$\|y_1\| \le \frac{3\tau_5\sqrt{s}\log^2 m}{m} \quad \text{and} \quad \|y_2\|_\infty \le \frac{3\tau_5 \log m}{m} \ .$$

Finally, taking ϵ-net over all s-sparse vectors x, we have the desired result. □

Combining Claim 6.12.3 and Claim 6.12.4, we have

Claim 6.12.5. *With probability at least* $1 - e^{-\Omega(\tau_1^{4/3} m^{1/3})}$*, whenever*

$$\|g'_{\ell-1,1}\|_2 \le \frac{\tau_1}{\sqrt{m}}, \qquad \|g'_{\ell-1,2}\|_\infty \le \frac{\tau_2}{m} \ ,$$

we have the vector $y = \widetilde{W}\widetilde{D}_\ell\widetilde{W} \cdots \widetilde{D}_b\widetilde{W}D'_{\ell-1}g_{\ell-1}$ *can be written as*

$$y = y_1 + y_2 \quad \textit{where} \quad \|y_1\|_2 \le \frac{3\tau_5\sqrt{\tau_4}\log^2 m}{m^{2/3}} \quad \textit{and} \quad \|y_2\|_\infty \le \frac{4\tau_5 \log m}{m}$$

6.12.2 Intermediate Layers

The goal of this subsection is to prove Lemma 6.12.6.

Lemma 6.12.6 (intermediate stability, c.f. (6.9))**.** *Letting $\varrho = nLd\log m$, for any $\tau_0 \in \left[\varrho^{-100}, \varrho^{100}\right]$, with probability at least $1 - e^{-\Omega(L^6\tau_0^{4/3}m^{1/3})}$ over the randomness of $\widetilde{W}, A, B$, for every $W' \in \mathbb{R}^{m\times m}$ with $\|W'\|_2 \le \frac{\tau_0}{\sqrt{m}}$, for every $i \in [n]$, for every $L \ge \ell_2 \ge \ell_1 \ge 1$, the following holds*

- $\left\|\prod_{\ell=\ell_2}^{\ell_1}(\widetilde{D}_{i,\ell} + D'_{i,\ell})\widetilde{W}\right\|_2 \le O(L^7)$.
- $\left\|\prod_{\ell=\ell_2}^{\ell_1}(\widetilde{D}_{i,\ell} + D'_{i,\ell})(\widetilde{W} + W')\right\|_2 \le O(L^7)$.

We show Claim 6.12.7 and then use it to prove Corollary 6.12.7.

Claim 6.12.7. *Let $s \in [m^{-1/4}, m^{1/8}]$ be any fixed real. With probability at least $1 - e^{-\Omega(s^2m^{2/3}\log^2 m)}$, it satisfies that for every $i \in [n]$, for every $L \ge \ell_2 \ge \ell_1 \ge 1$, if $D'_{i,\ell} \in \mathbb{R}^{m\times m}$ is an arbitrary diagonal matrix with entries in $\{-1, 0, 1\}$ of sparsity $\|D'_{i,\ell}\|_0 \le s^2m^{2/3}$ (for each $\ell = \ell_1, \ldots, \ell_2$), then*

$$\left\|\prod_{\ell=\ell_2}^{\ell_1}(\widetilde{D}_{i,\ell} + D'_{i,\ell})\widetilde{W}\right\|_2 \le O(L^7) \ .$$

Proof of Claim 6.12.7. We define set $\mathcal{C}$ as follows

$$\mathcal{C} = \left\{ \prod_{\ell=\ell_2}^{\ell_1}(D'_{i,\ell})^{c_\ell}(\widetilde{D}_{i,\ell})^{1-c_\ell}\widetilde{W} \;\middle|\; c_\ell \in \{0,1\}, \forall \ell \in [\ell_1, \ell_2] \right\}.$$

For ℓ, we define set $\mathcal{C}_\ell$ as follows

$$\mathcal{C}_\ell = \left\{ C \in \mathcal{C} \;\middle|\; D' \text{ appears } \ell \text{ times in } C \right\}$$

From Lemma 6.11.8, we can bound the $\ell = 0$ term

$$\left\| \sum_{C \in \mathcal{C}_\ell} C \right\|_2 \leq O(L^3).$$

For each $C \in \mathcal{C}_\ell$ with $\ell > 0$, we have

$$\|C\|_2 \leq O(L^3) \cdot \left(\frac{s \log m}{m^{1/6}} \right)^{\ell - 1} \cdot O(L^3)$$

where we apply Lemma 6.11.8 twice(one on the left, one on the right) and apply Corollary 6.11.12 $\ell - 1$ times. The probability of the above event is at least $1 - e^{\Omega(s^2 \cdot m^{2/3} \log^2 m)}$. Therefore, for each $\ell \in [\ell_2 - \ell_1]$, we can bound

$$\left\| \sum_{C \in \mathcal{C}_\ell} C \right\|_2 \leq O(L^6) \binom{\ell_2 - \ell_1}{\ell} \left(\frac{s \log m}{m^{1/6}} \right)^{\ell - 1}$$

which is at most $O(L^7)$ when $\ell = 1$ and $o(L^7)$ for $\ell > 1$ by our parameter choices. Putting it altogether,

$$\text{LHS} = \left\| \sum_{\ell=0}^{\ell_2 - \ell_1} \sum_{C \in \mathcal{C}_\ell} C \right\|_2 \leq \sum_{\ell=0}^{\ell_2 - \ell_1} \left\| \sum_{C \in \mathcal{C}_\ell} C \right\|_2 \leq O(L^3 + L^7) \qquad \square$$

Proof of Lemma 6.12.6. Applying Lemma 6.12.1c, we know with probability $\geq 1 - e^{-\Omega(L^6 \tau_0^{4/3} m^{1/3})}$ it satisfies $\|D'_{i,\ell}\|_0 \leq s^2 m^{2/3}$ for $s^2 = O(L^{10/3} \tau_0^{2/3})$. Therefore, applying Claim 6.12.7 we have

$$\left\| \prod_{\ell=\ell_2}^{\ell_1} (\widetilde{D}_{i,\ell} + D'_{i,\ell}) \widetilde{W} \right\|_2 \leq O(L^7) \ .$$

On the other hand, since $\|W'\| \leq \frac{\tau_0}{\sqrt{m}}$, we also have

$$\left\| \prod_{\ell=\ell_2}^{\ell_1} (\widetilde{D}_{i,\ell} + D'_{i,\ell}) (\widetilde{W} + W') \right\|_2 \leq O(L^7) \ .$$

by expanding out the $2^{\ell_2 - \ell_1 + 1}$ terms with respect to W'. $\square$

6.12.3 Backward

We state the main result in this section.

Lemma 6.12.8 (backward stability, c.f. (6.10)). *Letting $\varrho = nLd\log m$, for any $\tau_0 \in \left[\varrho^{-100}, \varrho^{100}\right]$, with probability at least $1 - e^{-\Omega(L^6\tau_0^{4/3}m^{1/3})}$ over the randomness of $\widetilde{W}, A, B$, for every $W' \in \mathbb{R}^{m\times m}$ with $\|W'\|_2 \le \frac{\tau_0}{\sqrt{m}}$, for every $1 \le \ell_1 \le \ell_2 \le L$, for every $i \in [n]$, the followings hold*

(a) $\left\|a^\top B\prod_{\ell=\ell_2}^{\ell_1}\widetilde{D}_{i,\ell}\widetilde{W} - a^\top B\prod_{\ell=\ell_2}^{\ell_1}D_{i,\ell}\widetilde{W}\right\|_2 \le O\big(\tau_0^{1/3}L^6\log m\cdot m^{1/3}\big)\cdot\|a\|_2.$

(b) $\left\|a^\top B\prod_{\ell=\ell_2}^{\ell_1}D_{i,\ell}\widetilde{W} - a^\top B\prod_{\ell=\ell_2}^{\ell_1}D_{i,\ell}W\right\|_2 \le O(\tau_0 L^{15})\cdot\|a\|_2.$

(c) $\left\|a^\top B\prod_{\ell=\ell_2}^{\ell_1}\widetilde{D}_{i,\ell}\widetilde{W} - a^\top B\prod_{\ell=\ell_2}^{\ell_1}\widetilde{D}_{i,\ell}W\right\|_2 \le O(\tau_0 L^{7})\cdot\|a\|_2.$

(d) $\left\|a^\top B\prod_{\ell=\ell_2}^{\ell_1}\widetilde{D}_{i,\ell}W - a^\top B\prod_{\ell=\ell_2}^{\ell_1}D_{i,\ell}W\right\|_2 \le O\big(\tau_0^{1/3}L^6\log m\cdot m^{1/3}\big)\cdot\|a\|_2.$

We first use Lemma 6.12.1 to derive that, with probability $\ge 1 - e^{-\Omega(L^6\tau_0^{4/3}m^{1/3})}$, $\|D'_{i,\ell}\|_0 \le s^2m^{2/3}$, for parameter $s = O(L^{5/3}\tau_0^{1/3})$. We prove the five statements separately.

Proof of Lemma 6.12.8a. Without loss of generality we assume $\|a\| = 1$. We define set $\mathcal{C}$ to be

$$\mathcal{C} = \left\{\prod_{\ell=\ell_2}^{\ell_1}(D'_{i,\ell})^{c_\ell}(\widetilde{D}_{i,\ell})^{1-c_\ell}\widetilde{W} \;\middle|\; c_{\ell_1},\ldots,c_{\ell_2}\in\{0,1\}\right\}.$$

For each $\ell \in [\ell_2 - \ell_1]$, we define set $\mathcal{C}_\ell$ to be

$$\mathcal{C}_\ell = \{C \in \mathcal{C} \mid D' \text{ appears } \ell \text{ times in } C\}.$$

Ignoring subscripts for the ease of presentation, by Corollary 6.11.15, we know

$$\|a^\top B\widetilde{D}\widetilde{W}\cdots\widetilde{D}\widetilde{W}D'\|_2 \le (s\log m)m^{1/3} \ .$$

By Corollary 6.11.12, we know

$$\left\|D'\widetilde{W}\widetilde{D}\cdots\widetilde{D}\widetilde{W}D'\right\|_2 \le (s\log m)m^{-1/6} \ .$$

Thus, for each ℓ and $C \in \mathcal{C}_\ell$,

$$\begin{aligned}\|a^\top BC\|_2 &\le \|a^\top B\widetilde{D}\widetilde{W}\cdots\widetilde{D}\widetilde{W}D'\|_2 \cdot \left\|D'\widetilde{W}\widetilde{D}\cdots\widetilde{D}\widetilde{W}D'\right\|_2^{\ell-1} \cdot \left\|D'\widetilde{W}\widetilde{D}\cdots\widetilde{D}\widetilde{W}\right\|_2 \\ &\le (s\cdot m^{1/3}\log m)\times(s\cdot m^{-1/6}\log m)^{\ell-1}\times O(L^3) \ .\end{aligned}$$

Finally, the LHS of the Lemma 6.12.8a becomes

$$\begin{aligned}\text{LHS} &= \left\|\sum_{\ell=1}^{\ell_2-\ell_1}\sum_{C\in\mathcal{C}_\ell} a^\top BC\right\|_2 \le \sum_{\ell=1}^{\ell_2-\ell_1}\sum_{C\in\mathcal{C}_\ell}\|a^\top BC\|_2 \\ &\le \sum_{\ell=1}^{\ell_2-\ell_1}\binom{\ell_2-\ell_1}{\ell}(s\cdot m^{1/3}\log m)\times(s\cdot m^{-1/6}\log m)^{\ell-1}\times O(L^3) \le O\big(L^4\cdot s\log m\cdot m^{1/3}\big) \ .\end{aligned}$$

□

Proof of Lemma 6.12.8b. Again we assume $\|a\| = 1$ without loss of generality. We define set $\mathcal{C}$ to be

$$\mathcal{C} = \left\{\prod_{\ell=\ell_2}^{\ell_1} D_{i,\ell}(W')^{c_\ell}\widetilde{W}^{1-c_\ell} \;\middle|\; c_{\ell_1},\ldots,c_{\ell_2}\in\{0,1\}\right\}$$

For each $\ell \in [\ell_2-\ell_1]$, we define set $\mathcal{C}_\ell$

$$\mathcal{C}_\ell = \{C\in\mathcal{C} \mid W' \text{ appears } \ell \text{ times in } C\} \ .$$

Ignoring subscripts for the ease of presentation, we have for each $C \in \mathcal{C}_\ell$,

$$\|a^\top BC\|_2 \le \|a^\top B\|_2 \cdot \|\widetilde{W}D \cdots \widetilde{W}D\|_2 \cdot \|W'\|_2 \cdot \left(\|D\widetilde{W} \cdots \widetilde{W}D\|_2 \|W'\|_2\right)^{\ell-1} \cdot \|D\widetilde{W} \cdots D\widetilde{W}\|_2 \tag{6.43}$$

$$\le O(\sqrt{m}L^7) \cdot \left(\frac{\tau_0}{\sqrt{m}}\right) \cdot \left(\frac{O(\tau_0 L^7)}{\sqrt{m}}\right)^{\ell-1} \cdot O(L^7) \ . \tag{6.44}$$

Above, we have used $\|W'\|_2 \le \frac{\tau_0}{\sqrt{m}}$, $\|\widetilde{W}D \cdots \widetilde{W}D\|_2 \le O(L^7)$ from Lemma 6.12.6, and $\|B\|_2 \le O(\sqrt{m})$ with high probability. Finally, by triangle inequality, the LHS of Lemma 6.12.8b becomes

$$\begin{aligned}
\text{LHS} &= \left\| \sum_{\ell=1}^{\ell_2-\ell_1} \sum_{C \in \mathcal{C}_\ell} a^\top BC \right\|_2 \le \sum_{\ell=1}^{\ell_2-\ell_1} \sum_{C \in \mathcal{C}_\ell} \left\| a^\top BC \right\|_2 \\
&\le \sum_{\ell=1}^{\ell_2-\ell_1} \binom{\ell_2-\ell_1}{\ell} O(\sqrt{m}L^7) \cdot \left(\frac{\tau_0}{\sqrt{m}}\right) \cdot \left(\frac{O(\tau_0 L^7)}{\sqrt{m}}\right)^{\ell-1} \cdot O(L^7) \\
&\le O(\tau_0 L^{15})
\end{aligned}$$

□

Proof of Lemma 6.12.8c. It is similar to the proof of Lemma 6.12.8b. □

Proof of Lemma 6.12.8d. It is similar to the proof of Lemma 6.12.8a. □

6.12.4 Special Rank-One Perturbation

We prove two lemmas regarding the special rank-one perturbation with respect to a coordinate $k \in [m]$. The first one talks about forward propagation.

Lemma 6.12.9 (c.f. (6.11))**.** *Let $\rho = nLd\log m$, $\varrho = nLd\delta^{-1}\log(m/\epsilon)$, $\tau_0 \in [\varrho^{-100}, \varrho^{50}]$, $N \in [1, \varrho^{100}]$. If $\widetilde{W}, A$ are at random initialization, then with probability at least $1 - e^{-\Omega(\rho^2)}$, for any rank-one perturbation $W' = yz^\top$ with $\|z\| = 1$, $\|y\|_0 = N$, $\|y\|_\infty \le \frac{\tau_0}{\sqrt{m}}$, it satisfies*

$$\forall i \in [n], \forall \ell \in [L], \forall k \in [m]: \quad |((\widetilde{W} + W')h'_{i,\ell})_k| \le O\Big(\frac{\rho^8 N^{2/3}\tau_0^{5/6}}{m^{2/3}}\Big) \ .$$

Proof. Without loss of generality we only prove the statement for fixed i, ℓ, k because one can take union bound at the end. We can rewrite $Wh'_{i,\ell}$ as follows

$$\begin{aligned}
(\widetilde{W} + W')h'_{i,\ell} &= W'h'_{i,\ell} + \widetilde{W}h'_{i,\ell} \\
&= W'h'_{i,\ell} + \widetilde{W}(h_{i,\ell} - \widetilde{h}_{i,\ell}) \\
&= W'h'_{i,\ell} + \widetilde{W}((\widetilde{D}_{i,\ell} + D'_{i,\ell})g_{i,\ell} - \widetilde{D}_{i,\ell}\widetilde{g}_{i,\ell}) \\
&= W'h'_{i,\ell} + \widetilde{W}\widetilde{D}_{i,\ell}g'_{i,\ell} + \widetilde{W}D'_{i,\ell}g_{i,\ell} \\
&= W'h'_{i,\ell} + \widetilde{W}\widetilde{D}_{i,\ell}((\widetilde{W} + W')h_{i,\ell-1} - \widetilde{W}\widetilde{h}_{i,\ell-1}) + \widetilde{W}D'_{i,\ell}g_{i,\ell} \\
&= W'h'_{i,\ell} + \widetilde{W}\widetilde{D}_{i,\ell}(\widetilde{W}h'_{i,\ell-1} + W'h_{i,\ell-1}) + \widetilde{W}D'_{i,\ell}g_{i,\ell} \\
&= W'h'_{i,\ell} + \widetilde{W}\widetilde{D}_{i,\ell}\underbrace{\widetilde{W}h'_{i,\ell-1}}_{\text{recurse}} + \widetilde{W}\widetilde{D}_{i,\ell}W'h_{i,\ell-1} + \widetilde{W}D'_{i,\ell}g_{i,\ell}
\end{aligned}$$

After recursively calculating as above, using $h'_{i,0} = 0$, and applying triangle inequality, we have

$$|((\widetilde{W} + W')h'_{i,\ell})_k| \le |(W'h'_{i,\ell})_k| + \sum_{a=0}^{\ell} |(\widetilde{W}\widetilde{D}_{i,\ell}\cdots\widetilde{W}\widetilde{D}_{i,a}W'h_{i,a})_k| + \sum_{a=0}^{\ell} |(\widetilde{W}\widetilde{D}_{i,\ell}\cdots\widetilde{W}\widetilde{D}_{i,a+1}\widetilde{W}D'_{i,a}g_{i,a})_k| \tag{6.45}$$

We bound the three types of terms on the RHS of (6.45) as follows.

- First, we can bound

$$|(W'h'_{i,\ell})_k| = |e_k^\top yz^\top h'_{i,\ell}| = |e_k^\top y| \cdot |z^\top h'_{i,\ell}| \le \|y\|_\infty \cdot \|h'_{i,\ell}\|_2 \le \frac{\tau_0}{\sqrt{m}} \cdot (\frac{O(L^6\tau_0\sqrt{N})}{\sqrt{m}})$$

  where the last step follows by $\|y\|_\infty \le \frac{\tau_0}{\sqrt{m}}$, $\|W'\|_2 \le \frac{\sqrt{N}\tau_0}{\sqrt{m}}$ and Lemma 6.12.1a.

- Second, we can bound

$$\begin{aligned}
|(\widetilde{W}\widetilde{D}_{i,\ell}\cdots\widetilde{W}\widetilde{D}_{i,a}W'h_{i,a})_k| &= |e_k^\top\widetilde{W}\widetilde{D}_{i,\ell}\cdots\widetilde{W}\widetilde{D}_{i,a}yz^\top h_{i,a}| \\
&= |e_k^\top\widetilde{W}\widetilde{D}_{i,\ell}\cdots\widetilde{W}\widetilde{D}_{i,a}y| \cdot |z^\top h_{i,a}| \\
&\overset{①}{\le} |e_k^\top\widetilde{W}\widetilde{D}_{i,\ell}\cdots\widetilde{W}\widetilde{D}_{i,a}y| \cdot \|h_{i,a}\|_2 \\
&\overset{②}{\le} |e_k^\top\widetilde{W}\widetilde{D}_{i,\ell}\cdots\widetilde{W}\widetilde{D}_{i,a}y| \cdot O(L) \\
&\le N \cdot \|y\|_\infty \max_j |e_k^\top\widetilde{W}\widetilde{D}_{i,\ell}\cdots\widetilde{W}\widetilde{D}_{i,a}e_j| \cdot O(L) \\
&\overset{③}{\le} N \cdot \|y\|_\infty \cdot \frac{\rho}{\sqrt{m}} \cdot O(L) \\
&\overset{④}{\le} N \cdot (\tau_0\frac{1}{\sqrt{m}}) \cdot \frac{\rho}{\sqrt{m}} \cdot O(L) \le O(\frac{N\tau_0\rho L}{m}).
\end{aligned}$$

  where ① follows by $\|z\|_2 \le 1$, ② follows by Lemma 6.11.1, ③ follows by Corollary 6.11.13, and ④ follows by $\|y\|_\infty \le \tau_0\frac{1}{\sqrt{m}}$.

- Third, we can bound,

$$\begin{aligned}
|(\widetilde{W}\widetilde{D}_{i,\ell}\cdots\widetilde{W}\widetilde{D}_{i,a+1}\widetilde{W}D'_{i,a}g_{i,a})_k| &\le \|e_k\widetilde{W}\widetilde{D}_{i,\ell}\cdots\widetilde{W}\widetilde{D}_{i,a+1}\widetilde{W}D'_{i,a}\|_2 \cdot \|D'_{i,a}g_{i,a}\|_2 \\
&\le O\left(\frac{L^{5/3}\tau_0^{1/3}N^{1/6}\log m}{m^{1/6}}\right) \cdot O\left(\frac{L^5\tau_0^{1/2}\sqrt{N}}{\sqrt{m}}\right)
\end{aligned}$$

  where the last step follows by Corollary 6.11.12 (which relies on Lemma 6.12.1b to give $s = O(L^{5/3}\tau_0^{1/3}N^{1/6})$) together with Lemma 6.12.1c.

Putting the three bounds into (6.45) (where the term one is the dominating term), we have

$$|((\widetilde{W}+W')h'_{i,\ell})_k| \le O\left(\frac{\rho^8 N^{2/3}\tau_0^{5/6}}{m^{2/3}}\right) \ . \qquad \square$$

The next one talks about backward propagation.

Lemma 6.12.10 (c.f. (6.12))**.** *Let $\rho = nLd\log m$, $\varrho = nLd\delta^{-1}\log(m/\epsilon)$, $\tau_0 \in [\varrho^{-100}, \varrho^{50}]$, $N \in [1, \varrho^{100}]$. If $\widetilde{W}, A, B$ are at random initialization, and given some perturbation $W' = yz^\top$ with $\|z\| = 1$, $\|y\|_0 = N$, $\|y\|_\infty \le \frac{\tau_0}{\sqrt{m}}$ where W' can only depend on the randomness of $\widetilde{W}$ and A (but not on B), then with probability at least $1-e^{-\Omega(\rho^2)}$, for all $i \in [n]$, all $1 \le \ell_1 \le \ell_2 \le L$, all $k \in [m]$:*

(a) $\left|a^\top B\left(\prod_{\ell=\ell_2}^{\ell_1} D_{i,\ell}\widetilde{W}\right)e_k - a^\top B\left(\prod_{\ell=\ell_2}^{\ell_1} \widetilde{D}_{i,\ell}\widetilde{W}\right)e_k\right| \le \|a\|_2 \cdot O\big(\frac{\rho^3\tau_0^{1/3}N^{1/6}}{m^{1/6}}\big)$,

(b) $\left|a^\top B\left(\prod_{\ell=\ell_2}^{\ell_1} D_{i,\ell}W\right)e_k - a^\top B\left(\prod_{\ell=\ell_2}^{\ell_1} D_{i,\ell}\widetilde{W}\right)e_k\right| \le \|a\|_2 \cdot O\big(\frac{\rho^{12}N^{5/6}\tau_0^{5/3}}{m^{1/3}}\big)$.

Proof. We prove for a fixed $i \in [n]$ and drop the subscript in i for notational simplicity. Without loss of generality we assume $\|a\| = 1$.

(a) We define set $\mathcal{C}$ to be

$$\mathcal{C} = \left\{ \prod_{\ell=\ell_2}^{\ell_1} (D'_\ell)^{c_\ell}(\widetilde{D}_\ell)^{1-c_\ell}\widetilde{W} \;\middle|\; c_\ell \in \{0,1\}, \forall \ell \in [\ell_1, \ell_2] \right\}.$$

For each $\ell \in [\ell_2 - \ell_1]$, we define set $\mathcal{C}_\ell$ to be

$$\mathcal{C}_\ell = \{C \in \mathcal{C} \mid D' \text{ appears } \ell \text{ times in } C\}.$$

Ignoring the subscripts for the ease of presentation, by Lemma 6.11.8, we know

$$\|\widetilde{D}\widetilde{W}\cdots\widetilde{D}\widetilde{W}\cdot D'\|_2 \le \|\widetilde{D}\widetilde{W}\cdots\widetilde{D}\widetilde{W}\|_2 \le O(L^3) \ .$$

By Corollary 6.11.12 (which relies on Lemma 6.12.1b to give $s = O(L^{5/3}\tau_0^{1/3}N^{1/6})$), we know

$$\left\|D'\widetilde{W}\widetilde{D}\cdots\widetilde{D}\widetilde{W}D'\right\|_2 \le O(\frac{L^{5/3}\tau_0^{1/3}N^{1/6}}{m^{1/6}}\log m) \ ,$$
$$\forall k\in[m]\colon \left\|D'\widetilde{W}\widetilde{D}\cdots\widetilde{D}\widetilde{W}e_k\right\|_2 \le O(\frac{L^{5/3}\tau_0^{1/3}N^{1/6}}{m^{1/6}}\log m) \ .$$

Thus, for each $k\in[m]$ and $C\in\mathcal{C}_\ell$, we can bound it

$$\|Ce_k\| \le L^3 \times \left(O(\frac{L^{5/3}\tau_0^{1/3}N^{1/6}}{m^{1/6}}\log m)\right)^\ell \ .$$

As a consequence, letting

$$v = \left(\prod_{\ell=\ell_2}^{\ell_1} D_{i,\ell}\widetilde{W}\right)e_k - \left(\prod_{\ell=\ell_2}^{\ell_1}\widetilde{D}_{i,\ell}\widetilde{W}\right)e_k$$

we have

$$\begin{aligned}\|v\| &= \left\|\sum_{\ell=1}^{\ell_2-\ell_1}\sum_{C\in\mathcal{C}_\ell} Ce_k\right\| \\ &\le \sum_{\ell=1}^{\ell_2-\ell_1}\sum_{C\in\mathcal{C}_\ell}\|Ce_k\| \le \sum_{\ell=1}^{\ell_2-\ell_1}\binom{L}{\ell}\cdot\left(O(\frac{L^{5/3}\tau_0^{1/3}N^{1/6}}{m^{1/6}}\log m)\right)^\ell \le O\left(\frac{\rho^3\tau_0^{1/3}N^{1/6}}{m^{1/6}}\right) \ .\end{aligned}$$

Finally, we use the randomness of B (recall that v does not depend on B), we conclude that with probability at least $1-e^{-\Omega(m)}$, it satisfies

$$\text{LHS} = \left\|a^\top Bv\right\| \le O\left(\frac{\rho^3\tau_0^{1/3}N^{1/6}}{m^{1/6}}\right) \ .$$

(b) This time, we define set $\mathcal{C}$ to be

$$\mathcal{C} = \left\{\prod_{\ell=\ell_2}^{\ell_1} D_\ell(W')^{c_\ell}\widetilde{W}^{1-c_\ell} \;\middle|\; c_\ell\in\{0,1\},\forall\ell\in[\ell_1,\ell_2]\right\}$$

For each $\ell \in [\ell_2 - \ell_1]$, we define set $\mathcal{C}_\ell$

$$\mathcal{C}_\ell = \{C \in \mathcal{C} \mid W' \text{ appears } \ell \text{ times in } C\}.$$

One can carefully derive that[19]

$$|a^\top B D\widetilde{W} \cdots \widetilde{W} D y| \le O\left(\frac{\rho^4 N^{5/6} \tau_0^{5/3}}{m^{1/3}}\right) .$$

By Lemma 6.12.6 we have

$$|z^\top D\widetilde{W} \cdots \widetilde{W} D y| \le \|z\|_2 \cdot O(L^7) \cdot \|y\|_2 \le O\Big(\frac{L^7 \sqrt{N}\tau_0}{\sqrt{m}}\Big) ,$$
$$|z^\top D\widetilde{W} \cdots \widetilde{W} D e_k| \le \|z\|_2 \cdot O(L^7) \cdot \|e_k\|_2 \le O(L^7) .$$

Therefore, for each $C \in \mathcal{C}_\ell$ we can upper bound $\|a^\top B C\|_2$ as follows

$$|a^\top B C e_k| \le O\left(\frac{\rho^4 N^{5/6} \tau_0^{5/3}}{m^{1/3}}\right) \cdot \left(O\Big(\frac{L^7 \sqrt{N}\tau_0}{\sqrt{m}}\Big)\right)^{\ell-1} \cdot O(L^7)$$

[19]Since we have done this too many times, let us quickly point out the calculation. By Corollary 6.11.16, we have with probability at least $1 - e^{-\Omega(\rho^2)}$

$$|a^\top B \widetilde{D}\widetilde{W} \cdots \widetilde{W}\widetilde{D} y| \le \|a\|_2 \cdot \sqrt{|N|} \log m \cdot \|y\|_2 \le \|y\|_\infty \|y\|_0 \cdot \rho \le \frac{\tau_0 |N|}{\sqrt{m}} .$$

Using binomial expansion again, we have

$$\begin{aligned}
|a^\top B \widetilde{D}\widetilde{W} \cdots \widetilde{W}\widetilde{D} y - a^\top B D\widetilde{W} \cdots \widetilde{W} D y| &\le \sum_{\ell=1}^{L} \binom{L}{\ell} \|a^\top B \widetilde{D}\widetilde{W} \cdots \widetilde{W} D'\| \cdot \|D'\widetilde{W} \cdots \widetilde{W} D'\|_2^{\ell-1} \cdot \|D'\widetilde{W} \cdots \widetilde{W}\widetilde{D} y\| \\
&\le \sum_{\ell=1}^{L} \binom{L}{\ell} (s m^{1/3} \log m) \cdot \left(\frac{s \log m}{m^{1/6}}\right)^{\ell-1} \cdot \left(\frac{s \log m}{m^{1/6}}\right) \cdot \|y\| \\
&\le O(L)(s m^{1/3} \log m) \left(\frac{s \log m}{m^{1/6}}\right) \cdot \frac{\sqrt{N}\tau_0}{\sqrt{m}} = O\left(\frac{\rho^4 N^{5/6} \tau_0^{5/3}}{m^{1/3}}\right)
\end{aligned}$$

Here, we have used Corollary 6.11.15 and Corollary 6.11.12 with parameter $s = O(L^{5/3} \tau_0^{1/3} N^{1/6})$ chosen from Lemma 6.12.1b, as well as $\|y\| \le \frac{\sqrt{N}\tau_0}{\sqrt{m}}$.

and therefore

$$\begin{aligned}\text{LHS} &= \left|\sum_{\ell=1}^{\ell_2-\ell_1}\sum_{C\in\mathcal{C}_\ell} a^\top BCe_k\right| \\ &\le \sum_{\ell=1}^{\ell_2-\ell_1}\binom{\ell_2-\ell_1}{\ell}\cdot O\left(\frac{\rho^4 N^{5/6}\tau_0^{5/3}}{m^{1/3}}\right)\cdot\left(O\Big(\frac{L^7\sqrt{N}\tau_0}{\sqrt{m}}\Big)\right)^{\ell-1}\cdot O(L^7) \le O\left(\frac{\rho^{12}N^{5/6}\tau_0^{5/3}}{m^{1/3}}\right)\ .\end{aligned}$$

□

6.13 Indicator and Backward Coordinate Bounds

In this section, let $W \in \mathbb{R}^{m \times m}$, $A \in \mathbb{R}^{m \times d_x}$ and $B \in \mathbb{R}^{d \times m}$ denote three random matrices where each entry of W and A is i.i.d. sampled from $\mathcal{N}(0, \frac{2}{m})$ and each entry of B is i.i.d. sampled from $\mathcal{N}(0, \frac{1}{d})$. We let $\{ \mathsf{loss}_{i,\ell} \in \mathbb{R}^d \colon i \in [n], \ell \in [L] \backslash \{1\} \}$ be arbitrary fixed vectors, and let $i^*, \ell^* = \arg\max_{i,\ell} \| \mathsf{loss}_{i,\ell} \|_2$.

- In Section 6.13.1, we prove that there are sufficiently many coordinates $k \in [m]$ such that the quantity $|(g_{i,\ell+1})_k| = |(Wh_{i,\ell} + Ax_{i,\ell+1})_k|$ is small for $(i, \ell) = (i^*, \ell^*)$ but large for $(i, \ell) \in [n] \times \{\ell^*, \ell^* + 1, L\} \backslash (i^*, \ell^*)$.
- In Section 6.13.2, we prove that there are sufficiently many coordinates $k \in [m]$ such that $|\sum_a (\mathsf{Back}_{i^*,\ell^* \to a}^\top \cdot \mathsf{loss}_{i^*,a})_k|$ is large.

6.13.1 Indicator Coordinate Bound

For this section, we fix some choice of $i^* \in [n]$ and $\ell^* \in [L]\backslash\{1\}$, and define two parameters

Definition 6.13.1.

$$\boxed{\beta_+ \stackrel{\text{def}}{=} \frac{\delta}{\rho^2} \quad \text{and} \quad \beta_- \stackrel{\text{def}}{=} \frac{\delta}{\rho^{10}}}$$

Lemma 6.13.1 (indicator coordinate bound, c.f. (6.15))**.** *Suppose W and A follow from random initialization. Given fixed set $N_1 \subseteq [m]$ and define*

$$N_4 \stackrel{\text{def}}{=} \left\{ k \in N_1 \;\middle|\; \begin{bmatrix} |\big(Wh_{i,\ell} + Ax_{i,\ell+1}\big)_k| \le \beta_-/\sqrt{m}, & \text{if } i = i^*, \ell = \ell^*; \\ |\big(Wh_{i,\ell} + Ax_{i,\ell+1}\big)_k| \ge \beta_+/\sqrt{m}, & \text{if } i \neq i^* \text{ and } \ell = \ell^*; \\ |\big(Wh_{i,\ell} + Ax_{i,\ell+1}\big)_k| \ge \beta_+/\sqrt{m}, & \text{if } i \in [n] \text{ and } \ell > \ell^*. \end{bmatrix} \right\}$$

Then, as long as $|N_1| \ge n^2L^2/\beta_-$, we have

$$\Pr_{W,A}\left[|N_4| \ge \frac{\beta_-|N_1|}{64L}\right] \ge 1 - e^{-\Omega(\beta_-|N_1|/L)}.$$

Lemma 6.13.2. *Suppose W and A follow from random initialization. Given fixed set $N_1 \subseteq [m]$ and define*

$$N_2 \stackrel{\text{def}}{=} \left\{ k \in N_1 \;\middle|\; |\langle W_k, h_{i^*,\ell^*}\rangle + \langle A_k, x_{i^*,\ell^*+1}\rangle| \le \frac{\beta_-}{\sqrt{m}} \right\}.$$

Then, we have

$$\Pr_{W,A}\left[|N_2| \ge \frac{\beta_-|N_1|}{16L}\right] \ge 1 - e^{-\Omega(\beta_-|N_1|/L)}.$$

Proof. Let $i = i^*$ and $\ell = \ell^*$. Similar to the proof of Lemma 6.11.1, we can write

$$y = Wh_{i,\ell} + Ax_{i,\ell+1} = WU_\ell U_\ell^\top h_{i,\ell} + Ax_{i,\ell+1} = M_1 z_1 + M_2 z_2$$

where $M_1 \in \mathbb{R}^{m \times n\ell}$, $M_2 = A$, $z_1 = U_\ell^\top h_{i,\ell}$ and $z_2 = x_{i,\ell+1}$. Again, the entries of M_1 and M_2 are i.i.d. from $\mathcal{N}(0, \frac{2}{m})$.

Now, suppose z_1 is fixed (instead of depending on the randomness of W and A), and it satisfies $\|z_1\| \le 6L$. We have each entry of y is is distributed as $\mathcal{N}\big(0, \frac{2\|z_1\|^2+2\|z_2\|^2}{m}\big)$. By property of Gaussian (see Fact 6.10.12), we have for each $y \in [m]$,

$$\Pr\left[|y_k| \le \frac{0.9\beta_-}{\sqrt{m}}\right] \ge \frac{2}{3}\frac{0.9\beta_-}{\sqrt{2\|z_1\|^2 + 2\|z_2\|^2}} \ge \frac{\beta_-}{15L} \ .$$

Since we have $|N_1|$ independent random variables, applying a Chernoff bound, we have

$$\Pr\left[\left|\left\{k \in N_1 \colon |y_k| \le \frac{0.9\beta_-}{\sqrt{m}}\right\}\right| \ge \frac{\beta_-}{16L}|N_1|\right] \ge 1 - \exp(-\Omega(\beta_-|N_1|/L)) \ .$$

Finally, by taking a union bound with respect to the ϵ-net over all possible choices of $z_1 \in \mathbb{R}^{n\ell}$, we have that

$$\Pr\left[\left|\left\{k \in N_1 \colon |y_k| \le \frac{\beta_-}{\sqrt{m}}\right\}\right| \ge \frac{\beta_-}{16L}|N_1|\right] \ge 1 - \exp(-\Omega(\beta_-|N_1|/L)) \ .$$

for all z_1 and a fixed z_2. Plugging in the random choice of $z_1 = U_\ell^\top h_{i,\ell}$ (and we have $\|z_1\| \le 6L$ by Lemma 6.11.1), we have the desired result. □

Observe that the set N_2 produced by Lemma 6.13.2 only depends on the randomness of A, WU_{ℓ^*-1} and Wh_{i^*,ℓ^*}. In other words, for any unit vector $z \in \mathbb{R}^m$ that is orthogonal to the columns of U_{ℓ^*-1} and orthogonal to h_{i^*,ℓ^*}, we have that Wz is independent of N_2.

We next proceed to refine N_2:

Lemma 6.13.3 ($i \ne i^*, \ell = \ell^*$)**.** *Suppose W and A follow from random initialization, and $N_2 \subseteq [m]$ is a given set (that may only depend on the randomness of A, WU_{ℓ^*-1} and Wh_{i^*,ℓ^*}).*

Define

$$N_3 = \left\{ k \in N_2 \;\middle|\; |\langle W_k, h_{i,\ell^*}\rangle + \langle A_k, x_{i,\ell^*+1}\rangle| \geq \frac{\beta_-}{\sqrt{m}}, \forall i \in [n]\backslash i^* \right\}.$$

Then

$$\Pr\left[|N_3| \geq \frac{1}{2}|N_2| \;\middle|\; A, WU_{\ell^*-1}, Wh_{i^*,\ell^*} \right] \geq 1 - e^{-\Omega(|N_2|/n)}.$$

Proof. First fix some $i \in [n]\backslash i^*$ and $\ell = \ell^*$. Define set $N_{3,i}$,

$$N_{3,i} \stackrel{\text{def}}{=} \left\{ k \in N_2 \;\middle|\; |\langle W_k, h_{i,\ell}\rangle + \langle A_k, x_{i,\ell+1}\rangle| \geq \frac{\beta_-}{\sqrt{m}} \right\} \ .$$

Let $v \stackrel{\text{def}}{=} \frac{(I - U_{\ell-1}U_{\ell-1}^\top)h_{i^*,\ell}}{\|(I - U_{\ell-1}U_{\ell-1}^\top)h_{i^*,\ell}\|}$, and let column orthonormal matrix

$$U \stackrel{\text{def}}{=} \mathsf{GS}(U_{\ell-1}, h_{i^*,\ell}) = [U_{\ell-1}, v] \ .$$

We have

$$\langle W_k, h_{i,\ell}\rangle + \langle A_k, x_{i,\ell+1}\rangle = \left(WUUh_{i,\ell} + Ax_{i,\ell+1} + W(I - UU^\top)h_{i,\ell} \right)_k \ .$$

The rest of the proof is focusing on having the WU and A being fixed (so the first two vectors $WUUh_{i,\ell} + Ax_{i,\ell+1}$ are fixed), and letting $W(I - UU^\top)h_{i,\ell}$ be the only random variable. We have

$$W(I - UU^\top)h_{i,\ell} \sim \mathcal{N}\left(0, \frac{2\|(I - UU^\top)h_{i,\ell}\|_2^2 \cdot I}{m}\right) \ .$$

and therefore for a fixed vector μ it satisfies

$$Wh_{i,\ell} + Ax_{i,\ell+1} = WUUh_{i,\ell} + Ax_{i,\ell+1} + W(I - UU^\top)h_{i,\ell} \sim \mathcal{N}\left(\mu, \frac{2\|(I - UU^\top)h_{i,\ell}\|_2^2 \cdot I}{m}\right) \ .$$

According to Lemma 6.11.6, with probability at least $1-\exp(-\Omega(\sqrt{m}))$, it satisfies $\|(I-UU^\top)h_{i,\ell}\|_2 \geq \frac{\delta}{2}$. By property of Gaussian distribution (see Fact 6.10.12), we have for each $k \in N_2$:

$$\Pr\left[|(Wh_{i,\ell}+Ax_{i,\ell+1})_k| \geq \frac{\beta_-}{\sqrt{m}} \,\middle|\, WU, A\right] \geq 1-\frac{\beta_-}{\delta} \geq 1-\frac{1}{4n} \ .$$

Applying Chernoff bound, with probability at least $1-e^{-\Omega(|N_2|/n)}$, we have $|N_{3,i}| \geq (1-\frac{1}{2n})|N_2|$. Applying union bound over all possible $i \in [n] \setminus \{i^*\}$, we have $N_3 = \bigcap_{i\neq i^*} N_{3,i}$ has cardinality at least $\frac{|N_2|}{2}$. □

Lemma 6.13.4 ($\ell > \ell^*$)**.** *Suppose W and A follow from random initialization, and $N_3 \subseteq [m]$ is a given set (that may only depend on the randomness of A, WU_{ℓ^*}). Define*

$$N_4 = \left\{ k \in N_3 \,\middle|\, |\langle W_k, h_{i,\ell}\rangle + \langle A_k, x_{i,\ell+1}\rangle| \geq \frac{\beta_-}{\sqrt{m}}, \forall i \in [n], \ell > \ell^* \right\} \ .$$

Then

$$\Pr\left[|N_4| \geq \frac{1}{2}|N_3| \,\middle|\, A, WU_{\ell^*}\right] \geq 1-e^{-\Omega(|N_3|/nL)} \ .$$

Proof. Letting $N_{4,\ell^*} \stackrel{\text{def}}{=} N_3$. For any $i \in [n]$ and $\ell = \ell^*+1, \ell^*+2, \ldots, L$, we define $N_{4,i,\ell}$ as

$$N_{4,i,\ell} \stackrel{\text{def}}{=} \left\{ k \in N_{4,\ell-1} \,\middle|\, |\langle W_k, h_{i,\ell}\rangle + \langle A_k, x_{i,\ell+1}\rangle| \geq \frac{\beta_-}{\sqrt{m}} \right\} \quad \text{and} \quad N_{4,\ell} \stackrel{\text{def}}{=} \bigcap_{i\in[n]} N_{4,i,\ell} \ .$$

We rewrite

$$\langle W_k, h_{i,\ell}\rangle + \langle A_k, x_{i,\ell+1}\rangle = (WU_{\ell-1}U_{\ell-1}^\top h_{i,\ell} + Ax_{i,\ell+1} + W(I-U_{\ell-1}U_{\ell-1}^\top)h_{i,\ell})_k \ .$$

The rest of the proof is focusing on having the $WU_{\ell-1}$ and A being fixed (so the first two vectors $WUUh_{i,\ell}+Ax_{i,\ell+1}$ are fixed and $N_{4,\ell-1}$ is fixed), and letting $W(I-U_{\ell-1}U_{\ell-1}^\top)h_{i,\ell}$ be

the only random variable. We have

$$W(I - U_{\ell-1}U_{\ell-1}^\top)h_{i,\ell} \sim \mathcal{N}\left(0, \frac{2\|(I - U_{\ell-1}U_{\ell-1}^\top)h_{i,\ell}\|_2^2 \cdot I}{m}\right) .$$

and therefore for a fixed vector μ it satisfies

$$Wh_{i,\ell}+Ax_{i,\ell+1} = WU_{\ell-1}U_{\ell-1}h_{i,\ell}+Ax_{i,\ell+1}+W(I-U_{\ell-1}U_{\ell-1}^\top)h_{i,\ell} \sim \mathcal{N}\left(\mu, \frac{2\|(I - U_{\ell-1}U_{\ell-1}^\top)h_{i,\ell}\|_2^2 \cdot I}{m}\right) .$$

According to Lemma 6.11.4, with probability at least $1 - \exp(-\Omega(\sqrt{m}))$, it satisfies $\|(I - UU^\top)h_{i,\ell}\|_2 \geq \frac{\delta}{2}$. By property of Gaussian distribution (see Fact 6.10.12), we have for each $k \in N_{4,\ell-1}$:

$$\Pr\left[|(\widetilde{W}h_{i,\ell} + Ax_{i,\ell+1})_k| \geq \frac{\beta_-}{\sqrt{m}} \,\middle|\, WU, A\right] \geq 1 - \frac{\beta_-}{\delta} \geq 1 - \frac{1}{4nL} .$$

Applying Chernoff bound, we have

$$\Pr\left[|N_{4,i,\ell}| \geq (1 - \frac{1}{3nL})|N_{4,\ell-1}| \,\middle|\, WU_{\ell-1}, A\right] \geq 1 - e^{-\Omega(|N_{4,\ell-1}|/nL)} .$$

Applying union bound over all possible $i \in [n]$, we have $N_{4,\ell} = \bigcap_{i\in[n]} N_{4,i,\ell}$ has cardinality at least $(1 - \frac{1}{3L})|N_{4,\ell-1}|$. After telescoping we have $|N_4| = |N_{4,L}| \geq \frac{1}{2}|N_3|$. □

6.13.2 Backward Coordinate Bound

Recall the following notions

Definition 6.13.2. Given matrices W, A, B. For each i, for each ℓ, we define

$$\mathsf{Back}_{i,\ell\to\ell} = B \in \mathbb{R}^{d\times m}.$$

For each i, for each $a \geq \ell + 1$, we define

$$\mathsf{Back}_{i,\ell\to a} = BD_{i,a}W \cdots D_{i,\ell+1}W \in \mathbb{R}^{d\times m}.$$

The goal of this section is to prove Lemma 6.13.5.

Lemma 6.13.5 (backward coordinate bound, c.f. (6.14))**.** *Let $\rho = ndL\log m$ and $\varrho = nLd\delta^{-1}\log(m/\epsilon)$. Let $\big\{\mathsf{loss}_{i,\ell} \in \mathbb{R}^d \colon i \in [n], \ell \in [L]\backslash\{1\}\big\}$ be fixed vectors and $i^*, \ell^* = \arg\max_{i,\ell} \|\mathsf{loss}_{i,\ell}\|_2$. If $W \in \mathbb{R}^{m\times m}, A \in \mathbb{R}^{m\times d_x}, B \in \mathbb{R}^{d\times m}$ are random, and $N_4 \subseteq [m]$ is a set with $|N_4| \in [\rho^4, \varrho^{100}]$ (N_4 can depend on the randomness of W and A, but not depend on B). Define*

$$N_5 \stackrel{\text{def}}{=} \left\{ k \in N_4 \;\middle|\; \left| \sum_{a=\ell^*}^{L} (\mathsf{Back}_{i^*,\ell^*\to a}^\top \cdot \mathsf{loss}_{i^*,a})_k \right| \geq \frac{\|\mathsf{loss}_{i^*,\ell^*}\|_2}{6\sqrt{d}nL} \right\}.$$

Then with probability at least $1 - e^{-\Omega(\rho^2)}$, we have

$$|N_5| \geq \left(1 - \frac{1}{2nL}\right)|N_4|.$$

Proof. For notational simplicity, in the proof we use ℓ to denote ℓ^*, use N to denote N_4, and drop the subscript i^*. We denote $B \in \mathbb{R}^{d\times m}$ as $[b_1, b_2, \cdots, b_m]$ where each $b_i \in \mathbb{R}^d$. We denote by $C_{a,\ell+1} = D_a W \cdots D_{\ell+1}W$.

We define a function $v_k(b_1, b_2, \cdots, b_m) \in \mathbb{R}$ as follows

$$v_k(b_1, b_2, \cdots, b_m) \stackrel{\text{def}}{=} \sum_{a=\ell}^{L} (\mathsf{Back}_{\ell \to a}^\top \cdot \mathsf{loss}_{i,a})_k.$$

We can rewrite

$$\begin{aligned}
v_k(b_1, b_2, \cdots, b_m) = & \left(B^\top \cdot \mathsf{loss}_{i,\ell} + \sum_{a=\ell+1}^{L} \mathsf{Back}_{\ell \to a}^\top \cdot \mathsf{loss}_{i,a} \right)_k \\
= & \left(B^\top \cdot \mathsf{loss}_{i,\ell} + \sum_{a=\ell+1}^{L} (C_{a,\ell+1})^\top B^\top \cdot \mathsf{loss}_{i,a} \right)_k \\
= & \langle b_k, \mathsf{loss}_{i,\ell} \rangle + \sum_{a=\ell+1}^{L} \left\langle \sum_{j=1}^{m} (C_{a,\ell+1})_{k,j} \cdot b_j, \mathsf{loss}_{i,a} \right\rangle \\
= & \left\langle b_k, \mathsf{loss}_{i,\ell} + \sum_{a=\ell+1}^{L} (C_{a,\ell+1})_{k,j} \cdot \mathsf{loss}_{i,a} \right\rangle + \sum_{j \in [m] \setminus k} \left\langle b_j, \sum_{a=\ell+1}^{L} (C_{a,\ell+1})_{k,j} \cdot \mathsf{loss}_{i,a} \right\rangle
\end{aligned}$$

where we have used $(C_{a,\ell+1})_{k,j}$ to denote the (k,j) entry of matrix $C_{a,\ell+1}$.

Choose parameter $\mathfrak{L}_h = \frac{3\sqrt{d}nL}{\| \mathsf{loss}_{i,\ell} \|_2}$ and we can define function $h \colon \mathbb{R} \to [0,1]$ by

$$h(t) \stackrel{\text{def}}{=} \begin{cases} |t| \cdot \mathfrak{L}_h, & \text{if } |t| \leq 1/\mathfrak{L}_h; \\ 1, & \text{otherwise.} \end{cases}$$

We have $h(t)$ is $\mathfrak{L}_h$-Lipschitz continuous.

We define two probabilistic events E_1, E_2.

- Event E_1 depends on the randomness of W and A:

$$E_1 \stackrel{\text{def}}{=} \left\{ |z^\top C_{a,\ell+1} y| \leq \frac{\rho}{\sqrt{m}} \|z\| \|y\| \text{ for all } a, \ell \text{ and for all 1-sparse vectors } y, z \right\}$$

Corollary 6.11.13 says $\Pr[E_1] \geq 1 - e^{-\Omega(\rho^2)}$.

- Event E_2 depends on the randomness of B:

$$E_2 \stackrel{\text{def}}{=} \left\{ |B_{i,j}| \leq \frac{\rho}{\sqrt{d}} \text{ for all } i,j \right\}$$

  By Gaussian tail bound, we have $\Pr[E_2] \geq 1 - e^{-\Omega(\rho^2)}$.

In the rest of the proof, we assume W and A are fixed and satisfy E_1. We let B be the only source of randomness but we condition on B satisfies E_2.

For simplicity we use subscript $-j$ to denote $[m] \setminus j$, and subscript $-N$ to denote $[m] \setminus N$. For instance, we shall write $b = (b_{-k}, b_k) = (b_N, b_{-N})$.

Step 1 Fixing b_{-N} and letting b_N be the only randomness, we claim for each $k \in N$:

$$\mathop{\mathbb{E}}_{b_N}[h(v_k)] \geq \Pr_{b_N}\left[|v_k| \geq \frac{1}{\mathfrak{L}_h}\right] \geq 1 - \frac{1}{4nL} \ .$$

We prove this inequality as follows. We can rewrite

$$\begin{aligned} v_k = & \left\langle b_k, \mathsf{loss}_{i,\ell} + \sum_{a=\ell+1}^{L} (C_{a,\ell+1})_{k,k} \cdot \mathsf{loss}_{i,a} \right\rangle + \sum_{j \in N \setminus \{k\}} \left\langle b_j, \sum_{a=\ell+1}^{L} (C_{a,\ell+1})_{k,j} \cdot \mathsf{loss}_{i,a} \right\rangle \\ & + \sum_{j \in [m] \setminus N} \left\langle b_j, \sum_{a=\ell+1}^{L} (C_{a,\ell+1})_{k,j} \cdot \mathsf{loss}_{i,a} \right\rangle \end{aligned}$$

and v_k is distributed as Gaussian random variable $\mathcal{N}(\mu, \sigma^2)$, and μ and σ^2 are defined as follows

$$\begin{aligned} \mu &= \sum_{j \in [m] \setminus N} \left\langle b_j, \sum_{a=\ell+1}^{L} (C_{a,\ell+1})_{k,j} \cdot \mathsf{loss}_{i,a} \right\rangle \\ \sigma^2 &= \frac{1}{d} \left\| \mathsf{loss}_{i,\ell} + \sum_{a=\ell+1}^{L} (C_{a,\ell+1})_{k,k} \cdot \mathsf{loss}_{i,a} \right\|_2^2 + \frac{1}{d} \sum_{j \in N \setminus \{k\}} \left\| \sum_{a=\ell+1}^{L} (C_{a,\ell+1})_{k,j} \cdot \mathsf{loss}_{i,a} \right\|_2^2 \end{aligned}$$

Meanwhile, we calculate that

$$\left\| \sum_{a=\ell+1}^{L} (C_{a,\ell+1})_{k,j} \cdot \mathsf{loss}_{i,a} \right\|_2 \overset{①}{\leq} \sum_{a=\ell+1}^{L} |(C_{a,\ell+1})_{k,j}| \cdot \| \mathsf{loss}_{i,a} \|_2$$
$$\overset{②}{\leq} \sum_{a=\ell+1}^{L} |(C_{a,\ell+1})_{k,j}| \cdot \| \mathsf{loss}_{i,\ell} \|_2 \overset{③}{\leq} \frac{L\rho}{\sqrt{m}} \cdot \| \mathsf{loss}_{i,\ell} \|_2 \tag{6.46}$$

where ① follows by triangle inequality, ② follows by $i, \ell = \arg\max_{i',\ell'} \| \mathsf{loss}_{i',\ell'} \|$, and ③ follows from event E_1. Therefore,

$$\sigma^2 \geq \frac{1}{d} \left(\| \mathsf{loss}_{i,\ell} \|_2 - \left\| \sum_{a=\ell+1}^{L} (C_{a,\ell+1})_{k,j} \cdot \mathsf{loss}_{i,a} \right\|_2 \right)^2$$
$$\geq \frac{1}{d} \left(1 - \frac{L\rho}{\sqrt{m}} \right)^2 \| \mathsf{loss}_{i,\ell} \|_2^2 \geq \frac{1}{2d} \| \mathsf{loss}_{i,\ell} \|_2^2$$

where the first step follows by $\|a+b\|_2 \geq \|a\|_2 - \|b\|_2$, the second step follows by (6.46). In sum, we have that v_k is a random Gaussian with $\sigma^2 \geq \frac{1}{2d} \| \mathsf{loss}_{i,\ell} \|_2^2$. This means,

$$\Pr_{b_N} \left[|v_k| \geq \frac{1}{\mathfrak{L}_h} \right] \geq \Pr_{b_N} \left[|v_k| \geq \frac{\| \mathsf{loss}_{i,\ell} \|_2}{3\sqrt{d} n L} \right] \geq 1 - \frac{1}{4nL} \ .$$

according to Fact 6.10.12.

Step 2 For every $b_1, b_2, \cdots, b_m$ and $b'_k \in \mathbb{R}^d$ satisfying event E_2, for every $j \neq k$, we have

$$|v_j(b_{-k}, b_k) - v_j(b_{-k}, b'_k)| = \left| \left\langle b_k - b'_k, \sum_{a=\ell+1}^{L} (C_{a,\ell+1})_{k,k} \cdot \mathsf{loss}_{i,a} \right\rangle \right|$$
$$\leq \|b_k - b'_k\|_2 \cdot \left\| \sum_{a=\ell+1}^{L} (C_{a,\ell+1})_{k,k} \cdot \mathsf{loss}_{i,a} \right\|_2$$
$$\leq \rho \cdot \frac{L\rho}{\sqrt{m}} \| \mathsf{loss}_{i,\ell} \|_2 \leq \frac{\rho^3}{\sqrt{m}} \| \mathsf{loss}_{i,\ell} \|_2 \tag{6.47}$$

where the second step follows by $|\langle a, b \rangle| \leq \|a\|_2 \cdot \|b\|_2$, the third step follows by (6.46).

Step 3 Fixing b_{-N} and letting function $g(b_1, b_2, \cdots, b_m) \stackrel{\text{def}}{=} \sum_{k\in N} h(v_k(b_1, b_2, \cdots, b_m))$. We have

$$\begin{aligned}
|g(b_j, b_{-j}) - g(b'_j, b_{-j})| &= \left|\sum_{k\in N} h(v_k(b_j, b_{-j})) - \sum_{k\in N} h(v_k(b'_j, b_{-j}))\right| \\
&\le \sum_{k\in N} \left|h(v_k(b_j, b_{-j})) - h(v_k(b'_j, b_{-j}))\right| \\
&= \left|h(v_j(b_j, b_{-j})) - h(v_j(b'_j, b_{-j}))\right| + \sum_{k\in N\setminus\{j\}} \left|h(v_k(b_j, b_{-j})) - h(v_k(b'_j, b_{-j}))\right| \\
&\overset{①}{\le} 1 + \sum_{k\in N\setminus\{j\}} \left|h(v_k(b_j, b_{-j})) - h(v_k(b'_j, b_{-j}))\right| \\
&\le 1 + \sum_{k\in N\setminus\{j\}} \mathfrak{L}_h \cdot |v_k(b_j, b_{-j}) - v_k(b'_j, b_{-j})| \\
&\overset{②}{\le} 1 + |N| \cdot \frac{3\sqrt{d}nL}{\|\,\mathsf{loss}_{i,\ell}\,\|_2} \cdot \left((nLd\log m)^3 \frac{\|\,\mathsf{loss}_{i,\ell}\,\|_2}{\sqrt{m}}\right) \\
&\le 1 + \frac{|N| \cdot 3(nLd\log m)^4}{\sqrt{m}} \le 2 \ .
\end{aligned}$$

Above, inequality ① follows by $h(t) \in [0, 1]$, inequality ② follows from (6.47). Using McDiarmid inequality (see Lemma 6.10.17), we have

$$\begin{aligned}
&\Pr_{b_N}\left[\left|g(b_1, \cdots, b_m) - \mathop{\mathbb{E}}_{b_N}[g]\right| \ge \epsilon\right] \le \exp\left(-\frac{\epsilon^2}{8|N|}\right) \\
\implies &\Pr_{b_N}\left[\left|g(b_1, \cdots, b_m) - \mathop{\mathbb{E}}_{b_N}[g]\right| \ge \frac{|N|}{4nL}\right] < \exp\left(-\frac{|N|}{128(nL)^2}\right)
\end{aligned}$$

where we choose $\epsilon = \frac{|N|}{4nL}$.

Finally, combining this with Step 1 —which gives $\mathbb{E}_{b_N}[g(b_1, \cdots, b_m)] \ge |N|(1 - \frac{1}{4nL})$— we have with probability at least $1 - e^{-\Omega(|N|/(nL)^2)}$ over randomness of b_N,

$$g(b_1, \cdots, b_m) \ge |N|\left(1 - \frac{1}{2nL}\right) \ .$$

This implies at least $(1 - \frac{1}{2nL})$-fraction of $k \in N$ will have $|v_k| \ge \frac{1}{2\mathfrak{L}_h} = \frac{\|\,\mathsf{loss}_{i,\ell}\,\|_2}{6nL\sqrt{d}}$. □

6.14 Gradient Bound at Random Initialization (Theorem 6.14.2)

The goal of this section is to understand the lower and upper bounds of gradients. Instead of analyzing the true gradient directly —where the forward and backward propagation have correlated randomness— we assume that the loss vectors $\{\mathsf{loss}_{i,a}\}_{i\in[n],a\in\{2,\dots,L\}}$ are fixed (no randomness) in this section, as opposed to being defined as $\mathsf{loss}_{i,a} = Bh_{i,a} - y^*_{i,a}$ which is random. We call this the "fake loss." We define a corresponding "fake gradient" with respect to this fixed loss.

Definition 6.14.1 (fake gradient, c.f. (6.13))**.** Given fixed vectors $\{\mathsf{loss}_{i,a}\}_{i\in[n],a\in\{2,\dots,L\}}$, we define

$$\widehat{\nabla}_k f(W) \stackrel{\text{def}}{=} \sum_{i=1}^n \sum_{a=2}^L \sum_{\ell=1}^{a-1} (\mathsf{Back}_{i,\ell+1\to a}^\top \cdot \mathsf{loss}_{i,a})_k \cdot h_{i,\ell} \cdot \mathbf{1}_{\langle W_k, h_{i,\ell}\rangle + \langle A_k, x_{i,\ell+1}\rangle \geq 0}$$

$$\widehat{\nabla} f(W) \stackrel{\text{def}}{=} \sum_{i=1}^n \sum_{a=2}^L \sum_{\ell=1}^{a-1} D_{i,\ell+1}(\mathsf{Back}_{i,\ell+1\to a}^\top \cdot \mathsf{loss}_{i,a}) \cdot h_{i,\ell}^\top$$

$$\widehat{\nabla} f_i(W) \stackrel{\text{def}}{=} \sum_{a=2}^L \sum_{\ell=1}^{a-1} D_{i,\ell+1}(\mathsf{Back}_{i,\ell+1\to a}^\top \cdot \mathsf{loss}_{i,a}) \cdot h_{i,\ell}^\top$$

In our analysis, we also write it as

$$\widehat{\nabla}_k f(W) = \sum_{i=1}^n \sum_{\ell=1}^{L-1} (u_{i,\ell})_k \cdot h_{i,\ell} \cdot \mathbf{1}_{(g_{i,\ell+1})_k \geq 0}$$

$$\widehat{\nabla} f(W) = \sum_{i=1}^n \sum_{\ell=1}^{L-1} D_{i,\ell+1} u_{i,\ell} \cdot h_{i,\ell}^\top$$

$$\widehat{\nabla} f_i(W) = \sum_{\ell=1}^{L-1} D_{i,\ell+1} u_{i,\ell} \cdot h_{i,\ell}^\top$$

where vectors $u_{i,\ell}, g_{i,\ell+1} \in \mathbb{R}^m$ are defined as

$$u_{i,\ell} \stackrel{\text{def}}{=} \sum_{a=\ell+1}^L \mathsf{Back}_{i,\ell+1\to a}^\top \cdot \mathsf{loss}_{i,a} \qquad\qquad g_{i,\ell+1} \stackrel{\text{def}}{=} Wh_{i,\ell} + Ax_{i,\ell+1} \ .$$

We first state a simple upper bound on the gradients.

Lemma 6.14.1. *Letting $\rho = nLd\log m$, given fixed vectors $\{\mathsf{loss}_{i,\ell} \in \mathbb{R}^d\}_{i\in[n],\ell\in[L]\setminus\{1\}}$, with probability at least $1 - e^{-\Omega(m/L^2)}$ over W, A, B, we have*

$$\|\widehat{\nabla} f(W)\|_F \le O(nL^6\sqrt{m}) \cdot \max_{i,\ell}\{\|\,\mathsf{loss}_{i,\ell}\,\|\} \quad \text{and}$$
$$\forall i \in [n]\colon \|\widehat{\nabla} f_i(W)\|_F \le O(L^6\sqrt{m}) \cdot \max_{i,\ell}\{\|\,\mathsf{loss}_{i,\ell}\,\|\}\ .$$

With probability at least $1 - e^{-\Omega(\rho^2)}$, we have for every $k \in [m]$:

$$\|\widehat{\nabla}_k f(W)\|_2 \le O(n\rho L^3) \cdot \max_{i,\ell}\{\|\,\mathsf{loss}_{i,\ell}\,\|\}\ .$$

Proof of Lemma 6.14.1. For each i, a, ℓ, we can calculate that

$$\begin{aligned}\left\|D_{i,\ell+1}(\mathsf{Back}_{i,\ell+1\to a}^\top \cdot \mathsf{loss}_{i,a}) \cdot h_{i,\ell}^\top\right\|_F &= \left\|D_{i,\ell+1}(\mathsf{Back}_{i,\ell+1\to a}^\top \cdot \mathsf{loss}_{i,a})\right\|_2 \cdot \|h_{i,\ell}^\top\|_2 \\ &\le \|\mathsf{Back}_{i,\ell+1\to a}\|_2 \cdot \|\,\mathsf{loss}_{i,a}\,\|_2 \cdot \|h_{i,\ell}^\top\|_2 \\ &\overset{①}{\le} O(\sqrt{m}) \cdot O(L^3) \cdot O(L) \cdot \|\,\mathsf{loss}_{i,a}\,\|_2\ .\end{aligned}$$

where inequality ① uses Lemma 6.11.8, Lemma 6.11.1, and $\|B\|_2 \le O(\sqrt{m})$ with high probability. Applying triangle inequality and using the definition of fake gradient (see Definition 6.14.1), we finish the proof of the first statement.

As for the second statement, we replace the use of Lemma 6.11.8 with Corollary 6.11.16.

□

The rest of this section is devoted to proving a (much more involved) lower bound on this fake gradient.

Theorem 6.14.2 (gradient lower bound at random init, restated)**.** *Letting* $\rho = nLd\log m$*, given fixed vectors* $\{\mathsf{loss}_{i,\ell} \in \mathbb{R}^d\}_{i\in[n],\ell\in[L]\setminus\{1\}}$*, with probability at least* $1-e^{-\Omega(\rho^2)}$ *over* W, A, B*, we have*

$$\|\widehat{\nabla} f(W)\|_F^2 \geq \Omega\left(\frac{\delta}{\rho^{14}}\right) \times m \times \max_{i,\ell}\{\|\,\mathsf{loss}_{i,\ell}\,\|^2\} \ .$$

In the rest of this section, we first present a elegant way to decompose the randomness in Section 6.14.1 (motivated by smooth analysis [ST04]). We give a lower bound on the *expected* fake gradient in Section 6.14.2. We then calculate the stability of fake gradient against rank-one perturbations in Section 6.14.3. Finally, in Section 6.14.4, we apply our extended McDiarmid's inequality to prove Theorem 6.14.2.

6.14.1 Randomness Decomposition

We introduce a parameter θ as follows:

Definition 6.14.2. Choose any

$$\boxed{\theta \in \left[\rho^4 \cdot \beta_-, \rho^{-3} \cdot \beta_+\right]}$$

where the notions β_- and β_+ come from Definition 6.13.1. We have $\theta \le \frac{\delta}{\rho}$.

In the next lemma, we introduce a way to decompose the randomness of W into $W = W_2 + W'$ for two corollated random matrices W_2 and W'. We will make sure that the entries of W_2 are also i.i.d. from $N(0, \frac{2}{m})$. This definition requires us to choose a specific pair $(i^*, \ell^*) \in [n] \times [L]$.

Lemma 6.14.3. *Suppose W and A follow from random initialization. Given any fixed $i^* \in [n]$, $\ell^* \in [L]$, and parameter $\theta \in \left(0, \frac{1}{2\rho}\right]$, letting random vector $v_{i^*,\ell^*} \in \mathbb{R}^m$ be defined as*

$$v_{i^*,\ell^*} \stackrel{\text{def}}{=} \frac{(I - U_{\ell^*-1}U_{\ell^*-1}^\top)h_{i^*,\ell^*}}{\|(I - U_{\ell^*-1}U_{\ell^*-1}^\top)h_{i^*,\ell^*}\|_2} \ ,$$

then we can write $W = W_2 + W'$ for two random matrices $W_2 \in \mathbb{R}^{m\times m}$ and $W' \in \mathbb{R}^{m\times m}$ where

(a) entries of W_2 are i.i.d. drawn from $\mathcal{N}(0, \frac{2}{m})$ (so W_2 is in the same distribution as W);

(b) W' is correlated with W_2 and can be written as $W' = uv_{i^*,\ell^*}^\top$ for some $u \in \mathbb{R}^m$;

(c) With probability at least $1 - e^{-\Omega(\rho^2)}$ over the randomness of W_2:

$$\forall k\colon \quad \Pr_{W'}\left[u_k = (W'v_{i^*,\ell^*})_k \ge \frac{\theta}{2\sqrt{m}} \;\middle|\; W_2\right] \ge \frac{1}{4} \quad \text{and}$$
$$\Pr_{W'}\left[u_k = (W'v_{i^*,\ell^*})_k \le -\frac{\theta}{2\sqrt{m}} \;\middle|\; W_2\right] \ge \frac{1}{4}$$

(d) With probability at least $1 - e^{-\Omega(\rho^2)}$ over the randomness of W_2:

$$\Pr_{W'}\left[\|u\|_\infty = \|W' v_{i^*,\ell^*}\|_\infty \le \frac{3\theta\rho}{2\sqrt{m}} \;\middle|\; W_2\right] \ge 1 - e^{-\Omega(\rho^2)} \ .$$

Proof. Since v_{i^*,ℓ^*} only depends on the randomness of WU_{ℓ^*-1}, we have $Wv_{i^*,\ell^*} \sim \mathcal{N}(0, \frac{2I}{m})$. As a result, letting $g_1, g_2 \in \mathbb{R}^m$ be two independent random Gaussian vectors sampled from $\mathcal{N}(0, \frac{2I}{m})$, we can couple them to make sure

$$Wv_{i^*,\ell^*} = \sqrt{1-\theta^2} g_1 + \theta g_2 \ .$$

We define $W_2, W' \in \mathbb{R}^{n\times n}$ as follows

$$\begin{aligned} W_2 &\stackrel{\text{def}}{=} WU_{\ell^*-1}U_{\ell^*-1}^\top + g_1 v_{i^*,\ell^*}^\top + W(I - U_{\ell^*-1}U_{\ell^*-1}^\top)(I - v_{i^*,\ell^*}v_{i^*,\ell^*}^\top) \\ W' &\stackrel{\text{def}}{=} uv_{i^*,\ell^*}^\top \quad \text{where} \quad u \stackrel{\text{def}}{=} \theta g_2 - (1-\sqrt{1-\theta^2})g_1 \end{aligned}$$

- It is easy to verify that

$$\begin{aligned} W_2 + W' &= W(I - v_{i^*,\ell^*}v_{i^*,\ell^*}^\top) + g_1 v_{i^*,\ell^*}^\top + (\theta g_2 - (1-\sqrt{1-\theta^2})g_1)v_{i^*,\ell^*}^\top \\ &= W(I - v_{i^*,\ell^*}v_{i^*,\ell^*}^\top) + (\theta g_2 + \sqrt{1-\theta^2} g_1)v_{i^*,\ell^*}^\top \\ &= W(I - v_{i^*,\ell^*}v_{i^*,\ell^*}^\top) + Wv_{i^*,\ell^*}v_{i^*,\ell^*}^\top \\ &= W. \end{aligned}$$

- We verify that the entries of W_2 are i.i.d. Gaussian in two steps.

  On one hand, let $U = [\widehat{h}_1, \ldots, \widehat{h}_m] \in \mathbb{R}^{m\times m}$ be an arbitrary orthonormal matrix where its first nL columns are identical to $U_L \in \mathbb{R}^{m\times nL}$. For each $k \in [nL]$, although $\widehat{h}_k$ may depend on the randomness of $W = W_2 + W'$, it can only depend on $W[\widehat{h}_1, \ldots, \widehat{h}_{k-1}]$.

Therefore, conditioning on the randomness of $W_2[\widehat{h}_1, \ldots, \widehat{h}_{k-1}]$, we have that $W_2\widehat{h}_k$ is still $\mathcal{N}(0, \frac{2I}{m})$. This shows that all entries of W_2U still follow from $N(0, \frac{2}{m})$.

On the other hand, let $V = W_2U = [v_1, \ldots, v_m] \in \mathbb{R}^{m\times m}$, then we have $v_k = W_2\widehat{h}_k$ is independent of $\widehat{h}_k$ as argued above. Since each $v_k \sim N(0, \frac{2I}{m})$, we have that all entries of $VU^\top = \sum_k v_k\widehat{k}_k^\top$ are i.i.d. from $N(0, \frac{2}{m})$. Since $W_2 = VU^\top$ this concludes the proof.

- By standard Gaussian tail bound, we have with probability at least $1 - e^{-\Omega(\rho^2)}$, it satisfies $\|g_1\|_\infty \le \frac{\rho}{\sqrt{m}}$ and therefore $\|(1-\sqrt{1-\theta^2})g_1\|_\infty \le \frac{\theta^2\rho}{\sqrt{m}}$. If this happens, for every $k \in [m]$, we have (see for instance Fact 6.10.12) that $\Pr_{g_2}[(\theta g_2)_k > \frac{\theta}{\sqrt{m}}] \ge \frac{1}{4}$ and $\Pr_{g_2}[(\theta g_2)_k < -\frac{\theta}{\sqrt{m}}] \ge \frac{1}{4}$. Recalling

$$u = (\theta g_2 - (1-\sqrt{1-\theta^2})g_1 \ .$$

and using the fact that $\theta \le \frac{1}{2\rho}$, we conclude that

$$\Pr_{W'}\left[u_k > \frac{\theta}{2\sqrt{m}} \,\middle|\, W_2, A\right] \ge \frac{1}{4} \quad \text{and} \quad \Pr_{W'}\left[u_k < -\frac{\theta}{2\sqrt{m}} \,\middle|\, W_2, A\right] \ge \frac{1}{4} \ .$$

- Again by Gaussian tail bound, we have with probability at least $1 - e^{-\Omega(\rho^2)}$, it satisfies $\|g_1\|_\infty, \|g_2\|_\infty \le \frac{\rho}{\sqrt{m}}$. Therefore, $\|(1-\sqrt{1-\theta^2})g_1\|_\infty \le \frac{\theta^2\rho}{\sqrt{m}}$ and using our assumption on θ, we have

$$\|u\|_\infty = \left\|(\theta g_2 - (1-\sqrt{1-\theta^2})g_1\right\|_\infty \le \frac{3\theta\rho}{2\sqrt{m}} \ .$$

□

Lemma 6.14.4. *Given any fixed $i^* \in [n]$, $\ell^* \in [L]$, and parameter $\theta \in \left(0, \frac{1}{2\rho}\right]$, and suppose suppose W and A are at random initialization and $W = W_2 + W'$ where $W' = uv_{i^*,\ell^*}^\top$ (following Lemma 6.14.3).*

Now, we introduce another two ways of decomposing randomness based on this definition.

(a) Given fixed $k \in [m]$, we can write $W = W_0 + W'_k$ where $W'_k = u_k v_{i^*,\ell^*}^\top$ for $u_k = (0, \ldots, 0, u_k, 0, \ldots, 0)$ that is non-zero only at the k-th coordinate. We again have the entries of W_0 are i.i.d. from $N(0, \frac{2}{m})$.

(b) Given fixed $N \subseteq [m]$, we can write $W = W_1 + W'_N$ where $W'_N = u_N v_{i^*,\ell^*}^\top$ for u_N being the projection of u onto coordinates in N (so $\|u\|_0 = |N|$). We again have the entries of W_0 are i.i.d. from $N(0, \frac{2}{m})$.

(c) Given fixed $N \subseteq [m]$, we can write $W = W_2 + W'_N + W'_{-N}$ where $W'_N = u_N v_{i^*,\ell^*}^\top$ and $W'_{-N} = u_{-N} v_{i^*,\ell^*}^\top$. Here, u_{-N} is the projection of u onto coordinates in $[m] \setminus N$.

Proof. The proofs are analogous to Lemma 6.14.3. □

6.14.2 Gradient Lower Bound in Expectation

The goal of this subsection is to prove Lemma 6.14.5 and then translate it into our Core Lemma A (see Lemma 6.14.6).

Lemma 6.14.5 (c.f. (6.18)). *Let $\rho = nLd\log m$ and $\varrho = nLd\delta^{-1}\log(m/\epsilon)$. Fix integer $N \in [\rho^6/\beta_-, \varrho^{100}]$, θ from Definition 6.14.2, fix vectors $\{\mathsf{loss}_{i,\ell} \in \mathbb{R}^d\}_{i\in[n],\ell\in[L]\setminus\{1\}}$, let $(i^*,\ell^*) = \mathrm{argmin}_{i,\ell}\ \{\|\,\mathsf{loss}_{i,\ell}\,\|\}$. Suppose W_1, A, B are at random initialization, and suppose W'_N is defined according to Lemma 6.14.4b so that $W = W_1 + W'_N$ is also at random initialization. With probability at least $1 - e^{-\Omega(\rho^2)}$ over the randomness of W_1, A, B, we have*

$$\left|\left\{k \in N \;\middle|\; \mathop{\mathbb{E}}_{W'_N}\Big[\|\widehat{\nabla}_k f(W_1+W'_N)\|_2\Big] \ge \Omega\Big(\frac{\|\,\mathsf{loss}_{i^*,\ell^*}\,\|_2}{nL\sqrt{d}}\Big)\right\}\right| \ge \frac{\beta_-|N|}{100L}$$

6.14.2.1 Proof of Lemma 6.14.5

Throughout this proof we denote by $\widetilde{W} = W_1$. We introduce some (old and new) notations:

$$\widetilde{\mathsf{Back}}_{i,\ell\to\ell} = B, \qquad \mathsf{Back}_{i,\ell\to\ell} = B$$
$$\widetilde{\mathsf{Back}}_{i,\ell\to a} = B\widetilde{D}_{i,a}\widetilde{W}\cdots\widetilde{D}_{i,\ell+1}\widetilde{W}, \qquad \mathsf{Back}_{i,\ell\to a} = BD_{i,a}W\cdots D_{i,\ell+1}W,$$

and we let $\mathsf{Back}'_{i,\ell\to a} = \mathsf{Back}_{i,\ell\to a} - \widetilde{\mathsf{Back}}_{i,\ell\to a}$. For each $i\in[n]$, $\ell\in[L]$, we define

$$\widetilde{g}_{i,\ell} = Ax_{i,\ell} + \widetilde{W}\widetilde{D}_{i,\ell-1}\widetilde{g}_{i,\ell-1} \qquad g_{i,\ell} = Ax_{i,\ell} + WD_{i,\ell-1}g_{i,\ell-1} \qquad g'_{i,\ell} = g_{i,\ell} - \widetilde{g}_{i,\ell}$$
$$\widetilde{h}_{i,\ell} = \widetilde{D}_{i,\ell}(Ax_{i,\ell} + \widetilde{W}\widetilde{h}_{i,\ell-1}) \qquad h_{i,\ell} = D_{i,\ell}(Ax_{i,\ell} + Wh_{i,\ell-1}) \qquad h'_{i,\ell} = h_{i,\ell} - \widetilde{h}_{i,\ell}$$
$$\widetilde{u}_{i,\ell} = \sum_{a=\ell}^{L} \widetilde{\mathsf{Back}}_{\ell\to a}^\top \mathsf{loss}_{i,a} \qquad u_{i,\ell} = \sum_{a=\ell}^{L} \mathsf{Back}_{\ell\to a}^\top \mathsf{loss}_{i,a} \qquad u'_{i,\ell} = u_{i,\ell} - \widetilde{u}_{i,\ell}$$

We also let

$$v_{i^*,\ell^*} = \frac{(I - U_{\ell^*-1}U_{\ell^*-1}^\top)\widetilde{h}_{i^*,\ell^*}}{\|(I - U_{\ell^*-1}U_{\ell^*-1}^\top)\widetilde{h}_{i^*,\ell^*}\|_2}$$

and recall from Lemma 6.14.3d, we have

$$W'_N = yv_{i^*,\ell^*}^\top \text{ for some } y \in \mathbb{R}^m \text{ with } \|y\|_0 = |N| \text{ and } \|y\|_\infty \le \frac{\tau_0}{\sqrt{m}} \stackrel{\text{def}}{=} \frac{3\theta\rho}{2\sqrt{m}} \in \left[\frac{\varrho^{-20}}{\sqrt{m}}, \frac{\varrho^{20}}{\sqrt{m}}\right]$$

Proof of Lemma 6.14.5. We let $N_1 = N$, apply Lemma 6.13.1 to obtain $N_4 \subseteq N_1$ with $|N_4| \ge \frac{\beta_-|N_1|}{64L}$, and apply Lemma 6.13.5 to obtain $N_5 \subseteq N_4$. According to the statements of these lemmas, we know that with probability at least $1 - e^{-\Omega(\rho^2)}$, the random choice of W_1, A, B will satisfy $|N_5| \ge \frac{\beta_-|N_1|}{100L}$.

Fixing such $k \in N_5$, we use $u_{i,\ell}$ to denote $(u_{i,\ell})_k$ and $g_{i,\ell}$ to denote $(g_{i,\ell})_k$ for notational simplicity. We can rewrite $\widehat{\nabla}_k f(W)$ as follows and apply triangle inequality:

$$\begin{aligned}
\|\widehat{\nabla}_k f(W)\| &= \left\|u_{i^*,\ell^*} \cdot h_{i^*,\ell^*} \cdot \mathbf{1}_{g_{i^*,\ell^*+1}\ge 0} + \textstyle\sum_{(i,\ell)\neq(i^*,\ell^*)} u_{i,\ell} \cdot h_{i,\ell} \cdot \mathbf{1}_{g_{i,\ell+1}\ge 0}\right\| \\
&\ge \underbrace{\left\|\widetilde{u}_{i^*,\ell^*} \cdot \widetilde{h}_{i^*,\ell^*} \cdot \mathbf{1}_{g_{i^*,\ell^*+1}\ge 0} + \textstyle\sum_{(i,\ell)\neq(i^*,\ell^*)} \widetilde{u}_{i,\ell} \cdot \widetilde{h}_{i,\ell} \cdot \mathbf{1}_{g_{i,\ell+1}\ge 0}\right\|_2}_{\clubsuit} \\
&\quad - \underbrace{\textstyle\sum_{i,\ell} \left\|u'_{i,\ell} \cdot \widetilde{h}_{i,\ell} + \widetilde{u}_{i,\ell} \cdot h'_{i,\ell} + u'_{i,\ell} \cdot h'_{i,\ell}\right\|_2}_{\spadesuit}
\end{aligned} \tag{6.48}$$

where recall $g_{i,\ell+1} = \langle W_k, h_{i,\ell}\rangle + \langle A_k, x_{i,\ell+1}\rangle$.

Step 1 To bound the ♠ term on the RHS of (6.48), we consider the following bounds:

1. Using Lemma 6.12.10, we have with probability at least $1 - e^{-\Omega(\rho^2)}$, for all i, ℓ, a

$$|(\widetilde{\mathsf{Back}}_{i,\ell\to a} \cdot \mathsf{loss}_{i,a})_k - (\mathsf{Back}_{i,\ell\to a} \cdot \mathsf{loss}_{i,a})_k| \le O\Big(\frac{\rho^3\tau_0^{1/3}N^{1/6}}{m^{1/6}}\Big) \cdot \|\,\mathsf{loss}_{i,a}\,\| \ .$$

This implies, by triangle inequality and the fact that $\| \mathsf{loss}_{i,a} \| \le \| \mathsf{loss}_{i^*,a^*} \|$,

$$\left|u'_{i,\ell}\right| = |u_{i,\ell} - \widetilde{u}_{i,\ell}| \le O\big(\frac{\rho^4 \tau_0^{1/3} N^{1/6}}{m^{1/6}}\big) \cdot \| \mathsf{loss}_{i^*,a^*} \| \ .$$

2. Using Corollary 6.11.16, we have with probability at least $1 - e^{-\Omega(\rho^2)}$, for all i, ℓ, a

$$|(\widetilde{\mathsf{Back}}_{i,\ell\to a} \cdot \mathsf{loss}_{i,a})_k| \le \rho \| \mathsf{loss}_{i,a} \| \ .$$

This implies, by triangle inequality and the fact that $\| \mathsf{loss}_{i,a} \| \le \| \mathsf{loss}_{i^*,a^*} \|$,

$$|\widetilde{u}_{i,\ell}| \le \rho^2 \| \mathsf{loss}_{i^*,a^*} \| \ .$$

3. Using Lemma 6.11.1 and Lemma 6.12.1a we have

$$\|\widetilde{h}_{i,\ell}\|_2 \le O(L) \quad \text{and} \quad \|h'_{i,\ell}\|_2 \le O(L^6 \tau_0 \sqrt{N})/m^{1/2}$$

All together, they imply

$$\sum_{i,\ell} \left\| u'_{i,\ell} \cdot \widetilde{h}_{i,\ell} + \widetilde{u}_{i,\ell} \cdot h'_{i,\ell} + u'_{i,\ell} \cdot h'_{i,\ell} \right\|_2 \le O\big(\tfrac{\rho^6 \tau_0^{1/3} N^{1/6}}{m^{1/6}}\big) \cdot \| \mathsf{loss}_{i^*,a^*} \| \ . \tag{6.49}$$

Step 2 We next turn to the ♣ term on the RHS of (6.48). We divide into three cases to analyze the sign change of $g_{i,\ell+1}$ for all possible i and ℓ in the above formula (under the randomness of W'_N). Before we do so, first note that Lemma 6.12.9 implies

$$|(W h'_{i,\ell})_k| = |(W_1 + W'_N) h'_{i,\ell}| \le O\Big(\frac{\rho^8 N^{2/3} \tau_0^{5/6}}{m^{2/3}}\Big) \tag{6.50}$$

1. Consider $g_{i,\ell+1}$ for the case of $i = i^*, \ell = \ell^*$. We want to show that the sign of $g_{i,\ell+1}$ changes (at least with constant probability) with respect to the randomness of W'_N.

One can easily check that:[20]

$$g_{i,\ell+1} = \langle A_k, x_{i,\ell+1}\rangle + (W_1\widetilde{h}_{i,\ell})_k + (Wh'_{i,\ell})_k + (W'_N v_{i,\ell})_k \cdot \|(I - U_{\ell-1}U_{\ell-1}^\top)h_{i,\ell}\|_2. \quad (6.51)$$

By Lemma 6.13.1, we have $|\langle A_k, x_{i,\ell}\rangle + (W_1\widetilde{h}_{i,\ell})_k| \le \frac{\beta_-}{\sqrt{m}}$ Thus, putting this and (6.50) into (6.51), we can write

$$g_{i,\ell+1} = \Xi + (W'_N v_{i,\ell})_k \cdot \|(I - U_{\ell-1}U_{\ell-1}^\top)\widetilde{h}_{i,\ell}\|_2$$

for some value Ξ with $|\Xi| \le \frac{2\beta_-}{\sqrt{m}}$. By Lemma 6.11.4, we have $\|(I - U_{\ell-1}U_{\ell-1}^\top)\widetilde{h}_{i,\ell}\|_2 \ge \Omega\big(\frac{1}{L^2\log^3 m}\big)$. By Lemma 6.14.3c, we have $(W'_N v_{i,\ell})_k$ shall be larger than $\frac{\theta}{2\sqrt{m}}$ and smaller than $-\frac{\theta}{2\sqrt{m}}$ each with probability at least $1/4$. Since by our choice of θ (see Definition 6.14.2),

$$\frac{\theta}{2\sqrt{m}} \cdot \Omega\big(\frac{1}{L^2\log^3 m}\big) \gg \frac{2\beta_-}{\sqrt{m}} \ge |\Xi|$$

This concludes that, with probability at least $1 - e^{-\Omega(\rho^2)}$ over W_1 and A, it satisfies

$$\Pr_{W'_N}[g_{i,\ell+1} > 0 \mid W_1, A] \ge \frac{1}{5} \quad \text{and} \quad \Pr_{W'_N}[g_{i,\ell+1} < 0 \mid W_1, A] \ge \frac{1}{5}$$

[20] We can write $(g_{i,\ell+1})_k = \langle A_k, x_{i,\ell+1}\rangle + (W(\widetilde{h}_{i,\ell} + h'_{i,\ell}))_k = \langle A_k, x_{i,\ell+1}\rangle + (W\widetilde{h}_{i,\ell})_k + (Wh'_{i,\ell})_k$. For the second term in the above equation, we have

$$\begin{aligned}
(W\widetilde{h}_{i,\ell})_k &= \left(WU_{\ell-1}U_{\ell-1}^\top\widetilde{h}_{i,\ell} + \frac{W(I - U_{\ell-1}U_{\ell-1}^\top)\widetilde{h}_{i,\ell}}{\|(I - U_{\ell-1}U_{\ell-1}^\top)\widetilde{h}_{i,\ell}\|_2}\|(I - U_{\ell-1}U_{\ell-1}^\top)\widetilde{h}_{i,\ell}\|_2\right)_k \\
&\overset{①}{=} \left(WU_{\ell-1}U_{\ell-1}^\top\widetilde{h}_{i,\ell} + Wv_{i,\ell}\|(I - U_{\ell-1}U_{\ell-1}^\top\widetilde{h}_{i,\ell})\|_2\right)_k \\
&\overset{②}{=} \left(W_1U_{\ell-1}U_{\ell-1}^\top\widetilde{h}_{i,\ell} + Wv_{i,\ell}\|(I - U_{\ell-1}U_{\ell-1}^\top\widetilde{h}_{i,\ell})\|_2\right)_k \\
&\overset{③}{=} \left(W_1U_{\ell-1}U_{\ell-1}^\top\widetilde{h}_{i,\ell} + W_1v_{i,\ell}\|(I - U_{\ell-1}U_{\ell-1}^\top\widetilde{h}_{i,\ell})\|_2\right)_k + \left(W'_N v_{i,\ell} \cdot \|(I - U_{\ell-1}U_{\ell-1}^\top\widetilde{h}_{i,\ell})\|_2\right)_k \\
&= (W_1\widetilde{h}_{i,\ell})_k + (W'_N v_{i,\ell})_k \cdot \|(I - U_{\ell-1}U_{\ell-1}^\top\widetilde{h}_{i,\ell})\|_2
\end{aligned}$$

where ① follows by definition of $v_{i,\ell}$, ② follows by definition of W_1, ③ follows by $W = W_1 + W'_N$.

2. Consider $g_{i,\ell+1}$ for the case of $i \in [n] \backslash \{i^*\}, \ell = \ell^*$ or $i \in [n], \ell > \ell^*$. We want to show the sign of $g_{i,\ell+1}$ is fixed, meaning with high probability independent of the choice of W'_N. One can easily check that:

$$g_{i,\ell+1} = \langle A_k, x_{i,\ell+1} \rangle + (W_1 \widetilde{h}_{i,\ell})_k + (W h'_{i,\ell})_k + (W'_N \widetilde{h}_{i,\ell})_k \ . \tag{6.52}$$

By Lemma 6.13.1, we have $|\langle A_k, x_{i,\ell} \rangle + (W_1 \widetilde{h}_{i,\ell})_k| \geq \frac{\beta_+}{\sqrt{m}}$. Thus, putting this and (6.50) into (6.53), we can write

$$g_{i,\ell+1} = \Xi + (W'_N \widetilde{h}_{i,\ell})_k = \Xi + (W'_N v_{i^*,\ell^*})_k \cdot \langle v_{i^*,\ell^*}, \widetilde{h}_{i,\ell} \rangle$$

for some value Ξ with $|\Xi| \geq \frac{\beta_+}{2\sqrt{m}}$. By Lemma 6.11.1, we have $|\langle v_{i^*,\ell^*}, \widetilde{h}_{i,\ell} \rangle| \leq \|\widetilde{h}_{i,\ell}\|_2 \leq O(L)$. By Lemma 6.14.3d, we have $|(W'_N v_{i^*,\ell^*})_k| \leq \frac{2\theta\rho}{\sqrt{m}}$ and by our choice of θ (see Definition 6.14.2),

$$\frac{2\theta\rho}{\sqrt{m}} \cdot O(L) \ll \frac{2\beta_+}{\sqrt{m}} \leq |\Xi|$$

This concludes that, with probability at least $1 - e^{-\Omega(\rho^2)}$ over W_1 and A, it satisfies

$$\exists s \in \{-1, 1\} \colon \quad \Pr_{W'_N} [\operatorname{sgn}(g_{i,\ell+1}) = s \mid W_1, A] \geq 1 - e^{-\Omega(\rho^2)} \ .$$

3. Consider $g_{i,\ell+1}$ for the case of $i \in [n]$ and $\ell < \ell^*$. We want to show such $g_{i,\ell+1}$ is fixed, meaning independent of W'_N. One can easily check that:

$$g_{i,\ell+1} = \langle A_k, x_{i,\ell+1} \rangle + (W_1 \widetilde{h}_{i,\ell})_k + (W h'_{i,\ell})_k + (W'_N \widetilde{h}_{i,\ell})_k \ . \tag{6.53}$$

Now, since $\ell < \ell^*$, we know $h_{i,\ell} = \widetilde{h}_{i,\ell}$ because both of which only depend on the randomness of $W U_{\ell^*-1} = W_1 U_{\ell^*-1}$ and A; this implies $h'_{i,\ell} = 0$. Also, since $W'_N = (W'_N v_{i^*,\ell^*}) v_{i^*,\ell^*}^\top$ but $\langle v_{i^*,\ell^*}, \widetilde{h}_{i,\ell} \rangle = 0$, we know that

$$g_{i,\ell+1} = \langle A_k, x_{i,\ell+1} \rangle + (W_1 \widetilde{h}_{i,\ell})_k \ .$$

This is a fixed value, independent of W'_N.

Therefore, combining the three cases above, we conclude that with at least constant probability, the sign of g_{i^*,ℓ^*+1} can possibly change, but the sign of $g_{i,\ell+1}$ will not change for any other $(i,\ell) \neq (i^*,\ell^*)$. This means,

$$\begin{aligned}\mathbb{E}[\clubsuit] &= \mathbb{E}\left[\left\|\widetilde{u}_{i^*,\ell^*} \cdot \widetilde{h}_{i^*,\ell^*} \cdot \mathbf{1}_{g_{i^*,\ell^*+1}\geq 0} + \textstyle\sum_{(i,\ell)\neq(i^*,\ell^*)} \widetilde{u}_{i,\ell} \cdot \widetilde{h}_{i,\ell} \cdot \mathbf{1}_{g_{i,\ell+1}\geq 0}\right\|_2\right] \\ &\geq \Omega(\|\widetilde{u}_{i^*,\ell^*}\widetilde{h}_{i^*,\ell^*}\|_2) = \Omega\big(|\widetilde{u}_{i^*,\ell^*}| \cdot \|\widetilde{h}_{i^*,\ell^*}\|_2\big)\end{aligned}$$

Finally, for each $k \in N_5$, owing to Lemma 6.13.5 and Lemma 6.11.1, we have

$$|\widetilde{u}_{i^*,\ell^*}|_2 = \left|\sum_{a=\ell^*}^{L} (\widetilde{\mathsf{Back}}_{i^*,\ell^*\to a}^{\top} \mathsf{loss}_{i^*,a})_k\right| \geq \frac{\|\,\mathsf{loss}_{i^*,\ell^*}\,\|_2}{6\sqrt{d}nL} \quad \text{and} \quad \|\widetilde{h}_{i^*,\ell^*}\|_2 \geq \frac{1}{2}\ .$$

This means,

$$\mathbb{E}[\clubsuit] \geq \Omega\left(\frac{\|\,\mathsf{loss}_{i^*,\ell^*}\,\|_2}{\sqrt{d}nL}\right)\ . \tag{6.54}$$

Putting (6.49) and (6.54) back to (6.48), we conclude that

$$\forall k \in N_5 \colon \quad \mathbb{E}[\|\widehat{\nabla}_k f(W)\|] \geq \Omega\left(\frac{\|\,\mathsf{loss}_{i^*,\ell^*}\,\|_2}{\sqrt{d}nL}\right)\ .$$

□

6.14.2.2 Core Lemma A

In Lemma 6.14.5, we split W into $W = W_1 + W'_N$ for a fixed $N \subseteq [m]$. In this subsection, we split W into three parts $W = W_2 + W'_N + W'_{-N}$ following Lemma 6.14.4c. Our purpose is to rewrite Lemma 6.14.5 into the following variant:

Lemma 6.14.6 (Core Lemma A). *Let $\rho = nLd\log m$ and $\varrho = nLd\delta^{-1}\log(m/\epsilon)$. Fix vectors $\{\mathsf{loss}_{i,\ell} \in \mathbb{R}^d\}_{i\in[n],\ell\in[L]\setminus\{1\}}$, integer $|N| \in [\rho^6/\beta_-, \varrho^{100}]$, and parameter $t = m/|N|$. Let $N_1, \ldots, N_t$ be i.i.d. random subsets of $[m]$ with cardinality $|N_i| = |N|$. Then,*

- *with probability $\geq 1 - e^{-\Omega(\rho^2)}$ over the randomness of W_2, A, B and $N_1, \cdots N_t$, the following holds:*
  - *for every $N \in \{N_1, N_2, \cdots, N_t\}$, the following holds:*
    - *with probability at $1 - e^{-\Omega(\rho^2)}$ over the randomness of W'_{-N}, the following holds:*

$$\sum_{k\in N} \mathop{\mathbb{E}}_{W'_N}\left[\|\widehat{\nabla}_k f(W_2 + W'_N + W'_{-N})\|_2^2\right] \geq \Omega\left(\frac{\beta_-|N|}{\rho^2}\right) \cdot \max_{i,\ell} \|\,\mathsf{loss}_{i,\ell}\,\|_2^2.$$

Proof of Lemma 6.14.6. We denote by $(i^*, \ell^*) = \mathrm{argmax}_{i,\ell} \|\,\mathsf{loss}_{i,\ell}\,\|_2$. We first note that Lemma 6.14.5 directly implies

- with probability $\geq 1 - e^{-\Omega(\rho^2)}$ over the randomness of W_2, W'_{-N}, A, B, N, the following holds:

$$\left|\left\{k \in N \;\middle|\; \mathop{\mathbb{E}}_{W'_N}\left[\|\widehat{\nabla}_k f(W_2 + W'_{-N} + W'_N)\|_2\right] \geq \Omega\Big(\frac{\|\,\mathsf{loss}_{i^*,\ell^*}\,\|_2}{nL\sqrt{d}}\Big)\right\}\right| \geq \frac{\beta_-}{100L}|N|$$

or putting it in another way, using our choice of q,

- with probability $\geq 1 - e^{-\Omega(\rho^2)}$ over the randomness of W_2, W'_{-N}, A, B, N, the following holds ($q = \frac{\beta_-|N|}{c\rho^2}$ for some sufficiently large constant):

$$\sum_{k\in N} \mathop{\mathbb{E}}_{W'_N}\left[\|\widehat{\nabla}_k f(W_2 + W'_{-N} + W_N)\|_2^2\right] \geq q \cdot \|\,\mathsf{loss}_{i^*,\ell^*}\,\|_2^2.$$

Applying simple tricks to switch the ordering of randomness (see Fact 6.10.2), we have

- with probability $\geq 1 - e^{-\Omega(\rho^2)}$ over the randomness of W_2, A, B, the following holds:

- with probability $\geq 1 - e^{-\Omega(\rho^2)}$ over the randomness of N, the following holds:
  - $*$ with probability $\geq 1 - e^{-\Omega(\rho^2)}$ over randomness of W'_{-N}, the following holds:

$$\sum_{k \in N} \mathop{\mathbb{E}}_{W'_N} [\|\widehat{\nabla}_k f(W_2 + W'_N + W'_{-N})\|_2^2] \geq q \cdot \| \mathsf{loss}_{i^*,\ell^*} \|_2^2.$$

Repeating the choice of N for t times, and using union bound, we have

- with probability $\geq 1 - e^{-\Omega(\rho^2)}$ over the randomness of W_2, A, B, the following holds:
  - with probability $\geq 1 - e^{-\Omega(\rho^2)}$ over the randomness of $N_1, N_2, \cdots, N_t$, the following holds:
    - $*$ for all $N \in \{N_1, N_2, \cdots, N_t\}$, the following holds:
      - $\cdot$ with probability $\geq 1 - e^{-\Omega(\rho^2)}$ over randomness of W'_{-N}, the following holds:

$$\sum_{k \in N} \mathop{\mathbb{E}}_{W'_N} [\|\widehat{\nabla}_k f(W_2 + W'_N + W'_{-N})\|_2^2] \geq q \cdot \| \mathsf{loss}_{i^*,\ell^*} \|_2^2.$$

Combining the above statement with Fact 6.10.1 (to merge randomness), we conclude the proof. □

6.14.3 Gradient Stability

The goal of this subsection is to prove Lemma 6.14.7 and then translate it into our Core Lemma B (see Lemma 6.14.9).

Lemma 6.14.7 (c.f. (6.19))**.** *Fix parameter* $\rho = nLd\log m$*, vectors* $\{\mathsf{loss}_{i,\ell} \in \mathbb{R}^d\}_{i\in[n],\ell\in[L]\setminus\{1\}}$*,* $(i^*,\ell^*) = \mathrm{argmin}_{i,\ell}\{\|\mathsf{loss}_{i,\ell}\|\}$*, index* $j \in [m]$*. Let* $N \subseteq [m]$ *be a random subset containing* j *of fixed cardinality* $|N|$*, let* W, A, B *be at random initialization, and* W_j'' *be defined as Definition 6.14.4a so that* $W + W_j''$ *is also at random initialization. Then, with probability at least* $1 - e^{-\Omega(\rho^2)}$

$$\sum_{k\in[N]} \left| \|\widehat{\nabla}_k(W)\|_2^2 - \|\widehat{\nabla}_k(W + W_j'')\|_2^2 \right| \le O\left(\rho^8 + \frac{\rho^{11}\theta^{1/3}}{m^{1/6}}|N|\right) \quad . \tag{6.55}$$

6.14.3.1 Proof of Lemma 6.14.7

Throughout this proof we denote by $\widetilde{W} = W$ to emphasize that W is at random initialization. We first show some properties before proving Lemma 6.14.7.

Claim 6.14.8. *Fix parameter* $\rho = nLd\log m$*, vectors* $\{\mathsf{loss}_{i,\ell} \in \mathbb{R}^d\}_{i\in[n],\ell\in[L]\setminus\{1\}}$*,* $(i^*,\ell^*) = \mathrm{argmin}_{i,\ell}\{\|\mathsf{loss}_{i,\ell}\|\}$*, index* $j \in [m]$*. Let* $\widetilde{W}, A, B$ *be at random initialization, and* W_j'' *be defined as Lemma 6.14.4a so that* $W = \widetilde{W} + W_j''$ *is also at random initialization. Then, then with probability* $\ge 1 - e^{-\Omega(\rho^2)}$ *over* $\widetilde{W}, A, B, W_j''$*,*

- $\left|\left\{k \in [m] \;\middle|\; \left|\|\widehat{\nabla}_k f(\widetilde{W} + W_j'')\|_2^2 - \|\widehat{\nabla}_k f(\widetilde{W})\|_2^2\right| \le O\left(\frac{\rho^{11}\theta^{1/3}}{m^{1/6}}\right)\right\}\right| \ge \left(1 - \frac{\rho^5\theta^{2/3}}{m^{1/3}}\right)m$.
- $\left|\|\widehat{\nabla}_k f(\widetilde{W} + W_j'')\|_2^2 - \|\widehat{\nabla}_k f(\widetilde{W})\|_2^2\right| \le O(\rho^6)$ *for every* $k \in [m]$*.*

Proof of Claim 6.14.8. As before, let $\widetilde{D}_{i,\ell}$ denote the diagonal matrix consisting of the indicator function $\mathbb{1}_{(\widetilde{g}_{i,\ell+1})_k \geq 0}$ for its k-th diagonal entry. Let $D_{i,\ell}$ be the same thing but with respect to weight matrix $\widetilde{W} + W_j''$. Let $D''_{i,\ell} = D_{i,\ell} - \widetilde{D}_{i,\ell}$ be the diagonal matrix of sign change.

- By Lemma 6.12.1 (with $\tau_0 = \frac{3\theta\rho}{2}$ owing to Lemma 6.14.3d), we have $\|D''_\ell\|_0 \leq \rho^4\theta^{2/3}m^{2/3}$. Therefore, letting $J \stackrel{\text{def}}{=} [m] \setminus \cup_{i,\ell} \operatorname{supp}(D_{i,\ell})$ be the set of all indices of sign changes, we have

$$|J| \geq m - nL\rho^4\theta^{2/3}m^{2/3} \geq \left(1 - \frac{\rho^5\theta^{2/3}}{m^{1/3}}\right) \cdot m.$$

  Next, for every index $k \in J$, we have

$$\begin{aligned}
&\|\widehat{\nabla}_k f(\widetilde{W} + W_j'') - \widehat{\nabla}_k f(\widetilde{W})\|_2 \\
\stackrel{①}{=}& \left\| \sum_{i=1}^n \sum_{\ell=1}^L (\widetilde{u}_{i,\ell})_k \cdot \widetilde{h}_{i,\ell} \cdot \mathbf{1}_{(\widetilde{g}_{i,\ell+1})_k \geq 0} - \sum_{i=1}^n \sum_{\ell=1}^L (u_{i,\ell})_k \cdot h_{i,\ell} \cdot \mathbf{1}_{(g_{i,\ell+1})_k \geq 0} \right\|_2 \\
\stackrel{②}{=}& \left\| \sum_{i=1}^n \sum_{\ell=1}^L (\widetilde{u}_{i,\ell})_k \cdot \widetilde{h}_{i,\ell} \cdot \mathbf{1}_{(g_{i,\ell+1})_k \geq 0} - \sum_{i=1}^n \sum_{\ell=1}^L (u_{i,\ell})_k \cdot h_{i,\ell} \cdot \mathbf{1}_{(g_{i,\ell+1})_k \geq 0} \right\|_2 \\
\stackrel{③}{\leq}& \sum_{i=1}^n \sum_{\ell=1}^L \|((\widetilde{u}_{i,\ell})_k \cdot \widetilde{h}_{i,\ell} - (u_{i,\ell})_k \cdot h_{i,\ell}) \cdot \mathbf{1}_{(g_{i,\ell+1})_k \geq 0}\|_2 \ .
\end{aligned}$$

  where ① follows by definition, ② follows by our choice of $k \in J$, and ③ follows by triangle inequality. For each i, ℓ, using the same proof as (6.49) (so invoking Lemma 6.12.10, Corollary 6.11.16, Lemma 6.11.1 and Lemma 6.12.1a), we have with probability at least $1 - e^{-\Omega(\rho^2)}$:

$$\|(\widetilde{u}_{i,\ell})_k \cdot \widetilde{h}_{i,\ell} - (u_{i,\ell})_k \cdot h_{i,\ell}\|_2 \leq O\Big(\frac{\rho^6 \tau_0^{1/3}}{m^{1/6}}\Big) \cdot \|\, \mathsf{loss}_{i^*,a^*} \,\| \ . \tag{6.56}$$

which implies

$$\|\widehat{\nabla}_k f(\widetilde{W}+W_j'')-\widehat{\nabla}_k f(\widetilde{W})\|_2 \le O\big(\frac{\rho^6\tau_0^{1/3}}{m^{1/6}}\big)\cdot\|\,\mathsf{loss}_{i^*,a^*}\,\|\ .$$

Combining this with $\|\widehat{\nabla}_k f(\widetilde{W})\|_2 \le O(\rho^4)$ from Lemma 6.14.1, we finish the proof of the first item.

- For every index $k\in[m]$, we have

$$\begin{aligned}
&\|\widehat{\nabla}_k f(\widetilde{W}+W_j'')-\widehat{\nabla}_k f(\widetilde{W})\|_2 \\
=\ &\left\|\sum_{i=1}^n\sum_{\ell=1}^L(\widetilde{u}_{i,\ell})_k\cdot\widetilde{h}_{i,\ell}\cdot\mathbf{1}_{(\widetilde{g}_{i,\ell+1})_k\ge0}-\sum_{i=1}^n\sum_{\ell=1}^L(u_{i,\ell})_k\cdot h_{i,\ell}\cdot\mathbf{1}_{(g_{i,\ell+1})_k\ge0}\right\|_2 \\
\le\ &\sum_{i=1}^n\sum_{\ell=1}^L\|(\widetilde{u}_{i,\ell})_k\cdot\widetilde{h}_{i,\ell}\|_2+\|(u_{i,\ell})_k\cdot h_{i,\ell})\|_2\ .
\end{aligned}$$

Using (6.56) and $\|(\widetilde{u}_{i,\ell})_k\cdot\widetilde{h}_{i,\ell}\|_2\le O(\rho^2 L)$ (see the proof of Lemma 6.14.1), we immediately have the desired bound.

□

Proof of Lemma 6.14.7. Letting $J\stackrel{\text{def}}{=}\left\{k\in[m]\ \middle|\ \left|\|\widehat{\nabla}_k f(\widetilde{W}+W_j'')\|_2^2-\|\widehat{\nabla}_k f(\widetilde{W})\|_2^2\right|\le O\left(\frac{\rho^{11}\theta^{1/3}}{m^{1/6}}\right)\right\}$, we have $|J|\ge\left(1-\frac{\rho^5\theta^{2/3}}{m^{1/3}}\right)m$ according to Claim 6.14.8.

Now, for the indices in N that are randomly sampled from $[m]$, if $|N\cap J|\ge|N|-S$ for a parameter $S=\rho^2$, then we have

$$\sum_{k\in N}\left|\|\widehat{\nabla}_k f(\widetilde{W})\|_2^2-\|\widehat{\nabla}_k f(\widetilde{W}+W_j'')\|_2^2\right|\le|S|\cdot O(\rho^6)+|N|\cdot O\left(\frac{\rho^{11}\theta^{1/3}}{m^{1/6}}\right)\le O\left(\rho^8+\frac{\rho^{11}\theta^{1/3}}{m^{1/6}}|N|\right)\ .$$

Otherwise, if $|N\cap J|\le|N|-S$, this means at least S indices that are chosen from N are outside $J\subseteq[m]$. This happens with probability at most $\left(\frac{\rho^5\theta^{2/3}}{m^{1/3}}\right)^S\le e^{-\Omega(\rho^2)}$. □

6.14.3.2 Core Lemma B

In this subsection, we split W into three parts $W = W_2 + W'_N + W'_{-N}$ following def:random-decomp:N-N. Our purpose is to rewrite Lemma 6.14.7 into the following variant:

Lemma 6.14.9 (Core Lemma B)**.** *Let $\rho = nLd\log m$ and $\varrho = nLd\delta^{-1}\log(m/\epsilon)$. Fix parameter θ from Definition 6.14.2, vectors $\{\mathsf{loss}_{i,\ell} \in \mathbb{R}^d\}_{i\in[n],\ell\in[L]\setminus\{1\}}$, integer $|N| \in [1, \varrho^{100}]$, parameter $t = m/|N|$. Let $N_1, \ldots, N_t$ be i.i.d. random subsets of $[m]$ with cardinality $|N_i| = |N|$. Then, with probability $\geq 1 - e^{-\Omega(\rho^2)}$ over the randomness of W_2, A, B and $N_1, \cdots N_t$, the following holds:*

- *for every $N \in \{N_1, N_2, \cdots, N_t\}$, the following holds:*
  - *with probability at $1 - e^{-\Omega(\rho^2)}$ over the randomness of W'_{-N}, the following holds:*
    - *with probability at least $1 - e^{-\Omega(\rho^2)}$ over the randomness of W'_N,we have :*
      - *for all $j \in N$, and for all $W''_j = u_j v_{i^*,\ell^*}^\top$ satisfying*

$$\|u_j\|_0 = 1, \quad \|u_j\|_\infty \leq \tfrac{3\theta\rho}{\sqrt{m}}, \quad v_{i^*,\ell^*} = \tfrac{(I - U_{\ell^*-1}U_{\ell-1}^\top)h_{i^*,\ell}h_{i^*,\ell^*}}{\|(I - U_{\ell^*-1}U_{\ell^*-1}^\top)h_{i^*,\ell^*}\|_2} \ , \tag{6.57}$$

we have

$$\begin{aligned}&\textstyle\sum_{k\in N}\Big|\|\widehat{\nabla}_k f(W_2 + W'_N + W'_{-N})\|_2^2 - \|\widehat{\nabla}_k f(W_2 + W'_N + W'_{-N} + W''_j)\|_2^2\Big| \\ &\leq O(\rho^8)\cdot \max_{i,\ell}\|\,\mathsf{loss}_{i,\ell}\,\|_2^2 \ .\end{aligned}$$

Proof of Lemma 6.14.9. Without loss of generality we assume $\max_{i,\ell}\|\,\mathsf{loss}_{i,\ell}\,\|_2^2 = 1$ in the proof. We now first rewrite Lemma 6.14.7 as follows (recalling m is sufficiently large so we only need to keep the $O(\rho^8)$ term):

- with probability $\geq 1 - e^{-\Omega(\rho^2)}$ over random $N \subset [m]$, random $j \in N$, random W, A, B, and random W'_j, the following holds:

$$\sum_{k \in [N]} \left| \|\widehat{\nabla}_k(W)\|_2^2 - \|\widehat{\nabla}_k(W + W''_j)\|_2^2 \right| \leq O\left(\rho^8 + \frac{\rho^{11}\theta^{1/3}}{m^{1/6}}|N|\right) \leq O(\rho^8) \ . \tag{6.58}$$

We can split the randomness (see Fact 6.10.1) and derive that

- With probability $\geq 1 - e^{-\Omega(\rho^2)}$ over $N \subseteq [m]$, W, A, B, the following holds:
  - With probability $\geq 1 - e^{-\Omega(\rho^2)}$ over $j \in N$ and W''_j, Eq. (6.58) holds.

Applying standard ϵ-net argument, we derive that

- With probability $\geq 1 - e^{-\Omega(\rho^2)}$ over $N \subseteq [m]$, W, A, B, the following holds:
  - for all $j \in N$, and for all $W''_j = u_j v_{i^*,\ell^*}^\top$ satisfying (6.57), we have Eq. (6.58) holds.

Finally, letting $W = W_2 + W'_N + W'_{-N}$, we have

- With probability $\geq 1 - e^{-\Omega(\rho^2)}$ over the randomness of $N \subseteq [m], W_2, W_N, W'_{-N}, A, B$, the following holds:
  - for all $j \in N$, and for all $W''_j = u_j v_{i^*,\ell^*}^\top$ satisfying (6.57), we have

$$\sum_{k \in N} \left| \|\widehat{\nabla}_k f(W_2 + W'_N + W'_{-N})\|_2^2 - \|\widehat{\nabla}_k f(W_2 + W'_N + W_{-N'} + W''_j)\|_2^2 \right| \leq O\left(\rho^8\right) \ .$$

Finally, splitting the randomness (using Fact 6.10.1), and applying a union bound over multiple samples $N_1, \ldots, N_t$, we finish the proof. □

6.14.4 Main Theorem: Proof of Theorem 6.14.2

Proof of Theorem 6.14.2. Without loss of generality, we assume that $\max_{i,\ell} \| \mathsf{loss}_{i,\ell} \|_2^2 = 1$.

Combining Core Lemma A and B (i.e., Lemma 6.14.6 and Lemma 6.14.9), we know that if $|N|$ is appropriately chosen, with probability $\geq 1 - e^{-\Omega(\rho^2)}$ over the randomness of W_2, A, B and $N_1, \cdots N_t$, the following holds:

- for every $N \in \{N_1, N_2, \cdots, N_t\}$, the following holds:
  - with probability at $1 - e^{-\Omega(\rho^2)}$ over the randomness of W'_{-N}, the following boxed statement holds:
- (core A)

$$\mathbb{E}_{W'_N} \left[\textstyle\sum_{k \in N} \| \widehat{\nabla}_k f(W_2 + W'_N + W'_{-N}) \|_2^2 \right] \geq q \stackrel{\text{def}}{=} \Omega\left(\tfrac{\beta_- |N|}{\rho^2} \right)$$

- (core B) with probability at least $1 - e^{-\Omega(\rho^2)}$ over the randomness of W'_N, the following holds:
  - for all $j \in N$, and for all $W''_j = u_j v_{i^*,\ell^*}^\top$ satisfying

$$\|u_j\|_0 = 1, \quad \|u_j\|_\infty \leq \tfrac{3\theta\rho}{\sqrt{m}}, \quad v_{i^*,\ell^*} = \tfrac{(I - U_{\ell^*-1} U_{\ell-1}^\top) h_{i^*,\ell} h_{i^*,\ell^*}}{\|(I - U_{\ell^*-1} U_{\ell^*-1}^\top) h_{i^*,\ell^*}\|_2} \ ,$$

    we have

$$\textstyle\sum_{k \in N} \left| \| \widehat{\nabla}_k f(W_2 + W'_N + W'_{-N}) \|_2^2 - \| \widehat{\nabla}_k f(W_2 + W'_N + W'_{-N} + W''_j) \|_2^2 \right| \leq p \stackrel{\text{def}}{=} O(\rho^8) \ .$$

We now wish to apply our extended McDiarmid's inequality (see Lemma 6.10.18) to the boxed statement. For this goal, recalling that $W'_N = u_N v_{i^*,\ell^*}^\top$ for a vector $u_N \in \mathbb{R}^m$ that is only supported on indices in N. That is, $(u_N)_k = 0$ for all $k \notin N$. Therefore, we can define function $F(u_N) \stackrel{\text{def}}{=} \sum_{k \in N} \| \widehat{\nabla}_k f(W_2 + W'_{-N} + W'_N) \|^2$ where $W'_N = u_N v_{i^*,\ell^*}^\top$. We emphasize

here that, inside the boxed statement, W_2, W'_{-N} and v_{i^*,ℓ^*} are already fixed, and u_N is the only source of randomness.

Below, we condition on the high probability event (see Lemma 6.14.3d) that $\|u_N\|_\infty \leq \frac{3\theta\rho}{2\sqrt{m}}$. The boxed statement tells us:

- $\mathbb{E}_{u_N}\left[\digamma(u_N)\right] \geq q$, and
- With probability $\geq 1 - e^{-\Omega(\rho^2)}$ over u_N, it satisfies

$$\forall j \in [N], \forall u''_j \colon \left|\digamma(u_{-j}, u_j) - \digamma(u_{-j}, u''_j)\right| \leq p \ .$$

At the same time, we have $0 \leq \digamma(u_N) \leq O(N\rho^8)$ owing to Lemma 6.14.1. Therefore, scaling down $\digamma(u_N)$ by $\Theta(N\rho^8)$ (to make sure the function value stays in $[0,1]$), we can applying extended McDiarmid's inequality (see Lemma 6.10.18), we have

$$\Pr_{u_N}\left[\digamma(u_N) \geq \frac{q}{2}\right] \geq 1 - N^2 \cdot e^{-\Omega(\rho^2)} - \exp\left(-\Omega\left(\frac{(q/\rho^8)^2}{N(p/\rho^8)^2 + Ne^{-\Omega(\rho^2)}}\right)\right) \ . \tag{6.59}$$

As long as $N \geq \frac{\rho^{22}}{\beta_-^2}$, we have that the above probability is at least $1 - e^{-\Omega(\rho^2)}$.

Finally, we choose

$$\boxed{N = \frac{\rho^{22}}{\beta_-^2}} \tag{6.60}$$

in order to satisfy Lemma 6.14.6. We replace the boxed statement with (6.59). This tells us

- with probability $\geq 1 - e^{-\Omega(\rho^2)}$ over the randomness of W_2, A, B and $N_1, \cdots N_t$, the following holds:
  - for every $N \in \{N_1, N_2, \cdots, N_t\}$, the following holds:

* with probability at $1 - e^{-\Omega(\rho^2)}$ over the randomness of W'_{-N} and W'_N, the following holds:

$$\sum_{k \in N} \|\widehat{\nabla}_k f(W_2 + W'_N + W'_{-N})\|_2^2 \geq \Omega\left(\frac{\beta_- |N|}{\rho^2}\right)$$

After rearranging randomness, and using $W = W_2 + W'_N + W'_{-N}$, we have

- with probability $\geq 1 - e^{-\Omega(\rho^2)}$ over the randomness of W, A, B, the following holds:
  - with probability $\geq 1 - e^{-\Omega(\rho^2)}$ over the randomness of $N_1, \cdots N_t$, the following holds:
    * for every $N \in \{N_1, N_2, \cdots, N_t\}$, the following holds:

$$\sum_{k \in N} \|\widehat{\nabla}_k f(W)\|_2^2 \geq \Omega\left(\frac{\beta_- |N|}{\rho^2}\right)$$

Finally, since $N_1, \ldots, N_t$ are t random subsets of $[m]$, we know that with probability at least $1 - e^{-\Omega(\rho^2)}$, for each index $k \in [m]$, it is covered by at most ρ^2 random subsets. Therefore,

$$\sum_{k \in N} \|\widehat{\nabla}_k f(W)\|_2^2 \geq \frac{1}{\rho^2} \times t \times \Omega\left(\frac{\beta_- |N|}{\rho^2}\right) = \Omega\left(\frac{\delta}{\rho^{14}}\right) \times m \ .$$

□

6.15 Gradient Bound After Perturbation (Theorem 6.15.2)

We use the same fake gradient notion (see Definition 6.14.1) and first derive the following result based on Lemma 6.14.1 and Theorem 6.14.2.

Lemma 6.15.1. *Let* $\rho = nLd\log m$ *and* $\varrho = nLd\delta^{-1}\log(m/\epsilon)$. *Given fixed vectors* $\{\mathsf{loss}_{i,\ell} \in \mathbb{R}^d\}_{i\in[n],\ell\in[L]\setminus\{1\}}$, *with probability at least* $1 - e^{-\Omega(\rho^2)}$ *over* $\widetilde{W}, A, B$, *it satisfies for* all $W' \in \mathbb{R}^{m\times m}$ *with* $\|W'\|_2 \le \frac{\tau_0}{\sqrt{m}}$ *with* $\tau_0 \le \varrho^{100}$,

$$\|\widehat{\nabla} f(\widetilde{W}+W')\|_F^2 \ge \Omega\left(\frac{\delta}{\rho^{14}}\right) \times m \times \max_{i,\ell}\{\|\mathsf{loss}_{i,\ell}\|^2\}$$
$$\|\widehat{\nabla} f(\widetilde{W}+W')\|_F^2 \le O(\rho^{12}m) \times \max_{i,\ell}\{\|\mathsf{loss}_{i,\ell}\|^2\}$$
$$\|\widehat{\nabla} f_i(\widetilde{W}+W')\|_F^2 \le \frac{1}{n^2}O(\rho^{12}m) \times \max_{i,\ell}\{\|\mathsf{loss}_{i,\ell}\|^2\} \ .$$

Proof of Lemma 6.15.1. Like before, we denote by $\widetilde{D}_{i,\ell}, \widetilde{u}_{i,\ell}, \widetilde{h}_{i,\ell}$ the corresponding matrices or vectors when the weight matrix is $\widetilde{W}$, and by $D_{i,\ell}, u_{i,\ell}, h_{i,\ell}$ the corresponding terms when when the weight matrix is $W = \widetilde{W} + W'$.

Lower bound on $\|\widehat{\nabla} f(\widetilde{W}+W')\|_F$ Recall

$$\widehat{\nabla} f(\widetilde{W}+W') = \sum_{i=1}^{n}\sum_{\ell=1}^{L} D_{i,\ell+1}\left(\sum_{a=\ell}^{L}\mathsf{Back}_{i,\ell\to a}^\top \mathsf{loss}_{i,a}\right) h_{i,\ell}^\top = \sum_{i=1}^{n}\sum_{\ell=1}^{L} D_{i,\ell+1}u_{i,\ell}h_{i,\ell}^\top$$

In Theorem 6.14.2 of the previous section, we already have a lower bound on $\|\widehat{\nabla} f(\widetilde{W})\|_F$.

We just need to upper bound $\|\widehat{\nabla} f(\widetilde{W}+W') - \widehat{\nabla} f(\widetilde{W})\|_F$.

$$\begin{aligned}
\|\widehat{\nabla} f(\widetilde{W}+W') - \widehat{\nabla} f(\widetilde{W})\|_F &\overset{①}{=} \left\| \sum_{i=1}^{n}\sum_{\ell=1}^{L} D_{i,\ell+1} u_{i,\ell} h_{i,\ell}^\top - \widetilde{D}_{i,\ell+1}\widetilde{u}_{i,\ell}\widetilde{h}_{i,\ell}^\top \right\|_F \\
&\overset{②}{\le} \sum_{i=1}^{n}\sum_{\ell=1}^{L} \left\| D_{i,\ell+1} u_{i,\ell} h_{i,\ell}^\top - \widetilde{D}_{i,\ell+1}\widetilde{u}_{i,\ell}\widetilde{h}_{i,\ell}^\top \right\|_F \\
&\overset{③}{\le} \sum_{i=1}^{n}\sum_{\ell=1}^{L} \|D'_{i,\ell+1}\widehat{u}_{i,\ell}\widehat{h}_{i,\ell}^\top\|_F + \|\widetilde{D}_{i,\ell+1}u'_{i,\ell}\widehat{h}_{i,\ell}^\top\|_F + \|\widetilde{D}_{i,\ell+1}\widetilde{u}_{i,\ell}{h'_{i,\ell}}^\top\|_F + o(m^{1/3}).
\end{aligned}$$

where ① follows by definition and ② and ③ follow by the triangle inequality. Note that in inequality ③ we have hidden four more higher order terms in $o(m^{1/3})$. We ignore the details for how to bound them, for the ease of presentation. We bound the three terms separately:

- Using Corollary 6.11.15 (with $s^2 = O(L^{10/3}\tau_0^{2/3})$ from Lemma 6.12.1b) and Lemma 6.11.1 we have

$$\begin{aligned}
\|D'_{i,\ell+1}\widehat{u}_{i,\ell}\widehat{h}_{i,\ell}^\top\|_F &\le \|D'_{i,\ell+1}\widehat{u}_{i,\ell}\|_2 \cdot \|\widehat{h}_{i,\ell}\|_2 \\
&\le O(L \cdot L^4\tau_0^{1/3}m^{1/3}\log m \cdot \|\,\mathsf{loss}_{i^*,\ell^*}\,\|_2)\cdot L
\end{aligned}$$

- Using $\|\widetilde{D}_{i,\ell+1}\|_2 \le 1$, Lemma 6.12.8 and Lemma 6.11.1 we have

$$\begin{aligned}
\|\widetilde{D}_{i,\ell+1}u'_{i,\ell}\widehat{h}_{i,\ell}^\top\|_F &\le \|\widetilde{D}_{i,\ell+1}\| \cdot \|u'_{i,\ell}\|_2 \cdot \|\widehat{h}_{i,\ell}\|_2 \\
&\le O(L \cdot L^6\tau_0^{1/3}m^{1/3}\log m \cdot \|\,\mathsf{loss}_{i^*,\ell^*}\,\|_2)\cdot L
\end{aligned}$$

- Using $\|\widetilde{D}_{i,\ell+1}\|_2 \le 1$, $\|\widetilde{u}_{i,\ell}\|_2 \le O(L^4)\sqrt{m}\|\,\mathsf{loss}_{i^*,\ell^*}\,\|_2$ (implied by Lemma 6.11.8) and Lemma 6.12.1a, we have

$$\begin{aligned}
\|\widetilde{D}_{i,\ell+1}\widetilde{u}_{i,\ell}{h'_{i,\ell}}^\top\|_F &\le \|\widetilde{D}_{i,\ell+1}\| \cdot \|\widetilde{u}_{i,\ell}\|_2 \cdot \|h'_{i,\ell}\|_2 \\
&\le O(L^4\sqrt{m} \cdot \|\,\mathsf{loss}_{i^*,\ell^*}\,\|_2)\cdot (L^6\tau_0^{1/2}\frac{1}{\sqrt{m}})
\end{aligned}$$

Putting it all together, we have

$$\|\widehat{\nabla} f(\widetilde{W}+W') - \widehat{\nabla} f(\widetilde{W})\|_F \le O(\rho^8 \tau_0^{1/3} m^{1/3}) \cdot \|\,\mathsf{loss}_{i^*,\ell^*}\,\|_F \ . \tag{6.61}$$

Finally, using $(a-b)^2 \ge \frac{1}{2}a^2 - b^2$, we have

$$\begin{aligned}
\|\widehat{\nabla} f(\widetilde{W}+W')\|_F^2 &\ge (\|\widehat{\nabla} f(\widetilde{W})\|_F - \|\widehat{\nabla} f(\widetilde{W}+W') - \widehat{\nabla} f(\widetilde{W})\|_F)^2 \\
&\ge \frac{1}{2}\|\widehat{\nabla} f(\widetilde{W})\|_F^2 - \|\widehat{\nabla} f(\widetilde{W}+W') - \widehat{\nabla} f(\widetilde{W})\|_F^2 \\
&\ge \Omega\left(\frac{\delta}{\rho^{14}}\right) \times m \times \|\,\mathsf{loss}_{i^*,\ell^*}\,\|^2 \ .
\end{aligned}$$

where the last step follows by (6.61) (with our sufficiently large choice of m) and Theorem 6.14.2.

Upper bound on $\|\widehat{\nabla} f(\widetilde{W}+W')\|_F$ Using $(a+b)^2 \le 2a^2 + 2b^2$, we have

$$\begin{aligned}
\|\widehat{\nabla} f(\widetilde{W}+W')\|_F^2 &\le 2\|\widehat{\nabla} f(\widetilde{W})\|_F^2 + 2\|\widehat{\nabla} f(\widetilde{W}+W') - \widehat{\nabla} f(\widetilde{W})\|_F^2 \\
&\overset{①}{\le} O(\rho^{12} m) \cdot \|\,\mathsf{loss}_{i^*,\ell^*}\,\|_F^2 + 2\|\widehat{\nabla} f(\widetilde{W}+W') - \widehat{\nabla} f(\widetilde{W})\|_F^2 \overset{②}{\le} O(\rho^{12} m) \cdot \|\,\mathsf{loss}_{i^*,\ell^*}\,\|_F^2
\end{aligned}$$

where ① follows by Lemma 6.14.1 and ② follows by (6.61).

Upper bound on $\|\widehat{\nabla} f_i(\widetilde{W}+W')\|_F$ This is completely analogous so we do not replicate the proofs here. □

By applying an ϵ-net argument over all possible $\mathsf{loss}_{i,\ell} \in \mathbb{R}^d$, we can modify Lemma 6.15.1 from "for fixed loss" to "for all loss." This allows us to plug in the true loss $\mathsf{loss}_{i,\ell} = Bh_{i,\ell} - y^*_{i,\ell}$ and derive that:

Theorem 6.15.2 (restated)**.** *Let* $\rho = nLd \log m$ *and* $\varrho = nLd\delta^{-1} \log(m/\epsilon)$*. With probability at least* $1 - e^{-\Omega(\rho^2)}$ *over* W, A, B*, it satisfies for* all $W' \in \mathbb{R}^{m \times m}$ *with* $\|W'\|_2 \leq \frac{\varrho^{100}}{\sqrt{m}}$*,*

$$\|\nabla f(\widetilde{W} + W')\|_F^2 \geq \Omega\left(\frac{\delta}{\rho^{14}}\right) \times m \times \max_{i,\ell}\{\| \mathsf{loss}_{i,\ell} \|^2\}$$
$$\|\nabla f(\widetilde{W} + W')\|_F^2 \leq O(\rho^{12} m) \times \max_{i,\ell}\{\| \mathsf{loss}_{i,\ell} \|^2\}$$
$$\|\nabla f_i(\widetilde{W} + W')\|_F^2 \leq \frac{1}{n^2} O(\rho^{12} m) \times \max_{i,\ell}\{\| \mathsf{loss}_{i,\ell} \|^2\} \ .$$

6.16 Objective Semi-Smoothness (Theorem 6.16.1)

The purpose of this section is to prove

Theorem 6.16.1 (objective semi-smoothness, restated)**.** *Let $\rho = nLd\log m$, $\varrho = nLd\delta^{-1}\log(m/\epsilon)$, $\tau_0 \in [\varrho^{-100}, \varrho^{100}]$, and $\widetilde{W}, A, B$ be at random initialization. With probability at least $1-e^{-\Omega(\rho^2)}$ over the randomness of $\widetilde{W}, A, B$, we have for every $\breve{W} \in \mathbb{R}^{m\times m}$ with $\|\breve{W}-\widetilde{W}\| \le \frac{\tau_0}{\sqrt{m}}$, and for every $W' \in \mathbb{R}^{m\times m}$ with $\|W'\| \le \frac{\tau_0}{\sqrt{m}}$, and letting $(i^*,\ell^*) = \mathrm{argmax}_{i,\ell}\{\|\breve{\mathsf{loss}}_{i,\ell}\|_2\}$,*

$$f(\breve{W}+W') \le f(\breve{W}) + \langle \nabla f(\breve{W}), W'\rangle + O(\rho^{11}\tau_0^{1/3}m^{1/3})\cdot\|\breve{\mathsf{loss}}_{i^*,\ell^*}\|_2\cdot\|W'\|_2 + O(L^{18}nm)\|W'\|_2^2$$

We introduce the following notations before we go to proofs.

Definition 6.16.1. For $i\in[n]$ and $\ell\in[L]$:

$$\begin{array}{lll}
\widetilde{g}_{i,0} = \widetilde{h}_{i,0} = 0 & \breve{g}_{i,0} = \hbar_{i,0} = 0 & g_{i,0} = h_{i,0} = 0 \\
\widetilde{g}_{i,\ell} = \widetilde{W}\widetilde{h}_{i,\ell-1} + Ax_{i,\ell} & \breve{g}_{i,\ell} = \breve{W}\hbar_{i,\ell-1} + Ax_{i,\ell} & g_{i,\ell} = (\breve{W}+W')h_{i,\ell-1} + Ax_{i,\ell} \\
\widetilde{h}_{i,\ell} = \phi(\widetilde{W}\widetilde{h}_{i,\ell-1} + Ax_{i,\ell}) & \hbar_{i,\ell} = \phi(\breve{W}\hbar_{i,\ell-1} + Ax_{i,\ell}) & h_{i,\ell} = \phi((\breve{W}+W')h_{i,\ell-1} + Ax_{i,\ell}) \\
 & \breve{\mathsf{loss}}_{i,\ell} = B\hbar_{i,\ell} - y^*_{i,\ell} &
\end{array}$$

Define diagonal matrices $\widetilde{D}_{i,\ell}$ and $\breve{D}_{i,\ell}$ respectively by letting

$$(\widetilde{D}_{i,\ell})_{k,k} = \mathbb{1}_{(\widetilde{g}_{i,\ell})_k\ge 0} \text{ and } (\breve{D}_{i,\ell})_{k,k} = \mathbb{1}_{(\breve{g}_{i,\ell})_k\ge 0}.$$

The following claim gives rise to a new recursive formula to calculate $h_{i,\ell} - \hbar_{i,\ell}$.

Claim 6.16.2 (c.f. (6.21))**.** *There exist diagonal matrices $D''_{i,\ell}\in\mathbb{R}^{m\times m}$ with entries in $[-1,1]$ such that,*

$$\forall i\in[n], \forall \ell\in[L]\colon \quad h_{i,\ell} - \hbar_{i,\ell} = \sum_{a=1}^{\ell-1}(\breve{D}_{i,\ell} + D''_{i,\ell})\breve{W}\cdots\breve{W}(\breve{D}_{i,a+1} + D''_{i,a+1})W'h_{i,a} \tag{6.62}$$

Furthermore, we have $\|h_{i,\ell} - \hbar_{i,\ell}\| \le O(L^9)\|W'\|_2$ *and* $\|D''_{i,\ell}\|_0 \le O(L^{10/3}\tau_0^{2/3}m^{2/3})$.

Proof of Claim 6.16.2. We ignore the subscript in i for cleanness, and calculate that

$$\begin{aligned} h_\ell - \hbar_\ell &\overset{①}{=} \phi((\breve{W} + W')h_{\ell-1} + Ax_\ell) - \phi(\breve{W}\hbar_{\ell-1} + Ax_\ell) \\ &\overset{②}{=} (\breve{D}_\ell + D''_\ell)\left((\breve{W} + W')h_{\ell-1} - \breve{W}\hbar_{\ell-1}\right) \\ &= (\breve{D}_\ell + D''_\ell)\breve{W}(h_{\ell-1} - \hbar_{\ell-1}) + (\breve{D}_\ell + D''_\ell)W'h_{\ell-1} \\ &\overset{③}{=} \sum_{a=1}^{\ell-1}(\breve{D}_\ell + D''_\ell)\breve{W}\cdots\breve{W}(\breve{D}_{a+1} + D''_{a+1})W'h_a \end{aligned}$$

Above, ① is by the recursive definition of h_ℓ and $\hbar_\ell$; ② is by Proposition 6.16.3 and D''_ℓ is defined according to Proposition 6.16.3; and inequality ③ is by recursively computing $h_{\ell-1} - \hbar_{\ell-1}$. As for the two properties:

- We have $\|h_\ell - \hbar_\ell\| \le O(L^9)\|W'\|_2$. This is because we have
  - $\left\|(\breve{D}_\ell + D''_\ell)\breve{W}\cdots\breve{W}(\breve{D}_{a+1} + D''_{a+1})\right\|_2 \le O(L^7)$ by Lemma 6.12.6;
  - $\|h_a\| \le O(L)$ (by $\|\widetilde{h}_a\| \le O(L)$ from Lemma 6.11.1 and $\|\widetilde{h}_a - h_a\| \le o(1)$ from Lemma 6.12.1a); and
  - $\|W'h_a\| \le \|W'\|_2\|h_a\|$.
- We have $\|D''_\ell\|_0 \le O(L^{10/3}\tau_0^{2/3}m^{2/3})$.

  This is because, $(D''_\ell)_{k,k}$ is non-zero only at the coordinates $k \in [m]$ where the signs of $\breve{g}_\ell$ and g_ℓ are opposite (by Proposition 6.16.3). Such a coordinate k must satisfy either $(\widetilde{D}_\ell)_{k,k} \ne (\breve{D}_\ell)_{k,k}$ or $(\widetilde{D}_\ell)_{k,k} \ne (D_\ell)_{k,k}$, and therefore by Lemma 6.12.1 — with probability $\ge 1 - e^{-\Omega(L^6\tau_0^{4/3}m^{1/3})}$ — there are at most $O(L^{10/3}\tau_0^{2/3}m^{2/3})$ such coordinates k.

□

Proof of Theorem 6.16.1. First of all, since

$$\frac{1}{2}\|Bh_{i,\ell}-y^*_{i,\ell}\|^2=\frac{1}{2}\|\breve{\mathsf{loss}}_{i,\ell}+B(h_{i,\ell}-\hbar_{i,\ell})\|^2=\frac{1}{2}\|\breve{\mathsf{loss}}_{i,\ell}\|^2+\breve{\mathsf{loss}}_{i,\ell}^\top B(h_{i,\ell}-\hbar_{i,\ell})+\frac{1}{2}\|B(h_{i,\ell}-\hbar_{i,\ell})\|^2 \tag{6.63}$$

we can write

$$\begin{aligned}
&f(\breve{W}+W')-f(W)-\langle\nabla f(W),W'\rangle\\
\overset{①}{=}&-\langle\nabla f(\breve{W}),W'\rangle+\frac{1}{2}\sum_{i=1}^n\sum_{\ell=2}^L\|Bh_{i,\ell}-y^*_{i,\ell}\|^2-\|B\hbar_{i,\ell}-y^*_{i,\ell}\|^2\\
\overset{②}{=}&-\langle\nabla f(\breve{W}),W'\rangle+\sum_{i=1}^n\sum_{\ell=2}^L\breve{\mathsf{loss}}_{i,\ell}^\top B(h_{i,\ell}-\hbar_{i,\ell})+\frac{1}{2}\|B(h_{i,\ell}-\hbar_{i,\ell})\|^2\\
\overset{③}{=}&\sum_{i=1}^n\sum_{\ell=2}^L\breve{\mathsf{loss}}_{i,\ell}^\top B\left((h_{i,\ell}-\hbar_{i,\ell})-\sum_{a=1}^{\ell-1}\breve{D}_{i,\ell}\breve{W}\cdots\breve{W}\breve{D}_{i,a+1}W'\hbar_{i,a}\right)+\frac{1}{2}\|B(h_{i,\ell}-\hbar_{i,\ell})\|^2\\
\overset{④}{=}&\sum_{i=1}^n\sum_{\ell=2}^L\breve{\mathsf{loss}}_{i,\ell}^\top B\left(\sum_{a=1}^{\ell-1}(\breve{D}_{i,\ell}+D''_{i,\ell})\breve{W}\cdots\breve{W}(\breve{D}_{i,a+1}+D''_{i,a+1})W'h_{i,a}-\breve{D}_{i,\ell}\breve{W}\cdots\breve{W}\breve{D}_{i,a+1}W'\hbar_{i,a}\right)\\
&+\frac{1}{2}\|B(h_{i,\ell}-\hbar_{i,\ell})\|^2
\end{aligned} \tag{6.64}$$

Above, ① is by the definition of $f(\cdot)$; ② is by (6.63); ③ is by the definition of $\nabla f(\cdot)$ (see Fact 6.2.3 for an explicit form of the gradient); ④ is by Claim 6.16.2.

We next bound the RHS of (6.64). We drop subscripts in i for notational simplicity.

We first use (6.62) in Claim 6.16.2 to calculate

$$\begin{aligned}\|B(h_\ell-\hbar_\ell)\| &\le \sum_{a=1}^{\ell-1}\|B(\breve{D}_\ell+D''_\ell)\breve{W}\cdots\breve{W}(\breve{D}_{a+1}+D''_{a+1})W'h_a\| \\ &\le \sum_{a=1}^{\ell-1}\|B\|_2\|(\breve{D}_\ell+D''_\ell)\breve{W}\cdots\breve{W}(\breve{D}_{a+1}+D''_{a+1})\|_2\|W'\|_2\|h_a\| \\ &\overset{①}{\le} \sum_{a=1}^{\ell-1} O(\sqrt{m})\cdot O(L^7)\cdot O(L)\cdot\|W'\|_2 \le O(L^9\sqrt{m})\cdot\|W'\|_2\ . \end{aligned} \tag{6.65}$$

In the last inequality ① above, we have used $\left\|(\breve{D}_\ell+D''_\ell)\breve{W}\cdots\breve{W}(\breve{D}_{a+1}+D''_{a+1})\right\|_2\le O(L^7)$ from Lemma 6.12.6; we have used $\|B\|_2\le O(\sqrt{m})$ with high probability; and we have used $\|h_a\|\le O(L)$ (by $\|\widetilde{h}_a\|\le O(L)$ from Lemma 6.11.1 and $\|\widetilde{h}_a-h_a\|\le o(1)$ from Lemma 6.12.1a).

Since for each D''_ℓ, we can write it as $D''_\ell=D_\ell^{0/1}D''_\ell D_\ell^{0/1}$, where each $D_\ell^{0/1}$ is a diagonal matrix satisfying

$$(D_\ell^{0/1})_{k,k}=\begin{cases}1, & (D''_\ell)_{k,k}\neq 0;\\ 0, & (D''_\ell)_{k,k}=0.\end{cases}\qquad\text{and}\quad \|D_\ell^{0/1}\|_0\le O(L^{10/3}\tau_0^{2/3}m^{2/3})$$

Therefore,

$$\begin{aligned}&\left|\breve{\mathsf{loss}}_\ell^\top B(\breve{D}_\ell+D''_\ell)\breve{W}\cdots\breve{W}(\breve{D}_{a+1}+D''_{a+1})W'h_a-\breve{\mathsf{loss}}_\ell^\top B\breve{D}_\ell\breve{W}\cdots\breve{D}_{a+1}W'h_a\right| \\ &\overset{①}{\le}\|\breve{\mathsf{loss}}_\ell\|_2\cdot\sum_{b=1}^{\ell-a}\binom{\ell-a}{b}\|B\breve{D}\breve{W}\cdots\breve{D}\breve{W}D^{0/1}\|_2\cdot\|D^{0/1}\breve{W}\breve{D}\cdots\breve{D}\breve{W}D^{0/1}\|_2^{b-1}\cdot\|D^{0/1}\breve{W}\cdots\breve{W}\breve{D}W'h_a\|_2 \\ &\overset{②}{\le}\|\breve{\mathsf{loss}}_\ell\|_2\cdot\sum_{b=1}^{\ell-a}\binom{\ell-a}{b}\cdot O(\rho^2\tau_0^{1/3}m^{1/3})\cdot\left(O(\frac{\rho^2}{m^{1/6}})\right)^{b-1}\cdot O(L^7)\cdot\|W'\|_2\cdot O(L) \\ &\overset{②}{\le}\|\breve{\mathsf{loss}}_\ell\|_2\cdot O(\rho^{11}\tau_0^{1/3}m^{1/3})\cdot\|W'\|_2\end{aligned} \tag{6.66}$$

Above, ① is an abbreviation and we have dropped the subscripts for the easy of presentation; ② uses Corollary 6.11.15, Corollary 6.11.12, Lemma 6.12.6 and Lemma 6.11.1.

Finally, we also have

$$\begin{aligned}\left|\breve{\mathsf{loss}}_\ell^\top B\breve{D}_\ell\breve{W}\cdots\breve{D}_{a+1}W'(h_a-\hbar_a)\right| &\overset{①}{\le} \|\breve{\mathsf{loss}}_\ell\|_2\cdot O(\sqrt{m})\cdot O(L^7)\cdot\|W'\|_2\cdot\|h_a-\hbar_a\|_2\\ &\overset{②}{\le} O(\rho^{16}\sqrt{m})\cdot\|\breve{\mathsf{loss}}_\ell\|_2\cdot\|W'\|_2^2 \qquad (6.67)\end{aligned}$$

where ① uses Lemma 6.12.6 and ② uses Claim 6.16.2 to bound $\|h_a-\hbar_a\|_2$.

Putting (6.65), (6.66) and (6.67) back to (6.64), and using triangle inequality, we have the desired result. □

6.16.1 Tool

Proposition 6.16.3. *Given vectors $a, b \in \mathbb{R}^m$ and $D \in \mathbb{R}^{m \times m}$ the diagonal matrix where $D_{k,k} = \mathbb{1}_{a_k \geq 0}$. Then, then there exists a diagonal matrix $D'' \in \mathbb{R}^{m \times m}$ with*

- $|D''_{k,k}| \leq 1$ *for every* $k \in [m]$,
- $D''_{k,k} \neq 0$ *only when* $\mathbb{1}_{a_k \geq 0} \neq \mathbb{1}_{b_k \geq 0}$, *and*
- $\phi(a) - \phi(b) = D(a-b) + D''(a-b)$

Proof. We verify coordinate by coordinate for each $k \in [m]$.

- If $a_k \geq 0$ and $b_k \geq 0$, then $(\phi(a) - \phi(b))_k = a_k - b_k = \big(D(a-b)\big)_k$.
- If $a_k < 0$ and $b_k < 0$, then $(\phi(a) - \phi(b))_k = 0 - 0 = \big(D(a-b)\big)_k$.
- If $a_k \geq 0$ and $b_k < 0$, then $(\phi(a) - \phi(b))_k = a_k = (a_k - b_k) + \frac{b_k}{a_k - b_k}(a_k - b_k) = \big(D(a-b) + D''(a-b)\big)_k$, if we define $(D'')_{k,k} = \frac{b_k}{a_k - b_k} \in [-1, 0]$.
- If $a_k < 0$ and $b_k \geq 0$, then $(\phi(a) - \phi(b))_k = -b_k = 0 \cdot (a_k - b_k) - \frac{b_k}{b_k - a_k}(a_k - b_k) = \big(D(a-b) + D''(a-b)\big)_k$, if we define $(D'')_{k,k} = \frac{b_k}{b_k - b_a} \in [0, 1]$. □

6.17 Convergence Rate of Gradient Descent (Theorem 6.17.1)

Theorem 6.17.1 (gradient descent, restated)**.** *There exists some absolute constant $C > 1$ such that the following holds. Let $\rho = nLd \log m$, $\epsilon \in (0, 1]$, $\varrho = nLd\delta^{-1} \log(m/\epsilon)$, and $\eta \stackrel{\text{def}}{=} \frac{\delta}{\rho^{44} m}$. Suppose $m \geq C\varrho^C$, and $W^{(0)}, A, B$ be at random initialization. Then, with probability at least $1 - e^{-\Omega(\rho^2)}$ over the randomness of $W^{(0)}, A, B$, suppose we start at $W^{(0)}$ and for each $t = 0, 1, \ldots, T-1$,*

$$W^{(t+1)} = W^{(t)} - \eta \nabla f(W^{(t)}) \ .$$

Then, it satisfies

$$f(W^{(T)}) \leq \epsilon \quad \text{for all} \quad T \in \left[\frac{\rho^{59}}{\delta^2} \log \frac{1}{\epsilon}, \frac{\rho^{59}\varrho^{28}}{\delta^2} \log \frac{1}{\epsilon} \right] \ .$$

In other words, the training loss of the recurrent neural network drops to ϵ in a linear convergence speed.

To present the simplest possible result, we have not tried to tighten the polynomial dependency with respect to n, d and L. We only tightened the dependency with respect to δ and ϵ. In fact, a more involved analysis can also get ride of the $\log(1/\epsilon)$ dependency in m [AZLS19].

Proof of Theorem 6.17.1. Using Lemma 6.11.1 and the randomness of B, it is easy to show that $\|Bh_{i,\ell} - y^*_{i,\ell}\|^2 \leq O(\rho^2 L^2)$ with at least $1 - e^{-\Omega(\rho^2)}$ (where $h_{i,\ell}$ is defined with respect to $W^{(0)}$), and therefore

$$f(W^{(0)}) \leq O(n\rho^2 L^3) \ .$$

In the rest of the proof, we first assume that for every $t = 0, 1, \ldots, T-1$, the following holds

$$\|W^{(t)} - W^{(0)}\|_F \leq \frac{\tau_0}{\sqrt{m}} \stackrel{\text{def}}{=} \frac{\varrho^{50}}{\sqrt{m}} \ . \tag{6.68}$$

We shall prove the convergence of gradient descent assuming (6.68), so that previous statements such as Theorem 6.16.1 and Theorem 6.15.2 can be applied. At the end of the proof, we shall verify that (6.68) is satisfied throughout the gradient descent process.

For each $t = 0, 1, \dots, T-1$, we let $\{\mathsf{loss}_{i,\ell}^{(t)}\}_{i,\ell}$ denote the loss vectors with respect to the current point $W^{(t)}$. We denote by $(i^*, \ell^*) = \mathrm{argmax}_{i,\ell}\{\| \mathsf{loss}_{i,\ell}^{(t)} \|_2\}$ and $\nabla_t = \nabla f(W^{(t)})$.

We calculate that

$$\begin{aligned}
f(W^{(t+1)}) &\overset{①}{\le} f(W^{(t)}) - \eta\|\nabla_t\|_F^2 + O(\rho^{11}\tau_0^{1/3}m^{1/3}) \cdot \| \mathsf{loss}_{i^*,\ell^*}^{(t)} \|_2 \cdot \eta\|\nabla_t\|_2 + O(L^{18}nm\eta^2)\|\nabla_t\|_2^2 \\
&\overset{②}{\le} f(W^{(t)}) - \eta\|\nabla_t\|_F^2 + O\left(\rho^{30}\eta^2m^2\right) \cdot \| \mathsf{loss}_{i^*,\ell^*}^{(t)} \|_2^2 \\
&\overset{③}{\le} f(W^{(t)}) - \left(\Omega\Big(\frac{\eta\delta}{\rho^{14}}m\Big) - O(\rho^{30}\eta^2m^2)\right) \cdot \| \mathsf{loss}_{i^*,\ell^*}^{(t)} \|_2^2 \\
&\overset{④}{\le} f(W^{(t)}) - \Omega\Big(\frac{\eta\delta}{\rho^{14}}m\Big) \cdot \| \mathsf{loss}_{i^*,\ell^*}^{(t)} \|_2^2 \\
&\overset{⑤}{\le} \left(1 - \Omega\Big(\frac{\eta\delta}{\rho^{15}}m\Big)\right) f(W^{(t)})
\end{aligned}$$

Above, ① uses Theorem 6.16.1; ② uses Theorem 6.15.2 (which gives $\|\nabla_t\|_2 \le \|\nabla_t\|_F \le O(\rho^6\sqrt{m}) \times \| \mathsf{loss}_{i^*,\ell^*}^{(t)} \|$), and our choices of τ_0 and m; ③ uses Theorem 6.15.2; ④ uses our choice of η; and ⑤ uses $f(W^{(t)}) \le nL\| \mathsf{loss}_{i^*,\ell^*}^{(t)} \|^2$. In other words, after $T = \Omega(\frac{\rho^{15}}{\eta\delta m}) \log \frac{nL^2}{\epsilon}$ iterations we have $f(W^{(T)}) \le \epsilon$.

We need to verify for each t, $\|W^{(t)} - W^{(0)}\|_F$ is small so that (6.68) holds. By Theorem 6.15.2,

$$\begin{aligned}
\|W^{(t)} - W^{(0)}\|_F \le \sum_{i=0}^{t-1} \|\eta\nabla f(W^{(i)})\|_F \le O(\eta\rho^6\sqrt{m}) \cdot \sum_{i=0}^{t-1} \sqrt{f(W^{(i)})} &\le O(\eta\rho^6\sqrt{m}) \cdot O(T \cdot \sqrt{n\rho^2L^3}) \\
&\le \eta T \cdot O(\rho^{8.5}\sqrt{m}) \le \frac{\rho^{50}}{\sqrt{m}} \ .
\end{aligned}$$

where the last step follows by our choice of T. □

6.18 Convergence Rate of Stochastic Gradient Descent (Theorem 6.18.1)

Theorem 6.18.1 (stochastic gradient descent, stated)**.** *There exists some absolute constant $C > 1$ such that the following holds. Let $\rho = nLd \log m$, $\epsilon \in (0,1]$, $\varrho = nLd\delta^{-1}\log(m/\epsilon)$, and $\eta \stackrel{\text{def}}{=} \frac{\delta}{\rho^{42} m}$. Suppose $m \geq C\varrho^C$, and $W^{(0)}, A, B$ be at random initialization. Then, with probability at least $1 - e^{-\Omega(\rho^2)}$ over the randomness of $W^{(0)}, A, B$, suppose we start at $W^{(0)}$ and for each $t = 0, 1, \ldots, T-1$,*

$$W^{(t+1)} = W^{(t)} - \eta \cdot \frac{n}{|S_t|} \sum_{i \in S_t} \nabla f(W^{(t)}) \quad \text{(for a random subset } S_t \subseteq [n] \text{ of fixed cardinality.)}$$

Then, it satisfies with probability at least $1 - e^{-\Omega(\rho^2)}$ over the randomness of $S_1, \ldots, S_T$:

$$f(W^{(T)}) \leq \epsilon \quad \textit{for all} \quad T \in \left[\frac{\rho^{57}}{\delta^2} \log \frac{1}{\epsilon}, \frac{\rho^{57}\varrho^{28}}{\delta^2} \log \frac{1}{\epsilon} \right] .$$

(To present the simplest possible result, we have not tried to tighten the polynomial dependency with respect to n, d and L. We only tightened the dependency with respect to δ and ϵ. In fact, a more involved analysis can also get ride of the $\log(1/\epsilon)$ dependency in m.)

Proof of Theorem 6.18.1. The proof is almost identical to that of Theorem 6.17.1. We again have with probability at least $1 - e^{-\Omega(\rho^2)}$

$$f(W^{(0)}) \leq O(n\rho^2 L^3) .$$

Again, we first assume for every $t = 0, 1, \ldots, T-1$, the following holds

$$\|W^{(t)} - W^{(0)}\|_F \leq \frac{\tau_0}{\sqrt{m}} \stackrel{\text{def}}{=} \frac{\varrho^{50}}{\sqrt{m}} . \tag{6.69}$$

We shall prove the convergence of SGD assuming (6.69), so that previous statements such as Theorem 6.16.1 and Theorem 6.15.2 can be applied. At the end of the proof, we shall verify that (6.69) is satisfied throughout the SGD with high probability.

For each $t = 0, 1, \ldots, T-1$, using the same notation as Theorem 6.17.1, except that we choose $\nabla_t = \frac{n}{|S_t|} \sum_{i \in S_t} \nabla f_i(W^{(t)})$. We have $\mathbb{E}_{S_t}[\nabla_t] = \nabla f(W^{(t)})$ and therefore

$$
\begin{aligned}
&\mathop{\mathbb{E}}_{S_t}[f(W^{(t+1)})] \\
&\overset{①}{\le} f(W^{(t)}) - \eta \|\nabla f(W^{(t)})\|_F^2 + O(\rho^{11}\tau_0^{1/3} m^{1/3}) \cdot \| \mathsf{loss}_{i^*,\ell^*}^{(t)} \|_2 \cdot \eta \mathop{\mathbb{E}}_{S_t}[\|\nabla_t\|_2] + O(L^{18} n m \eta^2) \mathop{\mathbb{E}}_{S_t}[\|\nabla_t\|_2^2] \\
&\overset{②}{\le} f(W^{(t)}) - \eta \|\nabla_t\|_F^2 + O\left(\rho^{30}\eta^2 m^2\right) \cdot \| \mathsf{loss}_{i^*,\ell^*}^{(t)} \|_2^2 \\
&\overset{③}{\le} \left(1 - \Omega\Big(\frac{\eta\delta}{\rho^{15}} m\Big)\right) f(W^{(t)}) \ . \qquad (6.70)
\end{aligned}
$$

Above, ① uses Theorem 6.16.1 and $\mathbb{E}_{S_t}[\nabla_t] = \nabla f(W^{(t)})$; ② uses Theorem 6.15.2 (which gives $\|\nabla_t\|_2^2 \le \frac{n^2}{|S_t|^2} \sum_{i \in S_t} \|\nabla f_i(W^{(t)})\|_F^2 \le O(\rho^{12} m) \times \| \mathsf{loss}_{i^*,\ell^*}^{(t)} \|^2$); ③ is identical to the proof of Theorem 6.17.1.

At the same time, we also have the following absolute value bound:

$$
\begin{aligned}
&f(W^{(t+1)}) \\
&\overset{①}{\le} f(W^{(t)}) + \eta \|\nabla f(W^{(t)})\|_F \cdot \|\nabla_t\|_F + O(\rho^{11}\tau_0^{1/3} m^{1/3}) \cdot \| \mathsf{loss}_{i^*,\ell^*}^{(t)} \|_2 \cdot \eta \|\nabla_t\|_2 + O(L^{18} n m \eta^2) \|\nabla_t\|_2^2 \\
&\overset{②}{\le} f(W^{(t)}) + O\left(\rho^{12}\eta m + \rho^{30}\eta^2 m^2\right) \cdot \| \mathsf{loss}_{i^*,\ell^*}^{(t)} \|_2^2 \\
&\overset{③}{\le} \big(1 + O\big(\rho^{12}\eta m\big)\big) f(W^{(t)}) \ . \qquad (6.71)
\end{aligned}
$$

Above, ① uses Theorem 6.16.1 and Cauchy-Shwartz $\langle A, B \rangle \le \|A\|_F \|B\|_F$, and ② uses Theorem 6.15.2 and the derivation from (6.70).

Next, taking logarithm on both sides of (6.70) and (6.71), and using Jensen's inequality $\mathbb{E}[\log X] \le \log \mathbb{E}[X]$, we have

$$
\mathbb{E}[\log f(W^{(t+1)})] \le \log f(W^{(t)}) - \Omega\Big(\frac{\eta\delta}{\rho^{15}} m\Big) \quad \text{and} \quad \log f(W^{(t+1)}) \le \log f(W^{(t)}) + O\big(\rho^{12}\eta m\big)
$$

By one-sided Azuma's inequality (a.k.a. martingale concentration), we have with probability at least $1 - e^{-\Omega(\rho^2)}$, for every $t = 1, 2, \ldots, T$:

$$\log f(W^{(t)}) - \mathbb{E}[\log f(W^{(t)})] \le \sqrt{t} \cdot O\big(\rho^{12}\eta m\big) \cdot \rho \ .$$

This implies two things.

- On one hand, after $T = \Omega(\frac{\rho^{15}\log(nL^2/\epsilon)}{\eta\delta m})$ iterations we have

$$\begin{aligned}\log f(W^{(T)}) &\le \sqrt{T} \cdot O\big(\rho^{12}\eta m\big) \cdot \rho + \log f(W^{(0)}) - \Omega\Big(\frac{\eta\delta}{\rho^{15}} m\Big) T \\ &\le \log f(W^{(0)}) - \Omega\Big(\frac{\eta\delta}{\rho^{15}} m\Big) T \le \log O(n\rho^2 L^3) - \Omega(\log \frac{\rho^5}{\epsilon}) \le \log \epsilon \ .\end{aligned}$$

  Therefore, we have $f(W^{(T)}) \le \epsilon$.

- On the other hand, for every $t = 1, 2, \ldots, T$, we have

$$\begin{aligned}\log f(W^{(t)}) &\le \sqrt{t} \cdot O\big(\rho^{12}\eta m\big) \cdot \rho + \log f(W^{(0)}) - \Omega\Big(\frac{\eta\delta}{\rho^{15}} m\Big) t \\ &\overset{①}{=} \log f(W^{(0)}) - \left(\sqrt{\frac{\eta\delta m}{\rho^{15}}} \cdot \Omega(\sqrt{t}) - \sqrt{\frac{\rho^{15}}{\eta\delta m}} \cdot O(\rho^{13}\eta m)\right)^2 + O\Big(\frac{\rho^{41}\eta m}{\delta}\Big) \\ &\overset{②}{\le} \log f(W^{(0)}) + 1\end{aligned}$$

  where in ① we have used $2a\sqrt{t} - b^2 t = -(b\sqrt{t} - a/b)^2 + a^2/b^2$, and in ② we have used $\eta \le O\big(\frac{\delta}{\rho^{42} m}\big)$. This implies $f(W^{(t)}) \le O(n\rho^2 L^3)$. We can now verify for each t, $\|W^{(t)} - W^{(0)}\|_F$ is small so that (6.69) holds. By Theorem 6.15.2,

$$\begin{aligned}\|W^{(t)} - W^{(0)}\|_F \le \sum_{i=0}^{t-1} \|\eta\nabla_t\|_F &\le O(\eta\rho^6\sqrt{m}) \cdot \sum_{i=0}^{t-1} \sqrt{f(W^{(i)})} \le O(\eta\rho^6\sqrt{m}) \cdot O(T\sqrt{n\rho^2 L^3}) \\ &\le \eta T \cdot O(\rho^{8.5}\sqrt{m}) \le \frac{\varrho^{50}}{\sqrt{m}} \ .\end{aligned}$$

  where the last step follows by our choice of T. □

Part II

Matrix Concentrations

Chapter 7

Matrix Chernoff Bound on Expander

We prove a Chernoff-type bound for sums of matrix-valued random variables sampled via a random walk on an expander, confirming a conjecture due to Wigderson and Xiao [WX06]. Our proof is based on a new multi-matrix extension of the Golden-Thompson inequality which improves in some ways the inequality in [SBT17] and may be of independent interest, as well as an adaptation of an argument for the scalar case due to Healy [Hea08]. Secondarily, we also provide a generic reduction showing that any concentration inequality for vector-valued martingales implies a concentration inequality for the corresponding expander walk, with a weakening of parameters proportional to the squared mixing time.

7.1 Introduction

The Chernoff Bound [Che52] is one of the most widely used probabilistic results in computer science. It states that a sum of independent bounded random variables exhibits subgaussian concentration around its mean. In particular, when the random variables are i.i.d. samples from a fixed distribution, it implies that the empirical mean of k samples is ϵ-close to the true mean with exponentially small deviation probability proportional to $e^{-\Omega(k\epsilon^2)}$.

An important generalization of this bound was achieved by Gillman [Gil98] (with refinements later by [Lez98, Kah97, LP04, WX05, Hea08, Wag08, CLLM12, RR17]), who significantly relaxed the independence assumption to Markov dependence. In particular, suppose G is a regular graph with vertex set $V = [n]$, $X : V \to \mathbb{C}$ is a bounded function, and $v_1, \ldots, v_k$ is a stationary random walk[1] of length k on G. Then, even though the random variables $X(v_i)$ are in not independent (except when G is the complete graph with self loops), it is shown that:

$$\Pr\left[\left|\frac{1}{k}\sum_{i=1}^{k} X(v_i) - \mathbb{E}[X]\right| > \epsilon\right] \leq 2 \cdot \exp(-\Omega((1-\lambda)k\epsilon^2)), \tag{7.1}$$

where $1 - \lambda$ is the spectral gap of the transition matrix of the random walk. The gain here is that sampling a stationary random walk of length k on a constant degree graph with constant spectral gap requires $\log(n) + O(k)$ random bits, which is much less than the $k \log(n)$ bits required to produce k independent samples. Since such graphs can be explicitly constructed, this leads to a generic "derandomization" of the Chernoff bound, which has

[1] That is the first vertex v_1 is chosen uniformly at random – which is the stationary distribution of the graph G.

had several important applications (see [WX05] for a detailed discussion). In particular, it leads to the following randomness efficient sampler for scalar-valued functions ([Gil98]) using known strongly explicit constructions of expander graphs [RVW00, LPS88]:

Theorem 7.1.1 ([Gil98])**.** *For any $\epsilon > 0$ and $k \geq 1$, there is a* poly(r)*-time computable sampler $\sigma : \{0,1\}^r \to [n]^k$, where $r = \log(n) + O(k)$ s.t. for all functions $f : [n] \to [-1,1]$ satisfying $\mathbb{E}\, f = 0$, we have that*

$$\Pr_{w \in_R \{0,1\}^r}\left[\left|\frac{1}{k}\sum_{i=1}^{k} f(\sigma(w)_i)\right| \geq \epsilon\right] \leq 2\exp\left(-\Omega\left(\epsilon^2 k\right)\right).$$

In many applications of interest k is about $\log(n)$, and going from $O(\log^2(n))$ to $O(\log(n))$ random bits leads to a complete derandomization by cycling over all seeds $w \in \{0,1\}^r$.

A different generalization of the Chernoff bound appeared in the works of Rudelson [Rud99], Ahlswede-Winter [AW02], and Tropp [Tro12], who showed that a similar concentration phenomenon is true for *matrix-valued* random variables. In particular, if $X_1, \ldots, X_k$ are independent $d \times d$ complex Hermitian random matrices with $\|X_i\| \leq 1$, then the following is true:

$$\Pr\left[\left\|\frac{1}{k}\sum_{i=1}^{k} X_i - \mathbb{E}[X]\right\| > \epsilon\right] \leq 2d \cdot \exp(-\Omega(k\epsilon^2)). \tag{7.2}$$

The only difference between this and the usual Chernoff bound is the factor of d in front of the deviation probability; to see that it is necessary, notice that the diagonal case simply corresponds to a direct sum of d arbitrarily correlated instances of the scalar Chernoff bound, so by the union bound the probability should be d times as large in the worst case. This so called "Matrix Chernoff Bound" has seen several applications as well, notably in quantum

information theory, numerical linear algebra, and spectral graph theory; the reader may consult e.g. the book [Tro15] for many examples.

We present two different extensions of the above results in this chapter.

7.1.1 A Matrix Expander Chernoff Bound

It is natural to wonder whether there is a common generalization of (7.1) and (7.2), i.e., a "Matrix Expander Chernoff Bound". Such a result was conjectured by Wigderson and Xiao in [WX06] — in fact, [WX05] contained a proof of it, but the authors later discovered a gap in the proof. In this chapter, we prove the Wigderson and Xiao conjecture, namely:

Theorem 7.1.2. *Let $G = (V, E)$ be a regular undirected graph whose transition matrix has second eigenvalue λ, and let $f : V \to \mathbb{C}^{d \times d}$ be a function such that:*

1. *For each $v \in V$, $f(v)$ is Hermitian and $\|f(v)\| \leq 1$.*
2. *$\sum_{v \in V} f(v) = 0$.*

Then, for a stationary random walk $v_1, \ldots, v_k$ with $\epsilon \in (0, 1)$ we have:

$$\Pr\left[\lambda_{\max}\left(\frac{1}{k}\sum_{j=1}^{k} f(v_j)\right) \geq \quad \epsilon\right] \leq d \cdot \exp\left(-\Omega\left(\epsilon^2(1-\lambda)k\right)\right),$$

$$\Pr\left[\lambda_{\min}\left(\frac{1}{k}\sum_{j=1}^{k} f(v_j)\right) \leq -\epsilon\right] \leq d \cdot \exp\left(-\Omega\left(\epsilon^2(1-\lambda)k\right)\right).$$

This theorem adds to the amazingly long list of pseudorandom properties of expander graphs. By applying the theorem with a strongly explicit bounded degree expander, one obtains the following randomness-efficient sampler for matrix-calued functions conjectured in [WX06].

Theorem 7.1.3. *For any $\epsilon > 0$, $k \geq 1$ and $d \geq 1$, there is a $\mathrm{poly}(r)$-time computable sampler $\sigma : \{0,1\}^r \to [n]^k$, where $r = \log(n) + O(k)$ s.t. for all functions $f : [n] \to \mathbb{C}^{d \times d}$ satisfying $\mathbb{E}\, f = 0$ and for each $v \in [n]$, $f(v)$ is Hermitian and $\|f(v)\| \leq 1$, we have that*

$$\Pr_{w \in_R \{0,1\}^r} \left[\left\| \frac{1}{k} \sum_{i=1}^{k} f(\sigma(w)_i) \right\| \geq \epsilon \right] \leq 2d \exp\left(-\Omega\left(-\epsilon^2 k\right)\right).$$

We remark that while the derandomization applications studied in [WX05] were later recovered in [WX08] using the method of pessimistic estimators, that method requires additional assumptions to be efficiently implementable (specifically, computability of the matrix moment generating function, which is problem-dependent) and therefore does not constitute a truly black box derandomization of the matrix Chernoff bound, whereas Theorem 7.1.3 does. Given the increasing ubiquity of applications of this bound, we therefore suspect that it will find further applications in the study of derandomization and expander graphs, beyond the ones mentioned in [WX05].

Techniques

To describe the ideas that go into the proof of Theorem 7.1.2, let us begin by recalling how the usual scalar Chernoff bound is proved, in the case when the random variables have mean zero. The key observation is that if $X_1, \ldots, X_k$ are independent random variables, then the moment generating function of the sum is equal to the product of the moment generating functions:

$$\mathbb{E}\left[\exp\left(t \sum_{i=1}^{k} X_i\right)\right] = \prod_{i=1}^{k} \mathbb{E}[\exp(tX_i)].$$

This is no longer true in case where the X_i come from a random walk, but we still have the algebraic fact that

$$\exp\left(t\sum_{i=1}^{k} X_i\right) = \prod_{i=1}^{k} \exp(tX_i), \tag{7.3}$$

which allows one to decompose the sum as a product. The latter allows one to consider the steps of the random walk separately and analyze the change in the expectation inductively.

The analogue of the moment generating function in the matrix setting is

$$\mathbb{E}\left[\operatorname{tr}\left[\exp\left(t\sum_{i=1}^{k} X_i\right)\right]\right],$$

and the main difficulty is that (7.3) no longer holds if the matrices X_i do not commute. A substitute for this fact is given by the Golden-Thompson inequality [Gol65, Tho65], which states that for any Hermitian A, B:

$$\operatorname{tr}[\exp(A+B)] \leq \operatorname{tr}[\exp(A)\exp(B)]. \tag{7.4}$$

The latter expression may further be bounded by
$\|\exp(A)\|\operatorname{tr}[\exp(B)]$, and this is sufficient to prove (7.2) in the independent case as is done in [AW02], where an inductive application of it yields

$$\mathbb{E}\left[\operatorname{tr}\left[\exp\left(t\sum_{i=1}^{k} X_i\right)\right]\right] \leq \operatorname{tr}[I]\cdot\prod_{i=1}^{k}\left\|\mathbb{E}[\exp(tX_i)]\right\|.$$

However, this approach is too crude to handle the Markov case, roughly because in the absence of inependence, passing to the norm makes it difficult to utilize the fact that the expectation of each X_i is zero.

The original proof of Wigderson-Xiao was based on the following plausible multi-matrix generalization of (7.4):

$$\operatorname{tr}\left[\exp\left(\sum_{i=1}^{k} A_i\right)\right] \leq \operatorname{tr}\left[\prod_{i=1}^{k} \exp(A_i)\right],$$

which turns out to be false for $k > 2$. To see why, observe that the left hand side is always nonnegative, whereas the right hand side can be the trace of a product of any three positive semidefinite matrices, which can be negative (and this is not the case for two matrices). This led to a fatal gap in their proof.

The main ingredient in our proof is a new multi-matrix generalization of (7.4), which is inspired by the following statement that was recently proven in [SBT17] (see also [HKT16]).

Theorem 7.1.4 (Corollary 3.3 in [SBT17])**.** *Let $H_1, \ldots, H_k \in \mathbb{C}^{d\times d}$ be Hermitian matrices. Then*

$$\log\left[\operatorname{tr}\left(\exp\left(\sum_{j=1}^{k} H_j\right)\right)\right] \leq \int_{-\infty}^{\infty} \log\left[\operatorname{tr}\left(\prod_{j=1}^{k} \exp\left(\frac{H_j(1+\mathbf{i}b)}{2}\right) \prod_{j=k}^{1} \exp\left(\frac{H_j(1-\mathbf{i}b)}{2}\right)\right)\right] \mathrm{d}\mu(b)$$

where μ is some probability distribution on $(-\infty, \infty)$.

The above inequality successfully relates the matrix exponential of a sum to a product of matrix exponential, but is not adequate for proving an optimal Chernoff bound. The reason is that all known arguments require a Taylor expansion, and Theorem 7.1.4 involves integration over an unbounded region (this region can be truncated, but this introduces a loss which leads to a suboptimal bound). To remedy this, we prove a new multi-matrix Golden-Thompson inequality, which only involves integration over a bounded region instead of a line.

Theorem 7.1.5 (Bounded Multi-matrix Golden-Thompson inequality)**.** *Let* $H_1, \ldots, H_k \in \mathbb{C}^{d\times d}$ *be Hermitian matrices. Then*

$$\log\left(\operatorname{tr}\left[\exp\left(\sum_{j=1}^{k} H_j\right)\right]\right) \leq \frac{4}{\pi}\int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \log\left(\operatorname{tr}\left[\prod_{j=1}^{k}\exp\left(\frac{e^{\mathbf{i}\phi}}{2}H_j\right)\prod_{j=k}^{1}\exp\left(\frac{e^{-\mathbf{i}\phi}}{2}H_j\right)\right]\right)\mathrm{d}\mu(\phi)$$

where μ *is some probability distribution on* $[-\frac{\pi}{2}, \frac{\pi}{2}]$.

We present the proof in Section 7.3. Theorem 7.1.5 is likely to be of independent interest and could have further applications, e.g. in quantum information theory. We draw attention to the following notable features of the above two theorems:

(a) Since $\exp(H(1+\mathbf{i}b)) = \exp(H(1-\mathbf{i}b))^*$ and $\exp\left(\frac{e^{\mathbf{i}\phi}}{2}H_j\right) = \exp\left(\frac{e^{-\mathbf{i}\phi}}{2}H_j\right)^*$ for Hermitian H, the right hand always considers the trace of a matrix times its adjoint, which is always positive semidefinite, ruling out the bad example described above.

(b) They are *average case* inequalities, where the averaging is done over specific distributions. We remark that the first inequality is known to be false in the worst case (i.e., with $b = 0$ and with other small values of b; see [SBT17] for a discussion).

The main point is that Theorem 7.1.5 allows one to relate the exponential of a sum of matrices to a (two-sided) product of bounded $d \times d$ matrices and their adjoints. In order to prove Theorem 7.1.2, we rewrite this as a one-sided matrix product of $d^2 \times d^2$ matrices acting on $\mathbb{C}^{d\times d}$, by encoding left and right multiplication on this space via a tensor product. However, these matrices are no longer Hermitian (or even normal), so it is difficult to analyze the moment generating function of their product over the random walk using the perturbation-theoretic approach of [WX05]. We surmount this difficulty by employing a

variant of the more robust linear algebraic proof technique of Healy [Hea08]. The proof of Theorem 7.1.2 is presented in Section 7.4.

7.1.2 Martingale Approximation of Expander Walks

While Theorem 7.1.2 provides a satisfactory generalization of the expander Chernoff bound to the case when one is interested in the spectral norm of a matrix-valued function on V, one could ask what happens for other matrix norms (such as Schatten norms), or even more generally, for functions taking values in an arbitrary Banach space. Our second contribution is a generic reduction from this problem, of proving concentration for random variables sampled using a Markov chain, to the much more well-studied problem (see e.g. [CL06]) of concentration for sums of *martingale* random variables. We remark that the technique used in the proof of Theorem 7.1.6 is not new, and was introduced in a slightly different context in the paper [NPSS06].

Theorem 7.1.6. *Suppose $G = (V, E)$ is a regular graph whose transition matrix has second eigenvalue λ and $f : V \to \mathbb{R}^N$ is a vector-valued function satisfying $\sum_{v \in V} f(v) = 0$ with $F := \sqrt{\sum_{v \in V} \|f(v)\|_2^2}$, where $\|\cdot\|_2$ denotes the Frobenius norm. If $v_1, \ldots, v_k$ is a stationary random walk on G, then for every $\epsilon > 0$, there is a martingale difference sequence $Z_1, \ldots, Z_k$ with respect to the filtration generated by initial segments of $v_1, \ldots, v_k$ such that*

$$\frac{1}{k} \sum_{i=1}^{k} f(v_i) = W + \frac{1}{k} \sum_{i=1}^{k} Z_i,$$

where

1. *W is a random vector satisfying $\|W\|_2 \leq \epsilon$.*

2. Each term Z_i satisfies

$$\|Z_i\|_* \leq \frac{2\log(F/\epsilon)}{1-\lambda} \cdot \max_{v \in V} \|f(v)\|_*$$

for every norm $\|\cdot\|_*$.

Thus, the empirical sums of any bounded (in any norm) function on a graph are well-approximated by a martingale whose increments are also bounded, with a loss in the bound depending on the ℓ_2 norm F of the function and the spectral gap of the graph. Since F will typically scale with the number of vertices, the ratio above is typically comparable to the mixing time.

To see the theorem in action, consider the case when $f(v)$ is matrix-valued in $d \times d$ Hermitian matrices and $\|\cdot\|_*$ is the operator norm. If $\|f(v)\| \leq 1$ then we have the bound

$$F^2 = \sum_{v \in V} \|f(v)\|_F^2 \leq dn.$$

Suppose we are interested in obtaining an estimate on the probability:

$$\Pr\left[\left\|\frac{1}{k}\sum_{v \in V} f(v)\right\| > \epsilon\right]. \tag{7.5}$$

Applying Theorem 7.1.6 with parameter $\epsilon/2$ and noting that $\|W\| \leq \|W\|_F$, we have that (7.5) is at most

$$\Pr\left[\left\|\frac{1}{k}\sum_{i=1}^{k} Z_i\right\| > \epsilon/2\right],$$

where Z_i is a martingale with bound

$$\|Z_i\| \leq \frac{\log(nd/\epsilon)}{1-\lambda}.$$

We now appeal to the existing martingale generalization of (7.2) (see e.g. [Tro12]) and find that this probability is at most

$$2d \cdot \exp\left(-\Omega\left(\frac{k\epsilon^2(1-\lambda)^2}{\log^2(nd)}\right)\right).$$

While this theorem is much weaker than the previous one in terms of parameters (depending on the square of the mixing time rather than on the spectral gap), it shows qualitatively that concentration for Markov chains is a generic phenomenon rather than something specific to matrices. It also allows one to instantly import the wealth of results regarding concentration for martingales in various Banach spaces (see e.g., [LT13]) to the random walk setting, albeit with suboptimal parameters. The simple proof of Theorem 7.1.6 is presented in Section 7.5.

7.2 Preliminaries

For an $n \in \mathbb{N}_+$, let $[n]$ denote the set $\{1, 2, \cdots, n\}$. Let $\mathbf{i}$ denote $\sqrt{-1}$. For $z = a + \mathbf{i}b$, where $a, b \in \mathbb{R}$, we define the complex conjugate of z to be $\overline{z} = a - \mathbf{i}b$ and $|z| = \sqrt{a^2 + b^2}$. Then $|z|^2 = z\overline{z}$. We define real part $\mathrm{Re}(z) = a$ and imaginary part $\mathrm{Im(z)} = b$. Then $\mathrm{Re}(z) = \frac{z+\overline{z}}{2}$ and $\mathrm{Im}(z) = \frac{z-\overline{z}}{2\mathbf{i}}$.

We will be working with D-regular undirected graphs $G = (V, E)$. The number of vertices of the graph, V will be denoted by n. A will denote the adjacency matrix of the graph and $P = A/D$ will denote its normalized adjacency matrix. A regular graph G will be called a λ-expander ($0 < \lambda < 1$) if $\|Px\| \leq \lambda \cdot \|x\|$ for all vectors $x \in \mathbb{C}^n$ s.t. $\sum_{i=1}^{n} x_i = 0$.

We will use $e_i \in \mathbb{C}^d$ to denote the standard basis vector with 1 in i^{th} position and 0 everywhere else.

7.2.1 Linear Algebra

Matrices and Norms. For matrix A, we use $A^\top$ to denote the transpose of A, we use $\overline{A}$ to denote the entry-wise complex conjugate of A. For square matrix $A \in \mathbb{C}^{n\times n}$, we use A^* to denote the conjugate transpose of matrix A. It is obvious that $A^* = \overline{A}^\top = \overline{A^\top}$. We say a complex square matrix A is Hermitian, if $A = A^*$, unitary if $AA^* = A^*A = I$, and positive-semidefinite (psd) if $A = A^*$ and $x^*Ax \geq 0$ for all $x \in \mathbb{C}^n$. We use $\succeq, \preceq$ to denote the semidefinite ordering, e.g. $A \succeq 0$ means that A is psd.

For $p \in [1, \infty)$, define the Schatten p-norm of A as

$$\|A\|_p = \left(\sum_{n \geq 1} s_n^p(A) \right)^{1/p}$$

for $s_1(A) \geq s_2(A) \geq \cdots \geq s_n(A) \geq \cdots \geq 0$ the singular values of A, i.e., the eigenvalues of the Hermitian matrix $|A| = \sqrt{(A^*A)}$. Then $\|A\|_p^p = \mathrm{tr}[|T|^p]$.

For matrix $A \in \mathbb{C}^{n\times n}$, we define $\|A\|$ to be the spectral norm of A, i.e.,

$$\|A\| = \max_{\|x\|_2=1, x\in\mathbb{C}^n} x^*Ax.$$

Tensor Products. Given two vectors $v \in \mathbb{C}^{d_1}$ and $w \in \mathbb{C}^{d_2}$, their tensor product $v \otimes w \in \mathbb{C}^{d_1 d_2}$ is the vector whose $(i,j)^{\text{th}}$ entry is $v(i)w(j)$ (for concreteness, assume the entries are in lexicographic order). Given two matrices $A_1 \in \mathbb{C}^{d_1\times d_1}$ and $A_2 \in \mathbb{C}^{d_2\times d_2}$, their tensor product $A_1 \otimes A_2 \in \mathbb{C}^{d_1 d_2 \times d_1 d_2}$ is the matrix whose $((i,k),(j,l))^{\text{th}}$ entry is $A_1(i,j)A_2(k,l)$. It is easy to see that

$$(A \otimes B)(v \otimes w) = Av \otimes Bw$$

and

$$(A \otimes B)(C \otimes D) = AB \otimes CD.$$

For a matrix $X \in \mathbb{C}^{d\times d}$, $\mathrm{vec}(X) \in \mathbb{C}^{d^2}$ will denote the vectorized version of the matrix X. That is

$$\mathrm{vec}(X) = \sum_{i,j=1}^{d} X(i,j) e_i \otimes e_j.$$

We have the following relationship between matrix multiplication and the tensor product:

$$\mathrm{vec}(AXB) = (A \otimes B^\top)\mathrm{vec}(X).$$

Exponential and Logarithm. All logarithms will be taken with the base e, and $\exp(x)$ will denote e^x. The matrix exponential of a complex matrix $A \in \mathbb{C}^{d\times d}$ is defined by the

Taylor expansion:

$$\exp(A) = \sum_{j=0}^{\infty} \frac{A^j}{j!},$$

which converges for all matrices A. We will use the fact that

$$\exp(A) \otimes \exp(B) = \exp(A \otimes I + I \otimes B),$$

which may be checked by expanding both sides and comparing terms.

The matrix logarithm of a positive definite matrix $A = UDU^*$ with D diagonal and positive is defined by

$$\log(A) := U \log(D) U^*,$$

where the logarithm of D is taken entrywise. For such matrices we have

$$\log(\exp(A)) = \exp(\log(A)) = A.$$

For positive definite A and complex z, we define
$A^z := \exp(z \log(A))$.

Polar Decomposition. The polar decomposition of a square complex matrix A is a matrix decomposition of the form

$$A = UV$$

where U is a unitary matrix and V is a psd matrix. The polar decomposition separates matrix A into a component that stretches the space along a set of orthogonal axes, represented by V, and a rotation (with possible reflection) represented by U. The decomposition of the complex conjugate of matrix A can written as

$$\overline{A} = \overline{UV}.$$

The decomposition of the conjugate transpose of matrix A can be written as

$$A^* = V^* U^*.$$

The following simple proposition will be useful in our proofs.

Proposition 7.2.1. *Let A and B be Hermitian psd matrices. Then*

$$\mathrm{tr}[AB] \leq \|A\| \cdot \mathrm{tr}[B]$$

Proof. Let $B = \sum_{j=1}^{n} \sigma_j v_j v_j^\dagger$ be the eigenvalue decomposition of B. Then

$$\begin{aligned}
\mathrm{tr}[AB] &= \sum_{j=1}^{n} \sigma_j \,\mathrm{tr}\left[A v_j v_j^\dagger\right] \\
&= \sum_{j=1}^{n} \sigma_j v_j^\dagger A v_j \\
&\leq \sum_{j=1}^{n} \sigma_j \cdot \|A\| \\
&= \|A\| \cdot \mathrm{tr}[B].
\end{aligned}$$

□

7.2.2 Complex analysis

A function $f : U \to \mathbb{C}$ on a domain $U \subseteq \mathbb{C}$ is *holomorphic* if it has a complex derivative in a neighborhood of every point $z \in U$. The existence of a complex derivative in a neighborhood is a very strong condition, for it implies that any holomorphic function is actually infinitely differentiable and equal to its own Taylor series (i.e., *analytic*) at every point in U. A *biholomorphic* function is a bijective holomorphic function whose inverse is

also holomorphic. It follows from the definition that sums, products, and compositions of holomorphic functions are holomorphic. We will also talk about matrix-valued holomorphic functions $f : U \to \mathbb{C}^{d\times d}$, which just means that every entry is holomorphic.

The main property that we will use is that the value of a holomorphic function at a point $z \in U$ can be related to values that it takes on the boundary of U, in the following way. A function $f : U \to \mathbb{R} \cup \{-\infty\}$ is called *subharmonic* if it is upper semicontinuous and

$$f(z) \leq \frac{1}{2\pi} \int_0^{2\pi} f(z + re^{i\theta}) d\theta$$

for all $z \in U$ and $r > 0$ such that the closed disk $D(z, r)$ is contained in U, and all of the above integrals converge.

We will make frequent use of the following standard fact.

Proposition 7.2.2. *If f is analytic on a domain $U \subset \mathbb{C}$ then $\log |f(z)|$ is subharmonic on U.*

Our main tool will be the Poisson Integral Formula for subharmonic functions.

Lemma 7.2.3 (Poisson integral formula on unit disk [Gao08, Eq 1.3.35])**.** *For any subharmonic function U defined on the unit disk $\{z \in \mathbb{C} : |z| \leq 1\}$, we have that*

$$U(z) \leq \frac{1}{2\pi} \int_{-\pi}^{\pi} U(e^{\mathbf{i}\varphi}) \frac{1 - |z|^2}{|e^{\mathbf{i}\varphi} - z|^2} \mathrm{d}\varphi, \qquad \forall\ |z| < 1.$$

7.3 New Golden-Thompson inequality

We begin by giving an outline of the proof of Theorem 7.1.5. The proof of Theorem 7.1.4 in [SBT17] relies on the multivariate Lie-Trotter product formula (e.g. see [Bha97]), which states that:

$$\exp\left(\sum_{j=1}^{k} L_j\right) = \lim_{\theta \to 0^+} \left(\prod_{j=1}^{k} \exp(\theta L_j)\right)^{\frac{1}{\theta}}.$$

For Hermitian L_j, a judicious application of the above allows one to rewrite the trace of the exponential as a limit of Schatten norms:

$$\log\left(\operatorname{tr}\left[\exp\left(\sum_{j=1}^{k} L_j\right)\right]\right) = \lim_{\theta \to 0^+} \frac{2}{\theta} \log \left\| \prod_{j=1}^{k} \exp\left(\frac{\theta}{2} L_j\right) \right\|_{2/\theta}.$$

Thus, understanding the matrix exponential of a sum is the same as understanding the behavior of a certain norm of the product as $\theta \to 0$. The idea of [SBT17] is to use *complex interpolation*, along the lines of the Stein-Hirschman theorem in complex analysis: for every fixed real θ near zero, find a complex function $F_\theta(z)$ that agrees with the right hand side at $z = \theta$ and is holomorphic on the strip $\{0 \leq \Re(z) \leq 1\}$. Since the value of a holomorphic function at any point can be related to an integral of its values on the boundary, this allows one to relate $F_\theta(\theta)$ to its integrals on $\{\Re(z) = 0\}$ and $\{\Re(z) = 1\}$, which are easy to understand. Taking the limit in θ yields Theorem 7.1.4.

To avoid integration on the whole vertical line $\{1 + \mathbf{i}b : b \in \mathbb{R}\}$, we observe that the above strategy only relies on the fact that θ is enclosed by the two vertical lines $\{\mathbf{i}b : b \in \mathbb{R}\}$ and $\{1 + \mathbf{i}b : b \in \mathbb{R}\}$, and we could have used any other region enclosing a neighborhood of real positive θ near zero, provided we can define the required holomorphic functions F_θ. We choose the half-circle (which is easy to work with because the Riemann map to the unit

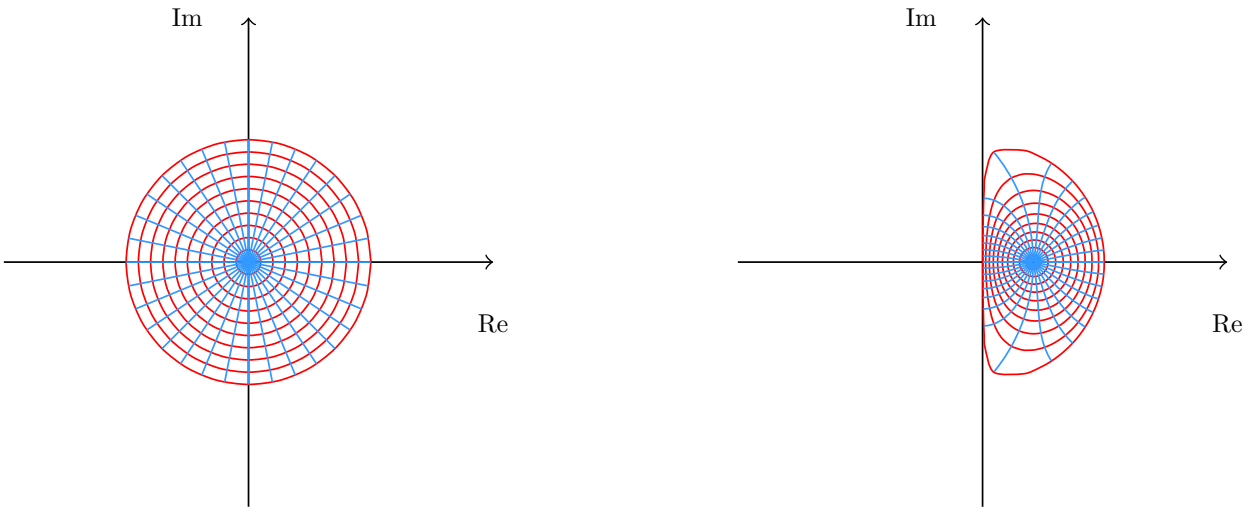


Figure 7.1: The function $h(z) = -\frac{1+z}{1-z} + \sqrt{\left(\frac{1+z}{1-z}\right)^2 + 1}$ maps the unit disk $\{z \in \mathbb{C} : |z| \leq 1\}$ to the half disk $\{z \in \mathbb{C} : |z| \leq 1 \text{ and } \mathrm{Re}(z) \geq 0\}$.

disk is explicit) and use it to derive a variant of the Riesz–Thorin theorem (Theorem 7.3.3), from which our new multi-matrix Golden-Thompson inequality follows by mimicking the remainder of the proof of 7.1.4 given in [SBT17].

7.3.1 Complex estimate on the half disk

In general, we can upper bound the value of any subharmonic function on a simply connected domain by mapping the domain to the unit disk via Riemann mapping theorem and applying the Poisson integral formula. In this section, we will give such estimate on a unit half disk.

The following lemma follows from the biholomorphic map from unit disk onto the half disk defined in Figure 7.1.

Lemma 7.3.1 (Poisson Integral Formula on the Half-Disk)**.** *For any analytic function F on*

the half disk $\{z \in \mathbb{C} : |z| \leq 1 \text{ and } \mathrm{Re}(z) \geq 0\}$, *we have that*

$$\log|F(x)| \leq \frac{1}{2\pi}\int_{-\pi}^{\pi} \log\left|F \circ h(e^{\mathbf{i}\varphi})\right| \frac{1-\rho^2}{1-2\rho\cos(\varphi)+\rho^2}\mathrm{d}\varphi \tag{7.6}$$

for any $0 \leq x \leq 1$ *where*

$$h(z) = -\frac{1+z}{1-z} + \sqrt{\left(\frac{1+z}{1-z}\right)^2 + 1} \quad \text{and} \quad \rho = \frac{x^2+2x-1}{x^2-2x-1}.$$

Proof. Note that the function $h(z)$ is a biholomorphic map from the unit disk $\{z \in \mathbb{C} : |z| \leq 1\}$ to the half disk $\{z \in \mathbb{C} : |z| \leq 1 \text{ and } \mathrm{Re}(z) \geq 0\}$. (We provide a proof for completeness, see Lemma 7.6.1 [2])

Since F and h are holomorphic, $\log|F \circ h(z)|$ is subharmonic. Therefore, we can apply the Poisson integral formula for subharmonic functions (Lemma 7.2.3) and get

$$\begin{aligned}
&\log|F \circ h(z)| \\
&\leq \frac{1}{2\pi}\int_{-\pi}^{\pi} \log\left|F \circ h(e^{\mathbf{i}\varphi})\right| \frac{1-|z|^2}{|e^{\mathbf{i}\varphi}-z|^2}\mathrm{d}\varphi \\
&= \frac{1}{2\pi}\int_{-\pi}^{\pi} \log\left|F \circ h(e^{\mathbf{i}\varphi})\right| \frac{1-\rho^2}{1-2\rho\cos(\theta-\varphi)+\rho^2}\mathrm{d}\varphi
\end{aligned}$$

for $z = \rho e^{\mathbf{i}\theta}$.

Setting $x = h(z)$, we obtain that

$$z = \frac{x^2+2x-1}{x^2-2x-1}.$$

[2] The similar version is an exercise 4 in page 163 (Section VII) of [Con78], and also can be found here, https://math.stackexchange.com/questions/882147/find-a-conformal-map-from-semi-disk-onto-unit-disk

Therefore, we have that

$$\theta = 0 \text{ and } \rho = \frac{x^2 + 2x - 1}{x^2 - 2x - 1}.$$

This gives the desired result.

□

The following lemma allows us to conveniently study the behavior of certain analytic functions near zero. The idea is that when $|F(z)|$ is at most 1 on the imaginary axis, $\log|F(\theta)|$ should be close to 0 for small θ, and the value of $\log|F(\theta)|/\theta$ can be upper bounded by a suitable average of the values on the boundary of the half disk.

Lemma 7.3.2. *Given any analytic function F on the half disk $\{z \in \mathbb{C} : |z| \le 1$ and $\mathrm{Re}(z) \ge 0\}$. Suppose that $|F(\mathbf{i}y)| \le 1$ for all $y \in [-1, 1]$. Then, for any $0 \le \theta \le 1/4$, we have*

$$\log|F(\theta)| \le \left(\frac{4\theta}{\pi} + O(\theta^2)\right) \int_{-\pi/2}^{\pi/2} \log\left|F(e^{\mathbf{i}\phi})\right| \mathrm{d}\mu_\theta(\phi)$$

where μ_θ is some probability distribution on $[-\frac{\pi}{2}, \frac{\pi}{2}]$ depending only on θ, and $\mu_\theta \to$ some probability distribution as $\theta \to 0^+$.

Proof. Since $\log|F(\mathbf{i}y)| \le 0$ for all $y \in [-1, 1]$ and since $h(e^{\mathbf{i}\varphi})$ is imaginary with modulus at most 1 whenever $|\varphi| \le \pi/2$, we can ignore these φ in the integral (7.6), namely,

$$\begin{aligned} &\log|F(x)| \\ &\le \frac{1}{2\pi} \int_{-\pi}^{\pi} \log\left|F \circ h(e^{\mathbf{i}\varphi})\right| \frac{1 - \rho^2}{1 - 2\rho\cos(\varphi) + \rho^2} \mathrm{d}\varphi \\ &\le \frac{1}{2\pi} \int_{\pi/2 \le |\varphi| \le \pi} \log\left|F \circ h(e^{\mathbf{i}\varphi})\right| \frac{1 - \rho^2}{1 - 2\rho\cos(\varphi) + \rho^2} \mathrm{d}\varphi. \end{aligned} \tag{7.7}$$

To bound the right hand side, for $\pi/2 \le |\varphi| \le \pi$ and $0 \le \rho \le 1$, we can prove the following statement using elementary calculations (see Lemma 7.6.4),

$$\frac{1-\rho^2}{1-2\rho\cos(\varphi)+\rho^2} = \frac{1-\rho}{1-\cos(\varphi)} \pm O(1-\rho)^2.$$

Note that $\rho = \frac{\theta^2+2\theta-1}{\theta^2-2\theta-1} \ge 1-4\theta$ for all $0 \le \theta \le 1/4$. Therefore, we have that $0 \le \rho \le 1$ and

$$\frac{1-\rho^2}{1-2\rho\cos(\varphi)+\rho^2} \le \frac{4\theta}{1-\cos(\varphi)} + O(\theta^2).$$

Putting this inequality into (7.7), we have that

$$\begin{aligned} &\log|F(x)| \\ &\le \frac{1}{2\pi}\int_{\pi/2\le|\varphi|\le\pi} \left(\frac{4\theta}{1-\cos(\varphi)} + O(\theta^2)\right) \log|F\circ h(e^{\mathbf{i}\varphi})|\mathrm{d}\varphi. \end{aligned}$$

We now observe that

$$\begin{aligned} &\int_{\pi/2\le|\varphi|\le\pi} \frac{1}{1-\cos(\varphi)}\mathrm{d}\varphi \\ &= \int_{-\pi}^{-\pi/2} \frac{1}{1-\cos(\varphi)}\mathrm{d}\varphi + \int_{\pi/2}^{\pi} \frac{1}{1-\cos(\varphi)}\mathrm{d}\varphi \\ &= 2. \end{aligned}$$

Note that h maps $e^{\mathbf{i}\varphi}$ for $\pi/2 \le |\varphi| \le \pi$ to the boundary of the half disk ($[-\pi/2, \pi/2]$), and let μ_θ be some probability distribution on $[-\frac{\pi}{2}, \frac{\pi}{2}]$ depending only on θ. Then we can have

$$\log|F(\theta)| \le \left(\frac{4\theta}{\pi} + O(\theta^2)\right) \int_{-\pi/2}^{\pi/2} \log\left|F(e^{\mathbf{i}\phi})\right| \mathrm{d}\mu_\theta(\phi).$$

Note that $\mu_\theta \to$ some probability distribution as $\theta \to 0^+$. □

7.3.2 Bounded Multimatrix Golden-Thompson type inequality

Plugging our new complex estimate into the proof of Theorem 3.1 in [SBT17], we obtain the following Riesz-Thorin-type inequality. We give a complete proof for completeness.

Theorem 7.3.3 (Riesz-Thorin-type inequality)**.** *Let* $S = \{z \in \mathbb{C} : |z| \leq 1 \text{ and } \mathrm{Re}(z) \geq 0\}$ *and let* G *be a holomorphic map from* S *to square matrices. Let* $p_0 \geq p_1 \in [1, \infty]$*, for* $\theta \in (0,1)$*, define* p_θ *by*

$$\frac{1}{p_\theta} = \frac{1-\theta}{p_0} + \frac{\theta}{p_1}.$$

If $z \to \|G(z)\|_{p_{\mathrm{Re}(z)}}$ *is uniformly bounded on* S *and* $\|G(\mathbf{i}t)\|_{p_0} \leq 1$*, then for any* $0 \leq \theta \leq 1/4$*,*

$$\log \|G(\theta)\|_{p_\theta} \leq \left(\frac{4\theta}{\pi} + O(\theta^2)\right) \int_{-\pi/2}^{\pi/2} \log \|G(e^{\mathbf{i}\phi})\|_{p_1} \mathrm{d}\mu_\theta(\phi)$$

where μ_θ *is some probability distribution on* $[-\frac{\pi}{2}, \frac{\pi}{2}]$ *depending only on* θ*.*

Proof. To apply Lemma 7.3.2, we want to define a holomorphic function $F(z)$ such that

$$|F(\mathbf{i}t)| \leq 1 \ , \ |F(z)| \leq \|G(z)\|_{p_1} \text{ and } F(\theta) = \|G(\theta)\|_{p_\theta}.$$

Now, we describe how to define such $F(z)$. For $x \in [0,1]$, define q_x as the Hölder conjugate of p_x such that $p_x^{-1} + q_x^{-1} = 1$. Hence, using the definition of p_x in the statement, we have

$$\frac{1}{q_x} = \frac{1-x}{q_0} + \frac{x}{q_1}.$$

Now for our fixed $\theta \in (0,1)$, let $G(\theta) = UV$ be the polar decomposition of $G(\theta)$, where V is positive definite since $G(\theta)$ is always invertible, and U is unitary. Finally, we define $X(z)$

and $F(z)$ by

$$X(z)^* = (V/c)^{p_\theta(\frac{1-z}{q_0}+\frac{z}{q_1})}U^*, \text{ where } c = \|V\|_{p_\theta} = \|G(\theta)\|_{p_\theta}$$
$$F(z) = \mathrm{tr}[X(z)^* G(z)].$$

Note that $F(z)$ is holomorphic. Due to the renormalization c, we can show $\|X(x+\mathbf{i}y))\|_{q_x}^{q_x} = 1$ for all $x \in [0,1]$:

$$\begin{aligned}
\|X(x+\mathbf{i}y)\|_{q_x}^{q_x} &= \mathrm{tr}\left[\sqrt{X^*(x+\mathbf{i}y)X(x+\mathbf{i}y)}^{q_x}\right] \\
&= \mathrm{tr}\left[(V/c)^{q_x p_\theta(\frac{1-x}{q_0}+\frac{x}{q_1})}\right] \\
&= \mathrm{tr}\left[(V/c)^{p_\theta}\right] \\
&= 1.
\end{aligned}$$

where the first step follows by definition of $\|\cdot\|_p$, the second step follows by $U^*U = I$, the third step follows by $\frac{1}{q_x} = \frac{1-x}{q_0} + \frac{x}{q_1}$, and the last step follows by $c = \|V\|_{p_\theta}$.

Therefore, $F(z)$ is bounded on S as follows

$$|F(x+\mathbf{i}y)| \le \|X(x+\mathbf{i}y)\|_{q_x} \cdot \|G(x+\mathbf{i}y)\|_{p_x} \le \|G(x+\mathbf{i}y)\|_{p_x}.$$

Using this, it is obvious that

$$|F(\mathbf{i}t)| \le \|G(\mathbf{i}t)\|_{p_0} \le 1, \text{ and } |F(z)| \le \|G(z)\|_{p_{\mathrm{Re}(z)}} \le \|G(z)\|_{p_1}.$$

for all $z \in S$ where we used that $p_0 \ge p_{\mathrm{Re}(z)} \ge p_1$.

Finally, we verify that $F(\theta) = \|G(\theta)\|_{p_\theta}$:

$$\begin{aligned} F(\theta) &= \operatorname{tr}[X(\theta)^* G(\theta)] \\ &= \operatorname{tr}[(V/c)^{p_\theta(\frac{1-\theta}{q_0}+\frac{\theta}{q_1})} U^* \cdot UV] \\ &= \operatorname{tr}[(V/c)^{p_\theta(\frac{1-\theta}{q_0}+\frac{\theta}{q_1})} V] \\ &= \operatorname{tr}[c^{-p_\theta/q_\theta} V^{1+p_\theta/q_\theta}] \\ &= \operatorname{tr}[c^{1-p_\theta} V^{p_\theta}] \\ &= c^{1-p_\theta} c^{p_\theta} \\ &= \|G(\theta)\|_{p_\theta}, \end{aligned}$$

where the first step follows by definition of X and G, the second step follows by $U^*U = I$, the third step follows by $\frac{1}{q_\theta} = \frac{1-\theta}{q_0} + \frac{\theta}{q_1}$, the fourth step follows by $p_\theta/q_\theta = p_\theta(1 - 1/p_\theta) = p_\theta - 1$, the fifth step follows by $(\operatorname{tr}[V^{p_\theta}])^{1/p_\theta} = c$, and the last step follows by $c = \|G(\theta)\|_{p_\theta}$.

Hence, the statement follows from Lemma 7.3.2. □

Now, we are ready to prove our variant of multimatrix Golden-Thompson inequality. It follows from plugging in Theorem 7.3.3 into the proof of Theorem 3.5 in [SBT17]. We give a complete proof for completeness.

Theorem 7.3.4 (Multimatrix Golden-Thompson inequality)**.** *For any k Hermitian matrices $H_1, \cdots, H_k$, we have:*

$$\log\left(\operatorname{tr}\left[\exp\left(\sum_{j=1}^{k} H_j\right)\right]\right) \leq \frac{4}{\pi}\int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \log\left(\operatorname{tr}\left[\prod_{j=1}^{k}\exp\left(\frac{e^{\mathbf{i}\phi}}{2}H_j\right)\prod_{j=k}^{1}\exp\left(\frac{e^{-\mathbf{i}\phi}}{2}H_j\right)\right]\right)\mathrm{d}\mu(\phi)$$

where μ is some probability distribution on $[-\frac{\pi}{2}, \frac{\pi}{2}]$.

Proof. Define

$$G(z) = \prod_{j=1}^{k} \exp\left(\frac{z}{2} H_j\right).$$

Note that $\|G(\mathbf{i}y)\|_\infty = 1$ for all $y \in \mathbb{R}$. We now have:

$$\begin{aligned}
& \log\left(\operatorname{tr}\left[\exp\left(\sum_{j=1}^{k} H_j\right)\right]\right) \\
=\ & \log\left(\operatorname{tr}\left[\exp\left(\sum_{j=1}^{k} H_j/2 + \sum_{j=k}^{1} H_j/2\right)\right]\right) \\
=\ & \log\left(\operatorname{tr}\left[\lim_{\theta\to 0^+} (G(\theta)G(\theta)^*)^{1/\theta}\right]\right) \\
=\ & \lim_{\theta\to 0^+} \frac{2}{\theta} \log\left(\operatorname{tr}\left[(G(\theta)G(\theta)^*)^{1/\theta}\right]^{\theta/2}\right) \\
=\ & \lim_{\theta\to 0^+} \frac{2}{\theta} \log \|G(\theta)\|_{2/\theta} \\
\leq\ & \lim_{\theta\to 0^+} 2\left(\frac{4}{\pi} + O(\theta)\right) \int_{-\pi/2}^{\pi/2} \log \|G(e^{\mathbf{i}\phi})\|_2 \mathrm{d}\mu_\theta(\phi) \\
=\ & \lim_{\theta\to 0^+} \left(\frac{4}{\pi} + O(\theta)\right) \int_{-\pi/2}^{\pi/2} \log\left(\operatorname{tr}\left[G(e^{\mathbf{i}\phi})G(e^{\mathbf{i}\phi})^*\right]\right) \mathrm{d}\mu_\theta(\phi),
\end{aligned}$$

where the first step follows from by the Lie-Trotter formula, the second step follows from H_j are Hermitian, the third step follows from continuity of log away from 0, last inequality follows from Theorem 7.3.3 with $p_0 = \infty$ and $p_1 = 2$.

When $\theta \to 0^+$, $\mu_\theta(\phi)$ converges to some probability distribution μ. This completes the proof. □

Remark 7.3.1. We suspect the constant $4/\pi$ is tight and that any constant larger than one is unavoidable if we consider the maximum over a bounded domain inside the strip $\{z \in \mathbb{C} : 0 \leq \operatorname{Re}(z) \leq 1\}$.

7.4 Proof of Theorem 7.1.2

In this section we present the proof of Theorem 7.1.2. We restate it here (with explicit constants) for convenience.

Theorem 7.4.1. *Let G be a regular λ-expander on V and let f be a function $f : V \to \mathbb{C}^{d\times d}$ s.t.*

1. *For each $v \in V$, $f(v)$ is Hermitian and $\|f(v)\| \leq 1$.*
2. *$\sum_{v\in V} f(v) = 0$.*

If $v_1, \ldots, v_k$ is a stationary random walk on G, and $\epsilon \in (0, 1)$,

$$\Pr\left[\lambda_{\max}\left(\sum_{j=1}^{k} f(v_j)\right) \geq +k\epsilon\right] \leq d^{2-\pi/4} \cdot \exp\left(-\epsilon^2(1-\lambda)k/80\right),$$
$$\Pr\left[\lambda_{\min}\left(\sum_{i=1}^{k} f(v_j)\right) \leq -k\epsilon\right] \leq d^{2-\pi/4} \cdot \exp\left(-\epsilon^2(1-\lambda)k/80\right).$$

Remark 7.4.1. Depsite the exponent of d is different from Theorem 7.1.2, we note that since the left hand side (the probability) is at most 1, one can prove the same statement with any positive exponent by changing the constant 80.

Proof. Due to symmetry, it suffices to prove just one of the statements. Let $t > 0$ be a

parameter to be chosen later. Then

$$\begin{aligned}
&\Pr\left[\lambda_{\max}\left(\sum_{j=1}^{k} f(v_j)\right) \geq k\epsilon\right] \\
\leq\ &\Pr\left[\operatorname{tr}\left[\exp\left(t\sum_{j=1}^{k} f(v_j)\right)\right] \geq \exp(tk\epsilon)\right] \\
\leq\ &\frac{\mathbb{E}\left[\operatorname{tr}\left[\exp\left(t\sum_{j=1}^{k} f(v_j)\right)\right]\right]}{\exp(tk\epsilon)}.
\end{aligned} \tag{7.8}$$

The second inequality follows from Markov's inequality.

Now the question is how to bound $\mathbb{E}_{v_1,\cdots,v_k}[\operatorname{tr}[\exp(t\sum_{j=1}^{k} f(v_j))]]$. Using Theorem 7.3.4 and note that $\mu(\phi)$ is a probability distribution on $[-\frac{\pi}{2}, \frac{\pi}{2}]$, we have

$$\begin{aligned}
&\log\left(\operatorname{tr}\left[\exp\left(t\sum_{j=1}^{k} f(v_j)\right)\right]\right) \\
\leq\ &\frac{4}{\pi}\int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \log\operatorname{tr}\left[\prod_{j=1}^{k}\exp\left(\frac{e^{\mathbf{i}\phi}}{2}tf(v_j)\right)\prod_{j=k}^{1}\exp\left(\frac{e^{-\mathbf{i}\phi}}{2}tf(v_j)\right)\right]\mathrm{d}\mu(\phi) \\
\leq\ &\frac{4}{\pi}\log\int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \operatorname{tr}\left[\prod_{j=1}^{k}\exp\left(\frac{e^{\mathbf{i}\phi}}{2}tf(v_j)\right)\prod_{j=k}^{1}\exp\left(\frac{e^{-\mathbf{i}\phi}}{2}tf(v_j)\right)\right]\mathrm{d}\mu(\phi),
\end{aligned}$$

where the the second step follows by concavity of log function. This implies that

$$\begin{aligned}
&\operatorname{tr}\left[\exp\left(t\sum_{j=1}^{k} f(v_j)\right)\right] \\
\leq\ &\left(\int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \operatorname{tr}\left[\prod_{j=1}^{k}\exp\left(\frac{e^{\mathbf{i}\phi}}{2}tf(v_j)\right)\prod_{j=k}^{1}\exp\left(\frac{e^{-\mathbf{i}\phi}}{2}tf(v_j)\right)\right]\mathrm{d}\mu(\phi)\right)^{\frac{4}{\pi}}.
\end{aligned} \tag{7.9}$$

Note that $\|x\|_p \leq d^{1/p-1}\|x\|_1$ for $p \in (0,1)$, choosing $p = \pi/4$ we have

$$\left(\operatorname{tr}\left[\exp\left(\frac{\pi}{4}t\sum_{j=1}^{k} f(v_j)\right)\right]\right)^{\frac{4}{\pi}} \leq d^{4/\pi-1}\operatorname{tr}\left[\exp\left(t\sum_{j=1}^{k} f(v_j)\right)\right]. \tag{7.10}$$

Combining Eq. (7.9) and Eq. (7.10), we have

$$\operatorname{tr}\left[\exp\left(\frac{\pi}{4}t\sum_{j=1}^{k} f(v_j)\right)\right] \leq d^{1-\pi/4}\int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \operatorname{tr}\left[\prod_{j=1}^{k}\exp\left(\frac{e^{\mathbf{i}\phi}}{2}tf(v_j)\right)\prod_{j=k}^{1}\exp\left(\frac{e^{-\mathbf{i}\phi}}{2}tf(v_j)\right)\right]\mathrm{d}\mu(\phi). \tag{7.11}$$

The core of the proof is the following bound on the moment generating function-like expression that appears above, thinking of $\mathbf{i}\phi$ as $\gamma + \mathbf{i}b$ with $\gamma^2 + b^2 = 1$:

Lemma 7.4.2. *Let G be a regular λ-expander on V, let f be a function $f : V \to \mathbb{C}^{d\times d}$ and $\sum_{v\in V} f(v) = 0$, let $v_1, \cdots, v_k$ be a stationary random walk on G, for any $t > 0$, $\gamma \geq 0$, $b > 0$, $t^2(\gamma^2 + b^2) \leq 1$, and $t\gamma \leq \frac{1-\lambda}{4\lambda}$ we have*

$$\begin{aligned}&\mathbb{E}\left[\operatorname{tr}\left[\prod_{j=1}^{k}\exp\left(\frac{tf(v_j)(\gamma+\mathbf{i}b)}{2}\right)\prod_{j=k}^{1}\exp\left(\frac{tf(v_j)(\gamma-\mathbf{i}b)}{2}\right)\right]\right]\\ &\leq d\cdot\exp\left(kt^2(\gamma^2+b^2)(1+\frac{8}{1-\lambda})\right).\end{aligned}$$

Assuming this lemma, we can easily complete the proof of the theorem as:

$$
\begin{aligned}
&\mathop{\mathbb{E}}_{v_1,\cdots,v_k}\left[\operatorname{tr}\left[\exp\left(\frac{\pi}{4}t\sum_{j=1}^{k}f(v_j)\right)\right]\right] \\
&\leq d^{1-\pi/4}\mathop{\mathbb{E}}_{v_1,\cdots,v_k}\left[\int_{-\frac{\pi}{2}}^{\frac{\pi}{2}}\operatorname{tr}\left[\prod_{j=1}^{k}\exp\left(\frac{e^{\mathbf{i}\phi}}{2}tf(v_j)\right)\prod_{j=k}^{1}\exp\left(\frac{e^{-\mathbf{i}\phi}}{2}tf(v_j)\right)\right]\mathrm{d}\mu(\phi)\right] \\
&= d^{1-\pi/4}\int_{-\frac{\pi}{2}}^{\frac{\pi}{2}}\mathop{\mathbb{E}}_{v_1,\cdots,v_k}\left[\operatorname{tr}\left[\prod_{j=1}^{k}\exp\left(\frac{e^{\mathbf{i}\phi}}{2}tf(v_j)\right)\prod_{j=k}^{1}\exp\left(\frac{e^{-\mathbf{i}\phi}}{2}tf(v_j)\right)\right]\right]\mathrm{d}\mu(\phi) \\
&\leq d^{1-\pi/4}\int_{-\frac{\pi}{2}}^{\frac{\pi}{2}}d\exp\left(kt^2|e^{\mathbf{i}\phi}|^2(1+\frac{8}{1-\lambda})\right)\mathrm{d}\mu(\phi) \\
&= d^{2-\pi/4}\exp\left(kt^2(1+\frac{8}{1-\lambda})\right)\int_{-\frac{\pi}{2}}^{\frac{\pi}{2}}\mathrm{d}\mu(\phi) \\
&= d^{2-\pi/4}\exp\left(kt^2(1+\frac{8}{1-\lambda})\right), \qquad (7.12)
\end{aligned}
$$

where the first step follows by the Equation (7.11), the second step follows by swapping $\mathbb{E}$ and $\int$, the third step follows by Lemma 7.4.2 , the fourth step follows by $|e^{\mathbf{i}\phi}| = 1$, and the last step follows by $\int_{-\frac{\pi}{2}}^{\frac{\pi}{2}}\mathrm{d}\mu(\phi) = 1$.

Finally, putting it all together,

$$
\begin{aligned}
&\Pr_{v_1,\cdots,v_k}\left[\lambda_{\max}\left(\sum_{j=1}^{k}f(v_j)\right)\geq k\epsilon\right] \\
&\leq d^{2-\pi/4}\cdot\exp\left((4/\pi)^2kt^2\frac{9}{1-\lambda}-kt\epsilon\right) \\
&= d^{2-\pi/4}\cdot\exp\left((4/\pi)^2k\epsilon^2(1-\lambda)^2\frac{1}{36^2}\frac{9}{1-\lambda}-k\frac{(1-\lambda)\epsilon}{36}\epsilon\right) \\
&\leq d^{2-\pi/4}\cdot\exp\left(-k\epsilon^2(1-\lambda)/72\right).
\end{aligned}
$$

where the first step follows by Eq. (7.12) the second step follows by choosing $t = (1-\lambda)\epsilon/36$. □

We now give the proof of Lemma 7.4.2

Proof of Lemma 7.4.2. We start by writing the expected trace expression in terms of the transition matrix of the random walk. This is an analogue of a step which is common to most of the expander chernoff bound proofs in the scalar case. Let P be the normalized adjacency matrix of G and let $\widetilde{P} = P \otimes I_{d^2}$. Let E denote the $nd^2 \times nd^2$ block diagonal matrix where the v^{th} diagonal block is the matrix

$$M_v = \exp\left(\frac{tf(v)(\gamma + \mathbf{i}b)}{2}\right) \otimes \exp\left(\frac{tf(v)(\gamma - \mathbf{i}b)}{2}\right). \tag{7.13}$$

Then $\left(E\widetilde{P}\right)^k$ is an $nd^2 \times nd^2$ block matrix whose $(u,v)^{\text{th}}$ $(d^2 \times d^2)$ block is given by the matrix

$$\sum_{v_1,\ldots,v_{k-1}} P_{u,v_1} \cdot \left(\prod_{j=1}^{k-2} P_{v_j,v_{j+1}}\right) \cdot P_{v_{k-1},v} \cdot M_u \cdot \left(\prod_{j=1}^{k-1} M_{v_j}\right). \tag{7.14}$$

Let $z_0 \in \mathbb{C}^{nd^2}$ be the vector $\frac{\mathbf{1}}{\sqrt{n}} \otimes \text{vec}(I_d)$. Here $\mathbf{1}$ is the all 1's vector and $\text{vec}(I_d)$ is the vector form of the identity matrix. Then, by applying

$$\langle \text{vec}(I_d), A_1 \otimes A_2 \,\text{vec}(I_d)\rangle = \text{tr}\left[A_1 A_2^\top\right],$$

it follows that for a stationary random walk $v_1, \ldots, v_k$:

$$\begin{aligned}
&\mathbb{E}\left[\text{tr}\left[\prod_{j=1}^{k} \exp\left(\frac{tf(v_j)(\gamma + \mathbf{i}b)}{2}\right) \prod_{j=k}^{1} \exp\left(\frac{tf(v_j)(\gamma - \mathbf{i}b)}{2}\right)\right]\right] \\
=\ &\mathbb{E}\left[\left\langle \text{vec}(I_d), \prod_{i=1}^{k} M_{v_i} \text{vec}(I_d)\right\rangle\right] \\
=\ &\left\langle z_0, \left(E\widetilde{P}\right)^k z_0\right\rangle.
\end{aligned}$$

Hence we can focus our attention on $\left\langle z_0, \left(E\widetilde{P}\right)^k z_0 \right\rangle$.

Let $\mathbf{1}$ be the all 1's vector. For a vector $z \in \mathbb{C}^{nd^2}$, let $z^{\|}$ denote the component of z which lies in the subspace spanned by the d^2 vectors $\mathbf{1} \otimes e_i$, $1 \leq i \leq d^2$. Let $z^{\perp}$ denote the component in the orthogonal space. Letting $z_j = \left(E\widetilde{P}\right)^j z_0$, we are interested in bounding

$$\langle z_0, z_k \rangle = \langle z_0, z_k^{\|} \rangle \leq \|z_0\| \cdot \|z_k^{\|}\| = \sqrt{d} \cdot \|z_k^{\|}\|.$$

The following lemma is the analogue of the main lemma in Healy's proof for the scalar valued expander Chernoff bound [Hea08]. Roughly speaking, it tracks how much a vector can move in and out of the subspace we are interested in as the operator $E\widetilde{P}$ is applied.

Lemma 7.4.3. *Given four parameters $\lambda \in [0,1]$, $\gamma \geq 0$, $\ell \geq 0$, and $t > 0$. Let G be a regular λ-expander on V. Suppose each vertex v is assigned a matrix $H_v \in \mathbb{C}^{d^2 \times d^2}$ s.t. $\|H_v\| \leq \ell$ and $\sum_v H_v = 0$. Let P be the normalized adjacency matrix of G and let $\widetilde{P} = P \otimes I_{d^2}$. Let E denote the $nd^2 \times nd^2$ block diagonal matrix where the v-th diagonal block is the matrix $\exp(tH_v)$. Also suppose that $\|E\| = \max_{v \in V} \|\exp(tH_v)\| \leq \exp(\gamma t)$. Then for any $z \in \mathbb{C}^{nd^2}$, we have:*

1. $\|(E\widetilde{P}z^{\|})^{\|}\| \leq \alpha_1 \|z^{\|}\|$ *where* $\alpha_1 = \exp(t\ell) - t\ell$,
2. $\|(E\widetilde{P}z^{\|})^{\perp}\| \leq \alpha_2 \|z^{\|}\|$ *where* $\alpha_2 = \exp(t\ell) - 1$,
3. $\|(E\widetilde{P}z^{\perp})^{\|}\| \leq \alpha_3 \|z^{\perp}\|$ *where* $\alpha_3 = \lambda \cdot (\exp(t\ell) - 1)$,
4. $\|(E\widetilde{P}z^{\perp})^{\perp}\| \leq \alpha_4 \|z^{\perp}\|$ *where* $\alpha_4 = \lambda \cdot \exp(t\gamma)$.

Proof of Lemma 7.4.3. Part 1.

Note that

$$(E\widetilde{P}z^{\|})^{\|} = (Ez^{\|})^{\|}.$$

Let $\mathbf{1} \in \mathbb{R}^n$ denote all ones vector, suppose $z^{\|} = \mathbf{1} \otimes \mathbf{w}$ for some $\mathbf{w} \in \mathbb{C}^{d^2}$. Then $\|z^{\|}\| = \sqrt{n} \cdot \|\mathbf{w}\|$ and

$$(Ez^{\|})^{\|} = \mathbf{1} \otimes \left(\frac{1}{n} \sum_{v \in V} \exp(H_v)\mathbf{w} \right).$$

We can upper bound $\|\frac{1}{n}\sum_{v\in V}\exp(tH_v)\|$ in the following way,

$$\begin{aligned}
\left\| \frac{1}{n} \sum_{v \in V} \exp(tH_v) \right\| &= \left\| \frac{1}{n} \sum_{v \in V} \sum_{j=0}^{\infty} \frac{t^j H_v^j}{j!} \right\| \\
&= \left\| I + \frac{1}{n} \sum_{v \in V} \sum_{j=2}^{\infty} \frac{t^j H_v^j}{j!} \right\| \\
&\leq 1 + \frac{1}{n} \sum_{v \in V} \sum_{j \geq 2} \frac{t^j}{j!} \|H_v\|^j \\
&= 1 + \sum_{j \geq 2} \frac{(t\ell)^j}{j!} \\
&= \exp(t\ell) - t\ell,
\end{aligned}$$

where the first step follows by Taylor expansion, the second step follows by $\sum_{v\in V} H_v = 0$, the third step follows by triangle inequality, the fourth step follows by $|V| = n$ and $\|H_v\| \leq \ell$, and last step follows by Taylor expansion.

Thus,

$$\begin{aligned}
\|(Ez^{\|})^{\|}\| &= \sqrt{n} \left\| \frac{1}{n} \sum_{v \in V} \exp(tH_v) w \right\| \\
&\leq \sqrt{n} \|w\| (\exp(t\ell) - t\ell) \\
&= \|z^{\|}\| (\exp(t\ell) - t\ell).
\end{aligned}$$

Part 2. Note that $(E\widetilde{P}z^{\|})^{\perp} = (Ez^{\|})^{\perp} = ((E-I)z^{\|})^{\perp}$. and $(z^{\|})^{\perp} = 0$. We can upper bound $\|((E-I)z^{\|})^{\perp}\|$ in the following way,

$$\begin{aligned}
\|((E-I)z^{\|})^{\perp}\| &\leq \|(E-I)z^{\|}\| \\
&\leq \|E-I\| \cdot \|z^{\|}\| \\
&= \max_{v \in V} \|\exp(tH_v) - I\| \cdot \|z^{\|}\| \\
&= \max_{v \in V} \left\| \sum_{j=1}^{\infty} \frac{t^j}{j!} H_v^j \right\| \cdot \|z^{\|}\| \\
&\leq \left(\sum_{j=1}^{\infty} \frac{t^j \ell^j}{j!} \right) \cdot \|z^{\|}\| \\
&= (\exp(t\ell) - 1) \cdot \|z^{\|}\|,
\end{aligned}$$

where the second step follows by $\|Ax\| \leq \|A\| \cdot \|x\|$, the third step follows by definition of E, the fourth step follows by Taylor expansion, the fifth step follows by triangle inequality and $\|H_v\| \leq \ell$, and the last step follows by Taylor expansion.

Part 3. Note that $(E\widetilde{P}z^{\perp})^{\|} = ((E-I)\widetilde{P}z^{\perp})^{\|}$. This is because $(\widetilde{P}z^{\perp})^{\|} = 0$ since $\widetilde{P}$ preserves the property of being orthogonal to the space spanned by the vectors $\mathbf{1} \otimes e_i$ (these are the top eigenvectors of $\widetilde{P}$). Hence we can bound

$$\begin{aligned}
\left\| ((E-I)\widetilde{P}z^{\perp})^{\|} \right\| &\leq \left\| (E-I)\widetilde{P}z^{\perp} \right\| \\
&\leq \|E-I\| \cdot \left\| \widetilde{P}z^{\perp} \right\| \\
&\leq (\exp(t\ell) - 1) \cdot \lambda \cdot \left\| z^{\perp} \right\|.
\end{aligned}$$

Third inequality follows from the fact that $\|E-I\| \leq \exp(t\ell) - 1$ and that G is a λ-expander.

Part 4. We can bound

$$\|(E\widetilde{P}z^{\perp})^{\perp}\| \leq \|E\widetilde{P}z^{\perp}\| \leq \exp(t\gamma)\cdot\lambda\|z^{\perp}\|,$$

where the second step follows by $\|E\| \leq \exp(\gamma t)$ and G is a λ-expander.

□

We now use the above Lemma 7.4.3 to analyze the evolution of $z_j^{\|}$ and $z_j^{\perp}$. Recall the definition of H_v,

$$H_v = \frac{f(v)(\gamma + \mathbf{i}b)}{2} \otimes I_d + I_d \otimes \frac{f(v)(\gamma - \mathbf{i}b)}{2}.$$

which means

$$\begin{aligned}
\exp(tH_v) =&\ \exp\left(\frac{tf(v)(\gamma+\mathbf{i}b)}{2}\otimes I_d + I_d \otimes \frac{tf(v)(\gamma-\mathbf{i}b)}{2}\right)\\
=&\ \exp\left(\frac{tf(v)(\gamma+\mathbf{i}b)}{2}\right)\otimes \exp\left(\frac{tf(v)(\gamma-\mathbf{i}b)}{2}\right)\\
=&\ M_v,
\end{aligned}$$

where the the first step follows by definition of H_v, the second step follows by $\exp(A\otimes I_d + I_d \otimes B) = \exp(A)\otimes\exp(B)$, and the last step follows by Eq. (7.13).

We can upper bound $\|H_v\| \leq \sqrt{\gamma^2+b^2}$ and then set $\ell = \sqrt{\gamma^2+b^2}$. We can also upper bound $\|\exp(tH_v)\|$,

$$\begin{aligned}
\|\exp(tH_v)\| &= \|\exp(t\cdot \mathrm{Re}(H_v))\|\\
&= \left\|\exp\left(\gamma t\left(\frac{f(v)}{2}\otimes I_d + I_d\otimes\frac{f(v)}{2}\right)\right)\right\|\\
&\leq \exp(\gamma t).
\end{aligned}$$

Note that $\sum_{v\in V} H_v = 0$ since $\sum_{v\in V} f(v) = 0$.

Claim 7.4.4. $\|z_i^{\perp}\| \leq \frac{\alpha_2}{1-\alpha_4} \max_{j<i} \|z_j^{\|}\|$.

Proof.

$$\begin{aligned}
\|z_i^{\perp}\| &= \|(E\widetilde{P}z_{i-1})^{\perp}\| \\
&\leq \|(E\widetilde{P}z_{i-1}^{\|})^{\perp}\| + \|(E\widetilde{P}z_{i-1}^{\perp})^{\perp}\| \\
&\leq \alpha_2\|z_{i-1}^{\|}\| + \alpha_4\|z_{i-1}^{\perp}\| \\
&\leq (\alpha_2 + \alpha_2\alpha_4 + \alpha_2\alpha_4 + \cdots) \cdot \max_{j<i} \|z_j^{\|}\| \\
&\leq \frac{\alpha_2}{1-\alpha_4} \max_{j<i} \|z_j^{\|}\|,
\end{aligned}$$

where the first step follows by definition of z_i, the second step follows by triangle inequality, the third step follows by part 2 and 4 of Lemma 7.4.3. □

Claim 7.4.5. $\|z_i^{\|}\| \leq (\alpha_1 + \frac{\alpha_2\alpha_3}{1-\alpha_4}) \max_{j<i} \|z_j^{\|}\|$.

Proof.

$$\begin{aligned}
\|z_i^{\|}\| &= \|(E\widetilde{P}z_{i-1})^{\|}\| \\
&\leq \|(E\widetilde{P}z_{i-1}^{\|})^{\|}\| + \|(E\widetilde{P}z_{i-1}^{\perp})^{\|}\| \\
&\leq \alpha_1\|z_{i-1}^{\|}\| + \alpha_3\|z_{i-1}^{\perp}\| \\
&\leq \alpha_1\|z_{i-1}^{\|}\| + \alpha_3\frac{\alpha_2}{1-\alpha_4} \max_{j<i-1} \|z_j^{\|}\| \\
&\leq (\alpha_1 + \frac{\alpha_2\alpha_3}{1-\alpha_4}),
\end{aligned}$$

where the first step follows by definition of z_i, the second step follows by triangle inequality, the third step follows by part 1 and 3 of Lemma 7.4.3, the fourth step follows by Claim 7.4.4. □

Combining Claim 7.4.4 and Claim 7.4.5 gives

$$\|z_k^{\|}\| \leq (\alpha_1 + \frac{\alpha_2\alpha_3}{1-\alpha_4})^k \|z_0^{\|}\| = \sqrt{d} \cdot (\alpha_1 + \frac{\alpha_2\alpha_3}{1-\alpha_4})^k,$$

which implies that

$$\left\langle z_0, (E\widetilde{P})^k z_0 \right\rangle \leq d \cdot (\alpha_1 + \frac{\alpha_2\alpha_3}{1-\alpha_4})^k.$$

Now the question is how to bound $(\alpha_1 + \frac{\alpha_2\alpha_3}{1-\alpha_4})^k$.

We can upper bound α_1, $\alpha_2\alpha_3$ and α_4 in the following sense,

$$\alpha_1 = \exp(t\ell) - t\ell \leq 1 + t^2\ell^2 = 1 + t^2(\gamma^2 + b^2),$$

and

$$\alpha_2\alpha_3 = \lambda(\exp(t\ell) - 1)^2 \leq \lambda(2t\ell)^2 = 4\lambda t^2(\gamma^2 + b^2),$$

where the second step follows by $t\ell < 1$ (because $\exp(x) \leq 1 + 2x, \forall x \in [0, 1]$),

$$\alpha_4 = \lambda \cdot \exp(t\gamma) \leq \lambda(1 + 2t\gamma) \leq \frac{1}{2} + \frac{1}{2}\lambda,$$

where the second step follows by $t\gamma < 1$, and the third step follows by $t\gamma \leq (1-\lambda)/4\lambda$.

Thus,

$$\begin{aligned}(\alpha_1 + \frac{\alpha_2 \cdot \alpha_3}{1-\alpha_4})^k \leq & \left(1 + t^2(\gamma^2 + b^2) + \frac{4\lambda t^2(\gamma^2 + b^2)}{\frac{1}{2} - \frac{1}{2}\lambda}\right)^k \\ \leq & \exp\left(kt^2(\gamma^2 + b^2)(1 + \frac{8}{1-\lambda})\right).\end{aligned}$$

□

Remark 7.4.2. As is the case with Healy's proof [Hea08], our proof also works for the case when there are different mean zero functions $f_1, \ldots, f_k$ for the different steps of the walk and also when there are k λ-expanders $G_1, \ldots, G_k$ and the j^{th} step of the walk is taken according to G_j.

Remark 7.4.3. We suspect that with appropriate modifications, our proof should generalize to random walks on irregular undirected graphs (or reversible Markov chains) as was done for Healy's proof in [CLLM12].

Remark 7.4.4. Although we have stated the theorem for Hermitian matrices, the same result can be obtained for general matrices by a standard dilation trick, namely replacing every $d \times d$ matrix M that appears with the $2d \times 2d$ Hermitian matrix

$$\begin{bmatrix} 0 & M \\ M^* & 0 \end{bmatrix},$$

whose norm is always within a factor of two of M.

7.5 Proof of Theorem 7.1.6

Proof. Observe that for every $i = 2, \ldots, k$ we have

$$\mathbb{E}[f(v_i) \mid v_{i-1}] = \sum_{u \sim v_{i-1}} P(v_{i-1}, u) f(u) = Pf(v_{i-1}),$$

whence the random vectors

$$Y_i^{(1)} := f(v_i) - Pf(v_{i-1})$$

satisfy

$$\mathbb{E}[Y_i^{(1)} \mid v_1, \ldots, v_{i-1}] = 0$$

and thus form a martingale difference sequence with respect to the filtration generated by initial segments of $v_1, \ldots, v_k$. Denoting $Y_1^{(1)} := f(v_1)$, we can write the sum of interest as a martingale part plus a remainder, which is a sum of $k-1$ (i.e., one fewer) random variables:

$$S = \sum_{i=1}^{k} Y_i^{(1)} + \sum_{i=1}^{k-1} Pf(v_i).$$

Notice that Pf is also a mean zero function on G, and by Jensen's inequality we have

$$\|(Pf)(v)\|_* \leq M := \max_{v \in V} \|f(v)\|_* \qquad \text{for all } v.$$

The key point is that the remainder terms $Pf(v_i)$ are smaller on average than the original terms $f(v_i)$ in squared Euclidean norm, because P is a contraction orthogonal to the constant vector; in particular, by considering the action of P on each coordinate of f separately, we have:

$$\sum_v \|Pf(v)\|_2^2 \leq \lambda \cdot \sum_v \|f(v)\|_2^2. \tag{7.15}$$

Iterating this construction on the remainder a total of $T \leq k$ times, we obtain a sequence of martingales $1 \leq t \leq T$:

$$\begin{aligned} Y_1^{(t)} &:= P^{t-1} f(v_1) \\ Y_i^{(t)} &:= P^{t-1} f(v_i) - P^t f(v_{i-1}), i = 1, \ldots, k-(t-1) \end{aligned}$$

which are related to the original sum as:

$$S = \sum_{t=1}^{T} \sum_{i=1}^{k-t+1} Y_i^{(t)} + \sum_{i=1}^{k-T} P^T f(v_i).$$

Interchanging the order of summation, we find that the random matrices

$$Z_i := \sum_{t=1}^{\min\{(k+1-i),T\}} Y_i^{(t)}$$

themselves form a martingale difference sequence, with each

$$\|Z_i\|_* \leq \sum_t \|Y_i^{(t)}\|_* \leq TM.$$

We bound the error $W := \frac{1}{k} \sum_{i=1}^{k-T} (P^T f)(v_i)$ crudely as:

$$\begin{aligned} \|W\|_2 &\leq \frac{1}{k} \sum_{i=1}^{k-T} \|P^T f(v_i)\|_2 \\ &\leq \frac{k-T}{k} \sum_{v \in V} \|(P^T f)(v)\|_2 \\ &\leq \lambda^{T/2} F \\ &\leq \exp(-(1-\lambda)T/2) F, \end{aligned}$$

where $F := (\sum_{v \in V} \|f(v)\|_2^2)^{1/2}$, by applying (7.15). Rearranging and setting $T = 2\log(F/\epsilon)/(1-\lambda)$ yields the advertised bound on $\|Z_i\|_*$. □

7.6 Elementary calculations

Lemma 7.6.1. *We define function $h(z) : \mathbb{C} \to \mathbb{C}$ as follows*

$$h(z) = -\frac{1+z}{1-z} + \sqrt{\left(\frac{1+z}{1-z}\right)^2 + 1}.$$

Then function $h(z)$ maps the unit disk $\{z \in \mathbb{C} : |z| \leq 1\}$ to the half disk $\{z \in \mathbb{C} : |z| \leq 1 \text{ and } \mathrm{Re}(z) \geq 0\}$.

Proof. We first compute the inverse of function $h(z)$, let $f = h^{-1}$. By definition of $h(z)$, we can do the following elementary calculations,

$$\begin{aligned}
& h(z) + \frac{1+z}{1-z} = \sqrt{\left(\frac{1+z}{1-z}\right)^2 + 1} \\
\implies & \left(h(z) + \frac{1+z}{1-z}\right)^2 = \left(\frac{1+z}{1-z}\right)^2 + 1 \\
\implies & (h(z))^2 + 2h(z)\frac{1+z}{1-z} = 1 \\
\implies & (1-z)(h(z))^2 + 2h(z)(1+z) = (1-z) \\
\implies & (h(z))^2 + 2h(z) - 1 = z((h(z))^2 - 2h(z) - 1),
\end{aligned}$$

Thus, we obtain that

$$f(z) = \frac{z^2 + 2z - 1}{z^2 - 2z - 1}.$$

To finish the proof, we need Claim 7.6.2 and Claim 7.6.3.

Claim 7.6.2. *For all $z \in \{z \in \mathbb{C} : |z| \leq 1 \text{ and } \mathrm{Re}(z) \geq 0\}$,*

$$|f(z)| \leq 1.$$

Proof. Let $z = re^{\mathbf{i}\phi}$, where $r \in [0,1]$ and

$\phi \in [-\pi/2, \pi/2]$. It is easy to observe that $\cos(\phi) \in [0,1]$ We have

$$\begin{aligned} |f(z)| &= \left|\frac{z^2+2z-1}{z^2-2z-1}\right| \\ &= \left|\frac{r^2e^{\mathbf{i}2\phi}+2re^{\mathbf{i}\phi}-1}{r^2e^{\mathbf{i}2\phi}-2re^{\mathbf{i}\phi}-1}\right| \\ &= \frac{|r^2e^{\mathbf{i}2\phi}+2re^{\mathbf{i}\phi}-1|}{|r^2e^{\mathbf{i}2\phi}-2re^{\mathbf{i}\phi}-1|}. \end{aligned}$$

We can compute the numerator,

$$\begin{aligned} & |r^2e^{\mathbf{i}2\phi}+2re^{\mathbf{i}\phi}-1|^2 \\ &= |r^2\cos 2\phi + \mathbf{i}r^2\sin 2\phi + 2r\cos\phi + 2r\mathbf{i}\sin\phi - 1|^2 \\ &= (r^2\cos 2\phi + 2r\cos\phi - 1)^2 + (r^2\sin 2\phi + 2r\sin\phi)^2 \\ &= r^4 + 4r^2 + 1 - 2r^2\cos 2\phi + 4r^3\cos 2\phi\cos\phi \\ &\quad - 4r\cos\phi + 4r^3\sin 2\phi\sin\phi \\ &= r^4 + 4r^2 + 1 - 2r^2\cos 2\phi + 4r^3\cos\phi - 4r\cos\phi. \end{aligned}$$

We can compute the denominator,

$$\begin{aligned} & |e^{\mathbf{i}2\phi}-2e^{\mathbf{i}\phi}-1|^2 \\ &= |r^2\cos 2\phi + \mathbf{i}r^2\sin 2\phi - 2r\cos\phi - 2r\mathbf{i}\sin\phi - 1|^2 \\ &= (r^2\cos 2\phi - 2r\cos\phi - 1)^2 + (r^2\sin 2\phi - 2r\sin\phi)^2 \\ &= r^4 + 4r^2 + 1 - 2r^2\cos 2\phi - 4r^3\cos 2\phi\cos\phi \\ &\quad + 4r\cos\phi - 4r^3\sin 2\phi\sin\phi \\ &= r^4 + 4r^2 + 1 - 2r^2\cos 2\phi - 4r^3\cos\phi + 4r\cos\phi. \end{aligned}$$

Note that, in order to show $|f(z)| \leq 1$, it is sufficient to prove

$$4r^3 \cos\phi - 4r\cos\phi \leq -4r^3\cos\phi + 4r\cos\phi$$

which is equivalent to

$$8r(1-r^2)\cos\phi \geq 0.$$

It follows by definition of r and ϕ. Thus, we complete the proof. □

Next, we can show that

Claim 7.6.3. *For all z is on the boundary of half disk,*

$$|f(z)| = 1.$$

Proof. First, we want to show that $\forall z \in [-\mathbf{i}, \mathbf{i}]$, $|f(z)| = 1$. Let $b \in [0,1]$, let $z = \mathbf{i}b$, then we have

$$|f(z)| = |f(\mathbf{i}b)| = \left|\frac{-b^2 + 2b\mathbf{i} - 1}{-b^2 - 2b\mathbf{i} - 1}\right| = 1.$$

Second, we want to show that for all z on half circle, $|f(z)| = 1$. We replace z by $e^{\mathbf{i}\phi}$, where $\phi \in [-\pi/2, \pi/2]$. Then we have

$$\begin{aligned} |f(z)| &= \left|\frac{z^2 + 2z - 1}{z^2 - 2z - 1}\right| \\ &= \left|\frac{e^{\mathbf{i}2\phi} + 2e^{\mathbf{i}\phi} - 1}{e^{\mathbf{i}2\phi} - 2e^{\mathbf{i}\phi} - 1}\right| \\ &= \frac{|e^{\mathbf{i}2\phi} + 2e^{\mathbf{i}\phi} - 1|}{|e^{\mathbf{i}2\phi} - 2e^{\mathbf{i}\phi} - 1|}. \end{aligned}$$

We can compute the numerator,

$$\begin{aligned}
&|e^{\mathbf{i}2\phi} + 2e^{\mathbf{i}\phi} - 1| \\
&= |\cos 2\phi + \mathbf{i}\sin 2\phi + 2\cos\phi + 2\mathbf{i}\sin\phi - 1| \\
&= \left((\cos 2\phi + 2\cos\phi - 1)^2 + (\sin 2\phi + 2\sin\phi)^2\right)^{1/2} \\
&= (6 - 2\cos 2\phi + 4\cos 2\phi\cos\phi - 4\cos\phi + 4\sin 2\phi\sin\phi)^{1/2} \\
&= (6 - 2\cos 2\phi + 4\cos\phi - 4\cos\phi)^{1/2} \\
&= (6 - 2\cos 2\phi)^{1/2}.
\end{aligned}$$

We can compute the denominator,

$$\begin{aligned}
&|e^{\mathbf{i}2\phi} - 2e^{\mathbf{i}\phi} - 1| \\
&= |\cos 2\phi + \mathbf{i}\sin 2\phi - 2\cos\phi - 2\mathbf{i}\sin\phi - 1| \\
&= \left((\cos 2\phi - 2\cos\phi - 1)^2 + (\sin 2\phi - 2\sin\phi)^2\right)^{1/2} \\
&= (6 - 2\cos 2\phi - 4\cos 2\phi\cos\phi + 4\cos\phi - 4\sin 2\phi\sin\phi)^{1/2} \\
&= (6 - 2\cos 2\phi + 4\cos\phi - 4\cos\phi)^{1/2} \\
&= (6 - 2\cos 2\phi)^{1/2}.
\end{aligned}$$

Thus, we have $|f(z)| = 1$. □

Note the biholomorphic is basically follows from the formula of f and h, because they are composition of holomorphic function. □

Lemma 7.6.4. *For any $\rho \in [0,1]$ and $\cos\varphi \in [-1,0]$, we have*

$$\frac{1-\rho}{1-\cos\varphi} - (1-\rho)^2 \leq \frac{1-\rho^2}{1-2\rho\cos\varphi+\rho^2} \leq \frac{1-\rho}{1-\cos\varphi} + 2(1-\rho)^2.$$

Proof. This directly follows by combining Claim 7.6.5 and Claim 7.6.6 □

Claim 7.6.5. *There exists some sufficiently large constant $c \geq 1$ such that for any $\rho \in [0,1]$ and $\cos\varphi \in [-1,0]$, we have*

$$\frac{1-\rho^2}{1-2\rho\cos\varphi+\rho^2} \leq \frac{1-\rho}{1-\cos\varphi} + c(1-\rho)^2.$$

Proof. It is equivalent to

$$\begin{aligned}
\frac{1+\rho}{1-2\rho\cos\varphi+\rho^2} &\leq \frac{1}{1-\cos\varphi} + c(1+\rho) \\
(1+\rho)(1-\cos\varphi) &\leq 1-2\rho\cos\varphi+\rho^2 + c(1+\rho)(1-\cos\varphi)(1-2\rho\cos\varphi+\rho^2) \\
\rho-\cos\varphi &\leq -\rho\cos\varphi+\rho^2 + c(1+\rho)(1-\cos\varphi)(1-2\rho\cos\varphi+\rho^2)
\end{aligned}$$

which is equivalent to,

$$-\rho\cos\varphi+\rho^2-\rho+\cos\varphi + c(1+\rho)(1-\cos\varphi)(1-2\rho\cos\phi+\rho^2) \geq 0.$$

Since $1-2\rho\cos\varphi+\rho^2 \geq 1$, thus it suffices to show

$$(\rho-1)(\rho-\cos\varphi) + c(1+\rho)(1-\cos\varphi) \geq 0.$$

Note that $(\rho-1)(\rho-\cos\varphi) \geq -2$, by choosing $c \geq 2$, we complete the proof. □

Claim 7.6.6. *There exists some sufficiently large constant $c \geq 1$ such that for any $\rho \in [0,1]$ and $\cos\varphi \in [-1,0]$, we have*

$$\frac{1-\rho^2}{1-2\rho\cos\varphi+\rho^2} \geq \frac{1-\rho}{1-\cos\varphi} - c(1-\rho)^2.$$

Proof. It is equivalent to

$$\begin{aligned}
\frac{1+\rho}{1-2\rho\cos\varphi+\rho^2} &\geq \frac{1}{1-\cos\varphi} + c(1+\rho) \\
(1+\rho)(1-\cos\varphi) &\geq 1-2\rho\cos\varphi+\rho^2 - c(1+\rho)(1-\cos\varphi)(1-2\rho\cos\varphi+\rho^2) \\
\rho-\cos\varphi &\geq -\rho\cos\varphi+\rho^2 - c(1+\rho)(1-\cos\varphi)(1-2\rho\cos\varphi+\rho^2)
\end{aligned}$$

which is equivalent to,

$$-\rho\cos\varphi+\rho^2-\rho+\cos\varphi - c(1+\rho)(1-\cos\varphi)(1-2\rho\cos\phi+\rho^2) \leq 0$$

which is equivalent to

$$(\rho-1)(\rho-\cos\varphi) \leq c(1+\rho)(1-\cos\varphi)(1-2\rho\cos\varphi+\rho^2).$$

It suffices to choose $c=1$. □

Chapter 8

Matrix Chernoff Bound for Random Spanning Trees

Strongly Rayleigh distributions are a class of negatively dependent distributions of binary-valued random variables [Borcea, Brändén, Liggett JAMS 09]. Recently, these distributions have played a crucial role in the analysis of algorithms for fundamental graph problems, e.g. Traveling Salesman Problem [Gharan, Saberi, Singh FOCS 11]. We prove a new matrix Chernoff bound for Strongly Rayleigh distributions.

As an immediate application, we show that adding together the Laplacians of $\epsilon^{-2}\log^2 n$ random spanning trees gives an $(1\pm\epsilon)$ spectral sparsifiers of graph Laplacians with high probability. Thus, we positively answer an open question posed in [Baston, Spielman, Srivastava, Teng JACM 13]. Our number of spanning trees for spectral sparsifier matches the number of spanning trees required to obtain a cut sparsifier in [Fung, Hariharan, Harvey, Panigraphi STOC 11]. The previous best result was by naively applying a classical matrix Chernoff bound which requires $\epsilon^{-2}n\log n$ spanning trees. For the tree averaging procedure to agree with the original graph Laplacian in expectation, each edge of the tree should be reweighted by the inverse of the edge leverage score in the original graph. We also show that when using this reweighting of the edges, the Laplacian of single random tree is bounded above in the PSD order by the original graph Laplacian times a factor $\log n$ with high probability, i.e. $L_T \preceq O(\log n)L_G$.

We show a lower bound that almost matches our last result, namely that in some graphs, with high probability, the random spanning tree is *not* bounded above in the spectral order by $\frac{\log n}{\log \log n}$ times the original graph Laplacian.

We also show a lower bound that in $\epsilon^{-2} \log n$ spanning trees are necessary to get a $(1 \pm \epsilon)$ spectral sparsifier.

8.1 Introduction

The study of concentration of sums of random variables dates back to Central Limit Theorems, and hence de Moivre and Laplace [Tij], while modern concentration bounds for sums of random variables were perhaps first established by Bernstein [Ber24], and a popular variant now known as Chernoff bounds was introduced by Rubin and published by Chernoff [Che52].

Concentration of measure for matrix-valued random variables is the phenomenon that many matrix valued distributions are to close their mean with high probability, closeness usually being measured by spectral norm. Modern quantitative bounds of the form often used in theoretical computer science were derived by Rudelson [Rud99], while Ahlswede and Winter [AW02] established a useful matrix-version of the Laplace transform that plays a central role in scalar concentration results such as those of Bernstein. [AW02] combined this with the Golden-Thompson trace inequality to prove matrix concentration results. Tropp refined this approach, and by replacing the use of Golden-Thompson with deep a theorem on concavity of certain trace functions due to Lieb, Tropp was able to recover strong versions of a wide range of scalar concentration results, including matrix Chernoff bounds, Azuma and Freedman's inequalities for matrix martingales [Tro12].

Matrix concentration results have had an enormous range of applications in computer science, and are ubiquitous throughout spectral graph theory [ST04, SS11, CKP+17], sketching [Coh16a], approximation algorithms [HSSS16], and deep learning [ZSJ+17, ZSD17, SY19]. Most applications are based on results for independent random matrices, but more flexible bounds, such as Tropp's Matrix Freedman Inquality [Tro11a], have been used to greatly simplify algorithms, e.g. for solving Laplacian linear equations [KS16] and for semi-streaming

graph sparsification [AG09, KPPS17]. Matrix concentration results are also closely related to other popular tools sampling tools, such as Karger's techniques for generating sparse graphs that approximately preserve the cuts of denser graphs [BK96].

Negative dependence of random variables is an appealing property that intuition suggests should help with concentration of measure. Notions of negative dependence can be formalized in many ways. Roughly speaking, these notions characterize distributions where where some event occurring ensures that other events of interest become less likely. A simple example is the distribution of a sequence of coin flips, conditioned on the total number of heads in the outcome. In this distribution, conditioning on some coin coming out heads makes all other coins less likely to come out heads. Unfortunately, negative dependence phenomena are not as robust as positive association which can be established from local conditions using the powerful FKG theorem [FKG71].

Strongly Rayleigh distributions were introduced recently by Borcea, Brändén, and Liggett [BBL09] as a class of negatively dependent distributions of binary-valued random variables with many useful properties. Strongly Rayleigh distributions satisfy useful negative dependence properties, and retain these properties under natural conditioning operations. Strongly Rayleigh distributions also satisfy a powerful stability property under conditioning known as *Stochastic Covering* [PP14], which is useful for analyzing them through martingale techniques. A measure on $\{0,1\}^n$ is said to be Strongly Rayleigh if its generating polynomial is real stable [BBL09]. There are many interesting examples of Strongly Rayleigh distributions [PP14]: The example mentioned earlier of heads of independent coin flips conditional on the total number of heads in the outcome; symmetric exclusion processes; determinental point processes and determinental measures on a boolean lattice. An example of particu-

lar interest to us is the edges of uniform or weighted random spanning trees, which form a Strongly Rayleigh distribution.

We prove a Matrix Chernoff bound for the case of k-homogeneous Strongly Rayleigh distributions. Our bound is slightly weaker than the bound for independent variables. We give lower bounds that show our bounds are close to tight in some regimes, but importantly, our lower bounds do not establish separation from the behaviour of indepedent random matrices, leaving open the question of whether the true bound should match the independent case in all regimes – which seems plausible. We use our bound to show new concentration results related to random spanning trees of graphs. An open question is to find other interesting applications of our concentration result, e.g. by analyzing concentration for matrices generated by exclusion processes.

Random spanning trees are one among the most well-studied probabilistic objects in graph theory, going back to the work of Kirchoff [Kir47] in 1847, who gave formula relating the number of spanning trees in a graph to the determinant of the Laplacian of the same graph.

Algorithms for sampling of random spanning trees have been studied extensively, [Gue83, Bro89, Ald90, Kul90, Wil96, CMN96, KM09, MST15, HX16, DKP+17, DPPR17, Sch18], and a random spanning tree can now be sampled in almost linear time [Sch18].

In theoretical computer science, random spanning trees have found a number of applications, most notably in breakthrough results on approximating the traveling salesperson problem with symmetric [GSS11] and asymmetric costs [AGM+10]. Goyal et al. [GRV09] demonstrated that adding just two random spanning trees sampled from a bounded degree

graph gives a $O(\log n)$ cut sparsifier with probability $1 - o(1)$. Later, it was shown by Fung, Hariharan, Harvey, Panigraphi [FHHP11], that if we sample $O(\epsilon^{-2} \log^2 n)$ random spanning trees from a graph, reweight the tree edges by the inverse of their leverage scores in the original graph, and average them together, then whp. we get a graph where every the weight of edges crossing every cut is approximately the same in as in the original graph, up to a factor $(1 \pm \epsilon)$. We refer to this as an ϵ-cut sparsifier. The techniques of Fung et al. unfortunately do not extend to proving spectral sparsifiers.

Spectral graph sparsifiers were introduced by Spielman and Teng [ST04], who for any graph G showed how to construct a another graph H with $\epsilon^{-2} n \operatorname{poly} \log n$ edges s.t. $(1 - \epsilon) L_G \preceq L_H \preceq (1 + \epsilon) L_G$, which we refer to as an ϵ-spectral sparsifier. The construction was refined by Spielman and Srivastava [SS11], who suggested sampling edges independently[1] with probability proportional to their *leverage scores*, and brought the number of required samples down to $\epsilon^{-2} n \log n$. This analysis is tight in the sense that if fewer than $o(\epsilon^{-2} n \log n)$ samples are used, there will be at least a $1/\operatorname{poly}(n)$ probability of failure. Meanwhile, $\epsilon^{-2} n \frac{\log n}{\log \log n}$ independent samples in a union of cliques can be shown whp. to fail to give a cut sparsifier. This can be observed directly from the degree distribution of a single vertex in the complete graph. For a variant of [SS11] sampling based on flipping a single coin for each edge to decide whether to keep it or not, it can also be shown that when the expected number of edges is $\epsilon^{-2} n \frac{\log n}{\log \log n}$, whp. the procedure fails to give a cut sparsifier. For arbitrary sparsification schemes, bounds in [BSS12] show that $\Theta(\epsilon^{-2} n)$ edges are necessary and sufficient to give an ϵ-spectral sparsifier.

[1] [SS11] analyzed sampling with replacement, but based on [Tro12], a folklore result shows the same behavior can be obtained by doing independent coin flips for every edge with low leverage score, again with inclusion probabilities proportional to leverage scores.

The marginal probability of an edge being present in a random spanning tree is exactly the leverage score of the edge. This seems to suggest that combining $\epsilon^{-2} \operatorname{poly} \log n$ spanning trees might give a spectral sparsifier, but the lack of independence between the sampled edges means the process cannot be analyzed using existing techniques. Observing this, Baston, Spielman, Srivastava, Teng [BSST13] in their excellent 2013 survey on sparsification noted that *"it remains to be seen if the union of a small number of random spanning trees can produce a spectral sparsifier."* We answer this question in the affirmative. In particular, we show that adding together $O(\epsilon^{-2} \log^2 n)$ spanning trees with edges scaled proportional to inverse leverage scores in the original graph leads to an ϵ-spectral sparsifier. This matches the bound obtained for cut sparsifiers in [FHHP11]. Our result also implies their earlier bound since a spectral sparsifier is always a cut sparsifier with the same approximation quality. Before our result, only a trivial bound on the number of spanning trees required to build a spectral sparsifier was known. In particular standard matrix concentration arguments like those in [SS11] prove that $O(\epsilon^{-2} n \log n)$ spanning trees suffice. Lower bounds in [FHHP11] show that whp. $\Omega(\log n)$ random spanning trees are required to give a constant factor spectral sparsifier.

We show that whp. $\epsilon^{-2} \frac{\log n}{\log \log n}$ random spanning trees do not give an ϵ-spectral sparsifier. We also show that the Laplacian of a single random tree with edges weighted as above satisfies $L_T \preceq O(\log n) L_G$ whp., and we give an almost matching lower bound, showing that in some graphs whp. $L_T \not\preceq \frac{1}{8} \frac{\log n}{\log \log n} L_G$. Before our work, the main result known about approximating graphs using $O(1)$ random spanning trees is due to Goyal, Rademacher, Vempala [GRV09], who showed that surprisingly, when the original graph has bounded degree, adding two random spanning trees gives a graph whose cuts approximate the cuts in the

original graph up to a factor $O(\log n)$ with good probability. As our result for a single tree establishes only a one-sided bound an interesting open question remains: Does sampling $O(1)$ random spanning trees give a $\log n$-factor spectral sparsifier with, say, constant probability?

8.1.1 Previous work

Chernoff-type bound for matrices. Chernoff-like bounds for matrices appear in Rudelson [Rud99] and Ahlswede and Winter [AW02]. The latter introduced a useful matrix-variant of the Laplace transform that is central in concentration bounds for scalar-valued matrices. Their bounds restricted to iid random matrices, an artifact of their use of the Golden-Thompson inequality for bounding traces. In contrast, Tropp obtained more flexible concentration bounds for random matrices by using a result of Lieb to bound the expected trace of various operators [Tro12], including bounds for matrix martingales [Tro11c].

In a recent work by Garg, Lee, Song and Srivastava [GLSS18], they show a Chernoff bound for sums of matrix-valued random variables sampled via a random walk on an expander graph. This work confirms a conjecture due Wigderson and Xiao. The proof of Garg et al. is also concerned with matrices that are not fully independent. In this case the matrices are generated from random walks on an expander graph. The main idea to deal with dependence issue is using a new multi-matrix extension of the Golden-Thompson inequality and an adaptation of Healy's proof of the expander Chernoff bound in the scalar's case [Hea08] to matrix case. Their techniques deal with fairly generic types of dependence, and cannot leverage the very strong stability properties that arise from the negative dependence and stochastic covering properties of Strongly Rayleigh distributions. Harvey and Olver [HO14] proved a matrix concentration result for randomized pipage rounding, which can be used to

show concentration results for random spanning trees obtained from pipage rounding, but not for (weighted) uniformly random spaning trees. The central technical element of their proof is a new variant of a theorem of Lieb on concavity of certain matrix trace functions.

Matrix martingales have played a central role in a number of algorithmic results in theoretical computer science [KS16, CMP16, KPPS17], but beyond a reliance on Tropp's Matrix Freedman Inequality, these works have little in common with our approach. However, our bound does share a technical similarity with [KS16], namely that a sequence of increasingly restricted random choices in a martingale process lead to a $\log n$ factor in a variance bound.

Strongly Rayleigh Distributions in Theoretical Computer Science. Perhaps the most prominent result on Strongly Rayleigh distributions in theoretical computer science is the generalization of [MSS15b] to Strongly Rayleigh distributions.

The central technical result of [MSS15b] essentially shows that given a collection of independent random vectors $v_1, \ldots, v_m$ with finite support in $\mathbb{C}^n$ s.t. $\sum_{i=1}^m \mathbb{E}[v_i v_i^*] = I$ and for all i, $\|v_i\|^2 \leq \epsilon$, then $\Pr[\|\sum_{i=1}^m v_i v_i^*\| \leq (1+\sqrt{\epsilon})^2] > 0$. [AG15] establishes a related result for k-homogeneous Strongly Rayleigh distributions, though they require an additional constraint on the marginal probability that any given random variable is non-zero being bounded above by δ, and then establish $\Pr[\|\sum_{i=1}^m v_i v_i^*\| \leq 4(\epsilon+\delta)+2(\epsilon+\delta)^2] > 0$. Based on this, [AG15] shows[2] that given an unweighted k-edge connected graph G where every edge has leverage score at most ϵ, there exists an unweighted spanning tree s.t. $L_T \preceq O(\frac{1}{k}+\epsilon)\cdot L_G$. This is referred to as a spectrally thin tree with parameter $O(\frac{1}{k}+\epsilon)$.

[2]Their full statement is more general, see [AG15] Corollary 1.9.

[AGR16] showed how to algorithmically sample from k-homogeneous Determinental Point Process in time $\text{poly}(k)n\log(n/\epsilon)$, where n is the dimension of matrix giving rise to the determinental point process and ϵ is the allowed total variation distance. Their techniques are based on generalization proofs of expansion in the base graph associated with a balanced matroid, a result first established by [FM92].

Random spanning trees. Algorithms for sampling random spanning trees have a long history, but only recently have they explicitly used matrix concentration [DKP+17, DPPR17, Sch18]. The matrix concentration arguments in these papers, however, deal mostly with how modifying a graph results in changes to the distribution of random spanning trees in the graph. We instead study how closely random spanning trees resemble the graph they were initially sampled from. Whether our result in turn has applications for improving sampling algorithms for random spanning trees is unclear.

The fact that spanning tree edges exhibit negative dependence has been used strikingly in concentration arguments by Goyal et al. [GRV09] to show that two random spanning trees gives $O(\log n)$-factor approximate cut sparsifier in bounded degree graphs, with good probability. This is clearly false when sampling the same number of edges independently, because this graph has large probability of having isolated vertices. Goyal et al. improve over independent sampling by leveraging the fact that for a fixed tree, in some sense, very few cuts of a given size exist. This is a variant of Karger's famous cut-counting techniques [Kar93, KS96] specialized to unweighted trees.

Uses of negatively dependent Chernoff bounds applied to tree edges also appeared in works on approximation algorithms for TSP problems [GSS11, AGM+10], where additionally

the connectivity properties of the tree play an important role

In contrast, the techniques of Fung et al. [FHHP11] show that $O(\epsilon^{-2}\log^2 n)$ spanning trees suffice to give a $(1\pm\epsilon)$-cut sparsifier, but they do not show that tree-based sparsifiers improve over independent sampling, The focus of their paper is to establish that wide range of different techniques for choosing sampling probabilities all give cut sparsifiers, by establishing a more flexible framework than the original cut-sparsifier results of Benczur-Karger [BK96], using related cut-counting techniques (see [Kar93, KS96]). To extend their results to spanning trees, they simply observe that the (scalar-valued) Chernoff bounds they use directly apply to negatively dependent variables, and hence edges in spanning trees.

Fung et at. [FHHP11] also establish a lower bound, showing that for any constant c, there exists a graph for which obtaining a factor c-cut sparsifier by averaging trees requires using at least $\Omega(\log n)$ trees to succeed with constant probability.

8.1.2 Our results and techniques

Theorem 8.1.1 (First main result, a Matrix Chernoff Bound k-homogeneous Strongly Rayleigh Distributions)**.** *Suppose $(\xi_1,\ldots,\xi_m)\in\{0,1\}^m$ is a random vector of $\{0,1\}$ variables whose distribution is k-homogeneous and Strongly Rayleigh.*

Given a collection of PSD matrices $A_1,\ldots A_m\in\mathbb{R}^{n\times n}$ s.t. for all $e\in[m]$ we have $\|A_e\|\leq R$ and $\|\mathbb{E}[\sum_e \xi_e A_e]\|\leq\mu$.

Then for any $\epsilon>0$,

$$\Pr\left[\left\|\sum_e \xi_e A_e-\mathbb{E}\left[\sum_e \xi_e A_e\right]\right\|\geq\epsilon\mu\right]\leq n\exp\left(-\frac{\epsilon^2\mu}{R(\log k+\epsilon)}\Theta(1)\right).$$

This Matrix Chernoff bound matches the bounds due to Tropp [Tro12], up to the $\log k$ factor in the exponent. Our lower bounds rule out that a much stronger bound is true in the $\epsilon \approx \log k$ regime, since the level of concentration we prove at this ϵ is stronger than what is ruled out for $\epsilon \approx \frac{\log k}{\log \log k}$ by the lower bound in Theorem 8.1.3. However, this does *not* rule out that the $\log k$ factor in our bound is unnecessary. I.e. we cannot rule out that a stronger concentration statement might match the bound for the independent case given by Tropp [Tro12].

Remark 8.1.1. When the Strongly Rayleigh distribution is in fact a product distribution on a collection of l Strongly Rayleigh distributions that are each t-homogeneous, then the joint distribution is lt-homogeneous Strongly Rayleigh, but in Theorem 8.1.1, the factor $\log(lt)$ can be replaced by a factor $\log t$. This applies to the case of independent random spanning trees, but only gives a constant factor improvement in the number of trees required.

Our work is related to the concentration inequality of Peres and Pemantle [PP14], who showed a concentration result for scalar-valued Lipschitz functions of Strongly Rayleigh distributions. They used Doob martingales (martingales constructed from sequences of conditional expectations) to prove their result. We use a similar approach for matrices, constructing Doob matrix martingales from our Strongly Rayleigh distributions. In addition, we use the stochastic covering property of Strongly Rayleigh distributions observed by Peres and Pemantle, but implicitly derived in [BBL09]. This property leads to bounded differences in Doob martingale sequences for scalars. As in the scalar setting, it is possible to show concentration results for matrix-valued martingales. We use the Matrix Freedman inequality[3]

[3]Note, however, that we are able to prove deterministic bounds on the *predictable quadratic variation*

of Tropp. This inequality allows makes it possible to establish strong concentration bounds based on control of sample norms and control of the *predictable quadratic variation process* of the martingale, a matrix-valued object that is used to measure variance (see [Tro11c]). We show that as in the scalar setting, the stochastic covering property of Strongly Rayleigh distributions leads to bounded differences for Doob matrix martingales. But, we also combine the stochastic covering property with deceptively simple matrix martingale properties and a negative dependence condition to derive additional bounds on the predictable quadratic variation process of the martingale. The key negative dependence property we use is a simple observation that generalizes, to k-homogeneous Strongly Rayleigh distributions, the fact that in a random spanning tree, conditioning on the presence of a set of edges lowers the marginal probability of every other graph edge being in the tree (see Lemma 8.1.7). While we frame it differently, it is essentially an immediate consequence of statements in [BBL09]. The surprise here is how useful this simple observation is for removing issues with characterizing conditional k-homogeneous Strongly Rayleigh distributions. As a corollary we get our second main result.

Theorem 8.1.2 (Second main result, concentration bound of a batch of independent random spanning trees)**.** *Given as input a weighted graph G with n vertices and a parameter $\epsilon > 0$, let $T_1, T_2, \cdots, T_t$ denote t independent inverse leverage score weighted random spanning trees, if we choose $t = O(\epsilon^{-2} \log^2 n)$ then with probability $1 - 1/\operatorname{poly}(n)$,*

$$(1-\epsilon)L_G \preceq \frac{1}{t}\sum_{i=1}^{t} L_{T_i} \preceq (1+\epsilon)L_G.$$

process, which means the bound we use is more analogous to a matrix version Bernstein's inequality, adapted to martingales. We resort to the more complicated Freedman's inequality only because it gives a directly applicable statement that is known in the literature.

Prior to our work, only a trivial bound on the number of spanning trees required to build a spectral sparsifier was known, namely that standard matrix concentration arguments like those in [SS11] prove that $O(\epsilon^{-2} n \log n)$ spanning trees suffice. Note that, the number of spanning trees required to build spectral sparsifier in our Theorem 8.1.2 matches the number of spanning trees required to construct cut sparsifier in previous best result [FHHP11]. The total edge count we require is $\Theta(\epsilon^{-2} n \log^2 n)$, worse by a factor $\log n$ than the bound for independent edge sampling obtained in [SS11]. It is not clear whether this factor in necessary.

Remark 8.1.2. Suppose we apply our Theorem 8.1.3 to show that any single random spanning tree satisfies $L_T \preceq O(\log n) \cdot L_G$ whp. This is tight up a $\log \log n$ factor. Then, one can from use this to derive Theorem 8.1.2 based on a standard (and tight) Matrix Chernoff bound, and a (laborious) combination of Doob martingales and stopping time arguments similar to those found in [Tro12, KS16]. This line of reasoning will lead to the same bounds as Theorem 8.1.3. Thus, unless one proves more than just a norm bound for each individual tree, it is not possible improve over our result, except for $\log \log n$ factors.

Like the work of Fung et al. our results for spanning trees do not improve over the independent case. Fung et al. achieved their result by combining cut counting techniques with Chernoff bounds for scalar-valued negatively dependent variables. In our random matrix setting, there are no clear candidates for a Chernoff bound for negatively dependent random matrices that we can adopt, and this type of bound is exactly what we develop in the Strongly Rayleigh case.

We establish a one-sided concentration result for a single tree, namely that whp. $L_T \preceq O(\log n) \cdot L_G$. Again, this is a direct application of Theorem 8.1.1.

Theorem 8.1.3 (Third main result, upper bound for the concentration of one random spanning tree)**.** *Given a graph G, let T be a random spanning tree, then with probability at least* $1 - 1/\operatorname{poly}(n)$

$$L_T \preceq O(\log n) \cdot L_G.$$

This upper bound is tight up to a factor $\log \log n$ as shown by our almost matching lower bound stated below.

Theorem 8.1.4 (Lower bound for the concentration of one random spanning tree)**.** *For any $n \geq 2^{2^6}$, there is an unweighted graph G with n nodes, s.t. if we sample an inverse leverage score weighted random spanning tree T, then with probability at least $1 - e^{-n \cdot 4}$,*

$$L_T \not\preceq \frac{\log n}{8 \log \log n} \cdot L_G.$$

Trivially, the presence of degree one nodes in L_T means that in a complete graph, $L_G \not\preceq L_T$. So choosing any other scaling of the tree will make at least one of the inequalities $L_T \not\preceq \frac{1}{8} \log n / \log \log n \cdot L_G$ and $L_G \not\preceq L_T$ true with a larger gap. Note that in the complete unweighted graph the trees we consider have weight $\Theta(n)$ on each edge. A random spanning tree in the complete graph has diameter about $\sqrt{n}$ [RS67]. This can be shown to imply that for an *unweighted* random tree $\widehat{T}$, $L_G \not\preceq \sqrt{n} L_{\widehat{T}}$. But once we scale up every edge of the tree by a factor $\Theta(n)$, the diameter bound no longer directly implies a spectral gap of the form $L_G \not\preceq \alpha L_T$ for some α. In a ring graph, we get $L_G \not\preceq (n-2) L_T$ and $L_T \not\preceq L_G$.

We can also show in general that $L_T \not\preceq 10 \log n \cdot L_G$, but with a much smaller probability.

Theorem 8.1.5 (Lower bound for the concentration of one random spanning tree)**.** *For any $n \geq 4$, there is an unweighted graph G with n nodes, s.t. if we sample an inverse leverage score weighted random spanning tree T, then with probability at least $2^{-150 \log n \log\log n}$,*

$$L_T \not\preceq 10 \log n \cdot L_G.$$

And we show a lower bound for ϵ-spectral sparsifiers for random spanning trees.

Theorem 8.1.6 (Lower bound for the concentration of multiple random spanning trees)**.** *For any $n \geq 2^{100}$, there is an unweighted graph G with n nodes, s.t. for any accuracy parameter $\epsilon \in (5n^{-0.1}, 1/2)$, if we sample $t = 0.05\epsilon^{-2} \log n$ independent random spanning trees with edges weighted by inverse leverage score, then with probability at least $1 - e^{-n^{.39}}$,*

$$(1-\epsilon)L_G \not\preceq \frac{1}{t}\sum_{i=1}^{t} L_{T_i} \text{ and } \frac{1}{t}\sum_{i=1}^{t} L_{T_i} \not\preceq (1+\epsilon)L_G.$$

Our lower bound is incomparable with that of Fung et al.[FHHP11], who showed that for any constant c, there exists a graph for obtaining a factor c-cut sparsifier by averaging trees requires using at least $\Omega(\log n)$ trees to succeed with constant probability. Where Fung et. al [FHHP11] used triangles in their lower bound construction, our bad examples are based on collections of small cliques, which lets us ensure cut differences in even a single tree, by giving longer-tailed degree distributions. All of our lower bounds are based on simple constructions from collections of edge disjoint cliques, and use the fact that the exact distribution of degrees of a fixed vertex in a random spanning tree of the complete graph is known. Note that a lower bound for cut approximation implies a lower-bound for spectral approximation, because the contrapositive statement is true: spectral approximation implies cut approximation.

Remark 8.1.3. In fact, all our lower bounds also directly apply for cut approximation, which is a strictly stronger result. For example, there is an unweighted graph G, s.t. if we sample an inverse leverage score weighted random spanning tree T, then with probability at least $1 - e^{-n \cdot 4}$, T has a cut which is larger than the corresponding cut in G by a factor $\frac{\log n}{8 \log \log n}$.

Connection to Spectrally Thin Trees Using their MSS-type existence proof for "small norm outcomes" of homogeneous Strongly Rayleigh distributions, [AG15] showed that in an unweighted k-edge connected graph G where every edge has leverage score at most ϵ, there exists an unweighted spanning tree $\widehat{T}$ s.t. $L_{\widehat{T}} \preceq O(\frac{1}{k} + \epsilon) \cdot L_G$. This is referred to a spectrally thin tree with parameter $O(\frac{1}{k} + \epsilon)$.

In contrast, applying our k-homogeneous Strongly Rayleigh Matrix Chernoff bound to an unweighted graph G where every edge has leverage score at most ϵ, we can show that an unweighted *random* spanning tree satisfies $L_T \preceq O(\epsilon \log n) L_G$ with high probability. This follows immediately from our Theorem 8.1.3, because if we let T denote the unweighted spanning tree and $\widehat{T}$ corresponding spanning tree with edges weighted by inverse leverage scores, then

$$\begin{aligned} L_{\widehat{T}} &= \sum_{e \in T} w(e) b_e b_e^\intercal \\ &\preceq \epsilon \sum_{e \in T} \frac{1}{l(e)} w(e) b_e b_e^\intercal \\ &= \epsilon L_{\widehat{T}} \\ &\preceq O(\epsilon \log n) L_G \end{aligned}$$

whp.

The proof in [AG15] is based on an adaptation of the [MSS15b] proof, and does not have clear parallels with our approach. Whereas the key properties of Strongly Rayleigh distributions that we use are stochastic covering (a property that limits change in a distribution under conditioning) and conditional negative dependence, the central element of their approach is a proof that certain mixed characteristic polynomials associated with k-homogeneous distributions are real stable when the original distribution is Strongly Rayleigh.

The following Lemma captures a simple but crucial property of Strongly Rayleigh distributions.

Lemma 8.1.7 (Shrinking Marginals)**.** *Suppose* $(\xi_1, \ldots, \xi_m) \in \{0,1\}^m$ *is a random vector of* $\{0,1\}$ *variables whose distribution is* k*-homogeneous and Strongly Rayleigh, then any set* $S \subseteq [m]$ *with* $|S| \leq k$ *for all* $j \in [m] \setminus S$

$$\Pr[\xi_j = 1 | \xi_S = \mathbf{1}_S] \leq \Pr[\xi_j = 1].$$

We provide a proof in Section 8.7.

8.2 Notation

We use $[n]$ to denote set $\{1, 2, \cdots, n\}$. Given a vector x, we use $\|x\|_0$ to denote the number of non-zero entries in the vector.

Matrix and norms. For a matrix A, we use $A^\top$ to denote the transpose of A. We say matrix A is positive semi-definite (PSD) if $A = A^\top$ and $x^\top Ax \geq 0$ for all $x \in \mathbb{R}^n$. We use $\succeq, \preceq$ to denote the semidefinite ordering, e.g. $A \succeq 0$ denotes that A is PSD, and $A \succeq B$ means $A - B \succeq 0$. We say matrix A is positive definite (PD) if $A = A^\top$ and $x^\top Ax > 0$ for all $x \in \mathbb{R}^n - \{0\}$. $A \succ B$ means $A - B$ is PD.

For matrix $A \in \mathbb{R}^{n \times n}$, we define $\|A\|$ to be the spectral norm of A, i.e.,

$$\|A\| = \max_{\|x\|_2 = 1, x \in \mathbb{R}^n} x^\top Ax.$$

Let $\mathrm{tr}(A)$ denote the trace of a square matrix A. We use $\lambda_{\max}(A)$ to denote the largest eigenvalue of matrix A. For symmetric matrix $A \in \mathbb{R}^{n \times n}$, $\lambda_{\max}(A) = \|A\|$ and $\mathrm{tr}(A) \leq n\|A\|$.

The Laplacian matrix-related definitions. Let $G = (V, E, w)$ be a connected weighted undirected graph with n vertices and m edges and edge weights $w_e > 0$. If we orient the edges of G arbitrarily, we can write its Laplacian as $L = B^\top WB$, where $B \in \mathbb{R}^{m \times n}$ is the signed edge-vertex incidence matrix and defined as follows

$$B(e, v) = \begin{cases} 1, & \text{if } v \text{ is } e\text{'s head;} \\ -1, & \text{if } v \text{ is } e\text{'s tail;} \\ 0, & \text{otherwise.} \end{cases}$$

and $W \in \mathbb{R}^{m \times m}$ is the diagonal matrix with $W(e, e) = w_e$.

8.3 Preliminaries

8.3.1 Useful facts and tools

This section, we provide some useful tools.

Fact 8.3.1. *For any two square matrices A and B, we have*

$$\frac{1}{2}(A+B)^2 + \frac{1}{2}(A-B)^2 = A^2 + B^2$$

Proof.

$$\begin{aligned}\frac{1}{2}(A+B)^2 + \frac{1}{2}(A-B)^2 &= \frac{1}{2}(A+B)(A+B) + \frac{1}{2}(A-B)(A-B) \\ &= \frac{1}{2}(A^2 + BA + AB + B^2) + \frac{1}{2}(A^2 - BA - AB + B^2) \\ &= A^2 + B^2,\end{aligned}$$

which completes the proof. □

Fact 8.3.2. *For any two symmetric matrices*

$$(A-B)^2 \preceq 2A^2 + 2B^2.$$

Proof. Using Fact 8.3.1, we have

$$(A+B)^2 + (A-B)^2 = 2A^2 + 2B^2.$$

Because A and B are symmetric matrices, then $(A+B)^2 \succeq 0$. It implies that

$$(A-B)^2 \preceq 2A^2 + 2B^2,$$

which completes the proof. □

8.3.2 Strongly Rayleigh distributions

This section provides definitions related to Strongly Rayleigh distributions. For more details, we refer the readers to [BBL09, PP14].

Let $\mu : 2^{[n]} \to \mathbb{R}_{\geq 0}$ denote a probability distribution over $2^{[n]}$, and $\sum_{S \subseteq [n]} \mu(S) = 1$.

Let $x_1, x_2, \cdots, x_n$ denote n variables, we use x to denote $(x_1, x_2, \cdots, x_n)$. For each set $S \subseteq [n]$, we define $x_S = \prod_{i \in S} x_i$. We define the **generating polynomial** for μ as follows

$$f_\mu(x) = \sum_{S \subseteq [n]} \mu(S) \cdot x_S.$$

We say distribution μ is k-homogeneous if the polynomial f_μ is a homogeneous polynomial of degree k. In other words, for each $S \in \mathrm{supp}(\mu)$, $|S| = k$.

We say a polynomial $p(x_1, x_2, \cdots, x_n)$ is stable, if $\mathrm{Im}(x_i) > 0, \forall i \in [n]$, then $p(x_1, \cdots, x_n) \neq 0$. We say polynomial p is real stable, it is stable and all of its coefficients are real. We say μ is a **Strongly Rayleigh** distribution if f_μ is a real stable polynomial.

Fact 8.3.3 (Conditioning on subset of coordinates)**.** *Consider a random vector $(\xi_1, \ldots, \xi_m) \in \{0,1\}^m$ whose distribution is k-homogeneous Strongly Rayleigh. Suppose we get a binary vector $\mathbf{b} = (b_1, \ldots, b_t) \in \{0,1\}^t$ with $\|\mathbf{b}\|_0 = l \leq k$, and we get a set $S \subset [m]$ with $|S| = t$. Then conditional on $\xi_S = \mathbf{b}$, the distribution of $\xi_{[m] \setminus S}$ is $(k-l)$-homogeneous Strongly Rayleigh.*

This fact tells us that if we condition on the value of some entries in the vector, the remaining coordinates still have a Strongly Rayleigh distribution.

Fact 8.3.4 (Stochastic Covering Property)**.** *Consider a random vector* $(\xi_1, \ldots, \xi_m) \in \{0,1\}^m$ *whose distribution is* k*-homogeneous Strongly Rayleigh. Suppose we are given an index* $i \in [m]$*. Let* $\xi' = \xi_{[m]\setminus\{i\}}$ *be the distribution on entries of* ξ *except* i*. Let* ξ'' *be the distribution of* $\xi_{[m]\setminus\{i\}}$ conditional *on* $\xi_i = 1$*. Then, there exists a coupling between* ξ' *and* ξ'' *(i.e. a joint distribution the two vectors), s.t. in every outcome of the coupling the value of* ξ' *can be obtained from the value of* ξ'' *by either changing a single from* 0 *to* 1 *or by leaving all entries unchanged.*

This fact is known as the *Stochastic Covering Property* (see [PP14]). It gives us a convenient tool for relating the conditional distribution of a subset of the coordinates of the vector to the unconditional distribution.

Note that by Fact 8.3.3, the distribution of ξ'' used in Fact 8.3.4 is $k-1$ homogeneous. In contrast, the outcomes of ξ' may have k or $k-1$ ones. Fact 8.3.4 tells us that we can pair up all the outcomes of the conditional distribution ξ'' with outcome of the unconditional distribution ξ' s.t. only a small change is required to make them equal. This tells us that the distribution is in some sense not changing too quickly under conditioning.

8.3.3 Random spanning trees

We provide the formal definition of random spanning tree in this section.

We use the same definitions about spanning trees as [DKP+17]. Let $\mathcal{T}_G$ denote the set of all spanning subtrees of G. We now define a probability distribution on these trees.

Definition 8.3.1 (w-uniform distribution on trees)**.** Let $\mathcal{D}_G$ be a probability distribution

on $\mathcal{T}_G$ such that

$$\Pr_{X \sim \mathcal{D}_G}[X = T] \propto \prod_{e \in T} w_e.$$

We refer to $\mathcal{D}_G$ as the w-uniform distribution on $\mathcal{T}_G$. When the graph G is unweighted, this corresponds to the uniform distribution on $\mathcal{T}_G$. Crucially, random spanning tree distributions are Strongly Rayleigh, as shown in [BBL09].

Fact 8.3.5 (Spanning Trees are Strongly Rayleigh)**.** *In a connected weighted graph G, the w-uniform distribution on spanning trees is $(n-1)$-homogeneous Strongly Rayleigh.*

Definition 8.3.2 (Effective Resistance)**.** The effective resistance of a pair of vertices $u, v \in V_G$ is defined as

$$R_{\text{eff}}(u, v) = b_{u,v}^\top L^\dagger b_{u,v},$$

where $b_{u,v}$ is an all zero vector corresponding to V_G, except for entries of 1 at u and -1 at v.

The a reference for following standard fact about random spanning trees can be found in [DKP+17].

Definition 8.3.3 (Leverage Score)**.** The statistical leverage score, which we will abbreviate to leverage score, of an edge $e = (u, v) \in E_G$ is defined as

$$l_e = w_e R_{\text{eff}}(u, v).$$

Fact 8.3.6 (Spanning Tree Marginals)**.** *The probability $\Pr[e]$ that an edge $e \in E_G$ appears in a tree sampled w-uniformly randomly from $\mathcal{T}_G$ is given by*

$$\Pr[e] = l_e,$$

where l_e is the leverage score of the edge e.

8.4 A Matrix Chernoff Bound for Strongly Rayleigh Distributions

We first define a mapping which maps an element into a psd matrix.

Definition 8.4.1 (Y-operator)**.** We use Γ to denote $[m]$, we define a mapping $Y : \Gamma \to \mathbb{R}^{n\times n}$ such that Y_e is a psd matrix and $\|Y_e\| \leq R$.

Throughout this section, we will use $\xi \in \{0,1\}^m$ to denote a random length m boolean vector whose distribution is k-homogeneous Strongly Rayleigh. For any set $S \subseteq [m]$, we use ξ_S to denote the length $|S|$ vector that only chooses the entry from indices in S.

We will frequently need to work with a different representation of the random variable ξ. We use γ to denote this second representation. The random variable γ is composed of a sequence of k random indices $\gamma_1, \gamma_2, \cdots, \gamma_k$, each of which takes a value $e_1, e_2, \cdots, e_k \in [m]$. The indices give the locations of the ones in ξ, i.e. in an outcome of the two variables (ξ, γ), we always have $\xi_{\{\gamma_1,\gamma_2,\cdots,\gamma_k\}} = \mathbf{1}$ and $\xi_{[m]\setminus\{\gamma_1,\gamma_2,\cdots,\gamma_k\}} = \mathbf{0}$. Additionally, we want to ensure that the distribution of γ is invariant under permutation: This can clearly be achieved by starting with any distribution for γ that satisfies the coupling with ξ and the applying a uniformly random permutation to reorder the k indices of γ (see [PP14] for a further discussion).

For convenience, for each $i \in [k]$, we define $\gamma_{\leq i}$ and $\gamma_{\geq i}$ as abbreviated notation for

$$\gamma_1, \gamma_2, \cdots, \gamma_i \text{ and } \gamma_i, \gamma_{i+1}, \cdots, \gamma_k$$

respectively. Let $S = \{e_1, e_2, \cdots, e_i\} \subset [m]$ be one possible assignment for indices of a subset of the ones in ξ, (we require $i \leq k$). Then the distribution of $\xi_{[m]\setminus S}$ conditional on $\xi_S = \mathbf{1}$ is the same as the the distribution of $\xi_{[m]\setminus S}$ conditional on $(\gamma_1, \gamma_2, \cdots, \gamma_i) = (e_1, e_2, \cdots, e_i)$. In

other words, in terms of the resulting distribution of $\xi_{[m]\setminus S}$, it is equivalent to condition on either $\gamma_{\leq i}$ or $\xi_{\gamma_{\leq i}} = \mathbf{1}$. We define matrix $Z \in \mathbb{R}^{n\times n}$ as follows.

Definition 8.4.2 (Z)**.** Let Z denote $\sum_{e\in\Gamma} \xi_e \cdot Y_e$ where $\|\xi\|_0 = k$. Due to the relationship between ξ and γ, we can also write Z as

$$Z = \sum_{i=1}^{k} Y_{\gamma_i}.$$

For simplicity, for each $i \in [k]$, we define $Z_{\leq i}$ and $Z_{\geq i}$ as follows,

$$Z_{\leq i} = \sum_{j=1}^{i} Y_{\gamma_j} \text{ and } Z_{\geq i} = \sum_{j=i}^{k} Y_{\gamma_j}.$$

We define a series of matrices $M_i \in \mathbb{R}^{n\times n}$ as follows

Definition 8.4.3 (M_i, martingale)**.** We define $M_0 = \mathbb{E}[Z]$. For each $i \in \{1, 2, \cdots, k-1\}$, we define M_i as follows

$$M_i = \mathop{\mathbb{E}}_{\gamma_{\geq i+1}}[Z \mid \gamma_1, \cdots, \gamma_i].$$

It is easy to see that

$$\mathop{\mathbb{E}}_{\gamma_{i+1}}[M_{i+1}] = M_i,$$

which implies

$$\mathop{\mathbb{E}}_{\gamma_{i+1}}[M_{i+1} - M_i \mid \gamma_1, \cdots, \gamma_i] = 0. \tag{8.1}$$

Note that we can split Z up as

$$\begin{aligned} Z &= \sum_{j=i+2}^{k} Y_{\gamma_j} + Y_{\gamma_{i+1}} + \sum_{j=1}^{i} Y_{\gamma_j} \\ &= Z_{\geq i+2} + Y_{\gamma_{i+1}} + Z_{\leq i} \end{aligned} \tag{8.2}$$

And similarly $Z = Z_{\geq i+1} + Z_{\leq i}$.

In order to relate M_i and M_{i+1}, we will consider a fresh copy of $\gamma_{\geq i+1}$ which we denote by $\widehat{\gamma}_{\geq i+1}$. We denote the corresponding fresh copy of $Z_{\geq i+1}$, by $\widehat{Z}_{\geq i+1}$. We can now give an equivalent definition of M_i in terms of the expectation over $\widehat{\gamma}_{\geq i+1}$, while M_{i+1} is still defined in terms of the expectation over $\gamma_{\geq i+2}$, so that

$$M_i = \mathop{\mathbb{E}}_{\widehat{\gamma}_{\geq i+1}}[\widehat{Z}_{\geq i+1} + Z_{\leq i} \mid \gamma_1, \cdots, \gamma_i] \text{ and } M_{i+1} = \mathop{\mathbb{E}}_{\gamma_{\geq i+2}}[Z_{\geq i+2} + Y_{\gamma_{i+1}} + Z_{\leq i} \mid \gamma_1, \cdots, \gamma_i, \gamma_{i+1}] \tag{8.3}$$

Note that both still depend on the same $\gamma_{\leq i}$ vector, and M_{i+1} depends on γ_{i+1}, but M_i does not. So far, we have simply introduced a sligtly different notation for M_i, since the expectation operation ensures that the value of M_i is unchanged.

We let $\xi \in \{0,1\}^m$ denote the binary vector indicating the positions of indices $\gamma_1, \cdots, \gamma_i, \gamma_{i+1}, \cdots, \gamma_k$. while letting $\widehat{\xi} \in \{0,1\}^m$ indicate the positions of indices of $\gamma_1, \cdots, \gamma_i$, $\widehat{\gamma}_{i+1}, \cdots, \widehat{\gamma}_k$.

Note that k-homogeneous Strongly Rayleigh implies the stochastic covering property. By Fact 8.3.4, the stochastic covering property implies that a coupling exists s.t. adding either *one* or *no* extra "ones" to the vector $\xi_{[m]\setminus\gamma_{i+1}}$ results in the vector $\widehat{\xi}_{[m]\setminus\gamma_{i+1}}$. But, since $\xi_{[m]\setminus\gamma_{i+1}}$ is $k-1$ homogenous and $\widehat{\xi}_{[m]}$ is k-homogenous, we can conclude that $\widehat{\xi}_{[m]\setminus\gamma_{i+1}}$ is obtained from $\xi_{[m]\setminus\gamma_{i+1}}$ by adding *no* ones, if and only $\widehat{\xi}_{[m]}$ has a one at index γ_{i+1}. From this we conclude a more helpful form of stochastic covering: we can construct an index $\widetilde{\gamma}_{i+1}$ and a coupling s.t. conditional on γ_{i+1}, the indices $\widetilde{\gamma}_{i+1}, \gamma_{i+2}, \cdots, \gamma_k$ have the same distribution as $\widehat{\gamma}_{i+1}, \widehat{\gamma}_{i+2}, \cdots, \widehat{\gamma}_k$. Thus

$$Z_{\geq i+2} + Y_{\widetilde{\gamma}_{i+1}} = \widehat{Z}_{\geq i+1}. \tag{8.4}$$

We define $X_{i+1} = M_{i+1} - M_i$ and then by Eq. (8.1)

$$\mathop{\mathbb{E}}_{\gamma_{i+1}}[X_{i+1} \mid \gamma_1, \cdots, \gamma_i] = 0. \tag{8.5}$$

Then we can rewrite X_{i+1} in the following way,

Claim 8.4.1. *Let $\mathcal{D}_{\gamma_{\leq i+1}}$ denote the coupling distribution between $Z_{\geq i+2}$ and $Y_{\widetilde{\gamma}_{i+1}}$ such that Eq. (8.4) holds. Then*

$$X_{i+1} = Y_{\gamma_{i+1}} - \mathop{\mathbb{E}}_{(Z_{\geq i+2}, Y_{\widetilde{\gamma}_{i+1}}) \sim \mathcal{D}_{\gamma_{\leq i+1}}} [Y_{\widetilde{\gamma}_{i+1}} \mid \gamma_{\leq i+1}].$$

Proof. Note that in the following proof, we should think of $\gamma_{\leq i+1}$ as fixed.

$$\begin{aligned}
X_{i+1} &= M_{i+1} - M_i \\
&= \mathop{\mathbb{E}}_{\gamma_{\geq i+2}}[Z \mid \gamma_{\leq i+1}] - \mathop{\mathbb{E}}_{\gamma_{\geq i+1}}[Z \mid \gamma_{\leq i}] \\
&= \mathop{\mathbb{E}}_{\gamma_{\geq i+2}}\left[Z_{\geq i+2} + Y_{\gamma_{i+1}} + Z_{\leq i} \;\middle|\; \gamma_{\leq i+1}\right] - \mathop{\mathbb{E}}_{\widehat{\gamma}_{\geq i+1}}\left[\widehat{Z}_{\geq i+1} + Z_{\leq i} \;\middle|\; \gamma_{\leq i}\right] \\
&= \mathop{\mathbb{E}}_{\gamma_{\geq i+2}}\left[Z_{\geq i+2} + Y_{\gamma_{i+1}} + Z_{\leq i} \;\middle|\; \gamma_{\leq i+1}\right] - \mathop{\mathbb{E}}_{(Z_{\geq i+2}, Y_{\widetilde{\gamma}_{i+1}}) \sim \mathcal{D}_{\gamma_{\leq i+1}}}\left[Z_{\geq i+2} + Y_{\widetilde{\gamma}_{i+1}} + Z_{\leq i} \;\middle|\; \gamma_{\leq i+1}\right] \\
&= \mathop{\mathbb{E}}_{(Z_{\geq i+2}, Y_{\widetilde{\gamma}_{i+1}}) \sim \mathcal{D}_{\gamma_{\leq i+1}}}\left[Y_{\gamma_{i+1}} - Y_{\widetilde{\gamma}_{i+1}} \mid \gamma_{\leq i+1}\right] \qquad (8.6)
\end{aligned}$$

where the first equality follows by definition of X_{i+1}, the second equality follows by Definition 8.4.3, the third equality follows by Eq. (8.3), the fourth equality follows by Eq. (8.4), and the fifth equality is by linearity of expectation and cancellation of terms that agree.

Once we condition on $\gamma_{\leq i}$ and γ_{i+1} being fixed, then $Y_{\gamma_{i+1}}$ is also fixed. Thus in Eq. (8.6), we can move $Y_{\gamma_{i+1}}$ out of Expectation, so that the right hand side of Eq. (8.6) becomes

$$\mathop{\mathbb{E}}_{(Z_{\geq i+2}, Y_{\widetilde{\gamma}_{i+1}}) \sim \mathcal{D}_{\gamma_{\leq i+1}}}\left[Y_{\gamma_{i+1}} - Y_{\widetilde{\gamma}_{i+1}} \mid \gamma_{\leq i+1}\right] = Y_{\gamma_{i+1}} - \mathop{\mathbb{E}}_{(Z_{\geq i+2}, Y_{\widetilde{\gamma}_{i+1}}) \sim \mathcal{D}_{\gamma_{\leq i+1}}}[Y_{\widetilde{\gamma}_{i+1}} \mid \gamma_{\leq i+1}]. \tag{8.7}$$

□

Fact 8.4.2. *We condition on $\gamma_{\leq i+1}$. Let $\mathcal{D}_{\gamma_{\leq i+1}}$ denote the coupling distribution such that $Z_{\geq i+2} + Y_{\widetilde{\gamma}_{i+1}} = \widehat{Z}_{\geq i+1}$ holds. We define $U_{\gamma_{i+1}}$ as follows*

$$U_{\gamma_{i+1}} = \mathop{\mathbb{E}}_{(Z_{\geq i+2}, Y_{\widetilde{\gamma}_{i+1}}) \sim \mathcal{D}_{\gamma_{\leq i+1}}} [Y_{\widetilde{\gamma}_{i+1}} \mid \gamma_{\leq i+1}].$$

Then, we have the following four properties,

$$\begin{aligned}
&\text{(I)} \ \mathop{\mathbb{E}}_{\gamma_{i+1}} [U_{\gamma_{i+1}} \mid \gamma_{\leq i}] = \mathop{\mathbb{E}}_{\gamma_{i+1}} [Y_{\gamma_{i+1}} \mid \gamma_{\leq i}], \\
&\text{(II)} \ \|Y_{\gamma_{i+1}}\| \leq R, \|U_{\gamma_{i+1}}\| \leq R, \\
&\text{(III)} \ \|Y_{\gamma_{i+1}} - U_{\gamma_{i+1}}\| \leq R, \\
&\text{(IV)} \ Y_{\gamma_{i+1}}^2 \preceq R \cdot Y_{\gamma_{i+1}}, U_{\gamma_{i+1}}^2 \preceq R \cdot U_{\gamma_{i+1}}.
\end{aligned}$$

Proof. Proof of (I). We have

$$\begin{aligned}
\mathop{\mathbb{E}}_{\gamma_{i+1}} [U_{\gamma_{i+1}} \mid \gamma_{\leq i}] &= \mathop{\mathbb{E}}_{\gamma_{i+1}} \left[\mathop{\mathbb{E}}_{(Z_{\geq i+2}, Y_{\widetilde{\gamma}_{i+1}}) \sim \mathcal{D}_{\gamma_{\leq i+1}}} \left[Y_{\widetilde{\gamma}_{i+1}} \mid \gamma_{\leq i+1} \right] \,\middle|\, \gamma_{\leq i} \right] \\
&= \mathop{\mathbb{E}}_{\gamma_{i+1}} \left[\mathop{\mathbb{E}}_{(Z_{\geq i+2}, Y_{\widetilde{\gamma}_{i+1}}) \sim \mathcal{D}_{\gamma_{\leq i+1}}} \left[Y_{\widetilde{\gamma}_{i+1}} \mid \gamma_{\leq i+1} \right] - Y_{\gamma_{i+1}} + Y_{\gamma_{i+1}} \,\middle|\, \gamma_{\leq i} \right] \\
&= \mathop{\mathbb{E}}_{\gamma_{i+1}} \left[-X_{i+1} + Y_{\gamma_{i+1}} \mid \gamma_{\leq i} \right] \\
&= \mathop{\mathbb{E}}_{\gamma_{i+1}} \left[-X_{i+1} \mid \gamma_{\leq i} \right] + \mathop{\mathbb{E}}_{\gamma_{i+1}} \left[Y_{\gamma_{i+1}} \mid \gamma_{\leq i} \right] \\
&= \mathop{\mathbb{E}}_{\gamma_{i+1}} \left[Y_{\gamma_{i+1}} \mid \gamma_{\leq i} \right],
\end{aligned}$$

where the third step follows by Eq. (8.7) and Eq. (8.6), the fourth step follows by linearity of expectation, and the last step follows by $\mathbb{E}_{\gamma_{i+1}}[-X_{i+1} \mid \gamma_{\leq i}] = 0$.

Proof of (II).

By definition of Y, we have $\|Y_{\gamma_{i+1}}\| \leq R$.

$$\begin{aligned}\|U_{\gamma_{i+1}}\| &= \left\| \mathop{\mathbb{E}}_{(Z_{\geq i+2}, Y_{\widetilde{\gamma}_{i+1}}) \sim \mathcal{D}_{\gamma_{\leq i+1}}} [Y_{\widetilde{\gamma}_{i+1}} \mid \gamma_{\leq i+1}] \right\| \\ &\leq \mathop{\mathbb{E}}_{(Z_{\geq i+2}, Y_{\widetilde{\gamma}_{i+1}}) \sim \mathcal{D}_{\gamma_{\leq i+1}}} \left[\left\| Y_{\widetilde{\gamma}_{i+1}} \right\| \mid \gamma_{\leq i+1} \right] \\ &\leq R.\end{aligned}$$

Proof of (III). For any two PSD matrices A and B, we have $\|A-B\| \leq \max(\|A\|, \|B\|)$. Because both $Y_{\gamma_{i+1}}$ and $U_{\gamma_{i+1}}$ are PSD matrices and $\max(\|Y_{\gamma_{i+1}}\|, \|U_{\gamma_{i+1}}\|) \leq R$, we get the desired property.

Proof of (IV).

It follows by (II) and that $Y_{\gamma_{i+1}}$ and $U_{\gamma_{i+1}}$ are both PSD matrices. □

We can show

Claim 8.4.3.

$$\mathop{\mathbb{E}}_{\gamma_{i+1}} \left[(Y_{\gamma_{i+1}} - U_{\gamma_{i+1}})^2 \mid \gamma_{\leq i} \right] \preceq 4R \cdot \mathop{\mathbb{E}}_{\gamma_{i+1}} [Y_{\gamma_{i+1}} | \gamma_{\leq i}]$$

Proof.

$$\begin{aligned} & \mathop{\mathbb{E}}_{\gamma_{i+1}} \left[(Y_{\gamma_{i+1}} - U_{\gamma_{i+1}})^2 \mid \gamma_{\leq i} \right] \\ \preceq & \mathop{\mathbb{E}}_{\gamma_{i+1}} \left[2Y_{\gamma_{i+1}}^2 + 2U_{\gamma_{i+1}}^2 \mid \gamma_{\leq i} \right] \\ \preceq & \mathop{\mathbb{E}}_{\gamma_{i+1}} \left[2R \cdot Y_{\gamma_{i+1}} + 2R \cdot U_{\gamma_{i+1}} \mid \gamma_{\leq i} \right] \\ \preceq & \mathop{\mathbb{E}}_{\gamma_{i+1}} \left[4R \cdot Y_{\gamma_{i+1}} \mid \gamma_{\leq i} \right], \end{aligned}$$

where the first step follows by Fact 8.3.2, and the second step follows by $U_{\gamma_{i+1}}^2 \preceq R \cdot U_{\gamma_{i+1}}$ and $Y_{\gamma_{i+1}}^2 \preceq R \cdot Y_{\gamma_{i+1}}$. □

Lemma 8.4.4. *Let $\mathbb{E}[\sum_{e\in\Gamma} \xi_e Y_e] \preceq \mu I$. For each $i \in \{1, 2, \cdots, k\}$, we have*

$$\mathop{\mathbb{E}}_{\gamma_i}[Y_{\gamma_i} \mid \gamma_{\le i-1}] \preceq \frac{1}{k+1-i}\mu I.$$

Proof. We use $\mathbf{1}$ to denote a length $i-1$ vector where each entry is one. We can think of $\gamma_{\le i-1}$ as having its values already set to some edges in Γ, for example $\gamma_1 = e_1, \cdots, \gamma_{i-1} = e_{i-1}$. Note that all of the $e_1, \cdots, e_{i-1}$ must be distinct. Then we use $\Gamma\backslash\gamma_{\le i-1}$ to denote $\Gamma\backslash\{e_1, \cdots, e_{i-1}\}$.

$$\begin{aligned}
\mathbb{E}[Y_{\gamma_i} \mid \gamma_{\le i-1}] &= \sum_{e\in\Gamma\backslash\gamma_{\le i-1}} \Pr[\gamma_i = e \mid \gamma_{\le i-1}] \cdot Y_e \\
&= \sum_{e\in\Gamma\backslash\gamma_{\le i-1}} \frac{\Pr[\xi_e = 1 \mid \gamma_{\le i-1}]}{k-(i-1)} \cdot Y_e \\
&= \sum_{e\in\Gamma\backslash\gamma_{\le i-1}} \frac{\Pr[\xi_e = 1 \mid \xi_{\gamma_{\le i-1}} = \mathbf{1}]}{k-(i-1)} \cdot Y_e \\
&\preceq \sum_{e\in\Gamma\backslash\gamma_{\le i-1}} \frac{\Pr[\xi_e = 1]}{k-(i-1)} \cdot Y_e \\
&\preceq \sum_{e\in\Gamma} \frac{\Pr[\xi_e = 1]}{k-(i-1)} \cdot Y_e \\
&= \frac{1}{k-(i-1)} \mathbb{E}\left[\sum_e \xi_e Y_e\right] \\
&\preceq \frac{1}{k+1-i}\mu I
\end{aligned}$$

where the first step follows by definition of expectation, the second step follows by $\Pr[\gamma_i = e \mid \gamma_{\le i-1}] = \Pr[\xi_e = 1 \mid \gamma_{\le i-1}]/(k-(i-1))$, the third step follows because $[\cdot|\gamma_{\le i-1}]$ is equivalent to $[\cdot|\xi_{\gamma\le i-1} = \mathbf{1}]$, the fourth step follows by $(\Pr[\xi_e = 1 \mid \xi_{\gamma\ge i-1} = \mathbf{1}] \le \Pr[\xi_e = 1])$

from the Shrinking Marginals Lemma 8.1.7, the fifth step follows by relaxing $\Gamma \backslash \gamma_{\leq i-1}$, the sixth step follows by $\Pr[\xi_e = 1] = \mathbb{E}[\xi_e]$ and linearity of expectation, and the last step follows by $\mathbb{E}[\sum_{e \in \Gamma} \xi_e Y_e] \preceq \mu I$.

□

Lemma 8.4.5. *For each* $i \in \{1, 2, \cdots, k\}$

$$\mathop{\mathbb{E}}_{\gamma_i}\left[X_i^2 \mid \gamma_{\leq i-1}\right] \preceq 4\mu R \frac{1}{k+1-i} I.$$

Proof. It follows by combining Claim 8.4.3 and Lemma 8.4.4 directly. □

The above lemma implies this corollary directly

Corollary 8.4.6.

$$\sum_{i=1}^{k} \mathop{\mathbb{E}}_{\gamma_i}\left[X_i^2 \mid \gamma_{\leq i-1}\right] \preceq 10\mu R \log k \cdot I.$$

8.4.1 Main result

To prove our main theorem 8.1.1, we need a useful tool for random matrices: Freedman's inequality for matrices (Lemma A.2.1).

Proof. of Theorem 8.1.1.

We use Y to denote A and Γ to denote $[m]$.

In order to use Lemma A.2.1, we first we define W_i as follows

$$W_i = \sum_{j=1}^{i} \mathop{\mathbb{E}}_{\gamma_i}\left[X_i^2 \mid \gamma_{\leq i-1}\right].$$

According to definition of M_i, $\{M_0, M_1, M_2 \cdots\}$ is a matrix martingale and $M_k - M_0 = \sum_e \xi_e A_e - \mathbb{E}[\sum_e \xi_e A_e]$.

We have proved the following facts,

The first one is, $\mathbb{E}_{\gamma_i}[X_i|\gamma_{\leq i-1}] = 0$. It follows by Eq. (8.5)

The second one is

$$\lambda_{\max}(X_i) \leq R$$

It follows by combining Property (III) of Fact 8.4.2 and Claim 8.4.1.

The third one is

$$\|W_i\| \leq \sigma^2, \forall i \in [k]$$

where $\sigma^2 = 10\mu R \log k$. It follows by Corollary 8.4.6.

Thus,

$$\Pr\left[\lambda_{\max}(M_k - M_0) \geq \epsilon\mu\right] \leq n \exp\left(-\frac{(\epsilon\mu)^2/2}{\sigma^2 + R(\epsilon\mu)/3}\right).$$

We have

$$\begin{aligned} \frac{t^2/2}{\sigma^2 + Rt/3} &= \frac{\epsilon^2\mu^2/2}{10\mu R \log k + R\epsilon\mu/3} && \text{by choosing } t = \epsilon\mu \\ &= \frac{3\epsilon^2\mu}{(60\log k + 2\epsilon)R}. \end{aligned}$$

Thus we prove one side of the bound. Since $\mathbb{E}_{\gamma_i}[-X_i|\gamma_{\leq i-1}] = 0$ and $\mathbb{E}_{\gamma_i}[(-X_i)^2|\gamma_{\leq i-1}] = \mathbb{E}_{\gamma_i}[X_i^2|\gamma_{\leq i-1}]$, then following the similar procedure as proving $\lambda_{\max}$, we have bound for $\lambda_{\min}$

$$\Pr[\lambda_{\min}(M_k - M_0) \leq -\epsilon\mu] \leq n \exp\left(-\frac{3\epsilon^2\mu}{(60\log k + 2\epsilon)R}\right).$$

Putting two sides of the bound together, we complete the proof. □

8.5 Applications to Random Spanning Trees

In this section, we show how to use Theorem 8.1.1 to prove the bound for one random spanning and also summation of random spanning trees.

Proof. of Theorem 8.1.3.

Let $G = (V, E, w)$ be a undirected weighted graph, $w : E \to R$, which is connected. The Laplacian of G is $L_G = \sum_{e \in E} w(e) b_e b_e^\intercal$.

Let $T \subseteq E$ be a random spanning tree of G in the sense of Definition 8.3.1. Let the weights of the edges in T be given by $w' : T \to R$ where $w'(e) = w(e)/l_e$, where l_e is the leverage score of e in G. Thus the Laplacian of the tree is $L_T = \sum_{e \in T} w'(e) b_e b_e^\intercal = \sum_{e \in T} \frac{w(e)}{l_e} b_e b_e^\intercal$. Then by Fact 8.3.6, $Pr[e \in T] = l_e$, and hence $\mathbb{E}[L_T] = L_G$.

Note also that for all $e \in E$, $||(L_G^\dagger)^{1/2} w(e) b_e b_e^\intercal (L_G^\dagger)^{1/2}|| = l_e$. Consider the random matrix $(L_G^\dagger)^{1/2} L_T (L_G^\dagger)^{1/2}$. The distribution of edge in the spanning tree can be seen as an $n-1$ homogeneous vector in $\{0,1\}^m$ where $m = |E|$. To apply Theorem 8.1.1, let ξ_e be the eth entry of this random vector, and

$$A_e = (L_G^\dagger)^{1/2} w'(e) b_e b_e^\intercal (L_G^\dagger)^{1/2}$$

Note $A_e \succeq 0$. Now $||A_e|| = 1$ and $\mathbb{E}\left[\sum_e \xi_e A_e\right] = \mathbb{E}[(L_G^\dagger)^{1/2} L_T (L_G^\dagger)^{1/2}] = (L_G^\dagger)^{1/2} L_G (L_G^\dagger)^{1/2} = \Pi = I - \frac{1}{n}\mathbf{1}\mathbf{1}^\intercal$, where we used in the last equality that the null space of the Laplacian of a connected graph is the span of the all ones vector. Thus, as each we get $||\mathbb{E}\left[\sum_e \xi_e A_e\right]|| = 1$ This means we can apply Theorem 8.1.1 with $R = 1$, $\mu = 1$ and $\epsilon = 100 \log n$ to whp. $||(L_G^\dagger)^{1/2} L_T (L_G^\dagger)^{1/2} - \Pi|| \leq 100 \log n$.

As L_T is a Laplacian, it has $\mathbf{1}$ in the null space, so can conclude that $(L_G^\dagger)^{1/2} L_T (L_G^\dagger)^{1/2} \preceq 100 \log n \Pi$. Hence $L_T \preceq \log n L_G$.

□

Proof. of Theorem 8.1.2.

The proof is similar to the proof of Theorem 8.1.3. Now we view the edges of $t = O(\epsilon^{-2} \log^2 n)$ independent random spanning trees as a $t(n-1)$-homogeneous Strongly Rayleigh Distribution a vector in $\{0,1\}^{t|E|}$. Note that the product of independent Strongly Rayleigh distributions is Strongly Rayleigh [BBL09]. Again we get $|| \mathbb{E} [\sum_e \xi_e A_e] || = 1$, but now we can take $R = \frac{1}{t}$, and hence we obtain the desired result by plugging into Theorem 8.1.1. □

8.6 Lower bounds

8.6.1 Single spanning tree, low probability

The goal of this section is to prove Theorem 8.1.5. First, we recall a helpful fact estbliashed by Prüfer [Prü18].

Fact 8.6.1. *If T is a uniformly random spanning tree of the complete graph G on n vertices, the degree distribution of a fixed node v in T is $1 + Binomial(n-2, 1/n)$.*

We now prove two claims that will serve as helpful tools, Claims 8.6.2 and 8.6.3.

Claim 8.6.2. *Let G be complete graph K_n with $n \geq 4$, let T denote a random spanning tree, the probability that at least one node of the T has degree at least $b \log n$ is at least $2^{-b \log n \log(b \log n) - 3}$.*

Proof. By Fact 8.6.1, the degree distribution of a fixed node in T is, $1 + \text{Binomial}(n-2, 1/n)$.

For a random variable x sampled from $\text{Binomial}(n-2, 1/n)$, we use q_i to denote the probability that $x = i$.

Let $p = 1/n$. We consider $q_{b\log n}$, which is

$$\begin{aligned}
q_{b\log n} &= \binom{n-2}{b\log n} \cdot p^{b\log n} \cdot (1-p)^{n-2-b\log n} \\
&= \binom{n-2}{b\log n} \cdot (1/n)^{b\log n} \cdot (1-1/n)^{n-2-b\log n} && \text{by } p = 1/n \\
&\geq ((n-2)/(b\log n))^{b\log n} \cdot (1/n)^{b\log n} \cdot (1-1/n)^{n-2-b\log n} \\
&= (b\log n)^{-b\log n} \cdot ((n-2)/n)^{b\log n} \cdot (1-1/n)^{n-2-b\log n} \\
&\geq (b\log n)^{-b\log n} \cdot (1-2/n)^{b\log n} \cdot (1-2/n)^{n-2-b\log n} \\
&= (b\log n)^{-b\log n} \cdot (1-2/n)^{n-2} \\
&\geq (b\log n)^{-b\log n} \cdot \frac{1}{e^2} \\
&\geq 2^{-b\log n \cdot \log(b\log n) - 3}.
\end{aligned}$$

where the seventh step follows by $(1-2/n)^{n-2} \geq 1/e^2$ when $n \geq 4$.Then the desired probability is

$$\sum_{i=b\log n}^{n} q_i \geq 2^{-b\log n \cdot \log(b\log n) - 3}.$$

□

Claim 8.6.3. *Let G be a complete graph K_n, let T denote a random spanning tree, if T has a node with degree at least d, then the inverse leverage score weighted Laplacian of the tree satisfies*

$$L_T \not\preceq (d/2) \cdot L_G.$$

Proof. There are $n(n-1)/2$ edges in the graph G. Let l_e denote the leverage of each $e \in G$. The the sum of the leverage scores is $\sum_{e\in G} l_e = n-1$, e.g. see [SS11]. Since all the edges in the graph G are symmetric, we have $l_e = 2/n$ for all edge e in G.

Let v denote a fixed node in graph G and let d be the degree of v. Let L_v denote the Laplacian matrix of the subgraph of T consisting of edges incident on v, i.e. the star of $d+1$ nodes with v at the center, and with edge weights as in T (which differ from those in G). We should think of L_v as a $n \times n$ matrix with only $d+1$ nonzeros on the diagonal.

Observe that $\lambda_{\max}(L_v) \leq \frac{n}{2}\lambda_{\max}(L_{K_{d+1}}) \leq \frac{n}{2}(d+1)$. We can also exhibit a unit vector $x = \frac{1}{\sqrt{d^2+d}}(d, -1, \cdots, -1)$, for which $x^\top L_v x = \frac{n}{2}(d+1)$ which implies that $\lambda_{\max}(L_v) \geq \frac{n}{2}(d+1)$. Therefore $\lambda_{\max}(L_v) = \frac{n}{2}(d+1)$.

We can split the L_T into two parts,

$$L_T = L_v + L_{T \setminus v}$$

and both parts are PSD matrices. We also know that $\lambda_{\max}(L_G) = n$. Thus,

$$L_T \not\preceq (d/2) \cdot L_G.$$

□

Proof. of Theorem 8.1.5.

The proof is a direct combination of Claim 8.6.2 and Claim 8.6.3.

The approximation ratio is

$$\frac{d}{2} = \frac{b \log n}{2} = 10 \log n,$$

where the last step follows by choosing $b = 20$.

Then the probability is

$$\begin{aligned} 2^{-b\log n\log(b\log n)-3} &= 2^{-b\log n\log\log n-b\log n\log b-3} \\ &\geq 2^{-20\log n\log\log n-20\log n\log 20-3} \\ &\geq 2^{-150\log n\log\log n} \end{aligned}$$

where the last step follows by $\log\log n \geq 1$ when $n \geq 4$. □

8.6.2 Single spanning tree, high probability

Proof. of Theorem 8.1.4.

Let $C = 4$, and let $\delta = 1/(C\log\log n)$. Note we have assumed $n \geq 2^{2^6}$, which ensures that $\delta < 0.05$.

We constuct a graph of size n as a union of $n^{1-\delta}$ cliques of size $n^{1-\delta}$ that are disjoint except they all share one central vertex. Applying Claim 8.6.2 with n replaced by n^δ, and $b = 1$, and assuming $n^\delta \geq 8$, we get that for each clique, the probability that at least one node has degree at least $\log n^\delta$ in T is at least

$$2^{-\delta c_0\log n\cdot\log(\delta\log n)}$$

where $c_0 = 2$. We can lower bound this probability:

$$\begin{aligned} 2^{-\delta c_0\log n\cdot\log(\delta\log n)} &= 2^{-\frac{c_0\log n}{C\log\log n}\log(\frac{1}{C\log\log n}\log n)} \\ &\geq 2^{-\frac{c_0\log n}{C\log\log n}\log\log n} \\ &= 2^{-(c_0\log n)/C} \\ &= n^{-c_0/C}, \end{aligned}$$

where the first step follows by $\delta = 1/(C \log\log n)$.

The probability that at least one node in G has degree at least $\log n^\delta$ in T is at least

$$\begin{aligned}
1 - (1 - 2^{-\delta c_0 \log n \log(\delta \log n)})^{n^{1-\delta}} &= 1 - \left(1 - \frac{1}{n^{c_0/C}}\right)^{n^{1-\delta}} \\
&= 1 - \left(1 - \frac{1}{n^{c_0/C}}\right)^{n^{c_0/C} \cdot \frac{n^{1-\delta}}{n^{c_0/C}}} \\
&\geq 1 - (1/e)^{n^{1-\delta-c_0/C}} \\
&\geq 1 - (1/e)^{n^{0.4}},
\end{aligned}$$

where the last step follows by $\delta \leq 0.05$, $c_0 = 2$ and $C = 4$. Thus, we have the desired probability.

Using Claim 8.6.3, we have the approximation ratio

$$\frac{d}{2} = \frac{\log(n^\delta)}{2} = \frac{\delta \log n}{2} = \frac{\log n}{2C \log\log n} = \frac{\log n}{8 \log\log n}.$$

Note that we still need to make sure $n^\delta \geq 8$, which is implied by $2^{\log n/4 \log\log n} \geq 2^3$ which is equivalent to $\log n \geq 12 \log\log n$. This holds for all $n \geq 2^{2^6}$: At $n = 2^{2^6}$, $\log n = 2^6 > 60 = 12 \cdot 5 = 12 \log\log n$, and as n grows, the left hand side grows faster than the right hand side.

□

8.6.3 Sum of a batch of spanning trees

Proof. of Theorem 8.1.6.

We constuct a graph of size n as a union of $n^{1-\delta}$ cliques of size $n^{1-\delta}$ that are disjoint except they all share one central vertex. The parameter $\delta \in (0, 1)$ will be decided later.

We use H to denote the graph formed by a collection of trees $T_1, T_2, \cdots, T_t$. Let L_H denote the Laplacian matrix of new graph H.

We use $\deg(v)$ to denote the degree of a vertex v. We use $\text{wdeg}(v)$ to denote weighted degree (after re-weighting). In the original graph G and the new graph H, we have for each vertex v, that

$$\text{wdeg}_H(v) = \frac{d}{2t}\deg_H(v), \text{ and } \text{wdeg}_G(v) = \deg_G(v).$$

By our construction of the graph, it is easy to see that $\deg_G(v) = n^\delta$ for all vertices v except the special central vertex that appears in all the cliques. Let $d = n^\delta$. Let ξ_1 denote the event that there exists a vertex $x \in V$ such that

$$\text{wdeg}_H(x) > (1+\epsilon)\,\text{wdeg}_G(x),$$

and let ξ_2 denote that there exists a vertex $y \in V$, such that

$$\text{wdeg}_H(y) < (1-\epsilon)\,\text{wdeg}_G(y).$$

We want to show that events ξ_1 and ξ_2 both occur simultaneously with probability at least $1 - e^{-n^{0.39}}$, which implies absence of spectral $(1 \pm \epsilon)$ approximation, as desired. We first bound the probability of ξ_1. Note that

$$\begin{aligned}
\Pr[\text{wdeg}_H(v) \geq (1+\epsilon)\,\text{wdeg}_G(v)] &= \Pr\left[\frac{d}{2t}\deg_H(v) \geq (1+\epsilon)\deg_G(v)\right] \\
&\geq \Pr\left[\frac{d}{2t}\deg_H(v) \geq (1+\epsilon)d\right] \\
&= \Pr[\deg_H(v) \geq 2t(1+\epsilon)] \\
&= \Pr[\deg_H(v) - t \geq t(1+2\epsilon)].
\end{aligned}$$

By Fact 8.6.1, the degree of a fixed node in T_i is distributed as $1+\text{Binomial}(d-2, 1/d)$. Then as H is a union of independent spanning trees, the degree in H of a fixed node is distributed as $t + \text{Binomial}(t(d-2), 1/d)$.

For a random variable x sampled from $\text{Binomial}(t(d-2), 1/d)$, we know that $\mathbb{E}[x] = t(1-2/d)$. For $\epsilon > 5/d$

$$\begin{aligned} t(1+2\epsilon) &= t(1-2/d) + (2t/d - 2\epsilon t + 8\epsilon t/d) + 4\epsilon t(1-2/d) \\ &\leq t(1-2/d) + (10t/d - 2\epsilon t) + 4\epsilon t(1-2/d) && \text{by } \epsilon \leq 1 \\ &\leq t(1-2/d) + 4\epsilon t(1-2/d) && \text{by } \epsilon > 5/d. \end{aligned}$$

So it suffices to calculate the probability that

$$x \geq t(1-2/d) + 4\epsilon t(1-2/d). \tag{8.8}$$

For any $k \geq 10, p \in (0, 1/2), \epsilon \in (0, 1/2)$ with $\epsilon^2 pk \geq 3$, using Lemma A.1.8, we can prove the probability that Eq. (8.8) holds is at least $2^{-c\epsilon^2 kp}$, where $c = 9$. We choose $k = t(d-2)$ and $p = 1/d$, and get that this probability is at least $2^{-c\epsilon^2 t(d-2)/d}$. Now, the probability that event ξ_1 holds is at least

$$1 - (1 - 2^{-c\epsilon^2 t(1-2/d)})^{n^{1-\delta}}$$

We have

$$\begin{aligned} 2^{-c\epsilon^2 t(1-2/d)} &\geq 2^{-c\epsilon^2 t} \\ &\geq 1/n^{0.5} \end{aligned}$$

where the last step follows by $t \leq 0.5 \log n/(c\epsilon^2)$.

Thus, we have

$$\begin{aligned} 1-(1-2^{-c\epsilon^2 t(1-2/d)})^{n^{1-\delta}} &\geq 1-(1-1/n^{0.5})^{n^{1-\delta}} \\ &\geq 1-e^{-n^{1-\delta-0.5}} \\ &= 1-e^{-n^{0.4}} && \text{by } \delta = 0.1 \end{aligned}$$

We summarize the conditions for ϵ:

$$\begin{aligned} \epsilon &\geq \max(5/d, 3/\sqrt{pn}) \\ &= \max(5/n^{0.1}, 3/\sqrt{pn}) && \text{by } d = n^{0.1} \\ &\geq \max(5/n^{0.1}, 5/\sqrt{t}) \end{aligned}$$

Since we choose $t = 0.05\epsilon^{-2}\log n$, then as long as $\log n \geq 100$ we have $\epsilon \geq 5/\sqrt{t}$.

Similarly, we can control the probability of event ξ_2 similarly, completing the proof.

□

8.7 Shrinking Marginals Lemma

Lemma 8.7.1 (Restatement of Lemma 8.1.7, Shrinking Marginals)**.** *Suppose* $(\xi_1, \ldots, \xi_m) \in \{0,1\}^m$ *is a random vector of* $\{0,1\}$ *variables whose distribution is* k*-homogeneous and Strongly Rayleigh, then any set* $S \subseteq [m]$ *with* $|S| \leq k$ *for all* $j \in [m] \setminus S$

$$\Pr[\xi_j = 1 | \xi_S = \mathbf{1}_S] \leq \Pr[\xi_j = 1]$$

Proof. Note that by an immediate consequence of negative association, for any pair $i, j \in [m]$, with $i \neq j$,

$$\Pr[\xi_j = 1 \mid \xi_i = 1] \leq \Pr[\xi_j = 1 \mid \xi_i = 0].$$

Hence

$$\begin{aligned}\Pr[\xi_j = 1 \mid \xi_i = 1] \leq\ & \Pr[\xi_j = 1 | \xi_i = 1] \cdot \Pr[\xi_i = 1] + \Pr[\xi_j = 1 | \xi_i = 0] \cdot (1 - \Pr[\xi_i = 1]) \\ =\ & \Pr[\xi_j = 1]\end{aligned}$$

By [BBL09], the distribution of $\xi_{[m]\setminus\{i\}} \in \{0,1\}^{m-1}$ conditional on $\xi_i = 1$ is Strongly Rayleigh.

With loss of generality, let us order the indices s.t. $S = \{1, \ldots, s\}$, where $s \leq k$. We use $[i]$ to denote $\{1, 2, \cdots, i\}$. Using the above observations, we can now prove the lemma by induction. The induction hypothesis at the i-th step (where $i \leq k$), is that the following two statements are true.

1. $\forall j \in \{i+1, \ldots, m\}. \Pr[\xi_j = 1 \mid \xi_{[i]} = \mathbf{1}] \leq \Pr[\xi_j = 1]$
2. The distribution of the vector of random variables $\xi_{m\setminus[i]} \in \{0,1\}^{m-i}$ conditional on $\xi_{[i]} = \mathbf{1}$ is Strongly Rayleigh.

□

Chapter 9

Four Derivation Suffice for Rank 1 Matrix

We prove a matrix discrepancy bound that strengthens the famous Kadison-Singer result of Marcus, Spielman, and Srivastava. Consider any independent scalar random variables $\xi_1, \ldots, \xi_n$ with finite support, e.g. $\{\pm 1\}$ or $\{0, 1\}$-valued random variables, or some combination thereof. Let $u_1, \ldots, u_n \in \mathbb{C}^m$ and

$$\sigma^2 = \left\| \sum_{i=1}^{n} \mathbb{V}[\xi_i](u_i u_i^*)^2 \right\|.$$

Then there exists a choice of outcomes $\epsilon_1, \ldots, \epsilon_n$ in the support of $\xi_1, \ldots, \xi_n$ s.t.

$$\left\| \sum_{i=1}^{n} \mathbb{E}[\xi_i] u_i u_i^* - \sum_{i=1}^{n} \epsilon_i u_i u_i^* \right\| \leq 4\sigma.$$

A simple consequence of our result is an improvement of a Lyapunov-type theorem of Akemann and Weaver.

9.1 Introduction

Discrepancy theory is an area of combinatorics that studies how well continuous objects can be approximated by discrete ones. It lies at the heart of numerous problems in mathematics and computer science [Cha00]. Although closely tied to probability theory, direct randomized approaches rarely yield the best bounds. In a classical formulation in discrepancy theory, we have n sets on n elements, and would like to two-color the elements so that each set has roughly the same number of elements of each color. Using a simple random coloring, it is an easy consequence of Chernoff's bound that there exists a coloring such that the discrepancy in all n sets is $O(\sqrt{n \log n})$ [AS16]. However, in a celebrated result, Spencer showed that in fact there is a coloring with discrepancy at most $6\sqrt{n}$ [Spe85].

Recently, there has been significant success in generalizing Chernoff [Che52], Hoeffding, Bernstein, and Bennett-type concentration bounds for scalar random variables to matrix-valued random variables [Rud99, AW02, Tro12]. Consider the following matrix concentration bound, which is a direct consequence of a matrix Hoeffding bound.

Theorem 9.1.1 ([Tro12])**.** *Let $\xi_i \in \{\pm 1\}$ be independent, symmetric random signs and $A_1, \ldots, A_n \in \mathbb{C}^{m \times m}$ be positive semi-definite matrices. Suppose $\max_{i \in [n]} \|A_i\| \leq \epsilon$ and $\|\sum_{i=1}^n A_i\| \leq 1$. Then,*

$$\Pr\left[\left\|\sum_{i=1}^n \xi_i A_i\right\| \geq t\sqrt{\epsilon}\right] \leq 2m \exp(-t^2/2).$$

A consequence of this theorem is that with high probability

$$\left\|\sum_{i=1}^n \xi_i A_i\right\| = O(\sqrt{\log m})\sqrt{\epsilon}. \tag{9.1}$$

The Kadison-Singer theorem of [MSS15b] is essentially equivalent to the following statement (which can readily be derived from the bipartition statement in [MSS15b]).

Theorem 9.1.2 ([MSS15b]). *Let* $u_1, \ldots, u_n \in \mathbb{C}^m$ *and suppose* $\max_{i\in[n]} \|u_i u_i^*\| \leq \epsilon$ *and* $\sum_{i=1}^n u_i u_i^* = I$. *Then, there exists signs* $\xi_i \in \{\pm 1\}$ *s.t.*

$$\left\| \sum_{i=1}^n \epsilon_i u_i u_i^* \right\| \leq O(\sqrt{\epsilon}).$$

Thus, for rank 1 matrices the theorem improves on the norm bound in Equation (9.1), by a factor $\sqrt{\log m}$, in a manner analogous to the improvement of Spencer's theorem over the bound based on the scalar Chernoff bound.

For random signings of matrices one can establish bounds in some cases that are much stronger than Theorem 9.1.1.

Theorem 9.1.3 ([Tro12]). *Let* $\xi_i \in \{\pm 1\}$ *be independent random signs, and let* $A_1, \ldots, A_n \in \mathbb{C}^{m\times m}$ *be Hermitian matrices. Let* $\sigma^2 = \left\| \sum_{i=1}^n \mathbb{V}[\xi_i] A_i^2 \right\|$. *Then,*

$$\Pr\left[\left\| \sum_{i=1}^n \mathbb{E}[\xi_i] A_i - \sum_{i=1}^n \xi_i A_i \right\| \geq t \cdot \sigma \right] \leq 2m \exp(-t^2/2).$$

From this theorem we deduce that with high probability

$$\left\| \sum_{i=1}^n \mathbb{E}[\xi_i] A_i - \sum_{i=1}^n \xi_i A_i \right\| = O(\sqrt{\log m})\sigma. \tag{9.2}$$

Of course, this implies that there exists a choice of signs $\epsilon_1, \ldots, \epsilon_n \in \{\pm 1\}$ such that

$$\left\| \sum_{i=1}^n \mathbb{E}[\xi_i] A_i - \sum_{i=1}^n \epsilon_i A_i \right\| = O(\sqrt{\log m})\sigma.$$

Our main result demonstrates that for rank-1 matrices, there exists a choice of signs with a stronger guarantee.

Theorem 9.1.4 (Main Theorem)**.** *Consider any independent scalar random variables $\xi_1, \dots, \xi_n$ with finite support. Let $u_1, \dots, u_n \in \mathbb{C}^m$ and*

$$\sigma^2 = \left\| \sum_{i=1}^{n} \mathbb{V}[\xi_i](u_i u_i^*)^2 \right\|.$$

Then there exists a choice of outcomes $\epsilon_1, \dots, \epsilon_n$ in the support of $\xi_1, \dots, \xi_n$

$$\left\| \sum_{i=1}^{n} \mathbb{E}[\xi_i] u_i u_i^* - \sum_{i=1}^{n} \epsilon_i u_i u_i^* \right\| \leq 4\sigma.$$

For rank 1 matrices our theorem improves on the norm bound in Equation (9.2), by a factor $\sqrt{\log m}$. Note for example, if ξ_i is $\{\pm 1\}$-valued, then $\mathbb{V}[\xi_i] = 1 - \mathbb{E}[\xi_i]^2$.

Specializing to centered random variables, we obtain the following corollary, which in a simple way generalizes the Kadison-Singer theorem, although with a slightly worse constant.

Corollary 9.1.5. *Let $u_1, \dots, u_n \in \mathbb{C}^m$ and*

$$\sigma^2 = \left\| \sum_{i=1}^{n} (u_i u_i^*)^2 \right\|.$$

There exists a choice of signs $\epsilon_i \in \{\pm 1\}$ such that

$$\left\| \sum_{i=1}^{n} \epsilon_i u_i u_i^* \right\| \leq 4\sigma.$$

Notice that if $\max_i \|u_i u_i^*\| \leq \epsilon$ and $\sum_{i=1}^{n} u_i u_i^* = I$, then $\sigma^2 \leq \epsilon$, and we obtain Theorem 9.1.2 as consequence of the corollary.

Our Theorem 9.1.4 has multiple advantages over Theorem 9.1.2. It allows us to show existence of solutions close to the mean of arbitrarily biased ± 1 random variables, instead

of only zero mean distributions. If some variable ξ_i is extremely biased, its variance is correspondingly low, as $\mathbb{V}[\xi_i] = 1 - \mathbb{E}[\xi_i]^2$.

If only a small subset of the rank 1 matrices obtain the ϵ norm bound, and the rest are significantly smaller in norm, then we can have $\sigma^2 \approx \epsilon^2$, so we prove a bound of $O(\epsilon)$ instead of the $O(\sqrt{\epsilon})$ bound of the Kadison-Singer theorem. On the other hand, when the problem is appropriately scaled, we always have $\sigma \geq \epsilon^2$, so the gap between the two results can never be more than a square root. Additionally, note that although the Kadison-Singer theorem requires $\sum_{i=1}^n u_i u_i^* = I$, a multi-paving argument[1] can be used to instead relax this to $\| \sum_{i=1}^n u_i u_i^* \| \leq 1$.

Approximate Lyapunov Theorems. Marcus, Spielman and Srivastava resolved the Kadison-Singer problem by proving Weaver's conjecture [Wea04], which was shown to imply the Kadison-Singer conjecture. In [AW14], Akemann and Weaver prove a generalization of Weaver's conjecture [Wea04].

Theorem 9.1.6 ([AW14]). *Let $u_1, \ldots, u_n \in \mathbb{C}^m$ such that $\| \sum_{i=1}^n u_i u_i^* \| \leq 1$ and $\max_i \|u_i u_i^*\| \leq \epsilon$. For any $t_i \in [0,1]$ and $1 \leq i \leq n$, there exists a set of indices $S \subset \{1, 2, \ldots, n\}$ such that*

$$\left\| \sum_{i \in S} u_i u_i^* - \sum_{i=1}^n t_i u_i u_i^* \right\| = O(\epsilon^{1/8}).$$

Due to the classical Lyapunov theorem [Lya40] and its equivalent versions [Lin66], in their study of operator algebras, Akemann and Anderson [AA91] refer to a result as a Lyapunov theorem if the result states that for a convex set C, the image of C under an affine

[1] This was pointed out to us by Tarun Kathuria.

map is equal to the image of the extreme points of C. Theorem 9.1.6 is an approximate Lyapunov theorem as it can be interpreted as saying that the image of $[0,1]^n$ under the map

$$f : (t_1, \ldots, t_n) \to \sum_{i=1}^{n} t_i u_i u_i^*,$$

can be approximated by the image of one of the vertices of the hypercube $[0,1]^n$. A corollary of our main result, Theorem 9.1.4, is the following strengthening of Theorem 9.1.6. This result greatly improves the ϵ dependence of the original Lyapunov-type theorem and provides a small explicit constant.

Corollary 9.1.7. *Let $u_1, \ldots, u_n \in \mathbb{C}^m$ such that $\| \sum_{i=1}^n u_i u_i^* \| \leq 1$ and $\max_i \|u_i u_i^*\| \leq \epsilon$. For any $t_i \in [0,1]$ and $1 \leq i \leq n$, there exists a set of indices $S \subset \{1, 2, \ldots, n\}$ such that*

$$\left\| \sum_{i \in S} u_i u_i^* - \sum_{i=1}^{n} t_i u_i u_i^* \right\| = 2\,\epsilon^{1/2}.$$

The corollary follows immediately from Theorem 9.1.4 by choosing as the ξ_i a set of independent $\{0,1\}$-valued random variables with means $t_i \in [0,1]$. Note then that $\mathbb{V}[\xi_i] = t_i(1-t_i) \leq 1/4$, and so by the assumptions $\max_i \|u_i u_i^*\| \leq \epsilon$ and $\sum_{i=1}^n u_i u_i^* \preceq I$, we have $\sigma^2 \leq \epsilon/4$, and we obtain Theorem 9.1.2 as consequence of the corollary.

9.1.1 Related work and open questions

Our work is an extension of the the Kadison-Singer theorem of [MSS15b]. This result has a rich history of connections with theoretical computer science, and, we hope, may represent a step toward getting polynomial time algorithms for finding Kadison-Singer partitions.

Expanders are sparse graphs that exhibit good connectivity. One view on this is that an expander graph approximates a complete graph in some sense, e.g. the spectrum of the adjacency matrix or Laplacian matrix associated with the graph resembles that of the complete graph. Ramanujan graphs [Mar73, LPS88] are sparse d-regular graphs whose adjacency matrix eigenvalues essentially optimally approximates those of a complete graph among all d-regular graphs for a given degree d. These initial constructions of Ramanujan graphs relied on number-theoretic techniques and could not show existence of Ramanujan for all degrees and sizes.

Using the *method of interlacing families of polynomials*, [MSS15a, MSS18], finally showed the existence of *bipartite*[2] Ramanujan graphs of all degrees and of all sizes. The existential result of [MSS15a] was turned into a polynomial time algorithm in [Coh16b]. The restriction to bipartite Ramanujan graphs in these papers arose from the fact that interlacing families most naturally control only the largest eigenvalue of a matrix, while Ramanujan graphs require both the largest and the smallest eigenvalue to be controlled. [MSS15a] overcame this obstacle by studying only bipartite graphs, whose adjacency matrix eigenvalues are symmetric around zero and hence controlling the largest eigenvalue[3] implies also controlling the smallest. Our techniques for simultaneously controlling the smallest and largest eigenvalues of a matrix are very different from earlier interlacing family-based methods, and an intriguing open question is whether similar ideas can be used to construct non-bipartite Ramanujan graphs of all degrees and sizes.

[2]Which approximates the complete bipartite graph.

[3]More precisely, it is the largest non-trivial eigenvalue that is being controlled. The largest eigenvalue of the adjacency matrix of a d-regular graph is the trivial eigenvalue of d.

[ST04] gave nearly-linear time algorithms for producing sparse graphs (with non-uniform weights) that closely approximate a given input graph, in the sense that the spectral behavior of the Laplacian matrices associated with the graphs is very similar. A greatly simplified and stronger result was given in [SS11], still with a nearly linear time algorithm. When used to construct a sparse graph that spectrally approximates a complete graph of size n, the result of [SS11] has a worse average degree than a corresponding Ramanujan graph by a factor $\Theta(\log n)$. Later, this was improved to within a factor $O(1)$, still with a polynomial time algorithm, in [BSS12]. Finally, a series of papers turned this latter result into a nearly-linear time algorithm [LS18, LS17]. The resolution of the Kadison-Singer theorem was motivated by [BSS12][4]. A major open question in the area is whether a polynomial time algorithm for finding explicit solutions to the Kadison-Singer problem exists, and we hope that our strengthened form of the result may make the problem more tractable.

The Kadison-Singer theorem and our result can be interpreted as analogous to Spencer's theorem in a matrix setting, in recent years that has been tremendous progress in obtaining polynomial time algorithms for Spencer's theorem and related problems in other settings [Bou10, LM15, Rot17, BDG16, BG17, BDGL18, NDTTJ18]. Techniques developed in these papers have also found algorithmic uses in approximation algorithms beyond typical discrepancy statements [Rot13]. While interlacing families have notably been used to to bound integrality gaps for ATSP [AG14] algorithmic results based on these have been limited, with [Coh16b] providing the most notable example of a polynomial time algorithm based on interlacing families. It seems likely, however, that a deeper understanding of the limits of matrix rounding could have many applications in approximation algorithms.

[4]Gil Kalai suggested the connection between that result and the Kadison-Singer problem.

9.1.2 Our techniques

Our proof is based on the method of interlacing polynomials introduced in [MSS15a, MSS15b]. One difficulty in applying the method of interlacing polynomials is the inability to control both the largest and smallest eigenvalues of a matrix simultaneously. Various techniques and restrictions are used to overcome this problem in [MSS15a, MSS15b] (studying bipartite graphs, assuming isotropic position). We develop a seemingly more natural approach for simultaneously controlling both the largest and smallest roots of the matrices we consider. We study polynomials that can be viewed as expected characteristic polynomials, but are more easily understood as the expectation of a product of multiple determinants. The using a product of two determinants helps us bound the upper and lower eigenvalues both at the same time.

We introduce an analytic expression for the expected polynomials in terms of linear operators that use second order derivatives. This has the advantage of allowing us to gain stronger control over the movement of roots of polynomials under the linear operators we apply than those used by [MSS15b]. This is because movement of the roots now depends on the curvature of our polynomials in favorable way. Interestingly, linear operators containing second order derivatives also appear in the work of [AG14] but for different reasons. This lets us reuse one of their lemmas for bounding position of the roots of a real stable polynomial.

9.2 Preliminaries

We gather several basic linear algebraic and analytic facts in the following sections.

9.2.1 Linear Algebra

Lemma 9.2.1. *Let $x \in \mathbb{R}^n$. Then*

$$\det(I - txx^*) = 1 - tx^*x.$$

Fact 9.2.2 (Jacobi's Formula)**.** *For $A(t) \in \mathbb{R}^{n \times n}$ a function of t,*

$$\frac{d}{dt}\det(A(t)) = \det(A(t))\operatorname{tr}\left[A^{-1}(t)\frac{d}{dt}A(t)\right].$$

9.2.2 Real Stability

Definition 9.2.1. A multivariate polynomial $p(z_1, \ldots, z_n) \in \mathbb{C}[z_1, \ldots, z_n]$ is *stable* if it has no zeros in the region $\{(z_1, \ldots, z_n) : \Im(z_i) > 0 \text{ for all } 1 \leq i \leq n\}$. p is *real stable* if p is stable and the coefficients of p are real.

Lemma 9.2.3 (Corollary 2.8, [AOGSS18])**.** *If $p \in \mathbb{R}[z_1, \ldots, z_n]$ is real stable, then for any $c > 0$, so is*

$$(1 - c\partial_{z_i}^2)p(z_1, \ldots, z_n)$$

for all $1 \leq i \leq n$.

Lemma 9.2.4 (Proposition 2.4, [BB08])**.** *If $A_1, \ldots, A_n$ are positive semidefinite symmetric matrices, then the polynomial*

$$\det\left(\sum_{i=1}^{n} z_i A_i\right)$$

is real stable.

We also need that real stability is preserved under fixing variables to real values (see [Wag11, Lemma 2.4(d)]).

Proposition 9.2.5. *If $p \in \mathbb{R}[z_1, \ldots, z_m]$ is real stable and $a \in \mathbb{R}$, then $p|_{z_1=a} = p(a, z_2, \ldots, z_m) \in \mathbb{R}[z_2, \ldots, z_m]$ is real stable.*

9.2.3 Interlacing Families

We recall the definition and consequences of interlacing families from [MSS15a].

Definition 9.2.2. We say a real rooted polynomial $g(x) = C\prod_{i=1}^{n-1}(x - \alpha_i)$ *interlaces* the real rooted polynomial $f(x) = C'\prod_{i=1}^{n}(x - \beta_i)$ if

$$\beta_1 \leq \alpha_1 \leq \cdots \leq \alpha_{n-1} \leq \beta_n.$$

Polynomials $f_1, \ldots, f_k$ have a *common interlacing* if there is a polynomial g that interlaces each of the f_i.

The following lemma relates the roots of a sum of polynomials to those of a common interlacer.

Lemma 9.2.6 (Lemma 4.2, [MSS15a])**.** *Let $f_1, \ldots, f_k$ be degree d real rooted polynomials with positive leading coefficients. Define*

$$f_\emptyset := \sum_{i=1}^{k} f_i.$$

If $f_1, \ldots, f_k$ have a common interlacing then there exists an i for which the largest root of f_i is upper bounded by the largest root of $f_\emptyset$.

Definition 9.2.3. Let $S_1, \ldots, S_n$ be finite sets. For every $S \in \mathcal{F}$, we let $f_S(x)$ be a real rooted polynomial of degree d with positive leading coefficient. For a choice of assignment $s_1, \ldots, s_n \in S_1 \times \cdots \times S_n$, let $f_{s_1,\ldots,s_n}(x)$ be a real rooted degree d polynomial with positive leading coefficient. For a partial assignment $s_1, \ldots, s_k \in S_1 \times \cdots \times S_k$ for $k < n$, we define

$$f_{s_1,\ldots,s_k} := \sum_{s_{k+1} \in S_{k+1}, \ldots, s_n \in S_n} f_{s_1,\ldots,s_k,s_{k+1},\ldots,s_n}. \tag{9.3}$$

Note that this is compatible with our definition of $f_\emptyset$ from Definition 9.2.2. We say that the polynomials $\{f_{s_1,\ldots,s_n}\}$ form an *interlacing family* if for all $k = 0, \ldots, n-1$ and all $s_1, \ldots, s_k \in S_1 \times \cdots \times S_k$, the polynomials have a common interlacing.

The following lemma relates the roots of the interlacing family to those of $f_\emptyset$.

Lemma 9.2.7 (Theorem 4.4, [MSS15a])**.** *Let $S_1, \ldots, S_n$ be finite sets and let $\{f_{s_1,\ldots,s_n}\}$ be an interlacing family. Then there exists some $s_1, \ldots, s_n \in S_1 \times \cdots \times S_n$ so that the largest root of $f_{s_1,\ldots,s_n}$ is upper bounded by the largest root of $f_\emptyset$.*

Finally, we recall a relationship between real-rootedness and common interlacings which has been discovered independently several times [DG94, Fel80, CS07].

Lemma 9.2.8. *Let $f_1, \ldots, f_k$ be univariate polynomials of the same degree with positive leading coefficient. Then $f_1, \ldots, f_k$ have a common interlacing if and only if $\sum_{i=1}^k \alpha_i f_i$ is real rooted for all nonnegative α_i such that $\sum_i \alpha_i = 1$.*

9.3 Expected Characteristic Polynomial

Instead of working with the random polynomial $\det(xI - \sum_{i=1}^n \xi_i u_i u_i^*)$, we consider

$$\det\left(x^2 I - \Big(\sum_{i=1}^n \xi_i u_i u_i^*\Big)^2\right) = \det\left(xI - \sum_{i=1}^n \xi_i u_i u_i^*\right) \det\left(xI + \sum_{i=1}^n \xi_i u_i u_i^*\right).$$

Observe that the largest root $\lambda_{\max}$ of this polynomial is

$$\lambda_{\max}\left(\det\left(x^2 I - \Big(\sum_{i=1}^n \xi_i u_i u_i^*\Big)^2\right)\right) = \left\|\sum_{i=1}^n \xi_i u_i u_i^*\right\|. \tag{9.4}$$

We gather some results that will allow us to extract an analytic expression for the expected characteristic polynomial.

Lemma 9.3.1. *For positive semidefinite (PSD) matrices $M, N \in \mathbb{R}^{m\times m}$, $v \in \mathbb{R}^m$ and ξ a random variable with zero mean and variance τ^2,*

$$\mathop{\mathbb{E}}_{\xi}[\det(M - \xi vv^*)\det(N + \xi vv^*)] = \left(1 - \frac{1}{2}\frac{d^2}{dt^2}\right)\Bigg|_{t=0} \det(M + t\tau vv^*)\det(N + t\tau vv^*). \tag{9.5}$$

Proof. We can assume that both M and N are positive definitive and hence invertible. The argument for the the positive semi-definite case follows by a continuity argument (using Hurwitz's theorem from complex analysis, see also [MSS15b]).

We show that the two sides of (9.5) are equivalent to the same expression. Beginning on the left hand side,

$$\begin{aligned}
&\mathbb{E}[\det(M - \xi vv^*)\det(N + \xi vv^*)] \\
&\qquad = \det(M)\det(N)\,\mathbb{E}[\det(I - \xi M^{-1/2} vv^* M^{-1/2})\det(I + \xi N^{-1/2} vv^* N^{-1/2})] \\
&\qquad = \det(M)\det(N)\,\mathbb{E}[1 + \xi b^* b - \xi a^* a - \xi^2 a^* a b^* b] \\
&\qquad = \det(M)\det(N)(1 - \tau^2 a^* a b^* b)
\end{aligned}$$

where $a := M^{-1/2}v$ and $b := N^{-1/2}v$. For the right hand side of (9.5),

$$\begin{aligned}\left(1 - \frac{1}{2}\frac{d^2}{dt^2}\right)\Bigg|_{t=0} &\det(M + t\tau vv^*)\det(N + t\tau vv^*) \\ &= \det(M)\det(N)\left(1 - \frac{1}{2}\frac{d^2}{dt^2}\right)\Bigg|_{t=0} \det(I + t\tau aa^*)\det(I + t\tau bb^*) \\ &= \det(M)\det(N)(1 - \tau^2 a^* a b^* b)\end{aligned}$$

where the last line follows from Lemma 9.2.1. □

Centering our random variables and applying the previous lemma leads to the following corollary for non-centered random variables.

Corollary 9.3.2. *Let $M, N \in \mathbb{R}^{m\times m}$ be arbitrary PSD matrices, $v \in \mathbb{R}^m$ and ξ a random variable with expectation μ and variance τ^2.*

$$\mathop{\mathbb{E}}_{\xi}[\det(M - (\xi-\mu)vv^*)\det(N + (\xi-\mu)vv^*)] = \left(1 - \frac{1}{2}\frac{d^2}{dt^2}\right)\Bigg|_{t=0} \det(M + t\tau vv^*)\det(N + t\tau vv^*).$$

We can now derive an expression for the expected characteristic polynomial.

Proposition 9.3.3. *Let $u_1, \ldots, u_n \in \mathbb{R}^m$. Consider independent random variables ξ_i with means μ_i and variances τ_i^2. Let $Q \in \mathbb{R}^{m\times m}$ be a symmetric matrix.*

$$\begin{aligned}\mathop{\mathbb{E}}_{\xi}\left[\det\left(x^2 I - \left(Q + \sum_{i=1}^n (\xi_i - \mu_i)u_i u_i^*\right)^2\right)\right] = \prod_{i=1}^n \left(1 - \frac{\partial_{z_i}^2}{2}\right)\Bigg|_{z_i=0} &\det\left(xI - Q + \sum_{i=1}^n z_i\tau_i u_i u_i^*\right) \\ &\times \det\left(xI + Q + \sum_{i=1}^n z_i\tau_i u_i u_i^*\right),\end{aligned}$$

and this is a real rooted polynomial in x.

Proof. For each i, let β_i denote the maximum value of $|\xi_i|$ among outcomes in the (finite) support of ξ_i. We begin by restricting the domain of our polynomials to $x > \|Q\| + 2\sum_{i=1}^n \beta_i \|u_i u_i^*\|$. We then proceed by induction. Our induction hypothesis will be that for $0 \leq k \leq n$

$$\begin{aligned}
&\mathbb{E}\left[\det\left(x^2 I - \left(Q + \sum_{i=1}^n (\xi_i - \mu_i) u_i u_i^*\right)^2\right)\right] \\
&= \mathop{\mathbb{E}}_{\xi_{k+1},\ldots,\xi_n} \prod_{i=1}^k \left(1 - \frac{\partial_{z_i}^2}{2}\right)\Bigg|_{z_i=0} \det\left(xI - Q - \sum_{i=k+1}^n (\xi_i - \mu_i) u_i u_i^* + \sum_{j=1}^k z_j \tau_j u_j u_j^*\right) \\
&\qquad \times \det\left(xI + Q + \sum_{i=k+1}^n (\xi_i - \mu_i) u_i u_i^* + \sum_{j=1}^k z_j \tau_j u_j u_j^*\right) \qquad (9.6)
\end{aligned}$$

The base case, $k = 0$ is trivially true as we get the same formula on both sides after recalling that for any matrix Y

$$\det\left(x^2 I - Y^2\right) = \det(xI - Y)\det(xI + Y).$$

By our assumption that $x > \|Q\| + 2\sum_{i=1}^n \beta_i \|u_i u_i^*\|$,

$$xI - \sum_{i=k+2}^n (\xi_i - \mu_i) u_i u_i^* + \sum_{j=1}^k z_j \tau_j u_j u_j^*$$

is PSD for any realization of $\xi_{k+2}, \ldots, \xi_n$ and in a neighborhood of zero for each z_j. Applying Corollary 9.3.2 to the right hand side of (9.6), yields

$$\begin{aligned}
&\mathbb{E}\left[\det\left(x^2 I - \left(Q + \sum_{i=1}^n (\xi_i - \mu_i) u_i u_i^*\right)^2\right)\right] \\
&= \mathop{\mathbb{E}}_{\xi_{k+2},\ldots,\xi_n} \prod_{i=1}^{k+1} \left(1 - \frac{\partial_{z_i}^2}{2}\right)\Bigg|_{z_i=0} \det\left(xI - Q - \sum_{i=k+2}^n (\xi_i - \mu_i) u_i u_i^* + \sum_{j=1}^{k+1} z_j \tau_j u_j u_j^*\right) \\
&\qquad \times \det\left(xI + Q + \sum_{i=k+2}^n (\xi_i - \mu_i) u_i u_i^* + \sum_{j=1}^{k+1} z_j \tau_j u_j u_j^*\right)
\end{aligned}$$

which completes the induction. To extend the proof to all x, we remark that we have shown the equivalence of two polynomials in the interval $x > \|Q\| + 2\sum_{i=1}^{n} \beta_i \|u_i u_i^*\|$, and two polynomials that agree on an interval are identical.

Real-rootedness of the right hand side follows by Lemma 9.2.4, Lemma 9.2.3, and that by Proposition 9.2.5 restriction to $z = \mathbf{0}$ preserves real-stability, and a univariate real stable polynomial is real rooted. □

9.4 Defining the Interlacing Family

The next proposition establishes that it suffices to bound the largest root of $q_\emptyset$.

Proposition 9.4.1. *There exists a choice of outcomes $\epsilon_1, \ldots, \epsilon_n$ in the finite support of $\xi_1, \ldots, \xi_n$, s.t.*

$$\left\| \sum_{i=1}^{n} \epsilon_i u_i u_i^* - \sum_{i=1}^{n} \mu_i u_i u_i^* \right\|$$

is less than the largest root of

$$\mathop{\mathbb{E}}_{\xi_1,\cdots,\xi_n} \left[\det \left(x^2 I - \Big(\sum_{i=1}^{n} (\xi_i - \mu_i) u_i u_i^* \Big)^2 \right) \right].$$

where $\mathbb{E}[\xi_i] = \mu_i, \forall i \in [n]$.

Proof. For a vector of independent random variables $(\xi_1, \ldots, \xi_n)$ with finite support, let $p_{i,x}$ be the probability that $\xi_i = x$. For $s = (\epsilon_1, \ldots, \epsilon_n)$ in the support of $\xi_1, \ldots, \xi_n$, we define

$$q_s(x) := \prod_{i=1}^{n} p_{i,\epsilon_i} \det \left(x^2 I - \Big(\sum_{i=1}^{n} (\epsilon_i - \mu_i) u_i u_i^* \Big)^2 \right).$$

Let t be a vector of k outcomes in the support of $\xi_1, \ldots, \xi_k$, i.e. a partial assignment of assignment of outcomes. Then we consider the conditional expected polynomial

$$q_t(x) := \left(\prod_{i=1}^{k} p_{i,t_i} \right) \mathop{\mathbb{E}}_{\xi_{k+1},\ldots,\xi_n} \left[\det \left(x^2 I - \left(\sum_{i=1}^{k} (t_i - \mu_i) u_i u_i^* + \sum_{j=k+1}^{n} (\xi_j - \mu_j) u_j u_j^* \right)^2 \right) \right]$$

which coincides with (9.3). We show that q_s is an interlacing family. Let $r_{k+1}^{(1)} \ldots r_{k+1}^{(l)}$ be the outcomes in the support of ξ_{k+1}. For a given t and $r \in \{r_{k+1}^{(1)}, \ldots, r_{k+1}^{(l)}\}$, let (t, r) denote the vector $(t_1, \ldots, t_k, r)$. By Lemma 9.2.8, it suffices to show that for any choice of non-negative numbers $\alpha_1, \ldots, \alpha_l$ such that $\sum_j \alpha_j = 1$, the polynomial

$$\sum_j \alpha_j q_{t, r_{k+1}^{(j)}}(x)$$

is real rooted. We can consider these α's as a probability distribution and define $p_{k+1,r_{k+1}^{(j)}} = \alpha_j$. Therefore,

$$\begin{aligned}
q_t(x) &= \left(\prod_{i=1}^{k} p_{i,t_i}\right) \mathop{\mathbb{E}}_{\xi_{k+1},\ldots,\xi_n} \left[\det\left(x^2 I - \left(\sum_{i=1}^{k}(t_i-\mu_i)u_i u_i^* + \sum_{j=k+1}^{n}(\xi_j-\mu_j)u_j u_j^*\right)^2\right)\right] \\
&= \sum_j \left(\prod_{i=1}^{k} p_{i,t_i}\right) p_{k+1,r_{k+1}^{(j)}} \\
&\quad \times \mathop{\mathbb{E}}_{\xi_{k+2},\ldots,\xi_n} \left[\det\left(x^2 I - \left(\sum_{i=1}^{k}(t_i-\mu_i)u_i u_i^* - (r_{k+1}^{(j)}-\mu_{k+1})u_{k+1}u_{k+1}^* - \sum_{j=k+2}^{n}(\xi_j-\mu_j)u_j u_j^*\right)^2\right)\right] \\
&= \sum_j \alpha_j q_{t,r_{k+1}^{(j)}}(x).
\end{aligned}$$

By Proposition 9.3.3, q_t is real rooted, which completes the proof that q_s is an interlacing family. Finally, by Lemma 9.2.7, there exists a choice of t so that the largest root of q_t is upperbounded by $q_\emptyset$.

□

9.5 Largest Root of the Expected Characteristic Polynomial

We use the barrier method approach to control the largest root [MSS15b].

Definition 9.5.1. For a multivariate polynomial $p(z_1, \ldots, z_n)$, we say $z \in \mathbb{R}^n$ is *above* all the roots of p if for all $t \in \mathbb{R}^n_+$,

$$p(z + t) > 0.$$

We use $\mathbf{Ab}_p$ to denote the set of points that are above all the roots of p.

We use the same barrier function as in [BSS12, MSS15b].

Definition 9.5.2. For a real stable polynomial p and $z \in \mathbf{Ab}_p$, the barrier function of p in direction i at z is

$$\Phi^i_p(z) := \frac{\partial_{z_i} p(z)}{p(z)}.$$

We will also make use of the following lemma that controls the deviation of the roots after applying a second order differential operator. The lemma is a slight variation of Lemma 4.8 in [AG14].

Lemma 9.5.1. *Suppose that p is real stable and $z \in \mathbf{Ab}_p$.*

If $\Phi^j_p(z) < \sqrt{2}$, then $z \in \mathbf{Ab}_{(1-\frac{1}{2}\partial^2_{z_j})p}$. If additionally for $\delta > 0$,

$$\frac{1}{\delta}\Phi^j_p(z) + \frac{1}{2}\Phi^j_p(z)^2 \leq 1,$$

then and for all i,

$$\Phi^i_{(1-\frac{1}{2}\partial^2_{z_j})p}(z + \delta \cdot \mathbf{1}_j) \leq \Phi^i_p(z).$$

We provide a proof in the Appendix 9.6 for completeness.

We can now bound the largest root of the expected characteristic polynomial for subisotropic vectors.

Proposition 9.5.2. *Let $\xi_1, \dots, \xi_n$ be independent scalar random variables with finite support, with $\mathbb{E}[\xi_i] = \mu_i$, and let $\tau_i^2 = \mathbb{E}[(\xi_i - \mu_i)^2]$.*

Let $v_1, \dots, v_n \in \mathbb{R}^n$ such that $\sum_{i=1}^n \tau_i^2 (v_i v_i^*)^2 \preceq I$. Then the largest root of

$$p(x) := \mathop{\mathbb{E}}_{\xi_1, \cdots, \xi_n} \left[\det \left(x^2 I - \Big(\sum_{i=1}^n (\xi_i - \mu_i) v_i v_i^* \Big)^2 \right) \right]$$

is at most 4.

Proof. Note that we must have

$$\max_{i \in [n]} \tau_i v_i^* v_i \leq 1, \tag{9.7}$$

since otherwise $\sum_{i=1}^n \tau_i^2 (v_i v_i^*)^2 \preceq I$ is false.

Let

$$Q(x, z) = \left(\det \left(xI + \sum_{i=1}^n z_i \tau_i v_i v_i^* \right) \right)^2.$$

For $t > 0$, define $\delta_i = t \tau_i v_i^* v_i$. For $t < \alpha(t)$ a parameter to be chosen later, we evaluate

our polynomial to find that

$$\begin{aligned}
Q(\alpha, -\delta_1, \ldots, -\delta_n) &= \left(\det\left(\alpha I - \sum_{i=1}^n \delta_i \tau_i v_i v_i^*\right)\right)^2 \\
&= \left(\det\left(\alpha I - t\sum_{i=1}^n \tau_i^2 (v_i^* v_i) v_i v_i^*\right)\right)^2 \\
&= \left(\det\left(\alpha I - t\sum_{i=1}^n \tau_i^2 (v_i v_i^*) v_i v_i^*\right)\right)^2 \\
&\geq \left(\det\left((\alpha - t) I\right)\right)^2 \\
&> 0.
\end{aligned}$$

where the second last step follows by $\sum_{i=1}^n \tau_i^2 (v_i v_i^*) v_i v_i^* \preceq I$.

This implies that $(\alpha, -\delta) \in \mathbb{R}^{n+1}$ is above the roots of $Q(x, z)$. We can upper bound $\Phi_Q^i(\alpha, -\delta)$ via Fact 9.2.2 as follows

$$\begin{aligned}
\Phi_Q^i(\alpha, -\delta) &= \left.\frac{\partial_{z_i} Q}{Q}\right|_{x=\alpha, z=-\delta} \\
&= \left.\frac{2\det\left(xI + \sum_{i=1}^n z_i \tau_i v_i v_i^*\right) \partial_{z_i} \det\left(xI + \sum_{i=1}^n z_i \tau_i v_i v_i^*\right)}{\left(\det\left(xI + \sum_{i=1}^n z_i \tau_i v_i v_i^*\right)\right)^2}\right|_{x=\alpha, z=-\delta} \\
&= 2\,\mathrm{tr}\left[\left(xI + \sum_{i=1}^n z_i \tau_i v_i v_i^*\right)^{-1} \tau_i v_i v_i^*\right]\Bigg|_{x=\alpha, z=-\delta} \\
&= 2\,\mathrm{tr}\left[\left(\alpha I - t\sum_{i=1}^n \tau_i^2 (v_i v_i^*)^2\right)^{-1} \tau_i v_i v_i^*\right] \\
&\leq 2\,\mathrm{tr}\left[\left(\alpha - t\right)^{-1} \tau_i v_i v_i^*\right] \\
&= \frac{2\tau_i v_i^* v_i}{\alpha - t},
\end{aligned}$$

where the fifth step follows by $\sum_{i=1}^n \tau_i^2 (v_i v_i^*) v_i v_i^* \preceq I$.

Choosing $\alpha = 2t$ and $t = 2$, and recalling Condition (9.7) we get

$$\Phi_Q^i(\alpha, -\delta) \leq \frac{2\tau_i v_i^* v_i}{\alpha - t} \leq 1 < \sqrt{2}.$$

By Lemma 9.2.3, $(1 - \frac{1}{2}\partial_{z_i}^2)Q$ is real stable. By Lemma 9.5.1, and $(4, z) \in \mathbf{Ab}_{(1-\frac{1}{2}\partial_{z_i}^2)Q}$ and hence $(4, z + \delta_i \mathbf{1}_i) \in \mathbf{Ab}_{(1-\frac{1}{2}\partial_{z_i}^2)Q}$. Also

$$\begin{aligned}
\frac{1}{\delta_i}\Phi_Q^i(\alpha, -\delta) + \frac{1}{2}\Phi_Q^i(\alpha, -\delta)^2 &\leq \frac{1}{t\tau_i v_i^* v_i}\frac{2\tau_i v_i^* v_i}{t} + \frac{1}{2}\left(\frac{2\tau_i v_i^* v_i}{t}\right)^2 \\
&\leq \frac{4}{t^2} \\
&= 1.
\end{aligned}$$

Therefore, by Lemma 9.5.1, for all j

$$\Phi^j_{(1-\frac{1}{2}\partial_{z_i}^2)Q}(4, z + \delta_i \mathbf{1}_i) \leq \Phi_Q^j(4, z).$$

Repeating this argument for each $i \in [n]$ demonstrates that $(4, 0, \ldots, 0)$ lies above the roots of

$$\prod_{i=1}^n \left(1 - \frac{\partial_{z_i}^2}{2}\right)\left(\det\left(xI + \sum_{i=1}^n z_i \tau_i u_i u_i^*\right)\right)\left(\det\left(xI + \sum_{i=1}^n z_i \tau_i u_i u_i^*\right)\right).$$

After restricting to $z_i = 0$ for all i, we then conclude by Proposition 9.3.3 is equivalent to a bound on the largest root of the expected characteristic polynomial. □

Having developed the necessary machinery, we now prove our main theorem.

Proof of Theorem 9.1.4. Define $v_i = \frac{u_i}{\sqrt{\sigma}}$. Then, $\| \sum_{i=1}^n \tau_i^2 (v_i v_i^*)^2 \| = 1$. Applying Proposition 9.5.2 and Proposition 9.4.1, we conclude that there exists a choice of outcomes ϵ_i such

that

$$\left\| \sum_{i=1}^{n} (\epsilon_i - \mu_i) v_i v_i^* \right\| \leq 4.$$

From this, we conclude that

$$\left\| \sum_{i=1}^{n} (\epsilon_i - \mu_i) u_i u_i^* \right\| \leq 4\sigma.$$

□

9.6 Omitted Proofs

9.6.1 Proof of Lemma 9.5.1

Choosing $c = 1/2$ in Lemma 9.6.1 gives a proof of Lemma 9.5.1.

Lemma 9.6.1 (Generalization of Lemma 4.8 in [AG14])**.** *Suppose that $p(z_1, \cdots, z_m)$ is real stable and $z \in Ab_p$. For any $c \in [0, 1]$.*

If $\Phi_p^j(z) < \sqrt{1/c}$, then $z \in \mathbf{Ab}_{(1-c\partial_{z_j}^2)p}$. If additionally for $\delta > 0$,

$$c \cdot \left(\frac{2}{\delta} \Phi_p^j(z) + (\Phi_p^j(z))^2 \right) \leq 1,$$

then, for all $i \in [m]$,

$$\Phi_{(1-c\partial_j^2)p}^i(z + \delta \mathbf{1}_j) \leq \Phi_p^i(z).$$

Proof. We write ∂_i instead of ∂_{z_i} for ease of notation. By Lemma 9.2.3, $(1 - c\partial_j^2)p$ is real stable. Recall the definitions of $\Phi_p^j(z)$ and $\Psi_p^j(z)$

$$\Phi_p^j(z) = \frac{\partial_{z_j} \det(M)}{\det(M)} = \frac{\partial_j p}{p} \text{ and } \Psi_p^j(z) = \frac{\partial_j^2 \det(M)}{\det(M)} = \frac{\partial_j^2 p}{p}.$$

Consider a non-negative vector t. By Lemmas 4.6 and 4.5 in [AG14], we have

$$\Phi_p^j(z + t) \leq \Phi_p^j(z), \qquad \Psi_p^j(z + t) \leq \Phi_p^j(z + t)^2 \leq \Phi_p^j(z)^2. \tag{9.8}$$

Thus, $\Psi_p^j(z + t) \leq \Phi_p^j(z)^2 < 1/c$ so $c \cdot \partial_j^2 p(z + t) < p(z + t)$, i.e. $(1 - c\partial_j^2)p(z + t) > 0$. Thus $z \in \mathbf{Ab}_{(1-c\partial_{z_j}^2)p}$.

Next, we write $\Phi^i_{p-c\cdot\partial_j^2 p}$ in terms of Φ^i_p and Ψ^j_p and $\partial_i \Psi^j_p$.

$$\begin{aligned}
\Phi^i_{p-c\cdot\partial_j^2 p} &= \frac{\partial_i(p - c\cdot\partial_j^2 p)}{p - c\cdot\partial_j^2 p} \\
&= \frac{\partial_i((1-c\cdot\Psi^j_p)p)}{(1-c\cdot\Psi^j_p)p} \\
&= \frac{(1-c\cdot\Psi^j_p)(\partial_i p)}{(1-c\cdot\Psi^j_p)p} + \frac{(\partial_i(1-c\cdot\Psi^j_p))p}{(1-c\cdot\Psi^j_p)p} \\
&= \Phi^i_p - \frac{c\cdot\partial_i\Psi^j_p}{1-c\cdot\Psi^j_p}.
\end{aligned}$$

We would like to show that $\Phi^i_{p-c\cdot\partial_j^2 p}(z+\delta\mathbf{1}_j) \le \Phi^i_p(z)$. Equivalently, it is enough to show that

$$-\frac{c\cdot\partial_i\Psi^j_p(z+\delta\mathbf{1}_j)}{1-c\cdot\Psi^j_p(z+\delta\mathbf{1}_j)} \le \Phi^i_p(z) - \Phi^i_p(z+\delta\mathbf{1}_j).$$

By convexity, it is enough to show that

$$-\frac{c\cdot\partial_i\Psi^j_p(z+\delta\mathbf{1}_j)}{1-c\cdot\Psi^j_p(z+\delta\mathbf{1}_j)} \le \delta\cdot(-\partial_j\Phi^i_p(z+\delta\mathbf{1}_j)).$$

By monotonicity, $\delta\cdot(-\partial_j\Phi^i_p(z+\delta\mathbf{1}_j)) > 0$ so we may divide both sides of the above inequality by this term and obtain that the above is equivalent to

$$\frac{-c\cdot\partial_i\Psi^j_p(z+\delta\mathbf{1}_j)}{-\delta\cdot\partial_i\Phi^j_p(z+\delta\mathbf{1}_j)}\cdot\frac{1}{1-c\Psi^j_p(z+\delta\mathbf{1}_j)} \le 1,$$

where we also used $\partial_j\Phi^i_p = \partial_i\Phi^j_p$. By Lemma 4.10 in [AG14], $\frac{\partial_i\Psi^j_p}{\partial_i\Phi^j_p} \le 2\Phi^j_p$. So, we can write,

$$\frac{2c}{\delta}\Phi^j_p(z+\delta\mathbf{1}_j)\cdot\frac{1}{1-c\cdot\Psi^j_p(z+\delta\mathbf{1}_j)} \le 1.$$

By Equation (9.8), with $t = \delta\mathbf{1}_j$,

$$\Phi^j_p(z+\delta\mathbf{1}_j) \le \Phi^j_p(z), \qquad \Psi^j_p(z+\delta\mathbf{1}_j) \le \Phi^j_p(z+\delta\mathbf{1}_j)^2 \le \Phi^j_p(z)^2.$$

So, it is enough to show that

$$\frac{2c}{\delta}\Phi_p^j(z) \cdot \frac{1}{1 - c \cdot \Phi_p^j(z)^2} \leq 1.$$

Using $\Phi_p^j(z) < 1$ and $c \in [0, 1]$ we know that $1 - c \cdot \Phi_p^j(z)^2 < 1$. We can multiply both sides with $1 - c \cdot \Phi_p^j(z)^2$ and we obtain that it suffices to have

$$c \cdot \left(\frac{2}{\delta}\Phi_p^j(z) + \Phi_p^j(z)^2\right) \leq 1,$$

which is true by assumption. □

Part III

Compressive Sensing and Sparse Fourier Transform

Chapter 10

Compressive Sensing

We consider the extensively studied problem of ℓ_2/ℓ_2 compressed sensing. The main contribution of our work is an improvement over [Gilbert, Li, Porat and Strauss, STOC 2010] with faster decoding time and significantly smaller column sparsity, answering two open questions of the aforementioned work.

Previous work on sublinear-time compressed sensing employed an iterative procedure, recovering the heavy coordinates in phases. We completely depart from that framework, and give the first sublinear-time ℓ_2/ℓ_2 scheme which achieves the optimal number of measurements without iterating; this new approach is the key step to our progress. Towards that, we satisfy the ℓ_2/ℓ_2 guarantee by exploiting the heaviness of coordinates in a way that was not exploited in previous work. Via our techniques we obtain improved results for various sparse recovery tasks, and indicate possible further applications to problems in the field, to which the aforementioned iterative procedure creates significant obstructions.

10.1 Introduction

Compressed Sensing, or sparse recovery, is a powerful mathematical framework the goal of which is to reconstruct an approximately k-sparse vector $x \in \mathbb{R}^n$ from linear measurements $y = \Phi x$, where $\Phi \in \mathbb{R}^{m \times n}$. The most important goal is to reduce the number of measurements m needed to approximate the vector x, avoiding the linear dependence on n. In discrete signal processing, where this framework was initiated [CRT06b, Don06], the core principle that the sparsity of a signal can be exploited to recover it using much fewer samples than the Shannon-Nyquist Theorem. We refer to the matrix Φ as the sketching or sensing matrix, and $y = \Phi x$ as the sketch of vector x.

Sparse recovery is the primary task of interest in a number of applications, such as image processing [TLW+06, LDP07a, DDT+08], design pooling schemes for biological tests [ECG+09, DWG+13], pattern matching [CEPR07], combinatorial group testing [SAZ09, ESAZ09, KBG+10], localizing sources in sensor networks [ZBSG05, ZPB06], as well as neuroscience [GS12]. Furthermore, not surprisingly, tracking heavy hitters in data streams, also known as frequent items, can be captured by the sparse recovery framework [Mut05a, CH09, KSZC03, Ind07]. In practice, streaming algorithms for detecting heavy hitters have been used to find popular destination addresses and heavy bandwidth users by AT&T [CJK+04] or answer "iceberg queries" in databases [FSGM+99].

Sparse recovery attracts researchers from different communities, from both theoretical and practical perspective. During the last ten years, hundreds of papers have been published by theoretical computer scientists, applied mathematicians and electrical engineers that specialize in compressed sensing. While numerous algorithms using space linear in the universe size n are known, [Don06, CRT06b, IR08a, NT09a, BIR08a, BD09a, SV16]

to name a few, our goal is to obtain algorithms that are sublinear, something that is crucial in many applications.

The desirable quantities we want to optimize may vary depending on the application. For example, in network management, x_i could denote the total number of packets with destination i passing through a network router. In such an application, storing the sketching matrix explicitly is typically not a tenable solution, since this would lead to an enormous space consumption; the number of possible IP addresses is 2^{32}. Moreover, both the query and the update time should be very fast, in order to avoid congestion on the network. Incremental updates to x come rapidly, and the changes to the sketch should also be implemented very fast; we note that in this case, even poly-logarithmic factors might be prohibitive. Interested readers can refer to [KSZC03, EV03] for more information about streaming algorithms for network management applications.

"The goal of that research is to obtain encoding and recovery schemes with good compression rate (i.e., short sketch lengths) as well as good algorithmic properties (i.e., low encoding, update and recovery times)." – *Anna Gilbert and Piotr Indyk* [GI10]

Sparse recovery schemes that are optimal across all axis are a challenge and an important theoretical and practical problem. For most sparse recovery tasks, we have algorithms that achieve different trade-offs for the various parameters of interest. One exception is the ℓ_∞/ℓ_2 guarantee, for which the breakthrough work of Larsen, Nelson, Nguyên and Thorup [LNNT16] shows that this trade-off is unnecessary.

10.1.1 Previous work

Since compressed sensing has been extensively studied in the literature for more than a decade, different guarantees of interest have been suggested (x_{-k} is the vector that occurs after zeroing out every i that does not belong among the largest k coordinates). In what follows $x \in \mathbb{R}^n$ is the vector we want to sketch, x' is the approximation to x, k is the sparsity and ϵ is the fineness of the approximation.

There are two different guarantees researchers consider in compressed sensing, one is the for-all guarantee and the other is the for-each guarantee. In the for-all guarantee, one wants to design a sketch that gives the desired result for all vectors $x \in \mathbb{R}^n$. In the for-each guarantee, one wants to design a distribution over sketches that gives the desired result for a fixed vector $x \in \mathbb{R}^n$. We note that ℓ_∞/ℓ_2, ℓ_2/ℓ_2 are impossible in the for-all model, unless $\Omega(n)$ measurements are used [CDD09]. The standard approach for the for-all guarantee is via RIP matrices, satisfying the so-called Restricted Isometry Property. In what follows, we will refer to the for-each model, unless stated otherwise.

- $\ell_2/\ell_2 : \|x - x'\|_2 \leq (1+\epsilon)\|x_{-k}\|_2$.
- $\ell_1/\ell_1 : \|x - x'\|_1 \leq (1+\epsilon)\|x_{-k}\|_1$.
- $\ell_\infty/\ell_2 : \|x - x'\|_\infty \leq (1+\epsilon)\frac{1}{\sqrt{k}}\|x_{-k}\|_2$.
- $\ell_2/\ell_1 : \|x - x'\|_2 \leq (1+\epsilon)\frac{1}{\sqrt{k}}\|x_{-k}\|_1$.

The first set of schemes that initiated the research on compressed sensing are given in [CRT06b, Don06]. There the authors show, for any $x \in \mathbb{R}^n$, given $y = \Phi x$, it is possible to satisfy the ℓ_2/ℓ_1 guarantee for all vectors, if Φ is a Gaussian matrix with $O(k \log(n/k))$ rows. The schemes in [CM06, CCF02] achieve the ℓ_∞/ℓ_2 guarantee with $O(k \log n)$ measurementz, matching known lower bounds [JST11], $O(n \log n)$ decoding time and $O(\log n)$

update time. The state of the art for ℓ_∞/ℓ_2 is [LNNT16], which gives optimal number of measurements, sublinear decoding time, $O(\log n)$ update time and $1/\operatorname{poly}(n)$ failure probability. Price and Woodruff [PW11] show that in order to get ℓ_2/ℓ_2 with constant failure probability$< 1/2$ with the output being exactly k-sparse output requires $\Omega(\epsilon^{-2}k)$ measurements. They also showed non-k-sparse output requires $\Omega(\epsilon^{-1}k\log(n/k))$ measurements in the regime $\epsilon > \sqrt{k\log n/n}$, and gave an upper bound of $O(\epsilon^{-1}k\log n)$ measurements, showing thus a separation in the measurement complexity between k-sparse and $O(k)$-sparse output. Later, in the breakthrough work of Gilbert, Li, Porat and Strauss [GLPS10] an algorithm that runs in sublinear time, and has $O(\log(n/k)\log^2 k)$ column sparsity, was devised. On generic norms, nearly optimal bounds have been given by Backurs, Indyk, Razenshteyn and Woodruff [BIRW16]. We note, however, that their schemes are not computationally efficient: they have exponential running time, except in the case of Earth-Mover-Distance, which has time polynomial in n and $\log^k n$.

Measurements. The number of measurements corresponds to physical resources: memory in monitoring devices of data streams, number of screens in biological applications, or number of filters in dynamic spectrum access (DSA) of radio signal [HMT+13].

In applications such as medical imaging, it is crucial to reduce the number of measurements, since the radiation used in CT scans could potentially increase cancer risks for patients. For instance, [PSL+12] showed that a positive association between radiation exposure from CT scans in childhood and subsequent risk of leukemia and brain tumors.

For more applications, we refer the readers to [QBI+13].

Encoding Time. Designing algorithms with fast update/encoding time is a well-motivated task for streaming algorithms, since the packets arrive at an extremely fast rate [TZ12]; even logarithmic factors are crucial in these applications. Also in digital signal processing applications, in the design of cameras or satellites which demand rapid imaging, when we observe a sequence of images that are close to each other, we may not need to encode the new signal from the beginning, rather than encode only that part which differs from the current signal; the delay is then defined by the update time of our scheme. Moreover, in Magnetic Resonance Imaging (MRI) update time or encoding time defines the time the patient waits for the scan to happen. Improvement of the runtime has benefits both for patients and for healthcare economics [LDSP08].

A natural question is the following: what are the time limitations of our data structures, regarding update time? Regarding the streaming setting, the first lower bounds are given in [LNN15] for non-adaptive algorithms. An algorithm is called non-adaptive if, during updates, the memory cells are written and read depend only on the index being updated and the random coins tossed before the stream is started to being processed. The lower bounds given concern both randomized and deterministic algorithms; the relevant bounds to sparse recovery are for ℓ_p/ℓ_q estimation. However, for constant failure probability their results do not give anything useful, since their lower bounds start to kick in when the failure probability becomes very small, namely $o(2^{-\sqrt{m \cdot \log n}})$.

For the column sparsity (which could be smaller than update time, and hence the lower bounds in [LNN15] might not apply[1]), the only known lower bounds are known for

[1]the lower bounds in [LNN15] also depend heavily on the streaming model, so they do not transfer necessarily to all scenarios where sparse recovery finds application.

RIP matrices, which are used in the for-all setting. To the best of our knowledge, the first non-trivial lower bounds were given by Nachin [Nac10], and then extended by Indyk and Razenshteyn in [IR13] for RIP-1 model-based compressed sensing matrices. Lower bounds for the column sparsity of RIP-2 matrices were given in Nelson and Nguyên [NN13b], and then to RIP-p matrices in Allen-Zhu, Gelashvili and Razenshteyn [AGR16]. Roughly speaking, the lower bounds for ℓ_2 indicate that if one aims for optimal measurements, $m = k \log(n/k)$, in the regime $k < n/\log^3 n$, one cannot obtain column sparsity better than $\Omega(m)$. This indicates that the for-all case should be significantly worse, in terms of column sparsity, than the for-each case.

Decoding Time. Another very important quantity we want to minimize is the time needed to reconstruct the approximation of x from its compressed version. This quantity is of enormous significance in cases where the universe size is huge and we cannot afford to iterate over it. This is often the case in networking applications, where the universe size is the number of distinct IP addresses. In MRI applications the decoding time corresponds to the time needed to reconstruct the image after the scan has been performed. Decoding time is highly important also in satellite systems, modern radars and airspace surveillance, where compressed sensing have found extensive application [End10].

The sparse Fourier transform can be regarded as another variant of the sparse recovery problem, the results developed along that line being incomparable with traditional sparse recovery results, where we have freedom over the design of the sketching matrix. Sparse Fourier transforms can be divided into two categories, one being sparse discrete Fourier transform, and the other being sparse continuous Fourier transform [BCG+12, Moi15, PS15, CKPS16].

The sparse discrete Fourier transform also can be split into two lines, the first category of results [GGI+02, GMS05, HIKP12a, HIKP12b, Iwe13, IKP14, IK14, Kap16, CKSZ17, Kap17, KVZ19] carefully choose measurements that allow for sublinear recovery time, and the second category of results [CRT06a, RV08, Bou14, HR16] focus proving Restricted Isometry Property. The techniques used in sparse Fourier transforms are related to standard compressed sensing, since the main approaches try to implement arbitrary linear measurements via sampling Fourier coefficients.

10.1.2 Our result

Our main result is a novel scheme for ℓ_2/ℓ_2 sparse recovery. Our contribution lies in obtaining better decoding time, and $O(\log(n/k))$ column sparsity via new techniques. The problem of improving the column sparsity to $O(\log(n/k))$ was explicitly stated in [GLPS10] as an open problem. We completely resolve the open problem. Moreover, as an important technical contribution, we introduce a different approach for sublinear-time optimal-measurement sparse recovery tasks. Since this iterative loop is a crucial component of almost all algorithms in sublinear-time compressed sensing [IPW11, PS12, HIKP12a, GNP+13, IKP14, Kap16, GLPS17, CKSZ17, Kap17, LNW18, NSWZ18], we believe our new approach and ideas will appear useful in the relevant literature, as well as be a starting point for re-examining sparse recovery tasks under a different lens, and obtaining improved bounds.

10.1.3 Notation

For $x \in \mathbb{R}^n$ we let $H(x,k)$ to be the set of the largest k in magnitude coordinates of x. We also write x_S for the vector obtained after zeroing out every $x_i, \notin S$, and $x_{-k} = x_{[n]\setminus H(x,k)}$.

Reference	Measurements	Decoding Time	Encoding Time
[Don06, CRT06b]	$k \log(n/k)$	LP	$k \log(n/k)$
[CCF02, CM06]	$\epsilon^{-2} k \log n$	$\epsilon^{-1} n \log n$	$\log n$
[NT09a]	$k \log(n/k)$	$nk \log(n/k)$	$\log(n/k)$
[CM04]	$\epsilon^{-2} k \log^2 n$	$\epsilon^{-1} k \log^c n$	$\log^2 n$
[CCF02, CM06]	$\epsilon^{-2} k \log^c n$	$\epsilon^{-1} k \log^2 n$	$\log^c n$
[GLPS10]	$\epsilon^{-1} k \log(n/k)$	$\epsilon^{-1} k \log^c n$	$\log(n/k) \cdot \log^2 k$
Our result	$\epsilon^{-1} k \log(n/k)$	$\epsilon^{-1} k \log^2(n/k)$	$\log(n/k)$

Table 10.1: (A list of ℓ_2/ℓ_2-sparse recovery results). We ignore the "O" for simplicity. LP denotes the time of solving Linear Programs [CLS19], and the state-of-the-art algorithm takes n^ω time where ω is the exponent of matrix multiplication. The results in [Don06, CRT06b, NT09a] do not explicitly state the ℓ_2/ℓ_2 guarantee, but their approach obtains it by an application of the Johnson-Lindenstrauss Lemma; they also cannot facilitate $\epsilon < 1$, obtaining thus only a 2-approximation. The c in previous work is a sufficiently large constant, not explicitly stated, which is defined by probabilistically picking an error-correcting code of short length and iterating over all codewords. We estimate $c \geq 4$. We note that our runtime is (almost) achieved

We use $\| \cdot \|_p$ to denote the ℓ_p norm of a vector, i.e. $\|x\|_p = (\sum_{i=1}^n |x_i|^p)^{1/p}$.

10.1.4 Technical statements

We proceed with the definition of the ℓ_2/ℓ_2 sparse recovery problem.

Problem 10.1.1 (ℓ_2/ℓ_2 sparse recovery)**.** *Given parameters ϵ, k, n, and a vector $x \in \mathbb{R}^n$. The goal is to design some matrix $\Phi \in \mathbb{R}^{m \times n}$ and a recovery algorithm $\mathcal{A}$ such that we can output a vector x' based on measurements $y = \Phi x$,*

$$\|x' - x\|_2 \leq (1+\epsilon) \min_{k\text{-sparse}} \|z - x\|_2.$$

We primarily want to minimize m (which is the number of measurements), the running time of $\mathcal{A}$ (which is the decoding time) and column sparsity of Φ.

In table 10.1, we provide a list of the previous results and compare with ours. Here, we formally present our main result.

Theorem 10.1.2 (stronger ℓ_2/ℓ_2 sparse recovery)**.** *There exists a randomized construction of a linear sketch $\Phi \in \mathbb{R}^{m\times n}$ with $m = O(\epsilon^{-1}k\log(n/k))$ and column sparsity $O(\log(n/k))$, such that given $y = \Phi x$, we can find an $O(k)$-sparse vector $x' \in \mathbb{R}^n$ in $O(m \cdot \log(n/k))$ time such that*

$$\|x' - x\|_2 \leq (1+\epsilon) \min_{k\text{-sparse } z\in\mathbb{R}^n} \|z - x\|_2.$$

holds with 9/10 probability.

Remark 10.1.1. In the regime where k is very close to n, for example $k = n/\text{poly}(\log n)$, we get an exponential improvement on the column sparsity over [GLPS10]. In many applications of compressed sensing, this is the desired regime of interest, check for example Figure 8 from [BI08]: $n = 71,542$ while $m \geq 10,000$, which corresponds to k being very close to n.

Remark 10.1.2. As can be inferred from the proof, our algorithms runs in time $O((k/\epsilon)\log^2(\epsilon n/k) + (k/\epsilon)\log(1/\epsilon))$, which is slightly better than the one stated in Theorem 10.1. The algorithm in [HIKP12a] achieves also the slightly worse running time of $O((k/\epsilon)\log n\log(n/k))$. That algorithm was the first algorithm that achieved running time $O(n\log n)$ for all values of k, ϵ for which the measurement complexity remained sublinear, smaller than γn, for some absolute constant γ. A careful inspection shows that our algorithm achieves running time that is **always** sublinear, as long as the measurement complexity remains smaller than γn.

10.1.5 Overview of techniques and difference with previous work

This subsection is devoted to highlighting the difference between our approach and the approach of [GLPS10]. We first give a brief high-level description of the state of the art algorithm before our work, then discuss our techniques, and try to highlight why the previous approach could not obtain the stronger result we present in this paper. Lastly, we show how our ideas can be possibly applied to other contexts.

10.1.5.1 Summary of [GLPS10].

The algorithm of [GLPS10] consists of $O(\log k)$ rounds: in the r-th round the algorithm finds a constant fraction of the remaining heavy hitters. Beyond this iterative loop lies the following idea about achieving the ℓ_2/ℓ_2 guarantee: in order to achieve it, you can find all but $\frac{k}{3^r}$ heavy hitters i such that $|x_i|^2 = \Omega(\frac{2^r\epsilon}{k}\|x_{-k}\|_2^2)$. This means that the algorithm is allowed to "miss" a small fraction of the heavy hitters, depending on their magnitude. For example, if all heavy hitters are as small as $\Theta(\sqrt{\epsilon/k}\|x_{-k}\|_2)$, a correct algorithm may even not find any of them. This crucial observation leads naturally to the main iterative loop of combinatorial compressed sensing, which, as said before, loop proceeds in $O(\log k)$ rounds. Every round consists of an identification and an estimation step: in the identification step most heavy hitters are recognized, while in the estimation step most of them are estimated correctly. Although in the estimation step some coordinates might have completely incorrect estimates, this is guaranteed (with some probability) be fixed in a later round. The reason why this will be fixed is the following. If a coordinate i is badly estimated, then it will appear very large in the residual signal and hence will be identified in later rounds, till it is estimated correctly. One can observe that the correct estimation of that round for coordinate

i cancels out (remedies) the mistakes of previous rounds on coordinate i. Thus, the identification and estimation procedure, which are interleaved, work complementary to each other. The authors of [GLPS10] were the first that carefully managed to argue that identifying and estimating a constant fraction of heavy hitters per iteration, gives the optimal number of measurements.

More formally, the authors prove the following iterative loop invariant, where $k_r = k3^{-r}, \epsilon_r = \epsilon_r 2^{-r}$ for $r \in [R]$ with $R = \log k$: Given $x \in \mathbb{R}^n$ there exists a sequence of vectors $\{x^{(r)}\}_{r\in[R]}$, such that $x^{(r+1)} = x^{(r)} - \widehat{x}^{(r)}$ and

$$\|(x - \widehat{x})_{-k_r}\|_2^2 \leq (1 + \epsilon_r)\|x_{-k_r}\|_2^2. \tag{10.1}$$

In the end, one can apply the above inequality inductively to show that

$$\|x - \sum_{r=1}^{R} \widehat{x}^{(r)}\|_2^2 \leq (1 + \epsilon)\|x_{-k}\|_2^2.$$

We now proceed by briefly describing the implementations of the identification and the estimation part of [GLPS10]. In the identification part, in which lies the main technical contribution of that work, every coordinate $i \in [n]$ is hashed to $O(k/\epsilon)$ buckets and in each bucket $O(\log(\epsilon n/k))$-measurement scheme based on error-correcting codes is used to identify a heavy hitter; the authors carefully use a random error-correcting code of length $O(\log\log(\epsilon n/k))$, so they afford to iterate over all codewords and employ a more sophisticated approach and use nearest-neighbor decoding. This difference is one of the main technical ideas that allow them to obtain $O(\epsilon^{-1}k\log(n/k))$ measurements, beating previous work, but it is also the main reason why they obtain $k \cdot \text{poly}(\log n)$ decoding time[2]: performing

[2]The authors do not specifically address the exponent in the $\text{poly}(\log n)$, but we estimate it to be ≥ 4.

nearest neighbor decoding and storing the code per bucket incurs additional poly$(\log n)$ factors. Moreover, for every iteration r they need to repeat the identification scheme r times in order to bring down the failure probability to 2^{-r}, so that they afford a union-bound over all iterations. This leads to an additional $O(\log^2 k)$ factor in the update time. The estimation step consists of hashing to $O(k/\epsilon)$ buckets and repeating $O(\log(1/\epsilon))$ times. Since the identification step returns $O(k/\epsilon)$ coordinates, the $O(\log(1/\epsilon))$ repetitions of the estimation step ensure that at most $k/3$ coordinates out of the $O(k/\epsilon)$ will not be estimated correctly. This is a desired property, since it allows the algorithm to keep the $2k$ coordinates with the largest estimates, subtract them from x and iterate.

In the next section, we will lay out our approach which improves the decoding time and the column sparsity of [GLPS10]. The iterative procedure of [GLPS10] lies in the heart of most compressed sensing schemes, so we believe that this new approach could be applied elsewhere in the sparse recovery literature.

10.1.5.2 Our approach

As we mentioned before, our approach is totally different from previous work, avoiding the iterative loop that all algorithms before applied. Our algorithm consists of four steps, each one being a different matrix responsible for a different task. The first matrix, with a constant number of rows allows us to approximate the tail of the vector x, an approximation that will appear useful in the next step. The second matrix along with its decoding procedure, which should be regarded as the identification step, enables us to find a list L of size $O(k/\epsilon)$

that contains k coordinates[3], which are sufficient for the ℓ_2/ℓ_2 guarantee. This matrix has $O(\epsilon^{-1}k \cdot \log(\epsilon n/k)))$ rows. The third matrix with its decoding procedure, which should be regarded as the pruning step, takes the aforementioned list L, and prunes it down to $O(k)$ coordinates, which are sufficient for the ℓ_2/ℓ_2 guarantee. This matrix again has $O(\epsilon^{-1}k \cdot \log(1/\epsilon))$ rows, for a total of $O(\epsilon^{-1}k\log(n/k))$ rows. The last matrix is a standard set-query sketch.

Step 1: Tail Estimation

Lemma 10.1.3 (tail estimation)**.** *Let $c_1 \geq 1$ denote some fixed constant. There is an oblivious construction of matrix $\Phi \in \mathbb{R}^{m\times n}$ with $m = O(\log(1/\delta))$ and column sparsity $O(\log(1/\delta))$ such that, given Φx, there is an algorithm that outputs a value $V \in \mathbb{R}$ in time $O(m)$ such that*

$$\frac{1}{10k}\|x_{-c_1\cdot k}\|_2^2 \leq V \leq \frac{1}{k}\|x_{-k}\|_2^2$$

holds with probability $1-\delta$.

Our first step towards the way for stronger sparse recovery is the design a routine that estimates the ℓ_2 norm of the tail of a vector $x \in \mathbb{R}^n$, which we believe might be interesting in its own right. More generally, our algorithm obtains a value V such that $\frac{1}{10k}\|x_{-c_1\cdot k}\|_p^p \leq V \leq \frac{1}{k}\|x_{-k}\|_p^p$, using $O(1)$ measurements. Here c_1 is some absolute constant. To obtain this result we subsample the vector at rate $\Theta(1/k)$ and then use a p-stable distribution to approximate the subsampled vector. While the upper bound is immediate, the Paley-Zygmund inequality

[3]We note that this term is exactly k, not $O(k)$. Although not important for our main result, it will be crucial for some of our applications of our techniques.

does not give a sufficient result for the lower bound, so more careful arguments are needed to prove the desired result. We obtain our result by employing a random walk argument.

One additional possible application in sparse recovery applications where a two-stage scheme is allowed, e.g. [DLTY06, DDT+08], would be to first use the above routine to roughly estimate how many heavy coordinates exist, before setting up the measurements. For example, we could first run the above routine for $k = 1, 2, 2^2, \ldots, 2^{\log n}$, obtaining values $V_1, V_2, V_4, \ldots, V_{\log n}$, and then use these values to estimate the size of the tail of the vector is, or equivalently approximate the size of the set of the heavy coordinates by a number k'. We can then run a sparse recovery algorithm with sparsity k'. Details can be found in Section 10.4.

Step 2: The Identification Step The goal of this step is to output a list L of size $O(k/\epsilon)$ that contains a set of k coordinates that are sufficient to satisfy the ℓ_2/ℓ_2 guarantee. The column sparsity we are shooting for at this point is $O(\log(\epsilon n/k))$, and the decoding time should be $O(m \log(\epsilon n/k))$.

Lemma 10.1.4 (identification sketch)**.** *There exists a randomized construction of a matrix $\Phi \in \mathbb{R}^{m \times n}$, with $m = O(\epsilon^{-1} k \cdot \log(\epsilon n/k))$ and column sparsity $O(\log(\epsilon n/k))$, such that given $y = \Phi x$, one can find a set L of size $O(k/\epsilon)$ in $O(m \log(\epsilon n/k))$ time, such that*

$$\exists T \subset S, |T| \leq k : \|x - x_T\|_2 \leq (1 + \epsilon)\|x_{-k}\|_2,$$

holds with probability at least $9/10$.

For this routine, we will set up a hierarchical separation of $[n]$ to trees. We will call this separation interval forest. we set $\tau = k/\epsilon$. Then we partition $[n]$ into τ intervals of

length $q = n/\tau$, and set up an interval tree for each interval in the following way: every interval tree has branching factor

$$\frac{\log q}{\log \log q}$$

with the same height (this is consistent with the fact that every tree contains q nodes). At the leaves of every interval tree there are the nodes of the corresponding interval.

Our approach consists is now the following. For each level of the interval forest we hash everyone to k/ϵ buckets in the following way: if two coordinates i, i' are in the same interval (node of the interval forest) they are hashed to the same bucket. The above property can be regarded as "hashing interval to buckets". Moreover, every x_i is multiplied by a **Gaussian** random variable. We repeat this process for $\log \log(\epsilon n/k)$ per level.

The decoding algorithm is the following. First, we obtain a value V using the routine in Step 1, Lemma 10.1.3. Then we proceed in a breadth-first (or depth-first, it will make no difference) search manner and find an estimate for every interval, similarly to [LNNT16, CH09], by taking the median of the $\log \log(\epsilon n/k)$ buckets it participates to. There are two technical things one needs to show: First, we should bound the decoding time, as well as the size of the output list L. Second, we need to show that there exists a set T' of size at most k that satisfies the guarantee of the Lemma 10.1.4. For the first part, we show that the branching process defined by the execution of the algorithm is bounded due to the $\log \log(\epsilon n/k)$ repetitions per level. For the second part, we show that for every coordinate $i \in H(x, k)$ the probability that $i \in L$ is proportional to $k|x_i|^2/(\epsilon \|x_{-k}\|_2^2)$. Then we show that the expected ℓ_2^2 mass of coordinates $i \in L \setminus H(x, k)$ is $\epsilon \|x_{-k}\|_2^2$. This suffices to give the desired guarantee for Lemma 10.1.4. We provide details in Section 10.2.

Step 3: The Pruning Step

Lemma 10.1.5 (pruning sketch)**.** *Let $c_2, c_3 > 1$ denote two fixed constants. There exists a randomized construction of a matrix $\Phi \in \mathbb{R}^{m \times n}$, with $m = O(\epsilon^{-1}k \cdot \log(1/\epsilon))$, with column sparsity $O(\log(1/\epsilon))$ such that the following holds :*
Suppose that one is given a (fixed) set $L \subseteq [n]$ such that

$$|L| = O(k/\epsilon), \qquad \exists T \subset L, |T| \leq k : \|x - x_T\|_2 \leq (1+\epsilon)\|x_{-k}\|_2.$$

Then one can find a set S of size $c_2 \cdot k$ in time $O(m)$, such that

$$\|x - x_S\|_2 \leq (1 + c_3 \cdot \epsilon)\|x_{-k}\|_2$$

holds with probability $9/10$.

We will now prune the list L obtained from the previous step, to $O(k)$ coordinates. We are going to use $O(\epsilon^{-1}k \cdot \log(1/\epsilon))$ measurements. We hash every coordinate to k/ϵ buckets, combining with **Gaussians**, and repeating $O(\log(1/\epsilon))$ times. A similar matrix was also used in [GLPS10], but there the functionality and the analysis were very different; moreover, the authors used random signs instead of Gaussians. We will heavily exploit the fact that a standard normal g satisfies $\Pr[|g| < x] = O(x)$. The reasons why we need Gaussians is the following: if we have a lot of large coordinates which are equal and much larger than $\sqrt{\epsilon}\|x_{-k}\|_2$, we want to find all of them, but due to the use of random signs, they might cancel each other in a measurement. Switching to Gaussians is the easiest way of avoiding this undesirable case.

Our algorithm computes, for every $i \in L$, an estimate $\widehat{x}_i$ by taking the median of the $O(\log(1/\epsilon))$ buckets it participates to, and then keeps the largest $O(k)$ coordinates in

magnitude to form a set S. We then show that these coordinates satisfy the ℓ_2/ℓ_2 guarantee. We will say that a coordinate is well-estimated if $|\widehat{x}_i| = \Theta(|x_i|) \pm \sqrt{\epsilon k^{-1}}\|x_{-k}\|_2$. For the analysis, we define a threshold $\tau = \|x_{-k}\|_2/\sqrt{k}$, and classify coordinates based on whether $|x_i| \geq \tau$ or not.

- In the case $|x_i| \geq \tau$ the expected mass of these coordinates $i \notin S$ is small;
- In the other case the number of coordinates i with $|x_i| < \tau$ are $O(k)$. This allows us to employ an exchange argument, similar to previous work, e.g. [PW11], but more tricky due to the use of Gaussians instead of random signs.

We note that the way we compute the estimates $\widehat{x}_i$ is different from previous work: one would expect to divide the content of a bucket that i hashed to by the coefficient assigned to x_i, in order to get an unbiased estimator, but this will not work. The details can be found in Section 10.3.

In the end, this step gives us a set S suitable for our goal, but **does not** give good estimations of the coordinates inside that set. For that we need another, standard step.

Step 4: Set Query We estimate every coordinate in S using a set query algorithm of Price [Pri11], obtaining the desired guarantee. This matrix needs only $O(k/\epsilon)$ measurements, and runs in $O(k)$ time, while having constant column sparsity.

Lemma 10.1.6 (set query, [Pri11])**.** *For any $\epsilon \in (0, 1/10]$. There exists a randomized construction of a matrix $\Phi \in \mathbb{R}^{m \times n}$, with $m = O(k/\epsilon)$ and column sparsity $O(1)$, such that given $y = \Phi x$ and a set $S \subseteq [n]$ of size at most k, one can find a k-sparse vector $\widehat{x}_S$, supported*

on S in $O(m)$ time, such that

$$\|\widehat{x}_S - x_S\|_2^2 \leq \epsilon \|x_{[n]\setminus S}\|_2^2.$$

holds with probability at least $1 - 1/\operatorname{poly}(k)$.

Our Theorem 10.1.2 follows from the above four steps by feeding the output of each step to the next one. In the end, we rescale ϵ. More specifically, by the identification step we obtain a set L of size $O(k/\epsilon)$ which contains a subset of size k that satisfies the ℓ_2/ℓ_2 guarantee. Then, the conditions for applying the pruning step are satisfied, and hence we can prune the set L down to $O(k)$ coordinates, which satisfy the ℓ_2/ℓ_2 guarantee. Then we apply the set-query sketch to obtain estimates of these coordinates.

In what follows we ignore constant terms. The number of measurements in total is

$$\underbrace{1}_{\text{tail estimation}} + \underbrace{(k/\epsilon)\log(\epsilon n/k)}_{\text{identification step}} + \underbrace{(k/\epsilon)\log(1/\epsilon)}_{\text{pruning step}} + \underbrace{(k/\epsilon)}_{\text{set query}}.$$

The decoding time equals

$$\underbrace{1}_{\text{tail estimation}} + \underbrace{(k/\epsilon)\log(\epsilon n/k)\log(\epsilon n/k)}_{\text{identification step}} + \underbrace{(k/\epsilon)\log(1/\epsilon)}_{\text{pruning step}} + \underbrace{k}_{\text{set query}}.$$

The column sparsity equals

$$\underbrace{1}_{\text{tail estimation}} + \underbrace{\log(\epsilon n/k)}_{\text{identification step}} + \underbrace{\log(1/\epsilon)}_{\text{pruning step}} + \underbrace{1}_{\text{set query}}.$$

10.1.6 Possible applications of our approach to other problems

Exactly k-sparse signals. When the vector we have to output has to be k-sparse, and not $O(k)$-sparse, the dependence on ϵ has to be quadratic [PW11]. Our algorithm yields a

state of the art result for this case, too. One can observe that the analysis of the algorithm in the Identification Linear Sketch outputs a $O(k/\epsilon)$-sized set which contains a set T of size k that allows ℓ_2/ℓ_2 sparse recovery. Performing a CountSketch with $O(k/\epsilon^2)$ columns and $O(\log(k/\epsilon)$ rows, and following a standard analysis, one can obtain a sublinear algorithm with measurement complexity $O(\epsilon^{-1}k\log(\epsilon n/k)+\epsilon^{-2}k\log(k/\epsilon))$. This is an improvemnet on both the runtime and measurement complexity over previous work [CCF02].

Block-Sparse Signals. Our algorithm easily extends to block-sparse signals. For a signal of block size b we obtain sample complexity $O(k/(\epsilon b)\log(bn/k)+(k/\epsilon))$, with a running time nearly linear in k. This matches the sample complexity of previous super-linear algorithms [BCDH10, CIHB09], which also could not facilitate $\epsilon < 1$.

Phaseless Compressed Sensing. In Phaseless Compressed Sensing, one wants to design a matrix Φ with m rows, such that given $y = \Phi x$, one can find a vector $\widehat{x}$ such that $\min_{\theta\in[0,2\pi]}\|x-e^{i\theta}\widehat{x}\|_2 \leq (1+\epsilon)\|x_{-k}\|_2$. This problem has received a fair amount of attention [OYDS11, LV13, CBJC14, YLPR15, PYLR17, Nak17a, LN18], and the state of the art algorithm has $O(k\log n)$ measurement complexity [Nak17a, LN18]. One of the problems is that the iterative loop approach cannot be used here, since it is heavily based on the linearity of the sketch. However, our identification and pruning step do not use the linearity of the sketch, and work also with phaseless measurements. Previous algorithms such as [Nak17a, LN18] suffered a $k\log n$ factor in the number of measurements already from the first step, but this is avoidablue using our new approach. We hope to see an extension down this avenue that gives $O(k\log(n/k))$ measurements.

One-Bit Compressed Sensing. Another important subfield of Sparse Recovery is one-bit compressed sensing, where one has access only to one-bit measurements, i.e. $y = \text{sign}(Ax)$, where the sign function on vectors should be understood as pointwise application of the sign function on each entry. Sublinear algorithms appear in [Nak17b, Nak19], but they both do not obtain the optimal number of measurements in terms of k and n, which is $k \log(n/k)$, but rather the slightly suboptimal $k \log n$. One of the most important reasons is that the iterative loop cannot be implemented in such a scenario. It is a natural question whether our new approach can give the optimal number of measurements. The thresholding step, namely the part where we take use V to filter out non-heavy intervals cannot be implemented here, but perhaps there is still a way to make a similar argument. One first approach should be to show that sublinear decoding with optimal measurements is achieved using non-adaptive threshold measurements, such as in [KSW16] and [BFN+17] (note that the latter one uses adaptive measurements though).

Sparse Fourier Transform. The standard approach to discrete sparse Fourier transform, is to implement linear measurements by using Fourier measurements [GGI+02, GMS05, HIKP12a, HIKP12b, Iwe13, IKP14, IK14, Kap16, CKSZ17, Kap17]. The idea is to hash the spectrum to B buckets by carefully multiplying in the time-domain the vector x with a sparse vector z. In the frequency domain this corresponds to convolving the spectrum of the x with an approximation of the filter of an indicator function of an interval of length roughly B. Due to the Uncertainty Principle, however, one has to exchange measurement complexity and decoding time with the quality of the filter. For example, implementing hashing to buckets using "crude" filters leads to leakage in subsequent buckets, giving additional error terms.

When iterating as usual, these errors accumulate and make identification much harder. The sophisticated approach of [Kap17] manages to design an iterative algorithm, in the same vein with previous algorithms, which takes $O(\epsilon^{-1}k \log n)$ measurements. It would be interesting to see if the approach we suggest avoids some of the problems created by this iterative loop, and can give simpler and faster sparse Fourier transform schemes. It would be interesting to obtain such a result even using adaptive measurements. The work [CKSZ17] has some interesting ideas in the context of block-sparse vectors that could be relevant.

10.2 The Identification Linear Sketch

The goal of this section is to prove the following result,

Theorem 10.2.1 (Restatement of Lemma 10.1.4)**.** *Let $C_L > 1$ be a fixed constant. There exists a randomized construction of a matrix $\Phi \in \mathbb{R}^{m \times n}$, with*

$$m = O(\epsilon^{-1} k \log(\epsilon n/k)),$$

with column sparsity $O(\log(\epsilon n/k))$ such that given $y = \Phi x$, one can find in time $O(m \log(n/k))$ a set L of size $C_L \cdot k/\epsilon$, such that

$$\exists T \subset L, |T| \leq k : \|x - x_T\|_2 \leq (1+\epsilon)\|x_{-k}\|_2,$$

with probability 9/10*.*

In Section 10.2.1, we provide the definition of sketching matrix Φ and present the decoding algorithm. We proved some concentration result in Section 10.2.2. We analyzed the running time of algorithm in Section 10.2.3. We proved the guarantees of the algorithm in Section 10.2.4. Finally, we bound the number of measurements in Section 10.2.5.

10.2.1 Design of the sketch and decoding algorithm

We are going to use a hierarchical separation of $[n]$ into intervals. We will call this separation an interval forest.

Before discussing the matrix, we need the following definitions. We define

Definition 10.2.1 (size of the each tree in the forest)**.** Let

$$\tau = k/\epsilon,$$

Notation	Choice	Statement	Parameter
C_H	4	Definition 10.2.2	H
C_R	100	Definition 10.2.3	R
C_B	10^5	Definition 10.2.4	B
C_0	10^3	Lemma 10.4.3	Blow up on tail size
C_L	10^4	Lemma 10.2.5	L
η	$1/9$	Lemma 10.2.2,10.2.3	Shrinking factor on V
ζ	$1/4000$	Lemma 10.2.2,10.2.3	$\zeta \leq \eta/400$

Table 10.2: Summary of constants in Section 10.2, the column "Parameter" indicates which parameter is depending on that constant. Note that constants C_H, C_R, C_B, C_0, η are used in both algorithm and analysis, but constants C_L and ζ are only being used in analysis. C_L is the related to the guarantee of the output of the algorithm.

assuming that $k/\epsilon \leq n/16$. The size of each tree in the forest is

$$q = n/\tau = n\epsilon/k$$

Definition 10.2.2 (degree and height of the interval forest)**.** Let $C_H > 1$ be a sufficiently large constant such that

$$(\log q/\log\log q)^{C_H \log q/\log\log q} \geq q.$$

Let D denote the degree of the tree, and let H denote the height of the tree. We set D and H, $D = \lceil \log q/\log\log q \rceil$, and $H = \lceil C_H \log q/\log\log q \rceil$.

Definition 10.2.3 (number of repetitions per level)**.** Let R denote the number of repetitions in each level. Let $C_R > 1$ denote some sufficiently large constant. We set $R = C_R \log\log q$.

For $\ell \in \{0, 1, \ldots, H\}$ we define $\mathcal{I}_\ell$, which is a family of sets. Every set $\mathcal{I}_\ell$ is a decomposition of $[n]$ to τD^ℓ intervals of (roughly) the same length. The set $\mathcal{I}_0$ is a decomposition

of $[n]$ to τ intervals of (roughly) the same length length q. If needed, we can round q to a power of 2. This means that

$$\mathcal{I}_0 = \{I_{0,1}, I_{0,2}, \ldots\}$$

where

$$I_{0,1} = [1, \tau], I_{0,2} = [\tau + 1, 2\tau], \ldots$$

We can conceptualize these sets as a forest consisting of τ trees, each of branching factor D and height H, where the ℓ-th level partitions $[n]$ into disjoint τD^ℓ intervals of length $n/(\tau \cdot D^\ell) = q \cdot D^{-\ell}$. For $\ell \in \{0, \ldots, H\}$, interval $I_{\ell,j}$ is decomposed to D disjoint and continuous intervals

$$I_{\ell+1,j\cdot D+1}, \ldots, I_{\ell+1,(j+1)\cdot D}$$

of the same length, except possibly the last interval.

We say that an interval $I_{\ell+1,j}$ is a child of interval $I_{\ell,j'}$ if $j' = \lceil j/D \rceil$.

Definition 10.2.4 (sketching matrix Φ)**.** Let $C_B > 1$ be a sufficiently large constant. Let $B = C_B k/\epsilon$. Let matrices $\Phi^{(1)}, \ldots, \Phi^{(H)}$, where every matrix $\Phi^{(\ell)}$ consists of R submatrices $\{\Phi_r^{(\ell)}\}_{r\in[R]}$. For every $\ell \in [H]$ and $r \in [R]$, we pick 2-wise independent hash functions $h_{\ell,r} : [\tau D^\ell] \to [B]$.

We define measurement $y_{\ell,r,b} = (\Phi_r^{(\ell)} x)_b$ as:

$$y_{\ell,r,b} = \sum_{j \in h_{\ell,r}^{-1}(b)} \sum_{i \in I_{\ell,j}} g_{i,\ell,r} x_j,$$

where $g_{i,\ell,r} \sim \mathcal{N}(0,1)$, i.e. independent standard Gaussians.

We slightly abuse notation and treat y as matrix the mapping to vector should be clear.

Note that C_B should be chosen such that $C_B \gg C_0$, where C_0 appears in tail estimation in Lemma 10.4.3.

10.2.2 Concentration of estimation

In the following lemmata, $\zeta, \eta \in (0,1)$ are absolute constants with $1 > \eta > 1/10 > \zeta$ (See a choice for our application in Table 10.2). The exact values of the constants will be chosen below.

The following lemma handles the probability of detecting a heavy interval at level ℓ.

Lemma 10.2.2 (handling the probability of catching a heavy hitter)**.** *Let $\overline{V}$ be the value in Line 8 of Algorithm 10.1. Let $V = \epsilon\overline{V}$ be the value in Line 9 of Algorithm 10.1. Let $j' \in T_{\ell-1}$, and let j be one of its children. Let z_j be defined as follows,*

$$z_j = \operatorname*{median}_{r\in[R]} \left|y_{\ell,r,h_{\ell,r}(j)}\right|^2.$$

If $\|x_{I_{\ell,j}}\|_2^2 \geq C_j \frac{\epsilon}{k}\|x_{-k}\|_2^2$, where $C_j \geq 2$, then with probability $1 - C_j^{-R/6}$ (over the randomness of $h_{\ell,r}$ and $g_{i,\ell,r}$ for $r \in [R], i \in [n]$ in Definition 10.2.4), we have that

$$z_j \geq \eta V.$$

Proof. Fix $r \in [R]$. Let $b = h_{\ell,r}(j)$, and define $J = \cup_{t\in h_{\ell,r}^{-1}(b)} I_{\ell,t}$. We observe that

$$|y_{\ell,r,b}|^2 = \|x_J\|_2^2 g^2, \text{ where } g \sim \mathcal{N}(0,1).$$

Algorithm 10.1 Interval-forest Sparse Recovery

procedure IntervalForestSparseRecovery(x, n, k, ϵ) ▷ Theorem 10.2.1
  Choose constants C_H, C_R, η, C_0 ▷ According to Table 10.2
  $\tau \leftarrow (k/\epsilon)$ ▷ Definition 10.2.1
  $q \leftarrow n/\tau$ ▷ Definition 10.2.1
  $H \leftarrow \lceil C_H \log q / \log\log q \rceil$ ▷ Definition 10.2.2
  $D \leftarrow \lceil \log q / \log\log q \rceil$ ▷ Definition 10.2.2
  $R \leftarrow \lceil C_R \log\log q \rceil$ ▷ Definition 10.2.3
  $\overline{V} \leftarrow$ LpLpTailEstimation$(x, k, 2, C_0, 1/100)$ ▷ Algorithm 10.3
  $V \leftarrow \epsilon \overline{V}$ ▷ Lemma 10.2.2
  $T_0 \leftarrow \{I_{0,1}, \ldots, I_{0,\tau}\}$
  for $\ell = 1 \to H$ **do**
    $T_\ell \leftarrow$ RecursiveBTree$(\ell, R, D, \eta, T_{\ell-1}, V)$ ▷ Lemma 10.2.5
  end for
  $L \leftarrow T_H$
  return L ▷ Lemma 10.2.6
end procedure
procedure RecursiveBTree(ℓ, R, D, η, T, V)
  $T' \leftarrow \emptyset$
  for $t \in T$ **do**
    Let $I_{\ell,j_1}, I_{\ell,j_2}, \ldots, I_{\ell,j_D}$ denote the child intervals of $I_{\ell-1,t}$
    for $p \in [D]$ **do**
      $z_{j_p} \leftarrow \mathrm{median}_{r\in[R]} |y_{\ell,r,h_{\ell,r}(j_p)}|^2$ ▷ Definition 10.2.4
    end for
    if $z_{j_p} \geq \eta V$ **then**
      $T' \leftarrow T' \cup \{j_p\}$
    end if
  end for
  return T'
end procedure

By property of Gaussian distribution, we have

$$\Pr\left[|y_{\ell,r,b}|^2 \leq (\eta/C_j)\|x_J\|_2^2\right] \leq \frac{2}{\sqrt{2\pi}}\sqrt{\frac{\eta}{C_j}} \leq \sqrt{\frac{2\eta}{\pi C_j}},$$

It implies, since $\|x_J\|_2^2 \geq \|x_{I_{\ell,j}}\|_2^2 \geq C_j \frac{\epsilon}{k}\|x_{-k}\|_2^2$, that

$$\Pr\left[|y_{\ell,r,b}|^2 \leq \eta \frac{\epsilon}{k}\|x_{-k}\|_2^2\right] \leq \sqrt{\frac{2\eta}{\pi C_j}}.$$

Since Lemma 10.4.3, we have $\overline{V} \leq \frac{1}{k}\|x_{-k}\|_2^2$. Because $V = \epsilon\overline{V}$,

$$\Pr\left[|y_{\ell,r,b}|^2 \leq \eta V\right] \leq \sqrt{\frac{2\eta}{\pi C_j}}.$$

The R repetitions ensure that the failure probability can be driven down to $C_j^{-R/6}$, because

$$\begin{aligned}
\Pr[z_j \leq \eta V] &\leq \binom{R}{R/2} \cdot \left(\sqrt{\frac{2\eta}{\pi C_j}}\right)^{R/2} \\
&\leq 2^R \cdot (2\eta/\pi)^{R/4} \cdot C_j^{-R/4} \\
&\leq (2^R \cdot (2\eta/\pi)^{R/4} \cdot 2^{-R/12}) \cdot C_j^{-R/6} \\
&\leq \left(2^{11} \cdot (2/9\pi)^3\right)^{R/12} \cdot C_j^{-R/6} \\
&\leq C_j^{-R/6},
\end{aligned}$$

where the first step follows from a union bound, the third step follows from $C_j \geq 2$, and the forth step follows from $\eta \leq 1/9$.

□

The following lemma handles the probability of a non-heavy interval being considered "heavy" by the algorithm at level ℓ.

Lemma 10.2.3 (handling the probability of false positives)**.** *Let V be the value in Line 9 of Algorithm 10.1. Let j' be an index in $T_{\ell-1}$, and let j be one of its children. If $\|x_{I_{\ell,j}}\|_2^2 \leq$*

$\zeta\frac{\epsilon}{k}\|x_{-C_0k}\|_2^2$, *then with probability* $1-2^{-R/3}$ *(over the randomness of* $h_{\ell,r}$ *and* $g_{i,\ell,r}$ *for* $r\in[R], i\in[n]$ *in Definition 10.2.4) we have that*

$$z_j < \eta V.$$

Proof. Fix $\ell\in[H]$ and consider the set H_ℓ that contains the C_0k coordinates j'' with the largest $\|x_{I_{\ell,j''}}\|_2^2$ values. Define $H_\ell^{(j)} = H_\ell\setminus\{j\}$. Fix $r\in[R]$ and observe that by a union-bound we get that

$$\Pr\left[\exists j''\in H_\ell^{(j)} \;\middle|\; h_{\ell,r}(j)=h_{\ell,r}(j'')\right] \le C_0k\cdot\frac{1}{C_Bk/\epsilon} = \frac{C_0\epsilon}{C_B}\le\frac{1}{20},$$

because $C_B\ge 20C_0$.

We condition on the event $\forall j''\in H_\ell^{(j)} : h_{\ell,r}(j)\neq h_{\ell,r}(j'')$. A standard calculation now shows that

$$\begin{aligned}\mathbb{E}\left[|y_{\ell,r,h_{r,\ell}(j)}|^2\right] &\le \|x_{I_{\ell,j}}\|_2^2 + \frac{1}{C_B}\frac{\epsilon}{k}\|x_{-C_0k}\|_2^2\\ &\le \zeta\frac{\epsilon}{k}\|x_{-C_0k}\|_2^2 + \frac{1}{C_B}\frac{\epsilon}{k}\|x_{-C_0k}\|_2^2\\ &\le \frac{2\zeta\epsilon}{k}\|x_{-C_0k}\|_2^2,\end{aligned}$$

where the last step follows from $\frac{1}{C_B}<\zeta$.

We now apply Markov's inequality to obtain

$$\begin{aligned}\Pr\left[|y_{\ell,r,h_{\ell,r}(j)}|^2\ge\eta V\right] &\le \Pr\left[|y_{\ell,r,h_{\ell,r}(j)}|^2\ge\frac{\eta\epsilon}{10k}\|x_{-C_0k}\|_2^2\right]\\ &\le \frac{\frac{2\zeta\epsilon}{k}\|x_{-C_0k}\|_2^2}{\frac{\eta\epsilon}{10k}\|x_{-C_0k}\|_2^2}\\ &= \frac{20\zeta}{\eta}\\ &\le \frac{1}{20}, && \text{by } \zeta\le\eta/400.\end{aligned}$$

By a union bound, the unconditional probability $\Pr\left[|y_{\ell,r,h_{\ell,r}(j)}|^2 \geq \eta V\right] \leq \frac{1}{10}$. Finally, we can upper bound the probability that $z_j = \text{median}_{r\in[R]}|y_{\ell,r,h_{\ell,r}(j)}|^2$ is greater than ηV,

$$\begin{aligned}\Pr[z_j \geq \eta V] &\leq \binom{R}{R/2}(1/10)^{R/2} \\ &< 2^R \cdot 2^{-\frac{3}{2}R} \\ &= 2^{-\frac{R}{2}} \\ &< 2^{-\frac{R}{3}}.\end{aligned}$$

□

10.2.3 Analysis of running time

Lemma 10.2.4 (bounds of D, H with respect to R)**.** *Let D, H as in Definition 10.2.2. It holds that $D \leq 2^{\frac{R}{6}-10}$, and $H \leq 2^{\frac{R}{6}-10}$.*

Proof. Since $H \geq D$, it suffices to prove the claim only for H.

$$H = C_H \frac{\log q}{\log\log q} < C_H \log q = C_H \log(\epsilon n/k) = C_H 2^{\log\log(\epsilon n/k)} \leq 2^{\frac{R}{6}-10},$$

where the third step follows from $q = \epsilon n/k$, and the last step follows from $\log\log(\epsilon n/k) + \log C_H \leq (C_R/6)\log\log(\epsilon n/k) - 10$ and note that $\log\log(\epsilon n/k) \geq 2$ because we assume $k/\epsilon \leq n/16$.

□

Lemma 10.2.5 (running time)**.** *Let R as in Definition 10.2.3 and D, H as in Definition 10.2.2. Let T_H be the set obtained by applying procedure* RecursiveBTree *H times (lines*

11-13) of Algorithm 10.1, and let $C_L > 1$ be some sufficiently large absolute constant. With probability $1 - H \cdot 2^{-R/6+1}$ we have that:

- $|T_H| \leq C_L \cdot k/\epsilon$,
- *The running time of* BTreeSparseRecovery *is*

$$O\left(\epsilon^{-1} k \cdot \log(\epsilon n/k) \cdot D\right).$$

Proof. Let $C_L = 2(C_0 + 1/\zeta)$.

First, it is easy to see that $|T_0|$ is bounded by

$$|T_0| = \tau = k/\epsilon < C_L k/\epsilon.$$

We claim that if we condition on the event that $|T_{\ell-1}| \leq C_L k/\epsilon$, then with probability $1 - 2^{-R/6+1}$, $|T_\ell| \leq C_L k/\epsilon$. The proof of both bullets will then follow by a union-bound over all H levels. Indeed, consider the set Q_ℓ containing the $|T_{\ell-1}|D \leq C_L \cdot (k/\epsilon) \cdot D$ coordinates j that are children of some $j' \in T_{\ell-1}$. Define

$$B_\ell = \left\{ j \in Q_\ell \;\middle|\; \|x_{I_{\ell,j}}\|_2^2 \leq \zeta \frac{\epsilon}{k} \|x_{-C_0 k}\|_2^2 \right\}.$$

By definition of B_ℓ and Q_ℓ, we have

$$|B_\ell| \leq |Q_\ell| \leq C_L \cdot (k/\epsilon) \cdot D.$$

Moreover, Lemma 10.2.3 gives

$$\forall j \in B_\ell, \Pr\left[z_j \geq \eta V\right] \leq 2^{-R/3}$$

Define random variables W_j to be 1 if $z_j \geq \eta V$, and 0 otherwise. Then

$$\mathbb{E}\left[\sum_{j \in B_\ell} W_j\right] \leq \frac{C_L k}{\epsilon} D \cdot 2^{-R/3}$$

An application of Markov's inequality gives

$$\Pr\left[\sum_{j \in B_\ell} W_j \geq \frac{C_L k}{2\epsilon} D \cdot 2^{-R/6}\right] \leq 2^{-R/6+1}.$$

Conditioning on $\sum_{j \in B_\ell} W_j \leq \frac{C_L k}{2\epsilon} D \cdot 2^{-R/6}$ we will upper bound the size of T_ℓ.

First, observe that there exist at most $(C_0 k + k/(\zeta\epsilon))$ $j \in \mathcal{I}_\ell$ for which $\|x_{I_{\ell,j}}\|_2^2 > \zeta \frac{\epsilon}{k} \|x_{-C_0 k}\|_2^2$. This gives

$$|T_\ell| \leq \left(C_0 k + \frac{k}{\zeta\epsilon}\right) + \frac{C_L k}{2\epsilon} D 2^{-R/6}. \tag{10.2}$$

If $C_L \geq 2(C_0 + 1/\zeta)$, then we can upper bound the first term in Eq. (10.2),

$$C_0 k + \frac{k}{\zeta\epsilon} \leq \left(C_0 + \frac{1}{\zeta}\right) \frac{k}{\epsilon} \leq \frac{1}{2} \frac{C_L k}{\epsilon}.$$

For the second term in Eq. (10.2), we can show that

$$\frac{C_L k}{2\epsilon} \cdot D 2^{-R/6} \leq \frac{1}{2} \frac{C_L k}{\epsilon},$$

or equivalently $D \leq 2^{R/6}$, which holds by Lemma 10.2.4.

We have D levels, and at each level ℓ we have $|T_\ell| = O(k/\epsilon)$, conditioned on the aforementioned events happening. The children of T_ℓ is then $O(k/\epsilon \cdot D)$. Since we have R repetitions the total running time per level is $O((k/\epsilon) \cdot D \cdot R)$, and the total running time is

$$O((k/\epsilon) \cdot D \cdot R \cdot H) = O\left((k/\epsilon) \cdot D \cdot \log\log q \cdot \frac{\log q}{\log\log q}\right) = O\left((k/\epsilon) \cdot D \cdot \log(\epsilon n/k)\right),$$

where the first step follows from definition of R and H, and the last step follows from definition of q.

Therefore, it gives the desired result. □

10.2.4 Guarantees of the algorithm

Lemma 10.2.6 (guarantees)**.** *Let $L = T_H$ be the set obtained by applying procedure* RecursiveBTree *R times (lines 11-13) of Algorithm 10.1, we have that, with probability* $9/10$*, there exist $T' \subseteq L$ of size at most k, such that*

$$\|x - x_{T'}\|_2^2 \leq (1+\epsilon)\|x_{-k}\|_2^2.$$

Proof. Define

$$\mathcal{H} = \left\{ j \in H(x,k) \;\middle|\; \exists C_j \geq 2, |x_j|^2 \geq C_j \frac{\epsilon}{k} \|x_{-k}\|_2^2 \right\}.$$

Moreover, associate every $j \in \mathcal{H}$ with its corresponding $C_j = \frac{|x_j|^2}{\frac{\epsilon}{k}\|x_{-k}\|_2^2}$.

Pick $j \in \mathcal{H}$. Let also $j_1, j_2, \ldots, j_H$, be numbers such that

$$j \in I_{1,j_1}, j \in I_{2,j_2}, \ldots, j \in I_{R,j_H}.$$

For any $t \in \{1, \ldots, H-1\}$, if $I_{t,j_t} \in T_t$, then $I_{t+1,j_{t+1}} \in T_{t+1}$ with probability $1 - C_j^{-R/6}$, by Lemma 10.2.2. Since $C_j \geq 2$, this allows us to take a union bound over all H levels and claim that with probablity $1 - HC_j^{-R/6}$, $j \in T_R$. For $j \in \mathcal{H}$ define random variable to δ_j to

be 1 if $j \notin T_H$.

$$\begin{aligned}
\mathbb{E}\left[\delta_j x_j^2\right] &\leq H \cdot C_j^{-R/6} x_j^2 \\
&\leq H \cdot C_j^{-R/6} C_j \frac{\epsilon}{k} \|x_{-k}\|_2^2 \\
&= H \cdot C_j^{-R/6+1} \frac{\epsilon}{k} \|x_{-k}\|_2^2 \\
&\leq \frac{\epsilon}{80k} \|x_{-k}\|_2^2,
\end{aligned}$$

where the first step follows by definition of δ_j, the second step follows by the fact that $j \in \mathcal{H}$, and the last step follows by Lemma 10.2.4. Since $|\mathcal{H}| \leq k$, we have that

$$\mathbb{E}\left[\sum_{j \in \mathcal{H}} \delta_j x_j^2\right] \leq |\mathcal{H}| \cdot \frac{\epsilon}{80k} \|x_{-k}\|_2^2 \leq \frac{\epsilon}{80} \|x_{-k}\|_2^2.$$

Then applying Markov's inequality, we have

$$\Pr\left[\sum_{j \in \mathcal{H}} \delta_j x_j^2 > \frac{\epsilon}{2} \|x_{-k}\|_2^2\right] \leq \frac{1}{40}.$$

We condition on the event $\sum_{j \in \mathcal{H}} \delta_j x_j^2 \leq \frac{\epsilon}{2} \|x_{-k}\|_2^2$. Setting $T' = \mathcal{H} \cap T_H$ we observe that

$$\begin{aligned}
\|x - x_{T'}\|_2^2 &= \sum_{j \in \mathcal{H} \cap H(x,k)} \delta_j x_j^2 + \|x_{H(x,k) \setminus \mathcal{H}}\|_2^2 + \|x_{[n] \setminus H(x,k)}\|_2^2 \\
&\leq \frac{\epsilon}{2} \|x_{-k}\|_2^2 + k \cdot \frac{2\epsilon}{k} \|x_{-k}\|_2^2 + \|x_{-k}\|_2^2 \\
&\leq (1 + 3\epsilon) \|x_{-k}\|_2^2.
\end{aligned}$$

where the second step follows by the bound on $\sum_{j \in T} \delta_j x_j^2$ and the fact that every $j \notin T$ satisfies $|x_j|^2 \leq (2\epsilon/k) \|x_{-k}\|_2^2$.

Rescaling for ϵ we get the desired result. □

10.2.5 Bounding the number of measurements

In this subsection, we prove that

Claim 10.2.7 (#measurements)**.** *The number of measurements is* $O(\epsilon^{-1}k \cdot \log(\epsilon n/k))$.

Proof. Recall the definition of τ and q,

$$\tau = k/\epsilon, \quad q = n/\tau.$$

We thus have the following bound on the number of measurements:

$$B \cdot H \cdot R = C_B(k/\epsilon) \cdot C_H \frac{\log(\epsilon n/k)}{\log\log(\epsilon n/k)} \cdot C_R \log\log(\epsilon n/k) = O\left((k/\epsilon)\log(\epsilon n/k)\right).$$

□

10.3 The Pruning Linear Sketch

The goal of this section is to prove Theorem 10.3.1.

Theorem 10.3.1 (Restatement of Lemma 10.1.5)**.** *Let $C_L, \alpha, \beta > 1$ be three fixed constants. There exists a randomized construction of a matrix $\Phi \in \mathbb{R}^{m \times n}$, with $m = O((k/\epsilon) \cdot \log(1/\epsilon))$, with column sparsity $O(\log(1/\epsilon))$ such that the following holds :*
Suppose that one is given a set $L \subseteq [n]$ such that

$$|L| = C_L \cdot k/\epsilon, \qquad \exists T \subset L, |T| \leq k : \|x - x_T\|_2 \leq (1+\epsilon)\|x_{-k}\|_2.$$

Then procedure Prune *(Algorithm 10.2) can find a set S of size $\beta \cdot k$ in time $O(m)$, such that*

$$\|x - x_S\|_2 \leq (1 + \alpha \cdot \epsilon)\|x_{-k}\|_2$$

holds with probability 9/10.

In Section 10.3.1, we provide some basic definitions and description of our algorithm. We analyze the coordinates from several perspectives in Section 10.3.2. We prove the correctness of our algorithm in Section 10.3.3 and analyze time, number of measurements column sparsity, success probability of algorithm in Section 10.3.4.

10.3.1 Design of the sketching matrix, and helpful definitions

Definition 10.3.1 (sketching matrix Φ)**.** Let $C_R, C_B > 1$ be absolute constants. Let $R = C_R \log(1/\epsilon)$. Let $B = C_B k/\epsilon$. For $r \in [R]$, we pick 2-wise independent hash function $h_r : [n] \to [B]$, as well as normal random variables $\{g_{i,r}\}_{i \in [n], r \in [R]}$ and take measurements

$$y_{r,b} = \sum_{i \in h_r^{-1}(b)} x_i g_{i,r}.$$

Notation	Choice	Statement	Parameter
C_R	$10^4 + 500C_L$	Definition 10.3.1	R
C_B	5×10^5	Definition 10.3.1	B
C_g	$4/5$	Fact 10.3.2	Gaussian variable
C_L	10^4	Theorem 10.3.1	L
α	5	Theorem 10.3.1	Blow up on ϵ
β	100	Theorem 10.3.1	Blow up on k

Table 10.3: Summary of constants in Section 10.3, the column "Parameter" indicates which parameter is depending on that constant. Note that set L is the input of the algorithm in Section 10.3 and the output of the algorithm in Section 10.2.

Given the set L, for every $i \in L$ we calculate

$$z_i = \underset{r \in [R]}{\text{median}} |y_{r,h_r(i)}|,$$

and keep the indices i with the βk largest z_i values to form a set S of indices, for some absolute constant β sufficiently large. We describe this pruning step in Algorithm 10.2. For the analysis, we define the threshold

$$\tau = \|x_{-k}\|_2/\sqrt{k}. \tag{10.3}$$

We will need the following standard fact about the Gaussian distribution. Then we proceed with a series of definitions and lemmata.

Fact 10.3.2 (property of Gaussian)**.** *Suppose $x \sim \mathcal{N}(0, \sigma^2)$ is a Gaussian random variable. For any $t \in (0, \sigma]$ we have*

$$\Pr[x \geq t] \in \left[\frac{1}{2}(1 - \frac{4}{5}\frac{t}{\sigma}), \frac{1}{2}(1 - \frac{2}{3}\frac{t}{\sigma})\right].$$

Similarly, if $x \sim \mathcal{N}(\mu, \sigma^2)$, for any $t \in (0, \sigma]$, we have

$$\Pr[|x| \geq t] \in \left[1 - \frac{4}{5}\frac{t}{\sigma}, 1 - \frac{2}{3}\frac{t}{\sigma}\right].$$

Algorithm 10.2 The Prune Procedure

procedure PRUNE(x, n, k, ϵ, L) ▷ Theorem 10.3.1
  $R \leftarrow C_R \log(1/\epsilon)$
  $B \leftarrow C_B k/\epsilon$
  for $r = 1 \to R$ **do**
    Sample $h_r : [n] \to [B] \sim$ 2-wise independent family
    for $i = 1 \to n$ **do**
      Sample $g_{i,r} \sim \mathcal{N}(0, 1)$
    end for
  end for
  for $r = 1 \to R$ **do**
    for $b = 1 \to B$ **do**
      $y_{r,b} \leftarrow \sum_{i \in h_r^{-1}(b)} x_i g_{i,r}$
    end for
  end for
  for $i \in L$ **do**
    $z_i \leftarrow \text{median}_{r \in [R]} |y_{r,h_r(i)}|$
  end for
  $S \leftarrow \{i \in L : z_i$ is in the top βk largest coordinates in vector $z\}$
  return S
end procedure

The form we will need is the following:

$$\Pr_{g \sim \mathcal{N}(0,1)}[|g| \leq t] \leq \frac{4}{5}t.$$

Thought the analysis, for convenience we will set $C_g = 4/5$. Another form we will need is:

$$\Pr_{g \sim \mathcal{N}(0,1)}\left[|g| \in \left[\frac{1}{3C_g}, 2\right]\right] \geq 0.63$$

Proof. The first form is true by simple calculation. The second form is holding due to

numerical values of cdf for normal distribution,

$$\Pr\left[|g_{i,r}| \in \left[\frac{1}{3C_g}, 2\right]\right] = 2(f(2) - f(1/3C_g)) = 2(f(2) - f(5/12)) \geq 2(0.977 - 0.662) = 0.63,$$

where $f(x) = \int_{-\infty}^{x} \frac{e^{-x^2/2}}{\sqrt{2\pi}} dx$ is the cdf of normal distribution. □

Stochastic dominance is a partial order between random variables and it is a classical concept in decision theory and decision analysis [HR69, Baw75]. We give the simplest definition below and it is sufficient for our application.

Definition 10.3.2 (stochastic domination of Gaussian random variables)**.** Let $\sigma_1 < \sigma_2$ and random variables $X \sim \mathcal{N}(0, \sigma_1^2), Y \sim \mathcal{N}(0, \sigma_2^2)$. Then we say that $|Y|$ stochastically dominates $|X|$, and it holds that

$$\Pr\left[|Y| \geq \lambda\right] \geq \Pr\left[|X| \geq \lambda\right], \quad \forall \lambda \geq 0.$$

We formally define the set L as follows:

Definition 10.3.3 (set L, input of the algorithm)**.** Let $C_L > 1$ be a fixed constant, and let set $L \subseteq [n]$ be defined as:

$$|L| = C_L \cdot k/\epsilon, \qquad \exists T \subset |T| \leq k : \|x - x_T\|_2 \leq (1+\epsilon)\|x_{-k}\|_2.$$

We provide a definition called "badly-estimated coordinate",

Definition 10.3.4 (badly-estimated coordinate)**.** We will say a coordinate $i \in [n]$ is badly-estimated if

$$z_i \notin \left[\frac{1}{3C_g}|x_i| - \frac{1}{100}\frac{\sqrt{\epsilon}}{\sqrt{k}}\|x_{-k}\|_2, 2|x_i| + \frac{1}{100}\frac{\sqrt{\epsilon}}{\sqrt{k}}\|x_{-k}\|_2\right],$$

Then, we can define "badly-estimated set",

Definition 10.3.5 (badly-estimated set $\mathcal{B}$)**.** Let set L be defined as Definition 10.3.3. We say $\mathcal{B} \subseteq L$ is a badly-estimated set if for all $i \in \mathcal{B}$, z_i is a badly estimated coordinate (see Definition 10.3.4).

We define a set of large coordinates in head,

Definition 10.3.6 (large coordinates in head)**.** Let τ be defined in (10.3). Let L be defined in Definition 10.3.3. Let C_g be the constant from Fact 10.3.2. Define set

$$\mathcal{M} = \{i \in L \cap H(x,k) : |x_i| \geq 3C_g\tau\},$$

which contains the head coordinates of x that are in L and are larger in magnitude than $3C_g\tau$.

10.3.2 Analyzing head and badly-estimated coordinates

Lemma 10.3.3 (expected error from coordinates above τ)**.** *We have that*

$$\mathbb{E}\left[\sum_{i\in\mathcal{M}} x_i^2 \cdot \mathbb{1}_{z_i<\tau}\right] \leq \frac{\epsilon}{100}\|x_{-k}\|_2^2.$$

Proof. Fix $i \in \mathcal{M}$. Observe that for $r \in [R]$

$$|y'_{r,h_r(i)}| \sim \|x_{h_r^{-1}(i)}\|_2 |\mathcal{N}(0,1)|.$$

Since

$$\|x_{h_r^{-1}(i)}\|_2 \geq |x_i|$$

we have that the random variable $|y'_{r,h_r(i)}|$ stochastically dominates the random variable $|x_i| \cdot |\mathcal{N}(0,1)|$.

By Fact 10.3.2, we have that

$$\Pr\left[|y'_{r,h_r(i)}| \leq \tau\right] \leq C_g \frac{\tau}{|x_i|}.$$

Because of the $R = C_R \log(1/\epsilon)$ repetitions, a standard argument gives that

$$\Pr\left[\mathbb{1}_{z_i<\tau} = 1\right] \leq \left(C_g \frac{\tau}{|x_i|}\right)^{C' \log(1/\epsilon)},$$

for some absolute constant $C' > C_R/3$.

We now bound

$$\begin{aligned}
\mathbb{E}\left[\sum_{i\in\mathcal{M}} x_i^2 \cdot \mathbb{1}_{z_i<\tau}\right] &\leq \sum_{i\in\mathcal{M}} x_i^2 \left(C_g \frac{\tau}{|x_i|}\right)^{C' \log(1/\epsilon)} \\
&= \sum_{i\in\mathcal{M}} C_g^2 \tau^2 \left(C_g \frac{\tau}{|x_i|}\right)^{C' \log(1/\epsilon)-2} \\
&\leq k \cdot \tau^2 \cdot \frac{\epsilon}{100} \\
&= \frac{\epsilon}{100} \|x_{-k}\|_2^2,
\end{aligned}$$

where the first step follows by the bound on $\mathbb{E}\left[\mathbb{1}_{z_i<\tau}\right] = \Pr\left[\mathbb{1}_{z_i<\tau} = 1\right]$, and the third step by choosing by choosing $C_R > 1$ to be some sufficiently large constant and the facts $C' > C_R/3$ and $(C_g\tau)/|x_i| \leq 1/3$.

□

Lemma 10.3.4 (probability of a fixed coordinate is badly-estimated)**.** *A coordinate i is badly-estimated (as in Definition 10.3.4) probability at most*

$$\frac{\epsilon^3}{100^2 C_L}.$$

Proof. Fix r and set set $b = h_r(i)$. Recall the definition of $y_{r,b} = \sum_{i \in h_r^{-1}(b)} x_i g_{i,r}$ in Definition 10.3.1. We have that

$$\left| |g_{i,r}x_i| - \left| \sum_{j \in h_r^{-1}(b) \setminus \{i\}} g_{j,r}x_j \right| \right| \leq |y_{r,b}| \leq |g_{i,r}x_i| + \left| \sum_{j \in h_r^{-1}(b) \setminus \{i\}} g_{j,r}x_j \right|,$$

Now, $|g_{i,r}x_i|$ will be at in $[(1/3C_g)|x_i|, 2|x_i|]$ with probability at least 0.63 (due to Fact 10.3.2).

Moreover, for any $j \in H(x,k) \setminus \{i\}$, $h_r(j) \neq b$ with probability $1 - 1/B = 1 - \epsilon/(C_B k) \geq 1 - 1/(C_B k)$. By a union bound, we get with probability at least $1 - 1/C_B$, for all $j \in H(x,k) \setminus \{i\}$, $h_r(j) \neq b$. Conditioning on this event, we have,

$$\mathbb{E}\left[\left(\sum_{j \in h_r^{-1}(b) \setminus \{i\}} g_{j,r}x_j \right)^2 \right] = \frac{\epsilon}{C_B k} \|x_{-k}\|_2^2.$$

We then apply Markov's inequality to get that with probability at least $1 - 10^4/C_B$,

$$\left(\sum_{j \in h_r^{-1}(b) \setminus \{i\}} g_{j,r}x_j \right)^2 \leq \frac{\epsilon}{10^4 k} \|x_{-k}\|_2^2.$$

Therefore, by a union bound,

$$\begin{aligned} \Pr\left[|y_{r,b}| \in \left[\frac{1}{3C_g}|x_i| - \frac{1}{100}\frac{\sqrt{\epsilon}}{\sqrt{k}}, 2|x_i| + \frac{1}{100}\frac{\sqrt{\epsilon}}{\sqrt{k}} \right] \right] &\geq 0.63 - \frac{1}{C_B} - \frac{10^4}{C_B} \\ &\geq 0.6, \end{aligned}$$

where the last step follows by $C_B \geq 5 \times 10^5$.

Note that z_i is obtained by taking median of R copies of i.i.d. $|y_{r,b}|$. For each $r \in [R]$, we define $Z_r = 1$ if $|y_{r,b}|$ falls into that region, and 0 otherwise. We have $\mathbb{E}[\sum_{r=1}^{R} Z_r] \geq 0.6R$. Using Chernoff bound, we have

$$\begin{aligned}\Pr\left[\sum_{r=1}^{R} Z_r < 0.9 \cdot 0.6R\right] &\leq \Pr\left[\sum_{r=1}^{R} Z_r < 0.9\,\mathbb{E}[\sum_{r=1}^{R} Z_r]\right] \\ &\leq e^{-\frac{1}{3}0.1^2\,\mathbb{E}[\sum_{r=1}^{R} Z_r]} \\ &\leq e^{-\frac{1}{3}0.1^2 \cdot 0.6R}\end{aligned}$$

Thus,

$$\begin{aligned}\Pr\left[z_i \notin \left[\frac{1}{3C_g}|x_i| - \frac{1}{100}\frac{\sqrt{\epsilon}}{\sqrt{k}}, 2|x_i| + \frac{1}{100}\frac{\sqrt{\epsilon}}{\sqrt{k}}\right]\right] &\leq e^{-\frac{1}{3}0.1^2 \cdot 0.6R} \\ &\leq 2^{-0.002R} \\ &= 2^{-0.002C_R \log(1/\epsilon)} \\ &\leq 2^{-0.002(10000+500C_L)\log(1/\epsilon)} \\ &\leq \frac{\epsilon^3}{100^2 C_L},\end{aligned}$$

where the third step follows from choice of C_R.

□

10.3.3 Guarantees of the algorithm

We now proceed with the proof of Theorem 10.3.1.

Proof. By Lemma 10.3.3 and an application of Markov's inequality we have that

$$\sum_{i \in \mathcal{M}} x_i^2 \cdot \mathbb{1}_{z_i < \tau} \leq \epsilon \|x_{-k}\|_2^2,$$

with probability 99/100. Let this event be $\mathcal{E}_1$.

Moreover, by Lemma 10.3.4,

$$\mathbb{E}[|\mathcal{B}|] \leq \frac{\epsilon^3 |L|}{100^2 C_L},$$

so, by Markov's inequality, we have

$$\Pr\left[|\mathcal{B}| \leq \frac{\epsilon^2 |L|}{100 C_L}\right] \geq 1 - \epsilon/100 \geq 99/100$$

Let this event be $\mathcal{E}_2$.

By taking a union bound, $\mathcal{E}_1$ and $\mathcal{E}_2$ both hold with probability 98/100. Plugging size of $|L|$ ($\leq C_L \cdot k/\epsilon$) into equation of event $\mathcal{E}_2$, we get

$$|\mathcal{B}| \leq \frac{\epsilon^2 |L|}{100 C_L} \leq \frac{\epsilon k}{100}. \tag{10.4}$$

It means there are at most $\epsilon k/100$ coordinates that badly-estimated.

We remind that our goal is to bound

$$\begin{aligned}
\|x - x_S\|_2^2 &= \|x_{\bar{S}}\|_2^2 \\
&= \|x_{\bar{S} \cap \mathcal{M}}\|_2^2 + \|x_{\bar{S} \setminus \mathcal{M}}\|_2^2 \\
&= \|x_{\bar{S} \cap \mathcal{M}}\|_2^2 + \|x_{(\bar{S} \setminus \mathcal{M}) \cap \mathcal{B}}\|_2^2 + \|x_{(\bar{S} \setminus \mathcal{M}) \setminus \mathcal{B}}\|_2^2
\end{aligned}$$

1. Bounding $\|x_{\mathcal{M} \cap \bar{S}}\|_2^2$. Consider the set

$$I = \{i \in L \setminus \mathcal{M} : |x_i| \geq \tau/3\},$$

which contains the coordinates in L with magnitude in the range $[\frac{1}{3}\tau, 3C_g\tau)$. By the definition of τ, clearly, $|I| \leq 3k + k = 4k$, because we can have at most k such elements in $H(x, k)$,

and at most $3k$ such elements in the tail $[n] \setminus H(x,k)$. Since the number of badly estimated coordinates is at most $\epsilon k/100$ and the size of S is βk for sufficiently large β, we can have at most $4k + \epsilon k/100 < \beta k$ coordinates $i \in L$ which are not in $\mathcal{M}$ and are larger than τ. This means that all coordinates in $\mathcal{M}$ with estimate $z_i \geq \tau$ will belong to S. This implies that

$$\mathcal{M} \cap \bar{S} = \{i \in \mathcal{M} : z_i < \tau\},$$

and hence

$$\|x_{\mathcal{M}\cap\bar{S}}\|_2^2 = \sum_{i\in\mathcal{M}} x_i^2 \cdot \mathbb{1}_{z_i<\tau} \leq \epsilon \|x_{-k}\|_2^2,$$

since we conditioned on event $\mathcal{E}_1$.

2. Bounding $\|x_{(\bar{S}\setminus\mathcal{M})\cap\mathcal{B}}\|_2^2$. For every $i \in (\bar{S} \setminus \mathcal{M}) \cap \mathcal{B})$ we have the trivial bound $|x_i| \leq \tau$. Since $(\bar{S} \setminus \mathcal{M}) \cap \mathcal{B}) \subseteq \mathcal{B}$, because the event $\mathcal{E}_2$ we get that

$$\|x_{(\bar{S}\setminus\mathcal{M})\cap\mathcal{B}}\|_2^2 \leq |\mathcal{B}| \cdot \tau^2 \leq \frac{\epsilon k}{100} \cdot \frac{\|x_{-k}\|_2^2}{k} = \frac{\epsilon}{100}\|x_{-k}\|_2^2,$$

where the second step follows from (10.4) and (10.3).

3. Bounding $\|x_{(\bar{S}\setminus\mathcal{M})\setminus\mathcal{B}}\|_2^2$. Observe that set $(\bar{S} \setminus \mathcal{M}) \setminus \mathcal{B}$ consists of well-estimated coordinates that are less than τ in magnitude, and their estimates do not belong to the largest βk estimates. For convenience, set $Q = (\bar{S} \setminus \mathcal{M}) \setminus \mathcal{B}$, then it is obvious that $Q = \overline{S}\setminus(\mathcal{M} \cup \mathcal{B})$.

We define three sets H_1, H_2, H_3 as follows,

$$H_1 = Q \cap T, \quad H_2 = Q \cap \bar{T}, \quad \text{and} \quad H_3 = (\overline{T}\setminus(\mathcal{M} \cup \mathcal{B}))\setminus\overline{S}.$$

Using the definition of Q, we can rewrite H_1, H_2, and H_3 as follows

$$\begin{aligned}
H_1 &= (\overline{S}\backslash(\mathcal{M}\cup\mathcal{B}))\cap T = \overline{S}\cap\overline{\mathcal{M}\cup\mathcal{B}}\cap T,\\
H_2 &= (\overline{S}\backslash(\mathcal{M}\cup\mathcal{B}))\cap \overline{T} = \overline{S}\cap\overline{\mathcal{M}\cup\mathcal{B}}\cap \overline{T},\\
H_3 &= (\overline{T}\backslash(\mathcal{M}\cup\mathcal{B}))\backslash\overline{S} = S\cap\overline{\mathcal{M}\cup\mathcal{B}}\cap \overline{T}.
\end{aligned}$$

We can show that

$$\begin{aligned}
H_2\cap H_3 &= (\overline{S}\cap\overline{\mathcal{M}\cup\mathcal{B}}\cap\overline{T})\cup(S\cap\overline{\mathcal{M}\cup\mathcal{B}}\cap\overline{T})\\
&= \emptyset, \qquad (10.5)
\end{aligned}$$

and

$$\begin{aligned}
H_2\cup H_3 &= (\overline{S}\cap\overline{\mathcal{M}\cup\mathcal{B}}\cap\overline{T})\cap(S\cap\overline{\mathcal{M}\cup\mathcal{B}}\cap\overline{T})\\
&= \overline{\mathcal{M}\cup\mathcal{B}}\cap\overline{T}\\
&= \overline{T}\backslash(\mathcal{M}\cup\mathcal{B}). \qquad (10.6)
\end{aligned}$$

Then,

$$\begin{aligned}
\|x_Q\|_2^2 &= \|x_{H_1}\|_2^2 + \|x_{H_2}\|_2^2\\
&= \|x_{H_1}\|_2^2 + (\|x_{\bar{T}\backslash(\mathcal{M}\cup\mathcal{B})}\|_2^2 - \|x_{H_3}\|_2^2)\\
&\leq \|x_{H_1}\|_2^2 + \|x_{\overline{T}}\|_2^2 - \|x_{H_3}\|_2^2\\
&\leq \|x_{H_1}\|_2^2 + (1+\epsilon)\|x_{-k}\|_2^2 - \|x_{H_3}\|_2^2,
\end{aligned}$$

where first step follows from $H_1\cap H_2=\emptyset$ and $H_1\cup H_2 = Q$, the second step follows from Eq. (10.5) and (10.6), the third step follows from $\|x_{\bar{T}\backslash(\mathcal{M}\cup\mathcal{B})}\|_2^2 \leq \|x_{\overline{T}}\|_2^2$ and the last step follows from $\|x_{\overline{T}}\|_2^2 \leq (1+\epsilon)\|x_{-k}\|_2^2$.

We define d, E, a, b as follows

$$d = |H_1|, \qquad E = \frac{1}{4}\sqrt{\epsilon/k}\|x_{-k}\|_2, \qquad a = \max_{i\in H_1}|x_i|, \qquad b = \min_{i\in H_3}|x_i|. \tag{10.7}$$

Let i^* and j^* be defined as follows:

$$i^* = \arg\max_{i\in H_1}|x_i|, \quad \text{and} \quad j^* = \arg\min_{j\in H_3}|x_j|. \tag{10.8}$$

Recall the definitions of H_1 and H_3, we know H_3 is a subset of S and H_1 is a subset of $\overline{S}$. Since the set S contains the largest βk coordinates, thus we have

$$z_j \geq z_i, \forall i \in H_1, j \in H_3.$$

It further implies $z_{j^*} \geq z_{i^*}$.

By Definition 10.3.4, we have

$$z_{i^*} \geq \frac{1}{3C_g}|x_{i^*}| - \frac{1}{100}\frac{\sqrt{\epsilon}}{\sqrt{k}}\|x_{-k}\|_2, \tag{10.9}$$

and

$$z_{j^*} \leq 2|x_{j^*}| + \frac{1}{100}\frac{\sqrt{\epsilon}}{\sqrt{k}}\|x_{-k}\|_2. \tag{10.10}$$

Then, we can show that $a \leq 6C_g b + E$ in the following sense:

$$\begin{aligned}
a &= |x_{i^*}| && \text{by def. of } i^*, a, (10.8), (10.7)\\
&\leq 3C_g z_{i^*} + \frac{3C_g}{100}\sqrt{\epsilon/k}\|x_{-k}\|_2 && \text{by (10.9)}\\
&\leq 3C_g z_{j^*} + \frac{3C_g}{200}\sqrt{\epsilon/k}\|x_{-k}\|_2 && \text{by } z_{i^*} \leq z_{j^*}\\
&\leq 6C_g|x_{j^*}| + \frac{6C_g}{200}\sqrt{\epsilon/k}\|x_{-k}\|_2 && \text{by (10.10)}\\
&= 6C_g b + \frac{6C_g}{200}\sqrt{\epsilon/k}\|x_{-k}\|_2 && \text{by def. of } j^*, b, (10.8), (10.7)\\
&\leq 6C_g b + E && \text{by def. of } E, (10.7)
\end{aligned}$$

Note that $H_3 = (\bar{T} \setminus (\mathcal{M} \cup \mathcal{B})) \setminus \overline{S} = S \setminus (T \cup \mathcal{M} \cup \mathcal{B})$. Therefore,

$$\begin{aligned} |H_3| &\geq |S| - |T| - |\mathcal{M}| - |\mathcal{B}| \\ &\geq \beta k - k - k - k \\ &= (\beta - 3)k. \end{aligned}$$

Finally, we can have

$$\begin{aligned} \|x_{H_1}\|_2^2 - \|x_{H_3}\|_2^2 &\leq da^2 - (\beta - 3)kb^2 \\ &\leq d(6C_g b + E)^2 - (\beta - 3)kb^2 && \text{by } a \leq 6C_g b + E \\ &= (36C_g^2 d - (\beta - 3)k)b^2 + 12C_g bdE + dE^2 \\ &\leq (36C_g^2 k - (\beta - 3)k)b^2 + 12C_g bkE + kE^2 && \text{by } d \leq k \\ &\leq (36C_g^2 k - (\beta - 5C_g^2)k)b^2 + 12C_g bkE + kE^2 && \text{by } C_g \geq 4/5 \\ &\leq -36kC_g^2 b^2 + 12C_g bkE + kE^2 && \text{by } \beta \geq 77C_g^2 \\ &= -k(6C_g b - E)^2 + 2kE^2 \\ &\leq 2kE^2 \\ &\leq \epsilon \|x_{-k}\|_2^2. \end{aligned}$$

where the last step follows from definition of E.

Thus, we have

$$\|x_Q\|_2^2 \leq (1 + 2\epsilon)\|x_{-k}\|_2^2.$$

Putting it all together. We have

$$\begin{aligned}\|x - x_S\|_2^2 &= \|x_{\overline{S}\cap\mathcal{M}}\|_2^2 + \|x_{(\overline{S}\setminus\mathcal{M})\cap\mathcal{B}}\|_2^2 + \|x_{(\overline{S}\setminus\mathcal{M})\setminus\mathcal{B}}\|_2^2 \\ &\leq \epsilon\|x_{-k}\|_2^2 + \frac{\epsilon}{100}\|x_{-k}\|_2^2 + (1+2\epsilon)\|x_{-k}\|_2^2 \\ &\leq (1+4\epsilon)\|x_{-k}\|_2^2\end{aligned}$$

Finally, we can conclude $\alpha = 5$ and $\beta = 100$.

□

10.3.4 Time, measurements, column sparsity, and probability

In this section, we will bound the decoding time, the number of measurements, column sparsity and success probability of algorithm.

Decoding time. For each $i \in L$, we compute z_i to be the median of R values. For this part, we spend $O(|L| \cdot R) = O((k/\epsilon) \cdot \log(1/\epsilon))$ time. Moreover, calculating the top βk estimates in L only takes $O(|L|)$ time. Therefore, the decoding time is $O((k/\epsilon) \cdot \log(1/\epsilon))$.

The number of measurements. The number of measurements is the bucket size B times the number of repetitions R, which is $O(BR) = O((k/\epsilon) \cdot \log(1/\epsilon))$.

Column sparsity. Each $i \in [n]$ goes to one bucket for each hash function, and we repeat R times, so the column sparsity is $O(R) = O(\log(1/\epsilon))$.

Success probability. By analysis in Section 10.3.3, the success probability is at least 0.98.

10.4 Tail Estimation

In Section 10.4.1, we present a standard result on random walks. In Section 10.4.2, we present some results on p-stable distribution. In what follows we asssume that $0 < p \leq 2$. We show an algorithm for ℓ_p tail estimation in Section 10.4.3.

10.4.1 Random walks

Theorem 10.4.1. *We consider the following random walk. We go right if $B_i = 1$ and we go left if $B_i = 0$. The probability of $B_i = 1$ is at least $9/10$ and the probability of $B_i = 0$ is at most $1/10$. With at least some constant probability bounded away from $\frac{1}{2}$, for all the possible length of the random walk, it will never return to the origin.*

This is a standard claim, that can be proved in numerous ways, such as martingales etc. For the completeness, we still provide a folklore proof here.

Proof. Let $p > 1/2$ be the probability of stepping to the right, and let $q = 1 - p$. For integer $m \geq 1$, let P_m be the probability of first hitting 0 in exactly m steps. It is obvious that $P_m = 0$ if n is even, and $P_1 = q$. In order to hit 0 for the first time on the third step you must Right-Left-Left, so $P_3 = pq^2$. To hit 0 for the first time in exactly $2k + 1$ steps, you must go right k times and left $k + 1$ times, your last step must be to the left, and through the first $2k$ steps you must always have made at least many right steps as left steps. It is well known that the number o such path is C_k, which is the k-th Catalan number. Thus,

$$P_{2k+1} = C_k q^k q^{k+1} = C_k \cdot q(pq)^k = \frac{q(pq)^k}{k+1}\binom{2k}{k},$$

since

$$C_k = \frac{1}{k+1}\binom{2k}{k}$$

By [Wil05], the generating function for the Catalan numbers is

$$c(x) = \sum_{k\geq 0} C_k x^k = \frac{1-\sqrt{1-4x}}{2x},$$

so the probability that the random walk will hit 0 is

$$\begin{aligned}
\sum_{k\geq 0} P_{2k+1} &= q\sum_{k\geq 0} C_k (pq)^k \\
&= q \cdot c(pq) \\
&= q \cdot \frac{1-\sqrt{1-4pq}}{2pq} && \text{by definition of } c(x) \\
&= \frac{1-\sqrt{1-4q(1-q)}}{2p} \\
&= \frac{1-\sqrt{1-4q+4q^2}}{2p} \\
&= \frac{1-(1-2q)}{2p} \\
&= q/p \\
&\leq 1/9.
\end{aligned}$$

Thus, we complete the proof. □

10.4.2 p-stable distributions

We first provide the definition of p-stable distribution. For the more details, we refer the readers to [Ind06].

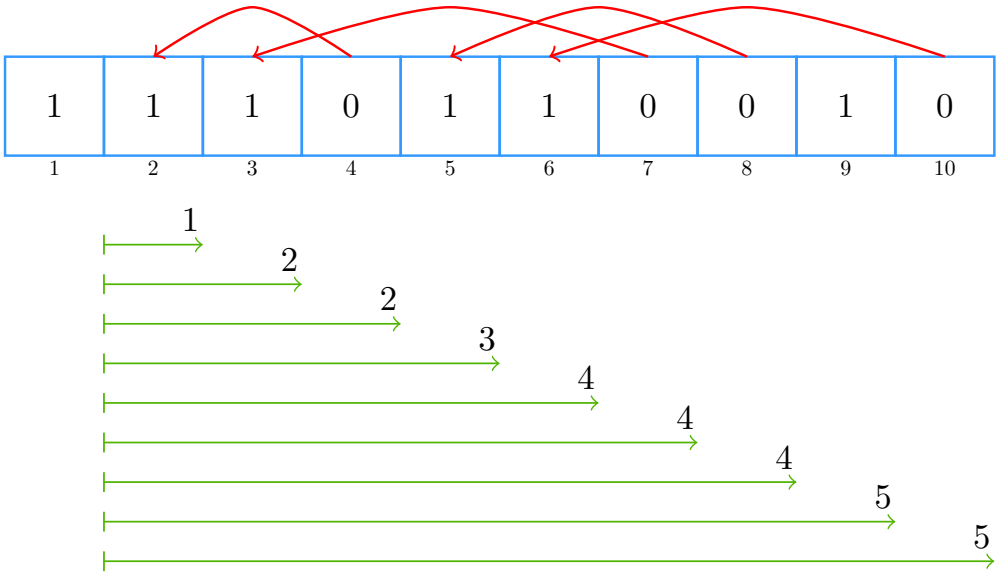


Figure 10.1: This is a visualization of part of the proof in Claim 10.4.7. We consider an example where there are $l = 10$ blocks, $B_1 = 1$, $B_2 = 1$, $B_1 = 1$, $B_3 = 0$, $B_4 = 0$, $B_5 = 1$, $B_6 = 1$, $B_7 = 0$, $B_8 = 0$, $B_9 = 1$ and $B_{10} = 0$. Recall the two important conditions in the proof of Claim 10.4.7, the first one is $B_1 = 1$ and the second one is, for all $j \in [l], \sum_{j'=2}^{j} B_{j'} > (j-1)/2$. The number on the green arrow is $\sum_{j'=2}^{j} B_{j'}$. It is to see that the example we provided here is satisfying those two conditions. Recall the definition of set S_1 and S_0. Here $S_1 = \{2, 3, 5, 6, 9\}$ and $S_0 = \{4, 7, 8, 10\}$. Then $S_1' = \{2, 3, 5, 6\}$. The mapping π satisfies that $\pi(4) = 2$, $\pi(7) = 3$, $\pi(8) = 5$ and $\pi(10) = 6$.

Definition 10.4.1 (p-stable distribution)**.** A distribution $\mathcal{D}$ over $\mathbb{R}$ is called p-stable, if there exists $p \geq 0$ such that for any n real numbers $a_1, a_2, \cdots, a_n$ and i.i.d. variables $x_1, x_2, \cdots, x_n$ from distribution $\mathcal{D}$, the random variable $\sum_{i=1}^{n} a_i x_i$ has the same distribution as the variable $\|a\|_p y$, where y is a random variable from distribution $\mathcal{D}$.

Theorem 10.4.2 ([Zol86])**.** *For any $p \in (0, 2]$, there exists a p-stable distribution.*

Gaussian distribution defined by the density function $f(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$, is 2-stable. Cauchy distribution defined by density function $f(x) = \frac{1}{\pi} \frac{1}{1+x^2}$ is 1-stable. Let $\mathcal{D}_p$ denote the

p-stable distribution. For $p = 2$, $\mathcal{D}_p$ is $\mathcal{N}(0, 1)$ and for $p = 1$, $\mathcal{D}_p$ is $\mathcal{C}(0, 1)$.

10.4.3 ℓ_p-tail estimation algorithm

The goal of this Section is prove Lemma 10.4.3.

Algorithm 10.3 ℓ_p-tail Estimation Algorithm

procedure LpLpTailEstimation(x, k, p, C_0, δ) ▷ Lemma 10.4.3
▷ Requires $C_0 \geq 1000$
  $m \leftarrow O(\log(1/\delta))$
  Choose $g_{i,t}$ to be random variable that sampled i.i.d. from distribution $\mathcal{D}_p$, $\forall i, t \in [n] \times [m]$
  Choose $\delta_{i,t}$ to be Bernoulli random variable with $\mathbb{E}[\delta_{i,t}] = 1/(100k)$, $\forall i, t \in [n] \times [m]$
  ▷ Matrix A is implicitly constructed based on $g_{i,t}$ and $\delta_{i,t}$
  for $t \in [m]$ **do**
    $y_t \leftarrow \sum_{i=1}^{n} \delta_{i,t} \cdot g_{i,t} \cdot x_i$
  end for
  $V \leftarrow \text{median}_{t \in [m]} |y_t|^2$
  return V ▷ $\frac{1}{10k}\|x_{-C_0 k}\|_p^p \leq V \leq \frac{1}{k}\|x_{-k}\|_p^p$
end procedure

One can try to prove such a claim for $p = 2$ with random signs, instead of Gaussians, by applying the Paley-Zygmund inequality to obtain the lower bound. A straightforward calculation indicates that this approach does not give the desired result, hence we need a new argument to deal with the lower bound.

Lemma 10.4.3 (Restatement of Lemma 10.1.3)**.** *Let $C_0 \geq 1000$ denote some fixed constant. There is an oblivious construction of matrix $A \in \mathbb{R}^{m \times n}$ with $m = O(\log(1/\delta))$ along with a decoding procedure* LpLpTailEstimation(x, k, p, C_0, δ) *(Algorithm 10.3) such that, given Ax, it is possible to output a value V in time $O(m)$ such that*

$$\frac{1}{10k}\|x_{-C_0 k}\|_p^p \leq V \leq \frac{1}{k}\|x_{-k}\|_p^p,$$

holds with probability $1-\delta$.

Proof. Let $m = O(\log(1/\delta))$. For each $i \in [n], t \in [m]$, we use $g_{i,t}$ to denote a random variable that sample from distribution $\mathcal{D}_p$.

For each $i \in [n], t \in [m]$, we use $\delta_{i,t}$ to denote a Bernoulli random variable such that

$$\delta_{i,t} = \begin{cases} 1, & \text{with prob. } \frac{1}{100k}; \\ 0, & \text{otherwise.} \end{cases}$$

Then we have

$$\mathbb{E}[\delta_{i,t}] = \frac{1}{100k}.$$

For each $t \in [m]$, we define y_t as follows

$$y_t = \sum_{i=1}^{n} \delta_{i,t} g_{i,t} x_i. \tag{10.11}$$

For each $t \in [m]$, we define Δ_t as follows

$$\Delta_t = \left(\sum_{i=1}^{n} \delta_{i,t}^p x_i^p \right)^{1/p}. \tag{10.12}$$

Using Claim 10.4.4 and Claim 10.4.7

$$\Pr_{g,\delta} \left[|y_t| < \alpha \frac{1}{(2C_0 k)^{1/p}} \|x_{-C_0 k}\|_p \right] \leq 1/5.$$

Using Claim 10.4.5 and Claim 10.4.6

$$\Pr_{g,\delta} \left[|y_t| > \beta \frac{1}{k^{1/p}} \|x_{-k}\|_p \right] \leq 1/5.$$

Finally, we just take the median over m different independent repeats. Since $m = O(\log(1/\delta))$, thus, we can boost the failure probability to δ.

□

It is a standard fact, due to p-stability, that y_t follows the p-stable distribution : $\Delta_t \cdot \mathcal{D}_p$. Since p-stable distributions are continuous functions, we have the following two Claims:

Claim 10.4.4 (upper bound on $|y_t|$)**.** *Let y_t be defined in Eq. (10.11), let Δ_t be defined in Eq. (10.12). There is some sufficiently small constant $\alpha \in (0,1)$ such that*

$$\Pr_g[|y_t| < \alpha \cdot \Delta_t] \leq 1/10.$$

Claim 10.4.5 (lower bound on $|y_t|$)**.** *Let y_t be defined in Eq. (10.11), let Δ_t be defined in Eq. (10.12). There is some sufficiently large constant $\beta > 1$ such that*

$$\Pr_g[|y_t| > \beta \cdot \Delta_t] \leq 1/10.$$

It remains to prove Claim 10.4.6 and Claim 10.4.7.

Claim 10.4.6 (lower bound on Δ_t)**.** *Let Δ_t be defined in Eq. (10.12). Then we have*

$$\Pr_\delta \left[\Delta_t > \frac{1}{k^{1/p}} \|x_{-k}\|_p \right] \leq 1/10.$$

Proof. The proof mainly includes three steps,

First, for a fixed coordinate $i \in [n]$, with probability at most $1/(100k)$, it got sampled. Taking a union bound over all k largest coordinates. We can show that with probability at least $1 - 1/100$, none of k largest coordinates is sampled. Let ξ be that event.

Second, conditioning on event ξ, we can show that

$$\mathbb{E}[\Delta_t] \leq \frac{1}{(100k)^{1/p}} \|x_{-k}\|_p.$$

Third, applying Markov's inequality, we have

$$\Pr[\Delta_t \geq a] \leq \mathbb{E}[\Delta_t]/a.$$

Choosing $a = \frac{1}{k^{1/p}}\|x_{-k}\|_p$, we have

$$\Pr\left[\Delta_t \geq \frac{1}{k^{1/p}}\|x_{-k}\|_p\right] \leq 1/10.$$

□

Claim 10.4.7 (upper bound on Δ_t)**.** *Let Δ_t be defined in Eq. (10.12). For any $C_0 \geq 1000$, we have*

$$\Pr_{\delta}\left[\Delta_t < \frac{1}{(2C_0k)^{1/p}}\|x_{-C_0k}\|_p\right] \leq 1/10.$$

Proof. Without loss of generality, we can assume that all coordinates of x_i are sorted, i.e. $x_1 \geq x_2 \geq \cdots \geq x_n$. Then we split length n vector into l blocks where each block has length $s = C_0k$. Note that it is obvious $l \cdot s = n$.

For each $j \in [l]$, we use boolean variable B_j to denote that if at least one coordinate in j-th block has been sampled. For a fixed block $j \in [l]$, the probability of sampling at least one coordinate from that block is at least

$$1 - \left(1 - \frac{1}{100k}\right)^s = 1 - \left(1 - \frac{1}{100k}\right)^{C_0k} \geq 9/10.$$

Thus, we know $1 \geq \mathbb{E}[B_j] \geq 9/10$.

Warm-up. Note the probability is not allowed to take a union over all the blocks. However, if we conditioned on that each block has been sampled at least one coordinate,

then we have

$$\begin{aligned}\Delta_t^p &= \sum_{i=1}^{n} \delta_{i,t} x_i^p \\ &\geq \sum_{j=1}^{l-1} x_{js}^p \\ &\geq \sum_{j=1}^{l-1} \frac{1}{s} \left(x_{js+1}^p + x_{js+2}^p + \cdots + x_{js+s}^p \right) \\ &= \frac{1}{s} \|x_s\|_p^p.\end{aligned}$$

Fixed. For simplicity, for each $j \in [l]$, we use set T_j to denote $\{(j-1)s+1, (j-1)s+2, \cdots, (j-1)s+s\}$.

Using random walk Lemma 10.4.1, with probability at least 99/100, we have : for all $j \in \{2, \cdots, l\}$,

$$\sum_{j'=2}^{j} B_{j'} > (j-1)/2.$$

We know that with probability at least 99/100, $B_1 = 1$. Then with probability at least 99/100, we have

$$B_1 = 1, \text{ and } \sum_{j'=2}^{j} B_{j'} > (j-1)/2, \forall j \in [l].$$

We conditioned on the above event holds. Let set $S_1 \subset [n]$ denote the set of indices j such that $B_j = 1$, i.e.,

$$S_1 = \left\{ j \;\middle|\; B_j = 1, j \in [n] \backslash \{1\} \right\}.$$

Let set $S_0 \subset [n]$ denote the set of indices j such that $B_j = 0$, i.e.,

$$S_0 = \left\{ j \;\middle|\; B_j = 0, j \in [n] \backslash \{1\} \right\}.$$

Due to $\sum_{j'=2}^{j} B_{j'} > (j-1)/2, \forall j \in [l]$, then it is easy to see $S_1 > S_0$ and there exists a one-to-one mapping $\pi : S_0 \to S_1'$ where $S_1' \subseteq S_1$ such that for each coordinate $j \in S_0$, $\pi(j) < j$. Since we are the coordinates are being sorted already, thus

$$\begin{aligned} \sum_{j \in S_1} \|x_{T_j}\|_p^p &= \sum_{j \in S_1'} \|x_{T_j}\|_p^p \\ &= \sum_{j \in S_1'} \|x_{T_{\pi^{-1}(j)}}\|_p^p \\ &\geq \sum_{j \in S_0} \|x_{T_j}\|_p^p \end{aligned}$$

which implies that

$$\Delta_t^p = \sum_{i=1}^{n} \delta_i^p x_i^p = \sum_{j \in S_1} \|x_{T_j}\|_p^p \geq \frac{1}{2s} \|x_{-s}\|_p^p.$$

Thus, with probability at least $9/10$, we have

$$\Delta_t \geq \frac{1}{(2s)^{1/p}} \|x_{-s}\|_p.$$

□

10.5 Putting It All Together

Our full algorithm first applies interval forest sparse recovery algorithm in Algorithm 10.1 with precision parameter $\frac{\epsilon}{10}$ to obtain a set L of size $O(k/\epsilon)$. By Theorem 10.2.1, with probability $\frac{9}{10}$, L contains a subset T of size at most k so that $\|x-x_T\|_2 \leq (1+\frac{\epsilon}{10})\|x_{-k}\|_2$. Then the algorithm feeds L into the pruning procedure in Algorithm 10.2 also with precision parameter $\frac{\epsilon}{10}$ to get S. By Theorem 10.3.1, with probability $\frac{9}{10}$, $|S| = O(k)$ and $\|x - x_S\|_2 \leq (1+\frac{\epsilon}{2})\|x-x_{-k}\|_2$. Finally, the algorithm uses set query data structure in Lemma 10.1.6 to give $\widehat{x}_S$ as approximation to x_S. With probability at least $1-1/\operatorname{poly}(k)$, $\|\widehat{x}_S-x_S\|_2^2 \leq \frac{\epsilon}{2}\|x-x_S\|_2^2$. Therefore, $\|x-\widehat{x}_S\|_2^2 = \|x-x_S\|_2^2+\|x_S-\widehat{x}_S\|_2^2 \leq (1+\frac{\epsilon}{4})(1+\frac{\epsilon}{2})^2\|x-x_{-k}\|_2^2 \leq (1+\epsilon)^2\|x-x_{-k}\|_2^2$. By union bound, the overall failure probability is at most $1/4$.

Algorithm 10.4 Stronger ℓ_2/ℓ_2 Algorithm

```
1: procedure Main(x, n, k, ε)                                  ▷ Theorem 10.1.2
2:     L ← IntervalForestSparseRecovery(x, n, k, ε/10)         ▷ Algorithm 10.1
3:     S ← Prune(x, n, k, ε/10, L)                             ▷ Algorithm 10.2
4:     x̂_S ← SetQuery(x, n, ε/4, S)                            ▷ Lemma 10.1.6
5:     return x̂_S
6: end procedure
```

Chapter 11

Continuous Fourier Transform I

In recent years, a number of works have studied methods for computing the Fourier transform in sublinear time if the output is sparse. Most of these have focused on the discrete setting, even though in many applications the input signal is continuous and naive discretization significantly worsens the sparsity level.

We present an algorithm for robustly computing sparse Fourier transforms in the continuous setting. Let $x(t) = x^*(t) + g(t)$, where x^* has a k-sparse Fourier transform and g is an arbitrary noise term. Given sample access to $x(t)$ for some duration T, we show how to find a k-Fourier-sparse reconstruction $x'(t)$ with

$$\frac{1}{T}\int_0^T |x'(t) - x(t)|^2 \mathrm{d}t \lesssim \frac{1}{T}\int_0^T |g(t)|^2 \mathrm{d}t.$$

The sample complexity is linear in k and logarithmic in the signal-to-noise ratio and the frequency resolution. Previous results with similar sample complexities could not tolerate an infinitesimal amount of i.i.d. Gaussian noise, and even algorithms with higher sample complexities increased the noise by a polynomial factor. We also give new results for how precisely the individual frequencies of x^* can be recovered.

11.1 Introduction

The Fourier transform is ubiquitous in digital signal processing of a diverse set of signals, including sound, image, and video. Much of this is enabled by the Fast Fourier Transform (FFT) [CT65], which computes the n-point discrete Fourier transform in $O(n \log n)$ time. But can we do better?

In many situations, much of the reason for using Fourier transforms is because the transformed signal is *sparse*—i.e., the energy is concentrated in a small set of k locations. In such situations, one could hope for a dependency that depends nearly linearly on k rather than n. Moreover, one may be able to find these frequencies while only sampling the signal for some period of time. This idea has led to a number of results on *sparse Fourier transforms*, including [GGI+02, GMS05, HIKP12a, IK14], that can achieve $O(k \log(n/k) \log n)$ running time and $O(k \log(n/k))$ sample complexity (although not quite both at the same time) in a robust setting.

These works apply to the discrete Fourier transform, but lots of signals including audio or radio originally come from a continuous domain. The standard way to convert a continuous Fourier transform into a discrete one is to apply a window function then subsample. Unfortunately, doing so "smears out" the frequencies, blowing up the sparsity. Thus, one can hope for significant efficiency gains by directly solving the sparse Fourier transform problem in the continuous setting. This has led researchers to adapt techniques from the discrete setting to the continuous both in theory [BCG+12, TBSR13, CF14, DB13] and in practice [SAH+13]. However, these results are not robust to noise: if the signal is sampled with a tiny amount of Gaussian noise or decays very slightly over time, no method has been known for computing a sparse Fourier transform in the continuous setting with sample

complexity linear in k and logarithmic in other factors. That is what we present in this paper.

Formally, a vector $x^*(t)$ has a k-sparse Fourier transform if it can be written as

$$x^*(t) = \sum_{i=1}^{k} v_i e^{2\pi \mathbf{i} f_i t}$$

for some *tones* $\{(v_i, f_i)\}$. We consider the problem where we can sample some signal

$$x(t) = x^*(t) + g(t)$$

at any t we choose in some interval $[0, T]$, where $x^*(t)$ has a k-sparse Fourier transform and $g(t)$ is arbitrary noise. As long as g is "small enough," one would like to recover a good approximation to x (or to x^*, or to $\{(v_i, f_i)\}$) using relatively few samples $t \in [0, T]$ and fast running time. Our algorithm achieves several results of this form, but a simple one is an ℓ_2/ℓ_2 guarantee: we reconstruct an $x'(t)$ with k-sparse Fourier transform such that

$$\frac{1}{T}\int_0^T |x'(t) - x(t)|^2 \mathrm{d}t \lesssim \frac{1}{T}\int_0^T |g(t)|^2 \mathrm{d}t$$

using a number of samples that is k times logarithmic factors[1]. To the best of our knowledge, this is the first algorithm achieving such a constant factor approximation with a sample complexity sublinear in T and the signal-to-noise ratio.

Our algorithm also gives fairly precise estimates of the individual tones (v_i, f_i) of the signal x^*. To demonstrate what factors are important, it is helpful to think about a concrete setting. Let us consider sound from a simplified model of a piano.

[1]We use $f \lesssim g$ to denote that $f \leq Cg$ for some universal constant C.

Thought experiment: piano tuning In a simplified model of a piano, we have keys corresponding to frequencies over some range $[-F, F]$. The noise $g(t)$ comes from ambient noise and the signals not being pure tones (because, for example, the notes might decay slowly over time). For concrete numbers, a modern piano has 88 keys spaced from about 27.5 Hz to $F = 4200$Hz. The space between keys ranges from a few Hz to a few hundred Hz, but most chords will have an $\eta = 30$Hz or more gap between the frequencies being played. One typically would like to tune the keys to within about $\pm\nu = 1$Hz. And piano music typically has k around 5.

Now, suppose you would like to build a piano tuner that can listen to a chord and tell you what notes are played and how they are tuned. For such a system, how long must we wait for the tuner to identify the frequencies? How many samples must the tuner take? And how robust is it to the noise?

If you have a constant signal-to-noise ratio, you need to sample for a time T of at least order $1/\nu = 1$ second in order to get 1Hz precision—frequencies within 1Hz of each other will behave very similarly over small fractions of a second, which noise can make indistinguishable. You also need at least $\Omega(k \log \frac{F}{k\nu}) \approx 50$ samples, because the support of the signal contains that many bits of information and you only get a constant number per measurement (at constant SNR). At higher signal-to-noise ratios ρ, these results extend to $\Omega(\frac{1}{\nu\rho})$ duration and $\Omega(k \log_\rho \frac{F}{k\nu})$ samples. But as the signal-to-noise ratio gets very high, there is another constraint on the duration: for $T < \frac{1}{\eta} \approx 33$ milliseconds the different frequencies start becoming hard to distinguish, which causes the robustness to degrade exponentially in k [Moi15] (though the lower bound there only directly applies to a somewhat restricted version of our setting).

This suggests the form of a result: with a duration $T > \frac{1}{\eta}$, one can hope to recover the frequencies to within $\frac{1}{\rho T}$ using $O(k \log_\rho \frac{FT}{k})$ samples. We give an algorithm that is within logarithmic factors of this ideal: with a duration $T > \frac{O(\log(k/\delta))}{\eta}$, we recover the frequencies to within $O(\frac{1}{\rho T})$ using $O(k \log_\rho(FT) \cdot \log(k/\delta) \log k)$ samples, where ρ and $1/\delta$ are (roughly speaking) the minimum and maximum signal-to-noise ratios that you can tolerate, respectively.

Instead of trying to tune the piano by recovering the frequencies precisely, one may simply wish to record the sound for future playback with relatively few samples. Our algorithm works for this as well: the combination $x'(t)$ of our recovered frequencies satisfies

$$\frac{1}{T}\int_0^T |x'(t) - x(t)|^2 \mathrm{d}t \lesssim \frac{1}{T}\int_0^T |g(t)|^2 \mathrm{d}t.$$

Let us now state our main theorems. The first shows how well we can estimate the frequencies f_i and their weights v_i; we refer to this (v_i, f_i) pair as a *tone*.

Theorem 11.1.1 (Tone estimation). Consider any signal $x(t) : [0, T] \to \mathbb{C}$ of the form

$$x(t) = x^*(t) + g(t),$$

for arbitrary "noise" $g(t)$ and an exactly k-sparse $x^* = \sum_{i \in [k]} v_i e^{2\pi \mathbf{i} f_i t}$ with frequencies $f_i \in [-F, F]$ and frequency separation $\eta = \min_{i \neq j} |f_i - f_j|$. For some parameter $\delta > 0$, define the "noise level"

$$\mathcal{N}^2 := \frac{1}{T}\int_0^T |g(t)|^2 \mathrm{d}t + \delta \sum_{i=1}^k |v_i|^2.$$

We give an algorithm that takes samples from $x(t)$ over any duration $T > O(\frac{\log(k/\delta)}{\eta})$ and returns a set of k tones $\{(v_i', f_i')\}$ that approximates x^* with error proportional to $\mathcal{N}$. In

particular, every large tone is recovered: for any v_i with $|v_i| \gtrsim \mathcal{N}$, we have for an appropriate permutation of the indices that

$$|f_i' - f_i| \lesssim \frac{\mathcal{N}}{T|v_i|} \qquad \text{and} \qquad |v_i' - v_i| \lesssim \mathcal{N}. \tag{11.1}$$

In fact, we satisfy a stronger guarantee that the *total* error is bounded:

$$\sum_{i=1}^{k} \frac{1}{T} \int_0^T |v_i' e^{2\pi \mathbf{i} f_i' t} - v_i e^{2\pi \mathbf{i} f_i t}|^2 \mathrm{d}t \lesssim \mathcal{N}^2. \tag{11.2}$$

The algorithm takes $O(k \log(FT) \log(\frac{k}{\delta}) \log(k))$ samples and $O(k \log(FT) \log(\frac{FT}{\delta}) \log(k))$ running time, and succeeds with probability at least $1 - 1/k^c$ for an arbitrarily large constant c.

We then show that the above approximation of the individual tones is good enough to estimate the overall signal $x(t)$ to within constant factors:

Theorem 11.1.2 (Signal estimation). In the same setting as Theorem 11.1.1, if the duration is slightly longer at $T > O(\frac{\log(1/\delta) + \log^2 k}{\eta})$, the reconstructed signal $x'(t) = \sum_{i=1}^{k} v_i' e^{2\pi \mathbf{i} f_i' t}$ achieves a constant factor approximation to the complete signal x:

$$\frac{1}{T} \int_0^T |x'(t) - x(t)|^2 \mathrm{d}t \lesssim \mathcal{N}^2. \tag{11.3}$$

The algorithm takes $O(k \log(FT) \log(\frac{k}{\delta}) \log(k))$ samples and $O(k \log(FT) \log(\frac{FT}{\delta}) \log(k))$ running time, and succeeds with probability at least $1 - 1/k^c$ for an arbitrarily large constant c.

The above theorems give three different error guarantees, which are all in terms of a "noise level" $\mathcal{N}^2$ that is the variance of the noise $g(t)$ plus δ times the energy of the signal. The algorithm depends logarithmically on δ, so one should think of $\mathcal{N}^2$ as being the variance of the noise, e.g. σ^2 if samples have error $N(0, \sigma^2)$.

Error guarantees. Our algorithm does a good job of estimating the signal, but how exactly should we quantify this? Because very few previous results have shown robust recovery in the continuous setting, there is no standard error measure to use. We therefore bound the error in three different ways: the maximum error in the estimation of any tone; the weighted total error in the estimation of all tones; and the difference between the reconstructed signal and the true signal over the sampled interval. The first measure has been studied before, while the other two are to the best of our knowledge new but useful to fully explain the robustness we achieve.

The error guarantee (11.1) says that we achieve good recovery of any tones with magnitude larger than $C\mathcal{N}$ for some constant C. Note that such a requirement is necessary: for tones with $|v_i| \leq \mathcal{N}$, one could have $g(t) = -v_i e^{2\pi \mathbf{i} f_i t}$, completely removing the tone (v_i, f_i) from the observed $x(t)$ and making it impossible to find. For the tones of sufficiently large magnitude, we find them to within $\frac{\mathcal{N}}{T|v_i|}$. This is always less than $1/T$, and converges to 0 as the noise level decreases. This is known as *superresolution*–one can achieve very high frequency resolution in sparse, nearly noiseless settings. Moreover, by Lemma 11.3.15 our "superresolution" precision $|f_i' - f_i| \lesssim \frac{\mathcal{N}}{T|v_i|}$ is optimal.

While the guarantee of (11.1) is simple and optimal given its form, it is somewhat unsatisfying. It shows that the maximum error over all k tones is $\mathcal{N}$, while one can hope to bound the *total* error over all k tones by $\mathcal{N}$. This is precisely what Equation (11.2) does. The guarantee (11.1) is the precision necessary to recover the tone to within $O(\mathcal{N})$ average error in time, that is (11.1) is equivalent to

$$\frac{1}{T}\int_0^T |v_i' e^{2\pi \mathbf{i} f_i' t} - v_i e^{2\pi \mathbf{i} f_i t}|^2 \mathrm{d}t \lesssim \mathcal{N}^2 \qquad \forall i \in [k].$$

In (11.2), we show that this bound holds even if we sum the left hand side over all $i \in [k]$, so the *average* error is a factor k better than would be implied by (11.1). It also means that the total mass of all tones that are not recovered to within $\pm 1/T$ is $O(\mathcal{N})$, not just that every tone larger than $O(\mathcal{N})$ is recovered to within $\pm 1/T$.

The stronger bound (11.2) can be converted to the guarantee (11.3), which is analogous to the ℓ_2/ℓ_2 recovery guarantee standard in compressive sensing. It bounds the error of our estimate in terms of the input noise, on average over the sampled duration. The standard form of the ℓ_2/ℓ_2 guarantee in the discrete sparse Fourier transform setting [GGI+02, GMS05, HIKP12a, IKP14, IK14] compares the energy in frequency domain rather than time domain. This cannot be achieved directly in the continuous setting, since frequencies infinitesimally close to each other are indistinguishable over a bounded time interval T. But if the signal is periodic with period T (the case where a discrete sparse Fourier transform applies directly), then (11.3) is equivalent to the standard guarantee by Parseval's identity. So (11.3) seems like the right generalization of ℓ_2/ℓ_2 recovery to our setting.

Other factors. Our algorithm succeeds with high probability in k, which could of course be amplified by repetition (but increasing the sample complexity and running time). Our running time, at $O(k \log(FT) \log(FT/\delta) \log k)$, is (after translating between the discrete and continuous setting) basically a $\log k$ factor larger than the fastest known algorithm for discrete sparse Fourier transforms ([HIKP12a]). But since that result only succeeds with constant probability, and repetition would also increase that by a $\log k$ factor, our running time is actually equivalent to the fastest known results that succeed with high probability in k.

Our sample complexity is $O(\log(k/\delta)\log k)$ worse than the presumptive optimal, which is known to be achievable in the discrete setting ([IK14]). However, the techniques that [IK14] uses to avoid the $\log(k/\delta)$ factor seem hard to adapt to the continuous setting without losing some of the robustness/precision guarantees.

Another useful property of our method, not explicitly stated in the theorem, is that the sampling method is relatively simple: it chooses $O(\log(FT)\log k)$ different random arithmetic sequences of points, where each sequence has $O(k\log(k/\delta))$ points spread out over a constant fraction of the time domain T. Thus one could implement this in hardware using a relatively small number of samplers, each of which performs regular sampling at a rate of $\frac{k\log(k/\delta)}{T}$. This is in contrast to the Nyquist rate for non-sparse signals of $2F$ — in the piano example, each sampler has a rate on the order of 50Hz rather than 8000Hz.

The superresolution frequency is optimal because two signals of magnitude v_i and frequency separation $\nu < 1/T$ will differ by $O(\nu^2T^2|v_i|^2)$ over the duration T, so for ν below our threshold the difference is just $\mathcal{N}^2$. Hence if the observed signal $x(t)$ looks identical to (v_i, f_i), it might actually be (v_i, f_i') with noise equaling the difference between the two.

The only previous result known in the form of (11.1) was [Moi15], which lost a $\text{poly}(k, \frac{\max|v_i|}{\min|v_i|}, \eta)$ factor in noise tolerance and also did not optimize for sample complexity.

11.1.1 Comparison to Naive Methods

This section compares our result to some naive ways one could try to solve this problem by applying algorithms not designed for a continuous, sparse Fourier transform setting. The next section will compare our result to algorithms that are designed for such a setting.

Nyquist Sampling. The traditional theory of band-limited signals from discrete samples says that, from samples taken at the Nyquist rate $2F$, one can reconstruct an F-band-limited signal exactly. The Whittaker-Shannon interpolation formula then says that

$$x^*(t) = \sum_{i=-\infty}^{\infty} x^*(2Fi)\,\mathrm{sinc}(2Ft - i) \tag{11.4}$$

where $\mathrm{sinc}(t)$ is the normalized sinc function $\frac{\sin(\pi t)}{\pi t}$. This is for the band-limited "pure" signal x^*, but one could then get a relationship for samples of the actual signal $x(t)$. This has no direct implications for learning the tones (e.g. our (11.1) or (11.2)), but for learning the signal (our (11.3)) there is also an issue. Even in the absence of noise and for $k = 1$, this method will have error polynomially rather than exponentially small in the number of samples.

That is, if there is no noise the method has zero error given infinitely many samples. But we only receive samples over the interval $[0, T]$, leading to error. Consider the trivial setting of $x(t) = 1$. The partial sum of (11.4) at a given t will be missing terms for $i > 2Ft$ and $i < 0$, which (for a random t in $[0, T/2]$) have magnitude at most $1/(Ft)$. The terms alternate in sign, so the sum has error approximately $1/(Ft)^2$. This means that the error over the first $1/F$ time is a constant, leading to average error of $\frac{1}{FT}$. This is with an algorithm that uses FT samples and time. By contrast, our algorithm in the noiseless setting has error exponentially small in the samples and running time.

Discrete Sparse Fourier Transforms. An option related to the previous would be to discretize very finely, then apply a discrete sparse Fourier transform algorithm to keep the sample complexity and runtime small. The trouble here is that sparse Fourier transforms

require sparsity, and this process decreases the sparsity. In particular, this process supposes that the signal is periodic with period T, so one can analyze this process as first converting the signal to one, equivalent over $[0, T]$, but with frequency spectrum only containing integer multiples of $1/T$. This is done by convolving each frequency f_i with a sinc function (corresponding to windowing to $[0, T]$) then restricting to multiples of $1/T$ (corresponding to aliasing). The result is that a one-sparse signal $e^{2\pi \mathbf{i} f_i t}$ is viewed as having Fourier spectrum

$$\widehat{x''}[j] = \operatorname{sinc}(f_i T - j)$$

for $j \in \mathbb{Z}$. When f_i is not a multiple of $1/T$, this means the signal is not a perfectly sparse signal. And this is true regardless of the discretization level, which only affects the subsequent error from aliasing $j \in \mathbb{Z}$ down to $\mathbb{Z}_n$. To have error proportional to $\delta \left\| \widehat{x}^* \right\|_2$, one would need to run such methods for a sparsity level of k/δ. Thus, as with Nyquist sampling, the sample and runtime will be polynomial, rather than logarithmic, in δ.

The above discussion refers to methods for learning the signal (our (11.3)). In terms of learning the tones, one could run the algorithm for sparsity $O(k)$ so that δ is a small constant, which would let one learn roughly where the peaks are and get most of the frequencies to the nearest $1/T$. This would give a similar bound to our (11.1), but without the superresolution effect as the noise becomes small. On the plus side, the duration could be just $O(1/\eta)$—which is sufficient for the different peaks to be distinguishable—rather than $O(\frac{\log k}{\eta})$ as our method would require, and the time and sample complexities could save a $\log k$ factor (if one did not want to recover *all* the tones, just most of them).

Essentially, this algorithm tries to round each frequency to the nearest multiple of $1/T$, which introduces noise that is a constant fraction of the signal. If the signal-to-noise

ratio is already low, this does not increase the noise level by that much so such an algorithm will work reasonably well. If the signal-to-noise ratio is fairly high, however, then the added noise leads to much worse performance. Getting a constant factor approximation to the whole signal is only nontrivial for high SNR, so such a method does badly in that setting. For approximating the tones, it is comparable to our method in the low SNR setting but does not improve much as SNR increases.

11.1.2 Previous Work In Similar Settings

There have been several works that recover continuous frequencies from samples in the time domain. Some of these are in our setting where the samples can be taken at arbitrary positions over $[0, T]$ and others are in the discrete-time (DTFT) setting where the samples must be taken at multiples of the Nyquist frequency $\frac{1}{2F}$.

The results of [TBSR13, CF14, YX15, DB13] show that a convex program can solve the problem in the DTFT setting using $O(k \log k \log(FT))$ samples if the duration is $T > O(\frac{1}{\eta})$, in the setting where $g(t) = 0$ and the coefficients of x^* have random phases. The sample complexity can be one log factor better than ours, which one would expect for the noiseless setting. None of these results show robustness to noise, and some additionally require a running time polynomial in FT rather than k.

The result of [BCG+12] is in a similar setting to our paper, using techniques of the same lineage. It achieves very similar sample complexity and running time to our algorithm, and a guarantee similar in spirit to (11.1) with some notion of robustness. However, the robustness is weaker than ours in significant ways. They consider the noise $g(t)$ in frequency space (i.e. $\widehat{g}(f)$), then require that $\widehat{g}(f)$ is zero at any frequency within η of the signal

frequencies f_i, and bound the error in terms of $\mathcal{N}' = \|\widehat{g}\|_1 / k$ instead of $\|g\|_2$. This fails to cover simple examples of noise, including i.i.d. Gaussian noise $g(t) \sim N(0, \sigma^2)$ and the noise one would get from slow decay of the signal over time (e.g. $x(t) = x^*(t)e^{-\frac{t}{100T}}$.). Both types of noise violate both assumptions on the noise: $\widehat{g}(f)$ will be nonzero arbitrarily close to each f_i and $\|\widehat{g}\|_1$ will be unbounded. Their result also requires a longer duration than our algorithm and has worse precision for any fixed duration.

The result of [Moi15] studies noise tolerance in the DTFT setting, ignoring sample complexity and running time. It shows that the matrix pencil method [HS90], using FT samples, achieves a guarantee of the form (11.1), except that the bounds are an additional $\text{poly}(FT, k, \delta)$ factor larger. Furthermore, it shows a sharp characterization of the minimal T for which this is possible by any algorithm: $T = (1 \pm o(1))\frac{2}{\eta}$ is necessary and sufficient. It is an interesting question whether the lower bound generalizes to our non-DTFT setting, where the samples are not necessarily taken from an even grid.

Lastly, [SAH+13] tries to apply sparse Fourier transforms to a domain with continuous signals. They first apply a discrete sparse Fourier transform then use hill-climbing to optimize their solution into a decent set of continuous frequencies. They have interesting empirical results but no theoretical ones.

11.2 Algorithm Overview

At a high level, our algorithm is an adaptation of the framework used by [HIKP12a] to the continuous setting. However, getting our result requires a number of subtle changes to the algorithm. This section will describe the most significant ones. We assume some familiarity with previous work in the area [GGI+02, CCF02, GMS05, GLPS10, HIKP12a].

First we describe a high-level overview of the structure. The algorithm proceeds in $\log k$ stages, where each stage attempts to recover each tone with a large constant probability (e.g. 9/10). In each stage, we choose a parameter $\sigma \approx \frac{T}{k \log(k/\delta)}$ that we think of as "hashing" the frequencies into random positions. For this σ, we will choose about $\log(FT)$ different random "start times" t_0 and sample an arithmetic sequence starting at t_0, i.e. observe

$$x(t_0), x(t_0 + \sigma), x(t_0 + 2\sigma), \ldots, x(t_0 + (k \log(k/\delta))\sigma)$$

We then scale these observations by a "window function," which has specific properties but among other things scales down the values near the ends of the sequence, giving a smoother transition between the time before and after we start/end sampling. We alias this down to $B = O(k)$ terms (i.e. add together terms $1, B+1, 2B+1, \ldots$ to get a B-dimensional vector) and take the B-dimensional DFT. This gives a set of B values $\widehat{u}_i$. The observation made in previous papers is that $\widehat{u}$ is effectively a *hashing* of the tones of $\widehat{x}$ into B buckets, where σ defines a permutation on the frequencies that affects whether two different tones land in the same bucket, and $\widehat{u}_j$ approximately equals the sum of all the tones that land in bucket j, each scaled by a phase shift depending on t_0.

Because of this phase shift, for each choice of t_0 the value of $\widehat{u}_j$ is effectively a sample from the Fourier transform of a signal that contains only the tones of $\widehat{x}^*$ that land in bucket

j, with zeros elsewhere. And since there are k tones and $O(k)$ buckets, most tones are alone in their bucket. Therefore this sampling strategy reduces the original problem of k-sparse recovery to one of 1-sparse recovery—we simply choose t_0 according to some strategy that lets us achieve 1-sparse recovery, and recover a tone for each bin.

One-sparse recovery. The algorithm for one-sparse recovery in [HIKP12a] is a good choice for adaptation to the continuous setting. It narrows down to the frequency in a locality-aware way, maintaining an interval of frequencies that decreases in size at each stage (in contrast to the method in [GMS05], which starts from the least significant bit rather than most significant bit).

If a frequency is perturbed slightly in time (e.g., by multiplying by a very slow decay over time) this will blur the frequency slightly into a narrow band. The one-sparse recovery algorithm of [HIKP12a] will proceed normally until it gets to the narrow scale, at which point it will behave semi-arbitrarily and return something near that band. This gives a desired level of robustness—the error in the recovered frequency will be proportional to the perturbation.

Still, to achieve our result we need a few changes to the one-sparse algorithm. One is related to the duration T: in the very last stage of the algorithm, when the interval is stretched at the maximal amount, we can only afford one "fold" rather than the typical $O(\log n)$. The only cost to this is in failure probability, and doing it for one stage is fine—but showing this requires a different proof. Another difference is that we need the final interval to have precision $\frac{1}{T\rho}$ if the signal-to-noise ratio is ρ—the previous analysis showed $\frac{1}{T\sqrt{\rho}}$ and needed to be told ρ, but (as we shall see) to achieve an ℓ_2/ℓ_2 guarantee we need the optimal

ρ-dependence and for the algorithm to be oblivious to the value of ρ. Doing so requires a modification to the algorithm and slightly more clever analysis.

k-sparse recovery. The changes to the k-sparse recovery structure are broader. First, to make the algorithm simpler we drop the [GLPS10]-style recursion with smaller k, and just repeat an $O(k)$-size hashing $O(\log k)$ times. This loses a $\log k$ factor in time and sample complexity, but because of the other changes it is not easy to avoid, and at the same time improves our success probability.

The most significant changes come because we can no longer measure the noise in frequency space or rely on the hash function to randomize the energy that collides with a given heavy hitter. Because we only look at a bounded time window T, Parseval's identity does not hold and the energy of the noise in frequency space may be unrelated to its observed energy. Moreover, if the noise consists of frequencies infinitesimally close to a true frequency, then because σ is bounded the true frequency will always hash to the same bin as the noise. These two issues are what drive the restrictions on noise in the previous work [BCG+12]—assuming the noise is bounded in ℓ_1 norm in frequency domain and is zero in a neighborhood of the true frequencies fixes both issues. But we want a guarantee in terms of the average ℓ_2 noise level $\mathcal{N}^2$ in time domain over the observed duration. If the noise level is $\mathcal{N}^2$, because we cannot hash the noise independently of the signal, we can only hope to guarantee reliable recovery of tones with magnitude larger than $\mathcal{N}^2$. This is in contrast to the $\mathcal{N}^2/k$ that is possible in the discrete setting, and would naively lose a factor of k in the ℓ_2/ℓ_2 approximation.

The insight here is that, even though the noise is not distributed randomly across bins, the total amount of noise is still bounded. If a heavy hitter of magnitude v^2 is not

recovered due to noise, that requires $\Omega(v^2)$ noise mass in the bin that is not in any other bin. Thus the total amount of signal mass not recovered due to noise is $O(\mathcal{N}^2)$, which allows for ℓ_2/ℓ_2 recovery.

This difference is why our algorithm only gets a constant factor approximation rather than the $1+\epsilon$ guarantee that hashing techniques for sparse recovery can achieve in other settings. These techniques hash into $B = O(k/\epsilon)$ bins so the average noise per bin is $O(\frac{\epsilon}{k}\mathcal{N}^2)$. In our setting, where the noise is not hashed independently of the signal, this would give no benefit.

Another difference arises in the choice of the parameter σ, which is the separation between samples in the arithmetic sequence used for a single hashing, and gives the permutation during hashing. In the discrete setting, one chooses σ uniformly over n, which in our setting would correspond to a scale of $\sigma \approx \frac{1}{\eta}$. Since the arithmetic sequences have $O(k\log(k/\delta))$ samples, the duration would then become at least $\frac{k\log(k/\delta)}{\eta}$ (which is why [BCG+12] has this duration). What we observe is that σ can actually be chosen at the scale of $\frac{1}{k\eta}$, giving the desired $O(\frac{\log(k/\delta)}{\eta})$ duration. This causes frequencies at the minimum separation η to always land in bins that are a constant separation apart. This is sufficient because we use [HIKP12a]-style window functions with strong isolation properties (and, in fact, [HIKP12a] could have chosen $\sigma \approx n/B$); it would be an issue if we were using the window functions of [GMS05, IK14] that have smaller supports but less isolation.

Getting an ℓ_2 bound Lastly, converting the guarantee (11.2) into (11.3) is a nontrivial task that is trivial in the discrete setting. In the discrete setting, it follows immediately from the different frequencies being orthogonal to each other. In our setting, we use that

the recovered frequencies should themselves have $\Omega(\eta)$ separation, and that well-separated frequencies are nearly orthogonal over long enough time scales $T \gg 1/\eta$.

This bears some similarity to issues that arise in sparse recovery with overcomplete dictionaries. It would be interesting to see whether further connections can be made between the problems.

11.3 Proof outline

In this section we present the key lemmas along the path to producing the algorithm. The full proof are presented later.

Notation. First we define the notation necessary to understand the lemmas. The full notation as used in the proofs appears in Section 11.4.

The algorithm proceeds in stages, each of which hashes the frequencies to B bins. The hash function depends on two parameters σ and b, and so we define it as $h_{\sigma,b}(f) : [-F, F] \to [B]$.

A tone with a given frequency f can have two "bad events" $E_{coll}(f)$ or $E_{off}(f)$ hold for a given hashing. These correspond to colliding with another frequency of x^* or landing within an α fraction of the edge, respectively; they each will occur with small constant probability.

For a given hashing, we will choose a number of different offsets a that let us perform recovery of the tones that have neither bad event in this stage.

We use $f \lesssim g$ to denote that there exists a constant C such that $f \leq Cg$, and $f \approx g$ to denote $f \lesssim g \lesssim f$.

Key Lemmas First, we need to be able to compare the distance between two pure tone signals in time domain to their differences in parameters. The relation is as follows:

Lemma 11.3.1. Let (v, f) and (v', f') denote any two tones, i.e., (magnitude, frequency)

pairs. Then for

$$\operatorname{dist}\left((v,f),(v',f')\right)^2 := \frac{1}{T}\int_0^T \left|ve^{2\pi ft\mathbf{i}} - v'e^{2\pi f't\mathbf{i}}\right|^2 \mathrm{d}t,$$

we have

$$\operatorname{dist}\left((v,f),(v',f')\right)^2 \eqsim (|v|^2+|v'|^2)\cdot\min(1,T^2|f-f'|^2) + |v-v'|^2,$$

and

$$\operatorname{dist}\left((v,f),(v',f')\right) \eqsim |v|\cdot\min(1,T|f-f'|) + |v-v'|.$$

The basic building block for our algorithm is a function HashToBins, which is very similar to one of the same name in [HIKP12a].

The key property of HashToBins is that, if neither "bad" event holds for a frequency f (i.e. it does not collide or land near the boundary of the bin), then for the bin $j = h_{\sigma,b}(f)$ we have that $|\widehat{u}_j| \approx |\widehat{x^*}(f)|$ with a phase depending on a.

How good is the approximation? In the discrete setting, one can show that each tone has error about $\mathcal{N}^2/B$ in expectation. Here, because the hash function cannot randomize the noise, we instead show that the total error over all tones is about $\mathcal{N}^2$:

Lemma 11.3.2. *Let $\sigma \in [\frac{1}{B\eta}, \frac{2}{B\eta}]$ uniformly at random, then $b \in [0, \frac{\lceil F/\eta\rceil}{\sigma B}]$, $a \in [0, \frac{cT}{\sigma}]$ be sampled uniformly at random for some constant $c > 0$. Let the other parameters be arbitrary in $\widehat{u} = \text{HashToBins}(x, P_{\sigma,a,b}, B, \delta, \alpha)$, and consider*

$$H = \{f \in \operatorname{supp}(\widehat{x}^*) \mid \textit{neither } E_{coll}(f) \textit{ nor } E_{off}(f) \textit{ holds}\}$$

and $I = [B] \setminus h_{\sigma,b}(\operatorname{supp}(\widehat{x}^*))$ to be the bins that have no frequencies hashed to them. Then

$$\mathop{\mathbb{E}}_{\sigma,b,a}\left[\sum_{f\in H}\left|\widehat{u}_{h_{\sigma,b}(f)} - \widehat{x^*}(f)e^{a\sigma 2\pi f\mathbf{i}}\right|^2 + \sum_{j\in I}\widehat{u}_j^2\right] \lesssim \mathcal{N}^2$$

We prove Lemma 11.3.2 by considering the cases of $x^* = 0$ and $g = 0$ separately; linearity then gives the result. Both follow from properties of our window functions.

Lemma 11.3.3. If $x^*(t) = 0, \forall t \in [0, T]$, then

$$\mathop{\mathbb{E}}_{\sigma,a,b}\left[\sum_{j=1}^{B} |\widehat{u}_j|^2\right] \lesssim \frac{1}{T}\int_0^T |g(t)|^2 \mathrm{d}t.$$

Lemma 11.3.4. If $g(t) = 0, \forall t \in [0, T]$. Let H denote a set of frequencies, $H = \{f \in \text{supp}(\widehat{x^*}) \mid \text{neither } E_{coll}(f) \text{ nor } E_{off}(f) \text{ holds}\}$. Then,

$$\mathop{\mathbb{E}}_{\sigma,a,b}\left[\sum_{f\in H} \left|\widehat{u}_{h_{\sigma,b}(f)} - \widehat{x^*}(f)e^{a\sigma 2\pi f \mathbf{i}}\right|^2\right] \leq \delta\|\widehat{x^*}\|_1^2.$$

Lemma 11.3.2 is essentially what we need for 1-sparse recovery. We first show a lemma about the inner call, which narrows the frequency from a range of size Δl to one of size $\frac{\Delta l}{\rho st}$ for some parameters ρst. This gives improved performance (superresolution) when the signal-to-noise ratio ρ within the bucket is high. The parameter s and t provide a tradeoff between success probability, performance, running time, and duration.

Lemma 11.3.5. *Consider any B, δ, α. Algorithm* HashToBins *takes $O(B\log(k/\delta))$ samples and runs in $O(\frac{B}{\alpha}\log(k/\delta) + B\log B)$ time.*

Lemma 11.3.6. Given σ and b, consider any frequency f for which neither $E_{coll}(f)$ nor $E_{off}(f)$ holds, and let $j = h_{\sigma,b}(f)$. Let $\mu^2(f) = \mathbb{E}_a[|\widehat{u}_j - \widehat{x^*}(f)e^{a\sigma 2\pi f\mathbf{i}}|^2]$ and $\rho^2 = |\widehat{x^*}(f)|^2/\mu^2(f)$. For sufficiently large ρ, and $\forall 0 < s < 1, t \geq 4$, consider any run of LocateInner with $f \in [l_j - \frac{\Delta l}{2}, l_j + \frac{\Delta l}{2}]$. It takes $O(R_{loc})$ random $(\gamma, \beta) \in [\frac{1}{2}, 1] \times [\frac{st}{4\sigma\Delta l}, \frac{st}{2\sigma\Delta l}]$ samples over duration $\beta\sigma = \Theta(\frac{st}{\Delta l})$, runs in $O(stR_{loc})$ time, to learn f within a region that has length $\Theta(\frac{\Delta l}{t})$ with failure probability at most $(\frac{4}{s\rho})^{R_{loc}} + t\cdot(60s)^{R_{loc}/2}$.

By repeating this inner loop, we can recover the tones in almost every bin that does not have the "bad" events happen, so we recover a large fraction of the heavy hitters in each stage.

Lemma 11.3.7. Algorithm LocateKSignal takes $O(k \log_C(FT) \log(k/\delta))$ samples over $O(\frac{\log(k/\delta)}{\eta})$ duration, runs in $O(k \log_C(FT) \log(FT/\delta))$ time, and outputs a set $L \subset [-F, F]$ of $O(k)$ frequencies with minimum separation $\Omega(\eta)$.

Given σ and b, consider any frequency f for which neither of $E_{coll}(f)$ or $E_{off}(f)$ hold. Let $j = h_{\sigma,b}(f)$, $\mu^2(f) = \mathbb{E}_a[|\widehat{u}_j - \widehat{x^*}(f)e^{a\sigma 2\pi f \mathbf{i}}|^2]$, and $\rho^2 = |\widehat{x^*}(f)|^2/\mu^2(f)$. If $\rho > C$, then with an arbitrarily large constant probability there exists an $f' \in L$ with

$$|f - f'| \lesssim \frac{1}{T\rho}.$$

Combining this with estimation of the magnitudes of recovered frequencies, we can show that the total error over all bins without "bad" events—that is, bins with either one well placed frequency or zero frequencies—is small. At this point we give no guarantee for the (relatively few) bins with bad events; the recovered values there may be arbitrarily large.

Lemma 11.3.8. Algorithm OneStage takes $O(k \log_C(FT) \log(k/\delta))$ samples over $O(\frac{\log(k/\delta)}{\eta})$ duration, runs in $O(k(\log_C(FT) \log(FT/\delta)))$ time, and outputs a set of $\{(v_i', f_i')\}$ of size $O(k)$ with $\min_{i \neq j} |f_i' - f_j'| \gtrsim \eta$. Moreover, one can imagine a subset $S \subseteq [k]$ of "successful" recoveries, where $\Pr[i \in S] \geq \frac{9}{10}$ $\forall i \in [k]$ and for which there exists an injective function $\pi : [k] \to [O(k)]$ so that

$$\mathop{\mathbb{E}}_{\sigma,b}\left[\sum_{i \in S} \frac{1}{T} \int_0^T \left| v_i' e^{2\pi f_i' t \mathbf{i}} - v_{\pi(i)} e^{2\pi f_{\pi(i)} t \mathbf{i}} \right|^2 \mathrm{d}t\right] \lesssim C^2 \mathcal{N}^2.$$

with $1 - 1/k^c$ probability for an arbitrarily large constant c.

We can repeat the procedure for $O(\log k)$ stages and merge the results, getting a list of $O(k)$ tones that includes k tones that match up well to the true tones. However, we give no guarantee for the rest of the recovered tones at this point—as far as the analysis is concerned, mistakes from bins with collisions may cause arbitrarily large spurious tones.

Lemma 11.3.9. Repeating algorithm OneStage $O(\log k)$ times, MergedStages returns a set $\{(v_i', f_i')\}$ of size $O(k)$ with $\min_{i \neq j} |f_i' - f_j'| \gtrsim \eta$ that can be indexed by π such that

$$\sum_{i=1}^{k} \frac{1}{T} \int_0^T \left| v_i' e^{2\pi f_i' t \mathbf{i}} - v_{\pi(i)} e^{2\pi f_{\pi(i)} t \mathbf{i}} \right|^2 \mathrm{d}t \lesssim C^2 \mathcal{N}^2.$$

with probability $1 - 1/k^c$ for an arbitrarily large constant c.

To address the issue of spurious tones, we run the above algorithm twice and only take the tones that are recovered in both stages. We show that the resulting $O(k)$ tones are together a good approximation to the vector.

Lemma 11.3.10. If we run MergedStages twice and take the tones $\{(v_i', f_i')\}$ from the first result that have f_i' within $c\eta$ for small c of some frequency in the second result, we get a set of $k'' = O(k)$ tones that can be indexed by some permutation π such that

$$\sum_{i=1}^{k} \frac{1}{T} \int_0^T \left| v_i' e^{2\pi f_i' t \mathbf{i}} - v_{\pi(i)} e^{2\pi f_{\pi(i)} t \mathbf{i}} \right|^2 \mathrm{d}t + \sum_{i=k+1}^{k''} |v_i'|^2 \lesssim C^2 \mathcal{N}^2. \tag{11.5}$$

Simply picking out the largest k recovered tones then gives the result (11.2).

Theorem 11.3.11. Algorithm ContinuousFourierSparseRecovery returns a set $\{(v_i', f_i')\}$ of size k with $\min_{i \neq j} |f_i' - f_j'| \gtrsim \eta$ for which

$$\sum_{i=1}^{k} \frac{1}{T} \int_0^T \left| v_i' e^{2\pi f_i' t \mathbf{i}} - v_{\pi(i)} e^{2\pi f_{\pi(i)} t \mathbf{i}} \right|^2 \mathrm{d}t \lesssim C^2 \mathcal{N}^2$$

with probability $1 - 1/k^c$ for an arbitrarily large constant c.

By only considering the term in the sum corresponding to tone i and applying Lemma 11.3.1, we get result (11.1):

Corollary 11.3.12. *With probability $1 - 1/k^c$ for an arbitrarily large constant c, we recover a set of tones $\{(v_i', f_i')\}$ such that, for any v_i with $|v_i| \gtrsim \mathcal{N}$, we have for an appropriate permutation of the indices that*

$$|f_i' - f_i| \lesssim \frac{\mathcal{N}}{T|v_i|} \qquad \text{and} \qquad |v_i' - v_i| \lesssim \mathcal{N}. \tag{11.6}$$

We then show that (11.2) implies (11.3) for sufficiently long durations T. A long duration helps because it decreases the correlation between η-separated frequencies.

Lemma 11.3.13. Let $\{(v_i, f_i)\}$ and $\{(v_i', f_i')\}$ be two sets of k tones for which $\min_{i \neq j} |f_i - f_j| \geq \eta$ and $\min_{i \neq j} |f_i' - f_j'| \gtrsim \eta$ for some $\eta > 0$. Suppose that $T > O(\frac{\log^2 k}{\eta})$. Then these sets can be indexed such that

$$\frac{1}{T} \int_0^T \left| \sum_{i=1}^{k} (v_i' e^{2\pi \mathbf{i} f_i' t} - v_i e^{2\pi \mathbf{i} f_i t}) \right|^2 \mathrm{d}t \lesssim \sum_{i=1}^{k} \frac{1}{T} \int_0^T \left| v_i' e^{2\pi \mathbf{i} f_i' t} - v_i e^{2\pi \mathbf{i} f_i t} \right|^2 \mathrm{d}t. \tag{11.7}$$

Combining Theorem 11.3.11 and Lemma 11.3.13 immediately implies

Theorem 11.3.14. *Suppose we sample for a duration $T > O(\frac{\log(1/\delta) + \log^2 k}{\eta})$. Then the reconstructed signal $x'(t) = \sum_{i=1}^{k} v_i' e^{2\pi \mathbf{i} f_i' t}$ achieves a constant factor approximation to the complete signal x:*

$$\frac{1}{T} \int_0^T |x'(t) - x(t)|^2 \mathrm{d}t \lesssim C^2 \mathcal{N}^2. \tag{11.8}$$

The algorithm takes $O(k \log \frac{F}{\eta} \log(\frac{k}{\delta}) \log(k))$ samples, runs in $O(k \log \frac{F}{\eta} \log(\frac{FT}{\delta}) \log(k))$ time, and succeeds with probability at least $1 - 1/k^c$ for an arbitrarily large constant c.

That finishes the proof of our main theorem. We also show that our "superresolution" precision from (11.1) is optimal, which is a simple corollary of Lemma 11.3.1.

Lemma 11.3.15. There exists a constant $c > 0$ such that, for a given sample duration T, one cannot recover the frequency f to within

$$c\frac{\mathcal{N}}{T|\widehat{x}^*(f)|}$$

with $3/4$ probability, for all $\delta > 0$, even if $k = 1$.

11.4 Notation and Definitions: Permutation, Hashing, Filters

This section gives definitions about the permutation, hashing, and filters that are used throughout the proofs. Let $[n]$ denote the set $\{1, 2, \cdots, n-1, n\}$. $\mathbb{R}$ denotes the real numbers, $\mathbb{Z}$ denotes the integer numbers and $\mathbb{C}$ denotes the complex numbers. The convolution of two continuous functions f and g is written as $f * g$,

$$(f * g)(t) := \int_{-\infty}^{+\infty} f(\tau) g(t-\tau) \mathrm{d}\tau$$

and the discrete convolution of f and g is given by,

$$(f * g)[n] := \sum_{m=-\infty}^{+\infty} f[m] g[n-m]$$

Let $\mathbf{i}$ denote $\sqrt{-1}$, and $e^{\mathbf{i}\theta} = \cos(\theta) + \mathbf{i}\sin(\theta)$. For any complex number $z \in \mathbb{C}$, we have $z = a + \mathbf{i}b$, where $a, b \in \mathbb{R}$. Define $\overline{z} = a - \mathbf{i}b$, $|z|^2 = z\overline{z} = a^2 + b^2$ and let $\phi(z)$ be the phase of z. Let $\mathrm{supp}(f)$ denote the support of function/vector f, and $\|f\|_0 = |\mathrm{supp}(f)|$. For any $p \in [1, \infty]$, the ℓ_p norm of a vector of x is $\|x\|_p = (\sum_i |x_i|^p)^{\frac{1}{p}}$, defined to be $\max_i |x_i|$ for $p = \infty$. Let k denote the sparsity of frequency domain. All the frequencies $\{f_1, f_2, \cdots, f_k\}$ are from $[-F, F]$. Let $B = O(k)$ denote the number of hash bins in our algorithm.

We translate the "permutation" $P_{\sigma,a,b}$ of [HIKP12a] from the DFT setting to the DTFT setting.

Definition 11.4.1. $(P_{\sigma,a,b}x^*)(t) = x^*(\sigma(t-a))e^{-2\pi\mathbf{i}\sigma bt}$.

Lemma 11.4.1. $\widehat{P_{\sigma,a,b}x^*}(\sigma(f-b)) = e^{-2\pi f\sigma a\mathbf{i}}\widehat{x}^*(f)$

Proof. The time domain representation of the given Fourier definition would be

$$\begin{aligned} x'(t) &= \sum_{f \in \text{supp}(\widehat{x}^*)} e^{-2\pi \mathbf{i} \sigma a f} \widehat{x}^*(f) e^{2\pi \mathbf{i} \sigma (f-b)t} \\ &= \sum_{f \in \text{supp}(\widehat{x}^*)} e^{-2\pi \mathbf{i} \sigma b t} \widehat{x}^*(f) e^{2\pi \mathbf{i} \sigma (t-a) f} \\ &= e^{-2\pi \mathbf{i} \sigma b t} x^*(\sigma(t-a)) \end{aligned}$$

which matches, so the formula is right. □

We also extend the flat window function for the DFT setting [HIKP12a], [HIKP12b] to the DTFT setting:

Definition 11.4.2. Let $M = O(B \log \frac{k}{\delta})$. We say that $(G, \widehat{G'}) = (G_{B,\delta,\alpha}, \widehat{G'}_{B,\delta,\alpha}) \in \mathbb{R}^M \times \mathbb{R}^{[-F,F]}$ is a flat window function with parameters $B \geq 1$, $\delta > 0$, and $\alpha > 0$. For simplicity, let's say B is a function of α. Define $|\text{supp}(G)| = M$ and $\widehat{G'}$ satisfies

- $G_i = \frac{\sin(i\frac{1}{B})}{i} \cdot e^{-\frac{i^2}{2\sigma^2}}$, where $\sigma = \Theta(B\sqrt{\log(k/\delta)})$.
- $\widehat{G}(f) = \sum_{i=1}^{M} G_i e^{f \cdot \frac{i}{M} 2\pi \mathbf{i}}$.
- $\text{supp}(\widehat{G'}) \subset [-\frac{2\pi}{2B}, \frac{2\pi}{2B}]$.
- $\widehat{G'}(f) = 1$ for all $f \in [-\frac{(1-\alpha)2\pi}{2B}, \frac{(1-\alpha)2\pi}{2B}]$.
- $\widehat{G'}(f) = 0$ for all $|f| \geq \frac{2\pi}{2B}$.
- $\widehat{G'}(f) \in [0,1]$ for all f.
- $\|\widehat{G'} - \widehat{G}\|_\infty^2 < \delta/k$.

Claim 11.4.2. $\sum_{i=1}^{M} G_i^2 \eqsim \frac{1}{B}$, *where* $M = O(B \log k/\delta)$.

Proof. By definition of G_i, we have

$$\sum_{i=1}^{M} G_i^2 = 2 \sum_{i=1}^{M/2} \frac{\sin^2(i\frac{1}{B})}{(i)^2} (e^{-\frac{(i)^2}{2\sigma^2}})^2.$$

There exists some constant $c \in [0, 2\pi)$, such that $\frac{\sin(i/B)}{i} \eqsim \frac{1}{B}$ if $i/B < c\pi$.

$$\begin{aligned}
\sum_{i=1}^{M} G_i^2 &= 2 \sum_{i=1}^{\lfloor Bc\pi \rfloor} \frac{\sin^2(i\frac{1}{B})}{(i)^2} (e^{-\frac{(i)^2}{2\sigma^2}})^2 + 2 \sum_{i=\lceil Bc\pi \rceil}^{M/2} \frac{\sin^2(i\frac{1}{B})}{(i)^2} (e^{-\frac{(i)^2}{2\sigma^2}})^2 \\
&\leq 2 \sum_{i=1}^{\lfloor Bc\pi \rfloor} \frac{\sin^2(i\frac{1}{B})}{(i)^2} \cdot 1 + 2 \sum_{i=\lceil Bc\pi \rceil}^{M/2} \frac{\sin^2(i\frac{1}{B})}{(i)^2} \cdot 1 \\
&\lesssim \sum_{i=1}^{\lfloor Bc\pi \rfloor} \frac{1}{B^2} + \sum_{i=\lceil Bc\pi \rceil}^{M/2} \frac{1}{i^2} \\
&\lesssim \frac{1}{B}.
\end{aligned}$$

Thus, we show an upper bound. It remains to prove the lower bound.

$$\begin{aligned}
\sum_{i=1}^{M} G_i^2 &\geq 2 \sum_{i=1}^{\lfloor Bc\pi \rfloor} \frac{\sin^2(i\frac{1}{B})}{(i)^2} (e^{-\frac{(i)^2}{2\sigma^2}})^2 \\
&\geq 2 \sum_{i=1}^{\lfloor Bc\pi \rfloor} \frac{\sin^2(i\frac{1}{B})}{(i)^2} (e^{-\frac{(Bc\pi)^2}{2\sigma^2}})^2 \\
&\gtrsim 2 \sum_{i=1}^{\lfloor Bc\pi \rfloor} \frac{1}{B^2} (e^{-\frac{(Bc\pi)^2}{2\sigma^2}})^2 \\
&\gtrsim \frac{1}{B} (e^{-\frac{(Bc\pi)^2}{2\sigma^2}})^2 \\
&\gtrsim \frac{1}{B} (e^{-c_0})^2.
\end{aligned}$$

The last inequality follows by there exists some universal constant $c_0 > 0$ such that $-\frac{1}{\log(k/\delta)} \gtrsim -c_0$. Thus, we show $\sum_{i=1}^{M} G_i^2 \gtrsim \frac{1}{B}$. □

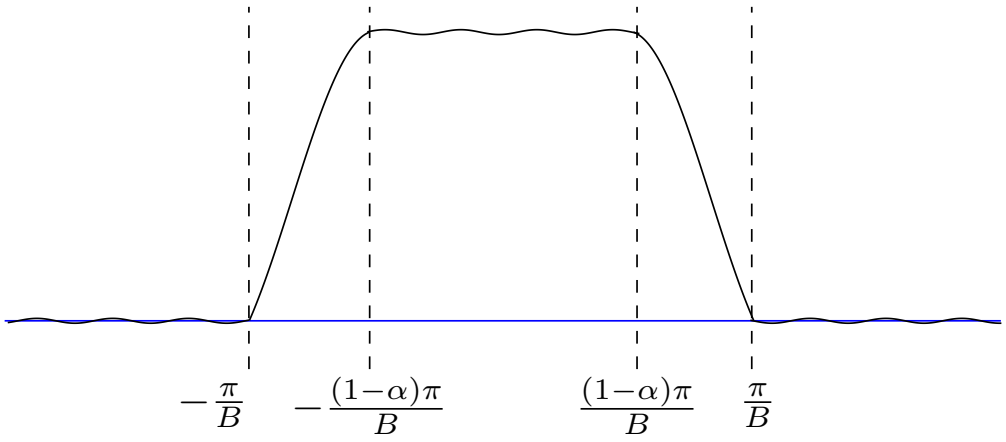


Figure 11.1: Filter $\widehat{G'}(f)$

To analyze the details of our algorithm, we explain some lower-level definitions and claims first. Here we give the definition of three notations that related to hash function.

Definition 11.4.3. $\pi_{\sigma,b}(f) = 2\pi\sigma(f - b) \pmod{2\pi}$. We maps frequency to a circle $[0, 2\pi)$, since our observation of sample is the phase of some complex number, which also belongs to $[0, 2\pi)$.

Definition 11.4.4. $h_{\sigma,b}(f) = \text{round}(\pi_{\sigma,b}(f) \cdot \frac{B}{2\pi})$. $h_{\sigma,b}(f)$ is a "hash function" that hashes frequency f into one of the B bins. The motivation is, it is very likely that each bin only has 1 heavy hitters if we choose large enough B. Then, for each bin, we can run a 1-sparse algorithm to recover the frequency.

Definition 11.4.5. $o_{\sigma,b}(f) = \pi_{\sigma,b}(f) - \frac{2\pi}{B} \cdot h_{\sigma,b}(f)$. Offset $o_{\sigma,b}(f)$ denotes the distance from $\pi_{\sigma,b}(f)$ to the center of the corresponding bin that frequency f was hashed into.

Then we define some events that might happen after applying hash function to the entire frequency domain.

Definition 11.4.6. "Collision" event $E_{coll}(f)$: holds iff $h_{\sigma,b}(f) \in h_{\sigma,b}(\mathrm{supp}(\widehat{x^*})\backslash\{f\})$. The "collision" event happening means there exists some other frequency f' such that both f and f' are hashed into the same bin. Once two frequencies are colliding in one bin, the algorithm will not be able to recover them.

Definition 11.4.7. "Large offset" event $E_{off}(f)$: holds iff $|o_{\sigma,b}(f)| \geq (1-\alpha)\frac{2\pi}{2B}$. The event holds if frequency f is not within factor $1-\alpha$ of the radius close to the center of that hash bin. It causes the frequency to be in the intermediate regime of filter and not recoverable, see Figure 11.1.

Definition 11.4.8. We sample σ uniformly at random from $[\frac{1}{B\eta}, \frac{2}{B\eta}]$. Conditioning on σ is chosen first, we sample b uniformly at random from $[0, \frac{\lceil F/\eta \rceil}{B\sigma}]$. Then we sample γ uniformly at random from $[\frac{1}{2}, 1]$ and β uniformly at random from $[\widehat{\beta}, 2\widehat{\beta}]$, where $\widehat{\beta}$ is dynamically changing during our algorithm(The details of setting $\widehat{\beta}$ are explained in Lemma 11.3.6). For $P_{\sigma,\gamma,b}$ and $P_{\sigma,\gamma+\beta,b}$, we take the following two sets of samples over time domain,

$$x(\sigma(1-\gamma)), x(\sigma(2-\gamma)), x(\sigma(3-\gamma)), \cdots, x(\sigma(B\log(k/\delta)-\gamma))$$
$$x(\sigma(1-\gamma-\beta)), x(\sigma(2-\gamma-\beta)), x(\sigma(3-\gamma-\beta)), \cdots, x(\sigma(B\log(k/\delta)-\gamma-\beta))$$

Conditioning on drawing σ, b from some distribution, we are able to show that the probability of "Collision" and "Large offset" event holding are small.

Lemma 11.4.3. *For any $\widetilde{T}$, and $0 \leq \widetilde{\epsilon}, \widetilde{\delta} \leq \widetilde{T}$, if we sample $\widetilde{\sigma}$ uniformly at random from $[A, 2A]$, then*

$$\frac{2\widetilde{\epsilon}}{\widetilde{T}} - \frac{2\widetilde{\epsilon}}{A} \leq \Pr\left[\widetilde{\sigma} \pmod{\widetilde{T}} \in [\widetilde{\delta}-\widetilde{\epsilon}, \widetilde{\delta}+\widetilde{\epsilon}\,]\right] \leq \frac{2\widetilde{\epsilon}}{\widetilde{T}} + \frac{4\widetilde{\epsilon}}{A}. \tag{11.9}$$

Proof. Since we sample $\widetilde{\sigma}$ uniformly at random from $[A, 2A]$, let I denote a set of candidate integers, then the smallest one is $\lfloor A \rfloor$ and the largest one is $\lceil 2A \rceil$. Thus, the original probability equation is equivalent to

$$\Pr\left[\widetilde{\sigma} \in [s \cdot \widetilde{T} + \widetilde{\delta} - \widetilde{\epsilon}, s \cdot \widetilde{T} + \widetilde{\delta} + \widetilde{\epsilon}\,] \; \exists s \in I\right], \tag{11.10}$$

where $I = \{\lfloor A/\widetilde{T} \rfloor, \cdots, \lceil 2A/\widetilde{T} \rceil\}$.

Consider any $s \in I$, the probability of $\widetilde{\sigma}$ belonging to the interval $[s\cdot\widetilde{T}+\widetilde{\delta}-\widetilde{\epsilon}, s\cdot\widetilde{T}+\widetilde{\delta}+\widetilde{\epsilon}]$ is

$$\Pr\left[\widetilde{\sigma} \in [s \cdot \widetilde{T} + \widetilde{\delta} - \widetilde{\epsilon}, s \cdot \widetilde{T} + \widetilde{\delta} + \widetilde{\epsilon}\,]\right] = \frac{2\widetilde{\epsilon}}{A}.$$

Taking the summation over all $s \in I$, we obtain

$$\begin{aligned}
& \Pr\left[\widetilde{\sigma} \in [s \cdot \widetilde{T} + \widetilde{\delta} - \widetilde{\epsilon}, s \cdot \widetilde{T} + \widetilde{\delta} + \widetilde{\epsilon}\,] \; \exists s \in I\right] \\
= & \sum_{s \in I} \Pr\left[\widetilde{\sigma} \in [s \cdot \widetilde{T} + \widetilde{\delta} - \widetilde{\epsilon}, s \cdot \widetilde{T} + \widetilde{\delta} + \widetilde{\epsilon}\,]\right] \\
= & \sum_{s \in I} \frac{2\widetilde{\epsilon}}{A} \\
= & \frac{2\widetilde{\epsilon}|I|}{A}.
\end{aligned}$$

It remains to bound $\frac{2\widetilde{\epsilon}|I|}{A}$. Since $|I| = \lceil 2A/\widetilde{T} \rceil - \lfloor A/\widetilde{T} \rfloor + 1$, then we have an upper bound for I,

$$|I| \leq A/\widetilde{T} + 2.$$

On the other side, we have an lower bound,

$$|I| \geq A/\widetilde{T} - 1.$$

Plugging upper bound of I into $\frac{2\widetilde{\epsilon}|I|}{A}$,

$$\frac{2\widetilde{\epsilon}|I|}{A} \leq \frac{2\widetilde{\epsilon}}{A}(A/\widetilde{T}+2) = \frac{2\widetilde{\epsilon}}{\widetilde{T}} + \frac{4\widetilde{\epsilon}}{A}.$$

Using the lower bound of $|I|$, we have

$$\frac{2\widetilde{\epsilon}|I|}{A} \geq \frac{2\widetilde{\epsilon}}{A}(A/\widetilde{T}-1) = \frac{2\widetilde{\epsilon}}{\widetilde{T}} - \frac{2\widetilde{\epsilon}}{A}.$$

Thus, we complete the proof. □

The following corollary will be used many times in this paper. The proof directly follows by Lemma 11.4.3.

Corollary 11.4.4. *For any $\widetilde{T}$, Δf, and $0 \leq \widetilde{\epsilon}, \widetilde{\delta} \leq \widetilde{T}$, if we sample $\widetilde{\sigma}$ uniformly at random from $[A, 2A]$, then*

$$\frac{2\widetilde{\epsilon}}{\widetilde{T}} - \frac{2\widetilde{\epsilon}}{A\Delta f} \leq \Pr\left[\widetilde{\sigma}\Delta f \pmod{\widetilde{T}} \in [\widetilde{\delta}-\widetilde{\epsilon}, \widetilde{\delta}+\widetilde{\epsilon}\,]\right] \leq \frac{2\widetilde{\epsilon}}{\widetilde{T}} + \frac{4\widetilde{\epsilon}}{A\Delta f}. \tag{11.11}$$

Proof. Since $\widetilde{\sigma}$ is sampled uniformly at random from $[A, 2A]$, then $\widetilde{\sigma}\Delta f$ is sampled uniformly at random from $[A\Delta f, 2A\Delta f]$. Now applying Lemma 11.4.3 by only replacing $A\Delta f$ by A. □

Claim 11.4.5. *Let σ be sampled uniformly at random from $[\frac{1}{B\eta}, \frac{2}{B\eta}]$ and $\min_{i\neq j}|f_i - f_j| > \eta$. $\forall i, j \in [k]$, if $i \neq j$, then $\Pr[h_{\sigma,b}(f_i) = h_{\sigma,b}(f_j)] \lesssim \frac{1}{B}$.*

Proof. To simplify the proof, define $\Delta f = |f_i - f_j|$. We consider two cases: (I) $\eta \leq |f_i - f_j| < \frac{(B-1)\eta}{2}$, (II) $\frac{(B-1)\eta}{2} \leq |f_i - f_j|$.

(I) If $\Delta f = \eta$, then $2\pi\sigma\Delta f$ is at least $\frac{2\pi}{\eta B} \cdot \eta = \frac{2\pi}{B}$, which means two frequencies have to go to different bins after hashing. If $\Delta f = \frac{(B-1)\eta}{2}$, then $2\pi\sigma\Delta f$ is at most $\frac{4\pi}{B\eta} \cdot \frac{(B-1)\eta}{2} =$

$(1-1/B)2\pi$. In order to make two frequencies collide, $2\pi\sigma\Delta f$ should belong to $[(1-1/B)2\pi, (1+1/B)2\pi)$. Since for any $\Delta f \in [\eta, \frac{(B-1)\eta}{2})$, we have $2\pi\sigma\Delta f \in [\frac{1}{B}2\pi, (1-1/B)2\pi)$, which does not intersect interval $[(1-1/B)2\pi, (1+1/B)2\pi)$. Thus,

$$\Pr_{\sigma,b}[h_{\sigma,b}(f_i) = h_{\sigma,b}(f_j)] = 0.$$

(II) We apply Corollary 11.4.4 by setting $\widetilde{T} = 2\pi$, $\widetilde{\sigma} = 2\pi\sigma$, $\widetilde{\delta} = 0$, $\widetilde{\epsilon} = \frac{2\pi}{2B}$, $A = 2\pi\frac{1}{B\eta}$. Then we have

$$\Pr_{\sigma,b}[h_{\sigma,b}(f_i) = h_{\sigma,b}(f_j)] = \Pr_{\sigma,b}\left[2\pi\sigma\Delta f \in \left[s\cdot 2\pi - \frac{2\pi}{2B}, s\cdot 2\pi + \frac{2\pi}{2B}\right] \exists\ s\ \in\ I\right] \quad (11.12)$$

where

$$I = \left\{\lfloor \frac{1}{B\eta}\Delta f\rfloor, \cdots, \lceil \frac{2}{B\eta}\Delta f\rceil\right\}.$$

By upper bound of Corollary 11.4.4, Equation (11.12) is at most

$$\frac{1}{\frac{2\pi}{B\eta}\Delta f}\cdot\frac{2\pi}{B}\cdot(\frac{1}{B\eta}\Delta f + 2) = \frac{1}{B} + \frac{2\eta}{\Delta f} \le \frac{1}{B} + \frac{4}{B-1} \lesssim \frac{1}{B},$$

where the first inequality follows by $\frac{(B-1)\eta}{2} \le |f_i - f_j|$ which is the assumption of part (II).

□

Claim 11.4.6. $\forall f, \forall\ 0<\alpha<1$, $\Pr_{\sigma,b}\left[|o_{\sigma,b}(f)| \le (1-\alpha)\frac{2\pi}{2B}\right] \ge 1 - O(\alpha)$.

Proof. Since we draw σ uniformly at random from $[\frac{1}{B\eta}, \frac{2}{B\eta}]$, then $2\pi\sigma(f-b) \in [\frac{2\pi}{B\eta}(f-b), \frac{4\pi}{B\eta}(f-b)]$ uniformly at random. The probability is equal to

$$\Pr_{\sigma,b}\left[2\pi\sigma(f-b) \in \left[s\cdot\frac{2\pi}{B} - (1-\alpha)\frac{2\pi}{2B}, s\cdot\frac{2\pi}{B} + (1-\alpha)\frac{2\pi}{2B}\right]\ \exists s\in I\right],$$

where

$$I = \left\{ \lfloor \frac{2\pi}{B\eta}(f-b)\frac{B}{2\pi} - \frac{1-\alpha}{2} \rfloor, \cdots, \lceil \frac{4\pi}{B\eta}(f-b)\frac{B}{2\pi} + \frac{1-\alpha}{2} \rceil \right\}.$$

We apply Corollary 11.4.4 by setting $\widetilde{T} = \frac{2\pi}{B}$, $\widetilde{\sigma} = 2\pi\sigma$, $\widetilde{\delta} = 0$, $\widetilde{\epsilon} = (1-\alpha)\frac{2\pi}{2B}$, $A = 2\pi\frac{1}{B\eta}$, $\Delta f = |f-b|$.

By lower bound of Corollary 11.4.4, we have

$$\Pr\left[|o_{\sigma,b}(f)| \leq (1-\alpha)\frac{2\pi}{2B}\right] \geq (1-\alpha) - (1-\alpha)\cdot\frac{\eta}{|f-b|}.$$

Since $\alpha \cdot \frac{\eta}{|f-b|} > 0$, then

$$\Pr\left[|o_{\sigma,b}(f)| \leq (1-\alpha)\frac{2\pi}{2B}\right] \geq (1-\alpha) - \frac{\eta}{|f-b|}.$$

Recall that we sample σ uniformly at random from $[\frac{1}{B\eta}, \frac{2}{B\eta}]$ and sample b uniformly at random from $[0, \frac{\lceil F/\eta \rceil}{B\sigma}]$. Since $f \in [-F, F]$ and b is uniformly chosen from range $(0, \frac{\lceil F/\eta \rceil}{B\sigma}]$, thus for any $C > 0$, $\Pr_b[|f-b| \leq C\eta] \lesssim \frac{C\eta}{F}$. Replacing C by $1/\alpha$, we have $\Pr[\frac{\eta}{|f-b|} \leq \alpha] \geq 1 - \Theta(\frac{\eta}{\alpha F})$. Compared to $\frac{\eta}{F}$, α is just a constant. Thus, we finish the proof.

□

11.5 Proofs of basic hashing-related lemmas

Lemma 11.3.1. Let (v, f) and (v', f') denote any two tones, i.e., (magnitude, frequency) pairs. Then for

$$\operatorname{dist}\left((v,f),(v',f')\right)^2 := \frac{1}{T}\int_0^T \left|ve^{2\pi ft\mathbf{i}} - v'e^{2\pi f't\mathbf{i}}\right|^2 \mathrm{d}t,$$

we have

$$\operatorname{dist}\left((v,f),(v',f')\right)^2 \eqsim (|v|^2 + |v'|^2)\cdot\min(1, T^2|f-f'|^2) + |v-v'|^2,$$

and

$$\operatorname{dist}\left((v,f),(v',f')\right) \eqsim |v|\cdot\min(1, T|f-f'|) + |v-v'|.$$

Proof. Define $\nu = |f - f'|$. First, let's show the first upper bound. We have that

$$\begin{aligned}
\text{LHS} &= \frac{1}{T}\int_0^T \left|ve^{2\pi ft\mathbf{i}} - v'e^{2\pi f't\mathbf{i}}\right|^2 \mathrm{d}t \\
&\le 2\cdot\frac{1}{T}\int_0^T \left|ve^{2\pi ft\mathbf{i}} - ve^{2\pi f't\mathbf{i}}\right|^2 + \left|ve^{2\pi f't\mathbf{i}} - v'e^{2\pi f't\mathbf{i}}\right|^2 \mathrm{d}t \\
&= 2|v|^2\cdot\frac{1}{T}\int_0^T \left|e^{2\pi\nu t\mathbf{i}} - 1\right|^2 \mathrm{d}t + 2|v-v'|^2 \\
&\le 2|v|^2\cdot\frac{1}{T}\int_0^T \min(2, 2\pi\nu t)^2 \mathrm{d}t + 2|v-v'|^2 \\
&\le 2|v|^2\cdot\min(4, \frac{4\pi^2}{3}\nu^2T^2) + 2|v-v'|^2,
\end{aligned}$$

as desired.

Now consider the lower bound. First we show this in the setting where $|v| = |v'|$.

Suppose $v' = ve^{-\theta \mathbf{i}}$. Then we want to bound

$$\begin{aligned}\text{LHS} &= \frac{1}{T}\int_0^T \left|ve^{2\pi f t\mathbf{i}} - ve^{(2\pi f' t-\theta)\mathbf{i}}\right|^2 \mathrm{d}t \\ &= |v|^2 \frac{1}{T}\int_0^T \left|e^{(2\pi\nu t+\theta)\mathbf{i}} - 1\right|^2 \mathrm{d}t,\end{aligned}$$

as being at least $\Omega(|v|^2(\min(1, \nu^2T^2) + \theta^2))$. In the case that $\nu T < 1/10$, then

$$\left|e^{(2\pi\nu t+\theta)\mathbf{i}} - 1\right| \gtrsim |2\pi\nu t + \theta|,$$

and

$$\mathop{\mathbb{E}}_{t}[(2\pi\nu t + \theta)^2] \geq (\theta - \frac{2\pi\nu T}{2})^2 + \mathbb{E}[(2\pi\nu(t - T/2))^2] \gtrsim \nu^2T^2 + (\theta - \frac{2\pi\nu T}{2})^2 \gtrsim \nu^2T^2 + \theta^2.$$

On the other hand, if $\nu T > 1/10$, then $2\pi\nu t - \theta$ is $\Omega(1)$ for at least a constant fraction of the t, giving that

$$\left|e^{(2\pi\nu t+\theta)\mathbf{i}} - 1\right| \gtrsim 1.$$

Hence the lower bound holds whenever $|v| = |v'|$.

Finally, consider the lower bound for $|v| \neq |v'|$. Without loss of generality assume $|v'| \geq |v|$, and define $v^* = \frac{|v|}{|v'|}v''$. For any two angles θ, θ' we have that

$$|ve^{\theta\mathbf{i}} - v'e^{\theta'\mathbf{i}}|^2 \geq |ve^{\theta\mathbf{i}} - v^*e^{\theta'\mathbf{i}}|^2 + |v^* - v'|^2,$$

because the angle $\angle vv^*v'$ is obtuse. Therefore

$$\begin{aligned}\text{LHS} &\geq \frac{1}{T}\int_0^T \left|ve^{2\pi f t\mathbf{i}} - v^*e^{2\pi f' t\mathbf{i}}\right|^2 + |v^* - v'|^2 \mathrm{d}t \\ &\gtrsim |v|^2 \min(1, \nu^2T^2) + |v - v^*|^2 + |v^* - v'|^2 \\ &\gtrsim |v|^2 \min(1, \nu^2T^2) + |v - v'|^2.\end{aligned}$$

Now, if $|v'|^2 \leq 2|v|^2$, this gives the desired bound. But otherwise, it also gives the desired bound because $|v - v'|^2 \gtrsim |v'|^2$. So we get the bound in all settings.

The second equation follows from the first, using that $(a+b)^2 \eqsim a^2 + b^2$ for positive a, b to show

$$\operatorname{dist}((v,f),(v',f')) \eqsim (|v| + |v'|) \cdot \min(1, T|f - f'|) + |v - v'|.$$

We can then replace $|v| + |v'|$ with $|v|$ because either they are equivalent up to constants or $|v - v'|$ is within a constant factor of $|v| + |v'|$. □

Lemma 11.3.3. If $x^*(t) = 0, \forall t \in [0, T]$, then

$$\mathop{\mathbb{E}}_{\sigma,a,b}\left[\sum_{j=1}^{B} |\widehat{u}_j|^2\right] \lesssim \frac{1}{T}\int_0^T |g(t)|^2 \mathrm{d}t.$$

Proof. Since $\widehat{u}_j = \mathrm{FFT}(u_j)$, then $\sum_{j=1}^{B} |\widehat{u}_j|^2 = B\sum_{j=1}^{B} |u_j|^2$. Recall that $(P_{\sigma,a,b}x)(t) = x(\sigma(t-a))e^{2\pi\sigma bt\mathbf{i}}$, $u_j = \sum_{i=1}^{\log(k/\delta)} y_{j+Bi}$ and $y_j = G_j \cdot (P_{\sigma,a,b}g)_j = G_j \cdot g(\sigma(j-a))e^{2\pi\mathbf{i}\sigma bj}$. Then

$$\begin{aligned}
\mathop{\mathbb{E}}_{\sigma,a,b}\left[\sum_{j=1}^{B} |u_j|^2\right] &= \mathop{\mathbb{E}}_{\sigma,a,b}\left[\sum_{j=1}^{B}\left|\sum_{i=1}^{\log(k/\delta)} y_{j+Bi}\right|^2\right] \\
&= \mathop{\mathbb{E}}_{\sigma,a}\left[\sum_{j=1}^{B}\mathop{\mathbb{E}}_{b}\left|\sum_{i=1}^{\log(k/\delta)} y_{j+Bi}\right|^2\right] \\
&= \mathop{\mathbb{E}}_{\sigma,a}\left[\sum_{j=1}^{B}\mathop{\mathbb{E}}_{b}\left[\left(\sum_{i=1}^{\log(k/\delta)} y_{j+Bi}\right)\left(\sum_{i'=1}^{\log(k/\delta)} \overline{y_{j+Bi'}}\right)\right]\right] \\
&= \mathop{\mathbb{E}}_{\sigma,a}\left[\sum_{j=1}^{B}\mathop{\mathbb{E}}_{b}\left[\sum_{i=1}^{\log(k/\delta)} y_{j+Bi}\overline{y_{j+Bi}} + \sum_{i\neq i'}^{\log(k/\delta)} y_{j+Bi}\overline{y_{j+Bi'}}\right]\right].
\end{aligned}$$

For any $(i,j) \in [\log(k/\delta)] \times [B]$, let $S_{i,j} = G_{j+Bi} g(\sigma(j+Bi-a)) = y_{j+Bi} e^{-2\pi \mathbf{i} \sigma b(j+Bi)}$, then

$$\mathop{\mathbb{E}}_{\sigma,a,b}\left[\sum_{j=1}^{B} |u_j|^2\right] = \mathop{\mathbb{E}}_{\sigma,a}\left[\sum_{j=1}^{B} \mathop{\mathbb{E}}_{b}\left[\underbrace{\sum_{i=1}^{\log(k/\delta)} |S_{i,j}|^2}_{C_1} + \underbrace{\sum_{i\neq i'}^{\log(k/\delta)} S_{i,j}\overline{S_{i',j}} e^{2\pi \mathbf{i}\sigma b B(i-i')}}_{C_2}\right]\right].$$

Consider the expectation of C_2:

$$\begin{aligned}
\mathop{\mathbb{E}}_{b}[C_2] &= \mathop{\mathbb{E}}_{b}\left[\sum_{i\neq i'}^{\log(k/\delta)} S_{i,j}\overline{S_{i',j}} e^{2\pi \mathbf{i}\sigma b B(i-i')}\right] \\
&= \sum_{i\neq i'}^{\log(k/\delta)} S_{i,j}\overline{S_{i',j}} \mathop{\mathbb{E}}_{b}\left[e^{2\pi \mathbf{i}\sigma b B(i-i')}\right] \\
&= 0 \qquad \text{by Definition 11.4.8} \qquad (11.13)
\end{aligned}$$

Note that term C_1 is independent of b which means $\mathop{\mathbb{E}}_{b} C_1 = C_1$. Thus, we can remove the expectation over b. Then,

$$\begin{aligned}
\mathop{\mathbb{E}}_{\sigma,a,b}\left[\sum_{j=1}^{B} |u_j|^2\right] &= \mathop{\mathbb{E}}_{\sigma,a}\left[\sum_{j=1}^{B}\sum_{i=1}^{\log(k/\delta)} |G_{j+Bi}|^2 \cdot |g(\sigma(j+Bi-a))|^2\right] \\
&= \mathop{\mathbb{E}}_{\sigma,a}\left[\sum_{i=1}^{B\log(k/\delta)} |G_i|^2 \cdot |g(\sigma(i-a))|^2\right].
\end{aligned}$$

Now, the idea is to replace the expectation term $\mathbb{E}_a$ by an integral term $\int_{a\in A}(\star)\mathrm{d}a$. Then, replace it by another integral term $\int_0^T(\star)\mathrm{d}t$. Let A denote a set of intervals that we will sample a from. It is easy to verify that $|A| \lesssim T/\sigma$, since $(\sigma(i-a))$ is sampled from $[0,T]$. If we choose T to be a constant factor larger than $\sigma|\operatorname{supp}(G)|$, then we also have $|A| \gtrsim T/\sigma$.

$$\begin{aligned}
\mathop{\mathbb{E}}_{\sigma}\left[\mathop{\mathbb{E}}_{a}\left[\sum_{i=1}^{B\log(k/\delta)}|G_i|^2\cdot|g(\sigma(i-a))|^2\right]\right] &= \mathop{\mathbb{E}}_{\sigma}\left[\frac{1}{|A|}\int_{a\in A}\sum_{i=1}^{B\log(k/\delta)}|G_i|^2\cdot|g(\sigma(i-a))|^2\mathrm{d}a\right] \\
&= \mathop{\mathbb{E}}_{\sigma}\left[\sum_{i=1}^{B\log(k/\delta)}|G_i|^2\cdot\frac{1}{|A|}\int_{a\in A}|g(\sigma(i-a))|^2\mathrm{d}a\right] \\
&= \mathop{\mathbb{E}}_{\sigma}\left[\sum_{i=1}^{B\log(k/\delta)}|G_i|^2\cdot\frac{1}{\sigma|A|}\int_{a\in A}|g(\sigma(i-a))|^2\mathrm{d}\sigma a\right] \\
&\lesssim \mathop{\mathbb{E}}_{\sigma}\left[\|G\|_2^2\cdot\frac{1}{T}\int_0^T|g(t)|^2\mathrm{d}t\right].
\end{aligned}$$

By Claim 11.4.2, we know that $\|G\|_2^2 \eqsim \frac{1}{B}$. Combining $\sum_{j=1}^{B}|\widehat{u}_j|^2 = B\sum_{j=1}^{B}|u_j|^2$ and $\|G\|_2^2 \eqsim \frac{1}{B}$ gives the desired result. □

Lemma 11.3.4. If $g(t) = 0, \forall t \in [0,T]$. Let H denote a set of frequencies, $H = \{f \in \mathrm{supp}(\widehat{x}^*) \mid \text{neither } E_{coll}(f) \text{ nor } E_{off}(f) \text{ holds}\}$. Then,

$$\mathop{\mathbb{E}}_{\sigma,a,b}\left[\sum_{f\in H}\left|\widehat{u}_{h_{\sigma,b}(f)} - \widehat{x^*}(f)e^{a\sigma 2\pi f\mathbf{i}}\right|^2\right] \le \delta\|\widehat{x^*}\|_1^2.$$

Proof. For simplicity, let $G = G_{B,\delta,\alpha}$ and $\widehat{G'} = \widehat{G'}_{B,\delta,\alpha}$. we have

$$\begin{aligned}
\widehat{y} &= G\widehat{\cdot P_{\sigma,a,b}x} \\
&= \widehat{G} * \widehat{P_{\sigma,a,b}x} \\
&= \widehat{G'} * \widehat{P_{\sigma,a,b}x} + (\widehat{G} - \widehat{G'}) * \widehat{P_{\sigma,a,b}x}.
\end{aligned}$$

The ℓ_∞ norm of second term can be bounded :

$$\|(\widehat{G} - \widehat{G'}) * \widehat{P_{\sigma,a,b}x}\|_\infty \le \|\widehat{G} - \widehat{G'}\|_\infty\|\widehat{P_{\sigma,a,b}x}\|_1 \le \sqrt{\delta/k}\|\widehat{x^*}\|_1.$$

Thus, consider the jth term of $\widehat{u}$,

$$\begin{aligned}
\widehat{u}_j &= \widehat{y}_{jF/B} \\
&= \sum_{|l|<F/(2B)} \widehat{G'}_{-l}(\widehat{P_{\sigma,a,b}x})_{jF/B+l} \pm \sqrt{\delta/k}\|\widehat{x^*}\|_1 \\
&= \sum_{|\pi_{\sigma,b}(f)-jF/B|<F/(2B)} \widehat{G'}_{jF/B-\pi_{\sigma,b}(f)}\widehat{P_{\sigma,a,b}x}_{\pi_{\sigma,b}(f)} \pm \sqrt{\delta/k}\|\widehat{x^*}\|_1 \\
&= \sum_{h_{\sigma,b}(f)=j} \widehat{G'}_{-o_{\sigma,b}(f)}\widehat{x^*}(f)e^{2\pi f\sigma a\mathbf{i}} \pm \sqrt{\delta/k}\|\widehat{x^*}\|_1.
\end{aligned}$$

If neither $E_{coll}(f)$ nor $E_{off}(f)$ happens, then we know that frequency f is the only heavy hitter hashed into bin j and $\widehat{G'}_{-o_{\sigma,b}(f)} = 1$ for frequency f. Thus,

$$\mathop{\mathbb{E}}_{\sigma,a,b}\left[\left|\widehat{u}_{h_{\sigma,b}(f)} - \widehat{x^*}(f)e^{a\sigma 2\pi f\mathbf{i}}\right|^2\right] \leq \delta/k\|\widehat{x^*}\|_1^2.$$

Since the above equation holds for all $f \in H$, we get

$$\sum_{f\in H}\mathop{\mathbb{E}}_{\sigma,a,b}\left[\left|\widehat{u}_{h_{\sigma,b}(f)} - \widehat{x^*}(f)e^{a\sigma 2\pi f\mathbf{i}}\right|^2\right] \leq k\delta/k\|\widehat{x^*}\|_1^2 = \delta\|\widehat{x^*}\|_1^2.$$

□

11.6 Proofs for one stage of recovery

Binary search of one-sparse algorithm We first explain a simple, clean, but not optimal one-sparse algorithm, then we try to optimize the algorithm step by step. Let "heavy" frequency $f \in [-F, F]$, we can split the frequency interval into two regions: left region $[-F, 0)$ and right region $[0, F]$. We can observe $\theta \equiv 2\pi\beta f \pmod{2\pi}$ by checking the phase difference between using $P_{1,a,b}$ and $P_{1,a+\beta,b}$, where β is uniformly at random sampled from some suitable range $[\widehat{\beta}, 2\widehat{\beta}]$. For each observation θ, we can guess m different possibilities for f, say $\theta_1, \theta_2, \cdots, \theta_m$, if θ_i belong to the left region, we add a vote to that, otherwise we add a vote to the right region. After taking enough samples, we choose the region that has the largest vote. This decision will let us narrow down the searching range of the true frequency by half with some good probability. Suppose we decide to choose the right region $[0, F]$, then we can just repeat the above binary search over $[0, F]$ again to get into a region that has size $F/2$. Repeating it D times, we can learn the frequency with a region that has size at most $2F/2^D$. But the binary search is not the best approach, actually, we can do much better with using t-ary search.

k-sparse To locate those k heavy signals in the frequency domain, we need to consider the "bins" computed by HashToBins with $P_{\sigma,a,b}$. One of the main difference from previous work [HIKP12a] is, instead of permuting the discrete coordinates according to $P_{\sigma,a,b}$ and partitioning the coordinates into $B = O(k)$ bins, we permute the continuous frequency domain and partition the frequency domain into $B = O(k)$ bins. With a large constant probability, we can obtain for each heavy signal f that neither E_{coll} nor E_{off} happens. After splitting those k frequencies into different bins, then we can run the one-sparse algorithm

for all the bins simultaneously.

t-ary search To argue the final succeed probability of our algorithm, we need to take the union bound for each array/region in each round and also take the union bound over each round. There are several benefits of changing binary search to t-ary search, (1) to reach the same accuracy, the number of rounds D for t-ary search is smaller than the number of rounds for binary search; (2) our searching procedure is a "noisy" searching problem, for the noiseless version of the searching problem, we do not need to take care of the union bound argument. Having a parameter for the number of arrays/regions is important to optimize the entire procedure.

Recall the traditional binary search problem(noiseless version), given a list of sorted numbers $a[1, 2 \cdots, n]$ in increasing order. We want to determine if some number x belongs to $a[1, 2, \cdots, n]$. When we compare some $a[i]$ with x, we will know the true answer.

But the noisy binary search problem is slightly harder. $a[1, 2, \cdots, n]$ is still a list of sorted numbers in increasing order, we want to determine if some number x belongs to $a[1, 2, \cdots, n]$. But when we compare the $a[i]$ with x, we will know the true answer with $9/10$ probability, and get the false answer with $1/10$ probability. In this case, we can not finish the task by following the procedure of traditional binary search algorithm, e.g. making the decision by just comparing $a[i]$ with x once. The reason is after taking the union bound of $\log n$ rounds, the failure probability can be arbitrarily large. One idea to fix this issue is independently comparing $a[i]$ with x multiple times, e.g. $\log \log n$ times. Then we can amplify the succeed probability of each round from $9/10$ to $1 - \frac{1}{\text{poly}(\log n)}$, after taking the union bound over $\log n$ rounds, we still have $1 - \frac{1}{\text{poly}(\log n)}$ succeed probability.

Our problem is still more complicated than the above noisy binary search problem. In each round of algorithm LocateInner, we do a t-array search by splitting the candidate frequency region(that has length Δl) into t consecutive regions, $Q_1, Q_2, \cdots, Q_t$, each of them has the equal size $\frac{\Delta l}{t}$. By using the hash values of $P_{\sigma,a,b}$ and $P_{\sigma,a+\beta,b}$ (Line 26-27 in Algorithm 11.2), we can have an observation over $[0, 2\pi)$. For each such observation over $[0, 2\pi)$, it was in fact scaled by $2\pi\sigma\beta$ and rounded over $[0, 2\pi)$. Thus the corresponding frequency location of this observation might belong to $m = \Theta(\sigma\beta\Delta l)$ different possible regions. Since we do not know which one, we just add a vote to all of the possible regions. To understand t-ary search, let's consider this example. Suppose we split the frequency into 20 regions, the true frequency belongs to region 9 and each observation is correct with probability 4/5. For each observation, we will add a vote to a batch of roughly evenly spaced regions.

1. For the observation 1, we add a vote to region $1, 5, 9, 13, 17$.
2. For the observation 2, we add a vote to region $3, 9, 15$.
3. For the observation 3, we add a vote to region $2, 9, 16$.
4. For the observation 4, we add a vote to region $4, 9, 14, 19$.
5. For the observation 5, we add a vote to region $2, 6, 10, 14, 20$.

where the first four observations are the correct observations and the last one is wrong. Then the true region will have more than half of the $R_{loc} = 5$ votes with some good probability.

Details of adding vote In previous description, to let people have a better understanding of t-ary search, we simplify the step of adding vote. In fact, adding a vote to each of the m

candidate region is not enough. The right thing to do is, not only adding a vote to those m regions, but also adding a vote to a constant number c_n of neighbors of each possible region (e.g. two neighbors nearby that region, line 33 in Algorithm 11.2). The reason for doing this is, if the frequency f is located very close to the boundary of two regions, then neither of them will get more than $R_{loc}/2$ votes. After we already have R_{loc} independent observations, we just choose any region that contains more than $R_{loc}/2$ votes and enlarge it by the same constant factor c_n to be the next candidate frequency region for t-array search, and plugging the new parameters into algorithm LocateInner and running it again.

The slightly different last round Recall that, in the previous description of each round of t-ary search, we're able to decrease the searching range of frequency geometrically. To achieve this progress, we have to increase the sample duration of β is geometrically. At some point, the sample duration will reach T. Suppose the sample duration of β is geometrically increasing during the first $D-1$ rounds (Line 13-15 in Algorithm 11.2) and it becomes T after $D-1$ rounds. If we want to use the same duration T to perform one more round again, can we get some benefit for learning the frequency by doing some tricks? (1) Suppose in all the first $D-1$ rounds, we use $R_{loc}(D-1)$ samples. Then we can use $R_{loc}(D-1)$ for the last round, since it does not increase the sample complexity. This way allows us to learn frequency within $\frac{1}{CT}$. But can we do better than that? (2) Using more samples is a nice observation, another right thing to do is changing the algorithm from reporting region to reporting frequency. In the previous $D-1$ rounds, as the description of t-ary search, we just report the region that has more than $R_{loc}/2$ votes. But, we can take the median over all the values that were assigned to any region that has votes more than $R_{loc}/2$. This way

can actually allow us to learn frequency location more accurately ($\approx \frac{1}{\rho T}$) without increasing the resolution for β! Another question people might ask is, if we take median in the last round, why not just do it in every round? The answer is, in the first $D-1$ rounds, our goal is just narrowing down the search range of frequency location and we do not need to report a frequency location. Another reason is reporting a candidate region needs less samples and has higher succeed probability, but taking the median is more expensive than reporting a candidate region. We cannot pay for taking the median in every round.

To prove main Lemma 11.3.6 for one stage recovery, we introduce Lemma 11.6.1. Before explaining the proof, we give some definitions. Consider a frequency f, and define $j = h_{\sigma,b}(f)$ to be the bin that frequency f was hashed into. Define $\theta = f - b \pmod F$. Define

$$\widehat{u} = \textsc{HashToBins}(x, P_{\sigma,\gamma,b}, B, \delta, \alpha) \quad \text{and} \quad \widehat{u}' = \textsc{HashToBins}(x, P_{\sigma,\gamma+\beta,b}, B, \delta, \alpha).$$

Note that "bin" and "region" are representing different things in this paper. "bin" is related to hash function $h_{\sigma,b}(f)$. "region" is only used in the algorithm of one stage recovery in this section. Define "true" region to be the region that contains frequency f. Define "wrong" region to be the region that is not within a constant number c_n of neighbors of the "true" region. Part (I) of Lemma 11.6.1 shows that for each observation generated by a batch of samples drawn from time domain, the counter $v_{j,q'}$ corresponding to the true region will increase by one, with some "good" probability. On the other side, Part (II) of Lemma 11.6.1 shows that the counter $v_{j,q}$ corresponding to wrong region will not increase by one, with some "good" probability. Note that, [HIKP12a] proved the case when $c_n = 6$ under discrete setting, we translate it into continuous setting, here.

Lemma 11.6.1. *Given σ and b. Assume $f \in \text{region}(j, q')$ and $\mathop{\mathbb{E}}_{\gamma}[\left|\widehat{u}_j - e^{2\pi\gamma\sigma\theta\mathbf{i}}\widehat{x}(\theta)\right|^2] \leq \frac{1}{\rho^2}|\widehat{x}(\theta)|^2$. $\forall 0 < s < 1$, for each two samples $(\gamma, 0)$ and (γ, β), where γ is sampled from $[\frac{1}{2}, 1]$ uniformly at random and β is sampled from $[\frac{st}{4\sigma\Delta l}, \frac{st}{2\sigma\Delta l}]$ uniformly at random, we have*

(I) for the q', with probability at least $1 - (\frac{2}{\rho s})^2$, $v_{j,q'}$ will increase by one.

(II) for any q such that $|q - q'| > 3$, with probability at least $1 - 15s$, $v_{j,q}$ will not increase.

Proof. Part (I) of Lemma 11.6.1. We have,

$$\mathop{\mathbb{E}}_{\gamma}[\left|\widehat{u}_j - e^{2\pi\gamma\sigma\theta\mathbf{i}}\widehat{x}(\theta)\right|^2] \leq \frac{1}{\rho^2}|\widehat{x}(\theta)|^2.$$

By Chebyshev's Inequality, we have $\forall\ g > 0$, with probability $1 - g$ we have

$$\left|\widehat{u}_j - e^{2\pi\gamma\sigma\theta\mathbf{i}}\widehat{x}(\theta)\right| \leq \frac{1}{\rho}\sqrt{\frac{1}{g}}\,|\widehat{x}(\theta)|$$
$$\|\phi(\widehat{u}_j) - (\phi(\widehat{x}(\theta)) - 2\pi\gamma\sigma\theta)\|_{\bigcirc} \leq \sin^{-1}(\frac{1}{\rho}\sqrt{\frac{1}{g}}),$$

where $\|x - y\|_{\bigcirc} = \min_{z\in\mathbb{Z}}|x - y + 2\pi z|$ denote the "circular distance" between x and y. Similarly, replacing γ by $\gamma + \beta$, with probability $1 - g$, we also have that

$$\|\phi(\widehat{u}'_j) - (\phi(\widehat{x}(\theta)) - 2\pi(\gamma + \beta)\sigma\theta)\|_{\bigcirc} \leq \sin^{-1}(\frac{1}{\rho}\sqrt{\frac{1}{g}}).$$

Define $c_j = \phi(\widehat{u}_j/\widehat{u}'_j)$. Combining the above two results, with probability $1 - 2g$ we have

$$\begin{aligned}
\|c_j - 2\pi\beta\sigma\theta\|_{\bigcirc} &= \|\phi(\widehat{u}_j) - \phi(\widehat{u}'_j) - 2\pi\beta\sigma\theta\|_{\bigcirc} \\
&= \|(\phi(\widehat{u}_j) - (\phi(\widehat{x}(\theta)) - 2\pi\gamma\sigma\theta))) - (\phi(\widehat{u}'_j) - (\phi(\widehat{x}(\theta)) - 2\pi(\gamma + \beta)\sigma\theta)))\|_{\bigcirc} \\
&\leq \|\phi(\widehat{u}_j) - (\phi(\widehat{x}(\theta)) - 2\pi\gamma\sigma\theta)\|_{\bigcirc} + \|\phi(\widehat{u}'_j) - (\phi(\widehat{x}(\theta)) - 2\pi(\gamma + \beta)\sigma\theta)\|_{\bigcirc} \\
&\leq 2\sin^{-1}(\frac{1}{\rho}\sqrt{\frac{1}{g}}).
\end{aligned}$$

Here, we want to set $g = (\frac{4}{s\pi\rho})^2$, thus, with probability at least $1 - (\frac{2}{s\rho})^2$

$$\|c_j - 2\pi\beta\sigma\theta\|_{\bigcirc} < s\pi/2. \tag{11.14}$$

The above equation shows that c_j is a good estimate for $2\pi\beta\sigma\theta$ with good probability. We will now show that this means the true region $Q_{q'}$ gets a vote with large probability.

For each q' with $f \in [l_j - \frac{\Delta l}{2} + \frac{q'-1}{t}\Delta l, l_j - \frac{\Delta l}{2} + \frac{q'}{t}\Delta l] \subset [-F, F]$, we have that $m_{j,q'} = l_j - \frac{\Delta l}{2} + \frac{q'-0.5}{t}\Delta l$ and $\theta_{j,q'} = m_{j,q'} - b \pmod F$ satisfies

$$|f - m_{j,q'}| \leq \frac{\Delta l}{2t} \text{ and } |\theta - \theta_{j,q'}| \leq \frac{\Delta l}{2t}.$$

Since we sample β uniformly at random from $[\frac{st}{4\sigma\Delta l}, \frac{st}{2\sigma\Delta l}]$, then $\beta \leq \frac{st}{2\sigma\Delta l}$, which implies that $2\pi\beta\sigma\frac{\Delta l}{2t} \leq \frac{s\pi}{2}$. Thus, we can show the observation c_j is close to the true region in the following sense,

$$\begin{aligned}
&\|c_j - 2\pi\beta\sigma\theta_{j,q'}\|_{\bigcirc} \\
\leq &\|c_j - 2\pi\beta\sigma\theta\|_{\bigcirc} + \|2\pi\beta\sigma\theta - 2\pi\beta\sigma\theta_{j,q'}\|_{\bigcirc} \text{ by triangle inequality} \\
< &\frac{s\pi}{2} + 2\pi\|\beta\sigma\theta - \beta\sigma\theta_{j,q'}\|_{\bigcirc} \text{ by Equation (11.14)} \\
\leq &\frac{s\pi}{2} + 2\pi\beta\sigma\frac{\Delta l}{2t} \\
= &\frac{s\pi}{2} + \frac{s\pi}{2} \\
\leq &s\pi.
\end{aligned}$$

Thus, $v_{j,q'}$ will increase in each round with probability at least $1 - (\frac{2}{\rho s})^2$. □

Proof. Part (II) of Lemma 11.6.1.

Consider q with $|q - q'| > 3$. Then $|f - m_{j,q}| \geq \frac{7\Delta l}{2t}$, and (assuming $\beta \geq \frac{st}{4\sigma\Delta l}$) we have

$$2\pi\beta\sigma|f - m_{j,q}| \geq 2\pi\frac{st}{4\sigma\Delta l}\sigma|f - m_{j,q}| = \frac{s\pi t}{2\Delta l}|f - m_{j,q}| \geq \frac{7s\pi}{4} > \frac{3s\pi}{2}. \tag{11.15}$$

There are two cases: $|f - m_{j,q}| \leq \frac{\Delta l}{st}$ and $|f - m_{j,q}| > \frac{\Delta l}{st}$.

First, if $|f - m_{j,q}| \leq \frac{\Delta l}{st}$. In this case, from the definition of β it follows that

$$2\pi\beta\sigma|f - m_{j,q}| \leq \frac{s\pi t}{\sigma\Delta l}\sigma|f - m_{j,q}| \leq \pi. \tag{11.16}$$

Combining equations (11.15) and (11.16) implies that

$$\Pr[2\pi\beta\sigma(f - m_{j,q}) \mod 2\pi \in [-\frac{3s}{4}2\pi, \frac{3s}{4}2\pi]] = 0.$$

Second, if $|f - m_{j,q}| > \frac{\Delta l}{st}$. We show this claim is true: $\Pr[2\pi\beta\sigma(f - m_{j,q}) \pmod{2\pi} \in [-\frac{3s}{4}2\pi, \frac{3s}{4}2\pi]] \lesssim s$. To prove it, we apply Corollary 11.4.4 by setting $\widetilde{T} = 2\pi$, $\widetilde{\sigma} = 2\pi\sigma\beta$, $\widetilde{\delta} = 0$, $\widetilde{\epsilon} = \frac{3s}{4}2\pi$, $A = 2\pi\sigma\widehat{\beta}$, $\Delta f = |f - m_{j,q}|$. By upper bound of Corollary 11.4.4, the probability is at most

$$\frac{2\widetilde{\epsilon}}{\widetilde{T}} + \frac{4\widetilde{\epsilon}}{A\Delta f} = \frac{3s}{2} + \frac{3s}{\sigma\widehat{\beta}\Delta f} \leq \frac{3s}{2} + \frac{3s}{\sigma\frac{st}{4\sigma\Delta l}\frac{\Delta l}{st}} < 15s.$$

Then in either case, with probability at least $1 - 15s$, we have

$$\|2\pi\beta\sigma m_{j,q} - 2\pi\beta\sigma f\|_{\bigcirc} > \frac{3s}{4}2\pi.$$

which implies that $v_{j,q}$ will not increase. □

Lemma 11.3.6. Given σ and b, consider any frequency f for which neither $E_{coll}(f)$ nor $E_{off}(f)$ holds, and let $j = h_{\sigma,b}(f)$. Let $\mu^2(f) = \mathbb{E}_a[|\widehat{u}_j - \widehat{x^*}(f)e^{a\sigma 2\pi f\mathbf{i}}|^2]$ and $\rho^2 = |\widehat{x^*}(f)|^2/\mu^2(f)$. For sufficiently large ρ, and $\forall 0 < s < 1, t \geq 4$, consider any run of LocateInner with $f \in [l_j - \frac{\Delta l}{2}, l_j + \frac{\Delta l}{2}]$. It takes $O(R_{loc})$ random $(\gamma, \beta) \in [\frac{1}{2}, 1] \times [\frac{st}{4\sigma\Delta l}, \frac{st}{2\sigma\Delta l}]$ samples over duration $\beta\sigma = \Theta(\frac{st}{\Delta l})$, runs in $O(stR_{loc})$ time, to learn f within a region that has length $\Theta(\frac{\Delta l}{t})$ with failure probability at most $(\frac{4}{s\rho})^{R_{loc}} + t \cdot (60s)^{R_{loc}/2}$.

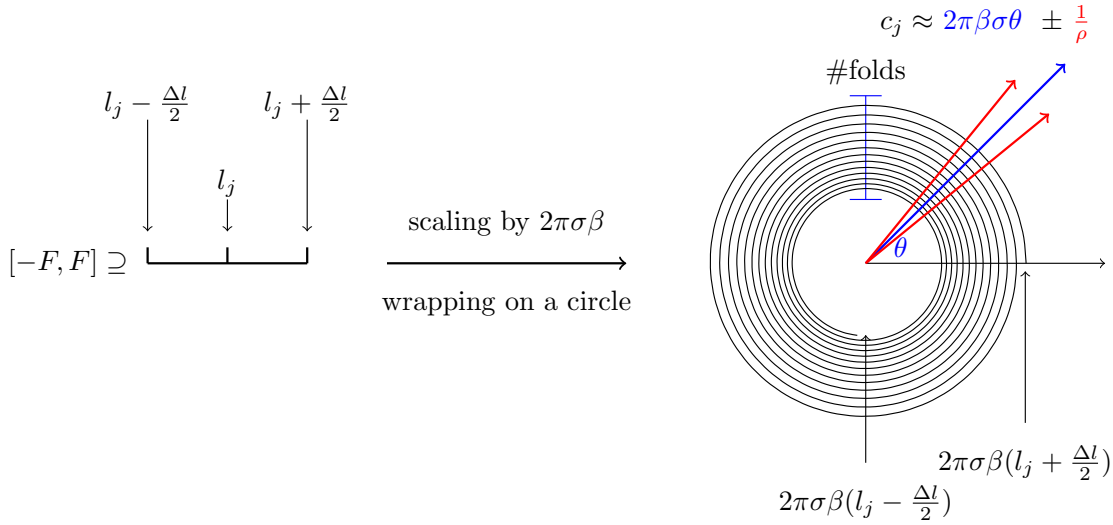


Figure 11.2: For an arbitrary frequency interval $[l_j - \frac{\Delta l}{2}, l_j + \frac{\Delta l}{2}]$, we scale it by $2\pi\sigma\beta$ to get a longer interval $[2\pi\sigma\beta(l_j + \frac{\Delta l}{2}), 2\pi\sigma\beta(l_j - \frac{\Delta l}{2})]$. Then, we wrap the longer interval on a circle $[0, 2\pi)$. The number of folds after wrapping is $\lceil \sigma\beta\Delta l \rceil$. For any random sample, the observation c_j is close to the true answer within $1/\rho$ with some "good" probability.

Proof. Let t denote the number of regions, and $[l_j - \frac{\Delta l}{2}, l_j + \frac{\Delta l}{2}]$ be the interval that contains frequency f. Let Q_q denote a region that is $[l_j - \frac{\Delta l}{2} + (q-1)\frac{\Delta l}{t}, l_j - \frac{\Delta l}{2} + q\frac{\Delta l}{t}]$. Let $\theta = f - b \pmod F$. Recall that we sample σ uniformly at random from $[\frac{1}{B\eta}, \frac{2}{B\eta}]$. Then we sample γ uniformly at random from $[\frac{1}{2}, 1]$ and sample β uniformly at random from $[\frac{st}{4\sigma\Delta l}, \frac{st}{2\sigma\Delta l}]$. Define $c_j = \phi(\widehat{u}_j/\widehat{u}'_j)$. Let m denote the number of folds, which is equal to $\lceil \sigma\beta\Delta l \rceil$. Let $v_{j,q}$ denote the vote of region(j, q).

We hope to show that in any round r, each observed c_j is close to $2\pi\sigma\beta\theta$ with good probability. On the other hand, for each observed c_j, we need to assign it to some regions and increase the vote of the corresponding region. The straightforward way is just checking all possible t regions, which takes $O(t)$ time. In fact, there are only $\Theta(m)$ regions close enough to the observation c_j, where $m = \Theta(\frac{2\pi\sigma\beta\Delta l}{2\pi}) = \Theta(st)$. The reason is $2\pi\sigma\beta$ will scale

the original length Δl interval to a new interval that has length $2\pi\sigma\beta\Delta l$. This new interval can only wrap around circle $[0, 2\pi)$ at most $\lceil \sigma\beta\Delta l \rceil$ times. The running time is $O(stR_{loc})$, since we take R_{loc} independent observations.

For each observed c_j: Part (I) of Lemma 11.6.1 says, we assign it to the true region with some good probability; Part (II) of Lemma 11.6.1 says, we do not assign it to the wrong region with some good probability. Thus taking R_{loc} independent c_j, we can analyze the failure probability of this algorithm based on these three cases.

(I) What's the probability of true fold fails?

$$
\begin{aligned}
& \Pr[\text{True fold fails}] \\
= \; & \Pr[\text{True region fails} \geq R_{loc}/2 \text{ times}] \\
\leq \; & 2 \cdot \binom{R_{loc}}{R_{loc}/2} \cdot (\Pr[\text{True region fails once}])^{R_{loc}/2} \\
\leq \; & 2 \cdot \binom{R_{loc}}{R_{loc}/2} \cdot (\frac{2}{s\rho})^{2R_{loc}/2} \quad \text{by Lemma 11.6.1} \\
\leq \; & (\frac{4}{s\rho})^{R_{loc}}.
\end{aligned}
$$

(II) What if the region that is "near"(within c_n neighbors) true region becomes true?

Any of those region gets a vote only if true region also gets a vote. Since our algorithm choosing any region that has more than $R_{loc}/2$ votes and enlarging the region size by containing c_n nearby neighbors of that chosen region, then the new larger region must contain the "real" true region.

(III) What if the region that is "far away"(not within c_n neighbors) from true region becomes true?

By Part (II) of Lemma 11.6.1, the probability of one such wrong region gets a vote is at most $15s$. Thus, one of the wrong region gets more than $R_{loc}/2$ votes is at most $(60s)^{R_{loc}/2}$. By taking the union bound over all t regions, we have the probability of existing one wrong region getting more than $R_{loc}/2$ vote is at most $t \cdot (60s)^{R_{loc}/2}$.

Thus, if we first find any region Q_q that has more than $R_{loc}/2$ votes, and report a slightly larger region $[l_j - \frac{\Delta l}{2} + (q-1)\frac{\Delta l}{t} - \frac{c_n}{2}\frac{\Delta l}{t}, l_j - \frac{\Delta l}{2} + q\frac{\Delta l}{t} + \frac{c_n}{2}\frac{\Delta l}{t}]$, it is very likely this large region contains the frequency f. Finally, the failure probability of this algorithm is at most $\Theta((\frac{4}{s\rho})^{R_{loc}} + t \cdot (60s)^{R_{loc}/2})$. □

Lemma 11.6.2. *Taking the median of values belong to any region getting at least* $\frac{1}{2}R_{loc}$ *votes, then we can learn frequency* f *within* $\Theta(\frac{\Delta l}{\rho s t})$ *with probability* $1 - \exp(-\Omega(R_{loc}))$.

Proof. Let region(j, q') be the region that getting at least $\frac{1}{2}R_{loc}$ votes. Let $R = |\text{region}(j, q')|$ denote the number of observations/votes assigned to region(j, q'). Since this region getting at least $\frac{1}{2}R_{loc}$ votes, then $\frac{1}{2}R_{loc} \leq R \leq R_{loc}$. Using Equation (11.14) in Lemma 11.6.1, $\forall g > 0$, we have

$$\|c_j - 2\pi\beta\sigma\theta\|_{\bigcirc} \leq 2 \cdot \sin^{-1}\left(\frac{1}{\rho}\sqrt{\frac{1}{g}}\right),$$

holds with probability $1 - 2g$. Choosing $g = \Theta(1)$, we have with constant success probability $p > \frac{1}{2}$,

$$\|c_j - 2\pi\beta\sigma\theta\|_{\bigcirc} \lesssim \frac{1}{\rho},$$

holds.

Taking the median over all the observations that belong to region(j, q') gives

$$
\begin{aligned}
&\Pr\left[\left|\underset{r\in\text{region}(j,q')}{\text{median}} c_j^r - 2\pi\beta^r\sigma\theta\right| \gtrsim \frac{1}{\rho}\right] \\
&< \sum_{i=R/2}^{R_{loc}} \binom{R_{loc}}{i}(1-p)^i p^{R_{loc}-i} \\
&= \sum_{i=R_{loc}/2}^{R_{loc}} \binom{R_{loc}}{i}(1-p)^i p^{R_{loc}-i} + \sum_{i=R/2}^{R_{loc}/2} \binom{R_{loc}}{i}(1-p)^i p^{R_{loc}-i} \\
&< \sum_{i=R_{loc}/2}^{R_{loc}} \binom{R_{loc}}{R_{loc}/2}(1-p)^i + \sum_{i=R/2}^{R_{loc}/2} \binom{R_{loc}}{R_{loc}/2}(1-p)^i \\
&< 2\binom{R_{loc}}{R_{loc}/2}(1-p)^{R/2} \\
&\leq 2(2e)^{R_{loc}/2}(1-p)^{R_{loc}/4} \\
&\leq e^{-cR_{loc}}, \qquad (11.17)
\end{aligned}
$$

where the the second inequality follows by $\binom{R}{i} \leq \binom{R}{R/2}$ and $p^{R_{loc}-i} < 1$, $\forall i$; the fourth inequality follows by $\binom{n}{k} \leq (ne/k)^k$; the last inequality follows by choosing some p such that $\frac{1}{2}\log_{2e}\frac{1}{1-p} - 1 > 2c$ where $c > 0$ is some constant. Equation (11.17) implies that

$$
\Pr\left(\left|\underset{r\in\text{region}(j,q')}{\text{median}} \theta^r - \theta\right| < \frac{1}{\rho 2\pi\sigma\widehat{\beta}}\right) > 1 - \exp(-\Omega(R_{loc})),
$$

where $\forall r \in [R_{loc}]$, β^r is sampled uniformly at random from $[\widehat{\beta}, 2\widehat{\beta}] = [\frac{st}{4\sigma\Delta l}, \frac{st}{2\sigma\Delta l}]$. Thus, we can learn f within $\Theta(\frac{1}{\rho 2\pi\sigma\widehat{\beta}}) = \Theta(\frac{\Delta l}{\rho st})$.

□

Lemma 11.3.7. Algorithm LocateKSignal takes $O(k\log_C(FT)\log(k/\delta))$ samples over $O(\frac{\log(k/\delta)}{\eta})$ duration, runs in $O(k\log_C(FT)\log(FT/\delta))$ time, and outputs a set $L \subset [-F, F]$ of $O(k)$ frequencies with minimum separation $\Omega(\eta)$.

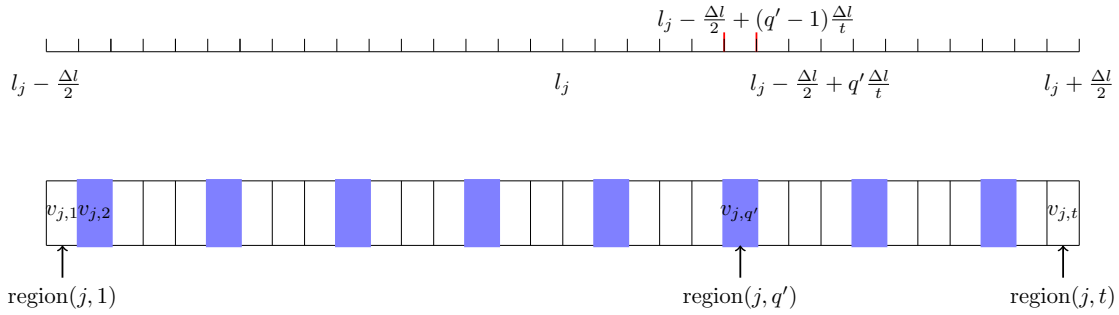


Figure 11.3: The number of "blue" regions is equal to the number of folds m. The number of total regions is t. For each observed c_j, instead of checking all the t regions, we only assign vote to the these "blue" regions. Since only these m "blue" regions can be the candidate region that contains frequency f.

Given σ and b, consider any frequency f for which neither of $E_{coll}(f)$ or $E_{off}(f)$ hold. Let $j = h_{\sigma,b}(f)$, $\mu^2(f) = \mathbb{E}_a[|\widehat{u}_j - \widehat{x^*}(f)e^{a\sigma 2\pi f \mathbf{i}}|^2]$, and $\rho^2 = |\widehat{x^*}(f)|^2/\mu^2(f)$. If $\rho > C$, then with an arbitrarily large constant probability there exists an $f' \in L$ with

$$|f - f'| \lesssim \frac{1}{T\rho}.$$

Proof. Algorithm LocateKSignal rerun procedure LocateInner D times. For the first $D-1$ rounds, the sampling range for β is increased by t every time. For the last round, the sampling range for β is not increasing any more. On the other hand, the sampling range for β for $D-1$ round and the last round are the same.

Recall that, we sample σ uniformly at random from $[\frac{1}{B\eta}, \frac{2}{B\eta}]$. Then we sample γ uniformly at random from $[\frac{1}{2}, 1]$ and β uniformly at random from $[\frac{st}{4\sigma\Delta l}, \frac{st}{2\sigma\Delta l}]$. c_n is some constant for the number of neighbor regions nearby "true" region. In the first $D-1$ rounds, we set $s = 1/\sqrt{C}$, $\Delta l = F/(t')_0^{i-1}, \forall i \in [D-1]$, $t \approx \log(FT)$, and $t' = \frac{t}{c_n+1}$. For the last round, we set $s \approx 1/C$, $\Delta l \approx st/T$ and $t \approx \log(FT)/s$. C is known as "approximation" factor.

Finally, we need to choose depth $D = \log_{t'}(FT/st) \approx \log_{t'}(FT)$ and set $R_{loc} = O(\log_C(tC))$ for all the rounds. Define $DR = \sum_{i=1}^{D} R_{loc}^i$, which is $(D-1)R_{loc} + R_{loc}^D = O(\log_C(FT))$.

After setting all the parameters for the first $D-1$ rounds and the last round, we explain some intuitions and motivations for setting the last round in a different way of the first $D-1$ rounds. For the first $D-1$ rounds, it is not acceptable to have constant failure probability for each round, since we need to take the union bound over $D-1$ rounds. But, for the last round, it is acceptable to allow just constant failure probability, since it is just a single round. That's the reason for setting C in a different way.

We have the following reason for choosing the number of regions($= t$) in the last round larger than that of first $D-1$ rounds. For the first $D-1$ rounds, we do not need to learn frequency within $\approx \frac{1}{T\rho}$. It is enough to know which region does frequency belong to, although the diameter of the region is large at the beginning. The algorithm is making progress round by round, since the diameter of each region is geometrically decreasing while $\widehat{\beta}$ is geometrically increasing. For the last round, by Lemma 11.6.2, we can learn f within $\Theta(\frac{\Delta l}{\rho st})$. Since after the last round, we hope to learn frequency within $\frac{1}{T\rho}$, thus we need to choose some s, t and Δl such that $\frac{1}{T} \approx \frac{\Delta l}{st}$. To get more accuracy result in the last round, we'd like to choose a larger t. But there is no reason to increase $\widehat{\beta}$ again, since the $\widehat{\beta}$ of the $(D-1)$th rounds can tolerance the t we choose at the last round.

To show the constant succeed probability of this Lemma, we separately consider about the failure probability of the first $D-1$ rounds and the last round. By the union bound,

the probability of existing one of the first $D-1$ rounds is failing is,

$$\begin{aligned}
& (D-1)\left((\frac{4}{s\rho})^{R_{loc}}+t\cdot(60s)^{R_{loc}/2}\right)\\
\leq\ & D\left((\frac{4}{s\rho})^{R_{loc}}+t\cdot(60s)^{R_{loc}/2}\right)\\
\leq\ & D\left((\frac{4}{sC})^{R_{loc}}+t\cdot(60s)^{R_{loc}/2}\right) \text{ by } \rho>C>1\\
=\ & D\left((\frac{4}{\sqrt{C}})^{R_{loc}}+t\cdot(\frac{60}{\sqrt{C}})^{R_{loc}/2}\right) \text{ by setting } s\approx 1/\sqrt{C}\\
\leq\ & D\cdot\frac{1}{(Ct)^c} \text{ by setting } R_{loc}=O(\log_C(tC)),
\end{aligned}$$

where c is some arbitrarily large constant. Using $t>D$, we can show that failure happening in any of the first $D-1$ rounds is small. Then, we still need to show that the probability of the last round is failing is also small,

$$\begin{aligned}
& (\frac{4}{s\rho})^{R_{loc}}+t\cdot(60s)^{R_{loc}/2}+e^{-\Theta(R_{loc})}\\
\leq\ & (\Theta(\frac{C}{\rho}))^{R_{loc}}+t\cdot(\Theta(\frac{1}{C}))^{R_{loc}/2}+e^{-\Theta(R_{loc})} \text{ by setting } s\approx 1/C\\
\leq\ & \frac{1}{c_1}+\frac{1}{(tC)^{c_2}}+e^{-\Theta(R_{loc})} \text{ by setting } R_{loc}=O(\log_C(tC))\\
\leq\ & \frac{1}{c_1}+\frac{1}{(tC)^{c_2}}+\frac{1}{c_3},
\end{aligned}$$

where in the first line, the first two terms are from Lemma 11.3.6 and the third term comes from Lemma 11.6.2; in the last line c_1, c_2 and c_3 are some arbitrarily large constants.

The expected running time includes the following part: Running HashtoBins algorithm $O(DR)$ times, each run takes $O(\frac{B}{\alpha}\log\frac{k}{\delta}+B\log B)$. Updating the counter v, which

takes $O(DR \cdot Bt)$ time. The total running time should be

$$\begin{aligned} & O(DR(\frac{B}{\alpha}\log\frac{k}{\delta} + B\log B) + (DR_{loc}Bt)) \\ = \; & O(DRB\log(k/\delta \cdot B \cdot FT)) \\ = \; & O(B\log_C(FT)\log(\frac{k}{\delta}BFT)) \\ = \; & O(B\log_C(FT)\log(FT/\delta)) \quad \text{by } FT \gg F\frac{1}{\eta} \gg k. \end{aligned}$$

The total number of samples is

$$O\left(DR \cdot B\log(\frac{k}{\delta})\right) = O(B\log_C(FT)\log(k/\delta)).$$

The sample duration of Algorithm LocateKSignal is $O(\frac{\log\frac{k}{\delta}}{\eta})$.

In conclusion, we can show that for frequency where neither E_{coll} nor E_{off} holds, we recover an f' with $|f - f'| \lesssim \frac{1}{T\rho}$ as long as $\rho > C$, with an arbitrarily large constant probability. □

Lemma 11.3.8. Algorithm OneStage takes $O(k\log_C(FT)\log(k/\delta))$ samples over $O(\frac{\log(k/\delta)}{\eta})$ duration, runs in $O(k(\log_C(FT)\log(FT/\delta)))$ time, and outputs a set of $\{(v_i', f_i')\}$ of size $O(k)$ with $\min_{i\neq j}|f_i' - f_j'| \gtrsim \eta$. Moreover, one can imagine a subset $S \subseteq [k]$ of "successful" recoveries, where $\Pr[i \in S] \geq \frac{9}{10}$ $\forall i \in [k]$ and for which there exists an injective function $\pi : [k] \to [O(k)]$ so that

$$\mathop{\mathbb{E}}_{\sigma,b}\left[\sum_{i\in S}\frac{1}{T}\int_0^T \left|v_i' e^{2\pi f_i' t\mathbf{i}} - v_{\pi(i)}e^{2\pi f_{\pi(i)}t\mathbf{i}}\right|^2 \mathrm{d}t\right] \lesssim C^2\mathcal{N}^2.$$

with $1 - 1/k^c$ probability for an arbitrarily large constant c.

Proof. Let H denote the set of frequencies f for which neither $E_{coll}(f)$ nor $E_{off}(f)$ holds. For each such f, let $v = \widehat{x^*}(f)$ and denote

$$\mu^2(f) = \mathop{\mathbb{E}}_{a}[|\widehat{u}_j - ve^{a\sigma 2\pi f\mathbf{i}}|^2] \tag{11.18}$$

and $\rho^2(f) = |v|^2/\mu^2(f)$, as in Lemma 11.3.7. We have that, if $\rho^2(f) > C^2$, then with an arbitrarily large constant probability one of the recovered $f' \in L$ has $|f' - f| \lesssim \frac{1}{T\rho}$. If this happens, then OneStage will estimate v using $v' = \widehat{u}_j e^{-a\sigma 2\pi f'\mathbf{i}}$. By triangle inequality,

$$|v' - v|^2 \lesssim |v|^2|e^{a2\pi\sigma(f'-f)\mathbf{i}} - 1|^2 + |\widehat{u}_j - ve^{a\sigma 2\pi f\mathbf{i}}|^2. \tag{11.19}$$

Since $a\sigma \leq T$ and $|f' - f| \lesssim \frac{1}{T\rho}$, then the first term of RHS of Equation (11.19) have

$$|v|^2|e^{a2\pi\sigma(f'-f)\mathbf{i}} - 1|^2 \lesssim |v|^2|a\sigma(f' - f)|^2.$$

For the second term of RHS of Equation (11.19). Using Equation (11.18), we have

$$|\widehat{u}_j - ve^{a\sigma 2\pi f\mathbf{i}}|^2 \lesssim \mu^2(f),$$

with arbitrarily large constant probability. Combining the bounds for those two terms gives

$$|v' - v|^2 \lesssim |v|^2|a\sigma(f' - f)|^2 + \mu^2(f),$$

with arbitrarily large constant probability. Since $a\sigma \leq T$, the first term is $|v|^2/\rho^2 = \mu^2$, for

$$|v' - v|^2 \lesssim \mu^2(f).$$

On the other hand, if $\rho^2(f) < C^2$, then $|v| = \rho(f)\mu(f) \lesssim C\mu(f)$ so regardless of the frequency f' recovered, the estimate v' will have

$$|v' - v|^2 \lesssim C^2\mu^2(f).$$

with arbitrarily large constant probability.

Combining with Lemma 11.3.1, we get for any $f \in H$ that the recovered f', v' will have

$$\frac{1}{T}\int_0^T \left|v'e^{2\pi f' t\mathbf{i}} - ve^{2\pi f t\mathbf{i}}\right|^2 \mathrm{d}t \lesssim C^2\mu^2(f).$$

with arbitrarily large constant probability. Let $S \subset H$ be the set of frequencies for which this happens. We can choose our permutation π to match frequencies in S to their nearest approximation. By Lemma 11.3.2, this means that

$$\mathop{\mathbb{E}}_{\sigma,b}\left[\sum_{i\in S}\frac{1}{T}\int_0^T \left|v_i'e^{2\pi f_i' t\mathbf{i}} - v_{\pi(i)}e^{2\pi f_{\pi(i)} t\mathbf{i}}\right|^2 \mathrm{d}t\right] \lesssim C^2\mathcal{N}^2.$$

as desired. □

11.7 Proofs for combining multiple stages

We first prove that the median of a bunch of estimates of a frequency has small error if most of the estimates have small error.

Lemma 11.7.1. *Let (v_i, f_i) be a set of tones for $i \in S$. Define v' and f' to be the (coordinate-wise) median of the (v_i, f_i). Then for any (v^*, f^*) we have*

$$\frac{1}{T}\int_0^T \left|v^* e^{2\pi f^* t\mathbf{i}} - v' e^{2\pi f' t\mathbf{i}}\right|^2 \, \mathrm{d}t \lesssim \operatorname*{median}_i \frac{1}{T}\int_0^T \left|v^* e^{2\pi f^* t\mathbf{i}} - v_i e^{2\pi f_i t\mathbf{i}}\right|^2 \mathrm{d}t.$$

Proof. By Lemma 11.3.1 we have that

$$\frac{1}{T}\int_0^T \left|v^* e^{2\pi f^* t\mathbf{i}} - v' e^{2\pi f' t\mathbf{i}}\right|^2 \, \mathrm{d}t \eqsim |v^*|^2 \min(1, T^2|f^* - f'|^2) + |v^* - v'|^2.$$

Using that v' is taken as a two dimensional median, it suffices to show: if $x^{(1)}, x^{(2)}, \ldots \in \mathbb{R}^3$ then $x' = \operatorname*{median}_i x^{(i)}$ has

$$\|x'\|_2^2 \lesssim \operatorname*{median}_i \left\|x^{(i)}\right\|_2^2. \tag{11.20}$$

This follows because in each of the three coordinates j, we have

$$(x'_j)^2 = (\operatorname*{median}_i x_j^{(i)})^2 \le \operatorname*{median}_i (x_j^{(i)})^2 \le \operatorname*{median}_i \left\|x^{(i)}\right\|_2^2.$$

so summing over the three coordinates gives (11.20), as desired.

Therefore, for two dimensional median and one dimension median, we have

$$|v^* - v'|^2 \lesssim \operatorname*{median}_i |v^* - v_i|^2 \tag{11.21}$$

and

$$|f^* - f'|^2 \lesssim \operatorname*{median}_i |f^* - f_i|^2.$$

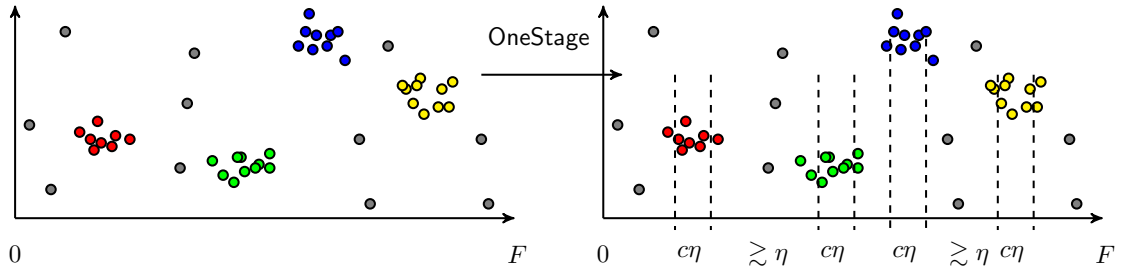


Figure 11.4: We demonstrate the algorithm for merging various stages($R = O(\log k)$) on 2-dimensional data. Note that the true data should be 3-dimensional, since for each tone (v_i, f_i), $v_i \in \mathbb{C}$ and $f_i \in \mathbb{R}$. The x-axis represents the frequency and the y-axis represents the real part of magnitude.

Moreover,

$$\begin{aligned} |v^*|^2 \cdot \min(1, T^2|f^* - f'|^2) &\lesssim |v^*|^2 \cdot \min(1, T^2 \operatorname*{median}_i |f^* - f_i|^2) \\ &= |v^*|^2 \cdot \min(1, \operatorname*{median}_i T^2|f^* - f_i|^2) \\ &= |v^*|^2 \cdot \operatorname*{median}_i \min(1, T^2|f^* - f_i|^2) \end{aligned} \tag{11.22}$$

Combining Equation (11.21) and (11.22), we have

$$\begin{aligned} & |v^* - v'|^2 + |v^*|^2 \cdot \min(1, T^2|f^* - f'|^2) \\ \lesssim\ & \operatorname*{median}_i |v^* - v_i|^2 + \operatorname*{median}_i |v^*|^2 \cdot \min(1, T^2|f^* - f_i|^2) \\ =\ & \operatorname*{median}_i |v^* - v_i|^2 + |v^*|^2 \cdot \min(1, T^2|f^* - f_i|^2). \end{aligned}$$

Thus, we complete the proof. □

Lemma 11.3.9. Repeating algorithm OneStage $O(\log k)$ times, MergedStages returns a set $\{(v'_i, f'_i)\}$ of size $O(k)$ with $\min_{i \neq j} |f'_i - f'_j| \gtrsim \eta$ that can be indexed by π such that

$$\sum_{i=1}^{k} \frac{1}{T} \int_0^T \left| v'_i e^{2\pi f'_i t \mathbf{i}} - v_{\pi(i)} e^{2\pi f_{\pi(i)} t \mathbf{i}} \right|^2 \mathrm{d}t \lesssim C^2 \mathcal{N}^2.$$

with probability $1 - 1/k^c$ for an arbitrarily large constant c.

Proof. Our only goal here is to recover the actual tones well, not to worry about spurious tones.

Suppose the number of stages we perform is $R = O(\log k)$. The algorithm for merging the various stages is to scan over a $c\eta$ size region for small c, and take the median (in both frequency and magnitude) over $3c\eta$ region around that $c\eta$ if there are at least $\frac{6}{10}R$ results in that $c\eta$ region. If so, the algorithm will jump to the first right point that is at least η far away from current region and look for the next $c\eta$ region. Because the minimum separation between frequencies for a given stage is $\Omega(\eta)$, this will have minimum separation η in the output, and because there are $O(k)$ tones output at each stage so will this method. What remains is to show that the total error is small.

We say a stage is "good" if the term inside the expectation of Lemma 11.3.8 is less than 10 times its expectation, as happens with 9/10 probability:

$$\Pr\left[\sum_{i\in S}\frac{1}{T}\int_0^T \left|v_i' e^{2\pi f_i' t\mathbf{i}} - v_{\pi(i)} e^{2\pi f_{\pi(i)} t\mathbf{i}}\right|^2 \mathrm{d}t \le 10 \mathop{\mathbb{E}}_{\sigma,b}\left[\sum_{i\in S}\frac{1}{T}\int_0^T \left|v_i' e^{2\pi f_i' t\mathbf{i}} - v_{\pi(i)} e^{2\pi f_{\pi(i)} t\mathbf{i}}\right|^2 \mathrm{d}t\right]\right] \ge 9/10.$$

We say a frequency f_i is "successful" in a given stage if the stage is good and i lies in S for that stage. Therefore a frequency is successful in each stage with at least 8/10 probability. For sufficiently large $R = O(\log k)$, this will cause all k frequencies to be successful more than $\frac{7}{10}R$ times with high probability. Suppose this happens.

Let $\mu^2(f_i)$ denote the error of (v_i, f_i). By Lemma 11.3.8, the total error over all good stages and every successful recovery of a tone in a good stage is $O(C^2\mathcal{N}^2R)$. We define $\mu^2(f)$ to be the $6/10R$ worst amount of error in the recovery of f over all stages. Because there are $R/10$ worse successful recoveries for each f, we have that $\sum_i \mu^2(f_i) \lesssim C^2\mathcal{N}^2$.

We will match each f_i with a recovered frequency f'_i with cost at most $\mu^2(f_i)$. If $\mu^2(f_i) \gtrsim |v_i|^2$, then we can set an arbitrary f'_i with $v'_i = 0$. Otherwise, more that $\frac{6}{10}R$ successful recoveries of f_i also yield f' that are within $O(\frac{1}{T}) \ll c\eta$ of f_i. Thus the algorithm for merging tones will find enough tones to report something. What it reports will be the median of at most R values, 6/10 of which have less error than $\mu^2(f)$. Therefore by Lemma 11.7.1 the reported frequency and magnitude will have error $O(\mu^2(f))$. This suffices to get the result. □

Lemma 11.3.10. If we run MERGEDSTAGES twice and take the tones $\{(v'_i, f'_i)\}$ from the first result that have f'_i within $c\eta$ for small c of some frequency in the second result, we get a set of $k'' = O(k)$ tones that can be indexed by some permutation π such that

$$\sum_{i=1}^{k} \frac{1}{T} \int_0^T \left| v'_i e^{2\pi f'_i t \mathbf{i}} - v_{\pi(i)} e^{2\pi f_{\pi(i)} t \mathbf{i}} \right|^2 \mathrm{d}t + \sum_{i=k+1}^{k''} |v'_i|^2 \lesssim C^2 \mathcal{N}^2. \tag{11.5}$$

Proof. By Lemma 11.3.9 and Lemma 11.3.1, the first run gives us a set of $k' = O(k)$ pairs $\{(v'_i, f'_i)\}$ such that they may be indexed by using permutation π such that the first k are a good approximation to $\{(v_i, f_i)\}$ in the sense that

$$\sum_{i=1}^{k} (|v_{\pi(i)}|^2 + |v'_i|^2) \cdot \min(1, T^2 |f_{\pi(i)} - f'_i|^2) + |v_{\pi(i)} - v'_i|^2 \lesssim C^2 \mathcal{N}^2.$$

We may as well index to match tones to their nearest match in frequency (because the separation for both f'_i and f_i is at least $\Omega(\eta) > 1/T$). That is, we may index such that $|f'_j - f_i| \gtrsim \eta$ for any $j \neq i$.

Now, for indices where $|f'_i - f_{\pi(i)}| > 1/T$, one would do better by setting the corresponding v_i to zero. In particular, suppose you knew the true frequencies $f_{\pi(i)}$ and only took

the (v_i', f_i') where f_i' is close to $f_{\pi(i)}$. That is, there exists some permutation π, for the subset $S \subseteq [k]$ containing $\{i : |f_{\pi(i)} - f_i'| \leq c/T\}$ for any $c \gtrsim 1$, we have

$$\sum_{i \in S}((|v_{\pi(i)}|^2 + |v_i'|^2) \cdot \min(1, T^2|f_{\pi(i)} - f_i'|^2) + |v_{\pi(i)} - v_i'|^2) + \sum_{i \in [k] \setminus S} (|v_{\pi(i)}|^2 + |v_i'|^2) \lesssim C^2\mathcal{N}^2. \tag{11.23}$$

The problem is that we do not know the set S, so we can not throw out the other frequencies. However, we can fake it by running the algorithm again on the signal. In particular, we apply Lemma 11.3.9 again to sparse recovery of the signal defined by

$$\{(v_i, f_i) \mid i \in [k]\} \cup \{(0, f_i') \mid i \in \{k+1, \ldots, k'\}\}.$$

That is, we pretend the "signal" has terms at the other f_i' for $k < i \leq k'$, but with magnitude zero. This is an identical signal in time domain, so it's really just an analytical tool; for the analysis, it has $k' = O(k)$ sparsity and $\Omega(\eta)$ separation, so Lemma 11.3.9 applies again and gets a set of $k'' = O(k)$ pairs $\{(v_j'', f_j'')\}$ such that, for the subset $S^* \subset [k']$ containing the indices i where $|f_j'' - f_i'| \leq c/T$ for some $j \in [k'']$. In other words, there exists some permutation τ such that for the subset $S^* \subset [k']$ containing $\{i : |f''_{\tau(i)} - f_i'| \leq c/T\}$. We have

by analogy to (11.23) that

$$\begin{aligned}
C^2\mathcal{N}^2 \gtrsim & \sum_{i\in S^*\cap[k]} ((|v_{\pi(i)}|^2+|v''_{\tau(i)}|^2)\cdot\min(1,T^2|f_{\pi(i)}-f''_{\tau(i)}|^2)+|v_{\pi(i)}-v''_{\tau(i)}|^2)\\
+ & \sum_{i\in S^*\setminus[k]} ((|0|^2+|v''_{\tau(i)}|^2)\cdot\min(1,T^2|f'_i-f''_{\tau(i)}|^2)+|v''_{\tau(i)}-0|^2)\\
+ & \sum_{i\in[k]\setminus S^*} (|v_{\pi(i)}|^2+|v''_{\tau(i)}|^2)+\sum_{i\in[k']\setminus S^*\setminus[k]} (|v''_{\tau(i)}|^2)\\
\gtrsim & \sum_{i\in S^*\cap[k]} ((|v_{\pi(i)}|^2+|v''_{\tau(i)}|^2)\cdot\min(1,T^2|f_{\pi(i)}-f''_{\tau(i)}|^2)+|v_{\pi(i)}-v''_{\tau(i)}|^2)\\
+ & \sum_{i\in S^*\setminus[k]} |v''_{\tau(i)}|^2+\sum_{i\in[k]\setminus S^*} |v_{\pi(i)}|^2,
\end{aligned}$$

where π and τ are two permutations and $|S^*|=k^*=O(k)$. This last term is precisely the desired total error for the set of tones $\{(v''_{\tau(i)},f''_{\tau(i)}):i\in S^*\}$, giving the result.

□

To prove Theorem 11.3.11, we still need the following "local" Lemma.

Lemma 11.7.2. *For any three tones $(v_{\pi(i)},f_{\pi(i)})$, (v_i^*,f_i^*) and (v'_i,f'_i), if $|v'_i|\geq|v_i^*|$ then,*

$$\begin{aligned}
& (|v'_i|^2+|v_{\pi(i)}|^2)\cdot\min(1,T^2|f'_i-f_{\pi(i)}|^2)+|v'_i-v_{\pi(i)}|^2\\
\lesssim\ & (|v_i^*|^2+|v_{\pi(i)}|^2)\cdot\min(1,T^2|f_i^*-f_{\pi(i)}|^2)+|v_i^*-v_{\pi(i)}|^2+|v'_i|^2.
\end{aligned}$$

Proof. First, we can show an upper bound for LHS. Using inequality $\min(1,T^2|f'_i-f_{\pi(i)}|^2)\leq 1$,

$$\text{LHS}\leq|v'_i|^2+|v_{\pi(i)}|^2+|v'_i-v_{\pi(i)}|^2.$$

By triangle inequality,

$$|v_i' - v_{\pi(i)}|^2 \leq 2|v_i'|^2 + 2|v_{\pi(i)}|^2.$$

Thus, we obtain,

$$\text{LHS} \lesssim |v_i'|^2 + |v_{\pi(i)}|^2.$$

Second, we can show a lower bound for RHS. Since first term of RHS is nonnegative, then

$$\text{RHS} \geq |v_i^* - v_{\pi(i)}|^2 + |v_i'|^2.$$

Using $|v_i'| \geq |v_i^*|$, we have

$$\text{RHS} \gtrsim |v_i'|^2 + 2|v_i^*|^2 + 2|v_i^* - v_{\pi(i)}|^2.$$

By triangle inequality,

$$2|v_i^*|^2 + 2|v_i^* - v_{\pi(i)}|^2 \geq |v_{\pi(i)}|^2.$$

Then, we prove the lower bound for RHS,

$$\text{RHS} \gtrsim |v_i'|^2 + |v_{\pi(i)}|^2.$$

Combining the lower bound of RHS and the upper bound of LHS completes the proof.

□

By plugging Lemma 11.3.1 into Lemma 11.7.2, we have

Corollary 11.7.3. *For any three tones $(v_{\pi(i)}, f_{\pi(i)})$, (v_i^*, f_i^*) and (v_i', f_i'), if $|v_i'| \geq |v_i^*|$, then*

$$\frac{1}{T}\int_0^T \left|v_i' e^{2\pi f_i' t\mathbf{i}} - v_{\pi(i)} e^{2\pi f_{\pi(i)} t\mathbf{i}}\right|^2 \mathrm{d}t \lesssim \frac{1}{T}\int_0^T \left|v_i^* e^{2\pi f_i^* t\mathbf{i}} - v_{\pi(i)} e^{2\pi f_{\pi(i)} t\mathbf{i}}\right|^2 \mathrm{d}t + |v_i'|^2.$$

We use Corollary 11.7.3 to present the proof of Theorem 11.3.11,

Theorem 11.3.11. Algorithm ContinuousFourierSparseRecovery returns a set $\{(v_i', f_i')\}$ of size k with $\min_{i \neq j} |f_i' - f_j'| \gtrsim \eta$ for which

$$\sum_{i=1}^{k} \frac{1}{T} \int_0^T \left| v_i' e^{2\pi f_i' t \mathbf{i}} - v_{\pi(i)} e^{2\pi f_{\pi(i)} t \mathbf{i}} \right|^2 \mathrm{d}t \lesssim C^2 \mathcal{N}^2$$

with probability $1 - 1/k^c$ for an arbitrarily large constant c.

Proof. Let $\{(v_i^*, f_i^*)\}_{i=1,2,\cdots,k''}$ denote the set of tones returned by Lemma 11.3.10, where $k'' = O(k)$ and $k'' > k$. For any $i \in [k]$, tone (v_i^*, f_i^*) was mapped to tone $(v_{\pi(i)}, f_{\pi(i)})$ in Lemma 11.3.10. Let $\{(v_i', f_i')\}_{i=1,2,\cdots,k}$ denote a subset of $\{(v_i^*, f_i^*)\}_{i=1,2,\cdots,k''}$ that satisfies the following two conditions (1) for any $i \in [k]$, $|v_i'|$ is one of the top-k largest magnitude tones; (2) for any $i \in [k]$, $|v_i'| \geq |v_i^*|$. By Lemma 11.3.10, we also know that $\min_{i \neq j} |f_i' - f_j'| \gtrsim \eta$.

For any $i \in [k]$, we consider these three tones $(v_{\pi(i)}, f_{\pi(i)}), (v_i^*, f_i^*), (v_i', f_i')$. If $(v_i^*, f_i^*) \neq (v_i', f_i')$, then applying Corollary 11.7.3 we have

$$\frac{1}{T} \int_0^T \left| v_i' e^{2\pi f_i' t \mathbf{i}} - v_{\pi(i)} e^{2\pi f_{\pi(i)} t \mathbf{i}} \right|^2 \mathrm{d}t \lesssim \frac{1}{T} \int_0^T \left| v_i^* e^{2\pi f_i^* t \mathbf{i}} - v_{\pi(i)} e^{2\pi f_{\pi(i)} t \mathbf{i}} \right|^2 \mathrm{d}t + |v_i'|^2.$$

Otherwise $(v_i^*, f_i^*) = (v_i', f_i')$, we also have

$$\frac{1}{T} \int_0^T \left| v_i' e^{2\pi f_i' t \mathbf{i}} - v_{\pi(i)} e^{2\pi f_{\pi(i)} t \mathbf{i}} \right|^2 \mathrm{d}t \quad = \quad \frac{1}{T} \int_0^T \left| v_i^* e^{2\pi f_i^* t \mathbf{i}} - v_{\pi(i)} e^{2\pi f_{\pi(i)} t \mathbf{i}} \right|^2 \mathrm{d}t,$$

which means there exists some universal constant α such that $\forall i \in [k]$, if $(v_i^*, f_i^*) = (v_i', f_i')$ then

$$\frac{1}{T} \int_0^T \left| v_i' e^{2\pi f_i' t \mathbf{i}} - v_{\pi(i)} e^{2\pi f_{\pi(i)} t \mathbf{i}} \right|^2 \mathrm{d}t \leq \alpha \cdot \left(\frac{1}{T} \int_0^T \left| v_i^* e^{2\pi f_i^* t \mathbf{i}} - v_{\pi(i)} e^{2\pi f_{\pi(i)} t \mathbf{i}} \right|^2 \mathrm{d}t + |v_i'|^2 \right)$$

holds. Otherwise $(v_i^*, f_i^*) \neq (v_i', f_i')$, then

$$\frac{1}{T}\int_0^T \left|v_i' e^{2\pi f_i' t\mathbf{i}} - v_{\pi(i)} e^{2\pi f_{\pi(i)} t\mathbf{i}}\right|^2 \mathrm{d}t \;=\; \alpha \frac{1}{T}\int_0^T \left|v_i^* e^{2\pi f_i^* t\mathbf{i}} - v_{\pi(i)} e^{2\pi f_{\pi(i)} t\mathbf{i}}\right|^2 \mathrm{d}t$$

holds. Let S denote the set of indices i such that $(v_i^*, f_i^*) \neq (v_i', f_i')$. Taking the summation from $i = 1$ to $i = k$,

$$\begin{aligned}
&\sum_{i=1}^{k} \frac{1}{T}\int_0^T \left|v_i' e^{2\pi f_i' t\mathbf{i}} - v_{\pi(i)} e^{2\pi f_{\pi(i)} t\mathbf{i}}\right|^2 \mathrm{d}t \\
\leq\; &\sum_{i\in S} \alpha \cdot \left(\frac{1}{T}\int_0^T \left|v_i^* e^{2\pi f_i^* t\mathbf{i}} - v_{\pi(i)} e^{2\pi f_{\pi(i)} t\mathbf{i}}\right|^2 \mathrm{d}t + |v_i'|^2\right) \\
+\; &\sum_{i\in[k]\setminus S} \alpha \cdot \left(\frac{1}{T}\int_0^T \left|v_i^* e^{2\pi f_i^* t\mathbf{i}} - v_{\pi(i)} e^{2\pi f_{\pi(i)} t\mathbf{i}}\right|^2 \mathrm{d}t\right),
\end{aligned}$$

furthermore, we have

$$\begin{aligned}
\sum_{i=1}^{k} \frac{1}{T}\int_0^T \left|v_i' e^{2\pi f_i' t\mathbf{i}} - v_{\pi(i)} e^{2\pi f_{\pi(i)} t\mathbf{i}}\right|^2 \mathrm{d}t \leq\; &\alpha \sum_{i=1}^{k} \left(\frac{1}{T}\int_0^T \left|v_i^* e^{2\pi f_i^* t\mathbf{i}} - v_{\pi(i)} e^{2\pi f_{\pi(i)} t\mathbf{i}}\right|^2 \mathrm{d}t\right) \\
&+ \alpha \sum_{i\in S} |v_i'|^2.
\end{aligned}$$

To finish the proof, we need to show that $\sum_{i\in S} |v_i'|^2 \leq \sum_{i=k+1}^{k''} |v_i^*|^2$. The point is, for any $i \in S$, we know that $(v_i', f_i') \neq (v_i^*, f_i^*)$ which implies that $(v_i', f_i') \notin \{(v_i^*, f_i^*)\}_{i=1}^{k}$. Thus,

$$\begin{aligned}
&\sum_{i=1}^{k} \frac{1}{T}\int_0^T \left|v_i' e^{2\pi f_i' t\mathbf{i}} - v_{\pi(i)} e^{2\pi f_{\pi(i)} t\mathbf{i}}\right|^2 \mathrm{d}t \\
\leq\; &\alpha \sum_{i=1}^{k} \left(\frac{1}{T}\int_0^T \left|v_i^* e^{2\pi f_i^* t\mathbf{i}} - v_{\pi(i)} e^{2\pi f_{\pi(i)} t\mathbf{i}}\right|^2 \mathrm{d}t\right) + \alpha \sum_{i=k+1}^{k''} |v_i^*|^2 \\
\lesssim\; &\sum_{i=1}^{k} \left(\frac{1}{T}\int_0^T \left|v_i^* e^{2\pi f_i^* t\mathbf{i}} - v_{\pi(i)} e^{2\pi f_{\pi(i)} t\mathbf{i}}\right|^2 \mathrm{d}t\right) + \sum_{i=k+1}^{k''} |v_i^*|^2 \\
\lesssim\; &C^2\mathcal{N}^2,
\end{aligned}$$

where $k'' = O(k)$ is defined in Lemma 11.3.10 and the last inequality follows by using Equation (11.5) in Lemma 11.3.10.

□

11.8 Proofs for converting (11.2) into (11.3)

In this section, we show that as long as the sample duration T is sufficiently large, it is possible to convert Equation (11.2) to Equation (11.3). First, we show an auxiliary lemma, Lemma 11.8.3, which bounds an integral that will appear in the analysis.

We will show that

$$\int_{-\infty}^{+\infty} \min(T, \frac{1}{|f_i - \theta|}) \cdot \min(T, \frac{1}{|f_j - \theta|}) \mathrm{d}\theta \lesssim \frac{\log(T|f_i - f_j|)}{|f_i - f_j|}.$$

for $f_j - f_i \geq 2/T$. We split this into two pieces.

Claim 11.8.1. *Given two frequencies f_i, f_j and $f_j - f_i \geq \frac{2}{T}$, we have*

$$\int_{f_i - \frac{1}{T}}^{f_i} \min(T, \frac{1}{|f_i - \theta|}) \cdot \frac{1}{|f_j - \theta|} \mathrm{d}\theta \lesssim \frac{1}{f_j - f_i}.$$

Proof. By $f_i - \frac{1}{T} < \theta < f_i$, we have

$$\text{LHS} = \int_{f_i - \frac{1}{T}}^{f_i} T \cdot \frac{1}{f_j - \theta} \mathrm{d}\theta.$$

Since $\frac{1}{f_j - \theta} \eqsim \frac{1}{f_j - f_i}$ for all $\theta \in [f_i - \frac{1}{T}, f_i]$,

$$\text{LHS} \lesssim \int_{f_i - \frac{1}{T}}^{f_i} \frac{T}{f_j - f_i} \mathrm{d}\theta = \frac{1}{f_j - f_i}.$$

□

Claim 11.8.2. *Given two frequencies f_i, f_j and $f_j - f_i \geq \frac{2}{T}$, we have*

$$\int_{-\infty}^{f_i - \frac{1}{T}} \min(T, \frac{1}{|f_i - \theta|}) \cdot \frac{1}{|f_j - \theta|} \mathrm{d}\theta \lesssim \frac{\log(T|f_j - f_i|)}{f_j - f_i}.$$

Proof. By $\theta < f_i - \frac{1}{T} < f_j$, we have that

$$\begin{aligned}
\text{LHS} &= \int_{-\infty}^{f_i - \frac{1}{T}} \frac{1}{f_i - \theta} \cdot \frac{1}{f_j - \theta} \mathrm{d}\theta \\
&= \frac{1}{f_j - f_i} \int_{-\infty}^{f_i - \frac{1}{T}} \frac{f_j - f_i}{(f_i - \theta)(f_j - \theta)} \mathrm{d}\theta \\
&= \frac{1}{f_j - f_i} \int_{-\infty}^{f_i - \frac{1}{T}} \frac{1}{f_i - \theta} - \frac{1}{f_j - \theta} \mathrm{d}\theta \\
&= -\frac{1}{f_j - f_i} \log \frac{f_i - f_i + \frac{1}{T}}{f_j - f_i + \frac{1}{T}} \\
&= -\frac{1}{f_j - f_i} \log \frac{1}{T(f_j - f_i) + 1} \\
&\lesssim \frac{\log(T(f_j - f_i))}{f_j - f_i}.
\end{aligned}$$

□

Lemma 11.8.3. *Given two frequencies f_i, f_j and $f_j - f_i \geq \frac{2}{T}$, we have*

$$\int_{-\infty}^{+\infty} \min(T, \frac{1}{|f_i - \theta|}) \cdot \min(T, \frac{1}{|f_j - \theta|}) \mathrm{d}\theta \lesssim \frac{\log(T|f_i - f_j|)}{|f_i - f_j|}.$$

Proof. By symmetry, we have

$$\text{LHS} = 2 \int_{-\infty}^{\frac{f_i + f_j}{2}} \min(T, \frac{1}{|f_i - \theta|}) \cdot \min(T, \frac{1}{|f_j - \theta|}) \mathrm{d}\theta.$$

Since $T > \frac{1}{f_j - \theta}$ when $\theta < \frac{f_i + f_j}{2}$,

$$\text{LHS} \leq 2 \int_{-\infty}^{\frac{f_i + f_j}{2}} \min(T, \frac{1}{|f_i - \theta|}) \cdot \frac{1}{|f_j - \theta|} \mathrm{d}\theta.$$

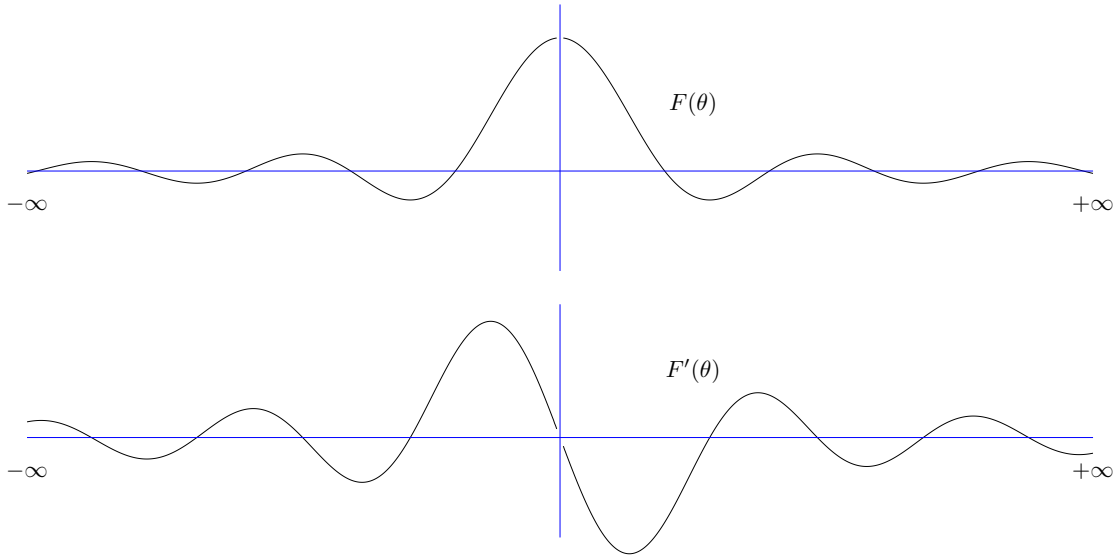


Figure 11.5: $F(\theta)$ is a sinc function and has derivate function $F'(\theta)$.

We also observe that $\frac{1}{|f_j-\theta|} \approx \frac{1}{|f_j-f_i|}$ for all $\theta \in [f_i - \frac{f_j-f_i}{2}, f_i + \frac{f_j-f_i}{2}]$,

$$\int_{f_i-\frac{f_j-f_i}{2}}^{\frac{f_j+f_i}{2}} \min(T, \frac{1}{|f_i-\theta|}) \cdot \frac{1}{|f_j-\theta|} \mathrm{d}\theta \approx \int_{f_i-\frac{f_j-f_i}{2}}^{f_i} \min(T, \frac{1}{|f_i-\theta|}) \cdot \frac{1}{|f_j-\theta|} \mathrm{d}\theta.$$

Thus, we get

$$\text{LHS} \lesssim \int_{-\infty}^{f_i} \min(T, \frac{1}{|f_i-\theta|}) \cdot \frac{1}{|f_j-\theta|} \mathrm{d}\theta.$$

Plugging Claim 11.8.1 and 11.8.2 into the above formula completes the proof. □

Lemma 11.8.4. *For any i, let $a_i(t) = v_i e^{2\pi f_i t \mathbf{i}} - v_i' e^{2\pi f_i' t \mathbf{i}}$, then for $i \neq j$,*

$$\frac{1}{T}\int_0^T a_i(t)\overline{a_j(t)}\mathrm{d}t \lesssim \frac{\log(\Delta f_{i,j} T)}{\Delta f_{i,j} T} \cdot \left(\frac{1}{T}\int_0^T |a_i(t)|^2 \mathrm{d}t \cdot \frac{1}{T}\int_0^T |a_j(t)|^2 \mathrm{d}t\right)^{\frac{1}{2}}, \tag{11.24}$$

where $\Delta f_{i,j} = \min(|f_i - f_j|, |f_i - f_j'|, |f_i' - f_j|, |f_i - f_j'|)$.

Proof. Let $\nu_i = |f_i - f_i'|$ and $\nu_j = |f_j - f_j'|$. Define $\|a_i\| = \left(\frac{1}{T}\int_0^T |a_i(t)|^2 \mathrm{d}t\right)^{1/2}$. We define $f(t)$ and $F(\theta)$ to be a rectangle and sinc function respectively:

$$f(t) = \begin{cases} 1 & \text{if } 0 \le t \le T \\ 0 & \text{otherwise} \end{cases}$$

$$F(\theta) = \frac{\sin(2\pi\theta T)}{2\pi\theta}$$

where $F = \widehat{f}$.

$F(\theta)$ has the derivative,

$$F'(\theta) = \frac{2\pi\theta T\cos(2\pi\theta T) - \sin(2\pi\theta T)}{2\pi\theta^2},$$

which means that

$$|F'(\theta)| \lesssim \begin{cases} T^2 & \text{if } \theta \le 1/T \\ T/|\theta| & \text{otherwise} \end{cases}$$

Let $y_i(t) = a_i(t) \cdot f(t)$, then

$$\begin{aligned} \widehat{y_i}(\theta) &= \widehat{a_i}(\theta) * \widehat{f}(\theta) \\ &= \widehat{a_i}(\theta) * F(\theta) \text{ by } F = \widehat{f} \\ &= v_i F(f_i - \theta) - v_i' F(f_i' - \theta) \\ &= (v_i - v_i')F(f_i - \theta) + v_i'(f_i - f_i') \cdot F'(x - \theta) \text{ some } x \in [f_i, f_i']. \end{aligned}$$

We split into two cases. First, if $\nu_i \le \frac{1}{T}$, then

$$\begin{aligned} |\widehat{y_i}(\theta)| &\lesssim (|v_i - v_i'| + \nu_i T|v_i'|) \cdot \min(T, \frac{1}{|f_i - \theta|}) \\ &\lesssim \|a_i\| \cdot \min(T, \frac{1}{|f_i - \theta|}) \quad \text{by Lemma 11.3.1,} \end{aligned} \tag{11.25}$$

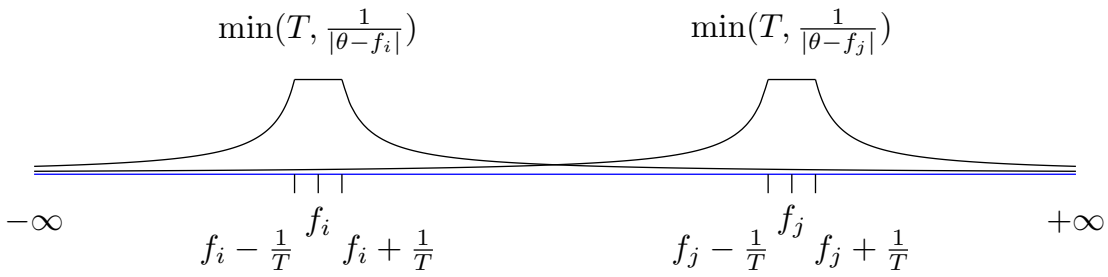


Figure 11.6: $\int_{-\infty}^{+\infty} \min(T, \frac{1}{|\theta-f_i|}) \cdot \min(T, \frac{1}{|\theta-f_j|}) \mathrm{d}\theta$

where the first step holds for both $f_i > f_i'$ and $f_i \leq f_i'$ since the triangle inequality.

Therefore

$$\begin{aligned}
\frac{1}{T}\int_0^T a_i(t)\overline{a_j(t)}\mathrm{d}t &= \frac{1}{T}\int_{-\infty}^{\infty} y_i(t)\overline{y_j(t)}\mathrm{d}t \\
&= \frac{1}{T}\int_{-\infty}^{\infty} \widehat{y}_i(\theta)\overline{\widehat{y}_j(\theta)}\mathrm{d}\theta \\
&\lesssim \frac{1}{T}\|a_i\|\|a_j\| \underbrace{\int_{-\infty}^{+\infty} \min(T, \frac{1}{|f_i-\theta|}) \cdot \min(T, \frac{1}{|f_j-\theta|})\mathrm{d}\theta}_{C}, \quad (11.26)
\end{aligned}$$

where the first step follows from $y_i(t) = a_i(t) \cdot f(t)$ and $f(t) = 1$ if $t \in [0, T]$, the second step follows from the property of Fourier transform, and the last step follows from Equation (11.25).

Using Lemma 11.8.3, we have following bound for term C,

$$\int_{-\infty}^{+\infty} \min(T, \frac{1}{|f_i-\theta|}) \cdot \min(T, \frac{1}{|f_j-\theta|})\mathrm{d}\theta \lesssim \frac{\log T|f_j - f_i|}{|f_j - f_i|}.$$

This gives the result for $\nu_i \leq \frac{1}{T}$. In the alternate case, we have $\nu_i > \frac{1}{T}$, then

$$\begin{aligned}
|\widehat{y}_i(\theta)| &\lesssim v_i \cdot \min(T, \frac{1}{|f_i - \theta|}) + v_i' \cdot \min(T, \frac{1}{|f_i' - \theta|}) \\
&\lesssim \|a_i\| \cdot \left(\min(T, \frac{1}{|f_i - \theta|}) + \min(T, \frac{1}{|f_i' - \theta|}) \right) \quad \text{by Lemma 11.3.1.}
\end{aligned}$$

By similar reason for Equation (11.26), we have

$$\begin{aligned}
& \frac{1}{T} \int_0^T a_i(t) \overline{a_j(t)} \mathrm{d}t \\
\lesssim\ & \frac{1}{T} \|a_i\| \|a_j\| \underbrace{\int_{-\infty}^{+\infty} \min(T, \frac{1}{|f_i - \theta|}) \cdot \min(T, \frac{1}{|f_j - \theta|}) \mathrm{d}\theta}_{C_1} \\
+\ & \frac{1}{T} \|a_i\| \|a_j\| \underbrace{\int_{-\infty}^{+\infty} \min(T, \frac{1}{|f_i - \theta|}) \cdot \min(T, \frac{1}{|f_j' - \theta|}) \mathrm{d}\theta}_{C_2} \\
+\ & \frac{1}{T} \|a_i\| \|a_j\| \underbrace{\int_{-\infty}^{+\infty} \min(T, \frac{1}{|f_i' - \theta|}) \cdot \min(T, \frac{1}{|f_j - \theta|}) \mathrm{d}\theta}_{C_3} \\
+\ & \frac{1}{T} \|a_i\| \|a_j\| \underbrace{\int_{-\infty}^{+\infty} \min(T, \frac{1}{|f_i' - \theta|}) \cdot \min(T, \frac{1}{|f_j' - \theta|}) \mathrm{d}\theta}_{C_4}.
\end{aligned}$$

Applying Lemma 11.8.3 on the term C_1, C_2, C_3 and C_4 respectively,

$$\frac{1}{T} \int_0^T a_i(t) \overline{a_j(t)} \lesssim \|a_i\| \|a_j\| \frac{\log(T \Delta f_{i,j})}{\Delta f_{i,j}},$$

where $\Delta f_{i,j} = \min(|f_i - f_j|, |f_i - f_j'|, |f_i' - f_j|, |f_i' - f_j'|)$.

□

Lemma 11.8.5. *Let $\{(v_i, f_i)\}$ and $\{(v'_i, f'_i)\}$ be two sets of k tones for which $\min_{i\neq j} |f_i - f_j| \geq \eta$ and $\min_{i\neq j} |f'_i - f'_j| \geq \eta$ for some $\eta > 0$. Suppose that $T > C/\eta$ for a sufficiently large constant C. Then these sets can be indexed such that*

$$\frac{1}{T}\int_0^T \left|\sum_{i=1}^{k}(v'_i e^{2\pi \mathbf{i} f'_i t} - v_i e^{2\pi \mathbf{i} f_i t})\right|^2 \mathrm{d}t \leq \alpha \cdot \sum_{i=1}^{k} \frac{1}{T}\int_0^T \left|v'_i e^{2\pi \mathbf{i} f'_i t} - v_i e^{2\pi \mathbf{i} f_i t}\right|^2 \mathrm{d}t, \tag{11.27}$$

where

$$\alpha = \left(1 + O\left(\frac{\log(k\eta T)\log(k)}{\eta T}\right)\right).$$

Proof. For simplicity, let $a_i(t) = v_i e^{2\pi f_i t \mathbf{i}} - v'_i e^{2\pi f'_i t \mathbf{i}}$. Let's express the square of summations by diagonal term and off-diagonal term, and then bound them separately.

$$\begin{aligned}
&\int_0^T \left|\sum_{i=1}^{k} a_i(t)\right|^2 \mathrm{d}t \\
=\ &\int_0^T \left(\sum_{i=1}^{k} a_i(t)\right)\left(\sum_{i=1}^{k} \overline{a_i(t)}\right) \mathrm{d}t \\
=\ &\int_0^T \sum_{i=1}^{k} \underbrace{a_i(t)\overline{a_i(t)}}_{\text{diagonal}} + \sum_{i\neq j}^{k} \underbrace{a_i(t)\overline{a_j(t)}}_{\text{off-diagonal}} \mathrm{d}t.
\end{aligned} \tag{11.28}$$

Using the result of Lemma 11.8.4 and $a^2 + b^2 \geq 2ab$, we can upper bound the off-diagonal term,

$$\begin{aligned}
&\int_0^T a_i(t)\overline{a_j(t)}\mathrm{d}t \\
\lesssim\ &\frac{\log(\Delta f_{i,j} T)}{\Delta f_{i,j} T} \cdot \left(\int_0^T |a_i(t)|^2 \mathrm{d}t \int_0^T |a_j(t)|^2 \mathrm{d}t\right)^{1/2} && \text{by Lemma 11.8.4} \\
\lesssim\ &\frac{\log(\Delta f_{i,j} T)}{\Delta f_{i,j} T} \cdot \left(\int_0^T |a_i(t)|^2 \mathrm{d}t + \int_0^T |a_j(t)|^2 \mathrm{d}t\right). && \text{by } 2ab \leq a^2 + b^2
\end{aligned}$$

where $\Delta f_{i,j} = \min(|f_i - f_j|, |f_i - f'_j|, |f'_i - f_j|, |f'_i - f'_j|)$. If we index such that any f_i is matched with any f'_j with $|f_i - f'_j| < \eta/3$ – which is possible, since at most one such f'_j will exist by the separation among the f'_j, and that f'_j will be within $\eta/3$ of at most on f_i – then we have $\Delta f_{i,j} \gtrsim |f_i - f_j|$. If we order the f_i in increasing order, then in fact $\Delta f_{i,j} \gtrsim \eta|i-j|$.

If $T > C/\eta$ for a sufficiently large constant C, this means that $\Delta f_{i,j} T \gtrsim |i-j|\eta T \geq e$. Since $\frac{\log x}{x}$ is decreasing on the region, this implies

$$\frac{\log(\Delta f_{i,j} T)}{\Delta f_{i,j} T} \lesssim \frac{\log(|i-j|\eta T)}{|i-j|\eta T}.$$

Thus, we have

$$\int_0^T a_i(t)\overline{a_j(t)}\mathrm{d}t \lesssim \frac{\log(|i-j|\eta T)}{|i-j|\eta T} \cdot \left(\int_0^T |a_i(t)|^2 \mathrm{d}t + \int_0^T |a_j(t)|^2 \mathrm{d}t \right) \quad . \quad (11.29)$$

Finally, we have

$$
\begin{aligned}
&\int_0^T \left|\sum_{i=1}^k a_i(t)\right|^2 \mathrm{d}t - \sum_{i=1}^k \int_0^T |a_i(t)|^2 \mathrm{d}t \\
&= \sum_{i\neq j}^k \int_0^T a_i(t)\overline{a_j(t)}\mathrm{d}t && \text{by Equation (11.28)} \\
&\lesssim \sum_{i\neq j}^k \frac{\log(|i-j|\eta T)}{|i-j|\eta T} \cdot \left(\int_0^T |a_i(t)|^2\mathrm{d}t + \int_0^T |a_j(t)|^2\mathrm{d}t\right) && \text{by Equation (11.29)} \\
&\leq \frac{\log(k\eta T)}{\eta T} \sum_{i=1}^k \sum_{j\neq i}^k \frac{1}{|i-j|} \cdot \left(\int_0^T |a_i(t)|^2\mathrm{d}t + \int_0^T |a_j(t)|^2\mathrm{d}t\right) \\
&= 2\frac{\log(k\eta T)}{\eta T} \sum_{i=1}^k \sum_{j\neq i}^k \frac{1}{|i-j|} \int_0^T |a_i(t)|^2\mathrm{d}t && \text{by symmetry} \\
&\lesssim \frac{\log(k\eta T)}{\eta T} \sum_{i=1}^k \int_0^T |a_i(t)|^2\mathrm{d}t \sum_{j\neq i}^k \frac{1}{|i-j|} \\
&\lesssim \frac{\log(k\eta T)\log(k)}{\eta T} \sum_{i=1}^k \int_0^T |a_i(t)|^2\mathrm{d}t \quad \text{by } \sum_{i=1}^k \frac{1}{i} \eqsim \log(k)\ .
\end{aligned}
$$

Thus, we complete the proof. □

Lemma 11.3.13. Let $\{(v_i, f_i)\}$ and $\{(v'_i, f'_i)\}$ be two sets of k tones for which $\min_{i\neq j} |f_i - f_j| \geq \eta$ and $\min_{i\neq j} |f'_i - f'_j| \gtrsim \eta$ for some $\eta > 0$. Suppose that $T > O(\frac{\log^2 k}{\eta})$. Then these sets can be indexed such that

$$
\frac{1}{T}\int_0^T \left|\sum_{i=1}^k (v'_i e^{2\pi \mathbf{i} f'_i t} - v_i e^{2\pi \mathbf{i} f_i t})\right|^2 \mathrm{d}t \lesssim \sum_{i=1}^k \frac{1}{T}\int_0^T \left|v'_i e^{2\pi \mathbf{i} f'_i t} - v_i e^{2\pi \mathbf{i} f_i t}\right|^2 \mathrm{d}t. \tag{11.30}
$$

Proof. Directly follows by Lemma 11.8.5. □

11.9 Lower Bound

Lemma 11.3.15. There exists a constant $c > 0$ such that, for a given sample duration T, one cannot recover the frequency f to within

$$c\frac{\mathcal{N}}{T|\widehat{x}^*(f)|}$$

with 3/4 probability, for all $\delta > 0$, even if $k = 1$.

Proof. Suppose this were possible, and consider two one-sparse signals y and y' containing tones (v, f) and (v, f'), respectively. By Lemma 11.3.1,

$$\int_0^T |y(t) - y(t)|^2 \mathrm{d}t \lesssim |v|^2 T^2 |f - f'|^2.$$

Consider recovery of the signal $x(t) = y(t)$, and suppose it outputs some frequency f^*. This must simultaneously be a good recovery for the decomposition $(x^*, g) = (y, 0)$ and $(x^*, g) = (y', y - y')$. These have noise levels $\mathcal{N}^2$ bounded by $\delta|v|^2$ and $\delta|v|^2 + O(|v|^2 T^2 |f - f'|^2)$, respectively. By the assumption of good recovery, and the triangle inequality, we require

$$c \cdot \frac{2\sqrt{\delta|v|^2} + \sqrt{O(|v|^2 T^2 |f - f'|^2)}}{Tv} \gtrsim |f - f'|$$

or

$$c \cdot O\left(\left(\frac{\delta}{T|f - f'|}\right)^{1/2} + 1\right) \geq 1.$$

Because δ may be chosen arbitrarily small, we can choose a small constant c such that this is a contradiction. □

Algorithm 11.1 Continuous Fourier Sparse Recovery - Part 1

procedure ContinuousFourierSparseRecovery($x, k, \delta, \alpha, C, F, T, \eta$) — The Main Algorithm
    F is the upper bound of frequency, T is the sample duration, C is the approximation factor.
    δ, α are the parameters associated with Hash function.
    $c = 1/10$ and $b = 8/10$
    $R_1 \leftarrow$ NoisyKSparseCFFT$(x, k, \delta, \alpha, C, F)$
    $S_1 \leftarrow$ MergedStages$(R_1, O(k \log k), \eta, c, b)$
    $R_2 \leftarrow$ NoisyKSparseCFFT$(x, O(k), \delta, \alpha, C, F, T)$
    $S_2 \leftarrow$ MergedStages$(R_2, O(k \log k), \Omega(\eta), c, b)$
    $S \leftarrow S_1 \cap S_2$, which means only keeping the tones that S_1 agrees with S_2 by Lemma 11.3.10.
    $S^* \leftarrow$ Prune(S, k), which means only keeping the top-k largest magnitude tones.
    return S^*
end procedure

Algorithm 11.2 Continuous Fourier Sparse Recovery - Part 2

procedure NoisyKSparseCFFT($x, k, \delta, \alpha, C, F, T$)
  Let $B = k/\epsilon$.
  for $c = 1 \to \log(k)$ **do**
    Choose σ uniformly at random from $[\frac{1}{B\eta}, \frac{2}{B\eta}]$.
    Choose b uniformly at random from $[0, \frac{2\pi\lceil F/\eta \rceil}{\sigma B}]$.
    $R_c \leftarrow$ OneStage($x, B, \delta, \alpha, \sigma, b, C, F, T$)
  end for
  return $(R_1, R_2, \cdots, R_{\log(k)})$.
end procedure
procedure MergedStages(R, m, η, c, b)
  R is a list of m tones (v'_i, f'_i)
  c is some constant < 1.
  b is some constant < 1.
  Sort list R based on f'_i.
  Building the 1D range search Tree based on m points by regarding each frequency f'_i as a 1D point on a line where $x_i = f'_i$.
  $S \leftarrow \emptyset$, $i \leftarrow 0$
  while $i < m$ **do**
    if Tree.Count$(f'_i, f'_i + c\eta) \geq b \log k$ **then**
      $f \leftarrow$ median $\{ f'_j \mid f'_j \in [f'_i - c\eta, f'_i + 2c\eta]\}$
      $v \leftarrow$ median $\{ v'_j \mid f'_j \in [f'_i - c\eta, f'_i + 2c\eta]\}$
      $S \leftarrow S \cup (f, v)$
      $i \leftarrow$ Tree.Search$(f'_i + 2c\eta + \eta/2)$, which means walk to the first point that is on the right of $f'_i + 2c\eta + \eta/2$
    else
      $i \leftarrow i + 1$
    end if
  end while
  return S
end procedure

Algorithm 11.3 Continuous Fourier Sparse Recovery - Part 3

procedure HASHTOBINS($x, P_{\sigma,a,b}, B, \delta, \alpha$)
  Compute $\widehat{y}_{jF/B}$ for $j \in [B]$, where $y = G_{B,\alpha,\delta} \cdot (P_{\sigma,a,b}x)$
  return $\widehat{u}$ given by $\widehat{u}_j = \widehat{y}_{jF/B}$
end procedure
procedure ONESTAGE($x, B, \delta, \alpha, \sigma, b, C, F, T$)
  $L \leftarrow$ LOCATEKSIGNAL($x, B, \delta, \alpha, \sigma, b, C, F, T$)
  Choose $a \in [0, 1]$ uniformly at random.
  $\widehat{u} \leftarrow$ HASHTOBINS($x, P_{\sigma,a,b}, B, \delta, \alpha$)
  return $\{(\widehat{u}_{h_{\sigma,b}(f')} e^{-2\pi\sigma a f' \mathbf{i}}, f')$ for $f' \in L$ if not $E_{off}(f')\}$.
end procedure
procedure LOCATEKSIGNAL($x, B, \delta, \alpha, \sigma, b, C, F, T$)
  Set $t \approx \log(FT)$, $t' = t/(c_n + 1)$, $D \approx \log_{t'}(FT)$, $R_{loc} \approx \log_C(tC)$, $l^{(1)} = F/2$.
  for $i \in [D - 1]$ **do**
    $\Delta l \approx F/(t')^{i-1}$, $s = \frac{1}{\sqrt{C}}$, $\widehat{\beta} = \frac{ts}{2\sigma\Delta l}$
    $l^{(i+1)} \leftarrow$ LOCATEINNER($x, B, \delta, \alpha, \sigma, b, \widehat{\beta}, l^{(i)}, \Delta l, t, R_{loc}$, **false**).
  end for
  Set $s = 1/C$, $t \approx \log(FT)/s$, $\Delta l \approx st/T$, $\widehat{\beta} = \frac{ts}{2\sigma\Delta l}$, $R_{loc} \approx \log_C(tC)$
  $l^{(*)} \leftarrow$ LOCATEINNER($x, B, \delta, \alpha, \sigma, b, \widehat{\beta}, l^{(D)}, \Delta l, t, R_{loc}$, **true**).
  return $l^{(*)}$.
end procedure

Algorithm 11.4 Continuous Fourier Sparse Recovery - Part 4

procedure LocateInner($x, B, \delta, \sigma, b, \widehat{\beta}, l, \Delta l, t, R_{loc}, last$) ▷ Lemma 11.3.6
  Let $v_{j,q} = 0$ for $(j, q) \in [B] \times [t]$.
  for $r \in [R_{loc}]$ **do**
    Choose $\gamma \in [\frac{1}{2}, 1]$ uniformly at random.
    Choose $\beta \in [\frac{1}{2}\widehat{\beta}, 1\widehat{\beta}]$ uniformly at random.
    $\widehat{u} \leftarrow$ HashToBins$(x, P_{\sigma,\gamma,b}, B, \delta, \alpha)$.
    $\widehat{u'} \leftarrow$ HashToBins$(x, P_{\sigma,\gamma+\beta,b}, B, \delta, \alpha)$.
    for $j \in [B]$ **do**
      for $i \in [m]$ **do**
        $\theta^r_{j,i} = \frac{1}{2\pi\sigma\beta}(\phi(\widehat{u}_j/\widehat{u'}_j) + 2\pi s_i), s_i \in [\sigma\beta(l_j - \Delta l/2), \sigma\beta(l_j + \Delta l/2)] \cap \mathbb{Z}_+$
        $f^r_{j,i} = \theta^r_{j,i} + b \pmod F$
        suppose $f^r_{j,i}$ belongs to region(j, q),
        add a vote to both region(j, q) and two neighbors nearby that region, e.g. region$(j, q-1)$ and region$(j, q+1)$
      end for
    end for
  end for
  for $j \in [B]$ **do**
    $q^*_j \leftarrow \{q | v_{j,q} > \frac{R_{loc}}{2}\}$
    if $last$ = **true then**
      $l^*_j \leftarrow \text{median}\{f^r_{j,i} | f^r_{j,i} \in \text{region}(j, q^*_j), i \in [f], r \in [R_{loc}]\}$
    else
      $l^*_j \leftarrow$ center of region(j, q^*_j)
    end if
  end for
  return l^*
end procedure

Chapter 12

Continuous Fourier Transform II

We consider the problem of estimating a Fourier-sparse signal from noisy samples, where the sampling is done over some interval $[0, T]$ and the frequencies can be "off-grid". Previous methods for this problem required the gap between frequencies to be above $1/T$, the threshold required to robustly identify individual frequencies. We show the frequency gap is not necessary to estimate the signal as a whole: for arbitrary k-Fourier-sparse signals under ℓ_2 bounded noise, we show how to estimate the signal with a constant factor growth of the noise and sample complexity polynomial in k and logarithmic in the bandwidth and signal-to-noise ratio.

As a special case, we get an algorithm to interpolate degree d polynomials from noisy measurements, using $O(d)$ samples and increasing the noise by a constant factor in ℓ_2.

12.1 Introduction

In an interpolation problem, one can observe $x(t) = x^*(t) + g(t)$, where $x^*(t)$ is a structured signal and $g(t)$ denotes noise, at points t_i of one's choice in some interval $[0, T]$. The goal is to recover an estimate $\widetilde{x}$ of x^* (or of x). Because we can sample over a particular interval, we would like our approximation to be good on that interval, so for any function $y(t)$ we define

$$\|y\|_T^2 = \frac{1}{T}\int_0^T |y(t)|^2 \mathrm{d}t.$$

to be the ℓ_2 error on the sample interval. For some parameters C and δ, we would then like to get

$$\|\widetilde{x} - x^*\|_T \leq C\,\|g\|_T + \delta\,\|x^*\|_T \tag{12.1}$$

while minimizing the number of samples and running time. Typically, we would like C to be $O(1)$ and to have δ be very small (either zero, or exponentially small). Note that, if we do not care about changing C by $O(1)$, then by the triangle inequality it doesn't matter whether we want to estimate x^* or x (i.e. we could replace the LHS of (12.1) by $\|\widetilde{x} - x\|_T$).

Of course, to solve an interpolation problem one also needs x^* to have structure. One common form of structure is that x^* have a sparse Fourier representation. We say that a function x^* is k-Fourier-sparse if it can be expressed as a sum of k complex exponentials:

$$x^*(t) = \sum_{j=1}^{k} v_j e^{2\pi \mathbf{i} f_j t}.$$

for some $v_j \in \mathbb{C}$ and $f_j \in [-F, F]$, where F is the "bandlimit". Given F, T, and k, how many samples must we take for the interpolation (12.1)?

If we ignore sparsity and just use the bandlimit, then Nyquist sampling and Shannon-Whittaker interpolation uses $FT + 1/\delta$ samples to achieve (12.1). Alternatively, in the absence of noise, x^* can be found from $O(k)$ samples by a variety of methods, including Prony's method from 1795 or Reed-Solomon syndrome decoding [Mas69], but these methods are not robust to noise.

If the signal is periodic with period T—i.e., the frequencies are multiples of $1/T$—then we can use sparse discrete Fourier transform methods, which take $O(k \log^c(FT/\delta))$ time and samples (e.g. [GGI+02, HIKP12a, IKP14]). If the frequencies are not multiples of $1/T$ (are "off the grid"), then the discrete approximation is only k/δ sparse, making the interpolation less efficient; and even this requires that the frequencies be well separated.

A variety of algorithms have been designed to recover off-grid frequencies directly, but they require the minimum gap among the frequencies to be above some threshold. With frequency gap at least $1/T$, we can achieve a k^c approximation factor using $O(FT)$ samples [Moi15], and with gap above $O(\log^2 k)/T$ we can get a constant approximation using $O(k \log^c(FT/\delta))$ samples and time [PS15].

Having a dependence on the frequency gap is natural. If two frequencies are very close together—significantly below $1/T$—then the corresponding complex exponentials will be close on $[0, T]$, and hard to distinguish in the presence of noise. In fact, from a lower bound in [Moi15], below $1/T$ frequency gap one cannot recover the frequencies in the presence of noise as small as $2^{-\Omega(k)}$. The lower bound proceeds by constructing two signals using significantly different frequencies that are exponentially close over $[0, T]$.

But if two signals are so close, do we need to distinguish them? Such a lower bound

doesn't apply to the interpolation problem, it just says that you can't solve it by finding the frequencies. Our question becomes: can we benefit from Fourier sparsity in a regime where we can't recover the individual frequencies?

We answer in the affirmative, giving an algorithm for the interpolation using

$$\mathrm{poly}(k \log(FT/\delta))$$

samples. Our main theorem is the following:

Theorem 12.1.1. Let $x(t) = x^*(t) + g(t)$, where x^* is k-Fourier-sparse signal with frequencies in $[-F, F]$. Given samples of x over $[0, T]$ we can output $\widetilde{x}(t)$ such that with probability at least $1 - 2^{-\Omega(k)}$,

$$\|\widetilde{x} - x^*\|_T \lesssim \|g\|_T + \delta \|x^*\|_T .$$

Our algorithm uses $\mathrm{poly}(k, \log(1/\delta)) \cdot \log(FT)$ samples and $\mathrm{poly}(k, \log(1/\delta)) \cdot \log^2(FT)$ time. The output $\widetilde{x}$ is $\mathrm{poly}(k, \log(1/\delta))$-Fourier-sparse signal.

Relative to previous work, this result avoids the need for a frequency gap, but loses a polynomial factor in the sample complexity and time. We lose polynomial factors in a number of places; some of these are for ease of exposition, but others are challenging to avoid.

Degree d polynomials are the special case of d-Fourier-sparse functions in the limit of $f_j \to 0$, by a Taylor expansion. This is a regime with no frequency gap, so previous sparse Fourier results would not apply but Theorem 12.1.1 shows that $\mathrm{poly}(d \log(1/\delta))$ samples suffices. In fact, in this special case we can get a better polynomial bound:

Theorem 12.1.2. For any degree d polynomial $P(t)$ and an arbitrary function $g(t)$, Procedure RobustPolynomialLearning in Algorithm 12.5 takes $O(d)$ samples from $x(t) = P(t) + g(t)$ over $[0, T]$ and reports a degree d polynomial $Q(t)$ in time $O(d^\omega)$ such that, with probability at least 99/100,

$$\|P(t) - Q(t)\|_T^2 \lesssim \|g(t)\|_T^2.$$

where $\omega < 2.373$ is matrix multiplication exponent [Str69],[CW87],[Wil12].

We also show how to reduce the failure probability to an arbitrary $p > 0$ with $O(\log(1/p))$ independent repetitions, in Theorem 12.4.5.

Although we have not seen such a result stated in the literature, our method is quite similar to one used in [CDL13]. Since d samples are necessary to interpolate a polynomial without noise, the result is within constant factors of optimal.

One could apply Theorem 12.1.2 to approximate other functions that are well approximated by polynomials or piecewise polynomials. For example, a Gaussian of standard deviation at least σ can be approximated by a polynomial of degree $O((\frac{T}{\sigma})^2 + \log(1/\delta))$; hence the same bound applies as the sample complexity of improper interpolation of a positive mixture of Gaussians.

12.1.1 Related work

Sparse discrete Fourier transforms. There is a large literature on sparse discrete Fourier transforms. Results generally are divided into two categories: one category of results that carefully choose measurements that allow for sublinear recovery time, including [GGI+02, GMS05, HIKP12b, Iwe13, HIKP12a, IK14, IKP14, Kap16]. The other category of results expect randomly chosen measurements and show that a generic recovery algorithm such as ℓ_1 minimization will work with high probability; these results often focus on proving the Restricted Isometry Property [CRT06b, RV08, Bou14, HR16]. At the moment, the first category of results have better theoretical sample complexity and running time, while results in the second category have better failure probabilities and empirical performance. Our result falls in the first category. The best results here can achieve $O(k \log n)$ samples [IK14], $O(k \log^2 n)$ time [HIKP12b], or within $\log \log n$ factors of both [IKP14].

For signals that are not periodic, the discrete Fourier transform will not be sparse: it takes k/δ frequencies to capture a $1 - \delta$ fraction of the energy. To get a better dependence on δ, one has to consider frequencies "off the grid", i.e. that are not multiples of $1/T$.

Off the grid. Finding the frequencies of a signal with sparse Fourier transform off the grid has been a question of extensive study. The first algorithm was by Prony in 1795, which worked in the noiseless setting. This was refined by classical algorithms like MUSIC [Sch81] and ESPRIT [RPK86], which empirically work better with noise. Matrix pencil [BM86] is a method for computing the maximum likelihood signal under Gaussian noise and evenly spaced samples. The question remained how accurate the maximum likelihood estimate is; [Moi15] showed that it has an $O(k^c)$ approximation factor if the frequency gap is at least

$1/T$.

Now, the above results all use FT samples, which is analogous to n in the discrete setting. This can be decreased down till $O(k)$ by only looking at a subset of time, i.e. decreasing T; but doing so increases the frequency gap needed for decent robustness results.

A variety of works have studied how to adapt sparse Fourier techniques from the discrete setting to get sublinear sample complexity; they all rely on the minimum separation among the frequencies to be at least c/T for $c \geq 1$. [TBSR13] showed that a convex program can recover the frequencies exactly in the noiseless setting, for $c \geq 4$. This was improved in [CF14] to $c \geq 2$ for complex signals and $c \geq 1.87$ for real signals. [CF14] also gave a result for $c \geq 2$ that was stable to noise, but this required the signal frequencies to be placed on a finely spaced grid. [YX15] gave a different convex relaxation that empirically requires smaller c in the noiseless setting. [DB13] used model-based compressed sensing when $c = \Omega(1)$, again without theoretical noise stability. Note that, in the noiseless setting, exact recovery can be achieved without any frequency separation using Prony's method or Berlekamp-Massey syndrome decoding [Mas69]; the benefit of the above results is that a convex program might be robust to noise, even if it has not been proven to be so.

In the noisy setting, [FL12] gave an extension of Orthogonal Matching Pursuit (OMP) that can recover signals when $c = \Omega(k)$, with an approximation factor $O(k)$, and a few other assumptions. Similarly, [BCG+12] gave a method that required $c = \Omega(k)$ and was robust to certain kinds of noise. [HK15] got the threshold down to $c = O(1)$, in multiple dimensions, but with approximation factor $O(FTk^{O(1)})$.

[TBR15] shows that, under Gaussian noise and with separation $c \geq 4$, a semidefinite

program can optimally estimate $x^*(t_i)$ at evenly spaced sample points t_i from observations $x^*(t_i) + g(t_i)$. This is somewhat analogous to our setting, the differences being that (a) we want to estimate the signal over the entire interval, not just the sampled points, (b) our noise g is adversarial, so we cannot hope to reduce it—if g is also k-Fourier-sparse, we cannot distinguish x^* and g, and of course (c) we want to avoid requiring frequency separation.

In [PS15], we gave the first algorithm with $O(1)$ approximation factor, finding the frequencies when $c \gtrsim \log(1/\delta)$, and the signal when $c \gtrsim \log(1/\delta) + \log^2 k$.

Now, all of the above results algorithms are designed to recover the frequencies; some of the ones in the noisy setting then show that this yields a good approximation to the overall signal (in the noiseless setting this is trivial). Such an approach necessitates $c \geq 1$: [Moi15] gave a lower bound, showing that any algorithm finding the frequencies with approximation factor $2^{o(k)}$ must require $c \geq 1$.

Thus, in the current literature, we go from not knowing how to get any approximation for $c < 1$, to getting a polynomial approximation at $c = 1$ and a constant approximation at $c \gtrsim \log^2 k$. In this work, we show how to get a constant factor approximation to the signal regardless of c.

Polynomial interpolation. Our result is a generalization of robust polynomial interpolation, and in Theorem 12.1.2 we construct an optimal method for polynomial interpolation as a first step toward interpolating Fourier-sparse signals.

Our result here can be seen as essentially an extension of a technique shown in [CDL13]. The focus of [CDL13] is on the setting where sample points x_i are chosen independently, so

$\Theta(d \log d)$ samples are necessary. One of their examples, however, shows essentially the same thing as our Corollary 12.4.2. From this, getting our theorem is not difficult.

The recent work [GZ16] looks at robust polynomial interpolation in a different noise model, featuring ℓ_∞ bounded noise with some outliers. In this setting they can get a stronger ℓ_∞ guarantee on the output than is possible in our setting.

Nyquist sampling. The classical method for learning bandlimited signals uses Nyquist sampling—i.e., samples at rate $1/F$, for FT points—and interpolates them using Shannon-Nyquist interpolation. This doesn't require any frequency gap, but also doesn't benefit from sparsity like sparse Fourier transform-based techniques. As discussed in [PS15], on the signal $x(t) = 1$ it takes $FT + O(1/\delta)$ samples to get δ error on average. Our dependence is logarithmic on both those terms.

12.1.2 Our techniques

Previous results on sparse Fourier transforms with robust recovery all required a frequency gap. So consider the opposite situation, where all the frequencies converge to zero and the coefficients are adjusted to keep the overall energy fixed. If we take a Taylor expansion of each complex exponential, then the signal will converge to a degree k polynomial. So robust polynomial interpolation is a necessary subproblem for our algorithm.

Polynomial interpolation. Let $P(x)$ be a degree d polynomial, and suppose that we can query $f(x) = P(x) + g(x)$ over the interval $[-1, 1]$, where g represents adversarial noise. We would like to query f at $O(d)$ points and output a degree d polynomial $Q(x)$ such that $\|P - Q\| \lesssim \|g\|$, where we define $\|h\|^2 := \int_{-1}^{1} |h(x)|^2 \mathrm{d}x$.

One way to do this would be to sample points $S \subset [-1, 1]$ uniformly, then output the degree d polynomial Q with the smallest empirical error

$$\|P + g - Q\|_S^2 := \frac{1}{|S|} \sum_{x \in S} |(P + g - Q)(x)|^2$$

on the observed points. If $\|R\|_S \approx \|R\|$ for all degree d polynomials R, in particular for $P - Q$, then since usually $\|g\|_S \lesssim \|g\|$ by Markov's inequality, the result follows.

This has two problems: first, uniform sampling is poor because polynomials like Chebyshev polynomials can have most of their energy within $O(1/d^2)$ of the edges of the interval. This necessitates $\Omega(d^2)$ uniform samples before $\|R\|_S \approx \|R\|$ with good probability on a single polynomial. Second, the easiest method to extend from approximating one polynomial to approximating all polynomials uses a union bound over a net exponential in d, which would give an $O(d^3)$ bound.

To fix this, we need to bias our sampling toward the edges of the interval and we need our sampling to not be iid. We partition $[-1, 1]$ into $O(d)$ intervals $I_1, \ldots, I_n$ so that the interval containing each x has width at most $O(\sqrt{1-x^2})$, except for the $O(1/d^2)$ size regions at the edges. For any degree d polynomial R and any choice of n points $x_i \in I_i$, the appropriately weighted empirical energy is close to $\|R\|$. This takes care of both issues with uniform sampling. If the points are chosen uniformly at random from within their intervals, then $\|g\|$ is probably bounded as well, and the empirically closest degree d polynomial Q will satisfy our requirements.

This result is shown in Section 12.4.

Clusters. Many previous sparse Fourier transform algorithms start with a one-sparse recovery algorithm, then show how to separate frequencies to get a k-sparse algorithm by reducing to the one-sparse case. Without a frequency gap, we cannot hope to reduce to the one-sparse case; instead, we reduce to individual clusters of nearby frequencies.

Essentially the problem is that one *cannot* determine all of the high-energy frequencies of a function x only by sampling it on a bounded interval, as some of the frequencies might cancel each other out on this interval. We also cannot afford to work merely with the frequencies of the truncation of x to the interval $[0, T]$, as the truncation operation will spread the frequencies of x over too wide a range. To fix this problem, we must do something in between the two. In particular, we instead study $x \cdot H$ for a judiciously chosen function H. We want H to approximate the indicator function of the interval $[0, T]$ and have small Fourier-support, $\text{supp}(\widehat{H}) \subset [-k^c/T, k^c/T]$. By using some non-trivial lemmas about the growth rate of x^*, we can show that the difference between $x \cdot H$ on $\mathbb{R}$ and the truncation of

x to $[0, T]$ has small L^2 mass, so that we can use the former as a substitute for the latter.

On the other hand, the Fourier transform of $x \cdot H$ is the convolution $\widehat{x} * \widehat{H}$, which has most of its mass within $\text{poly}(k)/T$ of the frequencies of x^*. Although it is impossible to determine the individual frequencies of x^*, we can hope to identify $O(k)$ intervals each of length $\text{poly}(k)/T$ so that all but a small fraction of the energy of $\widehat{x}$ is contained within these intervals.

Note that many of these intervals will represent not individual frequencies of x^*, but small clusters of such frequencies. Furthermore, some frequencies of x^* might not show up in these intervals either because they are too small, or because they cancel out other frequencies when convolved with $\widehat{H}$.

One-cluster recovery. Given our notion of clusters, we start looking at Fourier-sparse interpolation in the special case of *one-cluster recovery*. This is a generalization of one-sparse recovery where we can have multiple frequencies, but they all lie in $[f - \Delta, f + \Delta]$ for some base frequency f and bandwidth $\Delta = k^c/T$. Because all the frequencies are close to each other, values $x(a)$ and $x(a + \beta)$ will tend to have ratio close to $e^{2\pi \mathbf{i} f \beta}$ when β is small enough. We find that $\beta < \frac{1}{\Delta\sqrt{T\Delta}}$ is sufficient, which lets us figure out a frequency $\widetilde{f}$ with $|\widetilde{f} - f| \leq \Delta\sqrt{T\Delta} = k^{O(1)}/T$.

Once we have the frequency $\widetilde{f}$, we can consider $x'(t) = x(t)e^{-2\pi \mathbf{i} \widetilde{f}}$. This signal is k-Fourier-sparse with frequencies bounded by $k^{O(1)}/T$. By taking a Taylor approximation to each complex exponential[1], can show x^* is δ-close to $P(t)e^{2\pi \mathbf{i} \widetilde{f}}$ for a degree $d = O(k^c +$

[1]There is a catch here, that the coefficients of the exponentials are potentially unbounded, if the frequencies are arbitrarily close together. We first use Gram determinants to show that the signal is δ-close to one

$k \log(1/\delta))$ polynomial P. Thus we could apply our polynomial interpolation algorithm to recover the signal.

k-cluster frequency estimation. Reminiscent of algorithms such as [HIKP12a, PS15], we choose random variables $\sigma \approx T/k^c$, $a \in [0,1]$, and $b \in [0, 1/\sigma]$ and look at $v \in \mathbb{C}^{k^c}$ given by

$$v_i = (x \cdot H)(\sigma(i-a))e^{-2\pi \mathbf{i} \sigma b i} G(i)$$

where G is a filter function. That is, G has compact support ($\mathrm{supp}(G) \subset [-k^c, k^c]$), and $\widehat{G}$ approximates an interval of length $\Theta(\frac{2\pi}{k})$. In other words, G is the same as $\widehat{H}$ with different parameters: an interval convolved with itself k^c times, multiplied by a sinc function.

We alias v down to $O(k)$ dimensions and take the discrete Fourier transform, getting $\widehat{u}$. It has been implicit in previous work—and we make it explicit—that $\widehat{u}_j$ is equal to $z_{\sigma a}$ for a vector z defined by

$$\widehat{z} = (\widehat{x} * \widehat{H}) \cdot \widehat{G}^{(j)}_{\sigma,b}$$

where $\widehat{G}^{(j)}_{\sigma,b}$ is a particular permutation of $\widehat{G}$. In particular, $\widehat{G}^{(j)}_{\sigma,b}$ has period $1/\sigma$, and approximates an interval of size $\frac{1}{\sigma B}$ within each period.

In previous work, when σ and b were chosen randomly, each individual frequency would have a good chance of being the only frequency preserved in $\widehat{z}$, and we could apply one-sparse recovery by choosing a variety of a. Without a frequency gap we can't quite say that: we pick $1/\sigma \gg \Delta$ so that the entire cluster usually lands in the same bin, but then nearby clusters can also often land in the same bin. Fortunately, it is still usually true

with frequency gap $\delta 2^{-k}$, and coefficients at most $2^k/\delta$.

that only nearby clusters will collide. Since our 1-cluster algorithm works when the signal frequencies are nearby, we apply it to find a frequency approximation within $\frac{\sqrt{T/\sigma}}{\sigma} = k^{O(1)}/T$ of the cluster.

The above algorithm recovers each individual frequency with constant probability. By repeating it $O(\log k)$ times, with high probability we find a list L of $O(k)$ frequencies within $k^{O(1)}/T$ of each significant cluster.

k-sparse recovery. Because different clusters aren't anywhere close to orthogonal, we can't simply approximate each cluster separately and add them up. Instead, given the list L of candidate frequencies, we consider the $O(kd)$-dimensional space of functions

$$\widetilde{x}(t) := \sum_{\widetilde{f} \in L} \sum_{i=0}^{d} \alpha_{\widetilde{f},i} t^i e^{2\pi \mathbf{i} \widetilde{f} t}$$

where $d = O(k^{O(1)} + \log(1/\delta))$. We then take a bunch of random samples of x, and choose the $\widetilde{x}(t)$ minimizing the empirical error using linear regression. This regression can be made slightly faster using oblivious subspace embeddings [CW13], [NN13a], [Woo14b],[CNW15].

Our argument to show this works is analogous to the naive method we considered for polynomial recovery. Similarly to the one-cluster setting, using Taylor approximations and Gram determinants, we can show that this space includes a sufficiently close approximation to x. Since polynomials are the limit of sparse Fourier as frequencies tend to zero, these functions are arbitrarily close to $O(kd)$-Fourier-sparse functions. Hence we know that the maximum of $|\widetilde{x}(t)|$ is at most a poly(kd) factor larger than its average over $[0, T]$. Using a net argument, this shows poly(kd) samples are sufficient to find a good approximation to the nearest function in our space.

Growth rate of Fourier-sparse signals. We need that $\frac{1}{\sqrt{T}}\|x^* \cdot H\|_2 \approx \|x^*\|_T$, where H approximates the interval $1_{[0,T]}$. Because H has support size k^c/T, it has a transition region of size $T/k^{c'}$ at the edges, and it decays as $(t/T)^{-k^{c''}}$ for $t \gg T$. The difference between $\frac{1}{\sqrt{T}}\|x^* \cdot H\|_2$ and $\|x^*\|_T$ involves two main components: mass in the transition region that is lost, and mass outside the sampling interval that is gained. To show the approximation, we need that $|x^*(t)| \lesssim \widetilde{O}(k^2)\|x^*\|_T$ within the interval and $|x^*(t)| \lesssim (kt/T)^{O(k)}\|x^*\|_T$ outside.

We outline the bound of $\max_{t\in[0,T]}|x^*(t)|$ in terms of its average $\|x^*\|_T$ to bound $|x^*(t)|$ within the interval. Notice that we can assume $|x^*(0)| = \max_{t\in[0,T]}|x^*(t)|$: if $t^* = \arg\max_{t\in[0,T]}|x^*(t)|^2$ is not 0 or T, we can rescale the two intervals $[0,t^*]$ and $[t^*,T]$ to $[0,T]$ separately. Then we show that for any t', there exist $m = \widetilde{O}(k^2)$ and constants $C_1,\cdots,C_m$ such that $x^*(0) = \sum_{j\in[m]} C_j \cdot x^*(j\cdot t')$. Then we take the integration of t' over $[0,T/m]$ to bound $|x^*(0)|^2$ by its average. For any outside $t > T$, we follow this approach to show $x^*(t) = \sum_{j\in[k]} C_j \cdot x^*(t_j)$ where $t_j \in [0,T]$ and $|C_j| \leq \mathrm{poly}(k)\cdot(kt/T)^{O(k)}$ for each $j \in [k]$. These results are shown in Section 12.5.

12.1.3 Organization

This paper is organized as follows. We provide a brief overview about signal recovery in Section 12.2. We introduce some notations and tools in Section 12.3. Then we show our main Theorem 12.1.2 about polynomial interpolation in Section 12.4. For signals with k-sparse Fourier transform, we show two bounds on their growth rate in Section 12.5 and describe the hash functions and filter functions in Section 12.6. We provide the algorithm for frequency estimation and its proof in Section 12.7. In Section 12.8, we describe the algorithm for one-cluster recovery. In Section 12.9, we show the proof of Theorem 12.1.1. We defer several technical proofs in Section 12.10. Section 12.11 gives a summary of several well-known facts are existing in literature. We provide the analysis of hash functions and filter functions in Section 12.12.

12.2 Proof Sketch

We first consider one-cluster recovery centered at zero, i.e., $x^*(t) = \sum_{j=1}^{k} v_j \cdot e^{2\pi \mathbf{i} f_j t}$ where every f_j is in $[-\Delta, \Delta]$ for some small $\Delta > 0$. The road map is to replace x^* by a low degree polynomial P such that $\|x^*(t) - P(t)\|_T^2 \lesssim \delta \|x^*\|_T^2$ then recover a polynomial Q to approximate P through the observation $x(t) = P(t) + g'(t)$ where $g'(t) = g(t) + \big(x^*(t) - P(t)\big)$.

A natural way to replace $x^*(t) = \sum_{j=1}^{k} v_j e^{2\pi \mathbf{i} f_j t}$ by a low degree polynomial $P(t)$ is the Taylor expansion. To bound the error after taking the low degree terms in the expansion by $\delta \|x^*\|_T$, we show the existence of $x'(t) = \sum_{j=1}^{k} v'_j e^{2\pi \mathbf{i} f'_j t}$ approximating x^* on $[0, T]$ with an extra property—any coefficient v'_j in $x'(t)$ has an upper bound in terms of $\|x'\|_T^2 = \frac{1}{T}\int_0^T |x'(t)|^2 \mathrm{d}t$. We prove the existence of $x'(t)$ via two more steps, both of which rely on the estimation of some Gram matrix constituted by these k signals.

The first step is to show the existence of a k-Fourier-sparse signal $x'(t)$ with frequency gap $\eta \geq \frac{\exp(-\operatorname{poly}(k)) \cdot \delta}{T}$ that is sufficiently close to $x^*(t)$.

Lemma 12.2.1. There is a universal constant $C_1 > 0$ such that, for any $x^*(t) = \sum_{j=1}^{k} v_j e^{2\pi \mathbf{i} f_j t}$ and any $\delta > 0$, there always exist $\eta \geq \frac{\delta}{T} \cdot k^{-C_1 k^2}$ and $x'(t) = \sum_{j=1}^{k} v'_j e^{2\pi \mathbf{i} f'_j t}$ satisfying

$$\|x'(t) - x^*(t)\|_T \leq \delta \|x^*(t)\|_T$$

with $\min_{i \neq j} |f'_i - f'_j| \geq \eta$ and $\max_{j \in [k]} \{|f'_j - f_j|\} \leq k\eta$.

We outline our approach and defer the proof to Section 12.8. We focus on the replacement of one frequency f_k in $x^* = \sum_{j \in [k]} v_j e^{2\pi \mathbf{i} f_j t}$ by a new frequency $f_{k+1} \neq f_k$ and its error. The idea is to consider every signal $e^{2\pi \mathbf{i} f_j t}$ as a vector and prove that for any vector

x^* in the linear subspace $\text{span}\{e^{2\pi \mathbf{i} f_j t} | j \in [k]\}$, there exists a vector in the linear subspace $\text{span}\{e^{2\pi \mathbf{i} f_{k+1} t}, e^{2\pi \mathbf{i} f_j t} | j \in [k-1]\}$ with distance at most $\exp(k^2) \cdot (|f_k - f_{k+1}|T) \cdot \|x^*\|_T$ to x^*.

The second step is to lower bound $\|x'\|_T^2$ by its coefficients through the frequency gap η in x'.

Lemma 12.2.2. There exists a universal constant $c > 0$ such that for any $x(t) = \sum_{j=1}^{k} v_j e^{2\pi \mathbf{i} f_j t}$ with frequency gap $\eta = \min_{i \neq j} |f_i - f_j|$,

$$\|x(t)\|_T^2 \geq k^{-ck^2} \min\left((\eta T)^{2k}, 1\right) \sum_{j=1}^{k} |v_j|^2.$$

Combining Lemma 12.2.1 and Lemma 12.2.2, we bound $|v'_j|$ by $\exp(\text{poly}(k)) \cdot \delta^{-O(k)} \cdot \|x'\|_T$ for any coefficient v'_j in x'. Now we apply the Taylor expansion on $x'(t)$ and keep the first $d = O(\Delta T + \text{poly}(k) + k \log \frac{1}{\delta})$ terms of every signal $v'_j \cdot e^{2\pi \mathbf{i} f'_j t}$ in the expansion to obtain a polynomial $P(t)$ of degree at most d. To bound the distance between $P(t)$ and $x'(t)$, we observe that the error of every point $t \in [0, T]$ is at most $(\frac{2\pi\Delta \cdot T}{d})^d \sum_j |v'_j|$, which can be upper bounded by $\delta \|x'(t)\|_T$ via the above connection. We summarize all discussion above as follows.

Lemma 12.2.3. For any $\Delta > 0$ and any $\delta > 0$, let $x^*(t) = \sum_{j \in [k]} v_j e^{2\pi \mathbf{i} f_j t}$ where $|f_j| \leq \Delta$ for each $j \in [k]$. There exists a polynomial $P(t)$ of degree at most

$$d = O(T\Delta + k^3 \log k + k \log 1/\delta)$$

such that

$$\|P(t) - x^*(t)\|_T^2 \leq \delta \|x^*\|_T^2.$$

To recover $x^*(t)$, we observe $x(t)$ as a degree d polynomial $P(t)$ with noise. We use properties of the Legendre polynomials to design a method of random sampling such that we only need $O(d)$ random samples to find a polynomial $Q(t)$ approximating $P(t)$.

Theorem 12.1.2. For any degree d polynomial $P(t)$ and an arbitrary function $g(t)$, Procedure RobustPolynomialLearning in Algorithm 12.5 takes $O(d)$ samples from $x(t) = P(t) + g(t)$ over $[0, T]$ and reports a degree d polynomial $Q(t)$ in time $O(d^\omega)$ such that, with probability at least 99/100,

$$\|P(t) - Q(t)\|_T^2 \lesssim \|g(t)\|_T^2.$$

where $\omega < 2.373$ is matrix multiplication exponent [Str69],[CW87],[Wil12].

We can either report the polynomial $Q(t)$ or transfer $Q(t)$ to a signal with d-sparse Fourier transform. We defer the technical proofs and the formal statements to Section 12.8 and discuss the recovery of k clusters from now on.

As mentioned before, we apply the filter function $(H(t), \widehat{H}(f))$ on x^* such that $\widehat{x^* \cdot H}$ has at most k clusters given $\widehat{x^*}$ with k-sparse Fourier transform. First, we show that all frequencies in the "heavy" clusters of $\widehat{x^* \cdot H}$ constitute a good approximation of x^* in Section 12.9.

Definition 12.2.4. Given $x^*(t) = \sum_{j=1}^{k} v_j e^{2\pi \mathbf{i} f_j t}$, any $\mathcal{N} > 0$, and a filter function $(H, \widehat{H})$ with bounded support in frequency domain. Let L_j denote the interval of $\operatorname{supp}(\widehat{e^{2\pi \mathbf{i} f_j t} \cdot H})$ for each $j \in [k]$.

Define an equivalence relation $\sim$ on the frequencies f_i by the transitive closure of the relation $f_i \sim f_j$ if $L_i \cap L_j \neq \emptyset$. Let $S_1, \ldots, S_n$ be the equivalence classes under this relation.

Define $C_i = \bigcup_{f \in S_i} L_i$ for each $i \in [n]$. We say C_i is a "heavy" cluster iff $\int_{C_i} |\widehat{H \cdot x^*}(f)|^2 \mathrm{d}f \geq T \cdot \mathcal{N}^2/k$.

Claim 12.2.5. Given $x^*(t) = \sum_{j=1}^{k} v_j e^{2\pi \mathbf{i} f_j t}$ and any $\mathcal{N} > 0$, let H be the filter function defined in Section 12.12.1 and $C_1, \cdots, C_l$ be the heavy clusters from Definition 12.2.4. For

$$S = \left\{ j \in [k] \middle| f_j \in C_1 \cup \cdots C_l \right\},$$

we have $x^{(S)}(t) = \sum_{j \in S} v_j e^{2\pi \mathbf{i} f_j t}$ approximating x^* within distance $\|x^{(S)}(t) - x^*(t)\|_T^2 \lesssim \mathcal{N}^2$.

Hence it is enough to recover $x^{(S)}$ for the recovery of x^*. Let Δ_h denote the bandwidth of $\widehat{H}$. In Section 12.7, we choose $\Delta > k \cdot \Delta_h$ such that for any $j \in S$, $\int_{f_j - \Delta}^{f_j + \Delta} |\widehat{H \cdot x^*}(f)|^2 \mathrm{d}f \geq T \cdot \mathcal{N}^2/k$ from the fact $|C_i| \leq k \cdot \Delta_h$. Then we prove Theorem 12.2.6 in Section 12.7, which finds $O(k)$ frequencies to cover all heavy clusters of $\widehat{x^* \cdot H}$.

Theorem 12.2.6. Let $x^*(t) = \sum_{j=1}^{k} v_j e^{2\pi \mathbf{i} f_j t}$ and $x(t) = x^*(t) + g(t)$ be our observable signal where $\|g(t)\|_T^2 \leq c\|x^*(t)\|_T^2$ for a sufficiently small constant c. Then Procedure FrequencyRecoveryKCluster returns a set L of $O(k)$ frequencies that covers all heavy clusters of x^*, which uses $\mathrm{poly}(k, \log(1/\delta)) \log(FT)$ samples and $\mathrm{poly}(k, \log(1/\delta)) \log^2(FT)$ time. In particular, for $\Delta = \mathrm{poly}(k, \log(1/\delta))/T$ and $\mathcal{N}^2 := \|g(t)\|_T^2 + \delta \|x^*(t)\|_T^2$, with probability $1 - 2^{-\Omega(k)}$, for any f^* with

$$\int_{f^* - \Delta}^{f^* + \Delta} |\widehat{x \cdot H}(f)|^2 \mathrm{d}f \geq T\mathcal{N}^2/k, \tag{12.2}$$

there exists an $\widetilde{f} \in L$ satisfying

$$|f^* - \widetilde{f}| \lesssim \Delta\sqrt{\Delta T}.$$

Let $L = \{\widetilde{f}_1, \cdots, \widetilde{f}_l\}$ be the list of frequencies from the output of Procedure FrequencyRecoveryKCluster in Theorem 12.2.6. The guarantee is that, for any f_j in

$x^{(S)}$, there exists some $p_j \in [l]$ such that $|\widetilde{f}_{p_j} - f_j| \lesssim \Delta\sqrt{\Delta T}$ for $\Delta = \mathrm{poly}(k, \log(1/\delta))/T$. Hence we rewrite $x^{(S)}(t) = \sum_{i\in[l]} e^{2\pi\mathbf{i}\widetilde{f}_i t}(\sum_{j\in S:p_j=i} e^{2\pi\mathbf{i}(f_j-\widetilde{f}_i)t})$. For each $i \in [l]$, we apply Lemma 12.2.3 of one-cluster recovery on $\sum_{j\in S:p_j=i} e^{2\pi\mathbf{i}(f_j-\widetilde{f}_i)t}$ to approximate it by a degree d polynomial $P_i(t)$.

Now we consider $x(t) = \sum_{i\in[l]} e^{2\pi\mathbf{i}\widetilde{f}_i t} \cdot P_i(t) + g''(t)$ where $\|g''(t)\|_T \lesssim \|g(t)\|_T + \delta\|x^*(t)\|_T$. To recover $\sum_{i\in[l]} e^{2\pi\mathbf{i}\widetilde{f}_i t} \cdot P_i(t)$, we treat it as a vector in the linear subspace

$$V = \mathrm{span}\left\{ e^{2\pi\mathbf{i}\widetilde{f}_i t} \cdot t^j \middle| j \in \{0, \cdots, d\}, i \in [l] \right\}$$

with dimension at most $l(d+1)$ and find a vector in this linear subspace approximating it.

We show that for any $v \in V$, the average of $\mathrm{poly}(kd)$ random samples on v is enough to estimate $\|v\|_T^2$. In particular, any vector in this linear subspace satisfies that the maximum of it in $[0, T]$ has an upper bound in terms of its average in $[0, T]$. Then we apply the Chernoff bound to prove that $\mathrm{poly}(kd)$ random samples are enough for the estimation of one vector $v \in V$.

Claim 12.2.7. For any $\vec{u} \in \mathrm{span}\left\{ e^{2\pi\mathbf{i}\widetilde{f}_i t} \cdot t^j \middle| j \in \{0, \cdots, d\}, i \in [l] \right\}$, there exists some universal constants $C_1 \leq 4$ and $C_2 \leq 3$ such that

$$\max_{t\in[0,T]}\{|\vec{u}(t)|^2\} \lesssim (ld)^{C_1} \log^{C_2}(ld) \cdot \|\vec{u}\|_T^2$$

At last we use an ϵ-net to argue that $\mathrm{poly}(kd)$ random samples from $[0, T]$ are enough to interpolate $x(t)$ by a vector $v \in V$. Because the dimension of this linear subspace is at most $l(d+1) = O(kd)$, there exists an ϵ-net in this linear subspace for unit vectors with size at most $\exp(kd)$. Combining the Chernoff bound on all vectors in the ϵ-net and Claim 12.2.7, we know that $\mathrm{poly}(kd)$ samples are sufficient to estimate $\|v\|_T^2$ for any vector $v \in V$.

In Section 12.9, we show that a vector $v \in V$ minimizing the distance on $\mathrm{poly}(kd)$ random samples is a good approximation for $\sum_{i\in[l]} e^{2\pi \mathbf{i} \widetilde{f}_i t} \cdot P_i(t)$, which is a good approximation for $x^*(t)$ from all discussion above.

Theorem 12.1.1. Let $x(t) = x^*(t) + g(t)$, where x^* is k-Fourier-sparse signal with frequencies in $[-F, F]$. Given samples of x over $[0, T]$ we can output $\widetilde{x}(t)$ such that with probability at least $1 - 2^{-\Omega(k)}$,

$$\|\widetilde{x} - x^*\|_T \lesssim \|g\|_T + \delta \|x^*\|_T .$$

Our algorithm uses $\mathrm{poly}(k, \log(1/\delta)) \cdot \log(FT)$ samples and $\mathrm{poly}(k, \log(1/\delta)) \cdot \log^2(FT)$ time. The output $\widetilde{x}$ is $\mathrm{poly}(k, \log(1/\delta))$-Fourier-sparse signal.

12.3 Preliminaries

We first provide some notations in Section 12.3.1 and basic Fourier facts in Section 12.3.2. Then we review some probability inequalities in Section 12.3.3. At last, we introduce Legendre polynomials in Section 12.3.4 and review some basic properties of Gram matrix and its determinant in Section 12.3.5.

12.3.1 Notation

For any function f, we define $\widetilde{O}(f)$ to be $f \cdot \log^{O(1)}(f)$. We use $[n]$ to denote $\{1, 2, \cdots, n\}$. Let $\mathbf{i}$ denote $\sqrt{-1}$. For any Complex number $z = a + \mathbf{i}b \in C$, where $a, b \in \mathbb{R}$. We define $\overline{z}$ to be $a - \mathbf{i}b$ and $|z| = \sqrt{a^2 + b^2}$ such that $|z|^2 = z\overline{z}$. For any function $f(t) : \mathbb{R} \to \mathbb{C}$, we use $\mathrm{supp}(f)$ to denote the support of f.

For convenience, we define the sinc function and the Gaussian distribution $\mathrm{Gaussian}_{\mu,\sigma}$ on $\mathbb{R}$ with expectation μ and variance σ^2 as follows:

$$\mathrm{sinc}(t) = \frac{\sin(\pi t)}{\pi t}, \quad \mathrm{Gaussian}_{\mu,\sigma}(t) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(t-\mu)^2}{2\sigma^2}}.$$

For a fixed $T > 0$, we define the inner product of two functions $x, y : [0, T] \to \mathbb{C}$ as

$$\langle x, y \rangle_T = \frac{1}{T} \int_0^T x(t)\overline{y}(t)\mathrm{d}t.$$

We define the $\| \cdot \|_T$ norm as

$$\|x(t)\|_T = \sqrt{\langle x(t), x(t) \rangle_T} = \sqrt{\frac{1}{T} \int_0^T |x(t)|^2 \mathrm{d}t}.$$

12.3.2 Facts about the Fourier transform

In this work, we always use $x(t)$ to denote a signal from $\mathbb{R} \to \mathbb{C}$. The Fourier transform $\widehat{x}(f)$ of an integrable function $x : \mathbb{R} \to \mathbb{C}$ is defined as

$$\widehat{x}(f) = \int_{-\infty}^{+\infty} x(t) e^{-2\pi \mathbf{i} f t} \mathrm{d}t, \text{ for any real number } f.$$

Similarly, $x(t)$ is determined from $\widehat{x}(f)$ by the inverse transform:

$$x(t) = \int_{-\infty}^{+\infty} \widehat{x}(f) e^{2\pi \mathbf{i} f t} \mathrm{d}f, \text{ for any real number } t.$$

Let CFT denote the continuous Fourier transform, DTFT denote the discrete-time Fourier transform, DFT denote the discrete Fourier transform, and FFT denote the fast Fourier transform.

For any signal $x(t)$ and $n \in \mathbb{N}_+$, we define $x^{*n}(t) = \underbrace{x(t) * \cdots * x(t)}_{n}$ and $\widehat{x}^{\cdot n}(f) = \underbrace{\widehat{x}(f) \cdot \cdots \cdot \widehat{x}(f)}_{n}$.

Fact 12.3.1. *Let $\delta_\Delta(f)$ denote the Dirac delta at Δ. Then*

$$\widehat{\delta}_\Delta(t) = \int_{-\infty}^{+\infty} \delta_\Delta(f) e^{2\pi \mathbf{i} f t} \mathrm{d}f = e^{2\pi \mathbf{i} t \Delta}.$$

Fact 12.3.2. *For any $s > 0$, let $\mathrm{Comb}_s(t) = \sum_{j \in \mathbb{Z}} \delta_{js}(t)$. Then the Fourier transform of $\mathrm{Comb}_s(t)$ is*

$$\widehat{\mathrm{Comb}_s}(f) = \frac{1}{s} \mathrm{Comb}_{1/s}(f).$$

The following fact says the the Fourier transform of a rectangle function is a sinc function.

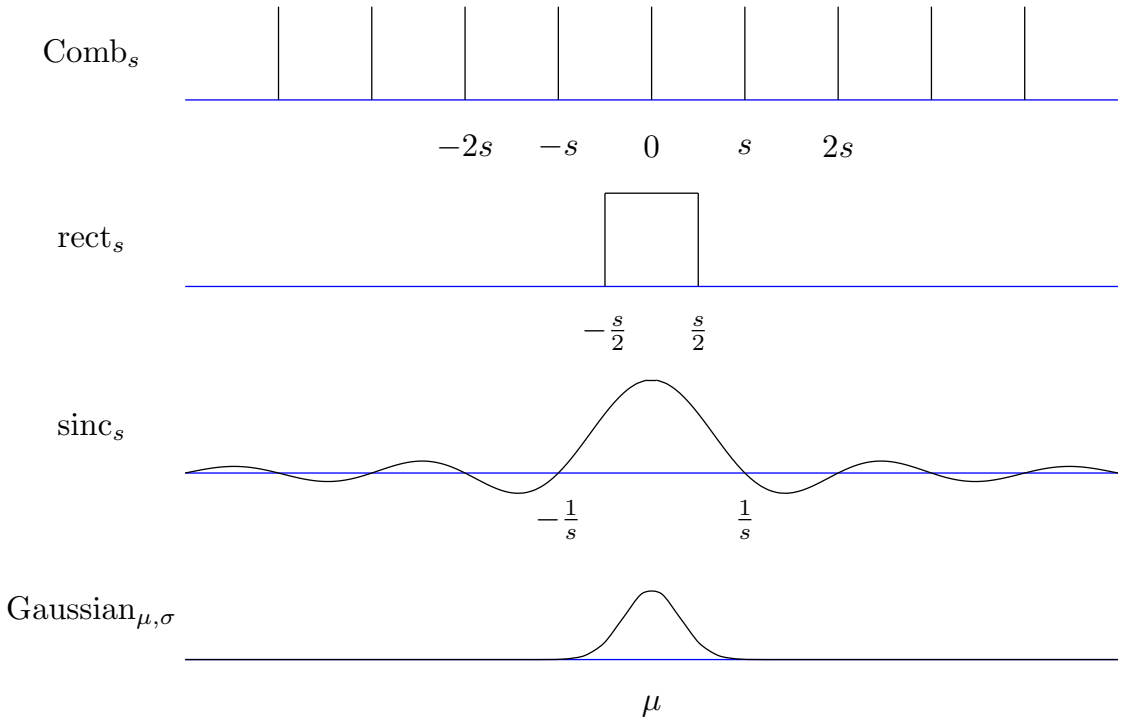


Figure 12.1: A picture of a Comb$_s$, rect$_s$, sinc$_s$, Gaussian$_{\mu,\sigma}$.

Fact 12.3.3. *We use*

$$\text{rect}_s(t) = \begin{cases} 1 & \text{if } |\text{t}| \leq \frac{\text{s}}{2}, \\ 0 & \text{otherwise.} \end{cases}$$

Then the Fourier transform of rect$_s(t)$ *is* $\widehat{\text{rect}_s}(f) = \frac{\sin(\pi f s)}{\pi f s} = \text{sinc}(fs)$.

The Fourier transform of a Gaussian function is another Gaussian function.

Fact 12.3.4. *For* $\text{Gaussian}_{\mu,\sigma}(t) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(t-\mu)^2}{2\sigma^2}}$. *Then the Fourier transform is*

$$\widehat{\text{Gaussian}}_{\mu,\sigma}(f) = e^{-2\pi \mathbf{i} f u} \frac{1}{\sigma\sqrt{2\pi}} \text{Gaussian}_{0,\sigma'}(f) \text{ for } \sigma' = 1/(2\pi\sigma).$$

Proof. From the definition of the Fourier transform,

$$\begin{aligned}
\widehat{\text{Gaussian}}_{\mu,\sigma}(f) &= \int_{-\infty}^{+\infty} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(t-\mu)^2}{2\sigma^2}} e^{-2\pi \mathbf{i} f t} \mathrm{d}t \\
&= e^{-2\pi \mathbf{i} f u} \int_{-\infty}^{+\infty} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{t^2}{2\sigma^2}} e^{-2\pi \mathbf{i} f t} \mathrm{d}t \\
&= e^{-2\pi \mathbf{i} f u} \int_{-\infty}^{+\infty} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(t+2\pi \mathbf{i}\sigma^2 f)^2}{2\sigma^2} - 2\pi^2 f^2 \sigma^2} \mathrm{d}t \\
&= e^{-2\pi \mathbf{i} f u} e^{-\frac{f^2}{2\sigma'^2}}
\end{aligned}$$

where $\sigma' = 1/(2\sigma\pi)$, which is $e^{-2\pi \mathbf{i} f u} \cdot \sigma'\sqrt{2\pi} \cdot \text{Gaussian}_{0,\sigma'}(f)$. □

12.3.3 Tools and inequalities

From the Chernoff Bound (Lemma A.1.1), we show that if the maximum of a signal is bounded by d times its energy over some fixed interval, then taking more than d samples (each sample is drawn i.i.d. over that interval) suffices to approximate the energy of the signal on the interval with high probability.

Lemma 12.3.5. Given any function $x(t) : \mathbb{R} \to \mathbb{C}$ with $\max_{t\in[0,T]} |x(t)|^2 \leq d\|x(t)\|_T^2$. Let S denote a set of points from 0 to T. If each point of S is chosen uniformly at random from $[0, T]$, we have

$$\Pr\left[\left|\frac{1}{|S|}\sum_{i\in S}|x(t_i)|^2 - \|x(t)\|_T^2]\right| \geq \epsilon\|x(t)\|_T^2\right] \quad \leq \quad e^{-\Omega(\epsilon^2|S|/d)}$$

We provide a proof in Section 12.10.5.

Because $d \cdot \frac{1}{2d} + \frac{1}{2} \cdot (1 - \frac{1}{2d}) \leq 1$, we have the following inequality when the maximum of $|x(t)|^2$ is at most d times its average.

Lemma 12.3.6. *Given any function $x(t) : \mathbb{R} \to \mathbb{C}$ with $\max_{t\in[0,T]} |x(t)|^2 \leq d\|x(t)\|_T^2$. Let S denote a set of points from 0 to T. For any point a is sampled uniformly at random from $[0, T]$, we have,*

$$\Pr_{a\sim[0,T]}\left[|x(a)|^2 \geq \frac{1}{2}\|x(t)\|_T^2\right] \geq \frac{1}{2d}.$$

12.3.4 Legendre polynomials

We provide an brief introduction to Legendre polynomials (please see [Dun10] for a complete introduction). For convenience, we fix $\|f(t)\|_T^2 = \frac{1}{2}\int_{-1}^{1} |f(t)|^2 \mathrm{d}t$ in this section.

Definition 12.3.1. Let $L_n(x)$ denote the Legendre polynomials of degree n, the solution to Legendre's differential equation:

$$\frac{\mathrm{d}}{\mathrm{d}x}\left[(1-x^2)\frac{\mathrm{d}}{\mathrm{d}x}L_n(x)\right] + n(n+1)L_n(x) = 0 \tag{12.3}$$

We will the following two facts about the Legendre polynomials in this work.

Fact 12.3.7. *$L_n(1) = 1$ for any $n \geq 0$ in the Legendre polynomials.*

Fact 12.3.8. *The Legendre polynomials constitute an orthogonal basis with respect to the inner product on interval $[-1, 1]$:*

$$\int_{-1}^{1} L_m(x)L_n(x)\mathrm{d}x = \frac{2}{2n+1}\delta_{mn}$$

where δ_{mn} denotes the Kronecker delta, i.e., it equals to 1 if $m = n$ and to 0 otherwise.

For any polynomial $P(x)$ of degree at most d with complex coefficients, there exists a set of coefficients from the above properties such that

$$P(x) = \sum_{i=0}^{d} \alpha_i \cdot L_i(x), \text{ where } \alpha_i \in \mathbb{C}, \forall i \in \{0, 1, 2, \cdots, d\}.$$

Lemma 12.3.9. For any polynomial $P(t)$ of degree at most d from R to $\mathbb{C}$, for any interval $[S, T]$,

$$\max_{t\in[S,T]} |P(t)|^2 \leq (d+1)^2 \cdot \frac{1}{T-S}\int_S^T |P(t)|^2 \mathrm{d}x.$$

We provide a proof in Section 12.10.6.

12.3.5 Gram matrix and its determinant

We provide an brief introduction to Gramian matrices (please see [Haz01] for a complete introduction). We use $\langle x, y\rangle$ to denote the inner product between vector x and vector y.

Let $\vec{v}_1, \cdots, \vec{v}_n$ be n vectors in an inner product space and $\text{span}\{\vec{v}_1, \cdots, \vec{v}_n\}$ be the linear subspace spanned by these n vectors with coefficients in $\mathbb{C}$, i.e.,

$$\left\{ \sum_{i\in[n]} \alpha_i \vec{v}_i \;\middle|\; \forall i \in [n], \alpha_i \in \mathbb{C} \right\}.$$

The Gram matrix Gram_n of $\vec{v}_1, \cdots, \vec{v}_n$ is an $n \times n$ matrix defined as $\text{Gram}_n(i, j) = \langle \vec{v}_i, \vec{v}_j \rangle$ for any $i \in [n]$ and $j \in [n]$.

Fact 12.3.10. $\det(\text{Gram}_n)$ *is the square of the volume of the parallelotope formed by* $\vec{v}_1, \cdots, \vec{v}_n$.

Let Gram_{n-1} be the Gram matrix of $\vec{v}_1, \cdots, \vec{v}_{n-1}$. Let $\vec{v}_n^{\parallel}$ be the projection of v_n onto the linear subspace $\text{span}\{\vec{v}_1, \cdots, \vec{v}_{n-1}\}$ and $\vec{v}_n^{\perp} = \vec{v}_n - \vec{v}_n^{\parallel}$. We use $\|\vec{v}\|$ to denote the length of $\vec{v}$ in the inner product space, which is $\sqrt{\langle \vec{v}, \vec{v} \rangle}$.

Claim 12.3.11.

$$\|\vec{v}_n^{\perp}\|^2 = \frac{\det(\text{Gram}_{n-1})}{\det(\text{Gram}_n)}.$$

Proof.

$$\det(\text{Gram}_n) = \text{volume}^2(\vec{v}_1, \cdots, \vec{v}_n) = \text{volume}^2(\vec{v}_1, \cdots, \vec{v}_{n-1}) \cdot \|\vec{v}_n^{\perp}\|^2 = \det(\text{Gram}_n) \cdot \|\vec{v}_n^{\perp}\|^2.$$

□

12.4 Robust Polynomial Interpolation Algorithm

In Section 12.4.1, we show how to learn a low degree polynomial by using linear number of samples, running polynomial time, and achieving constant success probability. In Section 12.4.2, we show to how boost the success probability by rerunning previous algorithm several times.

12.4.1 Constant success probability

We show how to learn a degree-d polynomial P with $n = O(d)$ samples and prove Theorem 12.1.2 in this section. For convenience, we first fix the interval to be $[-1, 1]$ and use $\|f\|^2_{[-1,1]} = \frac{1}{2}\int_{-1}^{1} |f(t)|^2 \mathrm{d}t$.

Lemma 12.4.1. *Let $d \in \mathbb{N}$ and $\epsilon \in \mathbb{R}^+$, there exists an efficient algorithm to compute a partition of $[-1, 1]$ to $n = O(d/\epsilon)$ intervals $I_1, \cdots, I_n$ such that for any degree d polynomial $P(t) : \mathbb{R} \to \mathbb{C}$ and any n points $x_1, \cdots, x_n$ in the intervals $I_1, \cdots, I_n$ respectively, the function $Q(t)$ defined by*

$$Q(t) = P(x_j) \quad \text{if} \quad t \in I_j$$

approximates P by

$$\|Q - P\|_{[-1,1]} \leq \epsilon \|P\|_{[-1,1]}. \tag{12.4}$$

One direct corollary from the above lemma is that observing $n = O(d/\epsilon)$ points each from $I_1, \cdots, I_n$ provides a good approximation for all degree d polynomials. For any set $S = \{t_1, \cdots, t_m\}$ where each $t_i \in [-1, 1]$ and a distribution with support $\{w_1, \cdots, w_m\}$ on S where $\sum_{i=1}^{m} w_i = 1$ and $w_i \geq 0$ for each $i \in [m]$, we define $\|x\|_{S,w} = (\sum_{i=1}^{m} w_i \cdot |x(t_i)|^2)^{1/2}$.

Corollary 12.4.2. *Let $I_1, \cdots, I_n$ be the intervals in the above lemma and $w_j = |I_j|/2$ for each $j \in [n]$. For any $x_1, \cdots, x_n$ in the intervals $I_1, \cdots, I_n$ respectively, we consider $S = \{x_1, \cdots, x_n\}$ with the distribution $w_1, \cdots, w_n$. Then for any degree d polynomial P, we have*

$$\|P\|_{S,w} \in \left[(1-\epsilon)\|P\|_{[-1,1]}, (1+\epsilon)\|P\|_{[-1,1]}\right].$$

We first state the main technical lemma and finish the proof of the above lemma (we defer the proof of Lemma 12.4.3 to Section 12.10.3).

Lemma 12.4.3. For any degree d polynomial $P(t) : \mathbb{R} \to \mathbb{C}$ with derivative $P'(t)$, we have,

$$\int_{-1}^{1} (1-t^2)|P'(t)|^2 \mathrm{d}t \leq 2d^2 \int_{-1}^{1} |P(t)|^2 \mathrm{d}t. \tag{12.5}$$

Proof of Lemma 12.4.1. We set $m = 10d/\epsilon$ and show a partition of $[-1, 1]$ into $n \leq 20m$ intervals. We define $g(t) = \frac{\sqrt{1-t^2}}{m}$ and $y_0 = 0$. Then we choose $y_i = y_{i-1} + g(y_{i-1})$ for $i \in \mathbb{N}^+$. Let l be the first index of y such that $y_l \geq 1 - \frac{9}{m^2}$. We show $l \lesssim m$.

Let j_k be the first index in the sequence such that $y_{j_k} \geq 1 - 2^{-k}$. Notice that

$$j_2 \leq \frac{3/4}{\frac{\sqrt{1-(3/4)^2}}{m}} \leq 1.5m$$

and

$$y_i - y_{i-1} = g(y_{i-1}) = \frac{\sqrt{1-y_{i-1}^2}}{m} \geq \frac{\sqrt{1-y_{i-1}}}{m}.$$

Then for all $k > 2$, we have

$$j_k - j_{k-1} \leq \frac{2^{-k}}{\frac{\sqrt{1-y_{(j_k-1)}}}{m}} \leq 2^{-k/2} m.$$

Therefore $j_k \leq \left(1.5 + (2^{-3/2} + \cdots 2^{-k/2})\right) m$ and $l \leq 10m$.

Because $y_{l-1} \leq 1 - \frac{9}{m^2}$, for any $j \in [l]$ and any $x \in [y_{i-1}, y_i]$, we have the following property:

$$\frac{1-x^2}{m^2} \geq \frac{1}{2} \cdot \frac{(1-y_{i-1}^2)}{m^2} = (y_i - y_{i-1})^2/2. \tag{12.6}$$

Now we set n and partition $[-1, 1]$ into $I_1, \cdots, I_n$ as follows:

1. $n = 2(l+1)$.

2. For $j \in [l]$, $I_{2j-1} = [y_{j-1}, y_j]$ and $I_{2j} = [-y_j, -y_{j-1}]$.

3. $I_{2l+1} = [y_l, 1]$ and $I_{2l+2} = [-1, -y_l]$.

For any $x_1, \cdots, x_n$ where $x_j \in I_j$ for each $j \in [n]$, we rewrite the LHS of (12.4) as follows:

$$\underbrace{\sum_{j=1}^{n-2} \int_{I_j} |P(x_j) - P(t)|^2 \, \mathrm{d}t}_{A} + \underbrace{\int_{I_{n-1}} |P(x_{n-1}) - P(t)|^2 \, \mathrm{d}t + \int_{I_n} |P(x_n) - P(t)|^2 \, \mathrm{d}t}_{B}. \tag{12.7}$$

For A in Equation (12.7), from the Cauchy-Schwarz inequality, we have

$$\sum_{j=1}^{n-2} \int_{I_j} |P(x_j) - P(t)|^2 \, \mathrm{d}t = \sum_{j=1}^{n-2} \int_{I_j} \left| \int_{x_j}^{t} P'(y) \mathrm{d}y \right|^2 \mathrm{d}t \leq \sum_{j=1}^{n-2} \int_{I_j} |t - x_j| \int_{x_j}^{t} |P'(y)|^2 \mathrm{d}y \mathrm{d}t.$$

Then we swap $\mathrm{d}t$ with $\mathrm{d}y$ and use Equation (12.6):

$$\sum_{j=1}^{n-2} \int_{I_j} |P'(y)|^2 \int_{t \notin (x_j, y)} |t - x_j| \mathrm{d}t \mathrm{d}y \leq \sum_{j=1}^{n-2} \int_{I_j} |P'(t)|^2 \cdot |I_j|^2 \mathrm{d}t \leq \sum_{j=1}^{n-2} \int_{I_j} |P'(t)|^2 \frac{2(1-t^2)}{m^2} \mathrm{d}t.$$

We use Lemma 12.4.3 to simplify it by

$$\sum_{j=1}^{n-2} \int_{I_j} |P(x_j) - P(t)|^2 \, \mathrm{d}t \leq \int_{-1}^{1} |P'(t)|^2 \frac{2(1-t^2)}{m^2} \mathrm{d}t \leq \frac{2d^2}{m^2} \int_{-1}^{1} |P(t)|^2 \mathrm{d}t.$$

For B in Equation (12.7), notice that $|I_{n-1}| = |I_n| = 1 - y_l \leq 9m^{-2}$ and for $j \in \{n-1, n\}$

$$|P(t) - P(x_j)|^2 \leq 4 \max_{t \in [-1,1]} |P(t)|^2 \leq 4(d+1)^2 \|P\|_{[-1,1]}^2$$

from the properties of degree-d polynomials, i.e., Lemma 12.3.9. Therefore B in Equation (12.7) is upper bounded by $2 \cdot 4(d+1)^2 (9m^{-2}) \|P(t)\|_{[-1,1]}^2$.

From all discussion above, $\|Q(t) - P(t)\|_{[-1,1]}^2 \leq \frac{99d^2}{m^2} \leq \epsilon^2$. □

Now we use the above lemma to provide a faster learning algorithm for polynomials on interval $[-1,1]$ with noise instead of using the ϵ-nets argument. Algorithm RobustPolynomialLearningFixedInterval works as follows:

1. Let $\epsilon = 1/20$ and $I_1, \cdots, I_n$ be the intervals for d and ϵ in Lemma 12.4.1.
2. Random choose $x_j \in I_j$ for every $j \in [n]$ and define $S = \{x_1, \cdots, x_n\}$ with weight $w_1 = \frac{|I_1|}{2}, \cdots, w_n = \frac{|I_n|}{2}$.
3. Find the degree d polynomial $Q(t)$ that minimizes $\|P(t) - Q(t)\|_{S,w}$ using Fact 12.11.1.

Lemma 12.4.4. *For any degree d polynomial $P(t)$ and an arbitrary function $g(t)$, Algorithm* RobustPolynomialLearningFixedInterval *takes $O(d)$ samples from $x(t) = P(t) + g(t)$ over $[-1,1]$ and reports a degree d polynomial $Q(t)$ in time $O(d^\omega)$ such that, with probability at least $99/100$,*

$$\|P(t) - Q(t)\|^2_{[-1,1]} \lesssim \|g(t)\|^2_{[-1,1]}.$$

Proof. Notice that $n = O(d/\epsilon) = O(d)$ and the running time depends on solving a linear regression problem(Fact 12.11.1), which takes $O(d^\omega)$ time. It is enough to bound the

distance between P and Q:

$$\begin{aligned}
& \|P-Q\|_{[-1,1]} & \\
\leq\ & 1.09\|P-Q\|_{S,w} & \text{by Corollary 12.4.2} \\
=\ & 1.09\|x-g-Q\|_{S,w} & \text{by } x=P+g \\
\leq\ & 1.09\|g\|_{S,w}+1.09\|x-Q\|_{S,w} & \text{by triangle inequality} \\
\leq\ & 1.09\|g\|_{S,w}+1.09\|x-P\|_{S,w} & Q=\arg\min_{\text{degree-}d\ R}\|R-x\|_{S,w} \\
\leq\ & 2.2\|g\|_{S,w} &
\end{aligned}$$

Because $\mathop{\mathbb{E}}_{S}[\|g\|_{S,w}^2]=\|g\|_{[-1,1]}^2$, we know that $\|P-Q\|_{[-1,1]}\leq 2200\|g\|_{[-1,1]}$ with probability $\geq .999$ by using Markov's inequality. □

For any function $f:[0,T]\to\mathbb{C}$, let $\widetilde{f}(t)=f(\frac{2t-T}{T})$. Then $\|\widetilde{f}\|_{[-1,1]}=\|f\|_T$ from the definition. Hence we can switch any interval $[0,T]$ to $[-1,1]$ and use Lemma 12.4.4.

Theorem 12.1.2. For any degree d polynomial $P(t)$ and an arbitrary function $g(t)$, Procedure RobustPolynomialLearning in Algorithm 12.5 takes $O(d)$ samples from $x(t)=P(t)+g(t)$ over $[0,T]$ and reports a degree d polynomial $Q(t)$ in time $O(d^\omega)$ such that, with probability at least 99/100,

$$\|P(t)-Q(t)\|_T^2\lesssim\|g(t)\|_T^2.$$

where $\omega<2.373$ is matrix multiplication exponent [Str69],[CW87],[Wil12].

12.4.2 Boosting success probability

Notice that the success probability of Theorem 12.1.2 is only constant, and the proof technique of obtaining that result cannot be modified to $1 - 1/\operatorname{poly}(d)$ or $1 - 2^{-\Omega(d)}$ success probability due to using Markov's inequality. However, we can use that algorithm as a black box, and rerun it $O(\log(1/p))$ (for any $p > 0$) times on fresh samples. Using the careful median analysis from [MP14] gives

Theorem 12.4.5. For any degree d polynomial $P(t)$, an arbitrary function $g(t)$, and any $p > 0$, Procedure RobustPolynomialLearning$^+$ in Algorithm 12.5 takes $O(d \log(1/p))$ samples from $x(t) = P(t) + g(t)$ over $[0, T]$ and reports a degree d polynomial $Q(t)$ in time $O(d^\omega \log(1/p))$ such that, with probability at least $1 - p$,

$$\|P(t) - Q(t)\|_T^2 \lesssim \|g(t)\|_T^2.$$

where $\omega < 2.373$ is matrix multiplication exponent.

Proof. We run algorithm RobustPolynomialLearning R rounds with $O(d)$ independent and fresh samples per round. We will obtain R degree-d polynomials $Q_1(t), Q_2(t), \cdots, Q_R(t)$. We say a polynomial $Q_i(t)$ is good if $\|Q_i(t) - P(t)\|_T^2 \lesssim \|g(t)\|_T^2$. Using the Chernoff bound, with probability at least $1 - 2^{-\Omega(R)}$, at least a $3/4$ fraction of the polynomials are "good". We output polynomial $Q(t) = Q_{j^*}(t)$ such that

$$j^* = \arg\min_{j \in [R]} (\text{median}\{\|Q_j(t) - Q_1(t)\|_T^2, \|Q_j(t) - Q_2(t)\|_T^2, \cdots, \|Q_j(t) - Q_R(t)\|_T^2\}) \quad (12.8)$$

The Equation (12.8) can be solved in following straightforward way. For $i \neq j$, it takes $O(d)$ time to compute $\|Q_j(t) - Q_i(t)\|_T^2$. Because of the number of pairs is $O(R^2)$, thus it takes

$O(R^2 d)$ time write down a $R \times R$ matrix. For each column, we run linear time 1-median algorithm. This step takes $O(R^2)$ time. At the end, j^* is index of the column that has the smallest median value. Thus, polynomial $Q(t) = Q_{j^*}(t)$ 1 the 0 with probability at least $1 - p$ by choosing $R = O(\log(1/p))$. The running time is not optimized yet.

To improve the dependence on R for running time, we replace the step of solving Equation (12.8) by an approach that is similar to [MP14]. We choose a new set of samples S, say $S = \{t_1, t_2, \cdots, t_n\}$ and $n = O(d)$. Using Fact 12.11.2, we can compute $Q_i(t_j)$ for all $i, j \in [R] \times [n]$ in $O(Rd \operatorname{poly}(\log(d)))$ time. Define

$$\widetilde{Q}_j = \operatorname*{median}_{i \in [R]} Q_i(t_j), \forall j \in [n]. \tag{12.9}$$

Our algorithm will output a degree-d polynomial Q which is the optimal solution of this problem, $\min_{\text{degree-}d\ Q'} \|Q' - \widetilde{Q}\|_{S,w}$.[2] In the rest of the proof, we will show that $\|Q - P\|_T \lesssim \|g\|_T$ with probability at least $1 - 2^{-\Omega(R)}$.

Notice that Equation (12.9) implies that $\widetilde{Q}_j - P(t_j) = \operatorname*{median}_{i \in [R]}(Q_i(t_j) - P(t_j))$. Fix a coordinate j and applying the proof argument of Lemma 6.1 in [MP14], we have

$$(\widetilde{Q}_j - P(t_j))^2 \lesssim \operatorname*{mean}_{\text{good } i}(Q_i(t_j) - P(t_j))^2$$

Taking the weighted summation over all the coordinates j, we have

$$\|\widetilde{Q} - P\|_{S,w}^2 \lesssim \operatorname*{mean}_{\text{good } i} \|Q_i - P\|_{S,w}^2$$

Using Corollary 12.4.2, for each good i,

$$\|Q_i - P\|_{S,w}^2 \lesssim \|Q_i - P\|_T^2$$

[2] Outputting $Q = \operatorname*{arg\,min}_{\text{degree-}d\ Q'} \|Q' - x\|_{S,w}$ is not good enough, because it only gives constant success probability.

Combining the above two inequalities gives

$$\|\widetilde{Q} - P\|_{S,w}^2 \lesssim \underset{\text{good } i}{\text{mean}} \|Q_i - P\|_T^2 \lesssim \|g\|_T^2 \tag{12.10}$$

Because Q is the optimal solution for $\widetilde{Q}$, then

$$\|\widetilde{Q} - Q\|_{S,w}^2 \leq \|\widetilde{Q} - P\|_{S,w}^2 \lesssim \|g\|_T^2 \tag{12.11}$$

Using Corollary 12.4.2 and for any good i, i', $\|Q_i - Q_{i'}\|_T \lesssim \|g\|_T$, we can replace P by $Q_{i'}$ in the Equation (12.10). Thus, for any $Q_{i'}$ where i' is good,

$$\|\widetilde{Q} - Q_{i'}\|_{S,w}^2 \lesssim \|g\|_T^2 \tag{12.12}$$

For any good i',

$$\begin{aligned}
&\|Q_{i'} - Q\|_T \\
\lesssim\ &\|Q_{i'} - Q\|_{S,w} && \text{by Corollary 12.4.2} \\
\leq\ &\|Q_{i'} - \widetilde{Q}\|_{S,w} + \|\widetilde{Q} - Q\|_{S,w} && \text{by triangle inequality} \\
\lesssim\ &\|g\|_T && \text{by Equation (12.11) and (12.12)}
\end{aligned}$$

Thus, our algorithm takes $O(dR)$ samples from $x(t) = P(t) + g(t)$ over $[0, T]$ and reports a polynomial $Q(t)$ in time $O(Rd^\omega)$ such that, with probability at least $1 - 2^{-\Omega(R)}$, $\|P(t) - Q(t)\|_T^2 \lesssim \|g(t)\|_T^2$. Choosing $R = O(\log(1/p))$ completes the proof. □

12.5 Bounding the Magnitude of a Fourier-sparse Signal in Terms of Its Average Norm

The main results in this section are two upper bounds, Lemma 12.5.1 on $\max_{t\in[0,T]}|x(t)|^2$ and Lemma 12.5.5 on $|x(t)|^2$ for $t > T$, in terms of the typical signal value $\|x\|_T^2 = \frac{1}{T}\int_0^T |x(t)|^2 \mathrm{d}t$. We prove Lemma 12.5.1 in Section 12.5.1 and Lemma 12.5.5 in Section 12.5.2

12.5.1 Bounding the maximum inside the interval

The goal of this section is to prove Lemma 12.5.1.

Lemma 12.5.1. For any k-Fourier-sparse signal $x(t) : \mathbb{R} \to \mathbb{C}$ and any duration T, we have

$$\max_{t\in[0,T]} |x(t)|^2 \lesssim k^4 \log^3 k \cdot \|x\|_T^2$$

Proof. Without loss of generality, we fix $T = 1$. Then $\|x\|_T^2 = \int_0^1 |x(t)|^2 \mathrm{d}t$. Because $\|x\|_T^2$ is the average over the interval $[0, T]$, if $t^* = \arg\max_{t\in[0,T]} |x(t)|^2$ is not 0 or $T = 1$, we can rescale the two intervals $[0, t^*]$ and $[t^*, T]$ to $[0, 1]$ and prove the desired property separately. Hence we assume $|x(0)|^2 = \max_{t\in[0,T]} |x(t)|^2$ in this proof.

Claim 12.5.2. *For any k, there exists $m = O(k^2 \log k)$ such that for any k-Fourier-sparse signal $x(t)$, any $t_0 \geq 0$ and $\tau > 0$, there always exist $C_1, \cdots, C_m \in \mathbb{C}$ such that the following properties hold,*

$$\text{Property I} \quad |C_j| \leq 11 \ \ \textit{for all } j \in [m],$$
$$\text{Property II} \quad x(t_0) = \sum_{j\in[m]} C_j \cdot x(t_0 + j \cdot \tau).$$

We first use this claim to finish the proof of Lemma 12.5.1. We choose $t_0 = 0$ such that $\forall \tau > 0$, there always exist $C_1, \cdots, C_m \in \mathbb{C}$, and

$$x(0) = \sum_{j\in[m]} C_j \cdot x(j \cdot \tau).$$

By the Cauchy-Schwarz inequality, it implies that for any τ,

$$\begin{aligned} |x(0)|^2 &\leq m \sum_{j\in[m]} |C_j|^2 |x(j \cdot \tau)|^2 \\ &\lesssim m \sum_{j\in[m]} |x(j \cdot \tau)|^2. \end{aligned} \tag{12.13}$$

At last, we obtain

$$\begin{aligned}
|x(0)|^2 = & \quad m \int_0^{1/m} |x(0)|^2 \mathrm{d}\tau \\
\lesssim & \quad m \cdot \int_0^{1/m} (m \sum_{j=1}^{m} |x(j \cdot \tau)|^2) \mathrm{d}\tau \\
= & \quad m^2 \cdot \sum_{j=1}^{m} \int_0^{1/m} |x(j \cdot \tau)|^2 \mathrm{d}\tau \\
= & \quad m^2 \cdot \sum_{j=1}^{m} \frac{1}{j} \int_0^{j/m} |x(\tau)|^2 \mathrm{d}\tau \\
\leq & \quad m^2 \cdot \sum_{j=1}^{m} \frac{1}{j} \cdot \int_0^{1} |x(\tau)|^2 \mathrm{d}\tau \\
\lesssim & \quad m^2 \log m \cdot \|x\|_T^2
\end{aligned}$$

where the first inequality follows by Equation (12.13), the second inequality follows by $j/m \leq 1$ and the last step follows by $\sum_{i=1}^{m} \frac{1}{i} = O(\log m)$. From $m = O(k^2 \log k)$, we obtain $|x(0)|^2 = O(k^4 \log^3 k \|x\|_T^2)$. □

To prove Claim 12.5.2, we use the following lemmas about polynomials. We defer their proofs to Section 12.10.2.

Lemma 12.5.3. Let $Q(z)$ be a degree k polynomial, all of whose roots are complex numbers with absolute value 1. For any integer n, let $r_{n,k}(z) = \sum_{l=0}^{k-1} r_{n,k}^{(l)} \cdot z^l$ denote the residual polynomial of

$$r_{n,k}(z) \equiv z^n \pmod{Q(z)}.$$

Then, each coefficient of $r_{n,k}$ is bounded: $|r_{n,k}^{(l)}| \leq 2^k n^{k-1}$ for any l.

Lemma 12.5.4. For any $k \in \mathbb{Z}$ and any $z_1, \cdots, z_k$ on the unit circle of $\mathbb{C}$, there always exists a degree $m = O(k^2 \log k)$ polynomial $P(z) = \sum_{j=0}^{m} c_j z^j$ with the following properties:

$$\begin{aligned} \text{Property I} \quad & P(z_i) = 0, \forall i \in \{1, \cdots, k\}, \\ \text{Property II} \quad & c_0 = 1, \\ \text{Property III} \quad & |c_j| \leq 11, \forall j \in \{1, \cdots, m\}. \end{aligned}$$

Proof of Claim 12.5.2. For $x(t) = \sum_{i=1}^{k} v_i e^{2\pi \mathbf{i} f_i t}$, we fix t_0 and τ then rewrite $x(t_0 + j \cdot \tau)$ as a polynomial of $b_i = v_i \cdot e^{2\pi \mathbf{i} f_i t_0}$ and $z_i = e^{2\pi \mathbf{i} f_i \tau}$ for each $i \in [k]$.

$$\begin{aligned} x(t_0 + j \cdot \tau) &= \sum_{i=1}^{k} v_i e^{2\pi \mathbf{i} f_i \cdot (t_0 + j \cdot \tau)} \\ &= \sum_{i=1}^{k} v_i e^{2\pi \mathbf{i} f_i t_0} \cdot e^{2\pi \mathbf{i} f_i \cdot j\tau} \\ &= \sum_{i=1}^{k} b_i \cdot z_i^j. \end{aligned}$$

Given k and $z_1, \cdots, z_k$, let $P(z) = \sum_{j=0}^{m} c_j z^j$ be the degree m polynomial in Lemma 12.5.4.

$$\begin{aligned} \sum_{j=0}^{m} c_j x(t_0 + j\tau) &= \sum_{j=0}^{m} c_j \sum_{i=1}^{k} b_i \cdot z_i^j \\ &= \sum_{i=1}^{k} b_i \sum_{j=0}^{m} c_j \cdot z_i^j \\ &= \sum_{i=1}^{k} b_i P(z_i) \\ &= 0, \end{aligned} \tag{12.14}$$

where the last step follows by Property I of $P(z)$ in Lemma 12.5.4. From the Property II and III of $P(z)$, we obtain $x(t_0) = -\sum_{j=1}^{m} c_j x(t_0 + j\tau)$. □

12.5.2 Bounding growth outside the interval

Here we show signals with sparse Fourier transform cannot grow too quickly outside the interval.

Lemma 12.5.5. *Let $x(t)$ be a k-Fourier-sparse signal. For any $T > 0$ and any $t > T$,*

$$|x(t)|^2 \leq k^7 \cdot (2kt/T)^{2.5k} \cdot \|x\|_T^2.$$

Proof. For any $t > T$, let $t = t_0 + n \cdot \tau$ such that $t_0 \in [0, T/k], \tau \in [0, T/k]$ and $n \leq \frac{2kt}{T}$. We define $b_i = v_i e^{2\pi \mathbf{i} f_i t_0}$, and $z_i = e^{2\pi \mathbf{i} f_i \tau}$ such that $x(t_0 + n \cdot \tau) = \sum_{j \in [k]} b_j z_j^n$.

By Lemma 12.5.3, we have for any $z_1, z_2, \cdots, z_k$ and any n,

$$z^n \equiv \sum_{i=0}^{k-1} a_i z^i \pmod{\prod_{i=1}^{k}(z - z_i)},$$

where $|a_i| \leq 2^k \cdot n^k, \forall i \in \{0, 1, \cdots, k-1\}$. Thus, we obtain

$$x(t_0 + n\tau) = \sum_{j=1}^{k} b_j z_j^n = \sum_{j=1}^{k} b_j (\sum_{i=0}^{k-1} a_i z_j^i).$$

From the fact that $x(t_0 + i \cdot \tau) = \sum_{j \in [k]} b_j z_j^i$, we simplify it to be

$$x(t_0 + n\tau) = \sum_{i=0}^{k-1} a_i \sum_{j=1}^{k} b_j z_j^i = \sum_{i=0}^{k-1} a_i x(t_0 + i \cdot \tau).$$

Because $(t_0 + i \cdot \tau) \in [0, T]$ for any $i = 0, \cdots, k-1$, we have

$$|x(t_0 + i\tau)|^2 \leq \max_{t \in [0,T]} |x(t)|^2 \lesssim k^4 \log^3 k \|x\|_T^2$$

from Lemma 12.5.1.

Hence

$$\begin{aligned} |x(t_0 + n \cdot \tau)|^2 \leq & \quad k \sum_{i=0}^{k-1} |a_i|^2 \cdot |x(t_0 + i \cdot \tau)|^2 \\ \leq & \quad k \sum_{i=0}^{k-1} n^{2.2k} \cdot \max_{t \in [0,T]} |x(t)|^2 \\ \leq & \quad k^7 \cdot (2kt/T)^{2.2k} \|x\|_T^2. \end{aligned}$$

Thus, we complete the proof. □

12.6 Hash Functions and Filter Functions

12.6.1 Permutation function and hash function

We first review the permutation function $P_{\sigma,a,b}$ and the hash function $h_{\sigma,b}$ in [PS15], which translates discrete settings to the continuous setting.

Definition 12.6.1. For any signal $x(t) : \mathbb{R} \to \mathbb{C}$ and $a, b, \sigma \in \mathbb{R}$, let $(P_{\sigma,a,b}x)(t) = x\big(\sigma(t-a)\big)e^{-2\pi \mathbf{i}\sigma bt}$.

Lemma 12.6.1. $\widehat{P_{\sigma,a,b}x}(\sigma(f-b)) = \frac{1}{\sigma}e^{-2\pi \mathbf{i}\sigma af}\widehat{x}(f)$ and $\widehat{P_{\sigma,a,b}x}(f) = \frac{1}{\sigma}e^{-2\pi \mathbf{i}\sigma a(f/\sigma+b)}\widehat{x}(f/\sigma+b)$

For completeness, we provide a proof of Lemma 12.6.1 in Section 12.10.4.

Definition 12.6.2. [PS15] Let $\pi_{\sigma,b}(f) = 2\pi\sigma(f-b) \pmod{2\pi}$ and $h_{\sigma,b}(f) = \mathrm{round}(\pi_{\sigma,b}(f) \cdot \frac{B}{2\pi})$ be the hash function that maps frequency $f \in [-F, F]$ into bins $\{0, \cdots, B-1\}$.

Claim 12.6.2. *[PS15] For any $\Delta > 0$, let σ be a sample uniformly at random from $[\frac{1}{B\Delta}, \frac{2}{B\Delta}]$.*

(I) If $\Delta \le |f^+ - f^-| < \frac{(B-1)\Delta}{2}$, then $\Pr[h_{\sigma,b}(f^+) = h_{\sigma,b}(f^-)] = 0$

(II) If $\frac{(B-1)\Delta}{2} \le |f^+ - f^-|$, then $\Pr[h_{\sigma,b}(f^+) = h_{\sigma,b}(f^-)] \lesssim \frac{1}{B}$

From previous work [HIKP12b, HIKP12a, PS15], uniformly sampling from $[A, 2A]$ for some large $A \ge \widetilde{T}$ provides an almost uniform sample on $[0, \widetilde{T}]$ when taken modulo over $\widetilde{T}$.

Lemma 12.6.3. *For any $\widetilde{T}$, and $0 \le \widetilde{\epsilon}, \widetilde{\delta} \le \widetilde{T}$, if we sample $\widetilde{\sigma}$ uniformly at random from $[A, 2A]$, then*

$$\frac{2\widetilde{\epsilon}}{\widetilde{T}} - \frac{2\widetilde{\epsilon}}{A} \le \Pr\Big[\widetilde{\sigma} \pmod{\widetilde{T}} \in [\widetilde{\delta} - \widetilde{\epsilon}, \widetilde{\delta} + \widetilde{\epsilon}\,]\Big] \le \frac{2\widetilde{\epsilon}}{\widetilde{T}} + \frac{4\widetilde{\epsilon}}{A}. \tag{12.15}$$

12.6.2 Filter function

We state the properties of filter function $(H(t), \widehat{H}(f))$ and $(G(t), \widehat{G}(f))$, the details of proofs are presented in Section 12.12.1 and 12.12.2.

Lemma 12.6.4. Given $s_0, s_1, 0 < s_3 < 1, \ell > 1, 0 < \delta < 1$, where $\ell = \Theta(k \log(k/\delta))$.The filter function $(H(t), \widehat{H}(f))$ has the following properties,

$$\begin{aligned}
\text{Property I}: \quad & H(t) \in [1-\delta, 1], \text{ when } |t| \leq (\frac{1}{2} - \frac{2}{s_1})s_3. \\
\text{Property II}: \quad & H(t) \in [0, 1], \text{ when } (\frac{1}{2} - \frac{2}{s_1})s_3 \leq |t| \leq \frac{1}{2}s_3. \\
\text{Property III}: \quad & H(t) \leq s_0 \cdot (s_1(\frac{|t|}{s_3} - \frac{1}{2}) + 2)^{-\ell}, \forall |t| > \frac{1}{2}s_3. \\
\text{Property IV}: \quad & \mathrm{supp}(\widehat{H}(f)) \subseteq [-\frac{s_1\ell}{2s_3}, \frac{s_1\ell}{2s_3}].
\end{aligned}$$

For any exact k-Fourier-sparse signal $x^*(t)$, we shift the interval from $[0, T]$ to $[-1/2, 1/2]$ and consider $x^*(t)$ for $t \in [-1/2, 1/2]$ to be our observation, which is also $x^*(t) \cdot \mathrm{rect}_1(t)$.

$$\begin{aligned}
\text{Property V}: \quad & \int_{-\infty}^{+\infty} \left|x^*(t) \cdot H(t) \cdot (1 - \mathrm{rect}_1(t))\right|^2 \mathrm{d}t < \delta \int_{-\infty}^{+\infty} |x^*(t) \cdot \mathrm{rect}_1(t)|^2 \mathrm{d}t. \\
\text{Property VI}: \quad & \int_{-\infty}^{+\infty} |x^*(t) \cdot H(t) \cdot \mathrm{rect}_1(t)|^2 \mathrm{d}t \in [1-\epsilon, 1] \cdot \int_{-\infty}^{+\infty} |x^*(t) \cdot \mathrm{rect}_1(t)|^2 \mathrm{d}t.
\end{aligned}$$

for arbitrarily small constant ϵ.

Lemma 12.6.5. Given $B > 1$, $\delta > 0$, $\alpha > 0$, we set $l = \Omega(\log(\delta/k))$. The filter function

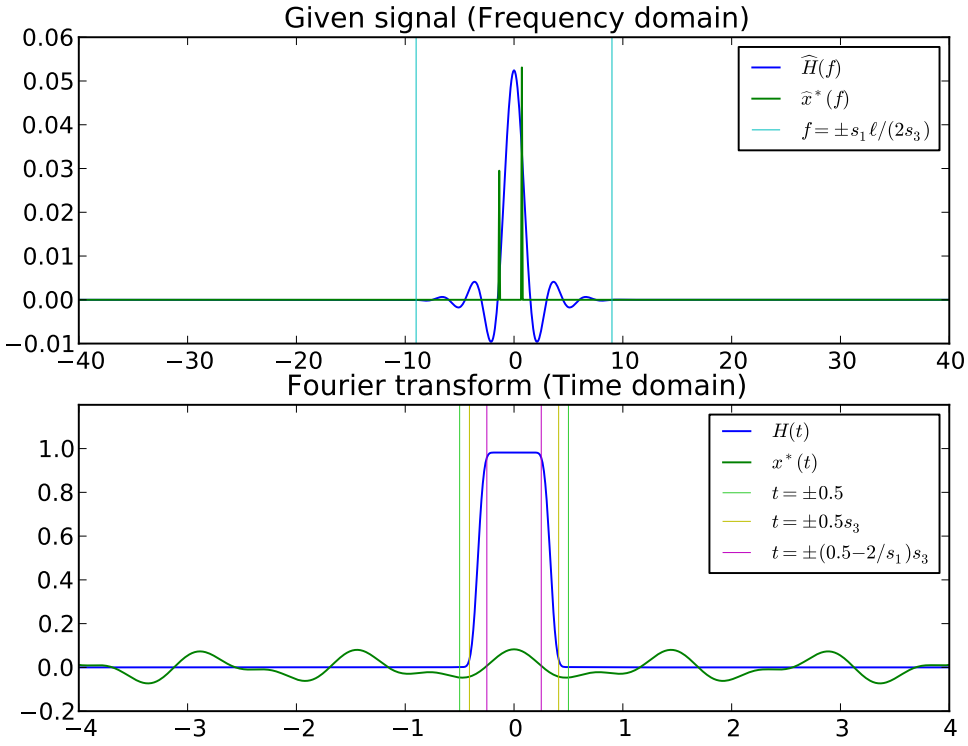


Figure 12.2: The filter function $(H(t), \widehat{H}(f))$ with a k-Fourier-sparse signal. The property I, II and III are presented in the bottom one, the property IV is presented in the top one.

$(G(t), \widehat{G}(f))[B, \delta, \alpha, l]$ satisfies the following properties,

$$\begin{aligned}
\text{Property I}: &\quad \widehat{G}(f) \in [1 - \delta/k, 1], \text{ if } |f| \leq (1-\alpha)\frac{2\pi}{2B}. \\
\text{Property II}: &\quad \widehat{G}(f) \in [0, 1], \text{ if } (1-\alpha)\frac{2\pi}{2B} \leq |f| \leq \frac{2\pi}{2B}. \\
\text{Property III}: &\quad \widehat{G}(f) \in [-\delta/k, \delta/k], \text{ if } |f| > \frac{2\pi}{2B}. \\
\text{Property IV}: &\quad \mathrm{supp}(G(t)) \subset \Big[\frac{l}{2} \cdot \frac{-B}{\pi\alpha}, \frac{l}{2} \cdot \frac{B}{\pi\alpha}\Big]. \\
\text{Property V}: &\quad \max_t |G(t)| \lesssim \mathrm{poly}(B, l).
\end{aligned}$$

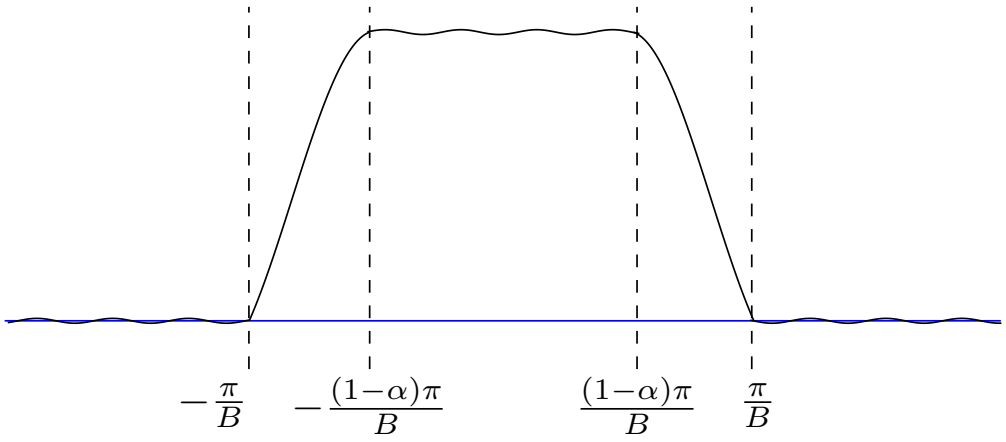


Figure 12.3: G and $\widehat{G}$. [PS15]

12.6.3 HashToBins

We first define two functions $G^{(j)}_{\sigma,b}(t)$ and $\widehat{G}^{(j)}_{\sigma,b}(f)$, then show the result returned by Procedure HashToBins in Algorithm 12.6 satisfying some nice properties. The details of proofs are presented in Section 12.12.4.

Definition 12.6.6. $\forall \sigma > 0, b$ and $j \in [B]$. Define,

$$\begin{aligned} G^{(j)}_{\sigma,b}(t) &= \frac{1}{\sigma} G(t/\sigma) e^{2\pi \mathbf{i} t(j/B - \sigma b)/\sigma} \\ \widehat{G}^{(j)}_{\sigma,b}(f) &= \widehat{G}^{\text{dis}}(\frac{j}{B} - \sigma f - \sigma b) = \sum_{i \in \mathbb{Z}} \widehat{G}(i + \frac{j}{B} - \sigma f - \sigma b) \end{aligned}$$

Lemma 12.6.7. Let $u \in \mathbb{C}^B$ be the result of HashToBins under permutation $P_{\sigma,a,b}$, and let $j \in [B]$. Define

$$\widehat{z} = \widehat{x \cdot H} \cdot \widehat{G}^{(j)}_{\sigma,b},$$

so

$$z = (x \cdot H) * G^{(j)}_{\sigma,b}.$$

Let vector $\widehat{u} \in \mathbb{C}^B$ denote the B-dimensional DFT of u, then $\forall j \in [B]$,

$$\widehat{u}[j] = z_{\sigma a}.$$

12.7 Frequency Recovery

The goal of this section is to prove Theorem 12.2.6, which is able to recover the frequencies of a signal x^* has k-sparse Fourier transform under noise.

Theorem 12.2.6. Let $x^*(t) = \sum_{j=1}^{k} v_j e^{2\pi \mathbf{i} f_j t}$ and $x(t) = x^*(t) + g(t)$ be our observable signal where $\|g(t)\|_T^2 \leq c\|x^*(t)\|_T^2$ for a sufficiently small constant c. Then Procedure FrequencyRecoveryKCluster returns a set L of $O(k)$ frequencies that covers all heavy clusters of x^*, which uses $\mathrm{poly}(k, \log(1/\delta)) \log(FT)$ samples and $\mathrm{poly}(k, \log(1/\delta)) \log^2(FT)$ time. In particular, for $\Delta = \mathrm{poly}(k, \log(1/\delta))/T$ and $\mathcal{N}^2 := \|g(t)\|_T^2 + \delta\|x^*(t)\|_T^2$, with probability $1 - 2^{-\Omega(k)}$, for any f^* with

$$\int_{f^*-\Delta}^{f^*+\Delta} |\widehat{x \cdot H}(f)|^2 \mathrm{d}f \geq T\mathcal{N}^2/k, \tag{12.16}$$

there exists an $\widetilde{f} \in L$ satisfying

$$|f^* - \widetilde{f}| \lesssim \Delta\sqrt{\Delta T}.$$

12.7.1 Overview

We give an overview of proving Theorem 12.2.6. Instead of starting with k-cluster recovery, we first show how to achieve one-cluster recovery.

One-cluster recovery. we start with $x^*(t) = \sum_{j=1}^{k} v_j e^{2\pi \mathbf{i} f_j t}$ where there exists f_0 and Δ such that f_j is in $[f_0 - \Delta, f_0 + \Delta]$ for each $j \in [k]$ and consider its properties for frequency recovery.

Definition 12.7.1 ((ϵ, Δ)-one-cluster signal)**.** We say that a signal $z(t)$ is an (ϵ, Δ)-one-cluster signal around f_0 iff $z(t)$ and $\widehat{z}(f)$ satisfy the following two properties:

$$\begin{aligned} \text{Property I} \quad &: \quad \int_{f_0-\Delta}^{f_0+\Delta} |\widehat{z}(f)|^2 \mathrm{d}f \geq (1-\epsilon) \int_{-\infty}^{+\infty} |\widehat{z}(f)|^2 \mathrm{d}f \\ \text{Property II} \quad &: \quad \int_{0}^{T} |z(t)|^2 \mathrm{d}t \geq (1-\epsilon) \int_{-\infty}^{+\infty} |z(t)|^2 \mathrm{d}t. \end{aligned}$$

The main result of one-cluster recovery is to prove that the two properties in Definition 12.7.1 with a sufficiently small constant ϵ are sufficient to return $\widetilde{f_0}$ close to f_0 with high probability, which provides a black-box for k-cluster recovery algorithm.

We first prove that the pair of conditions, Property I and Property II in Definition 12.7.1, are sufficient to obtain an estimation of $e^{2\pi \mathbf{i} f_0}$ in Section 12.7.2. We also provide the proof of the correctness of Procedures GetLegal1Sample and GetEmpirical1Engergy in Section 12.7.2.

Lemma 12.7.1. *For a sufficiently small constant $\epsilon > 0$, any $f_0 \in [-F, F]$, and $\Delta > 0$, given $\widehat{\beta} \eqsim \frac{1}{\Delta\sqrt{\Delta T}}$ and an (ϵ, Δ)-one-cluster signal $z(t)$ around f_0, Procedure* GetLegal1Sample

in Algorithm 12.3 with any $\beta \le 2\widehat{\beta}$ takes $O((T\Delta)^3)$ samples to output $\alpha \in \mathbb{R}$ satisfying

$$|z(\alpha+\beta) - z(\alpha)e^{2\pi \mathbf{i} f_0 \beta}| \le 0.08(|z(\alpha)| + |z(\alpha+\beta)|),$$

with probability at least 0.6.

The following lemma shows that for any (ϵ, Δ)-one-cluster signal $z(t)$ around f_0, we could use the above procedure to find a frequency $\widetilde{f_0}$ approximating f_0 with high probability.

Lemma 12.7.2. *For a sufficiently small constant $\epsilon > 0$, any $f_0 \in [-F, F]$, and $\Delta > 0$, given an (ϵ, Δ)-one-cluster signal $z(t)$ around f_0 , Procedure* FrequencyRecovery1Cluster *in Algorithm 12.4 returns $\widetilde{f_0}$ with $|\widetilde{f_0} - f_0| \lesssim \Delta \cdot \sqrt{\Delta T}$ with probability at least $1 - 2^{-\Omega(k)}$.*

We provide a proof of Lemma 12.7.2 in Section 12.7.4. We show $z(t) = (x^*(t) + g(t)) \cdot H(t)$ satisfy Properties I and II (Definition 12.7.1) when all frequencies in $\widehat{x^*}$ are in a small range in Section 12.7.3.

Lemma 12.7.3. *For any $f_0 \in [-F, F]$, $\Delta' > 0$, and $x^*(t) = \sum_{j=1}^k v_j e^{2\pi \mathbf{i} f_t}$ with $|f_j - f_0| \le \Delta'$ for all $j \in [k]$, let $x(t) = x^*(t) + g(t)$ be our observable signal whose noise $\|g\|_T^2 \le c\|x^*\|_T^2$ for a sufficiently small constant c and $H(t)$ be the filter function defined in Section 12.6 with $|\operatorname{supp}(\widehat{H})| = \Delta_h$. Then $z = H \cdot x$ is an $(O(\sqrt{c}), \Delta_h + \Delta')$-one-cluster signal around f_0.*

From all discussion above, we summarize the result of frequency recovery when $\widehat{x^*}$ is in one cluster.

Theorem 12.7.4. For any $f_0 \in [-F, F]$, $\Delta' > 0$, and $x^*(t) = \sum_{j=1}^k v_j e^{2\pi \mathbf{i} f_j t}$ with $|f_j - f_0| \le \Delta'$ for all $j \in [k]$, let $x(t) = x^*(t) + g(t)$ be our observable signal whose noise $\|g\|_T^2 \le c\|x^*\|_T^2$ for a sufficiently small constant c and $H(t)$ be the filter function defined in Section 12.6

with $|\mathrm{supp}(\widehat{H})| = \Delta_h$. Then Procedure FrequencyRecovery1Cluster in Algorithm 12.4 with $\Delta = \Delta' + \Delta_h$ takes $\mathrm{poly}(k, \log(1/\delta)) \cdot \log(FT)$ samples, runs in $\mathrm{poly}(k, \log(1/\delta)) \cdot \log^2(FT)$ time, returns a frequency $\widetilde{f}_0$ satisfying $|\widetilde{f}_0 - f_0| \lesssim \Delta\sqrt{\Delta T}$ with probability at least $1 - 2^{-\Omega(k)}$.

k-cluster recovery. Given any $x^*(t) = \sum_{j=1}^{k} v_j e^{2\pi \mathbf{i} f_j t}$, we plan to convolve the filter function $G(t)$ on $x(t) \cdot H(t)$ and use Lemma 12.7.2 as a black box to find a list of frequencies that covers $\{f_1, \cdots, f_k\}$.

We fix $\Delta = \mathrm{poly}(k, \log(1/\delta))/T$, $B = \Theta(k)$ and sample σ uniformly at random from $[\frac{1}{B\Delta}, \frac{2}{B\Delta}]$ for k-cluster recovery. We will cover all $f^* \in [-F, F]$ with the following property :

$$\int_{f^*-\Delta}^{f^*+\Delta} |\widehat{x \cdot H}(f)|^2 \mathrm{d}f \geq T\mathcal{N}^2/k, \tag{12.17}$$

We consider one frequency $f^* \in [-F, F]$ satisfying (12.17) and use $j = h_{\sigma,b}(f^*)$ to denote its index in $[B]$ after hashing (σ, b). Recall that for $j \in [B]$, any $\sigma > 0$ and any b,

$$G^{(j)}_{\sigma,b}(t) = \frac{1}{\sigma} G(t/\sigma) e^{2\pi \mathbf{i} t(j/B - \sigma b)/\sigma} \text{ such that } \widehat{G}^{(j)}_{\sigma,b}(f) = \sum_{i \in \mathbb{Z}} \widehat{G}(i + \frac{j}{B} - \sigma f - \sigma b).$$

We set $\widehat{z} = \widehat{x \cdot H} \cdot \widehat{G}^{(j)}_{\sigma,b}$ and $z = (x \cdot H) * G^{(j)}_{\sigma,b}$ for f^* and $j = h_{\sigma,b}(f^*)$. In Section 12.7.5, we show that with high probability over the hashing (σ, b), $(z, \widehat{z})$ satisfies Property I with $[f^* - \Delta, f^* + \Delta]$ and Property II in Definition 12.7.1 such that we could use Lemma 12.7.2 on z to recover f^*.

Lemma 12.7.5. *Let $f^* \in [-F, F]$ satisfy (12.17). For a random hashing (σ, b), let $j = h_{\sigma,b}(f^*)$ be the bucket that f^* maps to under the hash such that $z = (x \cdot H) * G^{(j)}_{\sigma,b}$ and $\widehat{z} = \widehat{x \cdot H} \cdot \widehat{G}^{(j)}_{\sigma,b}$. With probability at least 0.9, $z(t)$ is an (ϵ, Δ)-one-cluster signal around f^* .*

Combining Lemma 12.7.5 and Lemma 12.7.2, we could recover any heavy frequency f^* satisfying (12.17) with probability at least 0.8. Then we repeat this procedure to guarantee that we cover all heavy frequencies and finish the proof of the main frequency recovery Theorem 12.2.6 in Section 12.7.6.

12.7.2 Analysis of GetLegal1Sample and GetEmpirical1Energy

Let $I = [f_0 - \Delta, f_0 + \Delta]$ and $\overline{I} = (-\infty, +\infty) \setminus I$ in this proof. We define $\left(z^I(t), \widehat{z}^I(f)\right)$ and $\left(z^{\overline{I}}(t), \widehat{z}^{\overline{I}}(f)\right)$ as follows:

$$\widehat{z}^I(f) = \begin{cases} \widehat{z}(f) & \text{if } f \in I \\ 0 & \text{if } f \in \overline{I} \end{cases}, \quad \widehat{z}^{\overline{I}}(f) = \begin{cases} 0 & \text{if } f \in I \\ \widehat{z}(f) & \text{if } f \in \overline{I} \end{cases}$$

We consider $z^I(t)$ as the "signal" to recover f_0 and treat $z^{\overline{I}}(t)$ as the "noise". We first show some basic properties of $z^I(t)$.

Claim 12.7.6. *For $z^{\overline{I}}(t)$, we have $\int_0^T |z^{\overline{I}}(t)|^2 \mathrm{d}t \le \epsilon \int_{-\infty}^{+\infty} |z(t)|^2 \mathrm{d}t$. For $z^I(t)$, we have*

$$\int_0^T |z^I(t)|^2 \mathrm{d}t \ge (1 - 5\sqrt{\epsilon}) \int_{-\infty}^{+\infty} |z(t)|^2 \mathrm{d}t \text{ and } \int_0^T |z^I(t)|^2 \mathrm{d}t \ge (1 - 6\sqrt{\epsilon}) \int_{-\infty}^{+\infty} |z^I(t)|^2 \mathrm{d}t.$$

Proof. From the definition and Property I in Definition 12.7.1, we know

$$z(t) = z^I(t) + z^{\overline{I}}(t) \quad \text{and} \quad \int_{-\infty}^{+\infty} |\widehat{z}^{\overline{I}}(f)|^2 \mathrm{d}f \le \epsilon \int_{-\infty}^{+\infty} |\widehat{z}(f)|^2 \mathrm{d}f.$$

Notice that Property I(in Definition 12.7.1) indicates that

$$\int_0^T |z^{\overline{I}}(t)|^2 \mathrm{d}t \le \int_{-\infty}^{+\infty} |z^{\overline{I}}(t)|^2 \mathrm{d}t = \int_{-\infty}^{+\infty} |\widehat{z}^{\overline{I}}(f)|^2 \mathrm{d}f \le \epsilon \int_{-\infty}^{+\infty} |\widehat{z}(f)|^2 \mathrm{d}f.$$

On the other hand, from Property II(in Definition 12.7.1), we know

$$\begin{aligned} (1 - \epsilon) \int_{-\infty}^{+\infty} |z(t)|^2 \mathrm{d}t \le & \int_0^T |z^I(t) + z^{\overline{I}}(t)|^2 \mathrm{d}t \\ \le & \int_0^T |z^I(t)|^2 \mathrm{d}t + 2 \int_0^T |z^I(t)| \cdot |z^{\overline{I}}(t)| \mathrm{d}t + \int_0^T |z^{\overline{I}}(t)|^2 \mathrm{d}t. \end{aligned}$$

We have $\int_0^T |z^I(t)|^2 \mathrm{d}t \le 2 \int_{-\infty}^{+\infty} |z(t)|^2 \mathrm{d}t$ from the above inequality. From $\int_0^T |z^{\overline{I}}(t)|^2 \mathrm{d}t \le \epsilon \int_{-\infty}^{+\infty} |\widehat{z}(f)|^2 \mathrm{d}f$, we bound

$$\int_0^T |z^I(t)| \cdot |z^{\overline{I}}(t)| \mathrm{d}t \le \sqrt{2\epsilon} \int_{-\infty}^{+\infty} |z(t)|^2 \mathrm{d}t$$

by the Cauchy-Schwartz inequality and have

$$\int_0^T |z^I(t)|^2 \mathrm{d}t \geq (1 - 5\sqrt{\epsilon}) \int_{-\infty}^{+\infty} |z(t)|^2 \mathrm{d}t. \tag{12.18}$$

Because $\int_{-\infty}^{+\infty} |z^{\bar{I}}(t)|^2 \mathrm{d}t \leq \epsilon \int_{-\infty}^{+\infty} |z(t)|^2 \mathrm{d}t$, inequality (12.18) also indicates that

$$\int_0^T |z^I(t)|^2 \mathrm{d}t \geq (1 - 6\sqrt{\epsilon}) \int_{-\infty}^{+\infty} |z^I(t)|^2 \mathrm{d}t.$$

□

One useful property of $z^I(t)$ is that its maximum can be bounded by its average on $[0, T]$.

Claim 12.7.7. $\forall t \in [0, T], |z^I(t)| \leq 2\sqrt{\Delta T} \cdot \|z^I\|_T$.

Proof. From the definition $|z^I(t)|$, it is upper bounded by $\int_{f_0-\Delta}^{f_0+\Delta} |\widehat{z^I}(f)| \mathrm{d}f$ for any $t \in [0, T]$. On the other hand,

$$\begin{aligned}
\int_{f_0-\Delta}^{f_0+\Delta} |\widehat{z}^I(f)| \mathrm{d}f &\leq \sqrt{2\Delta} \left(\int_{f_0-\Delta}^{f_0+\Delta} |\widehat{z}^I(f)|^2 \mathrm{d}f \right)^{1/2} \\
&= \sqrt{2\Delta} \left(\int_{-\infty}^{+\infty} |z^I(t)|^2 \mathrm{d}t \right)^{1/2} \\
&\leq 2\sqrt{\Delta} \left(\int_0^T |z^I(t)|^2 \mathrm{d}t \right)^{1/2} \\
&= 2\sqrt{\Delta T} \|z^I\|_T.
\end{aligned}$$

□

Claim 12.7.8. *Given $\widehat{\beta} = \frac{C_\beta}{\Delta \cdot \sqrt{\Delta T}}$ with a sufficiently small constant C_β, for any two $\widehat{\beta}$-close samples in $z^I(t)$, we have that*

$$\forall \alpha \in [0, T],\ \forall \beta \in [\widehat{\beta}, 2\widehat{\beta}], \quad |z^I(\alpha) e^{2\pi \mathbf{i} f_0 \beta} - z^I(\alpha + \beta)| \leq 0.01 \cdot \|z^I\|_T.$$

Proof. From the definition of the Fourier transform, we have

$$\begin{aligned}
|z^I(a+\beta) - z^I(a)e^{2\pi \mathbf{i} f_0 \beta}| &= \left| \int_{f_0-\Delta}^{f_0+\Delta} \widehat{z^I}(f) e^{2\pi \mathbf{i}(fa+f_0\beta)} (e^{2\pi \mathbf{i}(f-f_0)\beta} - 1) \mathrm{d}f \right| \\
&\leq 2 \cdot (2\pi\Delta\beta) \cdot \int_{f_0-\Delta}^{f_0+\Delta} |\widehat{z^I}(f)| \mathrm{d}f && \text{by Taylor expansion} \\
&\leq 4\pi\beta\Delta \cdot \sqrt{2\Delta} \left(\int_{f_0-\Delta}^{f_0+\Delta} |\widehat{z^I}(f)|^2 \mathrm{d}f \right)^{\frac{1}{2}} && \text{by Hölder inequality} \\
&\leq 10\pi\widehat{\beta}\Delta \cdot \sqrt{2\Delta} \left(\int_0^T |z^I(t)|^2 \mathrm{d}t \right)^{\frac{1}{2}} && \text{by inequality (12.18)} \\
&\leq 10^{-2} \|z^I\|_T.
\end{aligned}$$

□

We consider how to output an α such that $e^{2\pi \mathbf{i} f_0 \beta} \approx z(\alpha+\beta)/z(\alpha)$ with high probability in the rest of this section.

If we can sample from $z^I(t)$, we already know $|z^I(\alpha)e^{2\pi \mathbf{i} f_0 \beta} - z^I(\alpha+\beta)| \leq 0.01\|z^I\|_T$ from Claim 12.7.8. Then it is enough to find any α such that $|z^I(\alpha)| \geq 0.5\|z^I\|_T$. From Claim 12.7.7, we can take $O(\sqrt{\Delta T})$ samples $\left(z^I(\alpha), z^I(\alpha+\beta)\right)$ where each α is uniformly sampled from $[0,T]$ such that with high probability, the sample $z^I(\alpha)$ with the largest norm $|z^I(\alpha)|$ satisfies $|z^I(\alpha)| \geq 0.5\|z^I\|_T$. Then we have $e^{2\pi \mathbf{i} f_0 \beta} \approx z^I(\alpha+\beta)/z^I(\alpha)$.

Next, we move to $z(t) = z^I(t) + z^{\overline{I}}(t)$ and plan to output $\alpha \in [0,T]$ with probability at least 0.5 such that $|z^{\overline{I}}(\alpha)| \leq 0.1|z^I(\alpha)|$ and $|z^{\overline{I}}(\alpha+\beta)| \leq 0.1|z^I(\alpha+\beta)|$. Because the "noise" $z^{\overline{I}}(t)$ has $\|z^{\overline{I}}(t)\|_T^2 \geq \epsilon\|z^I(t)\|_T^2$ for a constant ϵ and the bound $\sqrt{\Delta T}$ in Claim 12.7.7 is a polynomial in k, the approach for $z^I(t)$ cannot guarantee that $z(\alpha+\beta)/z(\alpha) \approx e^{2\pi \mathbf{i} f_0 \beta}$ with probability more than $1/2$.

The key observation is as follows:

Observation 12.7.9. *For a sufficiently small ϵ and $\|z^{\overline{I}}\|_T^2 \le \epsilon\|z\|_T^2$, let D_T be the weighted distribution on $[0,T]$ according to $|z(t)|^2$, i.e., $D_T(t) = \frac{|z(t)|^2}{T\|z\|_T^2}$. If we sample $\alpha \in [0,T]$ from the distribution D_T instead of the uniform distribution on $[0,T]$, $|z^{\overline{I}}(\alpha)| \le 0.01|z^I(\alpha)|$ with probability 0.9.*

It follows from the fact that

$$\mathop{\mathbb{E}}_{\alpha\sim D_T}\frac{|z^{\overline{I}}(\alpha)|^2}{|z(\alpha)|^2} = \int_0^T \frac{|z^{\overline{I}}(\alpha)|^2}{|z(\alpha)|^2}\cdot\frac{|z(\alpha)|^2}{T\|z\|_T^2}\mathrm{d}\alpha = \frac{\int_0^T |z^{\overline{I}}(\alpha)|^2\mathrm{d}\alpha}{T\|z\|_T^2} \le \epsilon.$$

In Procedure GetLegal1Sample, we collect $(\Delta T)^2$ samples (in expectation) $\big(z(\alpha), z(\alpha+\beta)\big)$ in S_{heavy} with $|z(\alpha)| \ge 0.49\|z\|_T$ and resample one α from these samples according to their norm $|z(\alpha)|^2 + |z(\alpha+\beta)|^2$. We show its correctness as follows.

Because we do not know $0.5\|z\|_T$, we use z_{emp} to approximate it.

Claim 12.7.10. *Procedure* GetEmpirical1Energy *in Algorithm 12.3 takes $O((T\Delta)^2)$ samples to output z_{emp} such that $z_{\text{emp}} \in [0.8\|z\|_T, 1.2\|z\|_T]$ with prob. 0.9.*

Proof. We know $z_{\text{emp}}^2 = \mathbb{E}_{i\in[R_{\text{est}}]}[|z(\alpha_i)|^2] = \mathbb{E}_{i\in[R_{\text{est}}]}[|z^I(\alpha_i) + z^{\overline{I}}(\alpha_i)|^2]$.

Notice that $\mathbb{E}_{i\in[R_{\text{est}}]}[|z^I(\alpha_i)|^2]$ is in $[0.99\|z^I\|_T, 1.01\|z^I\|_T]$ with prob. 0.99 from the Chernoff bound and Claim 12.7.7.

At the same time, $\mathbb{E}_{\alpha_i}[|z^{\overline{I}}(\alpha_i)|^2] = \|z^{\overline{I}}\|_T^2$. With prob. 0.92, $\mathbb{E}_{i\in[R_{\text{est}}]}[|z^{\overline{I}}(\alpha_i)|^2] \le 13\|z^{\overline{I}}\|_T^2$. For a sufficiently small ϵ and $\|z^{\overline{I}}\|_T^2 \le \epsilon\|z^I\|_T^2$, $\mathbb{E}_{i\in[R_{\text{est}}]}[|z^{\overline{I}}(\alpha_i)|^2] \le 13\epsilon\|z^I\|_T^2$.

At last, we bound the cross terms of $|z^I(\alpha_i) + z^{\overline{I}}(\alpha_i)|^2$ by the Cauchy-Schwartz in-

equality,

$$\begin{aligned}
&\mathop{\mathbb{E}}_{i\in R_{\text{est}}}[|\overline{z}^I(\alpha_i)z^{\overline{I}}(\alpha_i)| + |z^I(\alpha_i)\overline{z}^{\overline{I}}(\alpha_i)|] \\
&\le 2\mathop{\mathbb{E}}_{i\in R_{\text{est}}}[|z^I(\alpha_i)|\cdot|z^{\overline{I}}(\alpha_i)|] \\
&\le 2\left(\mathop{\mathbb{E}}_{i\in[R_{\text{est}}]}[|z^I(\alpha_i)|^2]\cdot\mathop{\mathbb{E}}_{i\in[R_{\text{est}}]}[|z^{\overline{I}}(\alpha_i)|^2]\right)^{1/2} \\
&\le 10\sqrt{\epsilon}\|z^I\|_T^2.
\end{aligned}$$

For a sufficiently small ϵ, we have $\mathbb{E}_{i\in[R_{\text{est}}]}[|z(\alpha_i)|^2]^{1/2}$ is in $[0.9\|z^I\|_T, 1.1\|z^I\|_T]$, which is also in $[0.8\|z\|_T, 1.2\|z\|_T]$ because of Property II. □

We assume $z_{\text{emp}} \in [0.8\|z\|_T, 1.2\|z\|_T]$ and focus on $U = \{t \in [0,T] \big| |z(t)| \ge 0.5 z_{\text{emp}}\}$. Notice that

$$\int_U |z(t)|^2 \mathrm{d}t = \int_0^T |z(t)|^2 \mathrm{d}t - \int_{[0,T]\setminus U} |z(t)|^2 \mathrm{d}t \ge (1-0.6^2)\int_0^T |z(t)|^2 \mathrm{d}t.$$

Let $R_{\text{heavy}} = |S_{\text{heavy}}|$. From Claim 12.7.7 and ϵ, $\mathbb{E}[R_{\text{heavy}}] \ge R_{\text{repeat}}/(T\Delta)$. So we assume $R_{\text{heavy}} \ge 0.01 R_{\text{repeat}}/(T\Delta) = 0.01(T\Delta)^2$ in the rest of this section and think each $\alpha_i \in S_{\text{heavy}}$ is a uniform sample from U over the randomness on S_{heavy}.

Claim 12.7.11. *With probability 0.95,* $\sum_{i\in S_{\text{heavy}}}(|z^{\overline{I}}(\alpha_i)|^2 + |z^{\overline{I}}(\alpha_i+\beta)|^2) \le 10^{-4}\sum_{i\in S_{\text{heavy}}}(|z(\alpha_i)|^2 + |z(\alpha_i+\beta)|^2)$ *for a sufficiently small* ϵ *and* $\|z^{\overline{I}}\|_T^2 \le \epsilon\|z\|_T^2$.

Proof. At first,

$$\mathop{\mathbb{E}}_{S_{\text{heavy}}}\left[\sum_{i\in S_{\text{heavy}}}(|z(\alpha_i)|^2 + |z(\alpha_i+\beta)|^2)\right] \ge R_{\text{heavy}}\cdot\mathop{\mathbb{E}}_{t\sim U}[|z(t)|^2] = R_{\text{heavy}}\cdot\frac{\int_U |z(t)|^2\mathrm{d}t}{|U|}.$$

At the same time,

$$\mathop{\mathbb{E}}_{S_{\text{heavy}}}\left[\sum_{i\in S_{\text{heavy}}}[|z^{\overline{I}}(\alpha_i)|^2+|z^{\overline{I}}(\alpha_i+\beta)|^2]\right]=R_{\text{heavy}}\cdot\mathop{\mathbb{E}}_{t\sim U}[|z^{\overline{I}}(t)|^2+|z^{\overline{I}}(t+\beta)|^2]\leq\frac{2\int_T^0|z^{\overline{I}}(t)|^2\mathrm{d}t}{|U|}.$$

From $\int_U|z(t)|^2\mathrm{d}t\geq 0.64\int_0^T|z(t)|^2\mathrm{d}t$ and $\int_T^0|z^{\overline{I}}(t)|^2\mathrm{d}t\leq\epsilon\int_0^T|z(t)|^2\mathrm{d}t$, we get the conclusion. □

We assume all results in the above claims hold and prove that the sample from S_{heavy} is a good sample such that $z^{\overline{I}}(\alpha)$ is small.

Claim 12.7.12. *If we sample $i\in S_{\text{heavy}}$ according to the weight $|z(\alpha_i)|^2+|z(\alpha_i+\beta)|^2$, with prob. at least 0.9, $|z^{\overline{I}}(\alpha_i)|+|z^{\overline{I}}(\alpha_i+\beta)|\leq 0.05(|z(\alpha_i)|+|z(\alpha_i+\beta)|)$.*

Proof. Similar to the proof of the key observation, we compute the expectation of $\frac{|z^{\overline{I}}(\alpha_i)|^2+|z^{\overline{I}}(\alpha_i+\beta)|^2}{|z(\alpha_i)|^2+|z(\alpha_i+\beta)|^2}$ over the sampling in S_{heavy}:

$$\begin{aligned}&\sum_{i\in S_{\text{heavy}}}\frac{|z(\alpha_i)|^2+|z(\alpha_i+\beta)|^2}{\sum_{j\in S_{\text{heavy}}}|z(\alpha_j)|^2+|z(\alpha_j+\beta)|^2}\cdot\frac{|z^{\overline{I}}(\alpha_i)|^2+|z^{\overline{I}}(\alpha_i+\beta)|^2}{|z(\alpha_i)|^2+|z(\alpha_i+\beta)|^2}\\ =\;&\frac{\sum_{i\in S_{\text{heavy}}}|z^{\overline{I}}(\alpha_i)|^2+|z^{\overline{I}}(\alpha_i+\beta)|^2}{\sum_{i\in S_{\text{heavy}}}|z(\alpha_i)|^2+|z(\alpha_i+\beta)|^2}\\ \leq\;&10^{-4}.\end{aligned}$$

By Markov's inequality, when we sample $i\in S_{\text{heavy}}$ according to the weight $|z(\alpha_i)|^2+|z(\alpha_i+\beta)|^2$, $\frac{|z^{\overline{I}}(\alpha_i)|^2+|z^{\overline{I}}(\alpha_i+\beta)|^2}{|z(\alpha_i)|^2+|z(\alpha_i+\beta)|^2}\leq 10^{-3}$ with probability 0.9. We have that with prob. at least 0.9, $|z^{\overline{I}}(\alpha_i)|+|z^{\overline{I}}(\alpha_i+\beta)|\leq 0.05(|z(\alpha_i)|+|z(\alpha_i+\beta)|)$. □

We assume all above claims hold and finish the proof by setting $\alpha = \alpha_i$. From Claim 12.7.8, we know that

$$|z^I(\alpha)e^{2\pi \mathbf{i} f_0 \beta} - z^I(\alpha+\beta)| \leq 0.01 \cdot \mathop{\mathbb{E}}_{t\in[0,T]}[|z^I(\alpha)|^2]^{1/2} \leq 0.03|z^I(\alpha)|.$$

Now we add back the noise $z^{\overline{I}}(\alpha)$ and $z^{\overline{I}}(\alpha+\beta)$ to get

$$\begin{aligned}|z(\alpha)e^{2\pi \mathbf{i} f_0 \beta} - z(\alpha+\beta)| &\leq |z^I(\alpha)e^{2\pi \mathbf{i} f_0 \beta} - z^I(\alpha+\beta)| + |z^{\overline{I}}(\alpha)| + |z^{\overline{I}}(\alpha+\beta)| \\ &\leq 0.08(|z(\alpha)| + |z(\alpha+\beta)|).\end{aligned}$$

12.7.3 A cluster of frequencies, times H, is a one-cluster signal per Definition 12.7.1

The goal of this section is to prove Lemma 12.7.3. Without loss of generality, we assume $g(t) = 0$ for any $t \notin [0, T]$ and notice that $\mathrm{supp}(\widehat{H} * \widehat{x^*}) \subseteq f_0 + [-\Delta, \Delta]$ for $\Delta = \Delta' + \Delta_h$ from the definition of $\widehat{H}$. From the Property VI (presented in Lemma 12.6.4) of $(H, \widehat{H})$,

$$\int_0^T |x^*(t)|^2 \mathrm{d}t = (1 \pm c) \int_{-\infty}^{+\infty} |H(t) \cdot x^*(t)|^2 \mathrm{d}t.$$

From the first two properties of $(H, \widehat{H})$, we bound the energy of $g \cdot H$:

$$\int_{-\infty}^{+\infty} |H(t) \cdot g(t)|^2 \mathrm{d}t \leq (1 + c) \int_0^T |g(t)|^2 \mathrm{d}t.$$

Let $z(t) = (x^*(t) + g(t))H(t)$. We use the triangle inequality on the above two inequalities:

$$\begin{aligned}
&\int_0^T |z(t)|^2 \mathrm{d}t \\
\geq\ & \int_0^T |H(t) \cdot x^*(t)|^2 \mathrm{d}t - \int_0^T |H(t) \cdot g(t)|^2 \mathrm{d}t - 2 \int_0^T |H(t) \cdot x^*(t)| \cdot |H(t) \cdot g(t)| \mathrm{d}t \\
\geq\ & (1 - c) \int_0^T |x^*(t)|^2 \mathrm{d}t - (1 + c) \int_0^T |g(t)|^2 \mathrm{d}t - 2 \int_0^T |H(t) \cdot x^*(t)| \cdot |H(t) \cdot g(t)| \mathrm{d}t \\
\geq\ & (1 - c) \int_0^T |x^*(t)|^2 \mathrm{d}t - (1 + c) \int_0^T |g(t)|^2 \mathrm{d}t - 2 \left((1 + c)^2 \int_0^T |g(t)|^2 \mathrm{d}t \int_0^T |x^*(t)|^2 \mathrm{d}t \cdot \right)^{1/2} \\
\geq\ & (1 - c) \int_0^T |x^*(t)|^2 \mathrm{d}t - (1 + c) c \int_0^T |x^*(t)|^2 \mathrm{d}t - 2\sqrt{c}(1 + c) \int_0^T |x^*(t)|^2 \mathrm{d}t \\
\geq\ & \left(1 - 5\sqrt{c}\right) \int_0^T |x^*(t)|^2 \mathrm{d}t,
\end{aligned}$$

where the third step follows from Cauchy-Schwarz inequality, the fourth step follows from $\int_0^T |g(t)|^2 \mathrm{d}t \leq c \int_0^T |x^*(t)|^2 \mathrm{d}t$, and the last step follows from choosing $c \in (0, 1)$ sufficiently small.

Similarly,

$$\begin{aligned}
&\int_{-\infty}^{+\infty} |z(t)|^2 \mathrm{d}t \\
&\leq (1+c)\int_0^T |x^*(t)|^2\mathrm{d}t + (1+c)\int_0^T |g(t)|^2\mathrm{d}t + 2\left((1+c)^2\int_0^T |x^*(t)|^2\mathrm{d}t\int_0^T |g(t)|^2\mathrm{d}t\right)^{1/2} \\
&\leq (1+5\sqrt{c})\int_0^T |x^*(t)|^2\mathrm{d}t.
\end{aligned}$$

Hence we obtain Property II(in Definition 12.7.1) when $c \in (0,1)$ is sufficiently small.

Then we observe that

$$\begin{aligned}
&\int_{f_0-\Delta_h}^{f_0+\Delta_h} |\widehat{z}(f)|^2\mathrm{d}f \\
&\geq \int_{f_0-\Delta_h}^{f_0+\Delta_h} |H\cdot\widehat{(x^*+g)}|^2\mathrm{d}f \\
&\geq \int_{f_0-\Delta_h}^{f_0+\Delta_h} |\widehat{H\cdot x^*}|^2 - |\widehat{H\cdot g}|^2 - 2|\widehat{H\cdot x^*}|\cdot|\widehat{H\cdot g}|\mathrm{d}f \\
&\geq \int_{f_0-\Delta_h}^{f_0+\Delta_h} |\widehat{H\cdot x^*}|^2\mathrm{d}f - \int_{-\infty}^{+\infty} |\widehat{H\cdot g}|^2\mathrm{d}f - 2\left(\int_{f_0-\Delta_h}^{f_0+\Delta_h} |\widehat{H\cdot x^*}|^2\mathrm{d}f\int_{-\infty}^{+\infty} |\widehat{H\cdot g}|^2\mathrm{d}f\right)^{1/2} \\
&= \int_{-\infty}^{+\infty} |H\cdot x^*|^2\mathrm{d}t - \int_{-\infty}^{+\infty} |H\cdot g|^2\mathrm{d}t - 2\left(\int_{f_0-\Delta_h}^{f_0+\Delta_h} |H\cdot x^*|^2\mathrm{d}t\int_{-\infty}^{+\infty} |H\cdot g|^2\mathrm{d}t\right)^{1/2} \\
&\geq \frac{(1-c)-c(1+c)-3\sqrt{c}}{1+5\sqrt{c}}\int_{-\infty}^{+\infty} |z(t)|^2\mathrm{d}t.
\end{aligned}$$

Thus we have Property I(in Definition 12.7.1) for z.

12.7.4 Frequency recovery of one-cluster signals

The goal of this section is prove Theorem 12.7.4. We first show the correctness of Procedure LOCATE1INNER. Second, we analyze the Procedure LOCATE1SIGNAL. At end, we rerun Procedure LOCATE1SIGNAL and use median analysis to boost the constant success probability.[3]

Lemma 12.7.13. *Let $f_0 \in \text{region}(q')$. Let β is sampled from $[\frac{st}{4\Delta}, \frac{st}{2\Delta l}]$ and let γ denote the output of Procedure* GETLEGAL1SAMPLE *in Algorithm 12.4. Then using the pair of samples $z(\gamma+\beta)$ and $z(\gamma)$, we have*

I. for the q' with probability at least $1-s$, $v_{q'}$ will increase by one.

II. for any q such that $|q-q'|>3$, with probability at least $1-15s$, v_q will not increase.

Proof. We replace f_0 by θ in the rest of the proof. By Lemma 12.7.1, we have that for any $\widehat{\beta} \le \beta \le 2\widehat{\beta}$, Procedure GETLEGAL1SAMPLE outputs a $\gamma \in [0,T]$ satisfying

$$|z(\gamma+\beta) - z(\gamma)e^{2\pi \mathbf{i} f_0 \beta}| \le 0.1(|z(\gamma)| + |z(\gamma+\beta)|)$$

with probability at least 0.6.

Furthermore, there exists such some constant $g \in (0,1)$ such that with probability $1-g$,

$$\|\phi(z(\gamma+\beta)) - (\phi(z(\gamma)) - 2\pi\beta\theta)\|_{\bigcirc} \lesssim \sin^{-1}(\frac{1}{g}),$$

where $\|x-y\|_{\bigcirc} = \min_{z\in\mathbb{Z}} |x-y+2\pi z|$ denote the "circular distance" between x and y. We can set $s = \Theta(g^{-1})$. There exists some constant $p = \Theta(s)$, with probability at least $1-p$,

$$\|o - 2\pi\beta\theta\|_{\bigcirc} < s\pi/2$$

[3]The proofs in this section are identical to [HIKP12a] and [PS15].

where $o := \phi(z(\gamma+\beta)/z(\gamma))$. The above equation shows that o is a good estimate for $2\pi\beta\theta$ with good probability. We will now show that this means the true region $Q_{q'}$ gets a vote with large probability.

For each q' with $\theta \in [l - \frac{\Delta l}{2} + \frac{q'-1}{t}\Delta l, l - \frac{\Delta l}{2} + \frac{q'}{t}\Delta l] \subset [-F, F]$, we have that $\theta_{q'} = l - \frac{\Delta l}{2} + \frac{q'-0.5}{t}\Delta l$ satisfies that

$$\theta - \theta_{q'} \leq \frac{\Delta l}{2t}.$$

Note that we sample β uniformly at random from $[\widehat{\beta}, 2\widehat{\beta}]$, then $2\widehat{\beta} = \frac{st}{2\Delta l} \leq \frac{cT}{10A^{\frac{3}{2}}}$ (Note that A is some constant > 1), which implies that $2\pi\beta\frac{\Delta l}{2t} \leq \frac{s\pi}{2}$. Thus, we can show the observation o is close to the true region in the following sense,

$$\begin{aligned}
& \|o - 2\pi\beta\theta_{q'}\|_{\bigcirc} \\
\leq\ & \|o - 2\pi\beta\theta\|_{\bigcirc} + \|2\pi\beta\theta - 2\pi\beta\theta_{q'}\|_{\bigcirc} \text{ by triangle inequality} \\
\leq\ & \frac{s\pi}{2} + 2\pi\|\beta\theta - \beta\theta_{q'}\|_{\bigcirc} \\
\leq\ & s\pi.
\end{aligned}$$

Thus, $v_{q'}$ will increase in each round with probability at least $1 - s$.

On the other side, consider q with $|q - q'| > 3$. Then $|\theta - \theta_q| \geq \frac{7\Delta l}{2t}$, and (assuming $\beta \geq \frac{st}{4\Delta l}$) we have

$$2\pi\beta|\theta - \theta_q| \geq 2\pi\frac{st}{4\Delta l}|\theta - \theta_q| = \frac{s\pi t}{2\Delta l}|\theta - \theta_q| \geq \frac{7s\pi}{4} > \frac{3s\pi}{2}.$$

There are two cases: $|\theta - \theta_q| \leq \frac{\Delta l}{st}$ and $|\theta - \theta_q| > \frac{\Delta l}{st}$.

First, if $|\theta - \theta_q| \leq \frac{\Delta l}{st}$. In this case, from the definition of β it follows that

$$2\pi\beta|\theta - \theta_q| \leq \frac{s\pi t}{\Delta l}|\theta - \theta_q| \leq \pi$$

Combining the above equations implies that

$$\Pr\big[2\pi\beta(\theta-\theta_q) \pmod{2\pi} \in [-\frac{3s}{4}2\pi, \frac{3s}{4}2\pi]\big] = 0$$

Second, if $|\theta-\theta_q| > \frac{\Delta l}{st}$. We show this claim is true : $\Pr[2\pi\beta(\theta-\theta_q) \pmod{2\pi} \in [-\frac{3s}{4}2\pi, \frac{3s}{4}2\pi]] \lesssim s$. To prove it, we apply Lemma 12.6.3 by setting $\widetilde{T} = 2\pi$, $\widetilde{\sigma} = 2\pi\beta$, $\widetilde{\delta} = 0$, $\epsilon = \frac{3s}{4}2\pi$, $A = 2\pi\widehat{\beta}$, $\Delta f = |\theta-\theta_q|$. By upper bound of Lemma 12.6.3, the probability is at most

$$\frac{2\widetilde{\epsilon}}{\widetilde{T}} + \frac{4\widetilde{\epsilon}}{A\Delta f} = \frac{3s}{2} + \frac{3s}{\widehat{\beta}\Delta f} \leq \frac{3s}{2} + \frac{3s}{\frac{st}{4\Delta l}\frac{\Delta l}{st}} < 15s$$

Then in either case, with probability at least $1-15s$, we have

$$\|2\pi\beta\theta_q - 2\pi\beta\theta\|_{\bigcirc} > \frac{3s}{4}2\pi$$

which implies that v_q will not increase. □

Lemma 12.7.14. *Procedure* LOCATE1INNER *in Algorithm 12.4 uses* R_{loc} *"legal" samples, and then after Procedure* LOCATE1SIGNAL *in Algorithm 12.4 running Procedure* LOCATE1INNER $D_{\max}$ *times, it outputs a frequency* $\widetilde{f}_0$ *such that*

$$|\widetilde{f}_0 - f_0| \lesssim \Delta \cdot \sqrt{T\Delta}$$

with arbitrarily large constant probability.

Proof. For each observation, $v_{q'}$ incremented with probability at least $1-p$ and v_q is incremented with probability at most $15s+p$ for $|q-q'| > 3$. The probabilities corresponding to different observations are independent. Then after R_{loc} observations, there exists some

constant $c < \frac{1}{2}$, for any q such that $|q - q'| > 3$,

$$\begin{aligned} & \Pr[\text{False region gets more than half votes}] \\ = \; & \Pr[v_{j,q} > R_{\text{loc}}/2] \\ \leq \; & \binom{R_{\text{loc}}}{R_{\text{loc}}/2}(15s+p)^{R_{\text{loc}}/2} \\ \leq \; & c^{\Omega(R_{\text{loc}})} \end{aligned}$$

Similarly, on the other side,

$$\begin{aligned} & \Pr[\text{True region gets less than half votes}] \\ = \; & \Pr[v_{j,q'} < R_{\text{loc}}/2] \\ \leq \; & \binom{R_{\text{loc}}}{R_{\text{loc}}/2}(p)^{R_{\text{loc}}/2} \\ \leq \; & c^{\Omega(R_{\text{loc}})} \end{aligned}$$

Taking the union bound over all the t regions, it gives with probability at least $1 - tf^{\Omega(R_{\text{loc}})}$ we can find some region q such that $|q - q'| < 3$.

If we repeat the above procedure $D_{\max}$ rounds, each round we choose the "False" region with probability at most $1 - tc^{\Omega(R_{\text{loc}})}$. Thus, taking the union bound over all the $D_{\max}$ rounds, we will report a region has size $\approx \Delta\sqrt{\Delta T}$ and contains f_0 with probability at least $1 - D_{\max}tc^{\Omega(R_{\text{loc}})}$.

The reason for not ending up with region that has size $\approx \Delta$ is, the upper bound of the sample range of β force us to choose β is at most $\lesssim \frac{T}{(\Delta T)^{\frac{3}{2}}}$ by Claim 12.7.8

It remains to explain how to set $D_{\max}, t$, and R_{loc}. At the beginning of the first round, we start with frequency interval of length $2F$, at the beginning of the last round, we start

with frequency interval of length $t \cdot \Delta\sqrt{T\Delta}$. Each round we do a t-ary search, thus

$$D_{\max} = \log_t(\frac{2F}{t\Delta\sqrt{T\Delta}}) \leq \log_t(F/\Delta).$$

We can set $R_{\text{loc}} \approx \log_{1/c}(t/c)$ and $t > D_{\max}$, e.g. $t = \log(F/\Delta)$. Thus, the probability becomes,

$$1 - D_{\max} t c^{\Omega(R_{\text{loc}})} \geq 1 - t^2 c^{\Omega(R_{\text{loc}})} \geq 1 - \text{poly}(1/t, c)$$

which is larger than any constant probability. □

Using the same parameters setting in the proof of Lemma 12.7.14, we show the running time and sample complexity of Procedure LOCATE1SIGNAL,

Lemma 12.7.15. *Procedure* LOCATE1SIGNAL *in Algorithm 12.4 uses* $O(\text{poly}(k, \log(1/\delta))) \cdot \log(FT)$ *samples and runs in* $O(\text{poly}(k, \log(1/\delta))) \cdot \log^2(FT)$ *time.*

Proof. The number of "legal" observations is

$$D_{\max} R_{\text{loc}} = O(\log_t(F/\Delta) \log_{1/c}(t/c)) = O(\log(F/\Delta))$$

The total number of samples is

$$R_{\text{est}} + R_{\text{repeat}} D_{\max} R_{\text{loc}} = O(T\Delta_h)^2 + (T\Delta_h)^3 \cdot \log(FT) = \text{poly}(k, \log(1/\delta)) \cdot \log(FT)$$

where the first step follows by Claim 12.7.10 and Lemma 12.7.1 and the last step follows by the setting of Δ_h in Section 12.12.3.

The running time includes two parts, one is approximately computing $H(t)$ for all the samples, each sample takes $\text{poly}(k, \log(1/\delta))$ time according to Lemma 12.12.5; the other is for each legal sample we need to assign vote to some regions.

$$\text{poly}(k, \log(1/\delta)) \cdot (R_{\text{est}} + R_{\text{repeat}} D_{\max} R_{\text{loc}}) + D_{\max} R_{\text{loc}} t = \text{poly}(k, \log(1/\delta)) \log^2(FT)$$

□

Lemma 12.7.16 only achieves constant success probability, using median analysis we can boost the success probability,

Lemma 12.7.16. *Let $\widetilde{f}_0$ denote the frequency output by Procedure* FrequencyRecovery1Cluster *in Algorithm 12.5, then with probability at least $1 - 2^{-\Omega(k)}$,*

$$|\widetilde{f}_0 - f_0| \lesssim \Delta\sqrt{T\Delta}$$

Proof. Because of Procedure FrequencyRecovery1Cluster taking the median of $O(k)$ independent results by repeating algorithm Locate1Signal $O(k)$ times. Each sample L_r is close to $\widetilde{f}_0$ with sufficiently large probability. Thus, using the Chernoff bound will output $\widetilde{f}_0$ with probability $1 - 2^{-\Omega(k)}$ such that

$$|\widetilde{f}_0 - f_0| \lesssim \Delta\sqrt{T\Delta}.$$

□

Combining Lemma 12.7.16 with the sample complexity and running time in Lemma 12.7.14, we are able to finish the proof of Theorem 12.7.4.

12.7.5 The full signal, after multiplying by H and convolving with G, is one-clustered.

The goal of this section is to prove Lemma 12.7.5. We fix $f^* \in [-F, F]$ satisfying (12.17) in this section. We first define a good hashing (σ, b) of f^* as follows.

Definition 12.7.2. We say that a frequency f^* is *well-isolated* under the hashing (σ, b) if, for $j = h_{\sigma,b}(f^*)$, we have that the signal

$$\widehat{z}^{(j)} = \widehat{x \cdot H} \cdot \widehat{G}^{(j)}_{\sigma,b}$$

satisfies, over the interval $\overline{I_{f^*}} = (-\infty, \infty) \setminus (f^* - \Delta, f^* + \Delta)$,

$$\int_{\overline{I_{f^*}}} |\widehat{z}^{(j)}(f)|^2 df \lesssim \epsilon \cdot T\mathcal{N}^2/k.$$

For convenience, we simplify $z^{(j)}$ by using z in the rest of this section.

Lemma 12.7.17. *Let f^* be any frequency. Then f^* is well-isolated by a hashing (σ, b) with probability ≥ 0.9 given $B = \Theta(k)$ and $\sigma \in [\frac{1}{B\Delta}, \frac{2}{B\Delta}]$ chosen uniformly at random.*

Proof. For any other frequency f' in x^*, its contribution in $\widehat{z}$ depends on how far it is from f^*. Either it is:

- Within Δ of f^*, f' and f^* will be mapped into the same bucket with probability at least 0.99.
- Between Δ and $1/\sigma$ far, from Claim 12.6.2, f' and f^* will always mapped into different buckets. Hence f' always contributes in the $\frac{\epsilon\delta}{k}$ region of Property III in Lemma 12.6.5

about filter function $(G(t), \widehat{G}(f))$, i.e., it contributes at most $\frac{\epsilon\delta}{k} \cdot \int_{f'-\Delta}^{f'+\Delta} |\widehat{x \cdot H}|^2 \mathrm{d}f$. Overall it will contribute

$$\frac{\epsilon\delta}{k} \cdot \int |\widehat{x \cdot H}|^2 \mathrm{d}f = \frac{\epsilon\delta}{k} \int |x \cdot H|^2 \mathrm{d}t.$$

- More than $1/\sigma$ far, in which case they contribute in the same region with probability at most $3/B$. By a union bound, it is at most $3k/B \leq 0.01$

□

Without loss of generality, we assume $\mathrm{supp}(\widehat{g \cdot H}) \cap \mathrm{supp}(\widehat{x^* \cdot H}) = \emptyset$, otherwise we treat it as a part of $x^* \cdot H$. We first consider frequency $f^* \in \widehat{x^* \cdot H}$ under $G^{(j)}_{\sigma,b}$.

Lemma 12.7.18. *Let f^* satisfying $\int_{f^*-\Delta}^{f^*+\Delta} |\widehat{x^* \cdot H}(f)|^2 \mathrm{d}f \geq T\mathcal{N}^2/k$ and $\widehat{z} = \widehat{x^* \cdot H} \cdot \widehat{G}^{(j)}_{\sigma,b}$ where $j = h_{\sigma,b}(f^*)$. If f^* is well-isolated, then z and $\widehat{z}$ satisfying Property I(in Definition 12.7.1), i.e.,*

$$\int_0^T |z(t)|^2 \mathrm{d}t \geq (1-\epsilon) \int_{-\infty}^{+\infty} |z(t)|^2 \mathrm{d}t.$$

Proof. We first notice that $z(t) = x^*(t) \cdot H(t) * G^{(j)}_{\sigma,b}(t)$ and lower bound $\int_{-\infty}^{+\infty} |z(t)|^2 \mathrm{d}t$ as

follows :

$$\begin{aligned}
& \int_{-\infty}^{+\infty} |x^*(t) \cdot H(t) * G_{\sigma,b}^{(j)}(t)|^2 \mathrm{d}t \\
= & \int_{-\infty}^{+\infty} |\widehat{x^* \cdot H}(f) \cdot \widehat{G}_{\sigma,b}^{(j)}(f)|^2 \mathrm{d}f \qquad \text{by FT} \\
\geq & \int_{f_0-\Delta}^{f_0+\Delta} |\widehat{x^* \cdot H}(f) \cdot \widehat{G}_{\sigma,b}^{(j)}(f)|^2 \mathrm{d}f \\
\geq & (1-\delta)^2 \int_{f_0-\Delta}^{f_0+\Delta} |\widehat{x^* \cdot H}(f)\ |^2 \mathrm{d}f \\
\geq & (1-\delta)^2 T \mathcal{N}^2 / k \\
\geq & 0.9 \frac{\delta}{k} \int_0^T |x^*(t)|^2 \mathrm{d}t \qquad (12.19)
\end{aligned}$$

We give an upper bound $\int_{-\infty}^{0} |z(t)|^2 \mathrm{d}t + \int_T^{+\infty} |z(t)|^2 \mathrm{d}t \lesssim \epsilon \frac{\delta}{k} \int_0^T |x^*(t) H(t)|^2 \mathrm{d}t$ in the rest of this proof.

Consider the case $t < 0$, by definition of Convolution,

$$z^{(j)}(t) \quad = \quad x^*(t) \cdot H(t) * G_{\sigma,b}^{(j)}(t) = \int_{-\infty}^{+\infty} G_{\sigma,b}^{(j)}(t-\tau) \cdot (x^* \cdot H)(\tau) \mathrm{d}\tau$$

Without loss of generality, we can shift the original signal and $H(t)$ from $[0, T]$ to $[-T/2, T/2]$, by Property of $H(t)$, we know that if $s_3 T/2 \leq |t| \leq T/2$, then $H(t) \leq 2^{-O\Theta(\ell)}$. Note that $G(t)$ is compact and has support DB, we also assume its compact region is $[-DB/2, DB/2]$ (Recall that $D = \frac{l}{\alpha\pi}$).

Thus, by definition of convolution,

$$\begin{aligned}
& z(t) \\
= & \int_{-DB\sigma/2}^{DB\sigma/2} G_{\sigma,b}^{(j)}(s) \cdot (x \cdot H)(t-\tau) \mathrm{d}\tau \\
= & \frac{1}{\sigma} \int_{-DB\sigma/2}^{DB\sigma/2} G(s/\sigma) e^{2\pi \mathbf{i} s(j/B - \sigma b)/\sigma} \cdot (x \cdot H)(t-\tau) \mathrm{d}\tau \\
\leq & \frac{1}{\sigma} \int_{-DB\sigma/2}^{DB\sigma/2} |G(\tau/\sigma)| \cdot |(x \cdot H)(t-\tau)| \mathrm{d}\tau \\
\leq & \left(\frac{1}{\sigma} \int_{-DB\sigma/2}^{DB\sigma/2} |G(\tau/\sigma)| \mathrm{d}\tau \right) \cdot \left(\max_{|\tau| \leq DB\sigma/2} |(x \cdot H)(t-\tau)| \right)
\end{aligned}$$

So, if $t \notin [-T/2, T/2]$, then $t - s \notin [-T/2 + DB\sigma/2, T/2 - DB\sigma/2]$. By Property V of $G(t)$, $|G(t)| \leq \mathrm{poly}(k, \log(1/\delta))$. Because of the parameter setting[4], we have the fact $[-Ts_3/2, Ts_3/2] \subseteq [-T/2 + DB\sigma/2, T/2 - DB\sigma/2] \subseteq [-T/2, T/2]$. Thus, we know $T(1 - s_3)/2 > DB\sigma/2$, then for any $t - \tau \in [-T/2, -T/2 + DB\sigma/2] \cup [T/2 - DB\sigma/2, T/2] = S$, then

$$|z(t)|^2 \lesssim \big(DB\sigma \cdot \frac{1}{\sigma} \cdot \mathrm{poly}(k, \log(1/\delta))\big)^2 \cdot 2^{-\Theta(\ell)} \cdot k^4 \cdot \|x^*(t)\|_T^2 \lesssim \mathrm{poly}(k, \log(1/\delta)) \cdot 2^{-\Theta(\ell)} \cdot \|x^*(t)\|_T^2.$$

Thus, taking the integral over S,

$$\int_S |z(t)|^2 \mathrm{d}t \lesssim |S| \cdot 2^{-\Theta(\ell)} \, \mathrm{poly}(k, \log(1/\delta)) \cdot \|x^*(t)\|^2 \lesssim 2^{-\Theta(\ell)} T \|x^*(t) \cdot H(t)\|_T^2$$

By property of filter function $H(t), \widehat{H}(f)$, we have

$$|(x \cdot H)(t)|^2 \leq (\frac{t}{T})^{-\ell} \|x^*(t) \cdot H(t)\|_T^2 \text{ if } t \geq 3T$$

[4]We will set B to be $O(k)$, D to be $\mathrm{poly}(k)$ and σ to be $T/\mathrm{poly}(k)$.

Thus for any constant ϵ,

$$\int_{-\infty}^{-T/2} |z(t)|^2 \mathrm{d}t + \int_{T/2}^{+\infty} |z(t)|^2 \mathrm{d}t \lesssim 2^{-\ell} T \|x^*(t) \cdot H(t)\|_T^2 \leq 0.9\epsilon \cdot \frac{\delta}{k} \int_{-T/2}^{T/2} |x^*(t)|^2 \mathrm{d}t \quad (12.20)$$

where the last inequality follows by $\ell \gtrsim k \log(k/\delta)$. Shifting the interval from $[-T/2, T/2]$ to $[0, T]$, the same result is still holding. Combining Equation (12.19) and (12.20) completes the proof of Property II.

□

We consider frequency $f^* \in \widehat{g \cdot H}$ under $G_{\sigma,b}^{(j)}$ and show the energy of noise $g(t)$ is evenly distributed over B bins on expectation.

Lemma 12.7.19. *Given any noise $g(t) : [0, T] \to \mathbb{C}$ and $g(t) = 0, \forall t \notin [0, T]$. We have, $\forall j \in [B]$,*

$$\mathop{\mathbb{E}}_{\sigma,b} \left[\int_{-\infty}^{+\infty} |g(t)H(t) * G_{\sigma,b}^{(j)}(t)|^2 \mathrm{d}t \right] \lesssim \frac{1}{B} \int_{-\infty}^{+\infty} |g(t)H(t)|^2 \mathrm{d}t$$

Proof. Because of Fourier Transform preserves ℓ_2 norm, it suffices to prove

$$\mathop{\mathbb{E}}_{\sigma,b} \left[\int_{-\infty}^{+\infty} |\widehat{g \cdot H}(f) \cdot \widehat{G}_{\sigma,b}^{(j)}(f)|^2 \mathrm{d}f \right] \lesssim \frac{1}{B} \int_{-\infty}^{+\infty} |\widehat{g \cdot H}(f)|^2 \mathrm{d}f$$

Since $\widehat{G}_{\sigma,b}^{(j)}(f)$ is a periodic function and outputs at most 1 on $O(1/B)$ fraction of the period, and outputs $\leq \delta$ on other part. Thus, for any frequency f, we have

$$\mathop{\mathbb{E}}_{\sigma,b} \left[|\widehat{G}_{\sigma,b}^{(j)}(f)|^2 \right] \lesssim \frac{1}{B}$$

Thus, we have

$$\begin{aligned}
& \mathop{\mathbb{E}}_{\sigma,b}\left[\int_{-\infty}^{+\infty} |\widehat{g\cdot H}(f)\cdot \widehat{G}^{(j)}_{\sigma,b}(f)|^2 \mathrm{d}f\right] \\
\leq\ & \mathop{\mathbb{E}}_{\sigma,b}\left[\int_{-\infty}^{+\infty} |\widehat{g\cdot H}(f)|^2\cdot |\widehat{G}^{(j)}_{\sigma,b}(f)|^2 \mathrm{d}f\right] \\
=\ & \int_{-\infty}^{+\infty} |\widehat{g\cdot H}(f)|^2\cdot \mathop{\mathbb{E}}_{\sigma,b}[|\widehat{G}^{(j)}_{\sigma,b}(f)|^2]\mathrm{d}f \\
\leq\ & \int_{-\infty}^{+\infty} |\widehat{g\cdot H}(f)|^2 \mathrm{d}f \cdot \max_f \left[\mathop{\mathbb{E}}_{\sigma,b}\left|\widehat{G}^{(j)}_{\sigma,b}(f)\right|^2\right] \\
\lesssim\ & \frac{1}{B}\int_{-\infty}^{+\infty} |\widehat{g\cdot H}(f)|^2 \mathrm{d}f,
\end{aligned}$$

which completes the proof. □

Proof of Lemma 12.7.5. Let $j = h_{\sigma,b}(f^*)$, signal

$$\widehat{z} = \widehat{x\cdot H}\cdot \widehat{G}^{(j)}_{\sigma,b}, \tag{12.21}$$

and region $I_{f^*} = (f^* - \Delta, f^* + \Delta)$ with complement $\overline{I_{f^*}} = (-\infty,\infty)\setminus I_{f^*}$. From Property I of G in Lemma 12.6.5, we have that

$$\widehat{G}^{(l)}_{\sigma,b}(f) \gtrsim 1$$

for all $f \in I_{f^*}$, so by (12.17)

$$\int_{I_{f^*}} |\widehat{z}(f)|^2 \mathrm{d}f \geq T\mathcal{N}^2/k.$$

On the other hand, f^* is will-isolated with probability 0.9:

$$\int_{\overline{I_{f^*}}} |\widehat{z}(f)|^2 \mathrm{d}f \lesssim \epsilon T\mathcal{N}^2/k.$$

Hence, $\widehat{z}$ satisfies the Property I(in Definition 12.7.1) of one-mountain recovery. Combining Lemma 12.7.18 and Lemma 12.7.19, we know that $(x^* \cdot H) * G_{\sigma,b}^{(j)}$ always satisfies Property II(in Definition 12.7.1) and $\int_{-\infty}^{+\infty} |g(t)H(t) * G_{\sigma,b}^{(j)}(t)|^2 \mathrm{d}t$ is less than $20T\mathcal{N}^2/B \leq \epsilon T\mathcal{N}^2/k$ with probability at least 0.95, which indicates that $z = (x^* + g) \cdot H * G_{\sigma,b}^{(j)}$ satisfies Property II(in Definition 12.7.1) with probability 0.95. □

12.7.6 Frequency recovery of k-clustered signals

The goal of this section is to prove that the frequencies found by Procedure FrequencyRecoveryKCluster in Algorithm 12.8 have some reasonable guarantee.

We first notice that Lemma 12.7.5 and Lemma 12.7.2 imply the following lemma by a union bound.

Lemma 12.7.20. *Let $x^*(t) = \sum_{j=1}^{k} v_j e^{2\pi \mathbf{i} f_j t}$. We observe $x(t) = x^*(t) + g(t)$, where $\|g(t)\|_T^2 \le c\|x^*(t)\|_T^2$ for a sufficiently small constant c and define $\mathcal{N}^2 := \|g(t)\|_T^2 + \delta\|x^*(t)\|_T^2$. Then Procedure OneStage returns a set L of $O(k)$ frequencies that covers the heavy frequencies of x^*. In particular, for any f^* with*

$$\int_{f^*-\Delta}^{f^*+\Delta} |\widehat{x \cdot H}(f)|^2 \mathrm{d}f \ge T\mathcal{N}^2/k, \tag{12.22}$$

there will exist an $\widetilde{f} \in L$ satisfying $|f^* - \widetilde{f}| \lesssim \sqrt{T\Delta} \cdot \Delta T$ with probability 0.99.

Lemma 12.7.21. *Let $x^*(t) = \sum_{j=1}^{k} v_j e^{2\pi \mathbf{i} f_j t}$ and $R = O(k)$. We observe $x(t) = x^*(t) + g(t)$, where $\|g(t)\|_T^2 \le c\|x^*(t)\|_T^2$ for a sufficiently small constant c and choose $\mathcal{N}^2 := \|g(t)\|_T^2 + \delta\|x^*(t)\|_T^2$. Then Algorithm MultipleStages returns a set L of $O(k)$ frequencies that approximates the heavy frequencies of x^*. In particular, with probability $1 - 2^{-\Omega(k)}$, for any f^* such that*

$$\int_{f^*-\Delta}^{f^*+\Delta} |\widehat{x \cdot H}(f)|^2 \mathrm{d}f \ge T\mathcal{N}^2/k, \tag{12.23}$$

there will exist an $\widetilde{f} \in L$ satisfying $|f^* - \widetilde{f}| \lesssim \sqrt{T\Delta}\Delta$.

Proof. Let $A \subset [-F, F]$ denote the set of frequencies f^* satisfying Equation (12.22). Let $A' \subset [-F, F]$ denote a net of A of distance 2Δ, so the intervals used in Equation (12.22) for

each $f^* \in A'$ are disjoint. Then

$$|A'| \le 2k + k = 3k$$

because each frequency in x^* contributes to at most two of the intervals, and the total mass of $\widehat{g}$ is at most k times the threshold $T\mathcal{N}^2$.

Let $L_1, \ldots, L_R$ be the results of R rounds of Algorithm OneStage. We say that a frequency $f \in A'$ is *successfully recovered in round* r if there exists an $\widetilde{f} \in L_r$ such that $|f - \widetilde{f}| \le \Delta_a$, where

$$\Delta_a = \Delta\sqrt{T\Delta} \lesssim \sqrt{T\Delta}\Delta.$$

By Lemma 12.7.20, each frequency is successfully recovered with 0.8 probability in each round. Then by the Chernoff bound, with $1 - 2^{-\Omega(k)}$ probability, every $f \in A'$ will be successfully recovered in at least $0.6R$ rounds.

Then, by Lemma 12.7.22, we output a set L of $O(B)$ frequencies such that every $f \in A'$ is within Δ_a of some $\widetilde{f} \in L$. Hence every $f \in A$ is within $2\Delta_a$ of some $\widetilde{f} \in L$. □

Lemma 12.7.22. *Let $L_1, \ldots, L_R$ by sets of frequencies and f^* be any frequency. Then $L = \text{MergedStages}(L_1, \ldots, L_R)$ is a set of $2\frac{\sum |L_r|}{R}$ frequencies satisfying*

$$\min_{\widetilde{f} \in L} |f^* - \widetilde{f}| \le \operatorname*{median}_{r \in [R]} \min_{f \in L_r} |f^* - f|.$$

Proof. The algorithm is to take the union, sort, and take every $\frac{R}{2}$th entry of the sorted list.

Let $\Delta = \text{median}_{r\in[R]} \min_{f\in L_r} |f^* - f|$. We have that at least $R/2$ different $f \in \bigcup_r L_r$ lie within Δ of f^*. This set forms a sequential subsequence of the sorted list of frequencies, so our output will include one. □

12.7.7 Time and sample complexity of frequency recovery of k-clustered signals

The goal of this section is to show that Procedure FrequencyRecoveryKCluster takes

$\mathrm{poly}(k, \log(1/\delta)) \log(FT)$ samples, and runs in $\mathrm{poly}(k, \log(1/\delta)) \log^2(FT)$ time.

In order to analyze the running time and sample complexity. We need to extend the one-cluster version Procedure GetLegal1Sample and GetEmpirical1Energy (in Algorithm 12.3) to k-cluster version GetLegalKSample and GetEmpiricalKEnergy(in Algorithm 12.7)[5],

Lemma 12.7.23. *Procedure* GetLegalKSample *in Algorithm 12.7 runs Procedure* HashToBins $R_{\mathrm{repeat}} = O((T\Delta)^3)$ *times to output two vectors* $\widehat{v}, \widehat{v}' \in \mathbb{C}^B$ *such that, for each* $j \in [B]$,

$$|\widehat{v}_j - \widehat{v}'_j e^{2\pi \mathbf{i} f_j \beta}| \leq 0.08(|\widehat{v}_j| + |\widehat{v}'_j|),$$

holds with probability at least 0.6.

Using the definition of z in Definition 12.7.2.

Claim 12.7.24. *Procedure* GetEmpiricalKEnergy *in Algorithm 12.7 runs Procedure* HashTobins $R_{\mathrm{est}} O((T\Delta)^2)$ *times to output a vector* $z_{\mathrm{emp}} \in \mathbb{R}^B$ *such that, for each* $j \in [B]$,

$$z^j_{\mathrm{emp}} \in [0.8\|z^{(j)}\|_T, 1.2\|z^{(j)}\|_T],$$

holds with probability at least 0.9.

[5] We omitted the proofs here, because the proofs are identical to the one-cluster situation.

Claim 12.7.25. *Algorithm* LocateKSignal *in Algorithm 12.6 uses* $O(\text{poly}(k, \log(1/\delta)) \cdot \log(FT))$, *and runs in* $O(\text{poly}(k, \log(1/\delta)) \cdot \log^2(FT))$.

Proof. We first calculate the number of samples. All the samples is basically all the Fourier samples, each time needs $B \log(k/\delta)$. In total it calls HashToBins $O(R_{\text{est}} + R_{\text{repeat}} D_{\max} R_{\text{loc}})$ times where $D_{\max} R_{\text{loc}} = \Theta(\log(FT))$ by similar analysis as one-cluster frequency recovery. Thus, the total number of samples is

$$(R_{\text{est}} + R_{\text{repeat}} D_{\max} R_{\text{loc}}) B \log(k/\delta) = \text{poly}(k, \log(1/\delta)) \cdot \log(FT).$$

Then, we analyze the running time.

The expected running time includes the following parts: the first part is running Procedure HashToBins $O(R_{\text{est}} + R_{\text{repeat}} D_{\max} R_{loc})$ times, each run takes $O(B \log(k/\delta) + B \log B)$ samples. For each such sample we need $\text{poly}(k, \log(1/\delta))$ time to compute $H(t)$ according to Lemma 12.12.5 and there are $\text{poly}(k, \log(1/\delta)) \log(FT))$ many samples; the second part is updating the counter v,which takes $O(D_{\max} R_{loc} B t)$ time. Thus, in total

$$\begin{aligned} &\text{poly}(k, \log(1/\delta)) \cdot O(R_{\text{est}} + R_{\text{repeat}} D_{\max} R_{\text{loc}}) \cdot O(B \log(k/\delta) + B \log B) + O(D_{\max} R_{loc} B t) \\ =& \text{poly}(k, \log(1/\delta)) \cdot \log^2(FT), \end{aligned}$$

where by similar analysis as one-cluster recovery, $t = \Theta(\log(FT))$ and $D_{\max} R_{\text{loc}} = \Theta(\log(FT))$.

□

To boost the success probability, Procedure MultipleStages reruns Procedure LocateKSignal $O(k)$ times. At the end, Procedure FrequencyRecoveryKCluster combining Procedure MultipleStages and MergedStages directly, and the running time

and sample complexity of MultipleStages are dominating MergedStages. Thus we have

Lemma 12.7.26. *Procedure* FrequencyRecoveryKCluster *in Algorithm 12.8 uses*

$$\text{poly}(k, \log(1/\delta)) \cdot \log(FT)$$

samples and runs in $\text{poly}(k, \log(1/\delta)) \cdot \log^2(FT)$ *time.*

12.8 One-cluster Signal Recovery

12.8.1 Overview

In this section, we consider x^* whose frequencies in $\widehat{x}^*$ are in the range $[f_0 - \Delta', f_0 + \Delta']$ for some frequency f_0 and $\Delta' > 0$ and provide an algorithm to approximate it by a polynomial.

We fix T in this section and recall that $\langle f(t), g(t)\rangle_T := \frac{1}{T}\int_0^T f(t)\overline{g}(t)\mathrm{d}t$ such that $\|e^{2\pi \mathbf{i} f_i t}\|_T = \sqrt{\langle e^{2\pi \mathbf{i} f_i t}, e^{2\pi \mathbf{i} f_i t}\rangle_T} = 1$. For convenience, given $\sum_{j=1}^{k} v_j e^{2\pi \mathbf{i} f_j t}$, we say the frequency gap of this signal is $\min_{i\neq j}|f_i - f_j|$.

For simplicity, we first consider frequencies clustered around 0. The main technical lemma in this section is that any signal x^* with bounded frequencies in $\widehat{x}^*$ can be approximated by a low-degree polynomial on $[0, T]$.

Lemma 12.2.3. For any $\Delta > 0$ and any $\delta > 0$, let $x^*(t) = \sum_{j\in[k]} v_j e^{2\pi \mathbf{i} f_j t}$ where $|f_j| \leq \Delta$ for each $j \in [k]$. There exists a polynomial $P(t)$ of degree at most

$$d = O(T\Delta + k^3 \log k + k \log 1/\delta)$$

such that

$$\|P(t) - x^*(t)\|_T^2 \leq \delta \|x^*\|_T^2.$$

One direct corollary is that when $\widehat{x}^*$ are in the range $[f_0 + \Delta', f_0 + \Delta']$, we can approximate x^* by $P(t) \cdot e^{2\pi \mathbf{i} f_0 t}$ for some low degree polynomial P.

We give an overview of this section first. We first show some technical tools in Section 12.8.2, 12.8.3. In Section 12.8.4, using those tools, we can show for any k-Fourier-sparse signal, there exists another k-Fourier-sparse signal with bounded frequency gap close to the

original signal. In Section 12.8.5, we show that for any k-Fourier-sparse signal with bounded frequency gap, then there exists a low degree polynomial close to it. In Section 12.8.6, we show how to transfer low degree polynomial back to a Fourier-sparse signal. Combining all the above steps finishes the proof of Lemma 12.2.3.

We apply Theorem 12.7.4 of frequency estimation on x^* to obtain an estimation $\widetilde{f}_0$ of f_0 and use Theorem 12.4.5 on the approximation $Q(t)e^{2\pi \mathbf{i} \widetilde{f}_0 t}$ of x^* to recover the signal. We summarize this result as follows.

Theorem 12.8.1 (One-cluster Signal Recovery)**.** *Let* $x^*(t) = \sum_{j=1}^{k} v_j e^{2\pi \mathbf{i} f_j t}$ *where* $\forall j \in [k], |f_j - f_0| \leq \Delta$ *and* $x(t) = x^*(t) + g(t)$ *be our observable signal. For any* $\delta > 0$ *and any* $T > 0$*, let* $\mathcal{N}^2 := \|g\|_T^2 + \delta \|x^*\|_T^2$*. Procedure* CFT1CULSTER *in Algorithm 12.5 finds a polynomial* $P(t)$ *of degree at most* $d = O\left((T\Delta_h + T\Delta)^{1.5} + k^3 \log k + k \log 1/\delta\right)$ *and a frequency* $\widetilde{f}_0$ *such that*

$$\|P(t) \cdot e^{2\pi \mathbf{i} \widetilde{f}_0 t} - x^*(t)\|_T^2 \lesssim \mathcal{N}^2 \tag{12.24}$$

The algorithm uses $O(kd)+\mathrm{poly}(k, \log(1/\delta)) \log(FT)$ *samples, run in* $O(kd^\omega)+\mathrm{poly}(k, \log(1/\delta)) \log^2(FT)$ *time, and succeeds with probability at least* $1 - 2^{-\Omega(k)}$*.*

Proof. We apply the algorithm in Theorem 12.7.4 to obtain an estimation $\widetilde{f}_0$ with $\mathrm{poly}(k) \log(FT)$ samples and $\mathrm{poly}(k) \log^2(FT)$ running time such that $|\widetilde{f}_0 - f_0| \lesssim (\Delta_h + \Delta)\sqrt{T(\Delta_h + \Delta)}$ holds with probability at least $1 - 2^{-\Omega(k)}$. Notice that $|f_j - \widetilde{f}_0| \leq |f_j - f_0| + |\widetilde{f}_0 - f_0| \lesssim (T(\Delta_h + \Delta))^{1.5}$.

We consider $x'(t) = e^{-2\pi \mathbf{i} \widetilde{f}_0 t} x(t) = \sum_{j=1}^{k} v_j e^{2\pi \mathbf{i} (f_j - \widetilde{f}_0) t}$. By Lemma 12.2.3, there exists a polynomial $P(t)$ of degree at most

$$d = O\left((T\Delta_h + T\Delta)^{1.5} + k^3 \log k + k \log 1/\delta\right)$$

such that it approximates x' by

$$\|P(t) - x'(t)\|_T \leq \frac{\delta}{4}\|x'(t)\|_T = \frac{\delta}{4}\|x^*(t)\|_T.$$

which indicates $\|Q(t) - e^{-2\pi \mathbf{i} \widetilde{f}_0 t} \cdot x^*(t)\|_T \leq \frac{\delta}{4}\|x^*(t)\|_T$.

Because we can sample $x(t)$, we can also sample $e^{-2\pi \mathbf{i} \widetilde{f}_0 t} \cdot x(t) = Q(t) + g'(t)$ for $g'(t) = e^{-2\pi \mathbf{i} \widetilde{f}_0 t} \cdot g(t) + (e^{-2\pi \mathbf{i} \widetilde{f}_0 t} \cdot x^*(t) - Q(t))$. Hence we apply the algorithm in Theorem 12.4.5 and choose $R = O(k)$ in that proof. Then Procedure RobustPolynomialLearning$^+$ takes $O(kd)$ samples and $O(kd^\omega)$ time to find a degree d polynomial $P(t)$ approximating $Q(t)$ such that

$$\|P(t) - Q(t)\|_T \lesssim \|g'(t)\|_T,$$

holds with probability at least $1 - 2^{-\Omega(k)}$. It indicates

$$\|P(t) - e^{-2\pi \mathbf{i} \widetilde{f}_0 t} \cdot x^*(t)\|_T \lesssim \|P(t) - Q(t)\|_T + \|Q(t) - x^*(t)\| \lesssim \delta \|x^*(t)\|_T + \|g(t)\|_T \eqsim \mathcal{N}.$$

Therefore we know $\|e^{2\pi \mathbf{i} \widetilde{f}_0 t} \cdot P(t) - x^*(t)\|_T^2 \lesssim \mathcal{N}^2$. □

12.8.2 Bounding the Gram matrix determinant

We define Gram matrix for $e^{2\pi \mathbf{i} f_1 t}, e^{2\pi \mathbf{i} f_2 t}, \cdots, e^{2\pi \mathbf{i} f_k t}$ and provide lower/upper bounds for its determinant.

Definition 12.8.1 (Gram matrix)**.** We define $\mathrm{Gram}_{f_1,\cdots,f_k}$ to be

$$\begin{bmatrix} \langle e^{2\pi \mathbf{i} f_1 t}, e^{2\pi \mathbf{i} f_1 t}\rangle_T & \langle e^{2\pi \mathbf{i} f_1 t}, e^{2\pi \mathbf{i} f_2 t}\rangle_T & \cdots & \langle e^{2\pi \mathbf{i} f_1 t}, e^{2\pi \mathbf{i} f_k t}\rangle_T \\ \langle e^{2\pi \mathbf{i} f_2 t}, e^{2\pi \mathbf{i} f_1 t}\rangle_T & \langle e^{2\pi \mathbf{i} f_2 t}, e^{2\pi \mathbf{i} f_2 t}\rangle_T & \cdots & \langle e^{2\pi \mathbf{i} f_2 t}, e^{2\pi \mathbf{i} f_k t}\rangle_T \\ \cdots & \cdots & \cdots & \cdots \\ \langle e^{2\pi \mathbf{i} f_k t}, e^{2\pi \mathbf{i} f_1 t}\rangle_T & \langle e^{2\pi \mathbf{i} f_k t}, e^{2\pi \mathbf{i} f_2 t}\rangle_T & \cdots & \langle e^{2\pi \mathbf{i} f_k t}, e^{2\pi \mathbf{i} f_k t}\rangle_T \end{bmatrix}$$

Note that the above matrix is a Hermitian matrix with complex entries, thus both its determinant and all eigenvalues are in $\mathbb{R}$.

We defer the proof of the following Theorem to Section 12.10.1.

Theorem 12.8.2. For real numbers $\xi_1, \ldots, \xi_k$, let $G_{\xi_1,\ldots,\xi_k}$ be the matrix whose (i,j)-entry is

$$\int_{-1}^{1} e^{2\pi \mathbf{i}(\xi_i - \xi_j)t} dt.$$

Then

$$\det(G_{\xi_1,\ldots,\xi_k}) = 2^{\widetilde{O}(k^2)} \prod_{i<j} \min(|\xi_i - \xi_j|^2, 1).$$

We use the following corollary in this section.

Corollary 12.8.3. *There exists a universal constant $\alpha > 0$ such that, for any $T > 0$ and real numbers $f_1, \cdots, f_k$, the $k \times k$ Gram matrix of $e^{2\pi \mathbf{i} f_1 t}, e^{2\pi \mathbf{i} f_2 t}, \cdots, e^{2\pi \mathbf{i} f_k t}$ whose (i,j)-entry is*

$$\mathrm{Gram}_{f_1,\cdots,f_k}(i,j) = \langle e^{2\pi \mathbf{i} f_i t}, e^{2\pi \mathbf{i} f_j t}\rangle_T = \frac{1}{T}\int_0^T e^{2\pi \mathbf{i}(f_i - f_j)t} \mathrm{d}t.$$

satisfies

$$k^{-\alpha k^2} \prod_{i<j} \min((|f_i - f_j|T)^2, 1) \le \det\left(\mathrm{Gram}_{f_1,\cdots,f_k}\right) \le k^{\alpha k^2} \prod_{i<j} \min((|f_i - f_j|T)^2, 1).$$

Based on Corollary 12.8.3, we show the coefficients of a k-Fourier-sparse signal can be upper bounded by the energy $\|x\|_T^2$.

Lemma 12.2.2. There exists a universal constant $c > 0$ such that for any $x(t) = \sum_{j=1}^{k} v_j e^{2\pi \mathbf{i} f_j t}$ with frequency gap $\eta = \min_{i \neq j} |f_i - f_j|$,

$$\|x(t)\|_T^2 \geq k^{-ck^2} \min\left((\eta T)^{2k}, 1\right) \sum_{j=1}^{k} |v_j|^2.$$

Proof. Let $\vec{v_i}$ denote the vector $e^{2\pi \mathbf{i} f_i t}$ and $V = \{\vec{v_1}, \cdots, \vec{v_k}\}$. Notice that $\|\vec{v_i}\|_T^2 = \langle \vec{v_i}, \vec{v_i} \rangle = 1$. For each $\vec{v_i}$, we define $\vec{v_i}^{\parallel}$ to be the projection of $\vec{v_i}$ into the linear subspace $\text{span}\{V \setminus \vec{v_i}\} = \text{span}\{\vec{v_1}, \cdots, \vec{v_{i-1}}, \vec{v_{i+1}}, \cdots, \vec{v_k}\}$ and $\vec{v_i}^{\perp} = \vec{v_i} - \vec{v_i}^{\parallel}$ which is orthogonal to $\text{span}\{V \setminus \vec{v_i}\}$ by the definition.

Therefore from the orthogonality,

$$\|x(t)\|_T^2 \geq \max_{j \in [k]} \{|v_j|^2 \cdot \|\vec{v_j}^{\perp}\|_T^2\} \geq \frac{1}{k} \sum_{j=1}^{k} |v_j|^2 \cdot \|\vec{v_j}^{\perp}\|_T^2.$$

It is enough to estimate $\|\vec{v_j}^{\perp}\|_T^2$ from Claim 12.3.11:

$$\|\vec{v_j}^{\perp}\|_T^2 = \frac{\det(\text{Gram}(V))}{\det(\text{Gram}(V \setminus \vec{v_i}))} \geq k^{-2\alpha k^2} \prod_{j \neq i} \min\left((f_j - f_i)T, 1\right)^2 \geq k^{-2\alpha k^2} (\eta T)^{2k-2},$$

where we use Corollary 12.8.3 to lower bound it in the last step. □

12.8.3 Perturbing the frequencies does not change the subspace much

We show that for a k-Fourier-sparse signal with unboundedly close frequency gap, there always exists another k-Fourier-sparse signal with slightly separated gap.

Lemma 12.8.4 (Slightly Shifting one Frequency)**.** *There is a universal constant $C_0 > 0$ such that for any $x(t) = \sum_{j=1}^{k} v_j e^{2\pi \mathbf{i} f_j t}$ and any frequency f_{k+1}, there always exists*

$$x'(t) = \sum_{j=1}^{k-1} v'_j e^{2\pi \mathbf{i} f_j t} + v'_{k+1} e^{2\pi \mathbf{i} f_{k+1} t}$$

with k coefficients $v'_1, v'_2, \cdots, v'_{k-1}, v'_{k+1}$ satisfying

$$\|x'(t) - x(t)\|_T \leq k^{C_0 k^2} \cdot (|f_k - f_{k+1}|T) \cdot \|x(t)\|_T$$

Proof. We abuse the notation $e^{2\pi \mathbf{i} f_j t}$ to denote a vector in the linear subspace. We plan to shift f_k to f_{k+1} and define

$$\begin{aligned} V &= \{e^{2\pi \mathbf{i} f_1 t}, \cdots, e^{2\pi \mathbf{i} f_{k-1} t}, e^{2\pi \mathbf{i} f_k t}\} \\ V' &= \{e^{2\pi \mathbf{i} f_1 t}, \cdots, e^{2\pi \mathbf{i} f_{k-1} t}, e^{2\pi \mathbf{i} f_{k+1} t}\} \\ U &= \{e^{2\pi \mathbf{i} f_1 t}, \cdots, e^{2\pi \mathbf{i} f_{k-1} t}\} \\ W &= \{e^{2\pi \mathbf{i} f_1 t}, \cdots, e^{2\pi \mathbf{i} f_{k-1} t}, e^{2\pi \mathbf{i} f_k t}, e^{2\pi \mathbf{i} f_{k+1} t}\} \end{aligned}$$

where $f_1, f_2, \cdots, f_k$ are original frequencies in x. The idea is to show that any vector in the linear subspace $\mathrm{span}\{V\}$ is close to some vector in the linear subspace $\mathrm{span}\{V'\}$.

For convenience, we use $\vec{u}^{\parallel}$ to denote the projection of vector $e^{2\pi \mathbf{i} f_k t}$ to the linear subspace $\mathrm{span}\{U\} = \mathrm{span}\{e^{2\pi \mathbf{i} f_1 t}, \cdots, e^{2\pi \mathbf{i} f_{k-1} t}\}$ and $\vec{w}^{\parallel}$ denote the projection of vector $e^{2\pi \mathbf{i} f_{k+1} t}$

to this linear subspace span$\{U\}$. Let $\vec{u}^{\perp} = e^{2\pi \mathbf{i} f_k t} - \vec{u}^{\parallel}$ and $\vec{w}^{\perp} = e^{2\pi \mathbf{i} f_{k+1} t} - \vec{w}^{\parallel}$ be their orthogonal part to span$\{U\}$.

From the definition $e^{2\pi \mathbf{i} f_k t} = \vec{u}^{\parallel} + \vec{u}^{\perp}$ and $\vec{u}^{\parallel} \in \text{span}\{U\} = \text{span}\{e^{2\pi \mathbf{i} f_1 t}, \cdots, e^{2\pi \mathbf{i} f_{k-1} t}\}$, we rewrite the linear combination

$$x(t) = \sum_{j=1}^{k} v_j e^{2\pi \mathbf{i} f_j t} = \sum_{j=1}^{k-1} \alpha_j e^{2\pi \mathbf{i} f_j t} + v_k \cdot \vec{u}^{\perp}$$

for some scalars $\alpha_1, \cdots, \alpha_{k-1}$.

We will substitute $\vec{u}^{\perp}$ by $\vec{w}^{\perp}$ in the above linear combination and find a set of new coefficients. Let $\vec{w}^{\perp} = \vec{w}_1 + \vec{w}_2$ where $\vec{w}_1 = \frac{\langle \vec{u}^{\perp}, \vec{w}^{\perp} \rangle}{\|\vec{u}^{\perp}\|_T^2} \vec{u}^{\perp}$ is the projection of $\vec{w}^{\perp}$ to $\vec{u}^{\perp}$. Therefore $\vec{w}_2$ is the orthogonal part of the vector $e^{2\pi \mathbf{i} f_{k+1} t}$ to span$\{V\} = \text{span}\{e^{2\pi \mathbf{i} f_1 t}, \cdots, e^{2\pi \mathbf{i} f_{k-1} t}, e^{2\pi \mathbf{i} f_k t}\}$. We use $\delta = \frac{\|\vec{w}_2\|_T}{\|\vec{w}^{\perp}\|_T}$ for convenience.

Notice that the $\min_{\beta \in \mathbb{C}} \frac{\|\vec{u}^{\perp} - \beta \cdot \vec{w}^{\perp}\|_T}{\|\vec{u}^{\perp}\|_T} = \delta$ and $\beta^* = \frac{\langle \vec{u}^{\perp}, \vec{w}^{\perp} \rangle}{\|\vec{w}^{\perp}\|_T^2}$ is the optimal choice. Therefore we set

$$x'(t) = \sum_{j=1}^{k-1} \beta_j e^{2\pi \mathbf{i} f_j t} + v_k \cdot \beta^* \cdot \vec{w}^{\perp} \in \text{span}\{e^{2\pi \mathbf{i} f_1 t}, \cdots, e^{2\pi \mathbf{i} f_{k-1} t}, e^{2\pi \mathbf{i} f_{k+1} t}\}$$

where the coefficients $\beta_1, \cdots, \beta_{k-1}$ guarantee that the projection of x' onto span$\{U\}$ is as same as the projection of x onto span$\{U\}$. From the choice of β^* and the definition of x',

$$\|x(t) - x'(t)\|_T^2 = \delta^2 \cdot |v_k|^2 \cdot \|\vec{u}^{\perp}\|_T^2 \leq \delta^2 \cdot \|x(t)\|_T^2.$$

Eventually, we show an upper bound for δ^2 from Claim 12.3.11.

$$\begin{aligned}
\delta^2 &= \frac{\|\vec{w}_2\|_T^2}{\|\vec{w}^{\perp}\|_T^2} \\
&= \frac{\det(\mathrm{Gram}_W)}{\det(\mathrm{Gram}_V)} \Big/ \frac{\det(\mathrm{Gram}_{V'})}{\det(\mathrm{Gram}_U)} \text{ by Claim 12.3.11} \\
&= \frac{\det(\mathrm{Gram}_W)}{\det(\mathrm{Gram}_V)} \cdot \frac{\det(\mathrm{Gram}_U)}{\det(\mathrm{Gram}_{V'})} \text{ by Corollary 12.8.3} \\
&\leq k^{4\alpha k^2} \cdot \frac{\prod_{i=1}^{k+1} \prod_{\substack{j=1 \\ j\neq i}}^{k+1} \min(|f_i - f_j|T, 1)}{\prod_{i=1}^{k} \prod_{\substack{j=1 \\ j\neq i}}^{k} \min(|f_i - f_j|T, 1)} \cdot \frac{\prod_{i=1}^{k-1} \prod_{\substack{j=1 \\ j\neq i}}^{k-1} \min(|f_i - f_j|T, 1)}{\prod_{i=1}^{k-1} \prod_{\substack{j=1 \\ j\neq i}}^{k-1} \min(|f_i - f_j|T, 1) \cdot \prod_{i=1}^{k-1} \min(|f_i - f_{k+1}|^2 T^2, 1)} \\
&= k^{4\alpha k^2} |f_k - f_{k+1}|^2 T^2
\end{aligned}$$

□

Lemma 12.8.5. *For any k frequencies $f_1 < f_2 < \cdots < f_k$, there exists k frequencies $f'_1, \cdots, f'_k$ such that $\min_{i\in[k-1]} f'_{i+1} - f'_i \geq \eta$ and for all $i \in [k]$, $|f'_i - f_i| \leq k\eta$.*

Proof. We define the new frequencies f'_i as follows: $f'_1 = f_1$ and $f'_i = \max\{f'_{i-1} + \eta, f_i\}$ for $i \in \{2, 3, \cdots, k\}$. □

12.8.4 Existence of nearby k-Fourier-sparse signal with frequency gap bounded away from zero

We combine the results in the above section to finish the proof of Lemma 12.2.3. We first prove that for any $x^*(t) = \sum_{j=1}^{k} v_j e^{2\pi \mathbf{i} f_j t}$, there always exists another k-Fourier-sparse signal x' close to $x^*(t) = \sum_{j=1}^{k} v_j e^{2\pi \mathbf{i} f_j t}$ such that the frequency gap in x' is at least $\eta \geq 2^{-\operatorname{poly}(k)}$. Then we show how to find a low degree polynomial $P(t)$ approximating $x'(t)$.

Lemma 12.2.1. There is a universal constant $C_1 > 0$ such that, for any $x^*(t) = \sum_{j=1}^{k} v_j e^{2\pi \mathbf{i} f_j t}$ and any $\delta > 0$, there always exist $\eta \geq \frac{\delta}{T} \cdot k^{-C_1 k^2}$ and $x'(t) = \sum_{j=1}^{k} v'_j e^{2\pi \mathbf{i} f'_j t}$ satisfying

$$\|x'(t) - x^*(t)\|_T \leq \delta \|x^*(t)\|_T$$

with $\min_{i \neq j} |f'_i - f'_j| \geq \eta$ and $\max_{j \in [k]} \{|f'_j - f_j|\} \leq k\eta$.

Proof. Using Lemma 12.8.5 on frequencies $f_1, \cdots, f_k$, we obtain k new frequencies $f'_1, \cdots, f'_k$ such that their gap is at least η and $\max_i |f_i - f'_i| \leq k\eta$. Next we use the hybrid argument to find x'.

Let $x^{(0)}(t) = x^*(t)$. For $i = 1, \cdots, t$, we apply Lemma 12.8.4 to shift f_i to f'_i and obtain

$$x^{(i)}(t) = \sum_{j=i+1}^{k} v_j^{(i)} e^{2\pi \mathbf{i} f_j t} + \sum_{j=1}^{i} v_j^{(i)} e^{2\pi \mathbf{i} f'_j t}.$$

From Lemma 12.8.4, we know $\|x^{(i)}(t) - x^{(i-1)}(t)\|_T \leq k^{C_0 k^2} (|f_i - f'_i| T) \|x^{(i-1)}\|_T$. Thus we obtain

$$\left(1 - k^{C_0 k^2} (k\eta T)\right)^i \|x^{(0)}(t)\|_T \leq \|x^{(i)}(t)\|_T \leq \left(1 + k^{C_0 k^2} (k\eta T)\right)^i \|x^{(0)}(t)\|_T,$$

which is between $\left[\left(1 - i \cdot k^{C_0 k^2}(k\eta T)\right)\|x^{(0)}(t)\|_T, \left(1 + 2i \cdot k^{C_0 k^2}(k\eta T)\right)\|x^{(0)}(t)\|_T\right]$ for $\eta \leq \frac{1}{5T} \cdot k^{-C_1 k^2}$ with some $C_1 > C_0$.

At last, we set $x'(t) = x^{(k)}(t)$ and bound the distance between $x'(t)$ and $x^*(t)$ by

$$\begin{aligned}
\|x^{(k)}(t) - x^{(0)}(t)\|_T &\leq \sum_{i=1}^{k} \|x^{(i)}(t) - x^{(i-1)}(t)\|_T && \text{by triangle inequality} \\
&\leq \sum_{i=1}^{k} k^{C_0 k^2}(|f_i - f_i'|T)\|x^{(i-1)}(t)\|_T && \text{by Lemma 12.8.4} \\
&\leq \sum_{i=1}^{k} 2k^{C_0 k^2}(k\eta T)\|x^{(i-1)}(t)\|_T && \text{by } \max_i |f_i - f_i'| \leq k\eta \\
&\leq k \cdot 2k^{C_0 k^2}(k\eta T)\|x^*(t)\|_T \\
&\leq \delta \|x^*(t)\|_T
\end{aligned}$$

where the last inequality follows by the sufficiently small η. □

12.8.5 Approximating k-Fourier-sparse signals by polynomials

For any k-Fourier-sparse signal with frequency gap bounded away from zero, we show that there exists a low degree polynomial which is close to the original k-Fourier-sparse signal in $\|\cdot\|_T$ distance.

Lemma 12.8.6 (Existence of low degree polynomial)**.** *Let* $x^*(t) = \sum_{j=1}^{k} v_j e^{2\pi \mathbf{i} f_j t}$*, where* $\forall j \in [k], |f_j| \leq \Delta$ *and* $\min_{i \neq j} |f_i - f_j| \geq \eta$*. There exists a polynomial* $Q(t)$ *of degree*

$$d = O\left(T\Delta + k \log 1/(\eta T) + k^2 \log k + k \log(1/\delta)\right)$$

such that,

$$\|Q(t) - x^*(t)\|_T^2 \leq \delta \|x^*(t)\|_T^2 \tag{12.25}$$

Proof. For each frequency f_j, let $Q_j(t) = \sum_{k=0}^{d-1} \frac{(2\pi \mathbf{i} f_j t)^k}{k!}$ be the first d terms in the Taylor Expansion of $e^{2\pi \mathbf{i} f_j t}$. For any $t \in [0, T]$, we know the difference between $Q_j(t)$ and $e^{2\pi \mathbf{i} f_j t}$ is at most

$$|Q_j(t) - e^{2\pi \mathbf{i} f_j t}| \leq |\frac{(2\pi \mathbf{i} f_j T)^d}{d!}| \leq (\frac{2\pi T \Delta \cdot e}{d})^d.$$

We define

$$Q(t) = \sum_{j=1}^{k} v_j Q_j(t)$$

and bound the distance between Q and x^* from the above estimation:

$$\begin{aligned}
\|Q(t) - x^*(t)\|_T^2 &= \frac{1}{T}\int_0^T |Q(t) - x^*(t)|^2 \mathrm{d}t \\
&= \frac{1}{T}\int_0^T |\sum_{j=1}^k v_j (Q_j(t) - e^{2\pi \mathbf{i} f_j t})|^2 \mathrm{d}t \\
&\leq 2k \sum_{j=1}^k \frac{1}{T}\int_0^T |v_j|^2 \cdot |Q_j(t) - e^{2\pi \mathbf{i} f_j t}|^2 \mathrm{d}t \qquad \text{by triangle inequality} \\
&\leq k \sum_{j=1}^k |v_j|^2 \cdot (\frac{2\pi T\Delta \cdot e}{d})^{2d} \qquad \text{by Taylor expansion}
\end{aligned}$$

On the other hand, from Lemma 12.2.2, we know

$$\|x^*(t)\|_T^2 \geq (\eta T)^{2k} \cdot k^{-ck^2} \sum_j |v_j|^2.$$

Because $d = 10 \cdot \pi e(T\Delta + k \log 1/(\eta T) + k^2 \log k + k \log(1/\delta))$ is large enough, we have $k(\frac{2\pi T\Delta \cdot e}{d})^{2d} \leq \delta(\eta T)^{2k} \cdot k^{-ck^2}$, which indicates that $\|Q(t) - x^*(t)\|_T^2 \leq \delta \|x^*\|_T^2$ from all discussion above. □

12.8.6 Transferring degree-d polynomial to (d+1)-Fourier-sparse signal

In this section, we show how to transfer a degree-d polynomial to (d+1)-Fourier-sparse signal.

Lemma 12.8.7. *For any degree-d polynomial $Q(t) = \sum_{j=0}^{d} c_j t^j$, any $T > 0$ and any $\epsilon > 0$, there always exist $\gamma > 0$ and*

$$x^*(t) = \sum_{i=1}^{d+1} \alpha_i e^{2\pi \mathbf{i}(\gamma i)t}$$

with some coefficients $\alpha_0, \cdots, \alpha_d$ such that

$$\forall t \in [0, T], |x^*(t) - Q(t)| \leq \epsilon.$$

Proof. We can rewrite $x^*(t)$,

$$\begin{aligned}
x^*(t) &= \sum_{i=1}^{d+1} \alpha_i e^{2\pi \mathbf{i}\gamma i t} \\
&= \sum_{i=1}^{d+1} \alpha_i \sum_{j=0}^{\infty} \frac{(2\pi \mathbf{i}\gamma i t)^j}{j!} \\
&= \sum_{j=0}^{\infty} \frac{(2\pi \mathbf{i}\gamma t)^j}{j!} \sum_{i=1}^{d+1} \alpha_i \cdot i^j \\
&= \sum_{j=0}^{d} \frac{(2\pi \mathbf{i}\gamma t)^j}{j!} \sum_{i=1}^{d+1} \alpha_i \cdot i^j + \sum_{j=d+1}^{\infty} \frac{(2\pi \mathbf{i}\gamma t)^j}{j!} \sum_{i=1}^{d+1} \alpha_i \cdot i^j \\
&= Q(t) + \underbrace{\left(\sum_{j=0}^{d} \frac{(2\pi \mathbf{i}\gamma t)^j}{j!} \sum_{i=1}^{d+1} \alpha_i \cdot i^j - Q(t) \right)}_{C_1} + \underbrace{\left(\sum_{j=d+1}^{\infty} \frac{(2\pi \mathbf{i}\gamma t)^j}{j!} \sum_{i=1}^{d+1} \alpha_i \cdot i^j \right)}_{C_2}.
\end{aligned}$$

Our goal is to show there exists some parameter γ and coefficients $\{\alpha_0, \alpha_1, \cdots, \alpha_d\}$ such that

the term $C_1 = 0$ and $|C_2| \leq \epsilon$. Let's consider C_1,

$$C_1 = \sum_{j=0}^{d} (\frac{t}{T})^j \left(\frac{(2\pi \mathbf{i}\gamma T)^j}{j!} \sum_{i=1}^{d+1} \alpha_i i^j - c_j \right)$$

To guarantee $C_1 = 0$, we need to solve a linear system with $d+1$ unknown variables and $d+1$ constraints,

$$\begin{aligned} \text{Find} \quad & \alpha_1, \alpha_2, \cdots \alpha_{d+1} \\ \text{s.t.} \quad & \frac{(2\pi \mathbf{i}\gamma T)^j}{j!} \sum_{i=1}^{d+1} \alpha_i i^j - c_j = 0, \forall j \in \{0, 1, \cdots, d\} \end{aligned}$$

Define $c'_j = c_j j!/(2\pi \mathbf{i}\gamma)^j$, let α and c' be the length-$(d+1)$ column vectors with α_i and c'_j. Let $A \in \mathbb{R}^{d+1 \times d+1}$ denote the Vandermonde matrix where $A_{i,j} = i^j, \forall i, j \in [d+1] \times \{0, 1, \cdots, d\}$. Then we need to guarantee $A\alpha = c'$. Using the definition of determinant, $\det(A) = \prod_{i<j} |i-j| \leq 2^{O(d^2 \log d)}$. Thus $\sigma_{\max}(A) \leq 2^{O(d^2 \log d)}$ and then

$$\sigma_{\min}(A) = \frac{\det(A)}{\prod_{i=1}^{d-1} \sigma_i} \geq 2^{-O(d^3 \log d)}.$$

We show how to upper bound $|\alpha_i|$,

$$\max_{i \in [d+1]} |\alpha_i| \leq \|\alpha\|_2 = \|A^\dagger c'\|_2 \leq \|A^\dagger\|_2 \cdot \|c'\|_2 \leq \frac{1}{\sigma_{\min}(A)} \sqrt{d+1} \max_{0 \leq j \leq d} \frac{|c_j| j!}{(2\pi\gamma T)^j}$$

Plugging the above equation into C_2, we have

$$\begin{aligned}
|C_2| &= \left| \sum_{j=d+1}^{\infty} \frac{(2\pi \mathbf{i}\gamma t)^j}{j!} \sum_{i=1}^{d+1} \alpha_i \cdot i^j \right| \\
&\leq \sum_{j=d+1}^{\infty} \frac{(2\pi\gamma t)^j}{j!} \sum_{i=1}^{d+1} |\alpha_i| \cdot i^j \\
&\leq \sum_{j=d+1}^{\infty} \frac{(2\pi\gamma t)^j}{j!} (d+1)^{d+1} \max_{i\in[d+1]} |\alpha_i| \\
&\leq \sum_{j=d+1}^{\infty} \frac{(2\pi\gamma t)^j}{j!} (d+1)^{d+2} \frac{1}{\sigma_{\min}(A)} \frac{d!}{(2\pi\gamma T)^d} \max_{0\leq j\leq d} |c_j| \\
&\leq \epsilon
\end{aligned}$$

where the last step follows by choosing sufficiently small

$$\gamma \lesssim \epsilon / \left(T 2^{\Theta(d^3 \log d)} \max_{0\leq j\leq d} |c_j| \right).$$

□

12.9 k-cluster Signal Recovery

12.9.1 Overview

In this section, we prove Lemma 12.9.1 as the main technical lemma to finish the proof of main Theorem 12.1.1, which shows how to learn $x^*(t) = \sum_{j=1}^{k} v_j e^{2\pi \mathbf{i} f_j t}$ with noise.

Lemma 12.9.1. *Let $x^*(t) = \sum_{j=1}^{k} v_j e^{2\pi \mathbf{i} f_j t}$ and $x(t) = x^*(t) + g(t)$ be our observation. For any $\delta > 0$ and $T > 0$, let $\mathcal{N}^2 := \frac{1}{T}\int_0^T |g(t)|^2 \mathrm{d}t + \delta \cdot \frac{1}{T}\int_0^T |x^*(t)|^2 \mathrm{d}t$. For $\Delta = \mathrm{poly}(k, \log(1/\delta))/T$, Procedure* SignalRecoveryKCluster$^+$ *in Algorithm 12.8 takes $l = O(k)$ frequencies $\widetilde{f}_1, \cdots, \widetilde{f}_l$ as input and finds l polynomials $Q_1, \cdots, Q_l$ of degree $d = O((T\Delta)^{1.5} + k^3 \log k + k \log 1/\delta)$ such that*

$$\widetilde{x}(t) = \sum_{j \in [l]} Q_j(t) e^{2\pi \mathbf{i} \widetilde{f}_j t} \text{ satisfies } \|\widetilde{x}(t) - x^*(t)\|_T^2 \lesssim \mathcal{N}^2. \tag{12.26}$$

The procedure succeeds with probability at least $1 - 2^{-\Omega(k)}$, uses $\mathrm{poly}(k, \log(1/\delta)) \cdot \log(FT)$ samples, and runs in $\mathrm{poly}(k, \log(1/\delta)) \cdot \log^2(FT)$ time.

For any set $W = \{t_1, \cdots, t_m\}$ where each $t_i \in [0, T]$, we use

$$\|\vec{v}\|_W = \sqrt{\frac{\sum_{i \in W} |\vec{v}(t_i)|^2}{|W|}} \text{ for any } \vec{v} : [0, T] \to \mathbb{C}$$

in this section. We first show that Procedure SignalRecoveryKCluster succeeds with constant probability, then prove that Procedure SignalRecoveryKCluster$^+$ succeeds with probability at least $1 - 2^{-\Omega(k)}$.

12.9.2 Heavy clusters separation

Recall the definition of "heavy" clusters.

Definition 12.2.4. Given $x^*(t) = \sum_{j=1}^{k} v_j e^{2\pi \mathbf{i} f_j t}$, any $\mathcal{N} > 0$, and a filter function $(H, \widehat{H})$ with bounded support in frequency domain. Let L_j denote the interval of $\mathrm{supp}(\widehat{e^{2\pi \mathbf{i} f_j t} \cdot H})$ for each $j \in [k]$.

Define an equivalence relation $\sim$ on the frequencies f_i by the transitive closure of the relation $f_i \sim f_j$ if $L_i \cap L_j \neq \emptyset$. Let $S_1, \ldots, S_n$ be the equivalence classes under this relation.

Define $C_i = \bigcup_{f \in S_i} L_i$ for each $i \in [n]$. We say C_i is a "heavy" cluster iff $\int_{C_i} |\widehat{H \cdot x^*}(f)|^2 \mathrm{d}f \geq T \cdot \mathcal{N}^2/k$.

By reordering C_i, we can assume $\{C_1, C_2, \cdots, C_l\}$ are heavy clusters, where $l \leq n \leq k$.

Claim 12.2.5. Given $x^*(t) = \sum_{j=1}^{k} v_j e^{2\pi \mathbf{i} f_j t}$ and any $\mathcal{N} > 0$, let H be the filter function defined in Section 12.12.1 and $C_1, \cdots, C_l$ be the heavy clusters from Definition 12.2.4. For

$$S = \left\{ j \in [k] \middle| f_j \in C_1 \cup \cdots C_l \right\},$$

we have $x^{(S)}(t) = \sum_{j \in S} v_j e^{2\pi \mathbf{i} f_j t}$ approximating x^* within distance $\|x^{(S)}(t) - x^*(t)\|_T^2 \lesssim \mathcal{N}^2$.

Proof. Let $x^{(\overline{S})}(t) = \sum_{j \in [k] \setminus S} v_j e^{2\pi \mathbf{i} f_j t}$. Notice that $\|x^* - x^{(S)}\|_T^2 = \|x^{(\overline{S})}\|_T^2$.

From the property VI of filter function $(H, \widehat{H})$ in Section 12.12.1, we have

$$\int_{-\infty}^{+\infty} |x^{(\overline{S})}(t) \cdot H(t)|^2 \mathrm{d}t \geq 0.9 \int_0^T |x^{(\overline{S})}(t)|^2 \mathrm{d}t = 0.9 \cdot T \|x^{\overline{S}}\|_T^2.$$

From Definition 12.2.4, we have

$$\begin{aligned}\int_{-\infty}^{+\infty} |x^{(\overline{S})}(t)\cdot H(t)|^2 \mathrm{d}t &= \int_{-\infty}^{+\infty} |\widehat{x^{(\overline{S})}\cdot H}(f)|^2 \mathrm{d}f \\ &= \int_{[-\infty,+\infty]\setminus C_1\cup\cdots\cup C_l} |\widehat{x^*\cdot H}(f)|^2 \mathrm{d}f \\ &\leq k\cdot T\mathcal{N}^2/k.\end{aligned}$$

Overall, we have $\|x^{(\overline{S})}\|_T^2 \lesssim \mathcal{N}^2$. □

From the guarantee of Theorem 12.2.6, for any $j\in S$, $\min_{i\in[l]}|f_j-\widetilde{f}_i|\leq \Delta\sqrt{\Delta T}$. From now on, we focus on the recovery of $x^{(S)}$, which is enough to approximate x^* from the above claim. Because we are looking for $\widetilde{x}$ approximating $x^{(S)}$ within distance $O(\mathcal{N}^2)$, from Lemma 12.2.1, we can assume there is a frequency gap $\eta\geq \frac{\delta}{10T}k^{-O(k^2)}$ among $x^{(S)}$.

12.9.3 Approximating clusters by polynomials

In this section, we show how to approximate $x^{(S)}$ by $x'(t) = \sum_{i\in[l]} e^{2\pi \mathbf{i}\widetilde{f}_i t} P_i(t)$ where $P_1, \cdots, P_l$ are low degree polynomials.

Claim 12.9.2. *For any $x^{(S)}(t) = \sum_{j\in S} v_j e^{2\pi \mathbf{i} f_j t}$ with a frequency gap $\eta = \min_{i\neq j}|f_i - f_j|$ and l frequencies $\widetilde{f}_1, \cdots, \widetilde{f}_l$ with the property $\forall j \in S, \min_{i\in[l]} |f_j - \widetilde{f}_i| \leq \Delta\sqrt{\Delta T}$, let*

$$d = 5\pi\left((T\Delta)^{1.5} + k^3 \log k + \log 1/\delta\right) \text{ and } V = \left\{t^j e^{2\pi \mathbf{i}\widetilde{f}_i t} | i \in [l], j \in \{0, \cdots, d\}\right\}.$$

There exists $x'(t) \in \text{span}\{V\}$ that approximates $x^{(S)}(t)$ as follows:

$$\forall t \in [0, T], |x'(t) - x^{(S)}(t)| \leq \delta \|x^{(S)}\|_T.$$

Proof. From Lemma 12.2.2, we know

$$\|x^{(S)}\|_T^2 \geq (\eta T)^{2k} \cdot k^{-ck^2} \sum_{j\in S} |v_j|^2.$$

For each frequency f_j, we use p_j to denote the index in $[l]$ such that $|f_j - \widetilde{f}_{p_j}| \leq \Delta\sqrt{\Delta T}$. We rewrite

$$x^{(S)}(t) = \sum_{i=1}^{l} e^{2\pi \mathbf{i}\widetilde{f}_i} \left(\sum_{j\in S: p_j = i} v_j e^{2\pi \mathbf{i}(f_j - \widetilde{f}_i)t}\right).$$

For $d = 5\pi((T\Delta)^{1.5} + k^3 \log k + \log 1/\delta)$ and each $e^{2\pi \mathbf{i}(f_j - \widetilde{f}_{p_j})t}$, let $Q_j(t) = \sum_{i=0}^{d-1} \frac{\left(2\pi \mathbf{i}(f_j - \widetilde{f}_{p_j})t\right)^i}{i!}$ be the first d terms in the Taylor Expansion of $e^{2\pi \mathbf{i}(f_j - \widetilde{f}_{p_j})t}$. For any $t \in [0, T]$, we know the difference between $Q_j(t)$ and $e^{2\pi \mathbf{i}(f_j - \widetilde{f}_{p_j})t}$ is at most

$$\forall t \in [0, T], |Q_j(t) - e^{2\pi \mathbf{i}(f_j - \widetilde{f}_{p_j})t}| \leq |\frac{(2\pi \mathbf{i}(f_j - \widetilde{f}_{p_j})T)^d}{d!}| \leq (\frac{8\pi(\Delta T)^{1.5}}{d})^d.$$

Let $x' = \sum_{i=1}^{l} e^{2\pi \mathbf{i} \widetilde{f}_i t} \left(\sum_{j \in S: p_j = i} v_j Q_j(t) \right)$. From all discussion above, we know for any $t \in [0, T]$,

$$\begin{aligned} |x'(t) - x^{(S)}(t)|^2 &\leq \left(\sum_{j \in S} |v_j| (\frac{8\pi (T\Delta)^{1.5}}{d})^d \right)^2 \\ &\leq k (\frac{8\pi (T\Delta)^{1.5}}{d})^{2d} \sum_j |v_j|^2 \\ &\leq \frac{k (\frac{8\pi (T\Delta)^{1.5}}{d})^{2d}}{(\eta T)^{2k} \cdot k^{-ck^2}} \|x^{(S)}\|_T^2 \\ &\leq \delta^2 \|x^{(S)}\|_T^2. \end{aligned}$$

□

We provide a property of functions in span$\{V\}$ such that we can use the Chernoff bound and the ϵ-net argument on vectors in span$\{V\}$.

Claim 12.2.7. For any $\vec{u} \in \text{span} \left\{ e^{2\pi \mathbf{i} \widetilde{f}_i t} \cdot t^j \middle| j \in \{0, \cdots, d\}, i \in [l] \right\}$, there exists some universal constants $C_1 \leq 4$ and $C_2 \leq 3$ such that

$$\max_{t \in [0,T]} \{|\vec{u}(t)|^2\} \lesssim (ld)^{C_1} \log^{C_2}(ld) \cdot \|\vec{u}\|_T^2$$

Proof. From Lemma 12.8.7, we can approximate each polynomial in $\vec{u}$ by a linear combination of $\{1, e^{2\pi \mathbf{i} \cdot \gamma t}, \cdots, e^{2\pi \mathbf{i} \cdot (\gamma d) t}\}$ such that we obtain $u^* \in \text{span} \left\{ e^{2\pi \mathbf{i} \cdot (\gamma j) t} \cdot e^{2\pi \mathbf{i} \widetilde{f}_i t} | i \in [l], j \in \{0, \cdots, d+1\} \right\}$ for some small γ such that $\forall t \in [0, T], |\vec{u}(t) - u^*(t)| \leq 0.01 \|\vec{u}\|_T$.

From Lemma 12.5.1, we know

$$\max_{t \in [0,T]} |u^*(t)|^2 \leq C \cdot \left((ld+1)^4 \cdot \log^3(ld+1) \right) \|u^*\|_T^2.$$

For some constant C', we have

$$\max_{t\in[0,T]} |\vec{u}(t)|^2 \leq C' \left((kd)^{C_1} \log^{C_2} d\right) \|\vec{u}\|_T^2.$$

□

12.9.4 Main result, with constant success probability

In this section, we show that the output $\widetilde{x}$ is close to x' with high probability using the ϵ-net argument, which is enough to prove $\|\widetilde{x}-x\|_T \lesssim \mathcal{N}^2$ from all discussion above. Because we can prove Lemma 12.9.6(which is the main goal of this section), then combining $\|x'-x^*\|_T \leq \|x'-x^{(S)}\|_T + \|x^{(S)}-x^*\|_T \lesssim \delta\|x^*\|_T$ and Lemma 12.9.6, we have $\|x^*-\widetilde{x}\|_T \lesssim \|g\|_T + \delta\|x^*\|_T$, which finishes the proof of Procedure SignalRecoveryKCluster in Algorithm 12.8 achieving the Equation (12.26) with constant success probability but not $1-2^{-\Omega(k)}$. We will boost the success probability in Section 12.9.5.

We first provide an ϵ-net $\mathcal{P}$ for the unit vectors $\mathcal{Q} = \{\vec{u} \in \mathrm{span}\{V\} \big| \|\vec{u}\|_T^2 = 1\}$ in the linear subspace $\mathrm{span}\{V\}$ where $V = \left\{ t^j \cdot e^{2\pi \mathbf{i} \widetilde{f}_i t} \big| j \in \{0,1,\cdots,d\}, i \in [l] \right\}$ from the above discussion. Notice that the dimension of $\mathrm{span}\{V\}$ is at most $l(d+1)$.

Claim 12.9.3. *There exists an ϵ-net $\mathcal{P} \subset \mathrm{span}\{V\}$ such that*

1. $\forall \vec{u} \in Q, \exists \vec{w} \in \mathcal{P}, \|\vec{u}-\vec{w}\|_T \leq \epsilon$.
2. $|\mathcal{P}| \leq \left(5\frac{l(d+1)}{\epsilon}\right)^{2l(d+1)}$.

Proof. Let $\mathcal{P}'$ be an $\frac{\epsilon}{l(d+1)}$-net in the unit circle of $\mathbb{C}$ with size at most $(4\frac{l(d+1)}{\epsilon}+1)^2$, i.e.,

$$\mathcal{P}' = \left\{ \frac{\epsilon}{2l(d+1)} j_1 + \mathbf{i}\frac{\epsilon}{2l(d+1)} j_2 \Big| j_1, j_2 \in \mathbb{Z}, |j_1| \leq \frac{2l(d+1)}{\epsilon}, |j_2| \leq \frac{2l(d+1)}{\epsilon} \right\}.$$

Observe that the dimension of $\mathrm{span}\{V\}$ is at most $l(d+1)$. Then we take an orthogonal basis $\vec{w}_1, \cdots, \vec{w}_{l(d+1)}$ in $\mathrm{span}\{V\}$ and set

$$P = \{ \sum_{i=1}^{l(d+1)} \alpha_i \vec{w}_i \big| \forall i \in [l(d+1)], \alpha_i \in \mathcal{P}' \}.$$

Therefore $\mathcal{P}$ is an ϵ-net for Q and $|\mathcal{P}| \leq \left(5\frac{l(d+1)}{\epsilon}\right)^{2l(d+1)}$. □

We first prove that W is a good estimation for all functions in the ϵ-net $\mathcal{P}$.

Claim 12.9.4. *For any $\epsilon > 0$, there exists a universal constant $C_3 \leq 5$ such that for a set S of i.i.d. samples chosen uniformly at random over $[0, T]$ of size $|S| \geq \frac{3(kd)^{C_3}\log^{C_3} d/\epsilon}{\epsilon^2}$,then with probability at least $1 - k^{-k}$, for all $\vec{w} \in \mathcal{P}$, we have*

$$\|\vec{w}\|_W \in \left[(1-\epsilon)\|\vec{w}\|_T, (1+\epsilon)\|\vec{w}\|_T\right].$$

Proof. From Claim 12.2.7 and Lemma 12.3.5, for each $\vec{w} \in \mathcal{P}$,

$$\Pr\left[\|\vec{w}(t)\|_W \notin \left[(1-\epsilon)\|\vec{w}\|_T, (1+\epsilon)\|\vec{w}\|_T\right]\right] \leq 2^{-\frac{|W|\epsilon^2}{3(kd)^{C_1}\log^{C_2+0.5} d}} \leq 2^{-kd\log^{1.5}\frac{d}{\epsilon}}.$$

From the union bound, $\|\vec{w}\|_W \in [(1-\epsilon)\|\vec{w}\|_T, (1+\epsilon)\|\vec{w}\|_T]$ for any $\vec{w} \in \mathcal{P}$ with probability at least $1 - (\frac{d}{\epsilon})^{-kd\log^{0.5} d} \cdot |\mathcal{P}| \geq 1 - d^{-d}$. □

Then We prove that W is a good estimation for all functions in $\text{span}\{V\}$ using the property of ϵ-nets.

Claim 12.9.5. *For any $\epsilon > 0$, there exists a universal constant $C_3 \leq 5$ such that for a set W of i.i.d. samples chosen uniformly at random over $[0, T]$ of size $|W| \geq \frac{3(kd)^{C_3}\log^{C_3} d/\epsilon}{\epsilon^2}$,then with probability at least $1 - d^{-d}$, for all $u \in \text{span}\{V\}$, we have*

$$\|\vec{u}\|_W \in \left[(1-3\epsilon)\|\vec{u}\|_T, (1+3\epsilon)\|\vec{u}\|_T\right]$$

Proof. We assume that the above claim is true for any $\vec{w} \in \mathcal{P}$. Without loss of generality, we consider $\vec{u} \in \Omega$ such that $\|\vec{u}\|_T = 1$.

Let $\vec{w}_0$ be the vector in $\mathcal{P}$ that minimizes $\|\vec{w}-\vec{u}\|_T$ for all $\vec{w} \in \mathcal{P}$, i.e., $\vec{w}_0 = \arg\min_{\vec{w}\in\mathcal{P}} \|\vec{w}-\vec{u}\|_T$. Define $\vec{u}_1 = \vec{u} - \vec{w}_0$ and notice that $\|\vec{u}_1\|_T \leq \epsilon$ because $\mathcal{P}$ is a ϵ-net. If $\|\vec{u}_1\|_T = 0$, then we skip the rest of this procedure. Otherwise, we define $\alpha_1 = \|\vec{u}_1\|_T$ and normalize $\widetilde{u}_1 = \vec{u}_1/\alpha_1$.

Then we choose $\vec{w}_1$ to be the vector in $\mathcal{P}$ that minimizes $\|\vec{w} - \widetilde{u}_1\|_T$ for all $\vec{w} \in \mathcal{P}$. Similarly, we set $\vec{u}_2 = \widetilde{u}_1 - \vec{w}_1$ and $\alpha_2 = \|\vec{u}_2\|_T$. Next we repeat this process for $\widetilde{u}_2 = \vec{u}_2/\alpha_2$ and so on. The recursive definition can be summarized in the following sense,

$$
\begin{aligned}
\text{initial}: \quad & \widetilde{u}_0 = \vec{u} \text{ and } m = 10\log_{1/\epsilon}(ld) + 1, \\
\text{For } i \in \{0,1,2,\cdots,m\}: \quad & \vec{w}_i = \arg\min_{\vec{w}\in\mathcal{P}} \|\vec{w} - \widetilde{u}_i\|_T, \\
& \vec{u}_{i+1} = \widetilde{u}_i - \vec{w}_i \text{ and } \alpha_{i+1} = \|\vec{u}_{i+1}\|_T, \\
& \text{if } \alpha_{i+1} = 0, \text{ stop.} \\
& \text{if } \alpha_{i+1} \neq 0, \widetilde{u}_{i+1} = \vec{u}_{i+1}/\alpha_{i+1} \text{ and continue,}
\end{aligned}
$$

Eventually, we have $\vec{u} = \vec{w}_0 + \alpha_1\vec{w}_1 + \alpha_1\alpha_2\vec{w}_2 + \cdots + \prod_{j=1}^{m}\alpha_j(\vec{w}_m + \vec{u}_{m+1})$ where each $|\alpha_i| \leq \epsilon$ and each $\vec{w}_i$ is in the ϵ-net $\mathcal{P}$. Notice that $\|\vec{u}_{m+1}\|_T \leq 1$ and $\|\vec{u}_{m+1}\|_W \leq (ld+1)^3 \cdot \|\vec{u}_{m+1}\|_T$ from Claim 12.2.7. We prove a lower bound for $\|\vec{u}\|_W$,

$$
\begin{aligned}
\|\vec{u}\|_W &= \|\vec{w}_0 + \alpha_1\vec{w}_1 + \alpha_1\alpha_2\vec{w}_2 + \cdots + \prod_{j=1}^{m}\alpha_j(\vec{w}_m + \vec{u}_{m+1})\|_W \\
&\geq \|\vec{w}_0\|_W - \|\alpha_1\vec{w}_1\|_W - \|\alpha_1\alpha_2\vec{w}_2\|_W - \cdots - \|\prod_{j=1}^{m}\alpha_j\vec{w}_m\|_W - \|\prod_{j=1}^{m}\alpha_j\vec{u}_{m+1}\|_W \\
&\geq (1-\epsilon) - \epsilon(1+\epsilon) - \epsilon^2(1+\epsilon) - \cdots - \epsilon^m(1+\epsilon) - \epsilon^m\|\vec{u}_{m+1}\|_W \\
&\geq 1 - \epsilon - \frac{(1+\epsilon)\epsilon}{1-\epsilon} - \epsilon^m \cdot (ld+1)^3 \geq 1 - 3\epsilon.
\end{aligned}
$$

Similarly, we have $\|\vec{u}\|_W \leq 1 + 3\epsilon$. □

Lemma 12.9.6. *With probability at least* 0.99 *over the* m *i.i.d samples in* W*,*

$$\|x'(t) - \widetilde{x}(t)\|_T \leq 2200 \left(\|g(t)\|_T + \|x^{(S)}(t) - x'(t)\|_T \right).$$

Proof. Let $g'(t) = g(t) + x^*(t) - x'(t)$ such that $x(t) = x'(t) + g'(t)$. Then we choose $\epsilon = 0.03$ and bound:

$$\begin{aligned}
&\|x'(t) - \widetilde{x}(t)\|_T \\
&\leq (1+3\epsilon)\|x'(t) - \widetilde{x}(t)\|_W && \text{with prob. } 1 - 2^{-\Omega(d \log d)} \text{ by Claim 12.9.5} \\
&= 1.09\|x'(t) - \widetilde{x}(t)\|_W && \text{by } \epsilon = 0.03 \\
&= 1.09\|x(t) - g'(t) - \widetilde{x}(t)\|_W && \text{by } x'(t) = x(t) - g'(t) \\
&\leq 1.09\|x(t) - \widetilde{x}(t)\|_W + 1.09\|g'(t)\|_W && \text{by triangle inequality} \\
&\leq 1.09\|x(t) - x'(t)\|_W + 1.09\|g'(t)\|_W && \text{by } \widetilde{x} = \operatorname*{arg\,min}_{y \in \text{span}\{V\}} \|x - y\|_W \\
&= 2.18\|g'(t)\|_W. && \text{by } x(t) - x'(t) = g(t)
\end{aligned}$$

From the fact that $\mathbb{E}_W[\|g'\|_W] = \|g'\|_T$, $\|g'\|_W \leq 1000\|g'\|_T$ with probability at least .999. It indicates $\|x'(t) - \widetilde{x}(t)\|_T \leq 2200\|g'\|_T$ with probability at least 0.99 from all discussion above. □

12.9.5 Boosting the success probability

In order to achieve $1 - 2^{-\Omega(k)}$ for the main theorem, we cannot combine Procedure SignalRecoveryKCluster with FrequencyRecoveryKCluster directly. However, using the similar proof technique in Theorem 12.4.5, we are able to boost the success probability by using Procedure SignalRecoveryKCluster$^+$ in Algorithm 12.8. It runs Procedure SignalRecoveryKCluster $R = O(k)$ times in parallel for independent fresh samples and report R different d-Fourier-sparse signals $\widetilde{x}_i(t)$. Then, taking $m = \mathrm{poly}(k)$ new locations $\{t_1, t_2, \cdots, t_m\}$, and computing $\widetilde{A}$ as before and $\widetilde{b}_j$ by taking the median of $\{\widetilde{x}_1(t_j), \cdots, \widetilde{x}_R(t_j)\}$. At the end, solving the linear regression for matrix $\widetilde{A}$ and vector $\widetilde{b}$. Thus, we complete the proof of Lemma 12.9.1.

Because we can transfer a degree-d polynomial to a d-Fourier-sparse signal by Lemma 12.8.7, the output of Procedure CFTKCluster in Algorithm 12.8 matches the main theorem,

Theorem 12.1.1. Let $x(t) = x^*(t) + g(t)$, where x^* is k-Fourier-sparse signal with frequencies in $[-F, F]$. Given samples of x over $[0, T]$ we can output $\widetilde{x}(t)$ such that with probability at least $1 - 2^{-\Omega(k)}$,

$$\|\widetilde{x} - x^*\|_T \lesssim \|g\|_T + \delta \|x^*\|_T .$$

Our algorithm uses $\mathrm{poly}(k, \log(1/\delta)) \cdot \log(FT)$ samples and $\mathrm{poly}(k, \log(1/\delta)) \cdot \log^2(FT)$ time. The output $\widetilde{x}$ is $\mathrm{poly}(k, \log(1/\delta))$-Fourier-sparse signal.

12.10 Technical Proofs

12.10.1 Proof of Theorem 12.8.2

We prove the following Theorem

Theorem 12.8.2. For real numbers $\xi_1, \ldots, \xi_k$, let $G_{\xi_1,\ldots,\xi_k}$ be the matrix whose (i,j)-entry is

$$\int_{-1}^{1} e^{2\pi \mathbf{i}(\xi_i - \xi_j)t} dt.$$

Then

$$\det(G_{\xi_1,\ldots,\xi_k}) = 2^{\widetilde{O}(k^2)} \prod_{i<j} \min(|\xi_i - \xi_j|^2, 1).$$

First, we note by the Cauchy-Binet formula that the determinant in question is equal to

$$\int_{-1}^{1}\int_{-1}^{1}\ldots\int_{-1}^{1} \left|\det([e^{2\pi i \xi_i t_j}]_{i,j})\right|^2 dt_1 dt_2 \ldots dt_k. \tag{12.27}$$

We next need to consider the integrand in the special case when $\sum |\xi_i| \leq 1/8$.

Lemma 12.10.1. *If $\xi_i \in R$ and $t_j \in \mathbb{R}$, $\sum_i |\xi_i|(\max_i |t_i|) \leq 1/8$ then*

$$|\det([e^{2\pi i \xi_i t_j}]_{i,j})| = \Theta\left(\frac{(2\pi)^{\binom{k}{2}} \prod_{i<j} |t_i - t_j||\xi_i - \xi_j|}{1!2!\cdots k!}\right).$$

Proof. Firstly, by adding a constant to all the t_j we can make them non-negative. This multiplies the determinant by a root of unity, and at most doubles $\sum_i |\xi_i|(\max_i |t_i|)$.

By continuity, it suffices to consider the t_i to all be multiples of $1/N$ for some large integer N. By multiplying all the t_j by N and all ξ_i by $1/N$, we may assume that all of the t_j are non-negative integers with $t_1 \leq t_2 \leq \ldots \leq t_k$.

Let $z_i = \exp(2\pi i \xi_i)$. Then our determinant is

$$\det\left(\left[z_i^{t_j}\right]_{i,j}\right),$$

which is equal to the Vandermonde determinant times the Schur polynomial $s_\lambda(z_i)$ where λ is the partition $\lambda_j = t_j - (j-1)$.

Therefore, this determinant equals

$$\prod_{i<j}(z_i - z_j) s_\lambda(z_1, z_2, \ldots, z_k).$$

The absolute value of

$$\prod_{i<j}(z_i - z_j)$$

is approximately $\prod_{i<j}(2\pi i)(\xi_i - \xi_j)$, which has absolute value $(2\pi)^{\binom{k}{2}} \prod_{i<j} |\xi_i - \xi_j|$. We have left to evaluate the size of the Schur polynomial.

By standard results, s_λ is a polynomial in the z_i with non-negative coefficients, and all exponents at most $\max_j |t_j|$ in each variable. Therefore, the monomials with non-zero coefficients will all have real part at least $1/2$ and absolute value 1 when evaluated at the z_i. Therefore,

$$|s_\lambda(z_1, \ldots, z_k)| = \Theta(|s_\lambda(1, 1, \ldots, 1)|).$$

On the other hand, by the Weyl character formula

$$s_\lambda(1, 1, \ldots, 1) = \prod_{i<j} \frac{t_j - t_i}{j - i} = \frac{\prod_{i<j} |t_i - t_j|}{1!2!\ldots k!}.$$

This completes the proof. □

Next we prove our Theorem when the ξ have small total variation.

Lemma 12.10.2. *If there exists a ξ_0 so that $\sum |\xi_i - \xi_0| < 1/8$, then*

$$\det(G_{\xi_1,\ldots,\xi_k}) = \Theta\left(\frac{2^{3k(k-1)/2}\pi^{k(k-1)}\prod_{i<j}|\xi_i - \xi_j|^2}{(k!)^3\prod_{n=0}^{k-1}(2n)!}\right).$$

Proof. By translating the ξ_i we can assume that $\xi_0 = 0$.

By the above we have

$$\Theta(\frac{(2\pi)^{k(k-1)}\prod_{i<j}|\xi_i - \xi_j|^2}{(1!2!\cdots k!)^2})\int_{-1}^{1}\cdots\int_{-1}^{1}\prod_{i<j}|t_i - t_j|^2 dt_1\ldots dt_k.$$

We note that by the Cauchy-Binet formula the latter term is the determinant of the matrix M with $M_{i,j} = \int_{-1}^{1} t^{i+j}dt$. This is the Graham matrix associated to the polynomials t^i for $0 \leq i \leq k-1$. Applying Graham-Schmidt (without the renormalization step) to this set yields the basis $P_n\alpha_n$ where $\alpha_n = \frac{2^n(n!)^2}{(2n)!}$ is the inverse of the leading term of P_n. This polynomial has norm $\alpha_n^2 2/(2n+1)$. Therefore, the integral over the t_i yields

$$\prod_{n=0}^{k-1}\frac{2^{n+1}(n!)^2}{(n+1)(2n)!}.$$

This completes the proof. □

Next we extend this result to the case that all the ξ are within poly(k) of each other.

Proposition 12.10.3. *If there exists a ξ_0 so that $|\xi_i - \xi_0| = \text{poly}(k)$ for all i, then*

$$\det(G_{\xi_1,\ldots,\xi_k}) = 2^{\widetilde{O}(k^2)}\prod_{i<j}\min(|\xi_i - \xi_j|^2, 1).$$

Proof. We begin by proving the lower bound. We note that for $0 < x < 1$,

$$\det(G_{\xi_1,\ldots,\xi_k}) \geq \int_{-x}^{x}\int_{-x}^{x}\cdots\int_{-1}^{1}\left|\det([e^{2\pi i\xi_i t_j}]_{i,j})\right|^2 dt_1 dt_2\ldots dt_k = x^k\det(G_{\xi_1/x,\xi_2/x,\ldots,\xi_k/k}).$$

Taking $x = 1/\operatorname{poly}(k)$, we may apply the above Lemma to compute the determinant on the right hand side, yielding an appropriate lower bound.

To prove the lower bound, we note that we can divide our ξ_i into clusters, $\mathcal{C}_i$, where for any i, j in the same cluster $|\xi_i - \xi_j| < 1/k$ and for i and j in different clusters $|\xi_i - \xi_j| \geq 1/k^2$. We then note as a property of Graham matrices that

$$\det(G_{\xi_1,\ldots,\xi_k}) \leq \prod_{\mathcal{C}_i} \det(G_{\{\xi_j \in \mathcal{C}_i\}}) = 2^{\widetilde{O}(k^2)} \prod_{i<j, \text{ in same cluster}} |\xi_i - \xi_j|^2 = 2^{\widetilde{O}(k^2)} \prod_{i<j} |\xi_i - \xi_j|^2.$$

This completes the proof. □

Finally, we are ready to prove our Theorem.

Proof. Let $I(t)$ be the indicator function of the interval $[-1, 1]$.

Recall that there is a function $h(t)$ so that for any function f that is a linear combination of at most k complex exponentials that $|h(t)f(t)|_2 = \Theta(|I(t)f(t)|_2)$ and so that $\widehat{h}$ is supported on an interval of length $\operatorname{poly}(k) < k^C$ about the origin.

Note that we can divide our ξ_i into clusters, $\mathcal{C}$, so that for i and j in a cluster $|\xi_i - \xi_j| < k^{C+1}$ and for i and j in different clusters $|\xi_i - \xi_j| > k^C$.

Let $\widetilde{G}_{\xi_1,\xi_2,\ldots,\xi'_k}$ be the matrix with (i,j)-entry $\int_{\mathbb{R}} |h(t)|^2 e^{(2\pi i)(\xi_i - \xi_j)t} dt$.

We claim that for any $k' \leq k$ that

$$\det(\widetilde{G}_{\xi_1,\xi_2,\ldots,\xi'_k}) = 2^{O(k')} \det(G_{\xi_1,\xi_2,\ldots,\xi'_k}).$$

This is because both are Graham determinants, one for the set of functions $I(t)\exp((2\pi i)\xi_j t)$ and the other for $h(t)\exp((2\pi i)\xi_j t)$. However since any linear combination of the former has

L^2 norm a constant multiple of that the same linear combination of the latter, we have that

$$\widetilde{G}_{\xi_1,\xi_2,\ldots,\xi_k'} = \Theta(G_{\xi_1,\xi_2,\ldots,\xi_k'})$$

as self-adjoint matrices. This implies the appropriate bound.

Therefore, we have that

$$\det(G_{\xi_1,\ldots,\xi_k}) = 2^{O(k)} \det(\widetilde{G}_{\xi_1,\ldots,\xi_k}).$$

However, note that by the Fourier support of h that

$$\int_{\mathbb{R}} |h(t)|^2 e^{(2\pi i)(\xi_i - \xi_j)t} dt = 0$$

if $|\xi_i - \xi_j| > k^C$, which happens if i and j are in different clusters. Therefore $\widetilde{G}$ is block diagonal and hence its determinant equals

$$\det(\widetilde{G}_{\xi_1,\ldots,\xi_k}) = \prod_{\mathfrak{C}} \det(\widetilde{G}_{\{\xi_j \in \mathfrak{C}_i\}}) = 2^{O(k)} \prod_{\mathfrak{C}} \det(G_{\{\xi_j \in \mathfrak{C}_i\}}).$$

However the Proposition above shows that

$$\prod_{\mathfrak{C}} \det(G_{\{\xi_j \in \mathfrak{C}_i\}}) = 2^{\widetilde{O}(k^2)} \prod_{i<j} \min(1, |\xi_i - \xi_j|^2).$$

This completes the proof. □

12.10.2 Proofs of Lemma 12.5.3 and Lemma 12.5.4

We fix $z_1, \cdots, z_k$ to be complex numbers on the unit circle and use $Q(z)$ to denote the degree-k polynomial $\prod_{i=1}^{k}(z - z_i)$.

Lemma 12.5.3. Let $Q(z)$ be a degree k polynomial, all of whose roots are complex numbers with absolute value 1. For any integer n, let $r_{n,k}(z) = \sum_{l=0}^{k-1} r_{n,k}^{(l)} \cdot z^l$ denote the residual polynomial of

$$r_{n,k}(z) \equiv z^n \pmod{Q(z)}.$$

Then, each coefficient of $r_{n,k}$ is bounded: $|r_{n,k}^{(l)}| \leq 2^k n^{k-1}$ for any l.

Proof. By definition, $r_{n,k}(z_i) = z_i^n$. From the polynomial interpolation, we have

$$r_{n,k}(z) = \sum_{i=1}^{k} \frac{\prod_{j \in [k] \setminus i}(z - z_j) z_i^n}{\prod_{j \in [k] \setminus i}(z_i - z_j)}.$$

Let $\mathrm{Sym}_{S,i}$ be the symmetry polynomial of $z_1, \cdots, z_k$ with degree i among subset $S \subseteq [k]$, i.e., $\mathrm{Sym}_{S,i} = \sum_{S' \subseteq \binom{S}{i}} \prod_{j \in S'} z_j$. Then

$$r_{n,k}^{(l)} = (-1)^{k-1-l} \sum_{i=1}^{k} \frac{\mathrm{Sym}_{[k] \setminus i, k-1-l} \cdot z_i^n}{\prod_{j \in [k] \setminus i}(z_i - z_j)}.$$

We omit $(-1)^{k-1-l}$ in the rest of proof and use induction on n, k, and l to prove $|r_{n,k}^{(l)}| \leq \binom{k-1}{l}\binom{n}{k-1}$.

Base Case of n: For any $n < k$, from the definition, $r(z) = z^n$ and $|r_{n,k}^{(l)}| \leq 1$.

Suppose it is true for any $n < n_0$. We consider $r_{n_0,k}^{l}$ from now on. When $k = 1$, $r_{n,0} = z_1^n$ is bounded by 1 because z_1 is on the unit circle of $\mathbb{C}$.

Given n_0, suppose the induction hypothesis is true for any $k < k_0$ and any $l < k$. For $k = k_0$, we first prove that $|r^{(k_0-1)}_{n_0,k_0}| \leq \binom{n_0}{k_0-1}$ then prove that $|r^{(l)}_{n_0,k_0}| \leq \binom{k_0-1}{l}\binom{n_0}{k_0-1}$ for $l = k_0 - 2, \cdots, 0$.

$$
\begin{aligned}
r^{(k_0-1)}_{n_0,k_0} &= \sum_{i=1}^{k_0} \frac{z_i^{n_0}}{\prod\limits_{j\in[k_0]\setminus i}(z_i - z_j)} \\
&= \sum_{i=1}^{k_0-1} \frac{z_i^{n_0}}{\prod\limits_{j\in[k_0]\setminus i}(z_i - z_j)} + \frac{z_{k_0}^{n_0}}{\prod\limits_{j\in[k_0]\setminus k_0}(z_{k_0} - z_j)} \\
&= \sum_{i=1}^{k_0-1} \frac{z_i^{n_0} - z_i^{n_0-1}z_{k_0} + z_i^{n_0-1}z_{k_0}}{\prod\limits_{j\in[k_0]\setminus i}(z_i - z_j)} + \frac{z_{k_0}^{n_0}}{\prod\limits_{j\in k_0\setminus k_0}(z_{k_0} - z_j)} \\
&= \sum_{i=1}^{k_0-1} \left(\frac{z_i^{n_0-1}}{\prod\limits_{j\in[k_0-1]\setminus i}(z_i - z_j)} + \frac{z_i^{n_0-1}z_{k_0}}{\prod\limits_{j\in k_0\setminus i}(z_i - z_j)} \right) + \frac{z_{k_0}^{n_0}}{\prod\limits_{j\in k_0\setminus k_0}(z_{k_0} - z_j)} \\
&= \left(\sum_{i=1}^{k_0-1} \frac{z_i^{n_0-1}}{\prod\limits_{j\in[k_0-1]\setminus i}(z_i - z_j)} \right) + \left(z_{k_0} \sum_{i=1}^{k_0} \frac{z_i^{n_0-1}}{\prod\limits_{j\in k_0\setminus i}(z_i - z_j)} \right) \\
&= r^{(k_0-2)}_{n_0-1,k_0-1} + z_{k_0} \cdot r^{(k_0-1)}_{n_0-1,k_0}
\end{aligned}
$$

Hence $|r^{(k_0-1)}_{n_0,k_0}| \le |r^{(k_0-2)}_{n_0-1,[k_0-1]}| + |r^{(k_0-1)}_{n_0-1,k_0}| \le \binom{n_0-2}{k_0-2} + \binom{n_0-2}{k_0-1} = \binom{n_0-1}{k_0-1}$. For $l < k_0 - 1$, we have

$$\begin{aligned}
r^{(l)}_{n_0,k_0} &= \sum_{i=1}^{k_0} \frac{\mathrm{Sym}_{[k_0]\setminus i,k_0-1-l} \cdot z_i^{n_0}}{\prod\limits_{j\in[k_0]\setminus i}(z_i - z_j)} \quad \text{let } l' = k_0 - 1 - l \\
&= \sum_{i=1}^{k_0-1} \frac{\left(\mathrm{Sym}_{[k_0-1]\setminus i,l'} + \mathrm{Sym}_{[k_0-1]\setminus i,l'-1} \cdot z_{k_0}\right) z_i^{n_0}}{\prod\limits_{j\in[k_0]\setminus i}(z_i - z_j)} + \frac{\mathrm{Sym}_{[k_0-1],l'} \cdot z_{k_0}^{n_0}}{\prod\limits_{j<k_0}(z_{k_0} - z_j)} \\
&= \sum_{i=1}^{k_0-1} \frac{\mathrm{Sym}_{[k_0-1]\setminus i,l'} \cdot (z_i - z_{k_0}) z_i^{n_0-1} + \mathrm{Sym}_{[k_0-1]\setminus i,l'} \cdot z_{k_0} z_i^{n_0-1} + \mathrm{Sym}_{[k_0-1]\setminus i,l'-1} \cdot z_{k_0} z_i^{n_0}}{\prod\limits_{j\in[k_0]\setminus i}(z_i - z_j)} \\
&\quad + \frac{\mathrm{Sym}_{[k_0-1],l'} \cdot z_{k_0}^{n_0}}{\prod\limits_{j<k_0}(z_{k_0} - z_j)} \\
&= \sum_{i=1}^{k_0-1} \frac{\mathrm{Sym}_{[k_0-1]\setminus i,l'} \cdot (z_i - z_{k_0}) z_i^{n_0-1} + \mathrm{Sym}_{[k_0-1],l'} \cdot z_{k_0} z_i^{n_0-1}}{\prod\limits_{j\in[k_0]\setminus i}(z_i - z_j)} + \frac{\mathrm{Sym}_{[k_0-1],l'} \cdot z_{k_0}^{n_0}}{\prod\limits_{j<k_0}(z_{k_0} - z_j)} \\
&= \sum_{i=1}^{k_0-1} \frac{\mathrm{Sym}_{[k_0-1]\setminus i,l'} \, z_i^{n_0-1}}{\prod\limits_{j\in[k_0-1]\setminus i}(z_i - z_j)} + \sum_{i=1}^{k_0-1} \frac{\mathrm{Sym}_{[k_0-1],l'} \cdot z_{k_0} z_i^{n_0-1}}{\prod\limits_{j\in[k_0]\setminus i}(z_i - z_j)} + \frac{\mathrm{Sym}_{[k_0-1],l'} \cdot z_{k_0}^{n_0}}{\prod\limits_{j<k_0}(z_{k_0} - z_j)} \\
&= r^{(l-1)}_{n_0-1,k_0-1} + \mathrm{Sym}_{[k_0-1],k_0-1-l} \cdot z_{k_0} \cdot r^{(k_0-1)}_{n_0-1,k_0}
\end{aligned}$$

By induction hypothesis, $|r^{(l)}_{n_0,k_0}| \le \binom{k_0-2}{l-1}\binom{n_0-1}{k_0-2} + \binom{k_0-1}{l}\binom{n_0-1}{k_0-1} \le \binom{k_0-1}{l}\binom{n_0}{k_0-1}$. □

Now we finish the proof of Lemma 12.5.4.

Lemma 12.5.4. For any $k \in \mathbb{Z}$ and any $z_1, \cdots, z_k$ on the unit circle of $\mathbb{C}$, there always exists a degree $m = O(k^2 \log k)$ polynomial $P(z) = \sum\limits_{j=0}^{m} c_j z^j$ with the following properties:

$$\begin{aligned}
&\text{Property I} && P(z_i) = 0, \forall i \in \{1, \cdots, k\}, \\
&\text{Property II} && c_0 = 1, \\
&\text{Property III} && |c_j| \le 11, \forall j \in \{1, \cdots, m\}.
\end{aligned}$$

Let $m = 10k^2 \log k$ and $\mathcal{P}$ denote a set of polynomials that has degree at most m, and all the coefficients are integers chosen from $\{-5, \cdots, -1, 0, 1, \cdots, 5\}$, i.e.,

$$\mathcal{P} := \left\{ P(z) = \sum_{i=0}^{m} \alpha_i z^i \mid \forall i \in \{0, 1, \cdots, m\}, |\alpha_i| \leq 2 \right\}.$$

Claim 12.10.4. *There exists $P^*(z) = \sum_{i=0}^{m} \alpha_i z^i$ with coefficient $|\alpha_i| \leq 10$ for any $i \in \{0, 1, \cdots, m\}$, such that every coefficient of $P^*(z) \mod Q(z)$ is bounded by 2^{-m}.*

Proof. For $P(z) = \sum_{i=0}^{m} \alpha_i z^i \in \mathcal{P}$, $P(z) \mod Q(z) \equiv \sum_{i=0}^{m} \alpha_i r_{n,k}(z)$ from the definition $r_{n,k}(z)$. Hence

$$P(z) \mod Q(z) = \sum_{i=0}^{m} \alpha_i \sum_{l=0}^{k-1} r_{i,k}^{(l)} z^l = \sum_{l=0}^{k-1} z^l \sum_{i=0}^{m} \alpha_i r_{i,k}^{(l)}.$$

Each coefficient in $P(z) \mod Q(z)$ is bounded by $|\sum_{i=0}^{m} \alpha_i r_{i,k}^{(l)}| \leq 5 \sum_{i=0}^{m} 2^k i^{k-1} \leq 2^k m^k$.

At the same time, $|\mathcal{P}| = 11^m$. From the pigeonhole principle and $\left(\frac{2^k m^k}{(2^{-m})^2}\right)^k < 11^m$, there exists $P_1, P_2 \in \mathcal{P}$ such that for $P^*(z) = P_1(z) - P_2(z)$, $P^*(z) \mod Q(z) = \sum_{i=0}^{k-1} \gamma_i z^i$ where each coefficient $|\gamma_i| \leq 2^{-m}$. □

Let $r(z) = \sum_{i=0}^{k-1} \gamma_i z^i = P^*(z) \mod Q(z)$ for convenience. If $P^*(0)$ (the constant term of P^*) is nonzero, then $|P^*(0) - r(0)| \geq 0.99$ from the above lemma. Therefore the polynomial $\frac{P^*(z) - r(z)}{P^*(0) - r(0)}$ satisfies the three properties in Lemma 12.5.4.

Otherwise, we assume z^l is the first term in $P^*(z)$ with a non-zero coefficient. Let

$$r_{-l,k}(z) = z^{-l} \mod Q(z) = \sum_{i=1}^{k} \frac{\prod_{j \in [k] \setminus i} (z - z_j) z_i^{-l}}{\prod_{j \in [k] \setminus i} (z_i - z_j)}.$$

For convenience, we use $z_S = \prod_{i \in S} z_i$ for any subset $S \subseteq [k]$. Notice that $z_i^{-l} = \frac{z_{[k]\setminus i}^l}{z_{[k]}^l}$. Hence $r_{-l,k}(z) = r'_{l,k}(z)/z_{[k]}^l$ where r' is the polynomial for k units roots $z_{[k]\setminus 1}, \cdots, z_{[k]\setminus k}$. So each coefficients of r is still bounded by $2^k l^k$, which is less than $2^{-m/2}$.

Eventually we choose $P^*(z)/z^l - r(z) \cdot r_{-l,k}(z)$ and renormalize it to satisfy the three properties in Lemma 12.5.4.

12.10.3 Proof of Lemma 12.4.3

Lemma 12.4.3. For any degree d polynomial $P(t) : \mathbb{R} \to \mathbb{C}$ with derivative $P'(t)$, we have,

$$\int_{-1}^{1} (1-t^2)|P'(t)|^2 \mathrm{d}t \le 2d^2 \int_{-1}^{1} |P(t)|^2 \mathrm{d}t. \tag{12.28}$$

Given a degree d polynomial $P(x)$, we rewrite $P(x)$ as a linear combination of the Legendre polynomials:

$$P(x) = \sum_{i=0}^{d} \alpha_i L_i(x).$$

We use $F_i(x) = (1-x^2)L_i'(x)$ for convenience. From the definition of the Legendre polynomials in the Equation (12.3), $F_i'(x) = -i(i+1) \cdot L_i(x)$ and $F_i''(x) = -i(i+1) \cdot L_i'(x)$.

Hence we have

$$\begin{aligned}
\int_{1}^{-1} (1-x^2)|P'(x)|^2 \mathrm{d}x = & \int_{1}^{-1} (1-x^2)P'(x) \cdot \overline{P'}(x) \mathrm{d}x \\
= & \int_{1}^{-1} \left(\sum_{i \in [d]} \alpha_i F_i(x) \right) \cdot \left(\sum_{i \in [d]} \overline{\alpha_i} \frac{-F_i''(x)}{i(i+1)} \right) \mathrm{d}x \\
= & \left(\sum_{i \in [d]} \alpha_i F_i(x) \right) \cdot \left(\sum_{i \in [d]} \overline{\alpha_i} \frac{-F_i'(x)}{i(i+1)} \right) \Bigg|_{-1}^{1} \\
+ & \int_{1}^{-1} \left(\sum_{i \in [d]} \alpha_i F_i'(x) \right) \cdot \left(\sum_{i \in [d]} \overline{\alpha_i} \frac{F_i'(x)}{i(i+1)} \right) \mathrm{d}x \\
= & \int_{1}^{-1} \left(\sum_{i \in [d]} \alpha_i \cdot i(i+1) \cdot L_i(x) \right) \cdot \left(\sum_{i \in [d]} \overline{\alpha_i} \frac{i(i+1) \cdot L_i(x)}{i(i+1)} \right) \mathrm{d}x \\
= & \sum_{i \in [d]} |\alpha_i|^2 i(i+1) \|L_i\|_T^2 \\
\le & d(d+1)\|P\|_T^2
\end{aligned}$$

12.10.4 Proof of Lemma 12.6.1

Lemma 12.6.1. $\widehat{P_{\sigma,a,b}x}(\sigma(f-b)) = \frac{1}{\sigma}e^{-2\pi\mathbf{i}\sigma af}\widehat{x}(f)$ and $\widehat{P_{\sigma,a,b}x}(f) = \frac{1}{\sigma}e^{-2\pi\mathbf{i}\sigma a(f/\sigma+b)}\widehat{x}(f/\sigma+b)$

Proof. Let's compute the Fourier Transform of $(P_{\sigma,a,b}x)(t)$,

$$
\begin{aligned}
&\widehat{P_{\sigma,a,b}x}(f) \\
=&\int_{-\infty}^{+\infty}(P_{\sigma,a,b}(x))(t)e^{-2\pi\mathbf{i}ft}\mathrm{d}t \\
=&\int_{-\infty}^{+\infty}x(\sigma(t-a))e^{-2\pi\mathbf{i}\sigma bt}e^{-2\pi\mathbf{i}ft}\mathrm{d}t \\
=&e^{-2\pi\mathbf{i}(\sigma ab+fa)}\int_{-\infty}^{+\infty}x(\sigma(t-a))e^{-2\pi\mathbf{i}\sigma b(t-a)}e^{-2\pi\mathbf{i}f(t-a)}\mathrm{d}t && \text{by shifting } t \text{ by } a \\
=&e^{-2\pi\mathbf{i}(\sigma ab+fa)}\int_{-\infty}^{+\infty}x(\sigma t)e^{-2\pi\mathbf{i}\sigma bt}e^{-2\pi\mathbf{i}ft}\mathrm{d}t && \text{by replacing } t-a \text{ by } t \\
=&\frac{1}{\sigma}e^{-2\pi\mathbf{i}(\sigma ab+fa)}\int_{-\infty}^{+\infty}x(\sigma t)e^{-2\pi\mathbf{i}b\sigma t}e^{-2\pi\mathbf{i}f\sigma t/\sigma}\mathrm{d}\sigma t \\
=&\frac{1}{\sigma}e^{-2\pi\mathbf{i}(\sigma ab+fa)}\int_{-\infty}^{+\infty}x(t)e^{-2\pi\mathbf{i}(b+f/\sigma)t}\mathrm{d}t && \text{by replacing } t\sigma \text{ by } t \\
=&\frac{1}{\sigma}e^{-2\pi\mathbf{i}a\sigma(f/\sigma+b)}\widehat{x}(f/\sigma+b) && \text{by definition of FT}
\end{aligned}
$$

The first result follows immediately by replacing $f/\sigma+b$ by f', which gives

$$\widehat{P_{\sigma,a,b}x}(\sigma(f'-b)) = \frac{1}{\sigma}e^{-2\pi\mathbf{i}a\sigma f'}\widehat{x}(f').$$

Thus, we complete the proof of this Lemma. □

12.10.5 Proof of Lemma 12.3.5

Lemma 12.3.5. Given any function $x(t) : \mathbb{R} \to \mathbb{C}$ with $\max_{t\in[0,T]} |x(t)|^2 \leq d\|x(t)\|_T^2$. Let S denote a set of points from 0 to T. If each point of S is chosen uniformly at random from $[0, T]$, we have

$$\Pr\left[\left|\frac{1}{|S|}\sum_{i\in S}|x(t_i)|^2 - \|x(t)\|_T^2]\right| \geq \epsilon\|x(t)\|_T^2\right] \quad \leq \quad e^{-\Omega(\epsilon^2|S|/d)}$$

Proof. Let M denote $\max_{t\in[0,T]} |x(t)|^2$. Replacing X_i by $\frac{|x(t_i)|^2}{M}$ and n by $|S|$ in Lemma A.1.1, we obtain that

$$\begin{aligned}
& \Pr[|X-\mu| > \epsilon\mu] \leq 2\exp(-\frac{\epsilon^2}{3}\mu) \\
\Longrightarrow \quad & \Pr\left[\left|\sum_{i\in S}\frac{|x(t_i)|^2}{M} - |S|\frac{\|x(t)\|_T^2}{M}\right| > \epsilon|S|\frac{\|x(t)\|_T^2}{M}\right] \leq 2\exp(-\frac{\epsilon^2}{3}\mu) \\
\Longrightarrow \quad & \Pr\left[\left|\frac{1}{|S|}\sum_{i\in S}|x(t_i)|^2 - \|x(t)\|_T^2\right| \geq \epsilon\|x(t)\|_T^2\right] \leq 2\exp(-\frac{\epsilon^2}{3}\mu) \\
\Longrightarrow \quad & \Pr\left[\left|\frac{1}{|S|}\sum_{i\in S}|x(t_i)|^2 - \|x(t)\|_T^2\right| \geq \epsilon\|x(t)\|_T^2\right] \leq 2\exp(-\frac{\epsilon^2}{3}|S|\frac{\|x(t)\|_T^2}{M})
\end{aligned}$$

which is less than $2\exp(-\frac{\epsilon^2}{3}|S|/d)$, thus completes the proof. □

12.10.6 Proof of Lemma 12.3.9

Lemma 12.3.9. For any polynomial $P(t)$ of degree at most d from R to $\mathbb{C}$, for any interval $[S,T]$,

$$\max_{t\in[S,T]} |P(t)|^2 \leq (d+1)^2 \cdot \frac{1}{T-S}\int_S^T |P(t)|^2 \mathrm{d}x.$$

Proof. Let $t^* = \arg\max_{t\in[S,T]} |P(t)|^2$. If $t^* \in (S,T)$, then it is enough to prove that

$$|P(t^*)|^2 \leq (d+1)^2 \frac{1}{t^*-S}\int_S^{t^*} |P(x)|^2 \mathrm{d}x \text{ and } |P(t^*)|^2 \leq (d+1)^2 \frac{1}{T-t^*}\int_{t^*}^T |P(x)|^2 \mathrm{d}x$$

on the two intervals $[S,t^*]$ and $[t^*,T]$ separately.

Without loss of generality, we will prove the inequality for $S=-1$ and $t^*=T=1$. We find the minimum $\|P(x)\|_T^2$ assuming $|P(1)|^2=1$. Because the first $(d+1)$ Legendre polynomials provide a basis of polynomials of degree at most d and their evaluation $L_n(1)=1$ for any n, we consider:

$$\begin{aligned} \min_{\alpha_0,\alpha_1,\cdots,\alpha_d\in\mathbb{C}} \quad & \int_{-1}^1 |P(x)|^2 \mathrm{d}x \\ \text{s.t.} \quad & P(x) = \sum_{i=0}^d \alpha_i L_i(x) \\ & |P(1)| = |\sum_{i=0}^d \alpha_i| = 1. \end{aligned}$$

We simplify the integration of $P(x)^2$ over $[-1,1]$ by the orthogonality of Legendre polyno-

mials:

$$\begin{aligned}\int_{-1}^{1}|P(x)|^2\mathrm{d}x &= \int_{-1}^{1}\left(\sum_{i=0}^{d}\alpha_i L_i(x)\right)\cdot\left(\sum_{j=0}^{d}\overline{\alpha_j}\overline{L_j}(x)\right)\mathrm{d}x \\ &= \int_{-1}^{1}\sum_{i=0}^{d}|\alpha_i|^2 L_i(x)^2+\sum_{i\neq j}\alpha_i\overline{\alpha_j}L_i(x)\overline{L_j}(x)\mathrm{d}x \\ &= \sum_{i=0}^{d}|\alpha_i|^2\frac{2}{2i+1} \text{ by Fact 12.3.8}\end{aligned}$$

Using $\int_{-1}^{1}|P(x)|^2\mathrm{d}x=\sum_{i=0}^{d}|\alpha_i|^2\frac{2}{2i+1}$, we simplify the optimization problem to

$$\begin{aligned}\min_{\alpha_0,\alpha_1,\cdots,\alpha_d\in\mathbb{C}} \quad & \sum_{i=0}^{d}|\alpha_i|^2\frac{2}{2i+1} \\ \text{s.t.} \quad & \left|\sum_{i=0}^{d}\alpha_i\right|=1\end{aligned}$$

From the Cauchy-Schwarz inequality, we have

$$\left|\sum_{i=0}^{d}\alpha_i\right|^2\leq\left(\sum_{i=0}^{d}|\alpha_i|^2\frac{2}{2i+1}\right)\left(\sum_{i=0}^{d}\frac{2i+1}{2}\right).$$

Therefore $\sum_{i=0}^{d}|\alpha_i|^2\frac{2}{2i+1}\geq\frac{2}{(d+1)^2}$ and $|P(1)|^2\leq(d+1)^2\cdot\frac{1}{2}\int_{-1}^{1}|P(x)|^2\mathrm{d}x$. □

Algorithm 12.1 Linear Regression Algorithms

procedure LINEARREGRESSION$(A, b,)$ — Fact 12.11.1
    $x' \leftarrow \arg\min_x \|Ax - b\|_2$.
    return x'
end procedure
procedure LINEARREGRESSIONW(A, b, w) — Fact 12.11.1
    $x' \leftarrow \arg\min_x \sum_{i=1}^{d} w_i |(Ax)_i - b_i|^2$.
    return x'
end procedure

12.11 Known Facts

This section provides a list of well-known facts existing in literature.

12.11.1 Linear regression

Given a linear subspace $\mathrm{span}\{\vec{v}_1, \cdots, \vec{v}_d\}$ and n points, we always use ℓ_2-regression to find a vector as the linear combination of $\vec{v}_1, \cdots, \vec{v}_d$ that minimizes the distance of this vector to those n points.

Fact 12.11.1. *Given an $n \times d$ matrix A and an $n \times 1$ column vector b , it takes $O(nd^{\omega-1})$ time to output an x' such that*

$$x' = \arg\min_x \|Ax - b\|_2.$$

where ω is the exponent of matrix multiplication[Wil12].

Notice that weighted linear regression can be solved by linear regression solver as a black-box.

Algorithm 12.2 Multipoint Evaluation of a Polynomial

1: **procedure** MULTIPOINTEVALUATION($P, \{t_1, t_2, \cdots, t_d\}$) — Fact 12.11.2
2: **return** $P(t_1), P(t_2), \cdots, P(t_d)$
3: **end procedure**

12.11.2 Multipoint evaluation of a polynomial

Given a degree-d polynomial, and n locations. The naive algorithm of computing the evaluations at those n locations takes $O(nd)$. However, the running time can be improved to $O(n \operatorname{poly}(\log d))$ by using this well-known result,

Fact 12.11.2 ([BS12]). *Let $c > 1$ denote some fixed constant. Given a degree-d polynomial $P(t)$, and a set of d locations $\{t_1, t_2, \cdots, t_d\}$. There exists an algorithm that takes $O(d \log^c d)$ time to output the evaluations*

$$\{P(t_1), P(t_2), \cdots, P(t_d)\}.$$

12.12 Analysis of Hash Functions and Filter Functions

12.12.1 Analysis of filter function $(H(t), \widehat{H}(f))$

We construct the Filter function $(H(t), \widehat{H}(f))$ in this section.

We fix the interval to be $\text{supp}(\text{rect}_1) = [-1/2, 1/2]$ instead of $[0, T]$ for convenience. We first define the filter function $H_1(t)$ which preserves the energy of a k-Fourier-sparse signal x^* on $[-1/2, 1/2]$ to the signal $H_1 \cdot x^*$ on $[-\infty, +\infty]$.

Definition 12.12.1. Let $s_1 = \Theta(k^4 \log^4 k)$, $\ell = \Omega(k \log k/\delta)$ be a even number, and $s_0 = C_0 s_1 \sqrt{\ell}$ for some constant C_0 that will normalize $H_1(0) = 1$. Recall that $\text{rect}_s(t) = 1$ iff $|t| \leq s/2$ and $\widehat{\text{rect}_s(f)} = \text{sinc}(fs) = \frac{\sin(\pi f s)}{\pi f s}$.

We define the filter function $H_1(t)$ and its Fourier transform $\widehat{H}_1(f)$ as follows:

$$\begin{aligned}
\widehat{H}_1(f) &= s_0 \cdot (\text{rect}_{s_1}(f))^{*\ell} \cdot \text{sinc}(f s_2), \\
&= s_0 \cdot \Big(\quad \Big)^{*\ell} \cdot \quad \text{(rect}_{s_1} \text{ with height 1, on } [-s_1/2, s_1/2]\text{)} \quad \text{(sinc}(f s_2) \text{ with height 1, on } [-1/s_2, 1/s_2]\text{)} \\
H_1(t) &= s_0 \cdot (\text{sinc}(s_1 t))^{\cdot \ell} * \text{rect}_{s_2}(t) \\
&= s_0 \cdot \Big(\quad \Big)^{\cdot \ell} * \quad \text{(sinc}(s_1 t) \text{ with height 1, on } [-1/s_1, 1/s_1]\text{)} \quad \text{(rect}_{s_2} \text{ with height 1, on } [-s_2/2, s_2/2]\text{)}
\end{aligned}$$

where s_0 is a fixed parameter s.t. $H_1(0) = 1$.

We provide some basic properties about our filter function. Notice that $\text{sinc}(t) = \frac{\sin(\pi t)}{\pi t}$ ($\text{sinc}(0)$ is defined to be 1) has the following properties (shown in Figure 12.4):

1. $\forall t \in \mathbb{R}, 1 - \frac{(\pi t)^2}{3!} \leq \text{sinc}(t) \leq 1$.

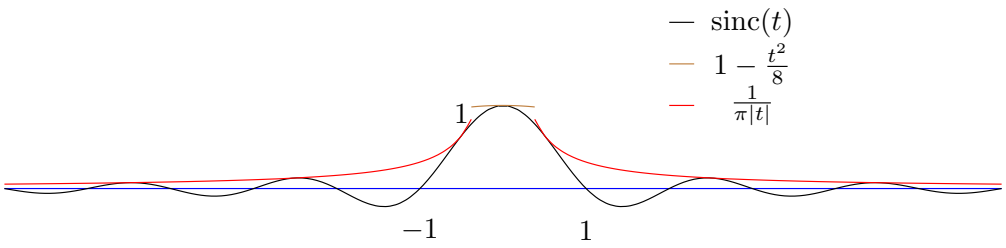


Figure 12.4: The Property of $\mathrm{sinc}(t)$.

2. $\forall |t| \leq 1.2/\pi, \mathrm{sinc}(t) \leq 1 - \frac{t^2}{8}$.

3. $\forall |t| > 1.2/\pi, |\mathrm{sinc}(t)| \leq \frac{1}{\pi|t|}$.

Claim 12.12.1. $\int_{-\frac{1.2}{\pi s_1}}^{\frac{1.2}{\pi s_1}} (\mathrm{sinc}(s_1 t))^\ell \mathrm{d}t \eqsim \frac{1}{s_1\sqrt{\ell}}$.

Proof. We use the above properties for the sinc function to prove the upper bound:

$$\begin{aligned}
\int_{-\frac{1.2}{\pi s_1}}^{+\frac{1.2}{\pi s_1}} (\mathrm{sinc}(s_1 t))^\ell \mathrm{d}t &= \frac{1}{s_1} \int_{-1.2/\pi}^{+1.2/\pi} (\mathrm{sinc}(t))^\ell \mathrm{d}t \\
&= \frac{2}{s_1} \left(\int_0^{\sqrt{8/\ell}} (\mathrm{sinc}\, t)^\ell \mathrm{d}t + \int_{\sqrt{8/\ell}}^{1.2/\pi} (\mathrm{sinc}(t))^\ell \mathrm{d}t \right) \\
&\leq \frac{2}{s_1} \left(\sqrt{8/\ell} + \sum_{i=1}^{\frac{1.2/\pi}{\sqrt{8/\ell}}-1} \int_{i\sqrt{8/\ell}}^{(i+1)\sqrt{8/\ell}} (1 - x^2/8)^\ell \mathrm{d}x \right) \\
&\leq \frac{2}{s_1} \left(\sqrt{8/\ell} + \sum_{i=1}^{\frac{1.2/\pi}{\sqrt{8/\ell}}-1} \sqrt{8/\ell} \cdot 2^{-i^2} \right) \\
&\lesssim \frac{1}{s_1\sqrt{\ell}}.
\end{aligned}$$

We prove the lower bound:

$$\begin{aligned}\int_{-\frac{1.2}{\pi s_1}}^{\frac{1.2}{\pi s_1}} (\mathrm{sinc}(s_1 t))^\ell \mathrm{d}t &= \frac{2}{s_1}\left(\int_0^{\sqrt{8/\ell}} (\mathrm{sinc}\, t)^\ell \mathrm{d}t + \int_{\sqrt{8/\ell}}^{1.2/\pi} (\mathrm{sinc}(t))^\ell \mathrm{d}t\right) \\ &\geq \frac{2}{s_1}\left(\int_0^{\sqrt{8/\ell}} (1-\frac{\pi^2 t^2}{6})^\ell \mathrm{d}t\right) \\ &\gtrsim \frac{1}{s_1\sqrt{\ell}}.\end{aligned}$$

□

We bound the integration outside $[-\frac{1.2}{\pi s_1}, \frac{1.2}{\pi s_1}]$ from the last property of the sinc function.

Claim 12.12.2. $\int_{\frac{1.2}{\pi s_1}}^{+\infty} (\mathrm{sinc}(s_1 t))^\ell \mathrm{d}t = O(1.2^{-\ell})$.

From these two claims, we have the existence of s_0.

Claim 12.12.3. *There exists a universal constant C_0 and $s_0 = C_0 s_1\sqrt{\ell}$ such that $H_1(0) = 1$.*

Proof. Because ℓ is a large even number, $\int_{\mathrm{rect}_{1-2/s_1}} \mathrm{sinc}(s_1 t)^\ell \mathrm{d}t \eqsim \frac{1}{s_1\sqrt{\ell}}$ from all discussion above. □

We show several useful properties about the Filter functions $\left(H_1(t), \widehat{H_1}(f)\right)$.

Lemma 12.12.4. *Given s_0, s_1, s_2, ℓ, where $\frac{s_2}{2} + \frac{1}{s_1} \leq 1/2$ and $s_0 = C_0 s_1\sqrt{\ell}$ for some constant*

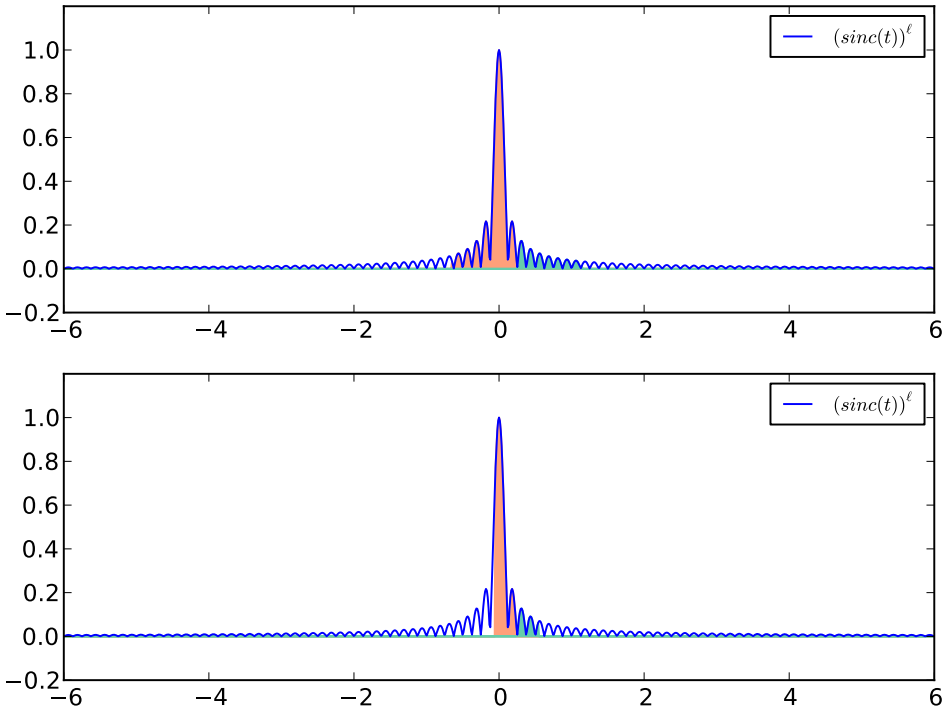


Figure 12.5: The light red area represents $\int_{(1/2-2/s_1)}^{(1/2-2/s_1)-t} s_0 \cdot \mathrm{sinc}\left(s_1(\tau)\right)^\ell \mathrm{d}\tau$ and the light green area represents $\int_{1/2-2/s_1}^{1/2-2/s_1+t} s_0 \cdot \mathrm{sinc}\left(s_1(\tau)\right)^\ell \mathrm{d}\tau$.

C_0. *The filter function* $(H_1(t), \widehat{H}_1(f))[s_0, s_1, s_2, \ell]$ *has the following properties,*

$$\begin{aligned}
\text{Property I}: &\quad H_1(t) \in [1 - \frac{s_0}{s_1} \cdot \frac{2\pi^{-\ell}}{\ell - 1}, 1], \ \textit{if } |t| \leq \frac{s_2}{2} - \frac{1}{s_1}. \\
\text{Property II}: &\quad H_1(t) \in [0, 1], \ \textit{if } \frac{s_2}{2} - \frac{1}{s_1} \leq |t| \leq \frac{1}{2} \\
\text{Property III}: &\quad H_1(t) \leq s_0 s_2 \left((s_1|t| - s_1 + 2)^2 + 1\right)^{-\ell}, \forall |t| > \frac{1}{2} \\
\text{Property IV}: &\quad \mathrm{supp}(\widehat{H}_1(f)) \subseteq [-\frac{s_1 \ell}{2}, \frac{s_1 \ell}{2}]
\end{aligned}$$

Proof of Property I. First, $H_1(0) = 1$ follows by definition of s_0, then we can prove the upper bound for $H_1(t)$ by showing for any $t > 0$, $H_1(0) - H_1(t) > 0$ always holds ,

By definition of sinc function, we know that $\text{sinc}(s_1 t)^\ell$ reaches 0 at all the points $\{\frac{1}{s_1} + i\frac{2}{s_1} | i \in \mathbb{N}\}$. By definition of s_1, we know that $\frac{1}{s_1} \ll \frac{1}{2} - \frac{1}{s_1}$. For any $t > 0$,

$$
\begin{aligned}
& H_1(0) - H_1(t) \\
= & \int s_0 \cdot \text{sinc}\left(s_1(\tau)\right)^\ell \cdot \text{rect}_{1-2/s_1}(0-\tau)\mathrm{d}\tau - \int s_0 \cdot \text{sinc}\left(s_1(\tau)\right)^\ell \cdot \text{rect}_{1-2/s_1}(t-\tau)\mathrm{d}\tau \\
= & \int_{-(1/2-2/s_1)}^{1/2-2/s_1} s_0 \cdot \text{sinc}\left(s_1(\tau)\right)^\ell \mathrm{d}\tau - \int_{-(1/2-2/s_1)+t}^{1/2-2/s_1+t} s_0 \cdot \text{sinc}\left(s_1(\tau)\right)^\ell \mathrm{d}\tau \\
= & \int_{-(1/2-2/s_1)}^{-(1/2-2/s_1)+t} s_0 \cdot \text{sinc}\left(s_1(\tau)\right)^\ell \mathrm{d}\tau - \int_{1/2-2/s_1}^{1/2-2/s_1+t} s_0 \cdot \text{sinc}\left(s_1(\tau)\right)^\ell \mathrm{d}\tau \\
\geq & 0,
\end{aligned}
$$

where the third step can be observed in Figure 12.5, and the last inequality follows by choosing s_1 to be an integer. Thus, we prove an upper bound for $H_1(t)$. Third, we show the lower bound for $H_1(t)$,

$$
\begin{aligned}
H_1(t) &= \int_{-\infty}^{+\infty} s_0 \cdot \text{sinc}(s_1\tau)^{\cdot\ell} \, \text{rect}_{s_2}(t-\tau)\mathrm{d}\tau \\
&= \int_{t-\frac{s_2}{2}}^{t+\frac{s_2}{2}} s_0 \cdot \text{sinc}(s_1\tau)^{\cdot\ell}\mathrm{d}\tau \\
&= 1 - \underbrace{\int_{t+\frac{s_2}{2}}^{+\infty} s_0 \cdot \text{sinc}(s_1\tau)^{\cdot\ell}\mathrm{d}\tau}_{A} - \underbrace{\int_{-\infty}^{t-\frac{s_2}{2}} s_0 \cdot \text{sinc}(s_1\tau)^{\cdot\ell}\mathrm{d}\tau}_{B}
\end{aligned}
$$

Thus, as long as we can upper bound the term A and B, then we will have a lower bound

for the $H_1(t)$, for any $|t| \leq \frac{s_2}{2} - \frac{1}{s_1}$.

$$\begin{aligned} A &= \int_{t+\frac{s_2}{2}}^{+\infty} s_0 \cdot \mathrm{sinc}(s_1\tau)^{\cdot\ell} \mathrm{d}\tau \\ &\leq \int_{\frac{1}{s_1}}^{+\infty} s_0 \cdot \mathrm{sinc}(s_1\tau)^{\cdot\ell} \mathrm{d}\tau \\ &\leq \int_{\frac{1}{s_1}}^{+\infty} s_0 \cdot (s_1\pi\tau)^{-\ell} \mathrm{d}\tau \\ &= s_0 \cdot (s_1\pi)^{-\ell} \frac{1}{\ell - 1} (1/s_1)^{-\ell+1} \\ &= \frac{s_0}{s_1} \cdot (\pi)^{-\ell} \frac{1}{\ell - 1} \end{aligned}$$

Similarly, we can bound the term B in the same way. □

Proof of Property II. In the proof of Property I, we already show that $\forall t$, $H_1(t) \leq 1$. Thus, the upper bound of Property II is also holding. The lower bound follows by both $\mathrm{sinc}(s_1 t)^{\cdot\ell}$ and $\mathrm{rect}_{s_2}(t)$ are always nonnegative, thus the convolution of these two functions has to be nonnegative. □

Proof of Property III. Let's prove the case when $t > 1$, since $H_1(t)$ is symmetric, then the case $t < -1$ will also hold. By definition of $(H_1(t), \widehat{H}_1(f))$, we have

$$\begin{aligned} H_1(t) &= s_0 \cdot \int_{-\infty}^{+\infty} \mathrm{sinc}(s_1(t-\tau))^{\cdot\ell} \, \mathrm{rect}_{s_2}(\tau) \mathrm{d}\tau \\ &= s_0 \cdot \int_{-\frac{s_2}{2}}^{\frac{s_2}{2}} \mathrm{sinc}(s_1(t-\tau))^{\cdot\ell} \mathrm{d}\tau \\ &= s_0 \cdot \int_{-\frac{s_2}{2}}^{\frac{s_2}{2}} \mathrm{sinc}(s_1(\tau-t))^{\cdot\ell} \mathrm{d}\tau \end{aligned}$$

We'd like to choose a middle point τ_0, and then separated the interval into two parts, one is $[-\frac{s_2}{2}, \tau_0]$ and the other is $[\tau_0, \frac{s_2}{2}]$. To choose a reasonable τ_0, we need to use the following simple facts,

$$\begin{aligned}(\frac{\sin(x)}{x})^\ell &\leq x^{-\ell} \text{ if } x \geq 1.2 \\ (\frac{\sin(x)}{x})^\ell &\leq (1 - \frac{x^2}{8})^\ell \text{ if } x < 1.2\end{aligned}$$

Thus, $|\pi s_1(\tau_0 - t)| = 1.2$, which implies that $\tau_0^+ = t + \frac{1.2}{\pi s_1}$ or $\tau_0^- = t - \frac{1.2}{\pi s_1}$. By relationship between s_1 and s_2, we know $\tau_0^- > \frac{1}{2} - \frac{1.2}{\pi s_1} > \frac{1}{2} - \frac{1}{s_1} \geq \frac{s_2}{2}$. Thus, we can use the case $x < 1.2$ to upper bound the $H_1(t)$,

$$\begin{aligned}& H_1(t) \\ \leq\ & s_0 \cdot s_2 \cdot \max_{\tau \in [-s_2/2, s_2/2]} \operatorname{sinc}(s_1(\tau - t))^\ell \\ \leq\ & s_0 \cdot s_2 \max_{\tau \in [-s_2/2, s_2/2]} (1 - \frac{(s_1\pi(\tau - t))^2}{8})^\ell \\ =\ & s_0 \cdot s_2 (1 - \frac{(s_1\pi(\frac{s_2}{2} - t))^2}{8})^\ell \\ \leq\ & s_0 \cdot s_2 \cdot (e^{-\frac{(s_1\pi(\frac{s_2}{2} - t))^2}{8}})^\ell && \text{by } 1 - x \leq e^{-x} \\ \leq\ & s_0 \cdot s_2 \cdot (e^{(s_1(t - s_2/2))^2})^{-\ell} && \text{by } 1 < \pi^2/8 \\ \leq\ & s_0 \cdot s_2 \cdot (1 + (s_1(t - s_2/2))^2)^{-\ell} && \text{by } 1 + x \leq e^x \\ \leq\ & s_0 \cdot (1 + (s_1(t - s_2/2))^2)^{-\ell} && \text{by } s_1 \leq 1.\end{aligned}$$

Thus, we complete the proof. □

Proof of Property IV. Because of the support of $\operatorname{rect}_{s_1}(f)$ is s_1, then the support of

$(\text{rect}_{s_1}(f))^{*\ell} = s_1\ell$. Since $\widehat{H}_1(f)$ is defined to be the $(\text{rect}_{s_1}(f))^{*\ell}$ multiplied by $\text{sinc}(fs_2)$, thus $\text{supp}(\widehat{H}_1(f)) \subseteq [-\frac{s_1\ell}{2}, \frac{s_1\ell}{2}]$. □

Definition 12.12.2. Given any $0 < s_3 < 1$, $0 < \delta < 1$, we define $(H(t), \widehat{H}(f))$ to be the filter function $(H_1(t), \widehat{H_1}(f))$ by doing the following operations

- Setting $\ell = \Theta(k \log(k/\delta))$,
- Setting $s_2 = 1 - \frac{2}{s_1}$,
- Shrinking by a factor s_3 in time domain,

$$H(t) = H_1(t/s_3) \tag{12.29}$$

$$\widehat{H}(f) = s_3\widehat{H_1}(s_3 f) \tag{12.30}$$

We call the "heavy cluster" around a frequency f_0 to be the support of $\delta_{f_0}(f) * \widehat{H}(f)$ in the frequency domain and use

$$\Delta_h = |\,\text{supp}(\widehat{H}(f))| = \frac{s_1 \cdot \ell}{s_3} \tag{12.31}$$

to denote the width of the cluster.

We show several useful properties about the Filter functions $\big(H(t), \widehat{H}(f)\big)$.

Lemma 12.6.4. Given $s_0, s_1, 0 < s_3 < 1, \ell > 1, 0 < \delta < 1$, where $\ell = \Theta(k \log(k/\delta))$.The filter

function $(H(t), \widehat{H}(f))$ has the following properties,

$$\begin{aligned}
\text{Property I}: &\quad H(t) \in [1-\delta, 1], \text{ when } |t| \leq (\frac{1}{2} - \frac{2}{s_1})s_3. \\
\text{Property II}: &\quad H(t) \in [0, 1], \text{ when } (\frac{1}{2} - \frac{2}{s_1})s_3 \leq |t| \leq \frac{1}{2}s_3. \\
\text{Property III}: &\quad H(t) \leq s_0 \cdot (s_1(\frac{|t|}{s_3} - \frac{1}{2}) + 2)^{-\ell}, \forall |t| > \frac{1}{2}s_3. \\
\text{Property IV}: &\quad \mathrm{supp}(\widehat{H}(f)) \subseteq [-\frac{s_1\ell}{2s_3}, \frac{s_1\ell}{2s_3}].
\end{aligned}$$

For any exact k-Fourier-sparse signal $x^*(t)$, we shift the interval from $[0, T]$ to $[-1/2, 1/2]$ and consider $x^*(t)$ for $t \in [-1/2, 1/2]$ to be our observation, which is also $x^*(t) \cdot \mathrm{rect}_1(t)$.

$$\begin{aligned}
\text{Property V}: &\quad \int_{-\infty}^{+\infty} \left|x^*(t) \cdot H(t) \cdot (1 - \mathrm{rect}_1(t))\right|^2 \mathrm{d}t < \delta \int_{-\infty}^{+\infty} |x^*(t) \cdot \mathrm{rect}_1(t)|^2 \mathrm{d}t. \\
\text{Property VI}: &\quad \int_{-\infty}^{+\infty} |x^*(t) \cdot H(t) \cdot \mathrm{rect}_1(t)|^2 \mathrm{d}t \in [1-\epsilon, 1] \cdot \int_{-\infty}^{+\infty} |x^*(t) \cdot \mathrm{rect}_1(t)|^2 \mathrm{d}t.
\end{aligned}$$

for arbitrarily small constant ϵ.

The Property I, II, III and IV follow by filter function $H(t), \widehat{H}(f)$ inheriting $H_1(t), \widehat{H}_1(f)$.

Proof of Property V. $\forall t \notin [-1/2, 1/2]$, we have,

$$\begin{aligned}
& |x^*(t) \cdot H(t)|^2 \\
\leq\ & |x^*(t)|^2 \cdot |H(t)|^2 \\
\leq\ & |x^*(t)|^2 \cdot (s_1(|t| - 1/2) + 2)^{-\ell} && \text{by Property III of } H_1(t) \\
\leq\ & k^7 \cdot (2kt)^{2.5k} \cdot \int_{-\infty}^{+\infty} |x^*(t) \cdot \mathrm{rect}_1(t)|^2 \mathrm{d}t \cdot (s_1(\frac{|t|}{s_3} - \frac{1}{2}) + 2)^{-\ell} && \text{by Lemma 12.5.5} \\
\leq\ & t^{O(k \log k)} \cdot \int_{-\infty}^{+\infty} |x^*(t) \cdot \mathrm{rect}_1(t)|^2 \mathrm{d}t \cdot (s_1(\frac{|t|}{s_3} - \frac{1}{2}) + 2)^{-\ell/2}. && (12.32)
\end{aligned}$$

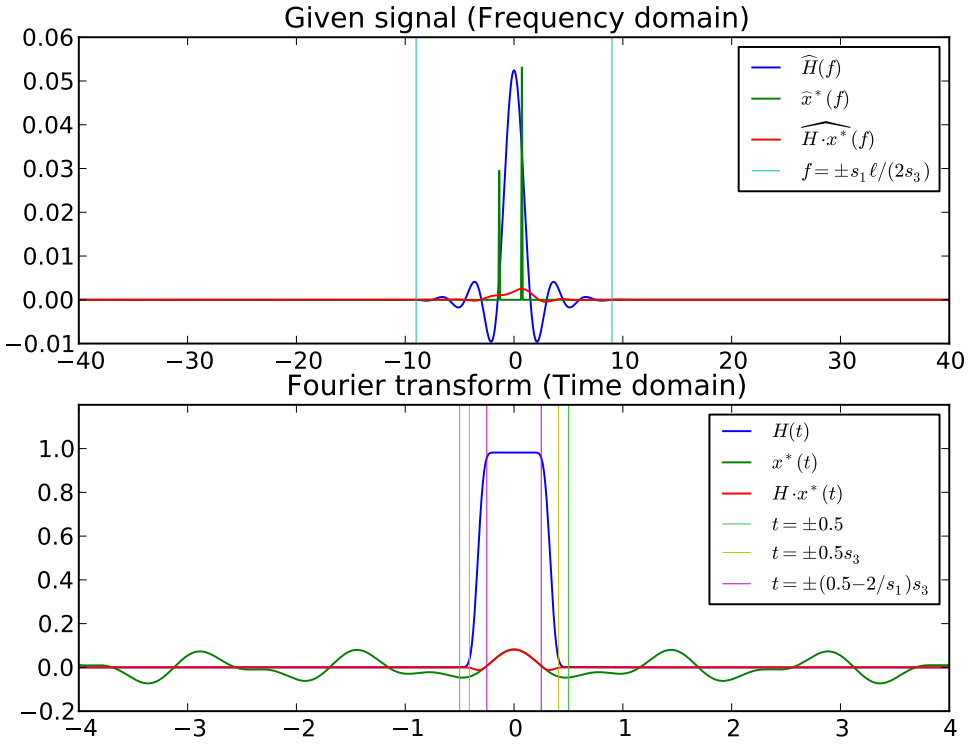


Figure 12.6: $\widehat{H \cdot x^*}(f)$ and $H \cdot x^*(t)$.

Thus taking the integral finishes the proof because $\ell \gtrsim k \log(k/\delta)$. □

Proof of Property VI. First, because of for any t, $|H_1(t)| \leq 1$, thus we prove the upper bound for LHS,

$$\int_{-\infty}^{+\infty} |x^*(t) \cdot H(t) \cdot \mathrm{rect}_1(t)|^2 \mathrm{d}t \leq \int_{-\infty}^{+\infty} |x^*(t) \cdot 1 \cdot \mathrm{rect}_1(t)|^2 \mathrm{d}t.$$

Second, as mentioned early, we need to prove the general case when $s_3 = 1 - 1/\operatorname{poly}(k)$. Define interval $S = [-s_3(\frac{1}{2} - \frac{1}{s_1}), s_3(\frac{1}{2} - \frac{1}{s_1})]$, by definition, $S \subset [-1/2, 1/2]$. Then define

$\overline{S} = [-1/2, 1/2] \setminus S$, which is $[-1/2, -s_3(\frac{1}{2} - \frac{1}{s_1})) \cup (s_3(\frac{1}{2} - \frac{1}{s_1}), 1/2]$. By Property I, we have

$$\int_S |x^*(t) \cdot H(t)|^2 \mathrm{d}t \geq (1-\delta)^2 \int_S |x^*(t)|^2 \mathrm{d}t \tag{12.33}$$

Then we can show

$$\begin{aligned}
&\int_{\overline{S}} |x^*(t)|^2 \mathrm{d}t \\
&\leq |\overline{S}| \cdot \max_{t \in [-1/2, 1/2]} |x^*(t)|^2 \\
&\leq (1 - s_3(1 - \frac{2}{s_1})) \cdot \widetilde{O}(k^4) \int_{-\frac{1}{2}}^{\frac{1}{2}} |x^*(t)|^2 \mathrm{d}t && \text{by Lemma 12.5.1} \\
&\lesssim \int_{-\frac{1}{2}}^{\frac{1}{2}} |x^*(t)|^2 \mathrm{d}t && \text{by } \min(\frac{1}{1-s_3}, s_1) \geq \widetilde{O}(k^4) \qquad (12.34)
\end{aligned}$$

Combining Equations (12.33) and (12.34) gives a lower bound for LHS,

$$\begin{aligned}
&\int_{-\infty}^{+\infty} |x^*(t) \cdot H(t) \cdot \mathrm{rect}_1(t)|^2 \mathrm{d}t \\
&\geq \int_S |x^*(t) H(t)|^2 \mathrm{d}t \\
&\geq (1-2\delta) \int_S |x^*(t)|^2 \mathrm{d}t && \text{by Equation (12.33)} \\
&\geq (1-2\delta) \int_{S \cup \overline{S}} |x^*(t)|^2 \mathrm{d}t - (1-2\delta) \int_{\overline{S}} |x^*(t)|^2 \mathrm{d}t \\
&\geq (1-2\delta) \int_{S \cup \overline{S}} |x^*(t)|^2 \mathrm{d}t - (1-2\delta)\epsilon \int_{S \cup \overline{S}} |x^*(t)|^2 \mathrm{d}t && \text{by Equation (12.34)} \\
&\geq (1-2\delta-\epsilon) \int_{-\frac{1}{2}}^{\frac{1}{2}} |x^*(t)|^2 \mathrm{d}t \\
&\geq (1-2\epsilon) \int_{-\infty}^{+\infty} |x^*(t) \cdot \mathrm{rect}_1(t)|^2 \mathrm{d}t && \text{by } \epsilon \gg \delta
\end{aligned}$$

□

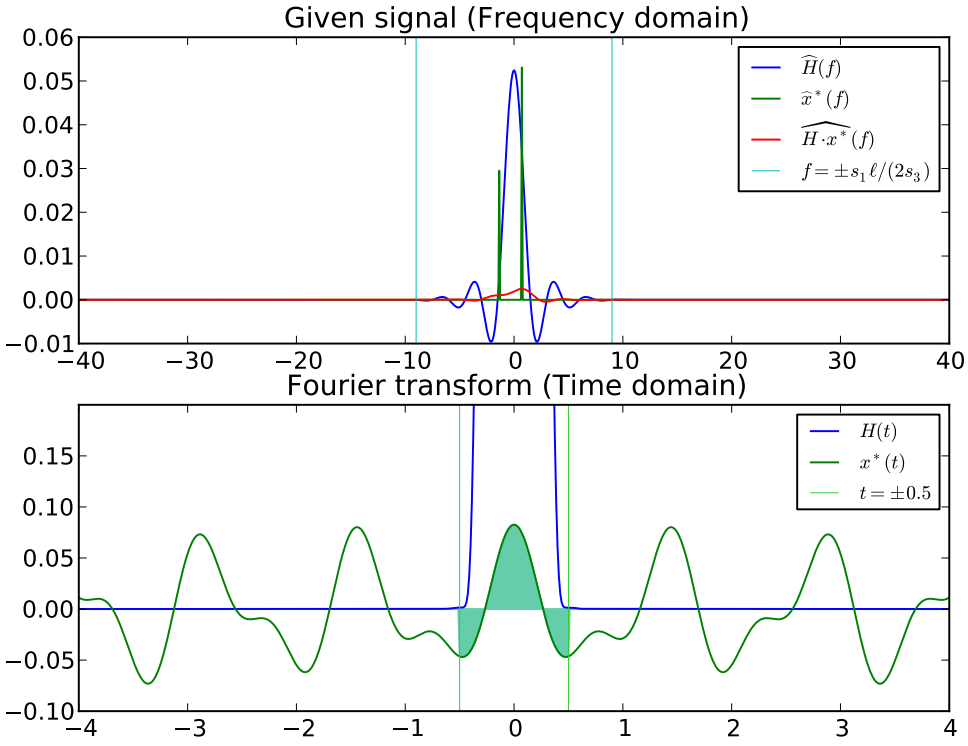


Figure 12.7: Property VI of filter function $H(t)$, first two figures of four figures. The light green area represents RHS(without scalar) of Property VI of filter H.

Remark 12.12.1. To match $(H(t), \widehat{H}(f))$ on $[-1/2, 1/2]$ with signal $x(t)$ on $[0, T]$, we will scale the time domain from $[-1/2, 1/2]$ to $[-T/2, T/2]$ and shift it to $[0, T]$. For example, the rectangle function in Property V and VI will be replaced by $\text{rect}_T(t - T/2)$. For the parameters $s_0, s_1, s_3, \delta, \ell$ in the definition of H, we always treat them as numbers. We assume T has seconds as unit and Δ_h has Hz as unit . For example, in time domain, the Property I becomes that given $T > 0$,

$$H(t) \in [1 - \delta, 1] \text{ if } |t - \frac{T}{2}| \leq (\frac{1}{2} - \frac{1}{s_1})s_3 \cdot T$$

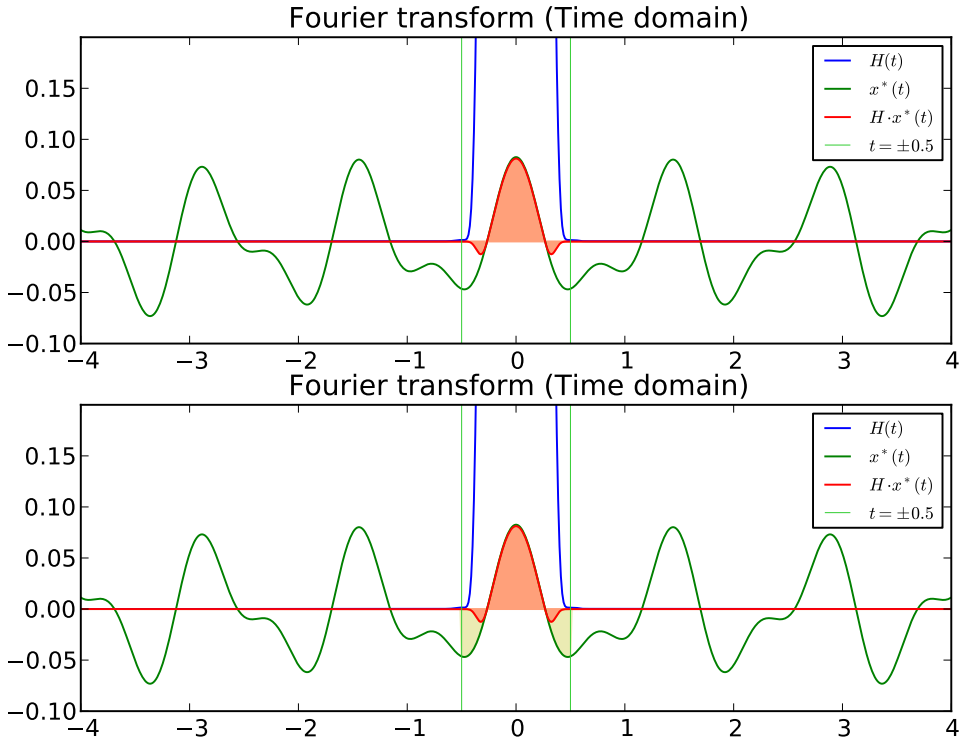


Figure 12.8: Property VI of filter function $H(t)$, last two figures of four figures. The light red area represents LHS of Property VI of filter H, the light yellow area represents the difference. Property VI says the light yellow area is only a small constant fraction of the light green area.

In frequency domain, the Property IV becomes

$$\mathrm{supp}(\widehat{H}(f)) \subseteq [-\frac{\Delta_h}{2}, \frac{\Delta_h}{2}], \text{ where } \Delta_h = \frac{s_1 \ell}{s_3 T}. \tag{12.35}$$

Lemma 12.12.5. *Let $H(t)$ denote the function defined in Definition 12.12.2. For any $t \in [-\frac{1}{2}, \frac{1}{2}]$, there exists an algorithm that takes $O(s_1 + \ell \log(s_1) + \log(1/\epsilon))$ time to output a*

value $\widetilde{H}(t)$ *such that*

$$(1-\epsilon)H(t) \leq \widetilde{H}(t) \leq (1+\epsilon)H(t).$$

Proof. We will show that using a low degree polynomial with sufficiently large degree is able to approximate the sinc function. By definition of filter function,

$$\begin{aligned}
H(t) &= s_0 \cdot \int_{-\infty}^{+\infty} \operatorname{sinc}(s_1\tau)^{\cdot \ell} \operatorname{rect}_{s_2}(t-\tau)\mathrm{d}\tau \\
&= s_0 \cdot \int_{t-\frac{s_2}{2}}^{t+\frac{s_2}{2}} (\frac{\sin(\pi s_1 \tau)}{\pi s_1 \tau})^{\ell}\mathrm{d}\tau \\
&= \frac{s_0}{\pi s_1} \int_{(t-\frac{s_2}{2})\pi s_1}^{(t+\frac{s_2}{2})\pi s_1} (\frac{\sin(\tau)}{\tau})^{\ell}\mathrm{d}\tau \\
&= \frac{s_0}{\pi s_1} \int_{(t-\frac{s_2}{2})\pi s_1}^{(t+\frac{s_2}{2})\pi s_1} \left(\sum_{i=0}^{\infty}(-1)^i \frac{\tau^{2i}}{(2i+1)!}\right)^{\ell} \mathrm{d}\tau \qquad \text{by Taylor expansion} \\
&= \frac{s_0}{\pi s_1} \int_{(t-\frac{s_2}{2})\pi s_1}^{(t+\frac{s_2}{2})\pi s_1} (A+B)^{\ell}\mathrm{d}\tau
\end{aligned}$$

where the last step follows by setting $A = \sum_{i=0}^{d}(-1)^i \frac{\tau^{2i}}{(2i+1)!}$, and $B = \sum_{i=d+1}^{\infty}(-1)^i \frac{\tau^{2i}}{(2i+1)!}$.

Denote $I^+ = (t+\frac{s_2}{2})\pi s_1$ and $I^- = (t-\frac{s_2}{2})\pi s_1$. Because of $t \in [-1/2, 1/2]$, then $\max(|I^+|, |I^-|) = O(s_1)$. The goal is to show that

$$(1-\epsilon)\int_{I^-}^{I^+}(A+B)^{\ell}\mathrm{d}\tau \leq \int_{I^-}^{I^+} A^{\ell}\mathrm{d}\tau \leq (1+\epsilon)\int_{I^-}^{I^+}(A+B)^{\ell}\mathrm{d}\tau$$

Let's prove an upper first,

$$
\begin{aligned}
&\int_{I^-}^{I^+} (A+B-B)^\ell \mathrm{d}\tau \\
=&\int_{I^-}^{I^+} (A+B)^\ell \mathrm{d}\tau + \sum_{j=1}^{\ell} \int_{I^-}^{I^+} \binom{\ell}{j} (A+B)^{\ell-j}(-B)^j \mathrm{d}\tau \\
\le&\int_{I^-}^{I^+} (A+B)^\ell \mathrm{d}\tau + \sum_{j=1}^{\ell} \int_{I^-}^{I^+} \binom{\ell}{j} |A+B|^{\ell-j}|B|^j \mathrm{d}\tau \\
\le&\int_{I^-}^{I^+} (A+B)^\ell \mathrm{d}\tau + \sum_{j=1}^{\ell} \int_{I^-}^{I^+} \binom{\ell}{j} |A+B|^{\ell-j}\mathrm{d}\tau \cdot \max_{\tau\in[I^-,I^+]} |B|^j \\
\le&\int_{I^-}^{I^+} (A+B)^\ell \mathrm{d}\tau + \ell 2^\ell \cdot \max_{\tau\in[I^-,I^+]} |B| && \text{by } |H(t)|\le 1 \text{ and } |B|^j \le |B| \\
\le&\int_{I^-}^{I^+} (A+B)^\ell \mathrm{d}\tau + \epsilon\cdot (s_1)^{-\Theta(\ell)} && \text{by Claim 12.12.6} \\
\le&(1+\epsilon)\int_{I^-}^{I^+} (A+B)^\ell \mathrm{d}\tau && \text{by Claim 12.12.7}
\end{aligned}
$$

where all the steps by setting $d \gtrsim s_1 + \ell\log(s_1) + \log(1/\epsilon)$. Similarly, we can prove a lower bound. □

Claim 12.12.6. *Let* $B(\tau) = \sum_{i=d+1}^{+\infty}(-1)\frac{\tau^{2i}}{(2i+1)!}$, *if* $d \gtrsim \tau + \ell\log(s_1) + \log(1/\epsilon)$ *then* $|B(\tau)| \le \epsilon(1/s_1)^{O(\ell)}$.

Proof. We first show, for any $i \ge d+1$,

$$
\begin{aligned}
&\frac{\tau^{2i}}{(2i+1)!} \\
\le&\frac{\tau^{2i}}{e((2i+1)/e)^{2i+1}} && \text{by } e(n/e)^n \le n! \\
\le&2^{-2i} && \text{by } i \gtrsim \tau \\
\le&\epsilon(1/s_1)^{O(\ell)} && \text{by } i \gtrsim \ell\log(s_1)+\log(1/\epsilon)
\end{aligned}
$$

Second, we can show that

$$\sum_{i=d+1}^{+\infty}(-1)\frac{\tau^{2i}}{(2i+1)!} \lesssim \frac{\tau^{2(d+1)}}{(2(d+1)+1)!} \leq \epsilon(1/s_1)^{O(\ell)}$$

Thus, we complete the proof. □

Claim 12.12.7. $\min_{t\in[-1/2,1/2]}|H(t)| \geq (s_1)^{-\Omega(\ell)}$.

Proof. By the property of $H(t)$,

$$\min_{\frac{1}{2}s_3<|t|\leq\frac{1}{2}} H(t) = \min_{|t|\leq\frac{1}{2}} H(t)$$

Thus, it suffices to prove a lower bound on $H(t)$ for any t such that $\frac{1}{2}s_3 < |t| \leq \frac{1}{2}$. Because of symmetric property, we only need to prove a lower bound for one side. Let's consider $t \in [\frac{1}{2}s_3, 1/2]$,

$$\begin{aligned}
H(t) &\geq \frac{s_0}{\pi s_1}\int_{(t-\frac{s_2}{2})\pi s_1}^{(t+\frac{s_2}{2})\pi s_1}(\frac{\sin(\tau)}{\tau})^{\ell}\mathrm{d}\tau \\
&\geq \frac{s_0}{\pi s_1}\int_{(t+\frac{s_2}{4})\pi s_1}^{(t+\frac{s_2}{2})\pi s_1}(\frac{\sin(\tau)}{\tau})^{\ell}\mathrm{d}\tau \\
&\geq \frac{s_0}{\pi s_1}\cdot\Theta((t+\frac{s_2}{2})s_1)\cdot\frac{1}{2}\cdot\pi\cdot\Theta((t+\frac{s_2}{2})\pi s_1)^{-\ell} \\
&\geq (s_1)^{-\Omega(\ell)}
\end{aligned}$$

□

12.12.2 Analysis of filter function $(G(t), \widehat{G}(f))$

We construct $(G(t), \widehat{G}(f))$ in a similar way of $(H_1(t), \widehat{H_1}(f))$ by switching the time domain and the frequency domain of $(H_1(t), \widehat{H_1}(f))$ and modify the parameters for the permutation hashing $P_{\sigma,a,b}$.

Definition 12.12.3. Given $B > 1$, $\delta > 0$, $\alpha > 0$, we construct $G(t), \widehat{G}(f)$ by doing the following operations,

- $s_2 = \frac{\pi}{2B}$,
- $s_1 = \frac{B}{\alpha\pi}$,
- $\ell = l = \Theta(\log(k/\delta))$.

Then $G(t), \widehat{G}(f)$ becomes

$$\begin{aligned} G(t) &= b_0 \cdot (\mathrm{rect}_{s_1}(t))^{*l} \cdot \mathrm{sinc}(ts_2) \\ &= b_0 \cdot (\mathrm{rect}_{\frac{B}{(\alpha\pi)}}(t))^{*l} \cdot \mathrm{sinc}(t\frac{\pi}{2B}), \\ \widehat{G}(f) &= b_0 \cdot (\mathrm{sinc}(s_1 f))^{\cdot l} * \mathrm{rect}_{s_2}(f) \\ &= b_0 \cdot (\mathrm{sinc}(\frac{B}{\alpha\pi} f))^{\cdot l} * \mathrm{rect}_{\frac{\pi}{2B}}(f). \end{aligned}$$

where the scalar $b_0 = \Theta(s_1\sqrt{l}) = \Theta(B\sqrt{l}/\alpha)$ satisfying $\widehat{G}(0) = 1$.

Lemma 12.6.5. Given $B > 1$, $\delta > 0$, $\alpha > 0$, we set $l = \Omega(\log(\delta/k))$. The filter function

$(G(t), \widehat{G}(f))[B, \delta, \alpha, l]$ satisfies the following properties,

$$\begin{aligned}
\text{Property I}: &\quad \widehat{G}(f) \in [1-\delta/k, 1], \text{ if } |f| \leq (1-\alpha)\frac{2\pi}{2B}. \\
\text{Property II}: &\quad \widehat{G}(f) \in [0, 1], \text{ if } (1-\alpha)\frac{2\pi}{2B} \leq |f| \leq \frac{2\pi}{2B}. \\
\text{Property III}: &\quad \widehat{G}(f) \in [-\delta/k, \delta/k], \text{ if } |f| > \frac{2\pi}{2B}. \\
\text{Property IV}: &\quad \mathrm{supp}(G(t)) \subset \Big[\frac{l}{2} \cdot \frac{-B}{\pi\alpha}, \frac{l}{2} \cdot \frac{B}{\pi\alpha}\Big]. \\
\text{Property V}: &\quad \max_t |G(t)| \lesssim \mathrm{poly}(B, l).
\end{aligned}$$

Proof. The first five Properties follows from Lemma 12.6.4 directly.

□

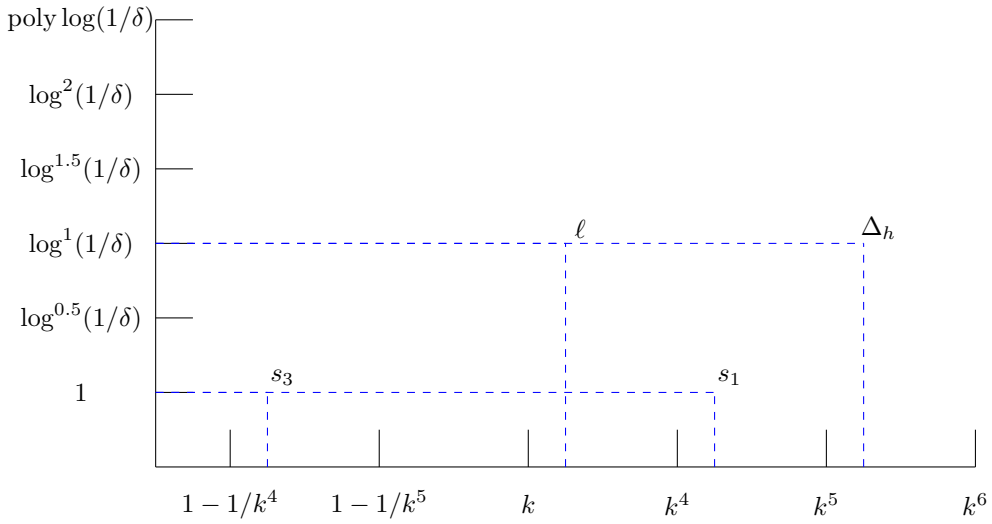


Figure 12.9: Parameters for s_1, s_3 and ℓ.

12.12.3 Parameters setting for filters

One-cluster Recovery. In one-cluster, we donot need filter function $(G(t), \widehat{G}(f))$.

In section 12.12.1, by Equation (12.34) in the proof of Property VI of filter function $(H(t), \widehat{H}(f))$, we need $\min(\frac{1}{1-s_3}, s_1) \geq \widetilde{O}(k^4)$.

In section 12.12.1, by Equation (12.32) in the proof of Property V of filter function $(H(t), \widehat{H}(f))$, we set $\ell \gtrsim k \log(k/\delta)$.

Δ_h is determined by the parameters of filter $(H(t), \widehat{H}(f))$ in Equation (12.35): $\Delta_h \approx \frac{s_1 \ell}{s_3 T}$ in section 12.12.1. Combining the setting of s_1, s_3 ℓ, we should set $\Delta_h \geq \widetilde{O}(k^5 \log(1/\delta))/T$.

k-cluster Recovery. Note that in the k-cluster recovery, we need to use filter function $(G(t), \widehat{G}(f))$. We choose $l = \log(k/\delta)$, $\alpha \approx 1$, $B \approx k$, and $D = l/\alpha$.

By proof of Property II of z in Lemma 12.7.18 from section 12.7.6, we need $T(1-s_3) >$

σBl. By the same reason in one-cluster recovery, $1-s_3 \leq \frac{1}{\widetilde{O}(k^4)}$. Combining $T(1-s_3) > \sigma Bl$ and $1-s_3 \leq \frac{1}{\widetilde{O}(k^4)}$, we obtain

$$\frac{T}{\widetilde{O}(k^4)} > \sigma Bl \tag{12.36}$$

Because in our algorithm, we will sample σ from $[\frac{1}{B\Delta_h}, \frac{2}{B\Delta_h}]$. Thus, plugging $\sigma = \Theta(\frac{1}{B\Delta_h})$ in Equation (12.36) we have

$$\frac{T}{\widetilde{O}(k^4)} > \frac{l}{\Delta_h}$$

which implies another lower bound for Δ_h,

$$\Delta_h \geq \widetilde{O}(k^4)l/T$$

Combining the above bound with previous lower bound in one-cluster recovery, we get

$$\Delta_h \geq \widetilde{O}(k^4 \log(1/\delta))/T + \widetilde{O}(k^5 \log(1/\delta))/T = \widetilde{O}(k^5 \log(1/\delta))/T$$

For s_1 and ℓ, we still choose the same setting as before, $s_1 \approx \widetilde{O}(k^4)$ and $\ell \approx O(k \log(k/\delta))$.

12.12.4 Analysis of HashToBins

In this section, we explain the correctness of Procedure HashToBins in Algorithm 12.6. Before giving the proof of that algorithm, we show how to connect CFT, DTFT and DFT.

Lemma 12.12.8. *For any signal $W : \mathbb{R} \to \mathbb{C}$, let $A : \mathbb{Z} \to \mathbb{C}$ and $B : [n] \to \mathbb{C}$ be defined as follows:*

$$A[i] = W(i), \forall i \in \mathbb{Z} \text{ and } B[i] = \sum_{j \in \mathbb{Z}} A[i + jn], \forall i \in [n].$$

Then we consider the Fourier transform on W, A, and B:

$$\begin{aligned} \text{CFT} \quad & \widehat{W} : \mathbb{R} \to \mathbb{C}, \\ \text{DTFT} \quad & \widehat{A} : [0, 1] \to \mathbb{C}, \\ \text{DFT} \quad & \widehat{B} : [n] \to \mathbb{C}. \end{aligned}$$

We have:

$$\forall f \in [0, 1), \widehat{A}(f) = \sum_{j \in \mathbb{Z}} \widehat{W}(f + j); \quad \forall i \in [n], \widehat{B}[i] = \sum_{j \in \mathbb{Z}} \widehat{W}(i/n + j).$$

Proof. Recall that $\mathrm{Comb}_s(t) = \sum_{j \in \mathbb{Z}} \delta_{js}(t)$. First, we show $\widehat{A}(f) = \sum_{j \in \mathbb{Z}} e^{2\pi \mathbf{i} j f} A[j]$ equals

to $\sum_{j\in\mathbb{Z}}\widehat{W}(f+j)$:

$$\begin{aligned}
\widehat{A}(f) = & \sum_{j\in\mathbb{Z}} e^{2\pi\mathbf{i}jf}W[j] && \text{by } A[j]=W(j)\\
= & \int_{-\infty}^{+\infty} e^{2\pi\mathbf{i}fj}W(j)\cdot \mathrm{Comb}_1(j)\mathrm{d}j\\
= & \widehat{W\cdot \mathrm{Comb}_1}(f)\\
= & (\widehat{W} * \widehat{\mathrm{Comb}_1})(f)\\
= & \sum_{j\in\mathbb{Z}} \widehat{W}(f+j). && (12.37)
\end{aligned}$$

Next, we prove that $\forall i\in[n], \widehat{B}[i]=\widehat{A}(i/n)$,

$$\begin{aligned}
\widehat{B}[i] = & \sum_{j=1}^{n} B[j]e^{\frac{2\pi\mathbf{i}}{n}ij} && \text{by DFT}\\
= & \sum_{j=1}^{n}(\sum_{k\in\mathbb{Z}} A[j+kn])e^{\frac{2\pi\mathbf{i}}{n}ij} && \text{by } B[j]=\sum_{k\in\mathbb{Z}}A[j+kn]\\
= & \sum_{j=1}^{n}\sum_{k\in\mathbb{Z}} A[j+kn]e^{\frac{2\pi\mathbf{i}}{n}i(j+kn)} && \text{by } e^{\frac{2\pi\mathbf{i}}{n}\cdot ikn}=1\\
= & \sum_{j\in\mathbb{Z}} A[j]e^{2\pi\mathbf{i}j\frac{i}{n}} = \widehat{A}(i/n) && \text{by DTFT.} \qquad (12.38)
\end{aligned}$$

Combining Equation (12.38) and Equation (12.37), we obtain that $\widehat{B}[j]=\widehat{A}(j/n)=\sum_{i\in\mathbb{Z}}\widehat{W}(j/n+i)$ for all $j\in[n]$. □

Claim 12.12.9. *Let $u\in\mathbb{C}^B$ and $V\in\mathbb{C}^{BD}$ such that for any $j\in B$, $u[j]=\sum_{i\in[D]}V[j+(i-1)B]$. Then*

$$\widehat{u}[j]=\widehat{V}[jD], \forall j\in[B].$$

Proof. We prove it through the definition of the Fourier transform:

$$\begin{aligned}
\widehat{V}[jD] = & \sum_{i=1}^{BD} V[i] \cdot e^{\frac{2\pi\mathbf{i}}{BD}\cdot i\cdot(jD)} && \text{by definition of DFT} \\
= & \sum_{i=1}^{B}\sum_{k=1}^{D} V[i+kB] e^{\frac{2\pi\mathbf{i}}{B}\cdot(i+kB)\cdot j} && \text{by replacing } i \text{ by } i+kB \\
= & \sum_{i=1}^{B} e^{\frac{2\pi\mathbf{i}}{B}\cdot j\cdot i}\sum_{k=1}^{D} V[i+(k-1)B] && \text{by } e^{2\pi\mathbf{i}jk}=1 \\
= & \sum_{i=1}^{B} e^{\frac{2\pi\mathbf{i}}{B}\cdot j\cdot i} u[i] = \widehat{u}[j] && \text{by definition of DFT on } u
\end{aligned}$$

□

We use Definition 12.6.1 and Lemma 12.6.1 to generalize Lemma 12.12.8,

Corollary 12.12.10. *If for all $j \in [n]$, $B[j] = \sum_{i\in\mathbb{Z}} W\big((j+in)\sigma - \sigma a\big)$, then $\forall j \in [n]$,*

$$\widehat{B}[j] = \sum_{i\in\mathbb{Z}} \widehat{W}\left((\frac{j}{n}+i)/\sigma\right)\cdot\frac{1}{\sigma}e^{-2\pi\mathbf{i}(\frac{j}{n}+i)a}.$$

If for all $j \in [n]$, $B[j] = \sum_{i\in\mathbb{Z}} W\big((j+in)\sigma - \sigma a\big)e^{-2\pi\mathbf{i}\sigma b(j+in)}$, then $\forall j \in [n]$,

$$\widehat{B}[j] = \sum_{i\in\mathbb{Z}} \widehat{W}\left((\frac{j}{n}+i)/\sigma + b\right)\cdot\frac{1}{\sigma}e^{-2\pi\mathbf{i}(\frac{j}{n}+i)a - 2\pi\mathbf{i}\sigma ab}.$$

Remark 12.12.2 (Samples of HashToBins). Procedure HashToBins in Algorithm 12.6 takes BD samples in $x(t)$:

$$x(\sigma(1-a)), x(\sigma(2-a)), \cdots, x(\sigma(BD-a)).$$

To analyze our algorithm, we use filter function $(G(t), \widehat{G}(f))$ and $\mathrm{Comb}_s(t) = \sum_{j\in\mathbb{Z}}\delta_{sj}(t)$ to define the discretization of G.

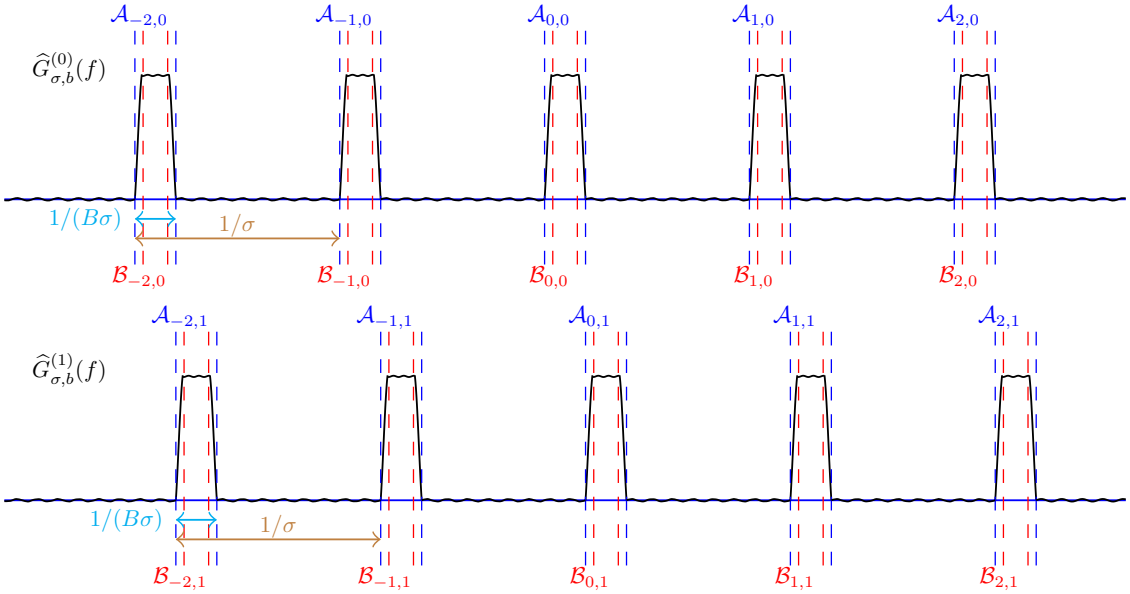
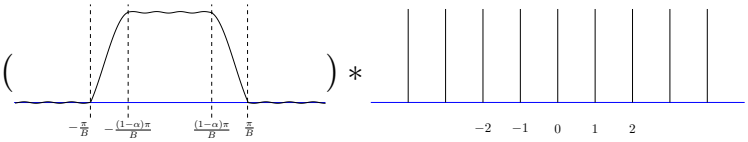


Figure 12.10: $\widehat{G}^{(j)}_{\sigma,b}(f)$ where the top one is $j = 0$ and the bottom one is $j = 1$, $\mathcal{A}_{i,j} = [\frac{1}{\sigma}(2\pi(i+\frac{j}{B}) - \frac{2\pi}{2B}), \frac{1}{\sigma}(2\pi(i+\frac{j}{B}) + \frac{2\pi}{2B})]$, $\mathcal{B}_{i,j} = [\frac{1}{\sigma}(2\pi(i+\frac{j}{B}) - \frac{2\pi(1-\alpha)}{2B}), \frac{1}{\sigma}(2\pi(i+\frac{j}{B}) + \frac{2\pi(1-\alpha)}{2B})]$

Definition 12.12.4. Define the discretization of $G(t)$ and $\widehat{G}(f)$,

$$
\begin{aligned}
G^{\text{dis}}(t) &= G(t) \cdot \text{Comb}_s(t) \\
\widehat{G}^{\text{dis}}(f) &= \frac{1}{s}(\widehat{G} * \text{Comb}_{1/s})(f) \\
&= (\widehat{G} * \text{Comb}_1)(f) \\
&= \Big(\quad \Big) * \quad
\end{aligned}
$$



where $|\text{supp}(G(t))| = \frac{lB}{\pi\alpha}$, $D = \frac{l}{\pi\alpha}$, $s = |\text{supp}(G(t))|/(BD) = l/(\pi\alpha D) = 1$.

Definition 12.6.6. $\forall \sigma > 0, b$ and $j \in [B]$. Define,

$$\begin{aligned} G^{(j)}_{\sigma,b}(t) &= \frac{1}{\sigma} G(t/\sigma) e^{2\pi \mathbf{i} t(j/B - \sigma b)/\sigma} \\ \widehat{G}^{(j)}_{\sigma,b}(f) &= \widehat{G}^{\mathrm{dis}}(\frac{j}{B} - \sigma f - \sigma b) = \sum_{i \in \mathbb{Z}} \widehat{G}(i + \frac{j}{B} - \sigma f - \sigma b) \end{aligned}$$

Lemma 12.6.7. Let $u \in \mathbb{C}^B$ be the result of HashToBins under permutation $P_{\sigma,a,b}$, and let $j \in [B]$. Define

$$\widehat{z} = \widehat{x \cdot H} \cdot \widehat{G}^{(j)}_{\sigma,b},$$

so

$$z = (x \cdot H) * G^{(j)}_{\sigma,b}.$$

Let vector $\widehat{u} \in \mathbb{C}^B$ denote the B-dimensional DFT of u, then $\forall j \in [B]$,

$$\widehat{u}[j] = z_{\sigma a}.$$

Proof. Recall B is the number of hash bins. $B \cdot D$ is the number of samples in time signal. Let $W(t) = x \cdot H(t)$, define vector $y \in \mathbb{C}^{BD}$, then $\forall j \in [BD]$, define

$$y[j] = W(\sigma(j - a)) e^{2\pi \mathbf{i} \sigma b j}$$

Recall $G(t)$ denote the $\mathrm{rect}^{*l}_{B/\alpha}(t) \cdot \mathrm{sinc}(t/B)$, then $|\mathrm{supp}(G(t))| = \frac{lB}{\alpha}$. Let vector $G' \in \mathbb{C}^{BD}$ is the discretization of $G(t)$, where $G'[i] = G(i)$. Then, $\forall j \in [B]$,

$$u[j] = \sum_{i \in [D]} V[j + iB]$$

where $V[j] = y[j] \cdot G'[j]$ and $G'[j]$ is the value at the jth nonzero point of $G^{\mathrm{dis}}(t)$. Applying Claim 12.12.9 with the definition of $u[j]$ and $V[j + iB]$, gives $\widehat{u}[j] = \widehat{V}[jD], \forall j \in [B]$.

Because of u is the result of HashToBins($x \cdot H, P_{\sigma,a,b}, G$) and $|\operatorname{supp}(G(t))| = BD$(choosing $D = l/\alpha$), then

$$u[j] = \sum_{i \in \mathbb{Z}} W(\sigma(j + iB - a))e^{-2\pi \mathbf{i}\sigma b(j+iB)}G(j + iB)$$

Then we define $G''(t) = G(t/\sigma + a)e^{-2\pi \mathbf{i}b\sigma(t/\sigma+a)}$ and $Y(t) = W(t) \cdot G''(t)$, then immediately, we have

$$\widehat{G''}(f) = \sigma\widehat{G}(\sigma(f - b))e^{2\pi \mathbf{i}a\sigma f} \quad \text{and} \quad \widehat{Y}(f) = \widehat{W}(f) * \widehat{G''}(f)$$

Thus, we can rewrite $u[j]$ in the following sense,

$$\begin{aligned}
& u[j] \\
= & \sum_{i \in \mathbb{Z}} W(\sigma(j + iB - a))e^{-2\pi \mathbf{i}\sigma b(j+iB)}G(j + iB) \\
= & \sum_{i \in \mathbb{Z}} W(\sigma(j + iB - a))G''(\sigma(j + iB - a)) && \text{by } G''(t) = G(t/\sigma + a)e^{-2\pi \mathbf{i}b\sigma(t/\sigma+a)} \\
= & \sum_{i \in \mathbb{Z}} Y(\sigma(j + iB - a)) && \text{by } Y(t) = W(t) \cdot G''(t)
\end{aligned}$$

Then

$$\begin{aligned}
&\widehat{u}[j] \\
=\ & \sum_{i\in\mathbb{Z}} \widehat{Y}((\frac{j}{B}+i)/\sigma)\cdot\frac{1}{\sigma}\cdot e^{-2\pi\mathbf{i}(\frac{j}{B}+i)a} && \text{by Corollary 12.12.10} \\
=\ & \sum_{i\in\mathbb{Z}} \int_{-\infty}^{+\infty} \widehat{W}(s)\cdot\widehat{G}''(\frac{j/B+i}{\sigma}-s)\cdot\frac{1}{\sigma}\cdot e^{-2\pi\mathbf{i}\cdot(j/B+i)a}\mathrm{d}s && \text{by } \widehat{Y}(f)=\widehat{W}(f)*\widehat{G}''(f) \\
=\ & \sum_{i\in\mathbb{Z}} \int_{-\infty}^{+\infty} \widehat{W}(s)\cdot\widehat{G}(j/B+i-\sigma s-\sigma b)\cdot e^{-2\pi\mathbf{i}\cdot(-\sigma s)a}\mathrm{d}s && \text{by } \widehat{G}''(f)=\sigma\widehat{G}(\sigma(f-b))e^{2\pi\mathbf{i}a\sigma f} \\
=\ & \int_{-\infty}^{+\infty} \widehat{W}(s)\cdot\sum_{i\in\mathbb{Z}}\widehat{G}((j/B+i)-\sigma s-\sigma b)e^{2\pi\mathbf{i}a\sigma s}\mathrm{d}s \\
=\ & \int_{-\infty}^{+\infty} \widehat{W}(s)\cdot\widehat{G}^{\mathrm{dis}}(\frac{j}{B}-\sigma s-\sigma b)e^{-2\pi\mathbf{i}a\sigma s}\mathrm{d}s && \text{by } \widehat{G}^{\mathrm{dis}}(f)=\sum_{i\in\mathbb{Z}}\widehat{G}(f+i)
\end{aligned}$$

By definition 12.12.4,

$$\widehat{G}^{(j)}_{\sigma,b}=\widehat{G}^{\mathrm{dis}}(\frac{j}{B}-\sigma s-\sigma b)=\sum_{i\in\mathbb{Z}}\widehat{G}(i+\frac{j}{B}-\sigma s-\sigma b)$$

By definition of $\widehat{z}$, we have

$$\widehat{z}(s)=\widehat{x\cdot H}(s)\cdot\widehat{G}^{(j)}(s)=\widehat{W}(s)\cdot\widehat{G}^{(j)}(s)$$

Then $\widehat{u}[j]$ is the $(a\sigma)^{th}$ inverse Fourier coefficients of $\widehat{z}$, basically,

$$\widehat{u}[j]=z_{a\sigma}=z(a\sigma)$$

Thus, we can conclude first computing vector $u\in\mathbb{C}^B$. Getting vector $\widehat{u}\in\mathbb{C}^B$ by using the Discrete Fourier transform $\widehat{u}=\mathrm{DFT}(u)$. This procedure allows us to sample from time domain to implicitly access the time signal's Fourier transform $\widehat{z}$. If z is one-cluster in frequency domain, then apply one-cluster recovery algorithm.

□

12.13 Algorithm

This section lists the pseudocode of our algorithms.

Algorithm 12.3 Get Empirical One Energy and Get Legal One Sample

procedure GetEmpirical1Energy(z, T, Δ) — Claim 12.7.10
  $R_{\text{est}} \leftarrow (T\Delta)^2$
  for $i = 1 \to R_{\text{est}}$ **do**
    Choose $\alpha_i \in [0, T]$ uniformly at random
    $z_{\text{emp}} \leftarrow z_{\text{emp}} + |z(\alpha_i)|^2$
  end for
  $z_{\text{emp}} \leftarrow \sqrt{z_{\text{emp}}/R_{\text{est}}}$
  return z_{emp}
end procedure
procedure GetLegal1Sample($z, \Delta, T, \beta, z_{\text{emp}}$) — Lemma 12.7.1
  $R_{\text{repeat}} \leftarrow (T\Delta)^3, S_{\text{heavy}} \leftarrow \emptyset$
  for $i = 1 \to R_{\text{repeat}}$ **do**
    Choose $\alpha_i \in [0, T]$ uniformly at random
    if $|z(\alpha_i)| \geq 0.5 \cdot z_{\text{emp}}$ **then**
      $S_{\text{heavy}} \leftarrow S_{\text{heavy}} \cup i$
    end if
  end for
  for $i \in S_{\text{heavy}}$ **do**
    $w(i) \leftarrow |z(\alpha_i)|^2 + |z(\alpha_i + \beta)|^2$
  end for
  $\alpha \leftarrow \alpha_i$ with probability $w(i) / \sum_{j \in S_{\text{heavy}}} w(j)$ for $i \in S_{\text{heavy}}$
  return α
end procedure

Algorithm 12.4 Locate One Signal, Locate One Inner, and Frequency Recovery One Cluster

procedure LOCATE1SIGNAL($z, T, F, \Delta, z_{\text{emp}}$) — Lemma 12.7.14
  Set $t \approx \log(FT)$, $t' = t/4$, $D_{\max} \approx \log_{t'}(FT)$, $R_{\text{loc}} \approx \log_{1/c}(tc)$, $L^{(1)} = 2F$
  for $i \in [D_{\max}]$ **do**
    $l \approx 2F/(t')^{i-1}\Delta$, $s \approx c$, $\widehat{\beta} = \frac{ts}{2\Delta l}$
    if $\widehat{\beta} \gtrsim T/(T\Delta)^{3/2}$ **then**
      break
    else
      $L^{(i)} \leftarrow$ LOCATE1INNER($z, \Delta, T, \widehat{\beta}, z_{\text{emp}}, L^{(i-1)}$)
    end if
  end for
  return $L^{(i)}$
end procedure
procedure LOCATE1INNER($z, \Delta, T, \widehat{\beta}, z_{\text{emp}}, \widetilde{L}$)
  Let $v_q \leftarrow 0$ for $q \in [t]$
  while $r = 1 \to R_{\text{loc}}$ **do**
    Choose $\beta \in [\frac{1}{2}\widehat{\beta}, \widehat{\beta}]$ uniformly at random
    $\gamma \leftarrow$ GETLEGAL1SAMPLE($z, \Delta, T, \beta, z_{\text{emp}}$)
    for $i \in [m]$ **do**
      $s_i \in [\beta(\widetilde{L} - \Delta l/2), \beta(\widetilde{L} + \Delta l/2)] \cap \mathbb{Z}_+, \theta_i = \frac{1}{2\pi\sigma\beta}(\phi(x(\gamma)/x(\gamma+\beta)) + 2\pi s_i)$
      Let θ_i belong to region(q)
      Then add a vote to region(q) and its two neighbors, i.e., region($q-1$) and region($q+1$)
    end for
  end while
  $q_j^* \leftarrow \{q | v_q > \frac{R_{\text{loc}}}{2}\}$
  return $L \leftarrow$ center of region(q_j^*)
end procedure
procedure FREQUENCYRECOVERY1CLUSTER(z, T, F, Δ) — Theorem 12.7.4
  $z_{\text{emp}} \leftarrow$ GETEMPIRICAL1ENERGY(z, T, Δ)
  for $r = 1 \to O(k)$ **do**
    $L_r \leftarrow$ LOCATE1SIGNAL($z, T, F, \Delta, z_{\text{emp}}$)
  end for
  return $L^* \leftarrow \underset{r \in [O(k)]}{\text{median}}\ L_r$
end procedure

Algorithm 12.5 Main Algorithm for One-cluster Recovery

procedure CFT1Culster(x, H, T, F) — Theorem 12.8.1
  $\widetilde{f}_0 \leftarrow$ FrequencyRecovery1Cluster(x, H, T, F)
  $\widetilde{x} \leftarrow$ SignalRecovery1Cluster($\widetilde{f}_0, \text{poly}(k)\Delta_h$)
  return $\widetilde{x}$
end procedure
procedure GenerateIntervals(d)
  $n \leftarrow y_0 \leftarrow i \leftarrow 0$, $m \leftarrow \Theta(d)$
  while $y_i \leq 1 - \frac{9}{m^2}$ **do**
    $y_{i+1} \leftarrow y_i + \frac{\sqrt{1-y_i^2}}{m}$, $I_{n+1} \leftarrow [y_i, y_{i+1}]$, $I_{n+2} \leftarrow [-y_{i+1}, -y_i]$
    $i \leftarrow i+1, n \leftarrow n+2$
  end while
  $I_{n+1} \leftarrow [y_i, 1], I_{n+2} \leftarrow [-y_i, -1]$, $n \leftarrow n+2$
  return n, I
end procedure
procedure RobustPolynomialLearning(x, d, T) — Theorem 12.1.2
  $(n, I) \leftarrow$ GenerateIntervals(d)
  for $j = 1 \to n$ **do**
    $w_j \leftarrow |I_j|/2$
    Choose t_j from I_j uniformly at random
    $z_j \leftarrow x(T \cdot \frac{t_j+1}{2})$
  end for
  $\widetilde{A}_{j,i} \leftarrow t_j^i$, for each $(j,i) \in [n] \times \{0, 1, \cdots, d\}$
  $\alpha \leftarrow$ LinearRegressionW($\widetilde{A}, \widetilde{b} = z, w$)
  $Q(t) \leftarrow \sum_{i=0}^{d} \alpha_i t^i$
  return $\widetilde{Q}(t) = Q(T \cdot \frac{t+1}{2})$
end procedure
procedure RobustPolynomialLearning$^+$(x, d, T) — Theorem 12.4.5 — a.k.a. SignalRecovery1Cluster
  $R \leftarrow \Theta(d)$
  $(n, I) \leftarrow$ GenerateIntervals(d)
  $w_j \leftarrow |I_j|/2$, for each $j \in [n]$
  for $i = 1 \to R$ **do**
    $Q_i \leftarrow$ RobustPolynomialLearning(x, d, T)
  end for
  Choose t_j from I_j uniformly at random, for each $j \in [n]$
  for $i = 1 \to R$ **do**
    $Q_i(t_1), Q_i(t_2), \cdots, Q_i(t_n) \leftarrow$ MultipointEvaluation($Q_i, \{t_1, t_2, \cdots, t_n\}$)
  end for
  $\widetilde{Q}_j \leftarrow \underset{i \in [R]}{\text{median}}\ Q_i(t_j)$, for each $j \in [n]$
  $\widetilde{A}_{j,i} \leftarrow t_j^i$, for each $(j,i) \in [n] \times \{0, 1, \cdots, d\}$
  $\alpha \leftarrow$ LinearRegressionW($\widetilde{A}, \widetilde{b} = \widetilde{Q}, w$)
  return $Q(t) \leftarrow \sum_{i=0}^{d} \alpha_i t^i$
end procedure

Algorithm 12.6 Locate k Signal, Locate k Inner, Hash To Bins

procedure LocateKSignal($x, H, G, T, \Delta, \sigma, b, z_{\text{emp}}$) — Clain 12.7.25
  Set $t \approx \log(FT)$, $t' = t/4$, $D_{\max} \approx \log_{t'}(FT)$, $R_{\text{loc}} \approx \log_{1/c}(tc)$, $L^{(1)} = 2F$
  for $i \in [D_{\max}]$ **do**
    $\Delta l \approx 2F/(t')^{i-1}$, $s \approx c$, $\widehat{\beta} = \frac{ts}{2\sigma\Delta l}$
    if $\sigma\widehat{\beta} \gtrsim T/(T\Delta)^{3/2}$ **then**
      break
    else
      $L^{(i)} \leftarrow$ LocateKInner($x, H, G, T, \Delta, \sigma, b, z_{\text{emp}}\widehat{\beta}, U, L^{(i-1)}$)
    end if
  end for
  return $L^{(i)}$
end procedure
procedure LocateKInner($x, H, G, T, \Delta, \sigma, b, z_{\text{emp}}\widehat{\beta}, U, \widetilde{L}$)
  Let $v_{j,q} \leftarrow 0$ for $(j, q) \in [B] \times [t]$
  for $r = 1 \to R_{\text{loc}}$ **do**
    Choose $\beta \in [\frac{1}{2}\widehat{\beta}, \widehat{\beta}]$ uniformly at random
    $\widehat{u}, \widehat{u}' \leftarrow$ GetLegalKSample($x, H, G, T, \Delta, \sigma, \beta, z_{\text{emp}}$)
    for $j \in [B]$ **do**
      for $i \in [m]$ **do**
        $\theta_{j,i} = \frac{1}{2\pi\sigma\beta}(\phi(\widehat{u}[j]/\widehat{u}'[j]) + 2\pi s_i), s_i \in [\sigma\beta(\widetilde{L}_j - \Delta l/2), \sigma\beta(\widetilde{L}_j + \Delta l/2)] \cap \mathbb{Z}_+$
        $f_{j,i} = \theta_{j,i} + b \pmod F$
        suppose $f_{j,i}$ belongs to region(j, q),
        add a vote to both region(j, q) and two neighbors nearby that region, e.g. region$(j, q-1)$ and region$(j, q+1)$
      end for
    end for
  end for
  for $j \in [B]$ **do**
    $q_j^* \leftarrow \{q | v_{j,q} > \frac{R_{\text{loc}}}{2}\}$
    $L_j \leftarrow$ center of region(j, q_j^*)
  end for
  return L
end procedure
procedure HashToBins($x, H, G, P_{\sigma,a,b}$) — Lemma 12.6.7
  Compute $u[j] = \sum_{i \in D} v[j + iB]$
  $\widehat{u} \leftarrow \text{FFT}(u)$
  return $\widehat{u}$
end procedure

Algorithm 12.7 Get Empirical k Energy, Get Legal k Sample, and Onestage

procedure GetEmpiricalKEnergy($x, H, G, T, \Delta, \sigma, b$) — Claim 12.7.24
  $R_{\text{est}} \leftarrow (T\Delta)^2$
  for $i = 1 \to R_{\text{est}}$ **do**
    Choose $\alpha \in [0, T]$ uniformly at random
    $\widehat{u} \leftarrow$ HashToBins($x, H, G, P_{\sigma,\alpha,b}$)
    for $j = 1 \to B$ **do**
      $z^j_{\text{emp}} \leftarrow z^j_{\text{emp}} + |\widehat{u}_j|^2$
    end for
  end for
  for $j = 1 \to B$ **do**
    $z^j_{\text{emp}} \leftarrow \sqrt{z^j_{\text{emp}}/R_{\text{est}}}$
  end for
  return z_{emp}.
end procedure
procedure GetLegalKSample($x, H, G, T, \Delta, \beta, z_{\text{emp}}$) — Lemma 12.7.23
  $R_{\text{repeat}} \leftarrow (T\Delta)^3$.
  $S^j_{\text{heavy}} \leftarrow \emptyset, \forall j \in [B]$
  for $i = 1 \to R_{\text{repeat}}$ **do**
    Choose $\alpha \in [0, T]$ uniformly at random
    $\widehat{u}^i \leftarrow$ HashToBins($x, H, G, P_{\sigma,\alpha,b}$)
    $\widehat{u}'^i \leftarrow$ HashToBins($x, H, G, P_{\sigma,\alpha+\beta,b}$)
    for $j = 1 \to B$ **do**
      if $|\widehat{u}^i_j| \geq 0.5 \cdot z^j_{\text{emp}}$ **then**
        $S_{\text{heavy},j} \leftarrow S^j_{\text{heavy}} \cup i$
      end if
    end for
  end for
  for $j = 1 \to B$ **do**
    for $i \in S^j_{\text{heavy}}$ **do**
      $w(i) \leftarrow |\widehat{u}^i_j|^2 + |\widehat{u}'^i_j|^2$
    end for
    $(\widehat{v}_j, \widehat{v}'_j) \leftarrow (\widehat{u}^i_j, \widehat{u}'^i_j)$ with probability $w(i)/\sum_{i' \in S^j_{\text{heavy}}} w(i')$ for $i \in S^j_{\text{heavy}}$
  end for
  return $\widehat{v}, \widehat{v}' \in \mathbb{C}^B$
end procedure
procedure OneStage(x, H, G, σ, b) — Lemma 12.7.20
  $z_{\text{emp}} \leftarrow$ GetEmpiricalKEnergy($x, H, G, T, \Delta, \sigma, b$)
  $L \leftarrow$ LocateKSignal($x, H, G, T, \Delta, \beta, b, z_{\text{emp}}$)
end procedure

Algorithm 12.8 Main Algorithm for k-cluster Recovery

procedure CFTKCluster(x, H, G, T, F)
  $\{\widetilde{f}_1, \cdots, \widetilde{f}_l\} \leftarrow$ FrequencyRecoveryKCluster(x, H, G, T, F)
  $\widetilde{x} \leftarrow$ SignalRecoveryKCluster$^+$($\widetilde{f}_1, \cdots, \widetilde{f}_l, \Delta = \mathrm{poly}(k, \log(1/\delta))/T, T$)
  return $\widetilde{x}$ as our hypothesis
end procedure
procedure FrequencyRecoveryKCluster(x, H, G) — Theorem 12.2.6
  for $r \in [R]$ **do**
    Choose $\sigma \in [\frac{1}{B\Delta_h}, \frac{2}{B\Delta_h}]$ uniformly at random
    Choose $b \in [0, \frac{2\pi\lfloor F/\Delta_h \rfloor]}{(\sigma B)}]$ uniformly at random
    $L_r \leftarrow$ OneStage(x, H, G, σ, b)
  end for
  $L^* \leftarrow$ MergedStages($L_1, L_2, \cdots, L_R$)
end procedure
procedure SignalRecoveryKCluster($\widetilde{f}_1, \cdots, \widetilde{f}_l, \Delta, T$)
  $d \leftarrow 5\pi((\Delta T)^{1.5} + k^3 \log k + k \log 1/\delta)$
  $m \leftarrow O((kd)^{C_3} \cdot \log^{C_3} d)$ for a constant $C_3 = 5$
  for $j = 1 \to m$ **do**
    Sample t_j from $[0, T]$ uniformly at random
    $\widetilde{A}_{j, i_1 \cdot l + i_2} \leftarrow t_j^{i_1} \cdot e^{2\pi \mathbf{i} \widetilde{f}_{i_2} t_j}$ for each $(i_1, i_2) \in \{0, \cdots, d\} \times [l]$
    $\widetilde{b}_j \leftarrow x(t_j)$
  end for
  $\alpha \leftarrow$ LinearRegression($\widetilde{A}, \widetilde{b}$)
  return $\widetilde{x}(t) \leftarrow \sum_{i_1=0}^{d} \sum_{i_2=1}^{l} \alpha_{i_1 \cdot l + i_2} t^{i_1} \cdot e^{2\pi \mathbf{i} \widetilde{f}_{i_2} t}$
end procedure
procedure SignalRecoveryKCluster$^+$($\widetilde{f}_1, \cdots, \widetilde{f}_l, \Delta, T$) — Theorem 12.9.1
  $R \leftarrow \Theta(k)$
  $d \leftarrow 5\pi((\Delta T)^{1.5} + k^3 \log k + k \log 1/\delta)$
  $m \leftarrow O((kd)^{C_3} \cdot \log^{C_3} d)$ for a constant $C_3 = 5$
  for $i = 1 \to R$ **do**
    $\widetilde{x}_i(t) \leftarrow$ SignalRecoveryKCluster($\widetilde{f}_1, \cdots, \widetilde{f}_l, \Delta, T$)
  end for
  for $j = 1 \to m$ **do**
    Sample t_j from $[0, T]$ uniformly at random
    $\widetilde{A}_{j, i_1 \cdot l + i_2} \leftarrow t_j^{i_1} \cdot e^{2\pi \mathbf{i} \widetilde{f}_{i_2} t_j}$ for each $(i_1, i_2) \in \{0, \cdots, d\} \times [l]$
    $\widetilde{b}_j \leftarrow \underset{i \in [R]}{\mathrm{median}}\ \widetilde{x}_i(t_j)$
  end for
  $\alpha \leftarrow$ LinearRegression($\widetilde{A}, \widetilde{b}$)
  return $\widetilde{x}(t) \leftarrow \sum_{i_1=0}^{d} \sum_{i_2=1}^{l} \alpha_{i_1 \cdot l + i_2} t^{i_1} \cdot e^{2\pi \mathbf{i} \widetilde{f}_{i_2} t}$
end procedure

Chapter 13

High Dimensional Fourier Transform

We consider the extensively studied problem of computing a k-sparse approximation to the d-dimensional Fourier transform of a length n signal. Our algorithm uses $O(k \log k \log n)$ samples, is dimension-free, operates for any universe size, and achieves the strongest ℓ_∞/ℓ_2 guarantee, while running in time comparable to the Fast Fourier Transform. All previous algorithms proceed either via the Restricted Isometry Property or via filter functions. Our approach totally departs from the aforementioned techniques, and we believe is a fresh look to the sparse Fourier transform problem.

13.1 Introduction

Initiated in discrete signal processing, compressed sensing/sparse recovery is an extensively studied branch of mathematics and algorithms, which postulates that a number of small linear measurements suffice to approximately reconstruct the best k-sparse approximation of a vector $x \in \mathbb{C}^n$ [CT06, CRT06b, Don06]. The literature on the subject is enormous, and has found applications in imaging, astronomy, seismology etc. One of the initial papers in the field, due to Candes, Romberg and Tao [CRT06a], counts almost $15,000$ references.

Probably the most important subtopic is the sparse Fourier transform, where one desires to reconstruct a k-sparse vector from Fourier measurements. In other words, measurements are not allowed to be generic, but have to belong to the so-called Fourier ensemble. In Optics imaging [Goo05, Voe11] and Magnetic resonance imaging (MRI) [ASSN08], the physics [Rey89] of the underlying device restricts us to the Fourier ensemble, where the sparse Fourier problem becomes highly relevant. In fact, one of the initial motivations of Candes, Romberg and Tao came out due to the aforementioned applications. The number of samples plays a crucial role: they determine the amount of radiation a patient receives in CT scans, and taking fewer samples can reduce the amount of time the patient needs to stay in the machine. The framework has found its way in practical life-changing applications. Software includes the Compressed Sensing GRAB-VIBE, CS SPACE, CS SEMAC and CS TOF by Siemens [Sie], as well as Compressed Sense by Phillips [Phi]. Its incorporation in the MRI technology allows faster acquisition rates, depiction of dynamic processes or moving organs, as well as acceleration of MRI scanning up to a factor of 40. On the webpage of SIEMENS Healthineers, for example, one can see the following, as well as numerous similar statements.

This allows bringing the advantages of Compressed Sensing GRASP-VIBE *to daily clinical routine.*

- *Perform push-button, free-breathing liver dynamics.*
- *Overcome timing challenges in dynamic imaging and respiratory artifacts.*
- *Expand the patient population eligible for abdominal* MRI.

The Fourier transform is in fact ubiquitous: image processing, audio processing, telecommunications, seismology, polynomial multiplication, SUBSET SUM and other textbook algorithms are a few of the examples where the Fast Fourier Transform finds applications. The Fast Fourier Transform by Cooley and Tukey [CT65] runs in $O(n \log n)$ time, and has far-reaching applications in all of the aforementioned cases. It is thus expected that algorithms which exploit sparsity assumptions about the input, and can outperform FFT in applications are of high practical value. Generally, the two most important parameters one would like to optimize are the sample complexity, i.e. the numbers needed to obtain from the time domain, as well as time needed to approximate the Fourier Transform.

Two different lines of research exist for the problem: the one focuses solely on sample complexity, while the other tries to achieve sublinear time while keeping the sample complexity as low as possible. The first line of research operates via the renowned Restricted Isometry Property (RIP), which proceeds by taking random samples and solving a linear/convex program, or an iterative thresholding procedure [CT06, DDTS06, TG07, BD08, DM08, RV08, BD09b, BD09a, NT09b, NV09, GK09, BD10, NV10, Fou11, Bou14, HR16]. The analysis of the algorithms is performed in the following way, in two steps. The first step ensures

that, after sampling an appropriate number of points from the time domain, the inverse DFT matrix restricted on the rows indexed by those points acts as a near isometry on the space of k-sparse vectors. All of the state of the art results [CT06, RV08, Bou14, HR16] employ chaining arguments to make the analysis of this sampling procedure as tight as possible. The second part is how to exploit the aforementioned near-isometry property to find the best k-sparse approximation to the signal. There the approaches either follow an iterative procedure which gradually denoise the signal [BD08, NT09b, NV09], or perform ℓ_1 minimization [CT06], a method that promotes sparsity of solutions.

The second line of research tries to implement arbitrary linear measurements via sampling Fourier coefficients [GL89, Man92, KM93, GGI+02, AGS03, GMS05, Iwe08, Iwe10, HIKP12a, HIKP12b, LWC13, Iwe13, PR14, IKP14, IK14, Kap16, Kap17, CI17, BZI17, MZIC17, LN19] and use sparse functions (in the time domain) which behave like bandpass filters in the frequency domain. The seminal work of Kapralov [Kap17] achieves $O(k \log n)$ samples and running time that is some log factors away from the sample complexity. This would be the end of the story, apart from the fact that this algorithm does not scale well with dimension, since it has an exponential dependence on d. Indeed, in many applications, one is interested in higher dimensions, rather than the one-dimensional case. The main reason[1] why this curse of dimensionality appears is due to the lack of dimension-independent ways to construct functions that approximate the ℓ_∞ ball and are sufficiently sparse in the time domain. A very nice work of Kapralov, Velingker and Zandieh [KVZ19] tries to remedy that by combining the standard execution of FFT with careful aliasing, but their algorithm

[1]But not the only one: pseudorandom permutations for sparse FT in high dimensions also incur an exponential loss, and it is not known whether this can be avoided.

works in a noiseless setting, and has a polynomial, rather than linear, dependence on k; the running time is polynomial in $k, \log n$ and the exponential dependence is avoided. It is an important and challenging question whether a robust and more efficient algorithm can be found.

We note that in many applications, such as MRI or computed tomography (CT), the main focus is the sample complexity; the algorithms that have found their way to industry are, to the best of our knowledge, not concerned with sublinear running time, but with the number of measurements, which determine the acquisition time, or in CT the radiation dose the patient receives. Lastly, we bring to the readers' attention the recent work on sparse Fourier transform in the continuous setting, see [Iwe10, Iwe13, Iwe13, BCG+12, PS15, CKPS16, AKM+19].

Our Contribution. We give a new algorithm for the sparse Fourier transform problem, which has $O(k \log n \log k)$ sample complexity for any dimension, and achieves the ℓ_∞/ℓ_2 guarantee[2], while running in time $\widetilde{O}(n)$. The previous state of the art algorithm that achieved such a guarantee is the work of Indyk and Kapralov [IK14], which has $2^{O(d \log d)} k \log n$ sample complexity; an exponentially worse dependence on d. The work of [HR16] obtains $O(k \log n \log^2 k)$ samples in any dimension, but has a much weaker guarantee, while their approach requires $\Omega(k \log n \log k)$ samples in high dimensions [Rao19]. Moreover, the algorithm in [IK14] operates when the universe size in each dimension is a power of 2, whereas there is no restriction in our work. To obtain our result, we introduce a set of new tech-

[2]This is the strongest guarantee in the sparse recovery literature. See also the caption of the table in Section 1.2

niques, deviating from previous work, which used the Restricted Isometry Property and/or filter functions.

13.1.1 Preliminaries

For any positive integer n, we use $[n]$ to denote $\{1, 2, \cdots, n\}$.

We assume that the universe size $n = p^d$ for any positive integer p. Our algorithm facilitates $n = \Pi_{j=1}^{d} p_j$ for any positive integers $p_1, \ldots, p_d$, but we decide to present the case $n = p^d$ for ease of exposition; the proof is exactly the same in the more general case. Let $\omega = e^{2\pi \mathbf{i}/p}$ where $\mathbf{i} = \sqrt{-1}$. We will work with the normalized d-dimensional Fourier transform

$$\widehat{x}_f = \frac{1}{\sqrt{n}} \sum_{t \in [p]^d} x_t \cdot \omega^{f^\top t}, \forall f \in [p]^d$$

and the inverse Fourier transform is

$$x_t = \frac{1}{\sqrt{n}} \sum_{f \in [p]^d} \widehat{x}_f \cdot \omega^{-f^\top t}, \forall t \in [p]^d.$$

For any vector x and integer k, we denote x_{-k} to be the vector obtained by zeroing out the largest (in absolute value) k coordinates from x.

13.1.2 Our result

Apart from being dimension-independent and working for any universe size, our algorithm satisfies ℓ_∞/ℓ_2, which is the strongest guarantee out of the standard guarantees considered in compressed sensing tasks. A guarantee G_1 is stronger than guarantee G_2 if for any k-sparse recovery algorithm that satisfies G_1 we can obtain a $\Omega(k)$-sparse recovery

algorithm that satisfies G_2. See also below for a comparison between ℓ_∞/ℓ_2 and ℓ_2/ℓ_2, the second stronger guarantee.

Previous work is summarized in Table 13.1. Our result is the following.

Theorem 13.1.1 (main result, informal version)**.** *Let $n = p^d$ where both p and d are positive integers. Let $x \in \mathbb{C}^{[p]^d}$. Let $k \in \{1, \ldots, n\}$. Assume that $R^* \geq \|\widehat{x}\|_\infty / \|\widehat{x}_{-k}\|_2$ where $\log R^* = O(\log n)$ (signal-to-noise ratio). There is an algorithm that takes $O(k \log k \log n)$ samples from x, runs in $\widetilde{O}(n)$ time, and outputs a $O(k)$-sparse vector y such that*

$$\|\widehat{x} - y\|_\infty \leq \frac{1}{\sqrt{k}} \|\widehat{x}_{-k}\|_2$$

holds with probability at least $1 - 1/\operatorname{poly}(n)$.

Comparison between ℓ_∞/ℓ_2 and ℓ_2/ℓ_2 (or ℓ_2/ℓ_1). For the sake of argument, we will consider only the ℓ_2/ℓ_2 guarantee which is stronger than ℓ_2/ℓ_1. The ℓ_2/ℓ_2 guarantee is the following: for $\widehat{x} \in \mathbb{C}^n$ one should output a z such that $\|\widehat{x} - z\|_2 \leq C\|\widehat{x}_{-k}\|_2$, where $C > 1$ is the approximation factor. Consider $C = 1.1$ [3], and think of the following signal: for a set S of size $0.05k$ we have $|\widehat{x}_i| = \frac{2}{\sqrt{k}} \|\widehat{x}_{\bar{S}}\|_2$. Then the all zeros vectors is a valid solution for the ℓ_2/ℓ_2 guarantee, since

$$\|\vec{0} - \widehat{x}\|_2^2 = \|\widehat{x}_S\|_2^2 + \|\widehat{x}_{\bar{S}}\|_2^2 = 0.05k \cdot \frac{4}{k} \|\widehat{x}_{\bar{S}}\|_2^2 + \|\widehat{x}_{\bar{S}}\|_2^2 = 1.2\|\widehat{x}_{\bar{S}}\|_2^2 < 1.1^2 \|\widehat{x}_{\bar{S}}\|_2^2.$$

[3] This is the case with the RIP based approaches, which obtain ℓ_2/ℓ_1. In fact many filter-based algorithms facilitate $(1 + \epsilon)$ on the right hand side, with the number of measurements being multiplied by ϵ^{-1}. By enabling the same dependence on ϵ^{-1} our algorithm facilitates a multiplicative ϵ factor on right hand side of the ℓ_∞/ℓ_2, which makes it much stronger. Thus, a similar argument can go through.

Reference	Samples	Time	Filter	RIP	Guarantee
[GMS05]	$k \log^{O(d)} n$	$k \log^{O(d)} n$	Yes	No	ℓ_2/ℓ_2
[CT06]	$k \log^6 n$	$\text{poly}(n)$	No	Yes	ℓ_2/ℓ_1
[RV08]	$k \log^2 k \log(k \log n) \log n$	$\widetilde{O}(n)$	No	Yes	ℓ_2/ℓ_1
[HIKP12a]	$k \log^d n \log(n/k)$	$k \log^d n \log(n/k)$	Yes	No	ℓ_2/ℓ_2
[CGV13]	$k \log^3 k \log n$	$\widetilde{O}(n)$	No	Yes	ℓ_2/ℓ_1
[IK14]	$2^{d \log d} k \log n$	$\widetilde{O}(n)$	Yes	No	ℓ_∞/ℓ_2
[Bou14]	$k \log k \log^2 n$	$\widetilde{O}(n)$	No	Yes	ℓ_2/ℓ_1
[HR16]	$k \log^2 k \log n$	$\widetilde{O}(n)$	No	Yes	ℓ_2/ℓ_1
[Kap16]	$2^{d^2} k \log n \log \log n$	$2^{d^2} k \log^{d+3} n$	Yes	No	ℓ_2/ℓ_2
[KVZ19]	$k^3 \log^2 k \log^2 n$	$k^3 \log^2 k \log^2 n$	Yes	Yes	Exactly k-sparse
Theorem 13.1.1	$k \log k \log n$	$\widetilde{O}(n)$	No	No	ℓ_∞/ℓ_2

Table 13.1: $n = p^d$. We ignore the O for simplicity. The ℓ_∞/ℓ_2 is the strongest possible guarantee, with ℓ_2/ℓ_2 coming second, ℓ_2/ℓ_1 third and exactly k-sparse being the less strong. We note that [CT06, RV08, CGV13, Bou14, HR16] obtain a uniform guarantee, i.e. with $1 - 1/\text{poly}(n)$ they allow reconstruction of all vectors; ℓ_∞/ℓ_2 and ℓ_2/ℓ_2 are impossible in the uniform case [CDD09]. We also note that [RV08, CGV13, Bou14, HR16] give improved analysis of the Restricted Isometry property; the algorithm is suggested and analyzed (modulo the RIP property) in [BD08]. The work in [HIKP12a] does not explicitly state the extension to the d-dimensional case, but can easily be inferred from the arguments. [HIKP12a, IK14, Kap16, KVZ19] work when the universe size in each dimension are powers of 2. We also assume that the signal-to-noise ratio is bounded by a polynomial of n, which is a standard assumption in the sparse Fourier transform literature [HIKP12a, IK14, Kap16, Kap17, LN19].

It is clear that since $\vec{0}$ is a possible output, we may not recover any of the coordinates in S, which is the set of "interesting" coordinates. On the other hand, the ℓ_∞/ℓ_2 guarantee does allow the recovery of every coordinate in S. This is a difference of recovering all $0.05k$ versus 0 coordinates. From the above discussion, one can conclude in the case where there is too much noise, ℓ_2/ℓ_2 becomes much weaker than ℓ_∞/ℓ_2, and can be even meaningless. Thus, ℓ_∞/ℓ_2 is highly desirable, whenever it is possible. The same exact argument holds for

ℓ_2/ℓ_1.

13.1.3 Summary of previous Filter function based technique

One of the two ways to perform Fourier sparse recovery is by trying to implement arbitrary linear measurements, with algorithms similar to the ubiquitous CountSketch [CCF02]. In the general setting CountSketch hashes every coordinate to one of the $O(k)$ buckets, and repeats $O(\log n)$ times with fresh randomness. Then, it is guaranteed that every heavy coordinate will be isolated, and the contribution from non-heavy elements is small. To implement this in the Fourier setting becomes a highly non-trivial task however: one gets access only to the time-domain but not the frequency domain. One natural way to do this is to exploit the convolution theorem and find a function which is sparse in the time domain and approximates the indicator of an interval (rectangular pulse) in the frequency domain; these functions are called (bandpass) filters. Appropriate filters were designed in [HIKP12a, HIKP12b]: they were very good approximations of the rectangular pulse, i.e. the contribution from elements outside the passband zone contributed only by $1/\operatorname{poly}(n)$ their mass. These filters had an additional $\log n$ factor (in one dimension) in the sparsity of the time domain and they are sufficient for the purposes of [HIKP12a], but in high dimensions this factor becomes $\log^d n$. Filters based on the Dirichlet kernel give a better dependence in terms of sparsity and dimension (although still an exponential dependence on the latter), but the leak to subsequent buckets, i.e. coordinates outside the passband zone contribute a constant fraction of their mass, in contrast to the filter used in [HIKP12a]. Thus one should perform additional denoising, which is a non-trivial task. The seminal work of Indyk and Kapralov [IK14] was the first that showed how to perform sparse recovery with these

filters, and then Kapralov [Kap16, Kap17] extended this result to run in sublinear time. We note, that any filter-based approach with filters which approximate the ℓ_∞ box, suffers from the curse of dimensionality. [KVZ19] devised an algorithm which avoids the curse of dimensionality by using careful aliasing, but it works in the noiseless case and has a cubic dependence on k.

13.1.4 RIP property-based algorithms: a quick overview

We say the matrix $A \in \mathbb{C}^{m\times n}$ satisfies RIP (Restricted Isometry Property [CT05]) of order k if for all k-sparse vectors $x \in \mathbb{C}^n$ we have $\|Ax\|_2^2 \approx \|x\|_2^2$. A celebrated result of Candes and Tao [CT06] shows that Basis Pursuit (ℓ_1 minimization) suffices for sparse recovery, as long as the samples from the time domain satisfy RIP. In [CT06] it was also proved using generic chaining that random sampling with oversampling factor $O(\log^6 n)$ gives RIP property for any orthonormal matrix with bounded entries by $1/\sqrt{n}$. Then [RV08] improved the bound to $O(k\cdot\log^2 k\cdot\log(k\log n)\cdot\log n)$ and [CGV13] improved it to $O(k\cdot\log^3 k\cdot\log n)$. Subsequent improvement by Bourgain [Bou14] has lead to $O(k\log k\cdot\log^2 n)$ samples, improved by Haviv and Regev to $O(k\log^2 k\cdot\log n)$[HR16]. The fastest set of algorithms are iterative ones: for example Iterative Hard Thresholding [BD09a] or CoSaMP [NT09b] run $O(\log n)$ iterations[4] and each iteration takes $\widetilde{O}(n)$ time.

We note the very recent lower bound of [Rao19]: a subsampled Fourier matrix that satisfies the RIP properties should have $\Omega(k\log k\cdot d)$ rows[5]. This bound is particularly useful in high dimensions, since it deteriorates to a trivial bound in low dimensions. We still believe

[4] To be precise, their running time is logarithmic in the signal-to-noise ratio, but we assumed throughout the paper that this quantity is polynomial in n.

[5] [BLLM19] independently gives a similar bound for $d = \log n$.

though that a bound of $\Omega(k \log k \log n)$ should hold in all dimensions. Thus, what remains is to obtain the ℓ_2/ℓ_2 guarantee by giving a tighter analysis, and removing the one $\log k$ factor to match the lower bound, but our algorithm already allows Fourier sparse recovery with these number of samples, even with a stronger guarantee.

13.1.5 Overview of our technique

Let $x \in \mathbb{C}^{[p]^d}$ denote our input signal in the time domain. In the following we assume the knowledge of $\mu = \frac{1}{\sqrt{k}}\|\widehat{x}_{-k}\|_2$ and R^* which is an upper bound of $\|\widehat{x}\|_\infty/\mu$, and bounded by $\mathrm{poly}(n)$. These are standard assumption [HIKP12a, IK14, Kap16, Kap17, LN19] in the sparse Fourier transform literature. The bound on R^* is useful for bounding the running time (or the number of measurements in [HIKP12a]) and in any of [HIKP12a, IK14, Kap16, Kap17, LN19] a $\log n$ can be substituted by $\log R^*$ in the general case, which is also the case for our algorithm. We note that our algorithm will be correct with probability $1 - 1/\mathrm{poly}(n)$ whenever $R^* < 2^{n^{100}}$; this is fine for every reasonable application. We assumed the rounding errors in FFT computation to be negligible, similarly to Remark 3.4 in [IK14].

Consider the simplest scenario: $d = 1$, p is a prime number and a 1-sparse signal $\widehat{x}$ which is 1 on some frequency f^*. From a sample x_t in the time-domain what would be the most reasonable way to find f^*? For every $f \in [p]$ we would compute

$$\sqrt{n}\omega^{ft}x_t = \sqrt{n}\omega^{ft} \cdot \frac{1}{\sqrt{n}} \sum_{f' \in [p]} \omega^{-f't}\widehat{x}_{f'} = \omega^{(f-f^*)t},$$

and keep, for $t \neq 0$, the frequency that gives a real number. Since $(f - f^*)t$ will be zero only for $f = f^*$, we are guaranteed correct recovery. In the noisy and multi-dimensional case or p is an arbitrary integer, however, this argument will not work, because of the presence of

contribution from other elements and the fact that $(f - f^*)^\top t$ can be zero modulo p for other frequencies apart from f. However, we can take a number of samples t and average $\sqrt{n}\omega^{f^\top t}$, and hope that this will make the contribution from other frequencies small enough, so that we can infer whether f corresponds to a heavy coordinate or not. More specifically, we pick a list T of size $O(k)$ uniformly at random from $[p]^d$ and compute

$$\frac{\sqrt{n}}{|T|} \sum_{t \in T} \omega^{f^\top t} x_t$$

for all frequencies f. We show that if $|T| = O(k)$ our estimator is good on average (and later we will maintain $O(\log n)$ independent instances and take the median to make sure with probability $1 - 1/\operatorname{poly}(n)$ the estimators for all the frequencies are good), and in fact behaves like a crude filter, similarly to the ones used in [IK14], in the sense that every coordinate contributes a non-trivial amount to every other coordinate. However, these estimators do not suffer from the curse of dimensionality and our case is a little bit different, requiring a quite different handling. The main reason is that in contrast to the filters used in [IK14], there is not an easy way to formulate an isolation argument from heavy elements that would allow easy measurement re-use, like Definition 5.2 and Lemma 5.4 from [IK14]. Buckets induced by filter functions have a property of locality, since they correspond to approximate ℓ_∞ boxes (with a polynomial decay outside of the box) in $[p]^d$: the closer two buckets are the more contribute the elements of one into the other. Our estimators on the other side do not enjoy such a property. Thus, one has to go via a different route.

In what follows, we will discuss how to combine the above estimators with an iterative loop that performs denoising, i.e. removes the contribution of every heavy element to other heavy elements.

We first implement a procedure which takes $O(k \log n)$ uniform random measurements from x and has the guarantee that for any $\nu \geq \mu$ any $y \in \mathbb{C}^{[p]^d}$ where $\|\widehat{x} - y\|_\infty \leq 2\nu$ and y is independent from the randomness of the measurements, the procedure outputs a $O(k)$-sparse $z \in \mathbb{C}^{[p]^d}$ such that $\|\widehat{x} - y - z\|_\infty \leq \nu$ with probability $1 - 1/\operatorname{poly}(n)$.

Lemma 13.1.2 (LinfinityReduce procedure/data structure, informal)**.** *Let $\mu = \frac{1}{\sqrt{k}}\|\widehat{x}_{-k}\|_2$, and $\nu \geq \mu$. Let $\mathcal{T}^{(0)}$ be a list of $O(k \log n)$ i.i.d. elements in $[p]^d$. Let S be top $O(k)$ coordinates in $\widehat{x}$. There is a procedure that takes $\{x_t\}_{t \in \mathcal{T}}$, $y \in \mathbb{C}^{[p]^d}$ and ν as input, runs in $\widetilde{O}(n)$ time, and outputs $z \in \mathbb{C}^{[p]^d}$ so that if $\|\widehat{x} - y\|_\infty \leq 2\nu$, $\operatorname{supp}(y) \subseteq S$ and y is independent from the randomness of $\mathcal{T}^{(0)}$, then $\|\widehat{x} - y - z\|_\infty \leq \nu$ and $\operatorname{supp}(z) \subseteq S$ with probability $1 - 1/\operatorname{poly}(n)$ under the randomness of $\mathcal{T}^{(0)}$.*

Namely, we can take $O(k \log n)$ measurements and run the procedure in Lemma 13.1.2 to reduce (the upper bound of) the ℓ_∞ norm of the residual signal by half. We call the procedure in Lemma 13.1.2 LinfinityReduce procedure. More generally, we can take $O(H \cdot k \log n)$ measurements and run the LinfinityReduce procedure H times to reduce the ℓ_∞ norm of the residual signal to $1/2^H$ of its original magnitude, with failure probability at most $1/\operatorname{poly}(n)$. This is because if $\nu \geq 2^H \mu$ and $\|\widehat{x} - y\|_\infty \leq \nu$, then we can proceed in H iterations where in the h-th iteration ($h \in [H]$) we can take $O(k \log n)$ fresh measurements from x and run the LinfinityReduce procedure to make the ℓ_∞ norm of the residual signal at most $2^{-h}\nu$. Note that if we set $H = \log R^*$, we have already obtained a recovery algorithm taking $O(k \log n \log R^*)$ measurements, because we can drive down (the upper bound of) the ℓ_∞ norm of the residual signal from $\|\widehat{x}\|_\infty$ to μ in $\log R^*$ iterations.

13.1.5.1 $O(k \log n)$ measurements for $k = O(\log n)$

We first discuss a measurement reuse idea that leads us to a sparse recovery algorithm (Algorithm 13.1) taking $O(k \log n)$ measurements for $k = O(\log n)$. We set $H = 5$, and let $\mathcal{T} = \{\mathcal{T}^{(1)}, \ldots, \mathcal{T}^{(H)}\}$, where each $\mathcal{T}^{(h)}$ is a list of $O(k \log n)$ i.i.d. elements in $[p]^d$. Note that $\mathcal{T}^{(1)}, \ldots, \mathcal{T}^{(H)}$ are independent. In our sparse Fourier recovery algorithm, we will measure x_t for all $t \in \mathcal{T}$.

In a nutshell, our approach finely discretizes the space of possible trajectories the algorithm could evolve, and carefully argues about the correctness of the algorithm by avoiding the intractable union-bound over all trajectories.

Recovery algorithm. The recovery algorithm proceeds in $\log R^* - H + 1$ iterations, where each iteration (except the last iteration) the goal is to reduce the upper bound of ℓ_∞ norm of the residual signal by half. Initially, the upper bound is R^*. It is important to note that we use the same measurements $\mathcal{T} = \{\mathcal{T}^{(1)}, \ldots, \mathcal{T}^{(H)}\}$ in all of these $\log R^* - H + 1$ iterations.

In the following, we will describe one iteration of the recovery algorithm. Let $y \in \mathbb{C}^{[p]^d}$ denote the sparse vector recovered so far, and let the upper bound of $\|\widehat{x} - y\|_\infty$ be 2ν. Running the LinfinityReduce procedure H times where in the h-th time we use measurements in $\mathcal{T}^{(h)}$, we obtain a $O(k)$-sparse z such that with probability $1 - 1/\operatorname{poly}(n)$, $\|\widehat{x} - y - z\|_\infty \leq 2^{1-H}\nu \leq 0.1\nu$ (we call such z a desirable output by the LinfinityReduce procedure). Instead of taking $y + z$ as our newly recovered sparse signal, for each $f \in \operatorname{supp}(y + z)$, we project $y_f + z_f$ to the nearst points in $\mathcal{G}_{0.6\nu} := \{0.6\nu(x + y\mathbf{i}) : x, y \in \mathbb{Z}\}$ and assign to y'_f, where y' denotes our newly recovered sparse signal. For all $f \notin \operatorname{supp}(y + z)$, we let $y'_f = 0$.

Algorithm 13.1 Fourier Sparse Recovery by Projection, $O(k \log n)$ Measurements When $k = O(\log n)$

procedure FOURIERSPARSERECOVERYBYPROJECTION(x, n, k, μ, R^*) ▷ Section 13.1.5.1
  Require that $\mu = \frac{1}{\sqrt{k}}\|\widehat{x}_{-k}\|_2$ and $R^* \geq \|\widehat{x}\|_\infty / \mu$
  $H \leftarrow 5$, $\nu \leftarrow \mu R^*/2$, $y \leftarrow \vec{0}$ ▷ $y \in \mathbb{C}^{[p]^d}$ refers to the sparse vector recovered so far
  Let $\mathcal{T} = \{\mathcal{T}^{(1)}, \cdots, \mathcal{T}^{(H)}\}$ where each $\mathcal{T}^{(h)}$ is a list of i.i.d. uniform samples in $[p]^d$
  while true do
    $\nu' \leftarrow 2^{1-H}\nu$
    Use $\{x_t\}_{t \in \mathcal{T}}$ to run the LINFINITYREDUCE procedure (in Lemma 13.1.2) H times (use samples in $\mathcal{T}^{(h)}$ for each $h \in [H]$), and finally it finds z so that $\|\widehat{x} - y - z\|_\infty \leq \nu'$
    if $\nu' \leq \mu$ **then return** $y + z$ ▷ We found the solution
    $y' \leftarrow \vec{0}$
    for $f \in \mathrm{supp}(y + z)$ **do**
      $y'_f \leftarrow \Pi_{0.6\nu}(y_f + z_f)$ ▷ We want $\|\widehat{x} - y'\|_\infty \leq \nu$ and the dependence between y' and $\mathcal{T}$ is under control
    end for
    $y \leftarrow y'$, $\nu \leftarrow \nu/2$
  end while
end procedure

To simplify our exposition, here we introduce some notations. We call $\mathcal{G}_{0.6\nu}$ a grid of side length 0.6ν, and we generalize the definition to any side length. Namely, for any $r_g > 0$, let grid $\mathcal{G}_{r_g} := \{r_g(x + y\mathbf{i}) : x, y \in \mathbb{Z}\}$. Moreover, we define $\Pi_{r_g} : \mathbb{C} \to \mathcal{G}_{r_g}$ to be the mapping that maps any element in $\mathbb{C}$ to the nearest element in $\mathcal{G}_{r_g}$. Now we can write y' as

$$y'_f = \begin{cases} \Pi_{0.6\nu}(y_f + z_f), & \text{if } f \in \mathrm{supp}(y + z); \\ 0, & \text{if } f \notin \mathrm{supp}(y + z). \end{cases}$$

At the end of each iteration, we assign y' to y, and shrink ν by half. In the last iteration, we will not compute y', instead we output $y + z$. We present the algorithm in Algorithm 13.1.

Analysis. We analyze y' conditioned on the event that $\|\widehat{x} - y - z\|_\infty \leq 0.1\nu$ (i.e. z is a desirable output by the LinfinityReduce procedure, which happens with probability $1 - 1/\operatorname{poly}(n)$). We will prove that y' has two desirable properties: (1) $\|\widehat{x} - y'\|_\infty \leq \nu$; (2) the dependence between y' and our measurements $\mathcal{T}$ is under control so that after taking y' as newly recovered sparse signal, subsequent executions of the LinfinityReduce procedure with measurements $\mathcal{T}$ still work with good probability. Property (1) follows from triangle inequality and the fact that $\|\widehat{x} - (y+z)\|_\infty \leq 0.1\nu$ and $\|(y+z) - y'\|_\infty \leq 0.6\nu$. We now elaborate on property (2). We can prove that for any $f \in [p]^d$,

$$y'_f \in \left\{\Pi_{0.6\nu}(\widehat{x}_f + 0.1\nu(\alpha + \beta\mathbf{i})) : \alpha, \beta \in \{-1, 1\}\right\}.$$

Let S denote top $26k$ coordinates (in absolute value) of $\widehat{x}$. We can further prove that for any $f \in \overline{S}$, $y'_f = 0$. Therefore, the total number of possible y' is upper bounded by $4^{|S|} = 4^{O(k)}$. If $k = O(\log n)$, we can afford union bounding all $4^{O(k)} = \operatorname{poly}(n)$ possible y', and prove that with probability $1 - 1/\operatorname{poly}(n)$ for all possible value of y' if we take y' as our newly recovered sparse signal then in the next iteration the LinfinityReduce procedure with measurements $\mathcal{T}$ gives us a desirable output.

Sufficient event. More rigorously, we formulate the event that guarantees successful execution of Algorithm 13.1. Let $\mathcal{E}_1$ be the event that for all $O(\log R^*)$ possible values of $\nu \in \{\mu\frac{R^*}{2}, \mu\frac{R^*}{4}, \ldots, \mu 2^{H-1}\}$, for all possible vector y where $y_f = 0$ for $f \in \overline{S}$ and $y_f \in \{\Pi_{0.6\nu}(\widehat{x}_f + 0.1\nu(\alpha + \beta\mathbf{i})) : \alpha, \beta \in \{-1, 1\}\}$ for $f \in S$ (we also need to include the case that $y = \vec{0}$ for the success of the first iteration), running the LinfinityReduce procedure (in Lemma 13.1.2) H times (where in the h-th time measurements $\{x_t\}_{t \in \mathcal{T}^{(h)}}$ are used

to reduce the error from $2^{2-h}\nu$ to $2^{1-h}\nu$) finally gives z so that $\|\widehat{x}-y-z\|_\infty \le 2^{1-H}\nu$. The randomness of $\mathcal{E}_1$ comes from $\mathcal{T} = \{\mathcal{T}^{(1)}, \ldots, \mathcal{T}^{(H)}\}$.

First, event $\mathcal{E}_1$ happens with probability $1 - 1/\operatorname{poly}(n)$. This is because there are $4^{O(k)} \log R^*$ possible combinations of ν and y to union bound, and each has failure probability at most $1/\operatorname{poly}(n)$. For $k = O(\log n)$, and any $R^* < 2^{n^{100}}$ this gives the desired result. Second, conditioned on event $\mathcal{E}_1$ happens, Algorithm 13.1 gives correct output. This can be proved by a mathematical induction that in the t-th iteration of the while-true loop in Algorithm 13.1, $\|\widehat{x}-y\|_\infty \le 2^{-t}\mu R^*$.

13.1.5.2 $O(k \log k \log n)$ measurements for arbitrary k

Using random shift to reduce projection size. We remark that in the analysis of the previous recovery algorithm, if we can make sure that every $y_f + z_f$ has only one possible outcome when projecting to the grid $\mathcal{G}_{0.6\nu}$, then we no longer need to union bound $4^{O(k)}$ events. However, if $\widehat{x}_f$ is very close to a grid point in $\mathcal{G}_{0.6\nu}$ (or $\widehat{x}_f \in \mathcal{G}_{0.6\nu}$), then no matter how close $y_f + z_f$ and $\widehat{x}_f$ are, $\Pi_{0.6\nu}(y_f + z_f)$ will have 4 possible values.

To address this, we introduce random shift, whose property is captured by Lemma 13.1.3. To simplify notation, for any $r_b > 0$ and $c \in \mathbb{C}$ we define box $\mathcal{B}_\infty(c, r_b) := \{c + r_b(x + y\mathbf{i}) : x, y \in [-1, 1]\}$. For any $S \subseteq \mathbb{C}$, let $\Pi_{r_g}(S) = \{\Pi_{r_g}(c) : c \in S\}$.

Lemma 13.1.3 (property of a randomly shifted box, informal)**.** *If we take a box of side length $2r_b$ and shift it randomly by an offset in $\mathcal{B}_\infty(0, r_s)$ (or equivalently, $[-r_s, r_s] \times [-r_s, r_s]$) where $r_s \ge r_b$, and next we round every point inside that shifted box to the closest point in G_{r_g} where $r_g \ge 2r_s$, then with probability at least $(1 - r_b/r_s)^2$ everyone will be rounded to the same point.*

In the following, we present a sparse Fourier recovery algorithm that incorporates the random shift idea. The algorithm takes $O(k \log k \log n)$ measurements. We set $H = O(\log k)$ and take measurements of $\mathcal{T} = \{\mathcal{T}^{(1)}, \ldots, \mathcal{T}^{(H)}\}$, where $\mathcal{T}^{(h)}$ is a list of $O(k \log n)$ i.i.d elements in $[p]^d$. Note that $\mathcal{T}^{(1)}, \ldots, \mathcal{T}^{(H)}$ are independent, and the choice of H is different from Section 13.1.5.1.

In a nutshell, our approach finely discretizes the space of possible trajectories the algorithm could evolve; in contrast to the case of $k = O(\log n)$, the number of trajectories becomes much larger. For that, we perform random shifting after the samples are taken, such that the number of trajectories is pruned, and we need to argue for a much smaller collection of events. We note that we make the decoding algorithm be randomized: the randomness in previous algorithms was present only when taking samples, and the rest of the algorithm was deterministic. However, here we need randomness in both cases, and that helps us prune the number of possible trajectories. To the best of our knowledge, this is a novel argument and approach, and might be helpful for future progress in the field.

Recovery algorithm. Similar to the $k = O(\log n)$ case (Section 13.1.5.1), we assume that we have already obtained a $O(k)$-sparse $y \in \mathbb{C}^{[p]^d}$ such that $\|\widehat{x} - y\|_\infty \leq 2\nu$ and y is "almost" independent from $\mathcal{T}$. We show how to obtain $y' \in \mathbb{C}^{[p]^d}$ such that $\|\widehat{x} - y'\|_\infty \leq \nu$ with probability $1 - 1/\operatorname{poly}(n)$ and y' is "almost" independent from $\mathcal{T}$. The main idea is we first run LinfinityReduce procedure $H = O(\log k)$ times to get an $O(k)$-sparse $z \in \mathbb{C}^{[p]^d}$ such that $\|\widehat{x} - y - z\|_\infty \leq \frac{1}{2^{20}k}\nu$. Then we repeatedly sample a uniform random shift $s \in \mathbb{C}$ (where $\|s\|_\infty \leq 10^{-3}\nu$; here we consider complex numbers as 2D vectors) until for every $f \in \operatorname{supp}(y + z)$, all the points (or complex numbers) of the form $y_f + z_f + s + a + b\mathbf{i}$ where

Algorithm 13.2 Fourier Sparse Recovery by Random Shift and Projection (Informal Version)

procedure FOURIERSPARSERECOVERY(x, n, k, μ, R^*) ▷ Theorem 13.1.1, $n = p^d$
  Require that $\mu = \frac{1}{\sqrt{k}}\|\widehat{x}_{-k}\|_2$ and $R^* \geq \|\widehat{x}\|_\infty / \mu$
  $H \leftarrow O(\log k)$, $\nu \leftarrow \mu R^*/2$, $y \leftarrow \vec{0}$ ▷ $y \in \mathbb{C}^{[p]^d}$ refers to the sparse vector recovered so far
  Let $\mathcal{T} = \{\mathcal{T}^{(1)}, \cdots, \mathcal{T}^{(H)}\}$ where each $\mathcal{T}^{(h)}$ is a list of i.i.d. uniform samples in $[p]^d$
  while true do
    $\nu' \leftarrow \frac{1}{2^{20}k}\nu$
    Use $\{x_t\}_{t\in\mathcal{T}}$ to run the LINFINITYREDUCE procedure (in Lemma 13.1.2) H times (use samples in $\mathcal{T}^h$ for each $h \in [H]$), and finally it finds z so that $\|\widehat{x} - y - z\|_\infty \leq \nu'$
    if $\nu' \leq \mu$ **then return** $y + z$ ▷ We found the solution
    repeat
      Pick $s \in \mathcal{B}_\infty(0, 10^{-3}\nu)$ uniformly at random
    until $\forall f \in \mathrm{supp}(y+z), |\Pi_{0.04\nu}(\mathcal{B}_\infty(y_f + z_f + s, \nu'))| = 1$
    $y' \leftarrow \vec{0}$
    for $f \in \mathrm{supp}(y+z)$ **do**
      $y'_f \leftarrow \Pi_{0.04\nu}(y_f + z_f + s)$ ▷ We want $\|\widehat{x} - y'\|_\infty \leq \nu$ and the dependence between y' and $\mathcal{T}$ is under control
    end for
    $y \leftarrow y'$, $\nu \leftarrow \nu/2$
  end while
end procedure

$a, b \in [-\frac{\nu}{2^{20}k}, \frac{\nu}{2^{20}k}]$ round to the same grid point in $\mathcal{G}_{0.04\nu}$. Finally, for every $f \in \mathrm{supp}(y+z)$, we assign $\Pi_{0.04\nu}(y_f + z_f + s)$ to y'_f; all remaining coordinates in y' will be assigned to 0. We present an informal version of our algorithm in Algorithm 13.2, and defer its formal version to the appendix.

Analysis. Now we analyze the above approach. First, we have the guarantee that $\|\widehat{x} - y'\|_\infty \leq \nu$. Moreover, note that by our choice of s, for every $f \in \mathrm{supp}(y+z)$, $y_f + z_f + s$ and $\widehat{x}_f + s$ round to the same grid point in $\mathcal{G}_{0.04\nu}$. Therefore, for the new vector y' we

have recovered, we "hide" the randomness in $\mathcal{T}$, and the randomness only leaks from failed attempts of the shifts. In the following, we show that each attempt of shift succeeds with probability $\frac{1}{2}$.

We can restate the procedure of choosing s to be:

repeatedly sample $s \sim \mathcal{B}_\infty(0, 10^{-3}\nu)$,

until for all $f \in \mathrm{supp}(y+z)$, $\left|\Pi_{0.04\nu}\left(\mathcal{B}_\infty\left(y_f + z_f + s, \frac{\nu}{2^{20}k}\right)\right)\right| = 1$.

Note that $|\mathrm{supp}(y+z)| = O(k)$. Let us say that we can always guarantee that $|\mathrm{supp}(y+z)| \le 50k$. By Lemma 13.1.3 where we let $r_b = \frac{\nu}{2^{20}k}$, $r_s = 10^{-3}\nu$ and $r_g = 0.04\nu$, for $f \in \mathrm{supp}(y+z)$,

$$\Pr\left[\left|\Pi_{0.04\nu}\left(\mathcal{B}_\infty\left(y_f + z_f + s, \frac{\nu}{2^{20}k}\right)\right)\right| = 1\right] \ge (1 - \frac{r_b}{r_s})^2 \ge 1 - \frac{1}{100k}.$$

By a union bound over $f \in \mathrm{supp}(y+z)$, the probability is at least $\frac{1}{2}$ that for all $f \in \mathrm{supp}(y+z)$, $|\Pi_{0.04\nu}(\mathcal{B}_\infty(y_f + z_f + s, \frac{\nu}{2^{20}k}))| = 1$.

Therefore, with probability $1 - 1/\mathrm{poly}(n)$, we will only try $O(\log n)$ shifts. We can apply a union bound over $O(\log n)$ possible shifts, and prove that with probability $1 - 1/\mathrm{poly}(n)$ if taking y' as our new y, and shrinking ν by half, the LinfinityReduce procedure will work as desired as if there is no dependence issue.

Sufficient event. Let S be top $O(k)$ coordinates in $\widehat{x}$. Let $L = O(\log R^*)$ denote the number of iterations in Algorithm 13.2. For $\ell \in [L]$, let $\nu_\ell = 2^{-\ell}\mu R^*$. For $\ell \in [L-1]$, let $s_\ell^{(a)}$ be the a-th uniform randomly sampled from $\mathcal{B}_\infty(0, 10^{-3}\nu_\ell)$ as appeared on Line 10 in Algorithm 13.2. For the sake of analysis, we assume that Algorithm 13.2 actually produces an infinite sequence of shifts $s_\ell^{(1)}, s_\ell^{(2)}, \ldots$. We formulate the event that guarantees successful

execution of Algorithm 13.2. We define event $\mathcal{E}_2$ to be all of the following events hold.

1. For all $\ell \in [L-1]$, there exists $a \in [10 \log n]$ so that for all $f \in S$,

$$\left| \Pi_{0.04\nu_\ell} \left(\mathcal{B}_\infty(\widehat{x}_f + s_\ell^{(a)}, \frac{1}{100k}\nu_\ell) \right) \right| = 1.$$

2. For $\ell = 1$, if we run the LinfinityReduce procedure H times with $y = \vec{0}$ and measurements in $\mathcal{T}$, we get z such that $\|\widehat{x} - z\|_\infty \leq 2^{1-H}\nu_1$ and $\text{supp}(z) \subseteq S$.

3. For all $\ell \in \{2, \ldots, L\}$, for all $a \in [10 \log n]$, if we run the LinfinityReduce procedure H times with $y = \xi$ where

$$\xi_f = \begin{cases} \Pi_{0.04\nu_\ell}(\widehat{x}_f + s_{\ell-1}^{(a)}), & \text{if } f \in S; \\ 0, & \text{if } f \in \overline{S}. \end{cases}$$

then we get z such that $\|\widehat{x} - y - z\|_\infty \leq 2^{1-H}\nu_\ell$ and $\text{supp}(y + z) \subseteq S$.

We can prove that event $\mathcal{E}_2$ happens with probability $1 - 1/\text{poly}(n)$. Moreover, we can prove that conditioned on event $\mathcal{E}_2$ Algorithm 13.2 gives correct output. We defer both proofs in the appendix.

13.2 Algorithm for d-dimensional Sparse Fourier Transform

In this section, we will give a Fourier sparse recovery algorithm that takes $O(k \log k \log n)$ measurements with "ℓ_∞/ℓ_2" guarantee. We assume the knowledge of $\mu = \frac{1}{\sqrt{k}}\|\widehat{x}_{-k}\|_2$. In fact, a constant factor approximation suffices, but we prefer to assume exact knowledge of it in order to simplify exposition. All of the arguments go through in the other case, with minor changes in constants. We also assume we know R^* so that $R^* \geq \|\widehat{x}\|_\infty / \mu$. We assume that $\log R^* = O(\log n)$. For larger $\log R^* = O(\mathrm{poly}(n))$, our algorithm will still work, but the decoding time will be worse by a factor of $\frac{\log R^*}{\log n}$. Note that our assumptions on μ and R^* are standard. For example, [IK14] make the same assumption. We assume that we can measure the signal x in the time domain, and we want to recover the signal $\widehat{x}$ in the frequency domain.

In our algorithm, we will use μ as a threshold for noise, and we will perform $\log R^*$ iterations, where in each iteration the upper bound of ℓ_∞ norm of the residual signal (in the frequency domain) shrinks by half. In Section 13.2.1, we give some definitions that will be used in the algorithm. Then we present our new algorithm for d-dimension Fourier sparse recovery in Section 13.2.2. In Section 13.2.3, we prove the correctness of the proposed algorithm.

13.2.1 Notations

For a subset of samples (or measurements) $\{x_t\}_{t \in T}$ from the time domain, where T is a list of elements in $[p]^d$, we define $\widehat{x}^{[T]}$ in Definition 13.2.1 as our estimation to $\widehat{x}$.

Definition 13.2.1 (Fourier transform of a subset of samples)**.** Let $x \in \mathbb{C}^{[p]^d}$. For any T

which is a list of elements in $[p]^d$, for any $f \in [p]^d$, we define

$$\widehat{x}_f^{[T]} = \frac{\sqrt{n}}{|T|} \sum_{t \in T} \omega^{f^\top t} x_t.$$

In order to reuse samples across different iterations where we drive down the upper bound of the residual signal by half, in each iteration after we obtain estimations to heavy hitters (or equivalently large coordinates), instead of subtracting the estimates directly, we need to "hide" the randomness leaked by the samples. We interpret each estimate (which is a complex number) as a point on a 2-dimension plane, and hide the randomness by rounding the estimate to the nearest grid point (where the side length of the grid is chosen to be a small constant fraction of the target ℓ_∞ norm of the residual signal in the frequency domain), which we call "projection onto grid". In Definition 13.2.2, we formally define box and grid, and in Definition 13.2.3 we define projection to grid. We illustrate these two definitions in Figure 13.1.

Definition 13.2.2 (box and grid)**.** For any $c \in \mathbb{C}$ and $r \geq 0$, we define box $\mathcal{B}_\infty(c, r) \subseteq \mathbb{C}$ as

$$\mathcal{B}_\infty(c, r) = \{c + x + y\mathbf{i} : x, y \in [-r, r]\}.$$

Namely, if we consider complex numbers as points on 2D plane, box $\mathcal{B}_\infty(c, r)$ refers to ℓ_∞ ball with radius r centered at c.

For any $r > 0$, we define grid $\mathcal{G}_r \subseteq \mathbb{C}$ as

$$\mathcal{G}_r = \{xr + yr\mathbf{i} : x, y \in \mathbb{Z}\}.$$

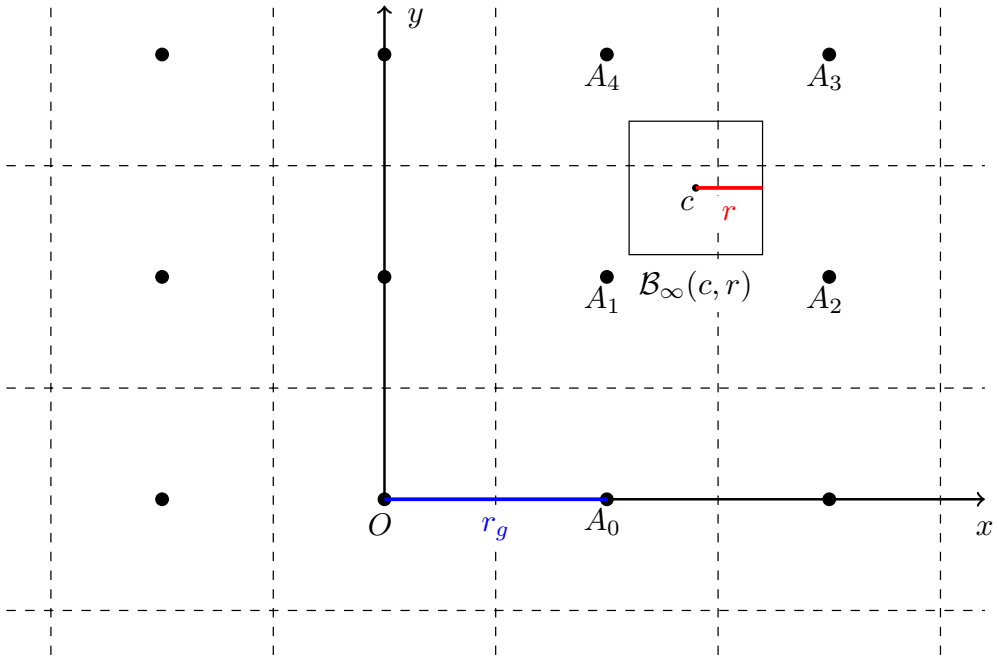


Figure 13.1: Illustration of box $\mathcal{B}_\infty(c,r)$ and grid $\mathcal{G}_{r_g}$. Box $\mathcal{B}_\infty(c,r)$ refers to all the points in the square centered at c with side length $2r$. Grid $\mathcal{G}_{r_g}$ refers to all the solid round points, and the distance between origin O and A_0 is r_g. Note that the dashed lines are decision boundaries of the projection Π_{r_g}, and all the points inside a minimum cell separated by the dashed lines are mapped (by Π_{r_g}) to the same grid point in $\mathcal{G}_{r_g}$ (which is the center of the cell). We have $\Pi_{r_g}(c) = A_1$ and $\Pi_{r_g}(\mathcal{B}_\infty(c,r)) = \{A_1, A_2, A_3, A_4\}$.

Definition 13.2.3 (projection onto grid)**.** For any $r > 0$, we define Π_r to be a maping from $\mathbb{C}$ to $\mathcal{G}_r$, so that for any $c \in \mathbb{C}$,

$$\Pi_r(c) = \arg\min_{c' \in \mathcal{G}_r} |c - c'|,$$

where we break the tie by choosing the one with minimum $|c'|$. As a natural generalization, For $C \subseteq \mathbb{C}$, we define

$$\Pi_r(C) = \{\Pi_r(c) : c \in C\}.$$

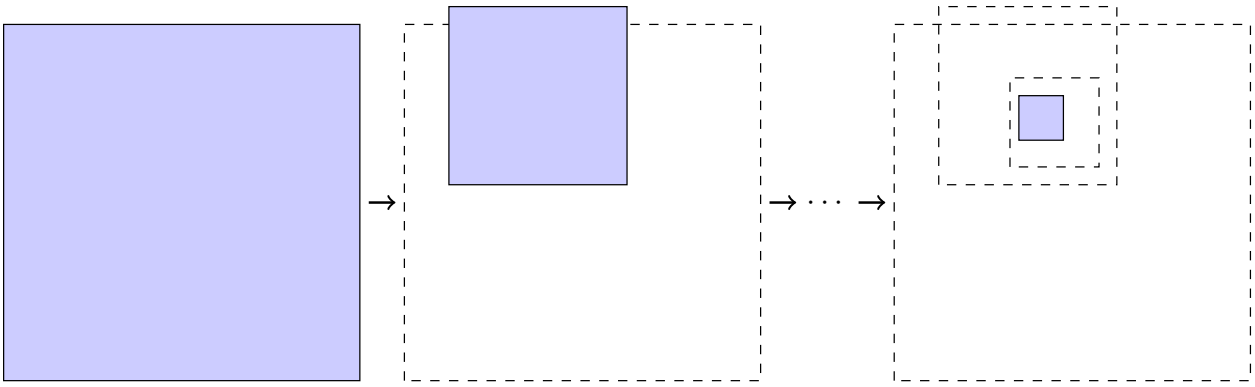
13.2.2 Algorithm

We present our new sparse Fourier recovery algorithm in Algorithm 13.3. Its auxiliary function LINFINITYREDUCE is in Algorithm 13.4. Important constants are summarized in Table 13.2.

In Algorithm 13.3, we define "bucket size" $B = O(k)$ and number of repetitions $R = O(\log n)$. For each $r \in [R]$, we choose $\mathcal{T}_r$ to be a list of B independent and uniformly random elements in $[p]^d$. We will measure x_t for all $t \in \cup_{r\in[R]}\mathcal{T}_r$, and use LINFINITYREDUCE in Algorithm 13.4 to locate and estimate all the "heavy hitters" of the residual signal so that if we substract them then the ℓ_∞ norm of the new residual signal shrinks by half. The input to LINFINITYREDUCE is a signal $x \in \mathbb{C}^{[p]^d}$ in the time domain (but we can only get access to x_t where $t \in \cup_{r\in[R]}\mathcal{T}_r$), a sparse vector $y \in \mathbb{C}^{[p]^d}$ in the frequency domain that we have recovered so far, and $\nu \geq \mu$ such that $\|\widehat{x}-y\|_\infty \leq 2\nu$ where we will refer $\widehat{x}-y$ as the currect residual signal (in the frequency domain). It is guaranteed that LINFINITYREDUCE$(x, n, y, \{\mathcal{T}_r\}_{r=1}^R, \nu)$ returns a $O(k)$-sparse z so that $\|\widehat{x} - y - z\| \leq \nu$ with probability $1 - 1/\operatorname{poly}(n)$.

Algorithm 13.3 in total maintains $H = O(\log k)$ independent copies of such error-reduce data structures, where in the h-th copy it measures $\mathcal{T}^{(h)} = \{\mathcal{T}_r^{(h)}\}_{r\in[R]}$ for $h \in [H]$. We denote $\mathcal{T} = \{\mathcal{T}^{(h)}\}_{r\in[R]}$. If $\log R^* \leq H$, then we can simply use different $\mathcal{T}^{(h)}$ in different iterations. In that case $L = 1$ and $H = \log R^*$ in Algorithm 13.3. We will get $z^{(1)}$ on Line 20 such that $\|\widehat{x} - y^{(0)} - z^{(1)}\|_\infty \leq \mu$ (we will prove in the analysis this holds with probability $1 - 1/\operatorname{poly}(n)$) where $y^{(0)} = 0$, and return $z^{(1)} + y^{(0)}$ on Line 22.

If $\log R^* > H$, we have to reuse the samples. We proceed in L iterations (in the loop between Line 14 and Line 33 in Algorithm 13.3), where $L = \log R^* - H + 1$. For $\ell \in [L]$,

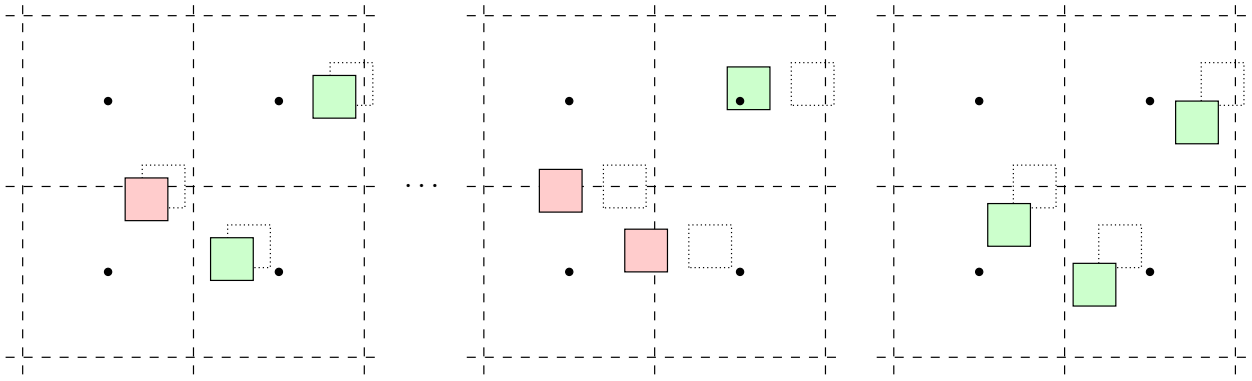


(a) $h = 0$ (b) $h = 1$ (c) $h = H$

Figure 13.2: Illustration of the behavior of Line 16 to Line 20 in Algorithm 13.3. For any $f \in [p]^d$, we draw box $\mathcal{B}_\infty(y_f^{(\ell-1)} + z_f, 2^{1-h}\nu_\ell)$ after h iterations of the for loop between Line 17 and Line 19 in Algorithm 13.3, where $h \in \{0, 1, \ldots, H\}$. Conditioned on LINFINITYREDUCE is correct, for every $h \in \{0, 1, \ldots, H\}$, after h-th iteration we have $\widehat{x}_f \in \mathcal{B}_\infty(y_f^{(\ell-1)} + z_f, 2^{1-h}\nu_\ell)$. When $h = 0$, i.e. before the loop between Line 17 and Line 19 starts, we know that $\widehat{x}_f \in \mathcal{B}_\infty(y_f^{(\ell-1)}, 2\nu_\ell)$ as depicted by (a). After each iteration in h, the radius of the box shrinks by half (and its center might change). Finally after H iterations, as depicted by (c), we obtain $z^{(\ell-1)}$ such that $\widehat{x}_f \in \mathcal{B}_\infty(y_f^{(\ell-1)} + z_f^{(\ell)}, 2^{1-H}\nu_\ell)$.

as defined in Line 15, $\nu_\ell = 2^{-\ell}\mu R^*$ refers to the target ℓ_∞ of the residual signal in the ℓ-th iteration (namely, for $\ell \in [L-1]$ we want to obtain $y^{(\ell)}$ so that $\|\widehat{x} - y^{(\ell)}\|_\infty \le \nu_\ell$). In the ℓ-th iteration where $\ell \in [L]$, by using the samples in $\mathcal{T} = \{\mathcal{T}^{(h)}\}_{h \in H}$ (Line 16 to Line 20), the algorithm tries to get $z^{(\ell)}$ so that $\|\widehat{x} - y^{(\ell-1)} - z^{(\ell)}\|_\infty \le 2^{1-H}\nu_\ell$. The intuition on the behavior of Line 16 to Line 20 is depicted in Figure 13.2.

If $\ell = L$ the algorithm will return $y^{(L-1)} + z^{(L)}$ as in Line 22; otherwise, the algorithm will try to compute $y^{(\ell)}$ based on $y^{(\ell-1)} + z^{(\ell)}$. In Line 25 to Line 28, the algorithm repeatedly samples a uniform random shift $s_\ell \in \mathcal{B}_\infty(0, \alpha\nu_\ell)$ (where $\alpha \in (0, 1)$ is a small constant chosen in Table 13.2) until the shift is good, where shift s_ℓ is good if and only if for each $f \in \mathrm{supp}(y^{(\ell-1)} + z^{(\ell)})$, all the points in $\mathcal{B}_\infty(y^{(\ell-1)} + z^{(\ell)} + s_\ell, 2^{1-H}\nu_\ell)$ (i.e. the box obtained by



(a) a failed attempt (b) another failed attempt (c) a successful attempt

Figure 13.3: Illustration of the iteration between Line 25 and Line 28 in Algorithm 13.3. The round solid points represent grid points in $\mathcal{G}_{\beta\nu}$, and the dashed lines represent decision boundaries of $\Pi_{\beta\nu_\ell}$. In this example we have $|\operatorname{supp}(y^{(\ell-1)}+z^{(\ell)})| = 3$, and the dotted squares represent boxes $\mathcal{B}_\infty(y_f^{(\ell-1)}+z_f^{(\ell)}, 2^{1-H}\nu_\ell)$ for $f \in \operatorname{supp}(y^{(\ell-1)}+z^{(\ell)})$. The algorithm repeatedly samples a random shift $s \sim \mathcal{B}_\infty(0, \alpha\nu_\ell)$, until all the shifted boxes $\{\mathcal{B}_\infty(y_f^{(\ell-1)}+z_f^{(\ell)}, 2^{1-H}\nu_\ell)+s\}_{f\in\operatorname{supp}(y^{(\ell-1)}+z^{(\ell)})}$ do not intersect with the dashed lines (i.e. decision boundaries of $\Pi_{\beta\nu_\ell}$). In the figure, we color a shifted box in green if it does not intersect with dashed lines, and color in red otherwise. After a series of failed attempts from (a) to (b), we finally have a successful attempt in (c).

applying shift s_ℓ to the box $\mathcal{B}_\infty(y^{(\ell-1)}+z^{(\ell)}, 2^{1-H}\nu_\ell))$ project to the same grid point in $\mathcal{G}_{\beta\nu_\ell}$. We depict the process of obtaining the shift s_ℓ in Figure 13.3. It is crucial to note that if the shift s_ℓ is good and the vector $z^{(\ell)}$ we get is desirable (namely $\|\widehat{x}-y^{(\ell-1)}-z^{(\ell)}\|_\infty \le 2^{1-H}\nu_\ell$), then for each $f \in \operatorname{supp}(y^{(\ell-1)}+z^{(\ell)})$, $\Pi_{\beta\nu_\ell}(y_f^{(\ell-1)}+z_f^{(\ell)}+s_\ell) = \Pi_{\beta\nu_\ell}(\widehat{x}_f+s_\ell)$.

On Line 31, we assign $\Pi_{\beta\nu_\ell}(y_f^{(\ell-1)}+z_f^{(\ell)}+s_\ell)$ to $y_f^{(\ell)}$. Because β is a small constant, we still have the guarantee that $\|\widehat{x}-y^{(\ell)}\|_\infty \le \nu_\ell$. Moreover, by assigning $\Pi_{\beta\nu_\ell}(y_f^{(\ell-1)}+z_f^{(\ell)}+s_\ell) = \Pi_{\beta\nu_\ell}(\widehat{x}_f+s_\ell)$ to $y_f^{(\ell)}$, we "hide" the randomness in $\mathcal{T} = \{\mathcal{T}^{(h)}\}_{h\in[H]}$. Now the randomness in $\mathcal{T}$ only leaks from failed attempts of the shifts. For analysis purpose, we maintain a counter a_ℓ for $\ell \in [L-1]$ recording the number of attempts until we have sampled a good

one. By our choice of parameters, we can prove that with high probability $a_\ell \le 10 \log n$ for each $\ell \in [L-1]$. Thus intuitively the leaked randomness is under control, and we can formally apply a union bound to prove that with good probability all possible invocations of LinfinityReduce by our FourierSparseRecovery produce desirable output.

13.2.3 Analysis

In order to analyze the algorithm, let $S \subseteq [n]$ be top $C_S k$ coordinates of $\widehat{x}$ where $C_S = 26$, and let $\overline{S} = [n] \setminus S$. In order to analyze the performance of LinfinityReduce in Algorithm 13.4, we need the following definition.

Definition 13.2.4 (uniform sample)**.** We say t is sampled from $[p]^d$ uniformly at random if for each $i \in [d]$, we independently sample t_i from $[p]$ uniformly at random. We use $t \sim [p]^d$ to denote it.

Fact 13.2.1. *Let $\omega = e^{2\pi \mathbf{i}/p}$ where p is any positive integer. For a fixed $f \in [p]^d \setminus \{\vec{0}\}$, $\mathbb{E}_{t \sim [p]^d}[\omega^{f^\top t}] = 0$.*

Proof. Note that $\mathbb{E}_{t \sim [p]^d}[\omega^{f^\top t}] = \prod_{i \in [d]} \mathbb{E}_{t_i \sim [p]}[\omega^{f_i t_i}]$ by the fact that $t_1, \ldots, t_d$ are independent. Because $f \neq \vec{0}$, there exists $i \in [d]$ so that $f_i \neq 0$. We have

$$\begin{aligned} \mathop{\mathbb{E}}_{t_i \sim [p]}[\omega^{f_i t_i}] &= \frac{1}{p} \sum_{j=0}^{p-1} (\omega^{f_i})^j \\ &= \frac{1}{p} \cdot \frac{(\omega^{f_i})^0 (1 - (\omega^{f_i})^p)}{1 - \omega^{f_i}} \\ &= 0, \end{aligned}$$

where the second step follows from the sum of geometry series where $\omega^{f_i} \neq 1$, adn the third step follow from $(\omega^{f_i})^p = e^{2\pi \mathbf{i} f_i} = 1$. Therefore, $\mathbb{E}_{t \sim [p]^d}[\omega^{f^\top t}] = 0$.

Algorithm 13.3 Fourier Sparse Recovery by Random Shift and Projection

procedure FourierSparseRecovery(x, n, k, μ, R^*) ▷ Theorem 13.2.9, $n = p^d$
  Require that $\mu = \frac{1}{\sqrt{k}}\|\widehat{x}_{-k}\|_2$ and $R^* \geq \|\widehat{x}\|_\infty / \mu$ ▷ R^* is a power of 2
  $B \leftarrow C_B \cdot k$ ▷ C_B is a constant defined in Table 13.2
  $R \leftarrow C_R \cdot \log n$ ▷ C_R is a constant defined in Table 13.2
  $H \leftarrow \min\{\log k + C_H, \log R^*\}$ ▷ C_H is a constant defined in Table 13.2
  for $h = 1 \to H$ **do**
    for $r = 1 \to R$ **do**
      $\mathcal{T}_r^{(h)} \leftarrow$a list of B i.i.d elements in $[p]^d$
    end for
    $\mathcal{T}^{(h)} \leftarrow \{\mathcal{T}_r^{(h)}\}_{r=1}^R$
  end for ▷ We will measure x_t for $t \in \cup_{h\in[H],r\in[R]}\mathcal{T}_r^{(h)}$
  $y^{(0)} \leftarrow \vec{0}$ ▷ $y^{(0)} \in \mathbb{C}^n$
  $L \leftarrow \log R^* - H + 1$
  for $\ell = 1 \to L$ **do**
    $\nu_\ell \leftarrow 2^{-\ell}\mu R^*$ ▷ Target ℓ_∞ of the residual signal in iteration t
    $z \leftarrow \vec{0}$ ▷ z is a temporary variable used to compute $z^{(\ell)}$
    for $h = 1 \to H$ **do**
      $z \leftarrow z +$ LinfinityReduce$(x, n, y^{(\ell-1)} + z, \mathcal{T}^{(h)}, 2^{1-h}\nu_\ell)$
    end for
    $z^{(\ell)} \leftarrow z$ ▷ We want $\|\widehat{x} - y^{(\ell-1)} - z^{(\ell)}\|_\infty \leq 2^{1-H}\nu_\ell$
    if $\ell = L$ **then**
      return $y^{(L-1)} + z^{(L)}$
    end if
    $a \leftarrow 0$ ▷ A temporary counter maintained for analysis purpose only
    repeat
      Pick $s_\ell \in \mathcal{B}_\infty(0, \alpha\nu_\ell)$ uniformly at random ▷ $\alpha \in (0,1)$ is a small constant
      $a \leftarrow a + 1$ ▷ β in the next line is a small constant where $\alpha < \beta < 0.1$
    until $\forall f \in \mathrm{supp}(y^{(\ell-1)} + z^{(\ell)}), |\Pi_{\beta\nu_\ell}(\mathcal{B}_\infty(y_f^{(\ell-1)} + z_f^{(\ell)} + s_\ell, 2^{1-H}\nu_\ell))| = 1$
    $a_\ell \leftarrow a$
    for $f \in \mathrm{supp}(y^{(\ell-1)} + z^{(\ell)})$ **do**
      $y_f^{(\ell)} \leftarrow \Pi_{\beta\nu_\ell}(y_f^{(\ell-1)} + z_f^{(\ell)} + s_\ell)$ ▷ We want $\|\widehat{x} - y^{(\ell)}\|_\infty \leq \nu_\ell$
    end for
  end for
end procedure

□

We define measurement coefficient as follows:

Definition 13.2.5 (measurement coefficient)**.** For any $f \in [p]^d$ and any T which is a list of elements in $[p]^d$, we define

$$c_f^{[T]} = \frac{1}{|T|} \sum_{t \in T} \omega^{f^\top t}.$$

By definition of $c_f^{[T]}$ and d-dimension Fourier transform, we can decompose $\widehat{x}_f^{[T]}$ as follows.

Lemma 13.2.2 (measurement decomposition)**.** *For any $f \in [p]^d$ and any T which is a list of elements in $[p]^d$,*

$$\widehat{x}_f^{[T]} = \sum_{f' \in [p]^d} c_{f-f'}^{[T]} \widehat{x}_{f'}.$$

Proof. We have

$$\begin{aligned}
\widehat{x}_f^{[T]} &= \frac{\sqrt{n}}{|T|} \sum_{t \in T} \omega^{f^\top t} x_t \\
&= \frac{\sqrt{n}}{|T|} \sum_{t \in T} \omega^{f^\top t} \frac{1}{\sqrt{n}} \sum_{f' \in [p]^d} \omega^{-f'^\top t} \widehat{x}_{f'} \\
&= \frac{\sqrt{n}}{|T|} \sum_{t \in T} \frac{1}{\sqrt{n}} \sum_{f' \in [p]^d} \omega^{(f-f')^\top t} \widehat{x}_{f'} \\
&= \sum_{f' \in [p]^d} \left(\frac{1}{|T|} \sum_{t \in T} \omega^{(f-f')^\top t} \right) \widehat{x}_{f'} \\
&= \sum_{f' \in [p]^d} c_{f-f'}^{[T]} \widehat{x}_{f'},
\end{aligned}$$

Notation	Choice	Statement	Parameter
C_B	10^6	Lemma 13.2.8	B
C_R	10^3	Lemma 13.2.7	R
C_H	20	Algorithm 13.3	H
α	10^{-3}	Algorithm 13.3 Line 26	shift range
β	0.04	Algorithm 13.3 Line 28	grid size
C_S	26	Lemma 13.2.7, Lemma 13.2.8	$\|S\|$

Table 13.2: Summary of important constants.

Lemma	Meaning
Lemma 13.2.2	measurement decomposition
Lemma 13.2.3	properties of coefficient
Lemma 13.2.4	noise bound
Lemma 13.2.5	guarantee of LINFINITYREDUCE
Lemma 13.2.6	property of a randomly shifted box
Lemma 13.2.7	event $\mathcal{E}$ happens
Lemma 13.2.8	correctness of our algorithm

Table 13.3: Summary of Lemmas.

where the first step follow by the definition of $\widehat{x}_f^{[T]}$ in Definition 13.2.1, second step follows by the definition of inverse d-dimensional Fourier transform (see Section 13.1.1), third and forth step follow by rearranging terms, last step follows by the definition of measurement coefficients c in Definition 13.2.5. □

Let T be a list of i.i.d. samples from $[p]^d$, then the coeffcients $c_f^{[T]}$ defined in Definition 13.2.5 have the following property.

Lemma 13.2.3 (properties of coeffcient c)**.** *Let T be a list of B independent and uniform random elements in $[p]^d$. Then we have*

1. $c_0^{[T]} = 1$.

2. *For any* $f \in [p]^d \setminus \{0\}$, $\mathbb{E}_T\left[|c_f^{[T]}|^2\right] = \frac{1}{B}$.
3. *For any* $f, f' \in [p]^d$, $f \neq f'$, $\mathbb{E}_T\left[c_f^{[T]} \cdot \overline{c_{f'}^{[T]}}\right] = 0$.

Proof. **Part 1.** By definition of $c_0^{[T]}$,

$$c_0^{[T]} = \frac{1}{|T|}\sum_{t\in T}\omega^{0\cdot t} = 1.$$

Part 2. Let $T = \{t_1, \ldots, t_B\}$, where t_i is independently and uniformly chosen from $[p]^d$. For any $f \in [p]^d \setminus \{0\}$,

$$\begin{aligned}
\mathop{\mathbb{E}}_T\left[|c_f^{[T]}|^2\right] &= \mathop{\mathbb{E}}_T\left[c_f^{[T]} \cdot \overline{c_f^{[T]}}\right] \\
&= \frac{1}{|T|^2}\mathop{\mathbb{E}}_T\left[\sum_{i,j\in[B]}\omega^{f^\top(t_i-t_j)}\right] \\
&= \frac{1}{|T|^2}\left(|T| + \mathop{\mathbb{E}}_T\left[\sum_{i,j\in[B],i\neq j}\omega^{f^\top(t_i-t_j)}\right]\right) \\
&= \frac{1}{|T|} + \frac{1}{|T|^2}\sum_{i,j\in[B],i\neq j}\mathop{\mathbb{E}}_T\left[\omega^{f^\top(t_i-t_j)}\right] \\
&= \frac{1}{|T|} - \frac{1}{|T|^2}\cdot 0 \\
&= \frac{1}{|T|} = \frac{1}{B},
\end{aligned}$$

where the forth step follows by $\mathbb{E}_T[\omega^{f^\top(t_i-t_j)}] = \mathbb{E}_{t\sim[p]^d}[\omega^{f^\top t}] = 0$, in which $\mathbb{E}_T[\omega^{f^\top(t_i-t_j)}] = \mathbb{E}_{t\sim[p]^d}[\omega^{f^\top t}]$ because $i \neq j$, t_i, t_j are independent and uniformly random distributed in $[p]^d$, $t_i - t_j \sim [p]^d$; $\mathbb{E}_{t\sim[p]^d}[\omega^{f^\top t}] = 0$ follows by by Fact 13.2.1 and f is not a zero vector.

Part 3. For any $f, f' \in [p]^d$, $f \neq f'$,

$$\begin{aligned}
\mathop{\mathbb{E}}_{T}\left[c_f^{[T]} \cdot \overline{c_{f'}^{[T]}}\right] &= \frac{1}{|T|^2} \mathop{\mathbb{E}}_{T}\left[\sum_{i,j\in[B]} \omega^{f^\top t_i - f'^\top t_j}\right] \\
&= \frac{1}{|T|^2}\left(\sum_{i,j\in[B],i\neq j} \mathop{\mathbb{E}}_{T}\left[\omega^{f^\top t_i - f'^\top t_j}\right] + \sum_{i\in[B]} \mathop{\mathbb{E}}_{T}\left[\omega^{(f-f')^\top t_i}\right]\right) \\
&= \frac{1}{|T|^2}\left(\sum_{i,j\in[B],i\neq j} \mathop{\mathbb{E}}_{t_i\sim[p]^d}\left[\omega^{f^\top t_i}\right] \mathop{\mathbb{E}}_{t_j\sim[p]^d}\left[\omega^{-f'^\top t_j}\right] + \sum_{i\in[B]} \mathop{\mathbb{E}}_{t_i\sim[p]^d}\left[\omega^{(f-f')^\top t_i}\right]\right) \\
&= 0,
\end{aligned}$$

where the second step follows from separating diagonal term and off-diagonal terms, the third step follows from t_i and t_j are independent, the last step follows from Fact 13.2.1 where $f - f' \neq \vec{0}$, and at least one of f and f' is not $\vec{0}$.

□

Let T be a list of independent and uniformly random elements from $[p]^d$. We are going to measure x_t for $t \in T$, and take $\widehat{x}_f^{[T]}$ (recall its definition in Definition 13.2.1) as estimate to $\widehat{x}_f$. By Lemma 13.2.2, $\widehat{x}_f^{[T]} = \sum_{f'\in[p]^d} c_{f-f'}^{[T]} \widehat{x}_{f'}$. The following lemma bounds the contribution of coordinates from V where $V \subseteq [p]^d \setminus \{f\}$, namely $|\sum_{f'\in V} c_{f-f'}^{[T]} \widehat{x}_{f'}|$. When analyzing the quality of $\widehat{x}_f^{[T]}$ as an approximation to $\widehat{x}_f$, we consider coordinates in V as noise, and we usually set $V = [p]^d \setminus \{f\}$.

Lemma 13.2.4 (noise bound)**.** *For any $f \in [p]^d$, T which is a list of B i.i.d. samples from $[p]^d$ and $V \subseteq [n]$ such that $f \notin V$,*

$$\Pr_{T}\left[\left|\sum_{f'\in V} c_{f-f'}^{[T]} \widehat{x}_{f'}\right| \geq \frac{10}{\sqrt{B}} \|\widehat{x}_V\|_2\right] \leq \frac{1}{100}.$$

Proof. First, we can prove that $\mathbb{E}_T\left[\left|\sum_{f'\in V} c^{[T]}_{f-f'}\widehat{x}_{f'}\right|^2\right] = \frac{1}{B}\|\widehat{x}_V\|_2^2$, because

$$\begin{aligned}
\mathop{\mathbb{E}}_{T}\left[\left|\sum_{f'\in V} c^{[T]}_{f-f'}\widehat{x}_{f'}\right|^2\right] &= \mathop{\mathbb{E}}_{T}\left[\sum_{f_1,f_2\in V}(c^{[T]}_{f-f_1}\widehat{x}_{f_1})\overline{(c^{[T]}_{f-f_2}\widehat{x}_{f_2})}\right] \\
&= \sum_{f_1,f_2\in V}\mathop{\mathbb{E}}_{T}\left[c^{[T]}_{f-f_1}\overline{c^{[T]}_{f-f_2}}\right]\widehat{x}_{f_1}\overline{\widehat{x}_{f_2}} \\
&= \sum_{f'\in V}\mathop{\mathbb{E}}_{T}\left[\left|c^{[T]}_{f-f'}\right|^2\right]|\widehat{x}_{f'}|^2 \\
&= \frac{1}{B}\|\widehat{x}_V\|_2^2,
\end{aligned}$$

where the third step follows from Lemma 13.2.3 that for $f-f_1 \neq f-f_2$, $\mathbb{E}_T\left[c^{[T]}_{f-f_1}\overline{c^{[T]}_{f-f_2}}\right] = 0$, and the last step follows from $\mathbb{E}_T\left[\left|c^{[T]}_{f-f'}\right|^2\right] = 1/B$ in Lemma 13.2.3.

Then the lemma follows from Chebyshev Inequality and the fact that

$$\mathop{\mathbb{V}}_{T}\left[\left|\sum_{f'\in V} c^{[T]}_{f-f'}\widehat{x}_{f'}\right|\right] \leq \mathop{\mathbb{E}}_{T}\left[\left|\sum_{f'\in V} c^{[T]}_{f-f'}\widehat{x}_{f'}\right|^2\right] = \frac{1}{B}\|\widehat{x}_V\|_2^2.$$

□

In the next lemma, we show the guarantee of LinfinityReduce in Algorithm 13.4.

Lemma 13.2.5 (guarantee of LinfinityReduce in Algorithm 13.4)**.** *Let $x \in \mathbb{C}^{[p]^d}$, and $n = p^d$. Let $R = C_R \log n$, and $B = C_B k$. Let $C_B \geq 10^6$ and $C_R \geq 10^3$. Let $\mu = \frac{1}{\sqrt{k}}\|\widehat{x}_{-k}\|_2$, and $\nu \geq \mu$. For $r \in [R]$, let $\mathcal{T}_r$ be a list of B i.i.d. elements in $[p]^d$. Let $z \in \mathbb{C}^n$ denote the output of*

$$\textsc{LinfinityReduce}(x, n, y, \{\mathcal{T}_r\}_{r=1}^R, \nu).$$

Algorithm 13.4 Procedure for Reducing ℓ_∞-norm of the Residual Signal

1: **procedure** LINFINITYREDUCE($x, n, y, \{\mathcal{T}_r\}_{r=1}^R, \nu$) ▷ Lemma 13.2.5
2: **Require** that $\|\widehat{x} - y\|_\infty \le 2\nu$
3: Let w be inverse Fourier transform of y ▷ We have $\widehat{w} = y$
4: **for** $r = 1 \to R$ **do**
5: **for** $f = 1 \to n$ **do** ▷ Implemented by FFT which takes $O(n \log n)$ time
6: $u_{f,r} \leftarrow \frac{\sqrt{n}}{|\mathcal{T}_r|} \sum_{t \in \mathcal{T}_r} \omega^{f^\top t}(x_t - w_t)$ ▷ $\omega = e^{2\pi \mathbf{i}/p}, u_{f,r} = \widehat{(x-w)}_f^{[\mathcal{T}_r]}$
7: **end for**
8: **end for**
9: **for** $f = 1 \to n$ **do**
10: $\eta = \mathrm{median}_{r \in [R]}\{u_{f,r}\}$ ▷ Take the median coordinate-wise
11: **if** $|\eta| \le \nu/2$ **then**
12: $z_f \leftarrow \eta$
13: **else**
14: $z_f \leftarrow 0$
15: **end if**
16: **end for**
17: **return** z ▷ Guarantee $\|\widehat{x} - y - z\|_\infty \le \nu$
18: **end procedure**

Let S be top $C_S k$ coordinates in $\widehat{x}$, where $C_S = 26$. If $\|\widehat{x} - y\|_\infty \le 2\nu$, $\mathrm{supp}(y) \subseteq S$ and y is independent from the randomness of $\{\mathcal{T}_r\}_{r=1}^R$, then with probability $1 - 1/\mathrm{poly}(n)$ under the randomness of $\{\mathcal{T}_r\}_{r=1}^R$, $\|\widehat{x} - y - z\|_\infty \le \nu$ and $\mathrm{supp}(z) \subseteq S$. Moreover, the running time of LINFINITYREDUCE *is $O(n \log^2 n)$.*

Proof. Note that $\forall f \in \overline{S}$,

$$|\widehat{x}_f| \le \sqrt{\frac{\|\widehat{x}_{-k}\|_2^2}{C_S k - k}} = \frac{1}{5}\mu,$$

where the last step follows from choice of C_S.

Let w denote the inverse Fourier transform of y. Note that on Line 6 in Algo-

rithm 13.4, for any $f \in [p]^d$ and $r \in [R]$,

$$\begin{aligned} u_{f,r} &= \frac{\sqrt{n}}{|\mathcal{T}_r|} \sum_{t \in \mathcal{T}_r} \omega^{f^\top t}(x_t - w_t) \\ &= \widehat{(x-w)}_f^{[\mathcal{T}_r]} \\ &= \sum_{f' \in [n]} c_{f-f'}^{[\mathcal{T}_r]} (\widehat{x}_{f'} - y_{f'}), \end{aligned}$$

where the second step follows by the notation in Definition 13.2.1, and the third step follows by Lemma 13.2.2. Therefore,

$$\widehat{x}_f - y_f = u_{f,r} - \sum_{f' \in [p]^d \setminus \{f\}} c_{f-f'}^{[\mathcal{T}_r]} (\widehat{x}_{f'} - y_{f'}), \tag{13.1}$$

By Lemma 13.2.4,

$$\Pr_{\mathcal{T}_r} \left[\left| \sum_{f' \in [p]^d \setminus \{f\}} c_{f-f'}^{[\mathcal{T}_r]} (\widehat{x}_{f'} - y_{f'}) \right| \geq \frac{10}{\sqrt{B}} \|(\widehat{x} - y)_{[p]^d \setminus \{f\}}\|_2 \right] \leq \frac{1}{100}. \tag{13.2}$$

We have

$$\begin{aligned} \frac{10}{\sqrt{B}} \|(\widehat{x} - y)_{[p]^d \setminus \{f\}}\|_2 &\leq \frac{10}{\sqrt{B}} \left(\|(\widehat{x} - y)_{S \setminus \{f\}}\|_2 + \|(\widehat{x} - y)_{\overline{S} \setminus \{f\}}\|_2 \right) \\ &\leq \frac{10}{\sqrt{B}} \left(\|\widehat{x} - y\|_\infty \cdot \sqrt{|S|} + \|\widehat{x}_{\overline{S} \setminus \{f\}}\|_2 \right) \\ &\leq \frac{10}{\sqrt{B}} \left(2\nu \cdot \sqrt{26k} + \sqrt{k}\mu \right) \\ &\leq \frac{1}{100\sqrt{k}} \left(2\nu \cdot \sqrt{26k} + \sqrt{k}\mu \right) \\ &< 0.12\nu, \end{aligned} \tag{13.3}$$

where the first step following by triangle inequality, the second step follows by the assumption that $\mathrm{supp}(y) \subseteq S$, the forth step follows by $C_B \geq 10^6$, the last step follows by $\mu \leq \nu$.

Therefore,

$$\begin{aligned}
\Pr_{\mathcal{T}_r}[|u_{f,r} - (\widehat{x}_f - y_f)| \le 0.12\nu] &= \Pr_{\mathcal{T}_r}\left[\left|\sum_{f' \in [p]^d \setminus \{f\}} c_{f-f'}^{[\mathcal{T}_r]}(\widehat{x}_{f'} - y_{f'})\right| \le 0.12\nu\right] \\
&= 1 - \Pr_{\mathcal{T}_r}\left[\left|\sum_{f' \in [p]^d \setminus \{f\}} c_{f-f'}^{[\mathcal{T}_r]}(\widehat{x}_{f'} - y_{f'})\right| > 0.12\nu\right] \\
&\ge 1 - \Pr_{\mathcal{T}_r}\left[\left|\sum_{f' \in [p]^d \setminus \{f\}} c_{f-f'}^{[\mathcal{T}_r]}(\widehat{x}_{f'} - y_{f'})\right| \ge \frac{10}{\sqrt{B}}\|(\widehat{x} - y)_{[p]^d \setminus \{f\}}\|_2\right] \\
&\ge 1 - \frac{1}{100},
\end{aligned}$$

where the first step follows by (13.1), the third step follows by (13.3), and the last step follows by (13.2).

Thus we have

$$\Pr_{\mathcal{T}_r}[u_{f,r} \in \mathcal{B}_\infty(\widehat{x}_f - y_f, 0.12\nu)] \ge \Pr_{\mathcal{T}_r}[|u_{f,r} - (\widehat{x}_f - y_f)| \le 0.12\nu] \ge 1 - \frac{1}{100}.$$

Let $\eta_f = \text{median}_{r \in [R]} u_{f,r}$ as on Line 10 in Algorithm 13.4. By Chernoff bound, with probability $1 - 1/\operatorname{poly}(n)$, more than $\frac{1}{2}R$ elements in $\{u_{f,r}\}_{r=1}^R$ are contained in box $\mathcal{B}_\infty(\widehat{x}_f - y_f, 0.12\nu)$, so that $\eta_f \in \mathcal{B}_\infty(\widehat{x}_f - y_f, 0.12\nu)$.

Therefore, we have

$$\Pr[|\eta_f - (\widehat{x}_f - y_f)| \le 0.17\nu] \ge \Pr[|\eta_f - (\widehat{x}_f - y_f)| \le \sqrt{2} \cdot 0.12\nu] \ge 1 - 1/\operatorname{poly}(n).$$

Let E be the event that for all $f \in [p]^d$, $|\eta_f - (\widehat{x}_f - y_f)| \le 0.17\nu$. By a union bound over $f \in [p]^d$, event E happens with probability $1 - 1/\operatorname{poly}(n)$. In the rest of the proof, we condition on event E.

(**Case 1**) For $f \in \overline{S}$, note that

$$|\eta_f| \le 0.17\nu + |\widehat{x}_f - y_f| = 0.17\nu + |\widehat{x}_f| \le 0.17\nu + 0.2\nu = 0.37\nu.$$

According to the if statement between Line 11 and Line 15 in Algorithm 13.4, z_f will be assigned to 0. Thus $\mathrm{supp}(z) \subseteq S$. In addition, $|\widehat{x}_f - y_f - z_f| = |\widehat{x}_f| \le \mu \le \nu$.

(**Case 2**) For $f \in S$, we have two cases. We prove that $|(\widehat{x}_f - y_f) - z_f| \le \nu$ for both cases.

(**Case 2.1**) $|\eta_f| \le 0.5\nu$. z_f is assigned as 0. Because

$$|\eta_f - (\widehat{x}_f - y_f)| \le 0.17\nu, \quad |\widehat{x}_f - y_f| \le |\eta_f| + 0.17\nu \le 0.67\nu.$$

Therefore,

$$|(\widehat{x}_f - y_f) - z_f| \le 0.67\nu \le \nu.$$

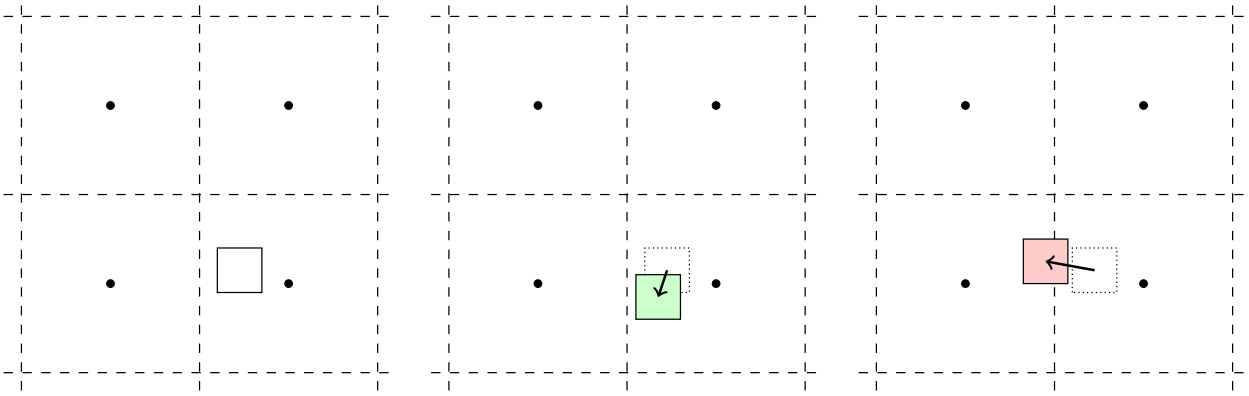
(**Case 2.2**) $|\eta_f| > 0.5\nu$. z_f is assigned as η_f. We have

$$|(\widehat{x}_f - y_f) - z_f| = |(\widehat{x}_f - y_f) - \eta_f| \le 0.17\nu \le \nu.$$

We thus have obtained that with probability $1 - 1/\operatorname{poly}(n)$, $\|(\widehat{x} - y) - z\|_\infty \le \nu$ and $\mathrm{supp}(z) \subseteq S$.

The running time of LinfinityReduce is dominated by the loop between Line 4 and Line 8, which takes $O(R \cdot n \log n) = O(n \log^2 n)$ by FFT. □

For a given box $\mathcal{B}_\infty(c, r)$ and grid $\mathcal{G}_{r_g}$, we say a shift $s \in \mathbb{C}$ is good if after applying the shift, all the points in the shifted box $\mathcal{B}_\infty(c, r) + s$ are mapped to the same point by



(a) a box and grid (b) a good shift (c) a bad shift

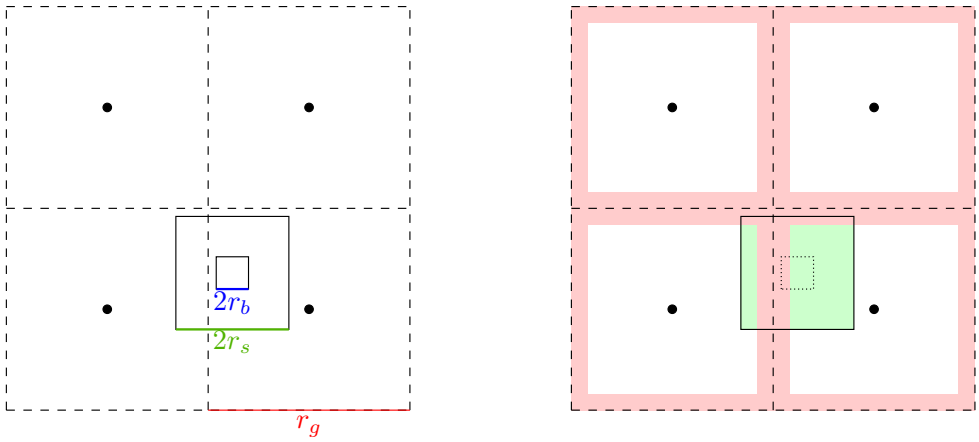
Figure 13.4: Illustration of good and bad shifts in Definition 13.2.6. In (a), the small square represents box $\mathcal{B}_\infty(c, r_b)$, and the dashed lines represent the decision boundary of Π_{r_g}. The arrows in (b) and (c) represent two different shifts, where the shift in (b) is an example of good shift, since the shifted box does not intersect with the decision boundaries of Π_{r_g}, while the shift in (c) is an example of bad shift, since the shifted box intersects with the decision boundaries of Π_{r_g}.

Π_{r_g} (recall that Π_{r_g} projects any point to the nearst grid point in $\mathcal{G}_{r_g}$). We formulate the notation of a good shift in the following definition, and illustrate in Figure 13.4.

Definition 13.2.6 (good shift)**.** For any r_g, r_b, and any $c \in \mathbb{C}$, we say shift $s \in \mathbb{C}$ is a good shift if

$$\left|\Pi_{r_g}(\mathcal{B}_\infty(c, r_b) + s)\right| = 1.$$

The following lemma intuitively states that if we take a box of radius r_b (or equivalently, side length $2r_b$) and shift it randomly by an offset in $\mathcal{B}_\infty(0, r_s)$ (or equivalently, $[-r_s, r_s] \times [-r_s, r_s]$) where $r_s \geq r_b$, and next we round everyone inside that shifted box to the closest point in G_{r_g} where $r_g \geq 2r_s$, then with probability at least $(1 - r_b/r_s)^2$ everyone will be rounded to the same point. In other words, let $s \sim \mathcal{B}_\infty(0, r_s)$, for box $\mathcal{B}_\infty(c, r_b)$ and



(a) $r_g \geq 2r_s \geq 2r_b$ (b) partition of good and bad shifts

Figure 13.5: Illustration of Lemma 13.2.6. In (a) the smallest square represents box $\mathcal{B}_\infty(c, r_b)$, the medium-sized square represents $\mathcal{B}_\infty(c, r_s)$, and the dashed lines represent decision boundaries of Π_{r_g}. Note that for $s \sim \mathcal{B}_\infty(0, r_s)$, the center of the shifted box $s+\mathcal{B}_\infty(c, r_b)$ is $s + c \sim \mathcal{B}_\infty(c, r_s)$. Shift s is good (recall in Definition 13.2.6) for box $\mathcal{B}_\infty(c, r_b)$ and grid $\mathcal{G}_{r_g}$ if and only if the distance between $s+c$ and decision boundaries of Π_{r_g} is greater than r_b. In (b), we draw in red the set of points which are within distance at most r_b to the decision boundaries of Π_{r_g}. Then in (b) the red part inside $\mathcal{B}_\infty(c, r_s)$ corresponds to bad shifts (plus c), and the green part corresponds to good shifts (plus c). Intuitively, the fraction of the green part is at least $(1 - r_b/r_s)^2$ because the vertical red strips can cover a width of at most $2r_b$ on the x-axis of $\mathcal{B}_\infty(c, r_s)$ (whose side length is $2r_s$), and the horizontal red strips can cover a width of at most $2r_b$ on the y-axis.

grid $\mathcal{G}_{r_g}$, s is a good shift with probability at least $(1 - r_b/r_s)^2$. We illustrate the lemma in Figure 13.5.

Lemma 13.2.6 (property of a randomly shifted box)**.** *For any r_g, r_s, r_b so that $r_g/2 \geq r_s \geq r_b > 0$ and any $c \in \mathbb{C}$, let $s \in \mathbb{C}$ be uniform randomly chosen in $\mathcal{B}_\infty(0, r_s)$, then*

$$\Pr_{s \sim \mathcal{B}_\infty(0, r_s)} \left[\left| \Pi_{r_g}(\mathcal{B}_\infty(c, r_b) + s) \right| = 1 \right] \geq \left(1 - \frac{r_b}{r_s}\right)^2,$$

where we refer r_g, r_s, r_b as the radius of grid, shift and box respectively, and we use notation $C + s$ to refer to $\{c + s : c \in C\}$.

Proof. We consider complex numbers as points in 2D plane, where the real part is the coordinate on x-axis, and the imaginary part is the coordinate on y-axis. Note that the "decision boundary" of projection Π_{r_g} from $\mathbb{C}$ onto grid $\mathcal{G}_{r_g}$ consists of verticle lines of form $x = (m+\frac{1}{2})r_g$ and horizontal lines of form $y = (m+\frac{1}{2})r_g$, where $m \in \mathbb{Z}$. $\left|\Pi_{r_g}(\mathcal{B}_\infty(c, r_b) + s)\right| = 1$ if and only if the shifted box $\mathcal{B}_\infty(c, r_b) + s$ does not intersect with the "decision boundary".

Let $s = s_x + s_y\mathbf{i}$ and $c = c_x + c_y\mathbf{i}$. Then the shifted box does not intersect with the "decision boundary" if and only if both the interval

$$[c_x - r_b + s_x, c_x + r_b + s_x] \text{ and } [c_y - r_b + s_y, c_y + r_b + s_y]$$

do not intersect with $\{(m + \frac{1}{2})r_g : m \in \mathbb{Z}\}$. The probability of each one is at least $1 - \frac{r_b}{r_s}$, and two events are independent. Therefore, we get the claimed result. □

In the following, we define event $\mathcal{E}$, which is a sufficient condition for the correctness of Algorithm 13.3. Event $\mathcal{E}$ consists of three parts. Part 1 of $\mathcal{E}$ is used to prove that $a_\ell \le 10 \log n$ for $\ell \in [L-1]$ on Line 29 in Algorithm 13.3. Part 2 and Part 3 of $\mathcal{E}$ are used to prove that Line 15 to Line 20 in Algorithm 13.3 give a desirable $z^{(\ell)}$ for $\ell \in [L]$.

Definition 13.2.7 (sufficient condition for the correctness of Algorithm 13.3)**.** For input signal $x \in \mathbb{C}^n$, let $\mu = \frac{1}{\sqrt{k}}\|\widehat{x}_{-k}\|_2$ and R^* is an upper bound of $\|\widehat{x}\|_\infty/\mu$. Let S be top $C_S k$ coordinates in $\widehat{x}$. Let $H = \min\{\log k + C_H, \log R^*\}$, and $L = \log R^* - H + 1$. For $\ell \in [L]$, let $\nu_\ell = 2^{-\ell}\mu R^*$. For $\ell \in [L-1]$, let $s_\ell^{(a)}$ be the a-th uniform randomly sampled from $\mathcal{B}_\infty(0, \alpha\nu_\ell)$ as appeared on Line 26 in Algorithm 13.3 (i.e. $s_\ell^{(1)}, \ldots, s_\ell^{(a_\ell)}$ are sampled, and $s_\ell^{(a_\ell)}$ is the first that satisfies the condition on Line 28). For the sake of analysis, we assume that Algorithm 13.3 actually produces an infinite sequence of shifts $s_\ell^{(1)}, s_\ell^{(2)}, \ldots$, and

chooses the smallest a_ℓ so that $s_\ell^{(a_\ell)}$ satisfies $\forall f \in \mathrm{supp}(y^{(\ell-1)} + z^{(\ell)}), |\Pi_{\beta\nu_\ell}(\mathcal{B}_\infty(y_f^{(\ell-1)} + z_f^{(\ell)} + s_\ell^{(a_\ell)}, 2^{1-H}\nu_\ell))| = 1$ on Line 28.

For $\ell \in [L-1]$, we define random variable a'_ℓ to be the smallest a' such that for all $f \in S$,

$$\left|\Pi_{\beta\nu_\ell}\left(\mathcal{B}_\infty(\widehat{x}_f + s_\ell^{(a')}, 2^{3-H}\nu_\ell)\right)\right| = 1.$$

We define event $\mathcal{E}$ to be all of the following events hold.

1. For all $\ell \in [L-1]$, $a'_\ell \leq 10 \log n$.
2. For $\ell = 1$, if we execute Line 15 to Line 20 in Algorithm 13.3 with $y^{(0)} = 0$, we get $z^{(1)}$ such that $\|\widehat{x} - z^{(1)}\|_\infty \leq 2^{1-H}\nu_1$ and $\mathrm{supp}(z^{(1)}) \subseteq S$.
3. For all $\ell \in \{2, \ldots, L\}$, for all $a \in [10 \log n]$, if we execute Line 15 to Line 20 in Algorithm 13.3 with $y^{(\ell-1)} = \xi$ where

$$\xi_f = \begin{cases} \Pi_{\beta\nu_\ell}(\widehat{x}_f + s_{\ell-1}^{(a)}), & \text{if } f \in S; \\ 0, & \text{if } f \in \overline{S}. \end{cases}$$

then we get $z^{(\ell)}$ such that $\|\widehat{x} - y^{(\ell-1)} - z^{(\ell)}\|_\infty \leq 2^{1-H}\nu_\ell$ and $\mathrm{supp}(y^{(\ell-1)} + z^{(\ell)}) \subseteq S$.

In the following, we will prove that for fixed x, under the randomness of $\{s_\ell^{(a)}\}_{\ell\in[L-1],a\in\{1,\ldots\}}$ and $\mathcal{T} = \{\mathcal{T}^{(h)}\}_{h\in[H]}$, event $\mathcal{E}$ (defined in Definition 13.2.7) happens with probability at least $1 - 1/\mathrm{poly}(n)$. Moreover, we will prove that event $\mathcal{E}$ is a sufficient condition for the correctness of Algorithm 13.3. Namely, conditioned on event $\mathcal{E}$, Algorithm 13.3 gives a desirable output.

Lemma 13.2.7 (event $\mathcal{E}$ happens with high probability)**.** *Let $\mathcal{E}$ in Definition 13.2.7. For any fixed $x \in \mathbb{C}^n$, under the randomness of shifts $\{s_\ell^{(a)}\}_{\ell\in[L-1],a\in\{1,\ldots\}}$ and $\mathcal{T} = \{\mathcal{T}^{(h)}\}_{h\in[H]}$,*

$$\Pr[\mathcal{E}] \geq 1 - 1/\mathrm{poly}(n).$$

Proof. We bound the failure probability of each parts in event $\mathcal{E}$ respectively as follows, and $\Pr[\mathcal{E}] \geq 1 - 1/\operatorname{poly}(n)$ follows by a union bound.

Part 1. If $H = \log R^*$, then $L = 1$ and it is trivially true that "for all $\ell \in [L-1], a'_\ell \leq 10 \log n$". Otherwise, we have $H = \log k + C_H$. By Lemma 13.2.6, for any $\ell \in [L-1]$, for any $f \in S$,

$$\Pr_{s \sim \mathcal{B}_\infty(0, \alpha\nu_\ell)} \left[\left| \Pi_{\beta\nu_\ell} \left(\mathcal{B}_\infty \left(\widehat{x}_f, 2^{3-H}\nu_\ell \right) + s \right) \right| = 1 \right] \geq \left(1 - \frac{2^{3-H}\nu_\ell}{\alpha\nu_\ell} \right)^2 = \left(1 - \frac{2^{3-H}}{\alpha} \right)^2,$$

where $(1 - 2^{3-H}/\alpha)^2 \geq 1 - 2^{4-C_H - \log k}/\alpha \geq 1 - \frac{1}{100k}$ by our choice of α and C_H in Table 13.2.

For each $\ell \in [L-1]$, by a union bound over all f in S, the probability is at least $1 - \frac{C_S k}{100k} = 1 - \frac{26k}{100k} \geq \frac{1}{2}$ that for all $f \in S$, $|\Pi_{\beta\nu_\ell}(\mathcal{B}_\infty(\widehat{x}_f + s, 2^{3-H}\nu_\ell))| = 1$ where $s \sim \mathcal{B}_\infty(0, \alpha\nu_\ell)$. Formally, we get

$$\Pr_{s \sim \mathcal{B}_\infty(0, \alpha\nu_\ell)} \left[\left| \Pi_{\beta\nu_\ell} \left(\mathcal{B}_\infty \left(\widehat{x}_f, 2^{3-H}\nu_\ell \right) + s \right) \right| = 1, \forall f \in S \right] \geq 1/2.$$

Therefore, by definition of a'_ℓ in Definition 13.2.7,

$$\Pr[a'_\ell \leq 10 \log n] \geq 1 - (1/2)^{10 \log n} = 1 - 1/n^{10}.$$

By a union bound over all $\ell \in [L-1]$, the probability is at least $1 - L/n^{10} = 1 - 1/\operatorname{poly}(n)$ that for all $\ell \in [L-1]$, $a'_\ell \leq 10 \log n$.

Part 2. By Lemma 13.2.5 and a union bound over all $h \in [H]$, the failure probability is at most $H/\operatorname{poly}(n) = 1/\operatorname{poly}(n)$, where $H = O(\log k)$ and so $H/\operatorname{poly}(n)$ is still $1/\operatorname{poly}(n)$.

Part 3. For each $\ell \in \{2, \ldots, L\}$ and $a \in [10 \log n]$, similar to the above argument, each has failure probability at most $1/\operatorname{poly}(n)$. By a union bound, the failure probability is

at most

$$(L-1)\cdot(10\log n)/\operatorname{poly}(n) = 1/\operatorname{poly}(n).$$

□

In the following lemma, we show that if event $\mathcal{E}$ (defined in Definition 13.2.7) happens, then Algorithm 13.3 gives a desirable output.

Lemma 13.2.8 (correctness of Algorithm 13.3 conditioned on $\mathcal{E}$)**.** *Let $n = p^d$, and let $k \in [n]$. Let $x \in \mathbb{C}^n$ be input signal. Let $\mu = \frac{1}{\sqrt{k}}\|\widehat{x}_{-k}\|_2$. Let $R^* \geq \|\widehat{x}\|_\infty / \mu$ and R^* is a power of 2. Let $H = \min\{\log k + C_H, \log R^*\}$. Let $L = \log R^* - H + 1$. For $\ell \in [L-1]$, let $y^{(\ell)}$ be the vector obtained on Line 31 of Algorithm 13.3. For $\ell \in [L]$, let $z^{(\ell)}$ be the vector obtained on Line 20. Note that $y^{(0)} = 0$, and $y^{(L-1)} + z^{(L)}$ is the output of* FourierSparseRecovery(x, n, k, R^*, μ) *in Algorithm 13.3. Conditioned on the event $\mathcal{E}$ (defined in Definition 13.2.7) happens, we have*

$$\|\widehat{x} - y^{(L-1)} - z^{(L)}\|_\infty \leq \frac{1}{\sqrt{k}}\|\widehat{x}_{-k}\|_2.$$

Proof. We first discuss the case that $H = \log R^*$. In that case, $L = 1$. Conditioned on the event $\mathcal{E}$ (Part 2 of $\mathcal{E}$), $z^{(1)}$ obtained through Line 15 to Line 20 in Algorithm 13.3 satisfies $\|\widehat{x} - z^{(1)}\|_\infty \leq 2^{1-H}\nu_1 = 2^{1-H}(2^{-1}\mu R^*) = \mu$.

In the rest of the proof, we discuss the case that $H > \log R^*$. For $\ell \in [L]$, let $\nu_\ell = 2^{-\ell}\mu R^*$. For $\ell \in [L-1]$, let $s_\ell^{(a_\ell)} \in \mathcal{B}_\infty(0, \alpha\nu_\ell)$ denote the first $s_\ell^{(a)}$ on Line 26 in Algorithm 13.3 such that for all $f \in \operatorname{supp}(y^{(\ell-1)} + z^{(\ell)})$,

$$\left|\Pi_{\beta\nu_\ell}\left(\mathcal{B}_\infty\left(y_f^{(\ell-1)} + z_f^{(\ell)} + s_\ell^{(a)}, 2^{1-H}\nu_\ell\right)\right)\right| = 1.$$

For $\ell \in [L-1]$, we define $\xi^{(\ell)} \in \mathbb{C}^{[p]^d}$ as follows

$$\xi_f^{(\ell)} = \begin{cases} \Pi_{\beta\nu_\ell}(\widehat{x}_f + s_\ell^{(a_\ell)}), & \text{if } f \in S; \\ 0, & \text{if } f \in \overline{S}. \end{cases}$$

We also define $\xi^{(0)} = 0$, $s_0^{(a)} = 0$ for $a \in \{1, \ldots\}$ and $a_0 = 1$.

(**Goal: Inductive Hypothesis**) We are going to prove that conditioned on event $\mathcal{E}$ (defined in Definition 13.2.7), for all $\ell \in \{0, \ldots, L-1\}$,

$$y^{(\ell)} = \xi^{(\ell)} \quad \text{and} \quad a_\ell \le 10 \log n.$$

(**Base case**) Note that $y^{(0)} = \xi^{(0)} = 0$ and $a_0 = 1 \le 10 \log n$.

(**Inductive step**) We will prove that conditioned on event $\mathcal{E}$, if $y^{(\ell-1)} = \xi^{(\ell-1)}$ and $a_{\ell-1} \le 10 \log n$ for $\ell \in [L-1]$, then $y^{(\ell)} = \xi^{(\ell)}$ and $a_\ell \le 10 \log n$.

(**Proving** $a_\ell \le 10 \log n$) Conditioned on event $\mathcal{E}$ (if $L = 1$ then from Part 2 of $\mathcal{E}$, otherwise from Part 3 of $\mathcal{E}$ and by the fact that $a_{\ell-1} \le 10 \log n$), $z^{(\ell)}$ obtained through Line 15 to Line 20 in Algorithm 13.3 satisfies $\|\widehat{x} - \xi^{(\ell-1)} - z^{(\ell)}\|_\infty \le 2^{1-H}\nu_\ell$ and $\mathrm{supp}(z^{(\ell)}) \subseteq S$. Namely, for all $f \in [p]^d$, $\xi_f^{(\ell-1)} + z_f^{(\ell)} \in \mathcal{B}_\infty(\widehat{x}_f, 2^{1-H}\nu_\ell)$. Recall the definition of a'_ℓ in Definition 13.2.7. We can prove that $a_\ell \le a'_\ell$ because if for all $f \in S$,

$$\left|\Pi_{\beta\nu_\ell}\left(\mathcal{B}_\infty\left(\widehat{x}_f + s_\ell^{(a'_\ell)}, 2^{3-H}\nu_\ell\right)\right)\right| = 1,$$

then for all $f \in \mathrm{supp}(y^{(\ell-1)} + z^{(\ell)})$,

$$\left|\Pi_{\beta\nu_\ell}\left(\mathcal{B}_\infty\left(y_f^{(\ell-1)} + z_f^{(\ell)} + s_\ell^{(a'_\ell)}, 2^{1-H}\nu_\ell\right)\right)\right| = 1$$

where

$$\mathcal{B}_\infty\left(y_f^{(\ell-1)} + z_f^{(\ell)} + s_\ell^{(a'_\ell)}, 2^{1-H}\nu_\ell\right) \subseteq \mathcal{B}_\infty\left(\widehat{x}_f + s_\ell^{(a'_\ell)}, 2^{3-H}\nu_\ell\right)$$

which follows by $\xi_f^{(\ell-1)} + z_f^{(\ell)} \in \mathcal{B}_\infty(\widehat{x}_f, 2^{1-H}\nu_\ell)$. Therefore, conditioned on $\mathcal{E}$ (Part 1 of $\mathcal{E}$), $a_\ell \le a'_\ell \le 10 \log n$.

(**Proving** $y_f^{(\ell)} = \xi_f^{(\ell)}$) For $f \in [p]^d$, we will prove that $y_f^{(\ell)} = \xi_f^{(\ell)}$ in two cases.

(**Case 1**) If $f \in \mathrm{supp}(y^{(\ell-1)} + z^{(\ell)}) \subseteq S$. We have

$$\begin{aligned} y_f^{(\ell)} &= \Pi_{\beta\nu_\ell}\left(y_f^{(\ell-1)} + z_f^{(\ell)} + s_\ell^{(a_\ell)}\right) \\ &= \Pi_{\beta\nu_\ell}\left(\xi_f^{(\ell-1)} + z_f^{(\ell)} + s_\ell^{(a_\ell)}\right). \end{aligned}$$

Because $\xi_f^{(\ell-1)} + z_f^{(\ell)} \in \mathcal{B}_\infty(\widehat{x}_f, 2^{1-H}\nu_\ell)$, we have $\xi_f^{(\ell-1)} + z_f^{(\ell)} + s_\ell^{(a_\ell)} \in \mathcal{B}_\infty(\widehat{x}_f + s_\ell^{(a_\ell)}, 2^{1-H}\nu_\ell)$. By the choice of $s_\ell^{(a_\ell)}$, $\Pi_{\beta\nu_\ell}(\xi_f^{(\ell-1)} + z_f^{(\ell)} + s_\ell^{(a_\ell)}) = \Pi_{\beta\nu_\ell}(\widehat{x}_f + s_\ell^{(a_\ell)})$. Thus $y_f^{(\ell)} = \xi_f^{(\ell)}$.

(**Case 2**) If $f \notin \mathrm{supp}(y^{(\ell-1)} + z^{(\ell)})$. We have $y_f^{(\ell)} = 0$. Because $\xi_f^{(\ell-1)} + z_f^{(\ell)} \in \mathcal{B}_\infty(\widehat{x}_f, 2^{1-H}\nu_\ell)$, we have $|\widehat{x}_f| < 2^{2-H}\nu_\ell < 0.1\beta\nu_\ell$ by our choice of H. We can easily prove that $\xi_f^{(\ell)} = 0 = y_f^{(\ell)}$ in the following two cases:

(**Case 2.1**) If $f \in S$, we have $\xi_f^{(\ell)} = \Pi_{\beta\nu_\ell}(\widehat{x}_f + s_\ell^{(a_\ell)}) = 0$ because

$$|\widehat{x}_f| + |s_\ell^{(a_\ell)}| < 0.1\beta\nu_\ell + 2\alpha\nu_\ell < 0.5\beta\nu_\ell.$$

(**Case 2.2**) If $f \in \overline{S}$, $\xi_f^{(\ell)} = 0$ by definition of $\xi^{(\ell)}$.

Therefore, for all $\ell \in [L-1]$, $y^{(\ell)} = \xi^{(\ell)}$ and $a_\ell \le 10 \log n$. Again conditioned on event $\mathcal{E}$ (Part 3 of $\mathcal{E}$), $z^{(L)}$ obtained through Line 15 to Line 20 in Algorithm 13.3 satisfies

$$\|\widehat{x} - y^{(L-1)} - z^{(L)}\|_\infty \le 2^{1-H}\nu_L = 2^{1-H}(2^{-(\log R^* - H + 1)}\mu R^*) = \mu.$$

Therefore, $y^{(L-1)} + z^{(L)}$ on Line 22 gives a desirable output.

□

Now we present our main theorem, which proves the correctness of Algorithm 13.3, and shows its sample complexity and time complexity.

Theorem 13.2.9 (main result, formal version)**.** *Let $n = p^d$ where both p and d are positive integers. Let $x \in \mathbb{C}^{[p]^d}$. Let $k \in \{1, \ldots, n\}$. Assume we know $\mu = \frac{1}{k}\|\widehat{x}_{-k}\|_2$ and $R^* \geq \|\widehat{x}\|_\infty / \mu$ where $\log R^* = O(\log n)$. There is an algorithm (Algorithm 13.3) that takes $O(k \log k \log n)$ samples from x, runs in $O(n \log^3 n \log k)$ time, and outputs a $O(k)$-sparse vector y such that*

$$\|\widehat{x} - y\|_\infty \leq \frac{1}{\sqrt{k}} \min_{k-\text{sparse } x'} \|\widehat{x} - x'\|_2$$

holds with probability at least $1 - 1/\operatorname{poly}(n)$.

Proof. The correctness of Algorithm 13.3 follows directly from Lemma 13.2.7 and Lemma 13.2.8. The number of samples from x is

$$B \cdot R \cdot H = O(k \cdot \log n \cdot \log k) = O(k \log k \log n).$$

Its running time is dominated by $L \cdot H = O(\log k \log n)$ invocations of LinfinityReduce (in Algorithm 13.4). By Lemma 13.2.5, the running time of LinfinityReduce is $O(n \log^2 n)$. Therefore, the running time of Algorithm 13.3 is

$$O(L \cdot H \cdot n \log^2 n) = O(\log k \cdot \log n \cdot n \log^2 n) = O(n \log^3 n \log k).$$

□

Chapter 14

Set Query

14.1 Introduction

This result presented in this section is based on a un-published manuscript, and is a joint work Vasileios Nakos and Zhengyu Wang [NSW18].

Our first result is on the set-query problem. We show that $O(k/\epsilon)$ measurements, $O(\log k)$ update time and $O(k \log k)$ query time are possible. Our hash functions are only 2-wise independent, and the amount of space to store the matrix Φ is only $O(\log k)$ words. The previous work of [Pri11] needed $\Omega(n)$ time to store the matrix, since the columns had to be fully independent, something that makes the particular sketch not applicable in the streaming model. The problem of whether the matrix can be stored in less space was explicitly stated in that paper as an open problem:

"Our analysis assumes that the columns of A are fully independent. It would be valuable to reduce the independence needed, and hence the space required to store A."

Theorem 14.1.1 (Classical set query)**.** *There exists a randomized construction of a linear sketch $\Phi \in \mathbb{R}^{m \times n}$ with $m = O(k/\epsilon)$, such that given $y = \Phi x$ and a set $S \subset [n]$ of cardinality at most k, we can find in $O(k \log k)$ time a k-sparse vector such that*

$$\|x' - x_S\|_2^2 \leq \epsilon \|x_{[n] \setminus S}\|_2^2$$

holds with probability $9/10$. The time to update the linear sketch is $O(\log k)$ and the space to store the matrix is $O(\log k)$ words.

Our second main result concerns the ubiquitous problem of frequency estimation from Fourier measurements. For any vector $x \in \mathbb{C}^n$, let $\widehat{x} \in \mathbb{C}^n$ denote the discrete Fourier transform of x. The only previous known result by the breakthrough work of [Kap17] achieves

	Space	Measurements	Decoding Time	Update Time
[Pri11]	n	k/ϵ	k	1
Theorem 14.1.1	$\log k$	k/ϵ	$k \log k$	$\log k$

Table 14.1: Results on set query. We ignore the "O" for simplicity. "Space" means the space to store matrix, which doesn't include number of measurements. We note that the decoding time of the set query algorithms do **not** depend on ϵ.

	Measurements	Running Time
[Kap17]	k/ϵ	$k/\epsilon \cdot \log^{3.001} n$
Theorem 14.1.2	$k/\epsilon \cdot \log n$	$k/\epsilon \cdot \log n$

Table 14.2: Results on Fourier set query. We ignore the "O" for simplicity. We assume signal is bounded by poly(n). In Fourier set query, there is no update time concept.

sample complexity $O(k/\epsilon)$ and running time $O(\epsilon^{-1} k \log^{2.001} n \log R^*)$ for ℓ_2/ℓ_2 Fourier set query. Here, R^* is an upper bound on the $\|\cdot\|_\infty$ norm of the vector, in most applications being considered poly(n). We indicate how our approach immediately gives an algorithm with $O(\epsilon^{-1} k \log n)$ running time. The result we get has no dependence on the running time on $\log R^*$, but does not achieve the optimal sample complexity.

Theorem 14.1.2 (Fourier set query)**.** *Given a vector $x \in \mathbb{C}^n$, for every $\epsilon \in (0,1)$ and $k \geq 1$, any $S \subseteq [n]$, $|S| = k$, there exists an algorithm that takes $O(\epsilon^{-1} k \log(n/\delta))$ samples, runs in $O(\epsilon^{-1} k \log(n/\delta))$ time, and outputs a vector $x' \in \mathbb{C}^n$ such that $\|(x' - \widehat{x})_S\|_2^2 \leq \epsilon \|\widehat{x}_{[n]\setminus S}\|_2^2 + \delta \|\widehat{x}\|_1^2$ holds with probability at least $9/10$.*

Note that δ dependence is standard in Fourier transform setting, see e.g. [HIKP12a, PS15, CKPS16].

14.2 Classical set query

In this section, we present our set query data structure. We first give the construction of sketching matrix in Section 14.2.1. Then we show our new set query data structure in Section 14.2.2. We give an analysis of the data structure in Section 14.2.4, whose correctness relies on a technical lemma on iterative loop, which is proved in Section 14.2.3.

14.2.1 Construction of sketching matrix

The sketching matrix used in our set query data structure consists of $\log k$ parts, where the i-th part ($i \in [\log k]$) of the sketching matrix corresponds to a count sketch data structure that hashes n elements into B_i bins. We defer the choice of B_i to Section 14.2.2. We present the sketching matrix Φ below.

Definition 14.2.1 (Sketching matrix Φ)**.** For each $i \in [\log k]$, we choose $h_i : [n] \to [B_i]$ to be a pairwise independent hash function, and choose $\sigma_i : [n] \to \{-1, +1\}$ to be a pairwise independent hash function.

For each $i \in [\log k]$, we define COUNT-SKETCH matrix $\Phi^{(i)} \in \mathbb{R}^{B_i \times n}$ as follows

$$\Phi^{(i)}_{j,l} = \begin{cases} \sigma_i(l) & \text{if } h_i(l) = j \\ 0 & \text{otherwise.} \end{cases}$$

Let $m = \sum_{i=1}^{\log k} B_i$. We define matrix $\Phi \in \mathbb{R}^{m \times n}$ as follows

$$\Phi = \begin{bmatrix} \Phi^{(1)} \\ \Phi^{(2)} \\ \cdots \\ \Phi^{(\log k)} \end{bmatrix}.$$

14.2.2 Algorithm

Let C be a sufficiently large constant, and γ be a sufficiently small constant. For $i \in [\log k]$, let $k_i = k\gamma^i$ and $\epsilon_i = \epsilon(10\gamma)^i$. Let $B_i = Ck_i/\epsilon_i$. The set query algorithm maintains the measurement $y = \Phi x$.

On update operation, the algorithm follows directly through the definition of sketching matrix Φ. Namely, on $\textsc{Update}(i, \Delta)$, the algorithm finds the non-zero elements of the i-th column of Φ, and update the measurement y accordingly.

On query operation $\textsc{Query}(x, S)$, the recovery procedure uses x' to record the current best approximation of x, and initially set $x' \leftarrow 0$. The procedure consists of $\log k$ iterations. In the i-th iteration, let $S_i \subseteq S$ denote the set of coordinates in x that are yet to be recovered. Initially $S_1 = S$. Remind that sketching matrix Φ in Definition 14.2.1 consists of $\log k$ parts, and the i-th part corresponds to a count sketch data structure with pair-wise independent hash function h_i and random sign function σ_i. In the i-th iteration of the set query algorithm,

$$T_i \leftarrow \left\{ j \in S_i \;\middle|\; h_i^{-1}(j) \cap S_i = \emptyset \right\}.$$

Intuitively the hash function h_i maps n elements into B_i bins, for any $j \in S_i$, we have $j \in T_i$ if and only if j does not collide with any other elements in S_i (in that case, we say j is isolated).

The procedure also maintains $y^{(i)}$, which is the measurement of the residual vector $x - x'$ (before the start of i-th iteration). After obtaining T_i, the procedure uses $y^{(i)}$ to estimate x_j for every $j \in T_i$. Then the procedure updates S_i and x' accordingly. Finally, the algorithm outputs x'. Our iterative set query algorithm is presented in Algorithm 14.1.

Algorithm 14.1 Iterative Set Query

procedure ITERATIVESETQUERY(ϵ, k) ▷ Theorem 14.2.2
  $C \leftarrow 20, \gamma \leftarrow 1/600$
  for $i = 1 \rightarrow \log k$ **do**
    $k_i \leftarrow k\gamma^i$, $\epsilon_i \leftarrow \epsilon(10\gamma)^i$
    $B_i \leftarrow Ck_i/\epsilon_i$
  end for
  $m \leftarrow \sum_{i=1}^{\log k} B_i$ ▷ m is the number of measurement
  Construct matrix Φ according to Definition 14.2.1 ▷ $\Phi \in \mathbb{R}^{m \times n}$
  $y \leftarrow 0$ ▷ $y \in \mathbb{R}^m$
  procedure UPDATE(i, Δ)
    $y \leftarrow y + \Delta \cdot (\Phi \cdot e_i)$
  end procedure
  procedure QUERY(x, S)
    $S_1 \leftarrow S$
    $y^{(1)} \leftarrow y$
    $x' \leftarrow 0$ ▷ $x' \in \mathbb{R}^n$ records an estimation of x
    for $i = 1 \rightarrow \log k$ **do**
      Find $T_i \subseteq S_i$ such that for each $j \in T_i$, it is isolated from other coordinates in S_i
      $l \leftarrow \sum_{i'=1}^{i-1} B_{i'}$
      For each $j \in T_i$, compute $\widehat{x}_j \leftarrow y^{(i)}_{l+h_i(j)}$
      $S_{i+1} \leftarrow S_i \backslash T_i$
      $y^{(i+1)} \leftarrow y^{(i)} - \Phi \widehat{x}_{T_i}$
      $x' \leftarrow x' + \widehat{x}_{T_i}$
    end for
    return x'
  end procedure
end procedure

14.2.3 Iterative loop analysis

In this subsection, we characterize and prove two properties of the iterative loop in Algorithm 14.1. The first property states that with good probability, as i increases, the size of S_i shrinks by a factor of at least $1 - \gamma$, and otherwise its size does not increase.

This property is guaranteed by identification property of count sketch and our choice of parameters. The second property states that with good probability, the estimation error in each round is small. This is guaranteed by estimation property of count sketch and our choice of parameters. We formalize both properties in Lemma 14.2.1 as follows.

Lemma 14.2.1. *Given parameters $\epsilon \in (0,1), k \geq 1$, for each $i \in [\log k]$, we define*

$$k_i = k\gamma^i, \epsilon_i = \epsilon(10\gamma)^i, B_i = C \cdot k_i/\epsilon_i.$$

For each $i \in [\log k]$, if

$$\forall j \in [i-1], |S_{j+1}| \leq \gamma |S_j|,$$

and

$$\forall j \in [i-1], \sum_{t \in T_j} |x_t - \widehat{x}_t|^2 \leq \epsilon \cdot 2^{-j} \|x_{\overline{S}}\|_2^2.$$

Then we have :

Property (I) : *With probability at least $1 - \frac{1}{C\gamma}(10\gamma)^i$,*

$$|S_{i+1}| \leq \gamma |S_i|,$$

and otherwise $|S_{i+1}| \leq |S_i|$.

Property (II) : *With probability at least $1 - \frac{2}{C}(20\gamma)^i$,*

$$\sum_{t \in T_i} |x_t - \widehat{x}_t|^2 \leq \epsilon \cdot 2^{-i} \|x_{\overline{S}}\|_2^2.$$

Proof. **Proof of Property (I).**

For each $i \in [\log k]$, for each $j \in S_i$, we use $Z_{i,j}$ to denote the boolean random variable where $Z_{i,j} = 1$ if there exists a $j' \in S_i \backslash \{j\}$ such that $h_i(j) = h_i(j')$ and $Z_{i,j} = 0$ otherwise. By the randomness of hash function h_i,

$$\Pr_{h_i}[Z_{i,j} = 1] = \Pr_{h_i}[j \text{ is not isolated from } S_i] \leq |S_i|/B_i.$$

By the statement of the lemma, we know that $|S_i| \leq \gamma^i k$ and $B_i = Ck\gamma^i/\epsilon_i$, thus

$$\Pr_{h_i}[Z_{i,j} = 1] \leq \gamma^i k/B_i \leq \epsilon_i/C.$$

We define $Z_i = \sum_{j \in S_i} Z_{i,j}$. Next, we can show

$$\mathop{\mathbb{E}}_{h_i}[Z_i] = \mathop{\mathbb{E}}_{h_i}[|\{j \in S_i \mid j \text{ is not isolated from } S_i\}|] = \mathop{\mathbb{E}}_{h_i}\left[\sum_{j \in S_i} Z_{i,j}\right] = \sum_{j \in S_i} \mathop{\mathbb{E}}_{h_i}[Z_{i,j}] \leq |S_i|\epsilon_i/C,$$

where the third step follows by linearity of expectation. Therefore,

$$\Pr_{h_i}[Z_i \geq \gamma|S_i|] \leq \mathop{\mathbb{E}}_{h_i}[Z_i]/(\gamma|S_i|) \leq |S_i|\epsilon_i/(C\gamma|S_i|) = \epsilon_i/(C\gamma),$$

where the first step follows by Markov's inequality, and second step follows by $\mathbb{E}_{h_i}[Z_i] \leq |S_i|\epsilon_i/C$. This implies

$$\Pr[|S_{i+1}| \geq \gamma|S_i|] \leq \epsilon(10\gamma)^i/(C\gamma) \leq \frac{1}{C\gamma}(10\gamma)^i.$$

Proof of Property (II).

We first define random variable V_i to be

$$V_i = \sum_{l \in \left(\cup_{j \in S_i} h_i^{-1}(j)\right) \backslash S_i} |x_l^{(i)}|^2.$$

That is, $x^{(i)}$ is the residual signal in the i-th round, and V_i refers to the noise coordinates that are hashed to the same bins as coordinates in S_i.

We can compute the expectation of V_i,

$$\begin{aligned}\mathop{\mathbb{E}}_{h_i}[V_i] &= \mathop{\mathbb{E}}_{h_i}\left[\sum_{l\in\left(\cup_{j\in S_i}h_i^{-1}(j)\right)\setminus S_i}|x_l^{(i)}|^2\right]\\ &\leq k_i\|x^{(i)}_{\overline{S}_i}\|_2^2/B_i\\ &= \frac{\epsilon_i}{C}\|x^{(i)}_{\overline{S}_i}\|_2^2\\ &= \frac{(10\gamma)^i\epsilon}{C}\|x^{(i)}_{\overline{S}_i}\|_2^2,\end{aligned}$$

where the first step follows by the fact that for each $l \in \overline{S_i}$, the probability that $l \in \left(\cup_{j\in S_i}h_i^{-1}(j)\right)\setminus S_i$ under the randomness of h_i is $|S_i|/B_i \leq k_i/B_i$, the second and third step follow by definitions of B_i and ϵ_i, respectively.

By Markov's inequality, we have

$$\Pr_{h_i}\left[V_i \geq \frac{1}{2}\cdot 10^{-i}\epsilon\left\|x^{(i)}_{\overline{S}_i}\right\|_2^2\right] \leq 2\frac{(100\gamma)^i}{C}. \tag{14.1}$$

In the following, we argue conditioned on the event that all elements in T_i are isolated under h_i and $V_i \leq 10^{-i}\epsilon\|x^{(i)}_{\overline{S}_i}\|_2^2/2$.

Recall the definition of $x^{(i)} \in \mathbb{R}^n$,

$$x^{(i)} = x - \sum_{t\in S\setminus S_i}\widehat{x}_t = x - \sum_{j=1}^{i}\sum_{t\in T_j}\widehat{x}_t. \tag{14.2}$$

For any fixed $t \in T_i$, let $b = h_i(t)$. We have

$$\begin{aligned}
\mathop{\mathbb{E}}_{\sigma_i}\left[|x_t - \widehat{x}_t|^2\right] &= \mathop{\mathbb{E}}_{\sigma_i}\left[\left|x_t - \sigma_i(t) \cdot \sum_{l:h_i(l)=b} \sigma_i(l) x_l^{(i)}\right|^2\right] \\
&= \mathop{\mathbb{E}}_{\sigma_i}\left[\left|\sum_{l \in h_i^{-1}(b)\setminus\{t\}} \sigma_i(l) x_l^{(i)}\right|^2\right] \\
&= \sum_{l \in h_i^{-1}(b)\setminus\{t\}} |x_l^{(i)}|_2^2, \qquad (14.3)
\end{aligned}$$

where the first step follows by definition of $\widehat{x}_t$ in Algorithm 14.1 and construction of sketching matrix Φ in Definition 14.2.1, the third step follows by pairwise independence property of random sign function σ_i.

We can upper bound $\mathbb{E}_{\sigma_i}[\sum_{t\in T_i} |x_t - \widehat{x}_t|^2]$ in the following way,

$$\begin{aligned}
\mathop{\mathbb{E}}_{\sigma_i}\left[\sum_{t\in T_i} |x_t - \widehat{x}_t|^2\right] &= \sum_{t\in T_i} \sum_{l \in h_i^{-1}(t)\setminus\{t\}} |x_l^{(i)}|_2^2 && \text{by Eq. (14.3)} \\
&\leq \sum_{l \in \cup_{t\in S_i} h_i^{-1}(t)\setminus\{t\}} |x_l^{(i)}|_2^2 && \text{by definition of } T_i \\
&\leq \epsilon 10^{-i} \|x_{\overline{S}_i}^{(i)}\|_2^2/2 && \text{by Eq. (14.1)} \\
&\leq \epsilon 10^{-i} \|x_{\overline{S}}\|_2^2 && \text{by Eq. (14.5)} \qquad (14.4)
\end{aligned}$$

Second, we show how to upper bound $\|x^{(i)}_{\overline{S}_i}\|_2^2$,

$$\begin{aligned}
\|x^{(i)}_{\overline{S}_i}\|_2^2 &= \|x^{(i)}_{\overline{S}\cup S\backslash S_i}\|_2^2 && \text{by } \overline{S_i} = \overline{S}\cup S\backslash S_i \\
&= \|x^{(i)}_{\overline{S}}\|_2^2 + \|x^{(i)}_{S\backslash S_i}\|_2^2 && \text{by } \overline{S}\cap(S\backslash S_i) = \emptyset \\
&= \|x_{\overline{S}}\|_2^2 + \|x^{(i)}_{S\backslash S_i}\|_2^2 && \text{by } x^{(i)}_{\overline{S}} = x_{\overline{S}} \\
&= \|x_{\overline{S}}\|_2^2 + \sum_{j=1}^{i-1}\|x^{(i)}_{T_j}\|_2^2 && \text{by } S\backslash S_i = T_1\cup T_2\cup\cdots\cup T_{i-1} \\
&= \|x_{\overline{S}}\|_2^2 + \sum_{j=1}^{i-1}\sum_{t\in T_j}|x_t - \widehat{x}_t|^2 && \text{by Eq. (14.2)} \\
&\leq \|x_{\overline{S}}\|_2^2 + \sum_{j=1}^{i-1}\epsilon 2^{-j}\|x_{\overline{S}}\|_2^2 && \text{by statement of Lemma} \\
&\leq 2\|x_{\overline{S}}\|_2^2. && \text{by geometric sum} \qquad (14.5)
\end{aligned}$$

Using Markov's inequality $\Pr[X \geq a] \leq \mathbb{E}[X]/a$, we have

$$\Pr\left[\sum_{t\in T_i}|x_t - \widehat{x}_t|^2 \geq \epsilon 2^{-i}\|x_{\overline{S}}\|_2^2\right] \leq 5^{-i}.$$

Therefore, we complete the proof. □

14.2.4 Main result

In this section, we prove the correctness of our iterative set query algorithm, as well as justify the claimed space complexity, update time and query time. At the core of correctness is an analysis of the iterative loop, which is given as Lemma 14.2.1 in Section 14.2.3.

Theorem 14.2.2. *Let $m = O(k/\epsilon)$. We can give a randomized construction of a matrix $\Phi \in \mathbb{R}^{m\times n}$ (Definition 14.2.1) such that for each set $S \subseteq [n]$, $|S| \leq k$, and for each vector*

$x \in \mathbb{R}^n$, there is an algorithm (Algorithm 14.1) that takes Φx as its input, and finds $x' \in \mathbb{R}^k$ such that

$$\|x' - x_S\|_2^2 \leq \epsilon \|x_{[n]\setminus S}\|_2^2$$

holds with probability 9/10. The space to store matrix Φ is $O(\log k)$ words. The columns of matrix Φ are 2-wise independent. The update time is $O(\log k)$. The decoding time is $O(k \log k)$.

Proof. Our proof is based on properties of the iterative loop in Algorithm 14.1. The properties are formulated as Lemma 14.2.1 in Section 14.2.3. In summary, property (I) in Lemma 14.2.1 shows that with good probability, as i increases, the size of S_i (i.e. the set of coordinates in S that yet to be recovered at the beginning of i-th round) shrinks by a constant factor, and otherwise its size does not increase. Property (II) in Lemma 14.2.1 shows that with good probability, the estimation error in each round is small.

Proof of success probability. It follows by recursively applying Lemma 14.2.1. By union bound, the failure probability is bounded by

$$\sum_{i=1}^{\log k} \left(\frac{1}{C\gamma} (10\gamma)^i + \frac{2}{C} (20\gamma)^i \right) < 1/10,$$

which follows from geometric sum and the choice of parameters.

Proof of estimation error.

$$\|x' - x_S\|_2^2 \leq \sum_{i=1}^{\log k} \sum_{t \in T_i} |x_t - \widehat{x}_t|^2 \leq \sum_{i=1}^{\log k} \epsilon \cdot 2^{-i} \|x_{\overline{S}}\|_2^2 \leq \epsilon \|x_{\overline{S}}\|_2^2.$$

Proof of space to store Φ. Since for each $i \in [\log k]$, $\Phi^{(i)}$ only uses 2 pairwise independent hash functions, the space to store matrix Φ is $O(\log k)$ words.

Proof of update time. Upon update (j, Δ), for each $\Phi^{(i)}$, the algorithm only needs to update one measurement (located at $h_i(j)$). Therefore, the update time is $O(\log k)$. Since every element of the matrix can be computed in $O(1)$ time.

Proof of query time. First of all, we note that the step in Line 18 can be performed in $O(|S_i|)$ time upon query, since we can check every $j \in S_i$, and mark the buckets that have more than one j hashed to them; in the next step we just keep only those coordinates j that are hashed to exactly one bucket. Recursively applying Lemma 14.2.1, we have with constant probability, such that for all $i \in [\log k]$, $|S_{i+1}| \leq \gamma |S_i|$. It implies that for all $i \in [\log k]$, $|S_i| \leq \gamma^{i-1}|S_1| \leq \gamma^{i-1}k$. Note that in iteration $i \in [\log k]$, we need $O(|S_i|)$ update time. Therefore, overall the iterations, the time of this part is

$$\sum_{i=1}^{\log k} O(|S_i|) \leq O\left(\sum_{i=1}^{\log k} \gamma^{i-1} k\right) = O(k),$$

where the last step follows by the choice of γ.

We know that $\|z\|_0 = O(k)$ and $|T_i| = O(k)$. Thus, in each iteration, the time of doing step $y^{(i+1)} \leftarrow y^{(i)} - \Phi Z_{T_i}$ requires $O(k)$ time. Thus the subtraction time over all iterations is $O(k \log k)$, since there are $O(\log k)$ iterations. Putting it together, we get $O(k \log k)$ query time.

□

14.3 Fourier set query

We present some definitions and backgrounds of Fourier transform in Section 14.3.1. Then we introduce spectrum permutations and filter functions in Section 14.3.2. They are used as hashing schemes in Fourier transform literatures. In Section 14.3.3, we introduce collision events. large offset events, and large noise events. We give a new algorithm and provide iterative loop analysis in Section 14.3.4. Finally, we present main result in Section 14.3.5.

14.3.1 Definitions and some backgrounds of Fourier transform

We use $\mathbf{i}$ to denote $\sqrt{-1}$. Note that $e^{\mathbf{i}\theta} = \cos(\theta) + \mathbf{i}\sin(\theta)$. For any complex number $z \in \mathbb{C}$, we have $z = a + \mathbf{i}b$, where $a, b \in \mathbb{R}$. We define the complement of z as $\overline{z} = a - \mathbf{i}b$. We define $|z| = \sqrt{z\overline{z}} = \sqrt{a^2 + b^2}$. For any complex vector $x \in \mathbb{C}^n$, we use $\mathrm{supp}(x)$ to denote the support of x, and then $\|x\|_0 = |\mathrm{supp}(x)|$.

The discrete convolution of functions f and g is given by,

$$(f * g)[n] = \sum_{m=-\infty}^{+\infty} f[m]g[n-m]$$

For a complex vector $x \in \mathbb{C}^n$, we use $\widehat{x} \in \mathbb{C}^n$ to denote its Fourier spectrum,

$$\widehat{x}_i = \frac{1}{\sqrt{n}} \sum_{j=1}^{n} e^{-2\pi \mathbf{i} ij/n} x_j, \forall i \in [n].$$

Then the inverse transform is

$$x_j = \frac{1}{\sqrt{n}} \sum_{i=1}^{n} e^{2\pi \mathbf{i} ij/n} \widehat{x}_i, \forall j \in [n].$$

We define

$$\mathrm{Err}(x, k) = \min_{k-\mathrm{sparse}\ y} \|x - y\|_2.$$

14.3.2 Permutation and filter function

We use the same (pseudorandom) spectrum permutation as [HIKP12a],

Definition 14.3.1. Suppose σ^{-1} exists mod n. We define the permutation $P_{\sigma,a,b}$ by

$$(P_{\sigma,a,b}x)_i = x_{\sigma(i-a)}e^{-2\pi \mathbf{i}\sigma bi/n}.$$

We also define $\pi_{\sigma,b} = \sigma(i-b) \pmod n$. Then we have

Claim 14.3.1. $\widehat{P_{\sigma,a,b}x}_{\pi_{\sigma,b}(i)} = \widehat{x}_i e^{-2\pi \mathbf{i}\sigma ai/n}$.

$h_{\sigma,b}(i) = \mathrm{round}(\pi_{\sigma,b}(i)B/n)$ and $o_{\sigma,b}(i) = \pi_{\sigma,b}(i) - h_{\sigma,b}(i)n/B$. We say $h_{\sigma,b}(i)$ is the "bin" that frequency i is mapped into, and $o_{\sigma,b}(i)$ is the "offset".

We use the same filter function as [HIKP12a, PS15, CKPS16],

Definition 14.3.2. Given parameters $B \geq 1$, $\delta > 0$, $\alpha > 0$. We say that $(G, \widehat{G}') = (G_{B,\delta,\alpha}, \widehat{G}'_{B,\delta,\alpha}) \in \mathbb{R}^n$ is a filter function(Figure 14.1) if it satisfies the following properties

(I) $|\operatorname{supp}(G)| = O(\alpha^{-1}B\log(n/\delta))$,

(I) $\widehat{G}'_i = 1$, if $|i| \leq (1-\alpha)n/(2B)$,

(II) $\widehat{G}'_i = 0$, if $|i| \geq n/(2B)$,

(III) $\widehat{G}'_i \in [0,1]$, for all i,

(IV) $\left\|\widehat{G}' - \widehat{G}\right\|_\infty < \infty$.

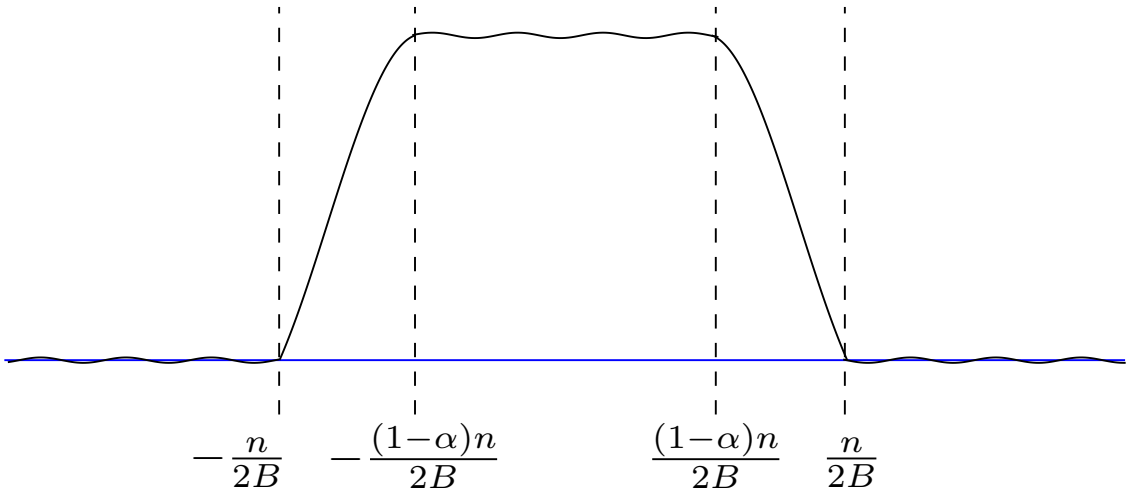


Figure 14.1: Filter $\widehat{G}'$

14.3.3 Collision event, large offset event, and large noise event

We use three types of events defined in [HIKP12a] as basic building blocks for analyzing Fourier set query algorithms. For any $i \in S$, we define three types of events associated with i and S and defined over the probability space induced by σ and b:

Definition 14.3.3 (Collision, large offset, large noise)**.** "Collision" event $E_{\text{coll}}(i)$: holds iff $h_{\sigma,b}(i) \in h_{\sigma,b}(S \backslash \{i\})$.

"Large offset" event $E_{\text{off}}(i)$: holds iff $|o_{\sigma,b}(i)| \geq (1-\alpha)n/(2B)$.

"Large noise" event $E_{\text{noise}}(i)$: holds iff

$$\mathbb{E}\left[\left\|\widehat{x}'_{h^{-1}_{\sigma,b}(h_{\sigma,b}(i)) \backslash S}\right\|_2^2\right] \geq \text{Err}^2(\widehat{x}', k)/(\alpha B).$$

Claim 14.3.2 (Claim 3.1 in [HIKP12a])**.** *For any $i \in S$, the event $E_{\text{coll}}(i)$ holds with probability at most $4|S|/B$.*

Claim 14.3.3 (Claim 3.2 in [HIKP12a]). *For any $i \in S$, the event $E_{\text{off}}(i)$ holds with probability at most α.*

Claim 14.3.4 (Claim 4.1 in [HIKP12a]). *For any $i \in S$, $\Pr[E_{\text{noise}(i)}] \leq 4\alpha$.*

Lemma 14.3.5 (Lemma 4.2 in [HIKP12a]). *Let $a \in [n]$ uniformly at random, B divide n, and the other parameters be arbitrary in*

$$\widehat{u} = \textsc{HashToBins}(x, \widehat{z}, P_{\sigma,a,b}, B, \delta, \alpha).$$

Then for any $i \in [n]$ with $j = h_{\sigma,b}(i)$ and none of $E_{\text{coll}}(i)$, $E_{\text{off}}(i)$ or $E_{\text{noise}}(i)$ holding,

$$\mathbb{E}\left[\left|\widehat{u}_j - \widehat{x}'_i e^{-\frac{2\pi \mathbf{i}}{n} a\sigma i}\right|^2\right] \leq 2\frac{\rho^2}{\alpha B}.$$

Lemma 14.3.6. *Suppose B divides n. The output $\widehat{u}$ of* HashToBins *satisfies*

$$\widehat{u}_j = \sum_{h_{\sigma,b}(i)=j} \widehat{(x-z)}_i (\widehat{G'_{B,\delta,\alpha}})_{-o_{\sigma,b}(i)} \omega^{a\sigma i} \pm \delta \|\widehat{x}\|_1.$$

Let $\zeta = |\{i \in \text{supp}(\widehat{z}) \mid E_{\text{off}}(i)\}|$. The running time of HashToBins *is $O(\frac{B}{\alpha}\log(n/\delta) + \|\widehat{z}\|_0 + \zeta \log(n/\delta))$.*

14.3.4 Iterative loop analysis

Iterative loop analysis for Fourier set query is more tricky than the classic set query, because in the Fourier case, hashing is not perfect, in the sense that by using spectrum permutation and filter function (as the counterpart of hashing techniques), one coordinate can non-trivially contribute to multiple bins. We give iterative loop induction in Lemma 14.3.7.

Lemma 14.3.7. *Given parameters $C \geq 1000$, $\gamma \leq 1/1000$. For any $k \geq 1, \epsilon \in (0,1)$, $R \geq 1$. For each $i \in [R]$, we define*

$$k_i = k\gamma^i, \epsilon_i = \epsilon(10\gamma)^i, \alpha_i = 1/(200i^3), B_i = C \cdot k_i/(\alpha_i^2 \epsilon_i).$$

For each $i \in [R]$:

If for all $j \leq [i-1]$ we have

$$\mathrm{supp}(\widehat{w}^{(j)}) \subseteq S_j, \quad |S_{j+1}| \leq k_{j+1}, \quad \widehat{z}^{(j+1)} = \widehat{z}^{(j)} + \widehat{w}^{(j)}, \quad \widehat{x}^{(j+1)} = \widehat{x} - \widehat{z}^{(j+1)},$$

and

$$\|\widehat{x}^{(j+1)}_{\overline{S}_{j+1}}\|_2^2 \leq (1+\epsilon_j)\|\widehat{x}^{(j)}_{\overline{S}_j}\|_2^2 + \epsilon_j \delta^2 n \|\widehat{x}\|_1^2.$$

Then, with probability $1 - 10\alpha_i/\gamma$, we have

$$\mathrm{supp}(\widehat{w}^{(i)}) \subseteq S_i, \quad |S_{i+1}| \leq k_{i+1}, \quad \widehat{z}^{(i+1)} = \widehat{z}^{(i)} + \widehat{w}^{(i)}, \quad \widehat{x}^{(i+1)} = \widehat{x} - \widehat{z}^{(i+1)},$$

and

$$\|\widehat{x}^{(i+1)}_{\overline{S}_{i+1}}\|_2^2 \leq (1+\epsilon_i)\|\widehat{x}^{(i)}_{\overline{S}_i}\|_2^2 + \epsilon_i \delta^2 n \|\widehat{x}\|_1^2.$$

Proof. We consider a particular step i. We can condition on $|S_i| \leq k_i$.

Collision. Using Claim 14.3.2, for any $t \in S_i$, the event $E_{\mathrm{coll}}(t)$ holds with probability at most

$$4|S_i|/B_i \leq \frac{4k_i}{Ck_i/(\alpha_i^2\epsilon_i)} = 4\alpha_i^2\epsilon_i/C \leq \alpha_i.$$

It means

$$\Pr_{\sigma,b}[E_{\mathrm{coll}}(t)] \leq \alpha_i.$$

Large offset. Using Claim 14.3.3, for any $t \in S_i$, the event $E_{\text{off}}(t)$ holds with probability at most α_i, i.e.

$$\Pr_{\sigma,b}[E_{\text{off}}(t)] \leq \alpha_i.$$

Large noise. Using Claim 14.3.4, for any $t \in S_i$,

$$\Pr_{\sigma,b}[E_{\text{noise}}(t)] \leq 4\alpha_i.$$

We say a coordinate t is "well isolated" if none of the above three events holding. By a union bound over the above three events, we have t is "well isolated" with probability at least $1 - 6\alpha_i$.

Therefore, each $t \in S_i$ lies in T_i with probability at least $1 - 6\alpha_i$. Then by Markov's inequality, we have $|S_i \backslash T_i| \leq \gamma k_i$ with probability $1 - 6\alpha_i/\gamma$. By definition $S_{i+1} = S_i \backslash T_i$, then we know that

$$|S_{i+1}| = |S_i \backslash T_i| \leq \gamma k_i \leq k_{i+1}.$$

Upper bound $\|\widehat{x}^{(i)}_{\overline{S}_i}\|_2^2$. By Lemma statement we have

$$\begin{aligned}
\|\widehat{x}^{(i)}_{\overline{S}_i}\|_2^2 &\leq (1+\epsilon_i)\|\widehat{x}^{(i)}_{\overline{S}_i}\|_2^2 + \epsilon_i \delta^2 n \|\widehat{x}\|_1^2 \\
&\leq (1+\epsilon_i)(1+\epsilon_{i-1})\|\widehat{x}^{(i-1)}_{\overline{S}_{i-1}}\|_2^2 + ((1+\epsilon_i)\epsilon_{i-1} + \epsilon_i)\delta^2 n \|\widehat{x}\|_1^2 \\
&\leq \cdots \\
&\leq \prod_{j=1}^{i}(1+\epsilon_j)\|\widehat{x}_{\overline{S}}\|_2^2 + \sum_{j=1}^{i} \epsilon_j \delta^2 n \|\widehat{x}\|_1^2 \prod_{l=j+1}^{i}(1+\epsilon_l) \\
&\leq 8(\|\widehat{x}_{\overline{S}}\|_2^2 + \delta^2 n \|\widehat{x}\|_1^2), \qquad (14.6)
\end{aligned}$$

where the third step refers to recursively apply the second step, the last step follows by a geometric sum.

For a fixed $t \in S_i$, let $j = h_{\sigma,b}(t)$. By Lemma 3.3, we have

$$\widehat{u}_j - \widehat{x}_t^{(i)}\omega^{a\sigma t} = \sum_{t' \in T_i} \widehat{G}'_{-o_\sigma(i')}\widehat{x}_{t'}^{(i)}\omega^{a\sigma t'} \pm \delta\|\widehat{x}\|_1$$

For each $t \in S_i$, we define set $Q_{i,t} = h_{\sigma,b}^{-1}(j)\backslash\{t\}$. Let T_i be the set of coordinates $t \in S_i$ such that $Q_{i,t} \cap S_i = \emptyset$. Then it is easy to observe that

$$\begin{aligned}\sum_{t\in T_i}\left|\sum_{t'\in Q_{i,t}} \widehat{G}'_{-o_\sigma(t)}\widehat{x}_{t'}^{(i)}\omega^{a\sigma t'}\right|^2 &= \sum_{t\in T}\left|\sum_{t'\in Q_{i,t}\backslash S} \widehat{G}'_{-o_\sigma(t')}\widehat{x}_{t'}^{(i)}\omega^{a\sigma t'}\right|^2 \\ &\le \sum_{t\in S}\left|\sum_{t'\in Q_{i,t}\backslash S} \widehat{G}'_{-o_\sigma(t')}\widehat{x}_{t'}^{(i)}\omega^{a\sigma t'}\right|^2\end{aligned}$$

We can calculate the expectation of $\|\widehat{x}_{T_i}^{(i)} - \widehat{w}^{(i)}\|_2^2$,

$$\begin{aligned}\mathop{\mathbb{E}}_{\sigma,a,b}\left[\left\|\widehat{x}_{T_i}^{(i)} - \widehat{w}^{(i)}\right\|_2^2\right] &= \mathop{\mathbb{E}}_{\sigma,a,b}\left[\sum_{t\in T_i}\left\|\widehat{x}_t^{(i)} - \widehat{w}_t^{(i)}\right\|_2^2\right] \\ &\le \sum_{i\in S} 2\mathop{\mathbb{E}}_{\sigma,a,b}\left[\left|\sum_{i'\in Q_i\backslash S}\widehat{G}'_{-o_\sigma(i)}\omega^{a\sigma i}\right|^2\right] + \delta^2\|\widehat{x}\|_1^2 \\ &\le \sum_{t\in S_i}(\frac{1}{B_i}\|\widehat{x}_{\overline{S}_i}^{(i)}\|_2^2 + \delta^2\|\widehat{x}\|_1^2) \\ &\le \frac{|S_i|}{B_i}\|\widehat{x}_{\overline{S}_i}^{(i)}\|_2^2 + \delta^2|S_i|\cdot\|\widehat{x}\|_1^2 \\ &\le \frac{\epsilon_i\alpha_i^2}{C}\|\widehat{x}_{\overline{S}_i}^{(i)}\|_2^2 + \delta^2|S_i|\cdot\|\widehat{x}\|_1^2,\end{aligned}$$

where the last step follows from $|S_i| \le k_i$ and $B_i = C \cdot k_i/(\alpha_i^2\epsilon_i)$.

Then, using Markov's inequality, we have,

$$\Pr\left[\left\|\widehat{x}^{(i)}_{T_i} - \widehat{w}^{(i)}\right\|_2^2 \geq \frac{\epsilon_i\alpha_i}{C}\|\widehat{x}^{(i)}_{\overline{S}_i}\|_2^2 + \delta^2\frac{|S_i|}{\alpha_i}\|\widehat{x}\|_1^2\right] \leq \alpha_i.$$

Note that

$$\begin{aligned}\frac{\epsilon_i\alpha_i}{C}\|\widehat{x}^{(i)}_{\overline{S}_i}\|_2^2 + \delta^2\frac{|S_i|}{\alpha_i}\|\widehat{x}\|_1^2 \leq& \frac{\epsilon_i}{C}\|\widehat{x}^{(i)}_{\overline{S}_i}\|_2^2 + \delta^2\frac{|S_i|}{\alpha_i}\|\widehat{x}\|_1^2\\ \leq& \frac{\epsilon_i}{C}\|\widehat{x}^{(i)}_{\overline{S}_i}\|_2^2 + \frac{\epsilon_i}{C}\delta^2 B_i\|\widehat{x}\|_1^2\\ \leq& \frac{\epsilon_i}{C}\|\widehat{x}^{(i)}_{\overline{S}_i}\|_2^2 + \frac{\epsilon_i}{C}\delta^2 n\|\widehat{x}\|_1^2\\ \leq& \frac{\epsilon_i}{20}(\|\widehat{x}^{(i)}_{\overline{S}_i}\|_2^2 + \delta^2 n\|\widehat{x}\|_1^2),\end{aligned}$$

where the first step follows by $\alpha_i \leq 1$, the second step follows by $|S_i| \leq k_i = \epsilon_i B_i\alpha_i^2/C$, the third step follows by $B_i \leq n$, the last step follows by $C \geq 1000$.

Thus, we have

$$\Pr\left[\left\|\widehat{x}^{(i)}_{T_i} - \widehat{w}^{(i)}\right\|_2^2 \leq \frac{\epsilon_i}{20}(\|\widehat{x}^{(i)}_{\overline{S}_i}\|_2^2 + \delta^2 n\|\widehat{x}\|_1^2)\right] \geq 1 - \alpha_i. \tag{14.7}$$

Recall that $\widehat{w}^{(i)} = \widehat{z}^{(i+1)} - \widehat{z}^{(i)} = \widehat{x}^{(i)} - \widehat{x}^{(i+1)}$. It is obvious that $\mathrm{supp}(\widehat{w}^{(i)}) \subseteq T_i$.

Conditioning on all coordinates in T_i are well isolated and Eq. (14.7) holds, we have

$$\begin{aligned}
\|\widehat{x}^{(i+1)}_{\overline{S}_{i+1}}\|_2^2 &= \|(\widehat{x}^{(i)} - \widehat{w}^{(i)})_{\overline{S}_{i+1}}\|_2^2 \\
&= \|\widehat{x}^{(i)}_{\overline{S}_{i+1}} - \widehat{w}^{(i)}_{\overline{S}_{i+1}}\|_2^2 \\
&= \|\widehat{x}^{(i)}_{\overline{S}_{i+1}} - \widehat{w}^{(i)}\|_2^2 \\
&= \|\widehat{x}^{(i)}_{\overline{S}_i \cup T_i} - \widehat{w}^{(i)}\|_2^2 && \text{by } S_i = T_i \cup S_{i+1} \\
&= \|\widehat{x}^{(i)}_{\overline{S}_i}\|_2^2 + \|\widehat{x}^{(i)}_{T_i} - \widehat{w}^{(i)}\|_2^2 && \text{by } \overline{S}_i \cap T_i = \emptyset \\
&\leq \|\widehat{x}^{(i)}_{\overline{S}_i}\|_2^2 + \epsilon_i(\|\widehat{x}^{(i)}_{\overline{S}_i}\|_2^2 + \delta^2 n \|\widehat{x}\|_1^2) && \text{by Eq. (14.7)} \\
&= (1+\epsilon_i)\|\widehat{x}^{(i)}_{\overline{S}_i}\|_2^2 + \epsilon_i \delta^2 n \|\widehat{x}\|_1^2. && \text{by definition } \mu_{(i)}
\end{aligned}$$

□

14.3.5 Main result

In this subsection, we give the main result as the following theorem.

Theorem 14.3.8 (Main result)**.** *Given a vector $x \in \mathbb{C}^n$, for every $\epsilon \in (0,1)$ and $k \geq 1$, any $S \subseteq [n]$, $|S| = k$, there exists an algorithm (Algorithm 14.2) that takes*

$$O(\epsilon^{-1} k \log(n/\delta))$$

samples, runs in

$$O(\epsilon^{-1} k \log(n/\delta))$$

time, and outputs a vector $x' \in \mathbb{C}^n$ such that

$$\|(x' - \widehat{x})_S\|_2^2 \leq \epsilon \|\widehat{x}_{[n]\setminus S}\|_2^2 + \delta \|\widehat{x}\|_1^2,$$

holds with probability at least $9/10$.

Proof. **Proof of Sample Complexity.** The sample complexity of ESTIMATION is

$$\sum_{i=1}^{R}(B_i/\alpha_i)\log(n/\delta) = \epsilon^{-1}k\log(n/\delta).$$

Proof of Success Probability.

The failure probability is

$$\sum_{i=1}^{R} 10\alpha_i/\gamma < 1/10.$$

Proof of Final Error

$$\begin{aligned}
\|\widehat{x}_S - \widehat{z}^{(R+1)}\|_2^2 &= \sum_{i=1}^{R}\|\widehat{x}_{T_i}^{(i)} - \widehat{w}^{(i)}\|_2^2 \\
&\le \sum_{i=1}^{R} k_i\mu^{(i)}/20 && \text{by Eq. (14.7)} \\
&\le \sum_{i=1}^{R}\epsilon_i(\|\widehat{x}_{\overline{S}_i}^{(i)}\|_2^2 + \delta^2 n\|\widehat{x}\|_1^2)/20 && \text{by Definition of } \mu^{(i)} \\
&\le \sum_{i=1}^{R}\epsilon_i\cdot 10(\|\widehat{x}_{\overline{S}}\|_2^2 + \delta^2 n\|\widehat{x}\|_1^2)/20 && \text{by Eq. (14.6)} \\
&\le \epsilon(\|\widehat{x}_{\overline{S}}\|_2^2 + \delta^2 n\|\widehat{x}\|_1^2). && \text{by geometric sum}
\end{aligned}$$

□

Algorithm 14.2 Fourier Set Query Algorithm

procedure FOURIERSETQUERY(x, S, ϵ, k) ▷ Theorem 14.3.8
  $\gamma \leftarrow 1/1000, C \leftarrow 1000, \widehat{z}^{(1)} \leftarrow 0, S_1 \leftarrow S$
  for $i = 1 \to R$ **do**
    $k_i \leftarrow k\gamma^i$, $\epsilon_i \leftarrow \epsilon(10\gamma)^i$, $\alpha_i \leftarrow 1/(100i^3)$, $B_i \leftarrow C \cdot k_i/(\alpha_i^2 \epsilon_i)$
    $\widehat{w}^{(i)}, T_i \leftarrow$ ESTIAMTEVALUES($x, \widehat{z}^{(i)}, S_i, B_i, \delta, \alpha_i$) ▷ $\widehat{w}^{(i)}$ is $|T_i|$-sparse
    $S_{i+1} \leftarrow S_i \backslash T_i$
    $\widehat{z}^{(i+1)} \leftarrow z^{(i)} + \widehat{w}^{(i)}$
  end for
  return $\widehat{z}^{(R+1)}$
end procedure
procedure ESTIMATEVALUES($x, \widehat{z}, S, B, \delta, \alpha$) ▷ Lemma 14.3.7
  Choose $a, b \in [n]$ uniformly at random
  Choose σ uniformly at random from the set of odd numbers in $[n]$
  $\widehat{u} \leftarrow$ HASHTOBINS($x, \widehat{z}, P_{\sigma,a,b}, B, \delta, \alpha$)
  $\widehat{w} \leftarrow 0, T \leftarrow \emptyset$
  for $t \in S$ **do**
    if t is isolated from other coordinates of S **then** ▷ $h_{\sigma,b}(t) \notin h_{\sigma,b}(S\backslash\{t\})$
      if no large offset **then** ▷ $|o_{\sigma,b}(t)| < (1-\alpha)n/(2B)$
        $\widehat{w}_t \leftarrow \widehat{u}_{h_{\sigma,b}(t)} e^{\frac{2\pi \mathbf{i}}{n}\sigma a t}$
        $T \leftarrow T \cup \{t\}$
      end if
    end if
  end for
  return $\widehat{w}, T$
end procedure
procedure HASHTOBINS($x, \widehat{z}, P_{\sigma,a,b}, B, \delta, \alpha$)
  Compute $\widehat{y}_{jn/B}$ for $j \in [B]$, where $y = G_{B,\alpha,\delta} \cdot (P_{\sigma,a,b} x)$
  Compute $\widehat{y}'_{jn/B} = \widehat{y}_{jn/B} - (\widehat{G'_{B,\alpha,\delta}} * \widehat{P_{\sigma,a,b} z})_{jn/B}$
  return $\widehat{u}_j = \widehat{y}'_{jn/B}$
end procedure

Chapter 15

Robust Sparse Recovery via M-estimators

We consider the extensively studied problem of sparse recovery, where the error is measured with respect to M-estimators, such as the Huber or the Tukey loss functions. All previous works have studied the problem under the ℓ_p norms. Thus, our work is the first to explore the more robust error measures of M-estimators, which are very popular in statistics. In specific, given linear measurements $y = Ax$, we want to output a k-sparse vector $x' \in \mathbb{R}^n$ such that

$$\|x' - x\|_M \leq (1+\epsilon) \min_{k-\text{sparse } y} \|y - x\|_M,$$

where $\|x\|_M$ is corresponds to some M-estimator of x. We show that the measurement complexity of different versions of the sparse recovery problem under these more sophisticated error measures does not change, igniting thus an interesting new line of research.

15.1 Introduction

The mathematical framework of compressed sensing, or sparse recovery, aims to reconstruct an approximately k-sparse vector $x \in \mathbb{R}^n$ from linear measurements $y = Ax$. This model can capture many significant problems, which appear in a variety of fields, such as Discrete Signal Processing[LDP07b], Machine Learning [LDW+18], Statistics [Rip04] etc. Depending on the application, the measurement matrix A is drawn from some random matrix ensemble, or can be designed without constraints.

What is important and interesting about this framework, is that it enables us to bypass the Shannon/Nyquist sampling theorem in the case of x being k-sparse or approximately k-sparse. The Shannon/Nyquist theorem tells us that in order to preserve information when uniformly sampling a signal, we must sample at least two times faster than its bandwidth, thus putting a lower bound on the number of measurements we need to obtain.

The applications of sparse recovery to Machine Learning are numerous, see [CDHB09, VGT12, DPFJ16, CMM11, SH11]. Moreover, deep learning approaches combined with sparse recovery techniques in [MPB15, ABEZ16, KLT+16, LDW+18, ZLSD18, BJPD17, BPD18] are proposed, in order to more finely reconstruct signals that are approximately k-sparse, with focus on images. In [VGT12], the authors employ compressed sensing techniques to develop routines for brain mapping situations in neuroscience, where the goal is to solve a support-recovery type of problem for functional MRI (fMRI) data. In [DPFJ16], the authors extend compressed sensing to signals that are represented via a graphical model.

On a high-level, the reason why machine learning applications benefit from sparse recovery is because in such applications we aim to find a mapping between sensory data and

data semantics (the label). In order to do that, a standard approach is to exploit a set of features which captures important data characteristics, and is a closely associated to data labels. We thus transform the raw data into a feature space, where the similarity between data points corresponds to the similarity between features.

The work on compressed sensing is vast, initiated with the seminal paper of Candes and Tao [CRT06b]. The authors show that ℓ_1 minimization satisfies the so-called ℓ_2/ℓ_1 guarantee, when the matrix A is chosen from a Gaussian ensemble. Other important guarantees, such as ℓ_1/ℓ_1 and ℓ_2/ℓ_2 have been extensively studied. The authors in [BIR08b, IR08b] showed that $O(\epsilon^{-1}k\log(n/k))$ measurements suffice for a uniform $(1+\epsilon)$ ℓ_1/ℓ_1 guarantee. It is known that uniform reconstruction (i.e. deterministic algorithms) is not possible under ℓ_2/ℓ_2, so one has to resort to randomized algorithm. In [PW11] Price and Woodruff give a variety of interesting results for the randomized cases of ℓ_2/ℓ_2 and ℓ_1/ℓ_1 sparse recovery, where they also show a difference between the exactly k-sparse and the $O(k)$-sparse case. More precisely, they show that if one is allowed to output a $2k$-sparse vector, the dependence on ϵ is strictly better than if one is allowed to output an exactly k-sparse vector. This fact, apart from being an interesting phenomenon, is useful in applications such as imaging, where we are not constrained to output an exactly k-spare vector. The state of the art for ℓ_2/ℓ_2 is [GLPS10], where the authors show that $O((k/\epsilon)\log(n/k))$ measurements are sufficient, and also give a sub-linear time recovery procedure.

All previous work was focused on ℓ_p/ℓ_p, where the error is measured with respect to some ℓ_p norm. In applications, sometimes these error measurements might be insufficient. For example, in the presence of outliers the strongest of the above ℓ_2/ℓ_2 guarantees, behaves poorly. For that, the Huber norm is introduced [Hub64], which combines an ℓ_2 norm for

small x with an ℓ_1 norm for large x. As mentioned in [CKY97, GS+99] the Huber norm is of particular interest, because it is "recommended for almost all situations", and because it is "most robust" with respect to the presence of outliers and to the number of iterations than the ℓ_2 norm. The aforementioned facts, along with its smoothness properties, have made the Huber norm useful in Machine learning applications. Moreover, while some measures, such as ℓ_1 , treat small residuals with the same importance as large residuals, it is often more appropriate to use Huber norm, since relatively small noise insensitive misfit measures can yield far more stable estimates in several cases, such as in geophysics applications [GS+99].

Recently, [SK18] studies a formulation of robust sparse recovery, where the noise is sparse but adversarial.

Roadmap. Section 15.1.1 provides a summary of our techniques. We present our main result in Section 15.1.2. We provide a complete proof for Huber set query and a short proof sketch for Huber sparse recovery in Section 15.2. We made a conclusion in Section 15.3. We deferred the complete proof for Huber, Tukey and other results into later sections.

Notations. We use $[n]$ to denote $\{1, 2, \cdots, n\}$. For a vector $x \in \mathbb{R}^n$, we use $\|x\|_1$ to denote its entry-wise ℓ_1 norm. We use $\|x\|_2$ to denote its entry-wise ℓ_2 norm. We use $\mathrm{supp}(x) \subseteq [n]$ to denote a set of non-zero indices. We use $\|x\|_0$ to denote the number of non-zero entries in x. ($\|x\|_0 = |\mathrm{supp}(x)|$) Let x_{-k} denote the vector obtained by zeroing out largest k coordinates (in absolute) of x. We will sometimes refer to x_{-k} as the tail of x, whereas we will refer to $x - x_{-k}$ as the head of x.

15.1.1 Our techniques

All of our results are based on the ubiquitous CountSketch data structure, which hashes every coordinate $i \in [n]$ to one of $O(k)$ buckets, and repeats $O(\log n)$ times. This data structure can be used to solve the ℓ_p heavy hitters problems, as well as the sparse recovery under the ℓ_2/ℓ_2 guarantee [PW11]. It is not hard to see that ℓ_1/ℓ_1 can also be achieved via the same data structure. Interestingly, our theorems indicate that the Huber and Tukey guarantees can also be obtained via CountSketch.

In specific, we use the CountSketch to obtain estimates for all $i \in [n]$, and then form the set S of the k coordinates with the largest estimates, as in previous work. Now, it might be the case that some elements in the head are not included in S, which means that they have been displaced by some other elements in the tail. In the case that the threshold τ for the Huber function is "small" enough, i.e. less than every $|x_j|$, for every $j \in S$ and $j \in \text{head}(k)$, we can bound the error similarly to the ℓ_1/ℓ_1 case. An analogous thing happens in the case that τ, is large enough, i.e. $\tau \geq \|x\|_\infty$, where can bound the error similarly to the ℓ_2/ℓ_2 case. Things are more complicated in the case that τ falls somewhere in between S and $\text{head}(k)$, where our guarantee needs to bound the ℓ_1 norm on the left-hand side, by the ℓ_2 norm on the right-hand side. This at first glance seems to be something impossible to do, but a careful glance reveals an interesting property: in the case of $\tau = 1$, if we estimate every $|x_i|$ up to Δ and if Δ is, then every displaced element in the head should should be $1 + \Delta = (1 + \Theta(\Delta))^2$, so we can hope to employ a similar argument to the ℓ_2/ℓ_2, as this property implicitly allows us to reduce the problem to the case where the right-hand side and the left-hand side are measured with respect to the ℓ_2 norm. If Δ is large enough then intuitively the mass of coordinates displaced "does not matter". because they were small

compared to the noise, and we can charge them to the tail. Of course, a careful combination of arguments and techniques needs to be addressed for the analysis to work out.

We note that the algorithm we suggest is not novel per se, but we rather indicate that the classic CountSketch data structure gives also the stronger Huber guarantee.

15.1.2 Our results

In the field of sparse recovery / compressive sensing, researchers usually consider ℓ_p norms as error estimates. Two well-known problems, arise when setting $p = 2$, giving birth to ℓ_2/ℓ_2 and ℓ_1/ℓ_1 problems.

$$\ell_2/\ell_2 : \|x - x'\|_2 \leq (1+\epsilon)\|x_{-k}\|_2,$$
$$\ell_1/\ell_1 : \|x - x'\|_1 \leq (1+\epsilon)\|x_{-k}\|_1,$$

where x is the ground-truth vector, x_{-k} is the vector that is obtained after zeroing out the largest k coordinates (in absolute) of x, x' is the k-sparse we output and $\epsilon \in (0,1)$ is the accuracy parameter.

We consider the problem of sparse recovery, when the error is measured with respect to an M-estimators. M-estimators are specified by a function $H : \mathbb{R} \to \mathbb{R}_{\geq 0}$ where $H(x) = H(-x)$, $H(0) = 0$, and $H()$ is non-decreasing $|x|$. We use $\|\cdot\|_H$ to denote a new "norm" (However, it might not be true norms in fact, because it doesn't satisfies triangle inequality). For a vector $x \in \mathbb{R}^n$, we use $\|x\|_H$ to denote $\sum_{i=1}^n H(x_i)$.

Huber function [Hub64] and Tukey function [Tuk60] have been two well-known M-estimators, which have been proven useful to researchers for more than 50 years. In this

work, we will mainly focus on these two estimators, which are also the most common ones. We first provide the definition of Huber function and Tukey function.

Definition 15.1.1 (Huber function)**.** Given parameter $\tau > 0$,

$$H(x) = \begin{cases} |x|^2/\tau & \text{if } |x| \leq \tau; \\ |x| & \text{if } |x| > \tau. \end{cases}$$

Note that an alternative definition of Huber is $|x|^2/2\tau$ if $|x| \leq \tau$ and $|x| - \tau/2$ if $|x| > \tau$. Our techniques work for both of those definitions. This is also being observed in some other problems, e.g. regression [CW15b] and low-rank approximation [CW15a].

We define Tukey function [Tuk60] as follows

Definition 15.1.2 (Tukey function)**.** Given parameter $\tau > 0$,

$$H(x) = \begin{cases} 1 - (1 - (x/\tau)^2)^3, & \text{if } |x| \leq \tau; \\ 1, & \text{if } |x| > \tau. \end{cases}$$

It is not hard to observe that both the Huber and Tukey function satisfy three basic properties: they vanish at zero, i.e., $H(0) = 0$, are symmetric and non-decreasing.

15.1.2.1 Set-query

We start by presenting a problem called "set query" which plays a crucial role in solving sparse recovery problem. Set query problem is first formally defined by Price [Pri11] and recently got studied in the context of Sparse Fourier transform[Kap17].

Definition 15.1.3 (Set-query)**.** Let $A \in \mathbb{R}^{m \times n}$ be some implicit matrix we want to design. For any accuracy parameter ϵ and set S with $|S| = k$, our goal is to read Ax and output

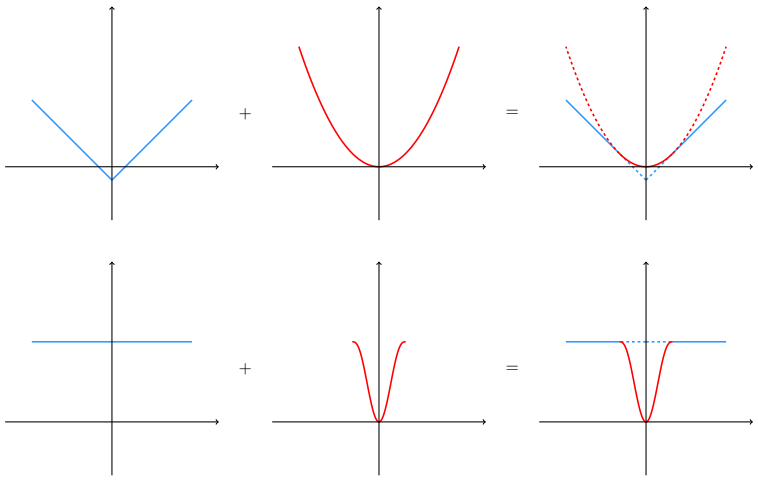


Figure 15.1: (a) Huber function (top) (b) Tukey function (bottom)

vector $x' \in \mathbb{R}^k$ such that

$$\|x' - x_S\|_H \leq \epsilon \cdot \|x_{[n]\setminus S}\|_H,$$

where $\|x\|_H = \sum_i H(x_i)$ and H is some function.

We give some formal definitions of what we are able to achieve in terms of measurements, column sparsity and decoding time.

Definition 15.1.4 (Measurements, column sparsity, decoding time)**.** There are several things we want to optimize,

1. The number of measurements, is the number of rows of A.
2. Column sparsity is, for any $u \cdot e_i$, the time we need to compute $A \cdot u \cdot e_i$, where $u \in \mathbb{R}$ and $e_i \in \mathbb{R}^n$ has 1 in the i-th coordinate and 0 everywhere else.
3. Decoding time means once Ax is given, the time we need to output x'.

Note that the decoding time is not necessarily larger than the number of measurements.

Our set-query results on M-estimators are given in the following theorem.

Theorem 15.1.1 (Set-query)**.** *Let function $H : \mathbb{R} \to \mathbb{R}_{\geq 0}$ be either Huber function or Tukey function. There is an implicit matrix $A \in \mathbb{R}^{m \times n}$ with $m = O(\epsilon^{-1} k \log k)$ and $O(\log k)$ column sparsity. For any $S \subseteq [n]$ such that $|S| = k$, there is an algorithm which given Ax, runs in $O(k \log k)$ decoding time, and outputs a vector $x' \in \mathbb{R}^k$ such that*

$$\|x' - x_S\|_H \leq \epsilon \cdot \|x_{[n] \setminus S}\|_H$$

holds with probability $9/10$.

Remark 15.1.1. Although the matrix has $m = O(\epsilon^{-1} k \log k)$, but our algorithm doesn't need to use all the measurements for the set-query algorithm. The way our algorithm is designed implies that the decoding time is independent of accuracy parameter ϵ.

For a vector $x \in \mathbb{C}^n$, we define its Fourier transform $\widehat{x} \in \mathbb{C}^n$ where $\forall i \in [n]$, $\widehat{x}_i = \sum_{j=1}^n \frac{1}{\sqrt{n}} x_j e^{-2\pi \mathbf{i} ij/n}$. Similarly, we define the inverse Fourier transform of $\widehat{x} \in \mathbb{C}^n$ to be x where $\forall j \in [n]$, $x_j = \sum_{i=1}^n \frac{1}{\sqrt{n}} \widehat{x}_i e^{2\pi \mathbf{i} ij/n}$. Now, we're ready to present our Fourier set query result,

Theorem 15.1.2 (Fourier set query)**.** *Given a vector $x \in \mathbb{C}^n$, for every $\epsilon \in (0, 1)$ and $k \geq 1$, any $S \subseteq [n]$, $|S| \leq k$, there exists an algorithm (Algorithm 15.3) that takes $O(\epsilon^{-1} k \log k \cdot \log(n/\delta))$ samples, runs in $O(\epsilon^{-1} k \log k \cdot \log(n/\delta))$ time, and outputs a vector $x' \in \mathbb{C}^n$ such that*

$$\|(x' - \widehat{x})_S\|_H \leq \epsilon \|\widehat{x}_{[n] \setminus S}\|_H + \delta \|\widehat{x}\|_H$$

holds with probability at least $1 - 1/\operatorname{poly}(k)$.

We can think of $\|\widehat{x}\|_H$ being bounded by $\mathrm{poly}(n)$. Then we can set $\delta = 1/\mathrm{poly}(n)$. This assumption, along with the additional δ factor is common in Fourier literature, e.g. [HIKP12a, PS15, CKPS16].

15.1.2.2 Sparse Recovery

We first present k-sparse recovery results under Huber/Tukey function. Our algorithm achieves $O(\epsilon^{-2}k\log n)$ measurements which matches the state-of-the-art ℓ_2/ℓ_2 algorithm (see [PW11]).

Theorem 15.1.3 (Main result, Sparse recovery, Optimal measurements)**.** *Let H denote either Huber function (see Def. 15.1.1) or Tukey function (see Def. 15.1.2). There exists a distribution of $\Phi \in \mathbb{R}^{m\times n}$ with $m = O(\epsilon^{-2}k\log n)$ and $O(\log n)$ column sparsity, and a recovery algorithm, such that for each $x \in \mathbb{R}^n$, $k \in [n]$ and $\epsilon \in (0,1)$, the recovery algorithm takes Φx as its input, runs in $O(n\log n)$ time, and outputs k-*sparse *vector $x' \in \mathbb{R}^n$ so that with probability $1 - 1/\mathrm{poly}(n)$,*

$$\|x' - x\|_H \leq (1+\epsilon) \min_{k-\text{sparse } y} \|y - x\|_H.$$

15.2 Algorithm and Analysis

We start by defining a well-known tool called Count-Sketch [CCF02].

Definition 15.2.1 (Count-Sketch [CCF02])**.** A count sketch data structure with bucket size B and replicates R corresponds to a measurement matrix $\Phi \in R^{BR\times n}$. Matrix Φ is a concatenation of $\Phi^{(1)}, \ldots, \Phi^{(R)}$, where each $\Phi^{(r)} \in \mathbb{R}^{B\times n}$ ($r \in [R]$) is defined by a pariwise hash function $h_i : [n] \to [B]$ and a pairwise sign function $\sigma_i : [n] \to \{-1, 1\}$, such that for

each $j \in [B]$ and $i \in [n]$,

$$\Phi_{j,i}^{(r)} = \begin{cases} \sigma_r(i) & \text{if } h_r(i) = j, \\ 0 & \text{otherwise.} \end{cases}$$

The data structure maintains the value of Φx and for each $i \in [n]$, it gives

$$x_i' = \text{median}\left(\sigma_1(i) \cdot (\Phi^{(1)}x)_{h_1(i)}, \ldots, \sigma_R(i) \cdot (\Phi^{(R)}x)_{h_R(i)}\right)$$

as estimation to x_i.

15.2.1 Set query algorithm for Huber

Our set query algorithm for Huber loss function is presented in Algorithm 15.1. The algorithm uses count-sketch data structure with bucket number $B = O(\epsilon^{-1}k)$ and replicate number $R = O(\log k)$.

Algorithm 15.1 Set Query Algorithm

procedure SETQUERY(n, k, ϵ, S, x) ▷ Theorem 15.2.1
  $B \leftarrow O(\epsilon^{-1}k)$
  $R \leftarrow O(\log k)$
  Construct count sketch with bucket size B and replicates R according to Definition 15.2.1
  $y \leftarrow \Phi x$ ▷ In streaming setting we maintain y when receiving updates to x
  for $i \in S$ **do**
    $x_i' \leftarrow \text{median}_{r \in [R]} \sigma_r(i) y_{h_r(i)}^{(r)}$ ▷ $y = (y^{(1)}, \ldots, y^{(R)})$
  end for
  return x_S'
end procedure

Now we provide a correctness proof for the set-query algorithm under Huber function. For simplicity we let $\tau = 1$, but our proof extends to arbitrary τ. In order to express $\|x_{[n]\setminus S}\|_H$

in a handy form, we first partition the coordinates in $[n]\backslash S$ into P and Q, where P contains small coordinates whose absolute values are less than τ and Q contains those who are larger than τ. By guarantee of the count-sketch, we can bound the error of each estimate for coordinates in S. We then bound the error in Huber loss function by discussing the absolute error is either less than or greater than τ.

Theorem 15.2.1 (Set-query, Huber case of Theorem 15.1.1)**.** *Let function H be defined as Definition 15.1.1. There is an algorithm (Algorithm 15.1) that takes $O(\epsilon^{-1}k\log k)$ samples/times and outputs a vector $x' \in \mathbb{R}^n$ such that*

$$\|x'_S - x_S\|_H \leq \epsilon \cdot \|x_{[n]\backslash S}\|_H$$

holds with probability $1 - 1/\operatorname{poly}(k)$.

Proof. We choose $B = \epsilon^{-1}k$. We define set $P, Q \subset [n]\backslash S$ as follows

$$P = \{i \mid |x_i| \leq 1, i \in [n]\backslash S\}, Q = \{i \mid |x_i| \geq 1, i \in [n]\backslash S\}.$$

We define $P_{B/6}$ to be top $B/6$ coordinates in P, and define $P' = P\backslash P_{B/6}$. Similarly we define $Q_{B/6}$ to be top $B/6$ coordinates in Q, and define $Q' = Q\backslash Q_{B/6}$.

Let $R = O(\log k)$. For each $i \in S$, we estimate x_i as $x'_i = \operatorname{median}_{j\in[R]}\{y^{(j)}_{h_j(i)}\}$. We can prove that with probability $1 - 1/\operatorname{poly}(k)$,

$$|x'_i - x_i| \leq \sqrt{(\|x_{P'}\|_2^2 + \|x_{Q'}\|_2^2)/B}.$$

Note that $\|x_{\overline{S}}\|_H = \|x_P\|_2^2 + \|x_Q\|_1$.

We prove the final result by considering two cases:

Case (1)

$$\sqrt{(\|x_{P'}\|_2^2 + \|x_{Q'}\|_2^2)/B} \leq 1;$$

Case (2)

$$\sqrt{(\|x_{P'}\|_2^2 + \|x_{Q'}\|_2^2)/B} > 1.$$

Case (1) $\sqrt{(\|x_{P'}\|_2^2 + \|x_{Q'}\|_2^2)/B} \leq 1$.

In the following, we prove that $\|x_{Q'}\|_2^2 \leq \|x_{Q'}\|_1$. First, note that

$$\|x_{Q'}\|_2 \leq \sqrt{\|x_{P'}\|_2^2 + \|x_{Q'}\|_2^2} \leq \sqrt{B}.$$

By Lemma 15.5.2, we can also bound $\|x_{Q'}\|_2$ to be

$$\|x_{Q'}\|_2 \leq \frac{1}{\sqrt{B}}\|x_Q\|_1.$$

Combining these two inequality on $\|x_{Q'}\|_2$, we get

$$\|x_{Q'}\|_2^2 \leq \sqrt{B} \cdot \frac{1}{\sqrt{B}}\|x_Q\|_1 \leq \|x_Q\|_1.$$

By definition of Huber norm,

$$\begin{aligned}\|x'_S - x_S\|_H &\leq \frac{k}{B}(\|x_{P'}\|_2^2 + \|x_{Q'}\|_2^2) \\ &\leq \frac{k}{B}(\|x_P\|_2^2 + \|x_Q\|_1) \\ &= \epsilon\|x_{\overline{S}}\|_H.\end{aligned}$$

Case (2) $\sqrt{\frac{\|x_{P'}\|_2^2 + \|x_{Q'}\|_2^2}{B}} > 1$.

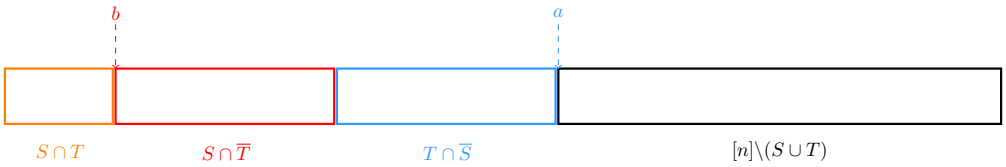


Figure 15.2: $\|x_{S\cap\overline{T}}\|_H - \|x_{T\cap\overline{S}}\|_H$ is the displacement error

The error is $\|x'_S - x_S\|_H \leq \frac{k}{\sqrt{B}}(\|x_{P'}\|_2 + \|x_{Q'}\|_2)$. We can bound $\|x_{Q'}\|_2 \leq \frac{1}{\sqrt{B}}\|x_Q\|_1$ by Lemma 15.5.2. Now trying to bound $\|x_{P'}\|_2$. If $\|x_{P'}\|_2 \leq \|x_{Q'}\|_2$ we are done. Otherwise we have $\|x_{P'}\|_2 \geq \sqrt{B/2}$ and so $\|x_{P'}\|_2 \leq \frac{\|x_{P'}\|_2}{\sqrt{B/2}}\|x_{P'}\|_2 \leq \sqrt{2/B}\|x_P\|_2^2$. Either case we can finally bound $\|x'_S - x_S\|_H$ by $2\epsilon\|x_{\overline{S}}\|_H$. □

15.2.2 Proof of Theorem 15.1.3

Algorithm 15.2 Sparse Recovery Algorithm

procedure SPARSERECOVERY(n, k, ϵ, x)
  $B \leftarrow O(\epsilon^{-2}k)$
  $R \leftarrow O(\log n)$
  Construct count sketch with bucket size B and replicates R according to Definition 15.2.1
  $y \leftarrow \Phi x$ ▷ In streaming setting we maintain y when receiving updates to x
  for $i = 1 \to n$ **do**
    $x'_i \leftarrow \text{median}_{r\in[R]}\, \sigma_r(i) y^{(r)}_{h_r(i)}$ ▷ $y = (y^{(1)}, \ldots, y^{(R)})$
  end for
  Let T denote the set of coordinates that are k largest (in absolute) of x'
  return x'_T
end procedure

The high-level idea. Let S denote the top-k largest coordinates in absolute. Let

T denote the set coordinates the algorithm outputs. Then we can rewrite $\|x - x'_T\|_H$ as

$$\begin{aligned}
&\|x - x'_T\|_H \\
&= \|(x - x')_T\|_H + \|x_{\overline{T}}\|_H \\
&= \|(x - x')_T\|_H + (\|x_{\overline{T}}\|_H - \|x_{\overline{S}}\|_H) + \|x_{\overline{S}}\|_H \\
&= \underbrace{\|(x - x')_T\|_H}_{\text{set-query}} + \underbrace{(\|x_{S\cap\overline{T}}\|_H - \|x_{T\cap\overline{S}}\|_H)}_{\text{displacement}} + \|x_{\overline{S}}\|_H.
\end{aligned}$$

Using Set-query, we know that $\|(x - x')_T\|_H \leq (\epsilon/2)\|x_{\overline{S}}\|_H$. Now the question is how to bound "displacement" by $(\epsilon/2)\|x_{\overline{S}}\|_H$ and we explain the formal proof in next a few pages.

Proof. of Theorem 15.1.3. Our algorithm is presented in Algorithm 15.2. Let x_{-k} denotes the vector x with absolute largest k coordinates removed. We use count sketch to give each x_i (where $i \in [n]$) an estimate. In particular, we choose the number of buckets $B = O(\epsilon^{-2}k)$ and the number of duplicates $R = O(\log n)$ for the count sketch data structure. Let x'_i denote the estimate obtained by the count sketch for x_i. Let T denote the largest k coordinates in absolute value in x'. The recovery algorithm outputs x'_T as a k-sparse approximation to x.

Note that the count sketch data structure takes $B \cdot R = O(\epsilon^{-2}k \log n)$ measurements. Its update time is $O(R) = O(\log n)$ time, and the time to obtain x' is $O(n \cdot R) = O(n \log n)$. Taking largest k coordinates from x' requires $O(n \log n)$ time. Thus the total query time is $O(n \log n)$. In the following, we show that $\|x - x'_T\|_H \leq (1 + \epsilon)\|x_{-k}\|_H$.

By Lemma 15.5.1, for each $i \in [n]$, with probability $1 - 1/\operatorname{poly}(n)$,

$$(x_i - x'_i)^2 \leq \frac{\epsilon^2}{4k}\|x_{-4k/\epsilon^2}\|_2^2 \leq \frac{\epsilon^2}{4k}\|x_{-4k}\|_2^2. \tag{15.1}$$

Let S denote the largest k coordinates in absolute value in x. Recall that $\overline{S} = [n]\backslash S$, $\overline{T} = [n]\backslash T$, and $x_{-k} = x_{\overline{S}}$. The recovery error is

$$\|x - x'_T\|_H = \|(x - x')_T\|_H + \|x_{\overline{T}}\|_H. \tag{15.2}$$

By point estimation guarantee given in Eq. (15.5) and results from Huber set query.

$$\begin{aligned}\|(x - x')_T\|_H &= \sum_{i \in T} H(x_i - x'_i) \\ &\leq |T| \cdot H(\frac{\epsilon}{2\sqrt{k}}\|x_{-4k}\|_2) \\ &\leq \frac{\epsilon}{2}\|x_{-k}\|_H.\end{aligned}$$

Note that

$$\begin{aligned}\|x_{\overline{T}}\|_H &= (\|x_{\overline{T}}\|_H - \|x_{\overline{S}}\|_H) + \|x_{\overline{S}}\|_H \\ &= (\|x_{S \cap \overline{T}}\|_H - \|x_{\overline{S} \cap T}\|_H) + \|x_{-k}\|_H\end{aligned}$$

where the second step follows by canceling out the coordinates in $\overline{S} \cap \overline{T}$. In the following, our goal is to show that $\sum_{i \in S \cap \overline{T}} H(x_i) - \sum_{i \in \overline{S} \cap T} H(x_i) \leq \frac{\epsilon}{2}\|x_{-k}\|_H$.

If $S \cap \overline{T} = \emptyset$, then $S = T$ and $\sum_{i \in S \cap \overline{T}} H(x_i) - \sum_{i \in \overline{S} \cap T} H(x_i) = 0$. In the following, we assume that $S \cap \overline{T} \neq \emptyset$. Let $b = \max_{i \in S \cap \overline{T}} |x_i|$, and $a = \min_{i \in \overline{S} \cap T} |x_i|$. Because Huber function $H(t)$ is monotonically increasing as $|t|$ increases,

$$\|x_{S \cap \overline{T}}\|_H - \|x_{\overline{S} \cap T}\|_H \leq |\overline{S} \cap T| \cdot (H(b) - H(a)).$$

By Lemma 15.2.5, we have

$$b - a \leq \frac{\epsilon}{\sqrt{k}}\|x_{-4k}\|_2. \tag{15.3}$$

Moreover, by definition that $a = \min_{i \in \overline{S} \cap T} |x_i|$ and let $r = |\overline{S} \cap T|$,

$$a \leq \frac{\|x_{\overline{S} \cap T}\|_2}{\sqrt{r}} \leq \frac{\|x_{-k}\|_2}{\sqrt{r}}. \tag{15.4}$$

We are going to prove that $H(b) - H(a) \leq \frac{\epsilon}{2} \frac{\|x_{-k}\|_H}{|\overline{S} \cap T|}$. We discuss in three cases: case (1) $\tau \leq a$, case (2) $a < \tau < b$, case (3) $b \leq \tau$. We analyze these three cases in Claim 15.2.2, Claim 15.2.3 and Claim 15.2.4. □

Claim 15.2.2. *Let $r = |\overline{S} \cap T|$. If $\tau \leq a$, then $H(b) - H(a) \leq \frac{\epsilon}{2} \frac{\|x_{-k}\|_H}{r}$.*

Proof. For simplicity, we use r to denote $|\overline{S} \cap T|$. By definition of T, we know that $1 \leq r \leq k$. Let P, Q be a partition of $\overline{S}$ where

$$P = \{i \mid |x_i| \leq \tau, i \in \overline{S}\}, Q = \{i \mid |x_i| > \tau, i \in \overline{S}\}.$$

By triangle inequality $\|x_{-5k}\|_2 \leq \|x_P\|_2 + \|(x_Q)_{-4k}\|_2$. By definition of Huber function $\|x_{-k}\|_H = \frac{1}{\tau}\|x_P\|_2^2 + \|x_Q\|_1$. Because $\tau \leq a$, $\overline{S} \cap T \subseteq Q$. Thus $\|x_Q\|_1 \geq \|x_{\overline{S} \cap T}\|_1 \geq \tau |\overline{S} \cap T| = \tau \cdot r$. Therefore,

$$\|x_{-k}\|_H \geq \frac{1}{\tau}\|x_P\|_2^2 + \tau r \geq 2\sqrt{\frac{1}{\tau}\|x_P\|_2^2 \cdot \tau r} = 2\sqrt{r} \cdot \|x_P\|_2.$$

So we have $\|x_P\|_2 \leq \frac{1}{2\sqrt{r}}\|x_{-k}\|_H$. By Lemma 15.5.2,

$$\|(x_Q)_{-4k}\|_2 \leq \frac{1}{2\sqrt{k}}\|x_Q\|_1 \leq \frac{1}{2\sqrt{r}}\|x_Q\|_1 \leq \frac{1}{2\sqrt{r}}\|x_{-k}\|_H.$$

Therefore,

$$\begin{aligned} H(b) - H(a) = b - a &\leq \frac{\epsilon}{\sqrt{r}} \|x_{-5k}\|_2 \\ &\leq \frac{\epsilon}{\sqrt{r}} (\|x_P\|_2 + \|(x_Q)_{-4k}\|_2) \\ &\leq \frac{\epsilon}{\sqrt{r}} \left(\frac{1}{2\sqrt{r}} \|x_{-k}\|_H + \frac{1}{2\sqrt{r}} \|x_{-k}\|_H \right) \\ &\leq \frac{\epsilon \|x_{-k}\|_H}{r}. \end{aligned}$$

□

Claim 15.2.3. *If $a < \tau < b$, then $H(b) - H(a) \leq \frac{\epsilon}{2} \frac{\|x_{-k}\|_H}{|\overline{S} \cap T|}$.*

Claim 15.2.4. *If $b \leq \tau$, then $H(b) - H(a) \leq \frac{\epsilon}{2} \frac{\|x_{-k}\|_H}{|\overline{S} \cap T|}$.*

We provide the proofs of the above two Claims in Section 15.7.

Lemma 15.2.5. *Let vector $u, v \in \mathbb{R}^n$ such that $\|u - v\|_\infty \leq \Delta$. Let S be the largest k coordinates in u, and let T be the largest k coordinates in v. Let $b = \max_{i \in S \cap \overline{T}} u_i$ and $a = \min_{i \in \overline{S} \cap T} u_i$. Then $b - a \leq 2\Delta$.*

Proof. Let $\lambda = \max_{i \in \overline{T}} v_i$. We have

$$b = \max_{i \in S \cap \overline{T}} u_i \leq \max_{i \in \overline{T}} u_i \leq \Delta + \max_{i \in \overline{T}} v_i = \Delta + \lambda.$$

On the other hand,

$$a = \min_{i \in \overline{S} \cap T} u_i \geq \min_{i \in T} u_i \geq -\Delta + \min_{i \in T} v_i \geq -\Delta + \lambda.$$

Therefore,

$$b - a \leq (\Delta + \lambda) - (-\Delta + \lambda) \leq 2\Delta.$$

□

15.3 Conclusion

In this work, we gave the first set of sparse recovery algorithms when the error is measured with respect to Huber and Tukey functions. Because of the usefulness of these M-estimators, we believe that this will be proved to be an important line of research, demanding new ideas and techniques, and offering fruitful results both in theory and in practice.

15.4 Preliminaries

In this section, we give definitions of Huber function, reverse Huber function and Tukey function. All these functions have a parameter $\tau > 0$ such that the function is defined as one "simple" function on domain $\{x : |x| \leq \tau\}$ and another "simple" function on domain $\{x : |x| > \tau\}$. In particular, Huber function is defined to be a quadratic function for $|x| \leq \tau$ and a linear function for $|x| > \tau$. Inverse Huber function is opposite. Tukey function is defined approximately to be a quadratic function for $|x| \leq \tau$, and a constant function for $|x| > \tau$.

When measuring errors in terms of these functions, we are more robust to outliers. Because these functions in domain $\{x : |x| > \tau\}$ grow slower than domain $\{x : |x| \leq \tau\}$.

15.4.1 Huber estimator

Given parameter $\tau > 0$, Huber function [Hub64] is defined as follows.

Definition 15.4.1 (Huber function)**.**

$$H(x) = \begin{cases} |x|^2/\tau & \text{if } |x| \leq \tau; \\ |x| & \text{if } |x| > \tau. \end{cases}$$

In addition, as appeared in [SWZ18], one can also define reverse Huber function with parameter τ as

$$H(x) = \begin{cases} |x| & \text{if } |x| \leq \tau; \\ |x|^2/\tau & \text{if } |x| > \tau. \end{cases}$$

15.4.2 Tukey estimator

Tukey function [Tuk60] is defined as follows.

Definition 15.4.2 (Tukey function)**.**

$$H(x) = \begin{cases} 1 - (1 - (x/\tau)^2)^3, & \text{if } |x| \leq \tau; \\ 1, & \text{if } |x| > \tau. \end{cases}$$

It is easy to observe that for $|x| \leq \tau$,

$$\begin{aligned} H(x) &= 1 - (1 - 3(x/\tau)^2 + 3(x/\tau)^4 - (x/\tau)^6) \\ &= 3(x/\tau)^2 - 3(x/\tau)^4 + (x/\tau)^6 \\ &= \frac{3}{4}(x/\tau)^2 + \frac{9}{4}(x/\tau)^2 - 3(x/\tau)^4 + (x/\tau)^6 \\ &= \frac{3}{4}(x/\tau)^2 + (x/\tau)^2(\frac{3}{2} - (x/\tau)^2)^2 \\ &\geq \frac{3}{4}(x/\tau)^2. \end{aligned}$$

Similarly, we can also give an upper bound of $H(x)$,

$$\begin{aligned} H(x) &= 3(x/\tau)^2 - 3(x/\tau)^4 + (x/\tau)^6 \\ &\leq 3(x/\tau)^2 + (x/\tau)^6 \\ &\leq 4(x/\tau)^2. \end{aligned}$$

Therefore, we have the following fact, i.e. Tukey function is within a constant factor of the quadratic function for $|x| \leq \tau$. Therefore, in our analysis in later sections, we approximate Tukey function as a quadratic function for $|x| \leq \tau$.

Fact 15.4.1. *If* $|x| \leq \tau$*,* $H(x) \in [\frac{3}{4}(x/\tau)^2, 4(x/\tau)^2]$*; Otherwise* $H(x) = 1$*.*

15.5 Set Query

In this section, we present our set query results on reverse Huber function and Tukey function. The results on Huber function is presented in the main text. We summarize our results on set query in Table 15.1. Since we extensively apply the count-sketch data structure, we present as follows.

Definition 15.5.1 (Count Sketch)**.** A count sketch data structure with bucket size B and replicates R corresponds to a measurement matrix $\Phi \in R^{BR\times n}$. Matrix Φ is a concatenation of $\Phi^{(1)}, \ldots, \Phi^{(R)}$, where each $\Phi^{(r)} \in \mathbb{R}^{B\times n}$ $(r \in [R])$ is defined by a pariwise hash function $h_i : [n] \to [B]$ and a pairwise sign function $\sigma_i : [n] \to \{-1, 1\}$, such that for each $j \in [B]$ and $i \in [n]$,

$$\Phi^{(r)}_{j,i} = \begin{cases} \sigma_r(i) & \text{if } h_r(i) = j, \\ 0 & \text{otherwise.} \end{cases}$$

The data structure maintains the value of Φx and for each $i \in [n]$, it gives

$$x'_i = \text{median}\left(\sigma_1(i) \cdot (\Phi^{(1)}x)_{h_1(i)}, \ldots, \sigma_R(i) \cdot (\Phi^{(R)}x)_{h_R(i)}\right)$$

as esimation to x_i.

Let $x' \in \mathbb{R}^n$ denote the approximation given by the count sketch data structure. We have the following guarantee for count sketch with bucket size B and replicates R.

Lemma 15.5.1 ([CCF02])**.** *Let $x \in \mathbb{R}^n$, and let $x' \in \mathbb{R}^n$ be the output of a count sketch data structure (Definition 15.5.1) with bucket size B and replicate R. For each $i \in [n]$, with probability at least $1 - 2^{-\Omega(R)}$,*

$$(x_i - x'_i)^2 \leq \frac{2}{B}\|x_{-B/2}\|_2^2.$$

Table 15.1: Set Query

	Measurements	Running time	Column sparsity
Theorem 15.2.1, Huber	$\epsilon^{-1}k\log k$	$k\log k$	$\log k$
Theorem 15.5.3, Reverse Huber	$\epsilon^{-1}k\log k$	$k\log k$	$\log k$
Theorem 15.5.4, Tukey	$\epsilon^{-1}k\log k$	$k\log k$	$\log k$

The following is a standard lemma in compressive sensing literatures that bounds the ℓ_2 norm of a vector with largest (in absolute value) k coordinates removed in terms of ℓ_1 norm of its original vector. We also include its proof for completeness.

Lemma 15.5.2. *For any $x \in \mathbb{R}^n$ and any $k \in [n]$, $\|x_{-k}\|_2 \leq \frac{1}{\sqrt{k}}\|x\|_1$, where x_{-k} denotes the vector obtained by zeroing out largest k coordinates (in absolute) of x.*

Proof. We partition the coordinates of x to be $B_1, B_2, \ldots$ where B_1 contains largest k coordinates, and B_2 contains the next largest k coordinates, and so on. We have

$$\begin{aligned}
\|x_{-k}\|_2 &\leq \sum_{i=2}^{n/k} \|x_{B_i}\|_2 \\
&\leq \sum_{i=2}^{n/k} \sqrt{k \cdot (\|x_{B_{i-1}}\|_1/k)^2} \\
&= \frac{1}{\sqrt{k}} \sum_{i=1}^{n/k-1} \|x_{B_i}\|_1 \\
&\leq \frac{1}{k}\|x\|_1.
\end{aligned}$$

where the second step follows by Cauchy-Schwarz inequality. □

15.5.1 ϵ-approximation for reverse Huber function

We present our set query result for reverse Huber functions. The result is quite symmetric to our result for Huber function.

Theorem 15.5.3 (Reverse Huber set-query)**.** *Let function H be defined as Definition 15.4.1. There is an algorithm that takes $O(\epsilon^{-1} k \log k)$ samples/times and outputs a vector $x' \in \mathbb{R}^n$ such that*

$$\|x'_S - x_S\|_H \leq \epsilon \cdot \|x_{[n]\backslash S}\|_H$$

holds with probability $1 - 1/\operatorname{poly}(k)$.

Proof. We choose $B = O(\epsilon^{-1}k)$ and $R = O(\log k)$, and use the count-sketch data structure with bucket size $O(B)$ and replicate number $O(R)$. Let $x' \in \mathbb{R}^n$ denote the estimate given by the count-sketch data structure. By Lemma 15.5.1, we have the guarantee that with probability $1 - 1/\operatorname{poly}(k)$,

$$|x'_i - x_i| \leq \sqrt{\frac{\|x_{-B}\|_2^2}{B}}.$$

We define set $P, Q \subset [n]\backslash S$ as follows

$$P = \{i \mid |x_i| \leq 1, i \in [n]\backslash S\}, Q = \{i \mid |x_i| \geq 1, i \in [n]\backslash S\}.$$

We define $P_{B/6}$ to be top $B/6$ coordinates in P, and define $P' = P\backslash P_{B/6}$. Similarly we define $Q_{B/6}$ to be top $B/6$ coordinates in Q, and define $Q' = Q\backslash Q_{B/6}$. We have $\sqrt{\frac{\|x_{-B}\|_2^2}{B}} \leq \sqrt{\frac{\|x_{P'}\|_2^2 + \|x_{Q'}\|_2^2}{B}}$.

Note that

$$\|x_{\overline{S}}\|_H = \|x_P\|_1 + \|x_Q\|_2^2.$$

We prove the final result by considering two cases: case (1) $\sqrt{\frac{\|x_{P'}\|_2^2+\|x_{Q'}\|_2^2}{B}} \leq 1$; Case (2) $\sqrt{\frac{\|x_{P'}\|_2^2+\|x_{Q'}\|_2^2}{B}} > 1$.

Case (1) $\sqrt{\frac{\|x_{P'}\|_2^2+\|x_{Q'}\|_2^2}{B}} \leq 1$.

The error is

$$\begin{aligned}\|x'_S - x_S\|_H &\leq k\sqrt{\frac{\|x_{P'}\|_2^2 + \|x_{Q'}\|_2^2}{B}} \\ &\leq \frac{k}{\sqrt{B}}(\|x_{P'}\|_2 + \|x_{Q'}\|_2).\end{aligned}$$

By Lemma 15.5.2, we have $\|x_{P'}\|_2 \leq \frac{1}{\sqrt{B}}\|x_P\|_1$. Now we are trying to upper bound $\|x_{Q'}\|_2$. If $\|x_{Q'}\|_2 \leq \|x_{P'}\|_2$ we are done. Otherwise we have $\|x_{Q'}\|_2 \geq \sqrt{B/2}$ and so $\|x_{Q'}\|_2 \leq \frac{\|x_{Q'}\|_2}{\sqrt{B/2}}\|x_{Q'}\|_2 \leq \sqrt{2/B}\|x_Q\|_2^2$. Either case we can finally bound $\|x'_S - x_S\|_H$ by $2\epsilon\|x_{\overline{S}}\|_H$.

Case (2) $\sqrt{\frac{\|x_{P'}\|_2^2+\|x_{Q'}\|_2^2}{B}} > 1$.

We have

$$\begin{aligned}\|x'_S - x_S\|_H &\leq \frac{k}{B}(\|x_{P'}\|_2^2 + \|x_{Q'}\|_2^2) \\ &\leq \frac{k}{B}(\|x_{P'}\|_1 + \|x_{Q'}\|_2^2) \\ &\leq \frac{k}{B}(\|x_P\|_1 + \|x_Q\|_2^2) \\ &= \epsilon\|x_{\overline{S}}\|_H,\end{aligned}$$

where the second step follows by the fact that each coordinate in P' is less than 1. □

15.5.2 ϵ-approximation for Tukey function

Theorem 15.5.4 (Tukey set-query)**.** *Let function H be defined as Definition 15.4.2. There is an algorithm that takes $O(\epsilon^{-1} k \log k)$ samples, $O(k \log k)$ decoding times, $O(\log k)$ update time, and outputs a vector $x' \in \mathbb{R}^n$ such that*

$$\|x'_S - x_S\|_H \leq \epsilon \cdot \|x_{[n]\backslash S}\|_H$$

holds with probability $9/10$.

Proof. Without loss of generality, we assume that $\tau = 1$. We approximate Tukey function to be $H(x) = 1$ for $|x| > 1$ and $H(x) = x^2$ for $|x| \leq 1$.

Let $B = O(\epsilon^{-1} k)$ and $R = O(\log k)$, and use the count-sketch data structure with bucket size $O(B)$ and replicate number $O(R)$. Let $x' \in \mathbb{R}^n$ denote the estimate given by the count-sketch data structure.

We define $P, Q \subseteq [n]\backslash S$ as follows

$$P = \{i \mid |x_i| \leq 1, i \in [n]\backslash S\}, Q = \{i \mid |x_i| > 1, i \in [n]\backslash S\}.$$

We consider the following two cases.

Case 1. $|Q| > B/4$.

We have

$$\|x'_S - x_S\|_H \leq k = \frac{k}{|Q|} \cdot |Q| \leq \frac{k}{B/4}|Q| \leq \frac{4k}{B}\|x_{-k}\|_H \leq \epsilon \|x_{-k}\|_H.$$

Case 2. $|Q| \leq B/4$.

By Lemma 15.5.1, for each $i \in S$, with probability at least $1 - 1/\operatorname{poly}(k)$, we have

$$|x'_i - x_i| \leq \sqrt{\frac{\|x_{-2B}\|_2^2}{2B}} \leq \sqrt{\frac{\|x_P\|_2^2}{2B}}.$$

Applying a union bound to coordinates in S, we have with probability at least $1 - \operatorname{poly}(k)$,

$$\|x'_S - x_S\|_H \leq kH(\sqrt{\frac{\|x_P\|_2^2}{2B}}).$$

Then Case 2(a) $\sqrt{\|x_P\|_2^2/2B} \geq 1$. We have

$$\|x'_S - x_S\|_H \leq k \leq \frac{k}{2B}\|x_P\|_2^2 \leq \frac{k}{2B}\|x_{-k}\|_H \leq \epsilon\|x_{-k}\|_H.$$

Case 2(b) $\sqrt{\|x_P\|_2^2/2B} < 1$. We have

$$\|x'_S - x_S\|_H \leq \frac{k}{2B}\|x_P\|_2^2 \leq \frac{k}{2B}\|x_{-k}\|_H \leq \epsilon\|x_{-k}\|_H.$$

□

15.6 Fourier set query

In this section, we give set query algorithms from Fourier measurements, when the error is measured with respect to Huber norm and Tukey norm. All of the results in this section follow from the fact that we can simulate the count sketch data structure in Fourier case, by losing a factor of $\log n$ on the number of measurements (we also lose a $\log n$ factor in update time and query time).

This simulation is standard (as in for example [HIKP12a]) by using (pseudo)random spectral permutation and filtering functions. Let B denote bucket size. Let $P = \{i \in [n] \mid |\widehat{x}_i| < \tau\}$ and $Q = \{i \in [n] \mid |\widehat{x}_i| \geq \tau\}$. Let P' denote the set after removing top $O(k)$ coordinates from P, and similarly let Q' denote the set after removing top $O(k)$ coordinates from Q. We can guarantee that with high probability for each coordinate $i \in S$, the error for our estimation to $\widehat{x}_i$ is bounded by $\sqrt{\frac{\|\widehat{x}_{P'}\|_2^2 + \|\widehat{x}_{Q'}\|_2^2}{B}}$. Following the proof on set query results in Section 15.5, this yields to $\ell_H/\ell_H \leq \epsilon$ guarantee for function H being Huber function or Tukey function.

Theorem 15.6.1 (Fourier point estimation guarantee)**.** *Given a vector $x \in \mathbb{C}^n$, for every $\epsilon \in (0,1)$ and $k \geq 1$, any $S \subseteq [n]$, $|S| \leq k$, there exists an algorithm (Algorithm 15.3) that takes*

$$O(\epsilon^{-1} k \log k \cdot \log n)$$

samples, runs in

$$O(\epsilon^{-1} k \log k \cdot \log n)$$

time, and outputs a vector $x' \in \mathbb{C}^n$ *such that for each* $i \in S$,

$$|x'_i - \widehat{x}_i| \leq \sqrt{\frac{\|\widehat{x}_{P'}\|_2^2 + \|\widehat{x}_{Q'}\|_2^2}{B}},$$

holds with probability at least $1 - 1/\operatorname{poly}(k)$.

Before showing the proof to Theorem 15.6.1, we first present some basic definitions and basic building blocks for Fourier sparse recovery. Readers can refer to [HIKP12a] for more details.

For a complex vector $x \in \mathbb{C}^n$, we use $\widehat{x} \in \mathbb{C}^n$ to denote its Fourier spectrum,

$$\widehat{x}_i = \frac{1}{\sqrt{n}} \sum_{j=1}^{n} e^{-2\pi \mathbf{i} ij/n} x_j, \forall i \in [n].$$

Then the inverse transform is

$$x_j = \frac{1}{\sqrt{n}} \sum_{i=1}^{n} e^{2\pi \mathbf{i} ij/n} \widehat{x}_i, \forall j \in [n].$$

The discrete convolution of functions f and g is given by,

$$(f * g)[n] = \sum_{m=-\infty}^{+\infty} f[m] g[n-m]$$

We use the same (pseudorandom) spectrum permutation as [HIKP12a],

Definition 15.6.1. Suppose σ^{-1} exists mod n. We define the permutation $P_{\sigma,a,b}$ by

$$(P_{\sigma,a,b} x)_i = x_{\sigma(i-a)} e^{-2\pi \mathbf{i} \sigma b i/n}.$$

We also define $\pi_{\sigma,b} = \sigma(i-b) \pmod{n}$. Let $h_{\sigma,b}(i) = \operatorname{round}(\pi_{\sigma,b}(i) B/n)$ and $o_{\sigma,b}(i) = \pi_{\sigma,b}(i) - h_{\sigma,b}(i) n/B$. We say $h_{\sigma,b}(i)$ is the "bin" that frequency i is mapped into, and $o_{\sigma,b}(i)$ is the "offset".

We use the same filter function as [HIKP12a, PS15, CKPS16],

Definition 15.6.2. Given parameters $B \geq 1$, $\delta > 0$, $\alpha > 0$. We say that $(G, \widehat{G}') = (G_{B,\delta,\alpha}, \widehat{G}'_{B,\delta,\alpha}) \in \mathbb{R}^n$ is a filter function if it satisfies the following properties

$$
\begin{aligned}
&\text{(I)} \ |\operatorname{supp}(G)| = O(\alpha^{-1} B \log(n/\delta)), \\
&\text{(I)} \ \widehat{G}'_i = 1, && \text{if } |i| \leq (1-\alpha)n/(2B), \\
&\text{(II)} \ \widehat{G}'_i = 0, && \text{if } |i| \geq n/(2B), \\
&\text{(III)} \ \widehat{G}'_i \in [0,1], && \text{for all } i, \\
&\text{(IV)} \ \left\| \widehat{G}' - \widehat{G} \right\|_\infty < \infty.
\end{aligned}
$$

We present our Fourier set query algorithm in Algorithm 15.3. We choose bucket size $B = O(k/\epsilon)$ and number of repetitions $R = O(\log k)$. For the filter function, we choose $\delta = 1/\operatorname{poly}(n)$ so that the noise introduced by the filter is negligible. Let α be a very small constant. We use procedure EstimateValues in r-th repetition where $r \in [R]$ to obtain an unbiased estimator for $\widehat{x}_S$ (up to error $\delta \|\widehat{x}\|_1$), denoted by $w^{(r)}$. Finally, for each $i \in S$, we estimate $\widehat{x}_i$ by $x'_i \leftarrow \text{median}_{r \in [R]} w_i^{(r)}$. By taking the median of a set of complex numbers, we mean that we take the median of real part and imaginary part separately, and then add these two parts together.

We use three types of events defined in [HIKP12a] as basic building blocks for analyzing Fourier set query algorithms. For any $i \in S$, we define three types of events associated with i and S and defined over the probability space induced by σ and b:

Definition 15.6.3 (Collision, large offset, large noise)**.** "Collision" event $E_{\text{coll}}(i)$: holds iff $h_{\sigma,b}(i) \in h_{\sigma,b}(S \backslash \{i\})$.

"Large offset" event $E_{\text{off}}(i)$: holds iff $|o_{\sigma,b}(i)| \geq (1-\alpha)n/(2B)$.

"Large noise" event $E_{\text{noise}}(i)$: holds iff

$$\mathbb{E}\left[\left\|\widehat{x}'_{h^{-1}_{\sigma,b}(h_{\sigma,b}(i))\setminus S}\right\|_2^2\right] \geq \text{Err}^2(\widehat{x}', k)/(\alpha B).$$

Claim 15.6.2 (Claim 3.1 in [HIKP12a]). *For any $i \in S$, the event $E_{\text{coll}}(i)$ holds with probability at most $4|S|/B$.*

Claim 15.6.3 (Claim 3.2 in [HIKP12a]). *For any $i \in S$, the event $E_{\text{off}}(i)$ holds with probability at most α.*

Claim 15.6.4 (Claim 4.1 in [HIKP12a]). *For any $i \in S$, $\Pr[E_{\text{noise}(i)}] \leq 4\alpha$.*

Lemma 15.6.5 (Lemma 4.2 in [HIKP12a]). *Let $a \in [n]$ uniformly at random, B divide n, and the other parameters be arbitrary in*

$$\widehat{u} = \text{HashToBins}(x, P_{\sigma,a,b}, B, \delta, \alpha).$$

Then for any $i \in [n]$ with $j = h_{\sigma,b}(i)$ and none of $E_{\text{coll}}(i)$, $E_{\text{off}}(i)$ or $E_{\text{noise}}(i)$ holding,

$$\mathbb{E}\left[\left|\widehat{u}_j - \widehat{x}'_i e^{-\frac{2\pi \mathbf{i}}{n} a\sigma i}\right|^2\right] > 2\frac{\rho^2}{\alpha B}.$$

Lemma 15.6.6. *Suppose B divides n. The output $\widehat{u}$ of* HashToBins *satisfies*

$$\widehat{u}_j = \sum_{h_{\sigma,b}(i)=j} \widehat{x}_i \widehat{(G'_{B,\delta,\alpha})}_{-o_{\sigma,b}(i)} \omega^{a\sigma i} \pm \delta\|\widehat{x}\|_1.$$

The running time of HashToBins *is $O(\frac{B}{\alpha}\log(n/\delta))$.*

Proof of Theorem 15.6.1. We give Fourier set query algorithm in Algorithm 15.3. The total number of measurements and decoding time is $O(R) \cdot O(B \log \frac{n}{\alpha}) = O(\epsilon^{-1} k \log k \log n)$.

For each repetition r, for each $i \in S$, we can union bound the probability that none of three bad events (collision, large offset, large noise) happens by $\frac{1}{10}$. Conditioned on no bad events happened, by Lemma 15.6.5, $w_i^{(r)}$ is a good estimation for $\widehat{x}_i$ in the sense that $\mathbb{E}[|\widehat{x}_i - w_i^{(r)}|^2] = O(\frac{\|x_P\|_2^2 + \|x_Q\|_2^2}{B})$. By Markov inequality with only very small constant probability that $|\widehat{x}_i - w_i^{(r)}| \geq \frac{1}{2}\sqrt{\frac{\|x_P\|_2^2 + \|x_Q\|_2^2}{B}}$. By applying Chernoff bound, we can prove that with probability $2^{-\Omega(R)}$, $|\widehat{x}_i - \mathrm{median}_{r \in [R]} w_i^{(r)}| \geq \frac{1}{2}\sqrt{\frac{\|x_P\|_2^2 + \|x_Q\|_2^2}{B}}$. □

Corollary 15.6.7 (Fourier set query)**.** *Given a vector $x \in \mathbb{C}^n$, for every $\epsilon \in (0,1)$ and $k \geq 1$, any $S \subseteq [n]$, $|S| \leq k$, there exists an algorithm (Algorithm 15.3) that takes*

$$O(\epsilon^{-1} k \log k \cdot \log n)$$

samples, runs in

$$O(\epsilon^{-1} k \log k \cdot \log n)$$

time, and outputs a vector $x' \in \mathbb{C}^n$ such that

$$\|(x' - \widehat{x})_S\|_H \leq \epsilon \|\widehat{x}_{\overline{S}}\|_H,$$

holds with probability at least $1 - 1/\operatorname{poly}(k)$. The corollary holds when function H is Huber function or Tukey function.

Algorithm 15.3 Fourier set query algorithm

procedure FOURIERCOUNTSKETCH(x, S, ϵ, k) ▷ Theorem 15.6.1
  $B \leftarrow O(k/\epsilon)$, $R \leftarrow O(\log k)$, $\delta \leftarrow 1/\operatorname{poly}(n)$, $\alpha \leftarrow 1/100$
  for $r = 1 \to R$ **do**
    $\widehat{w}^{(r)} \leftarrow$ ESTIAMTEVALUES(x, S, B, δ, α)
  end for
  for $i \in S$ **do**
    $x'_i \leftarrow \operatorname{median}_{r \in [R]} \widehat{w}_i^{(r)}$
  end for
  return x'_S
end procedure
procedure ESTIMATEVALUES(x, S, B, δ, α)
  Choose $a, b \in [n]$ uniformly at random
  Choose σ uniformly at random from the set of odd numbers in $[n]$
  $\widehat{u} \leftarrow$ HASHTOBINS($x, P_{\sigma,a,b}, B, \delta, \alpha$)
  $\widehat{w} \leftarrow 0$
  for $t \in S$ **do**
    $\widehat{w}_t \leftarrow \widehat{u}_{h_{\sigma,b}(t)} e^{\frac{2\pi \mathbf{i}}{n} \sigma a t}$
  end for
  return $\widehat{w}$
end procedure
procedure HASHTOBINS($x, P_{\sigma,a,b}, B, \delta, \alpha$)
  Compute $\widehat{y}_{jn/B}$ for $j \in [B]$, where $y = G_{B,\alpha,\delta} \cdot (P_{\sigma,a,b} x)$
  return $\widehat{u}_j = \widehat{y}_{jn/B}$
end procedure

15.7 Sparse recovery

15.7.1 k-sparse recovery for Huber function

In this subsection, we present our k-sparse recovery results for Huber function.

Theorem 15.7.1. *Let H denote Huber function defined in Definition 15.4.1. There exists a distribution of $\Phi \in \mathbb{R}^{m \times n}$ with $m = O(\epsilon^{-2} k \log n)$ and a recovery algorithm, such that for each $x \in \mathbb{R}^n$, $k \in [n]$ and $\epsilon \in (0, 1)$, the recovery algorithm takes Φx as its input and outputs*

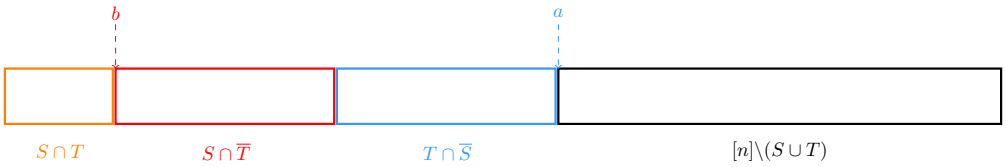


Figure 15.3: $\|x_{S\cap\overline{T}}\|_H - \|x_{T\cap\overline{S}}\|_H$ is the displacement error

k-sparse *vector* $x' \in \mathbb{R}^n$ *so that with probability* $1 - 1/\operatorname{poly}(n)$,

$$\|x' - x\|_H \leq (1+\epsilon) \min_{k-\text{sparse } y} \|y - x\|_H.$$

The update time is $O(\log n)$ *and the recovery time is* $O(n \log n)$.

Proof. Let x_{-k} denotes the vector x with the largest k coordinates in magnitude zeroed out.

We use count sketch to give each x_i (where $i \in [n]$) an estimate. In particular, we choose the number of buckets to be $B = O(\epsilon^{-2}k)$ and the number of repetitions to be $R = O(\log n)$. Let x'_i denote the estimate obtained by the count sketch for x_i. Let T denote the largest k coordinates in absolute value in x'. The recovery algorithm outputs x'_T as a k-sparse approximation to x.

Note that the count sketch data structure takes $B \cdot R = O(\epsilon^{-2}k \log n)$ measurements. Its update time is $O(R) = O(\log n)$ time, and the time to obtain x' is $O(n \cdot R) = O(n \log n)$. Taking largest k coordinates from x' requires $O(n \log n)$ time. Thus the total query time is $O(n \log n)$. In the following, we show that $\|x - x'\|_H \leq (1+\epsilon)\|x_{-k}\|_H$.

By Lemma 15.5.1, for each $i \in [n]$, with probability $1 - 1/\operatorname{poly}(n)$,

$$(x_i - x'_i)^2 \leq \frac{\epsilon^2}{64k}\|x_{-64k/\epsilon^2}\|_2^2 \leq \frac{\epsilon^2}{64k}\|x_{-64k}\|_2^2. \tag{15.5}$$

Let S denote the largest k coordinates of x in magnitude. The error is

$$\|x - x'\|_H = \sum_{i \in T} H(x_i - x'_i) + \sum_{i \in \overline{T}} H(x_i). \tag{15.6}$$

By point estimation guarantee given in Eq. (15.5) and results from Huber set query,

$$\begin{aligned}\sum_{i \in T} H(x_i - x'_i) &\leq |T| \cdot H(\frac{\epsilon}{8\sqrt{k}} \|x_{-64k}\|_2) \\ &\leq \frac{\epsilon}{8} \|x_{-k}\|_H.\end{aligned}$$

Note that

$$\begin{aligned}\sum_{i \in \overline{T}} H(x_i) &= \left(\sum_{i \in \overline{T}} H(x_i) - \sum_{i \in \overline{S}} H(x_i) \right) + \sum_{i \in \overline{S}} H(x_i) \\ &= \left(\sum_{i \in S \cap \overline{T}} H(x_i) - \sum_{i \in \overline{S} \cap T} H(x_i) \right) + \|x_{-k}\|_H,\end{aligned}$$

where the second step follows by canceling out the coordinates in $\overline{S} \cap \overline{T}$. In the following, our goal is to show that $\sum_{i \in S \cap \overline{T}} H(x_i) - \sum_{i \in \overline{S} \cap T} H(x_i) \leq \frac{7}{8} \epsilon \|x_{-k}\|_H$.

If $S \cap \overline{T} = \emptyset$, then $S = T$ and $\sum_{i \in S \cap \overline{T}} H(x_i) - \sum_{i \in \overline{S} \cap T} H(x_i) = 0$. In the following, we assume that $S \cap \overline{T} \neq \emptyset$. Let $b = \max_{i \in S \cap \overline{T}} |x_i|$, and $a = \min_{i \in \overline{S} \cap T} |x_i|$. Because Huber function $H(t)$ is monotonically increasing as $|t|$ increases,

$$\sum_{i \in S \cap \overline{T}} H(x_i) - \sum_{i \in \overline{S} \cap T} H(x_i) \leq |\overline{S} \cap T| \cdot (H(b) - H(a)).$$

Our final goal is to prove that $H(b) - H(a) \leq \frac{7\epsilon \|x_{-k}\|_H}{8|\overline{S} \cap T|}$. We discuss in three cases: case (1) $\tau \leq a$, case (2) $a < \tau < b$, case (3) $b \leq \tau$. We analyze these three cases in Claim 15.7.2, Claim 15.7.3 and Claim 15.7.4.

Before diving into case analysis, we first prove two inequalities: one is an upper bound of $b - a$, and the other is an upper b ound of a. They will be used repeatedly in the case analysis.

By Eq. (15.5), any $i \in [n]$, $|x_i - x'_i| \leq \frac{\epsilon}{8\sqrt{k}}\|x_{-64k}\|_2$. That is, $\|x - x'\|_\infty \leq \Delta$ where $\Delta = \frac{\epsilon}{8\sqrt{k}}\|x_{-64k}\|_2$. By Lemma 15.7.5,

$$b - a \leq \frac{\epsilon}{4\sqrt{k}}\|x_{-64k}\|_2. \tag{15.7}$$

Moreover, by definition that $a = \min_{i \in \overline{S} \cap T} |x_i|$,

$$a \leq \frac{\|x_{\overline{S}\cap T}\|_2}{\sqrt{|\overline{S} \cap T|}} \leq \frac{\|x_{-k}\|_2}{\sqrt{|\overline{S} \cap T|}}. \tag{15.8}$$

□

Claim 15.7.2. *If $\tau \leq a$, then $H(b) - H(a) \leq \frac{7\epsilon}{8}\frac{\|x_{-k}\|_H}{|\overline{S}\cap T|}$.*

Proof. Let P, Q be a partition of $\overline{S}$ where

$$P = \{i \mid |x_i| \leq \tau, i \in \overline{S}\}, Q = \{i \mid |x_i| > \tau, i \in \overline{S}\}.$$

Note that the coordinates in x_{-64k}, which equals to $\overline{S} = P \cup Q$ with largest (in absolute value) $63k$ coordinates removed, are fully contained in P union Q with largest (in absolute value) $63k$ coordinates removed. By triangle inequality,

$$\|x_{-64k}\|_2 \leq \|x_P\|_2 + \|(x_Q)_{-63k}\|_2. \tag{15.9}$$

In the following, we are going to bound both $\|x_P\|_2$ and $\|(x_Q)_{-63k}\|_2$ in terms of $\|x_{-k}\|_H$.

By definition of Huber function $\|x_{-k}\|_H = \frac{1}{\tau}\|x_P\|_2^2 + \|x_Q\|_1$. Because $\tau \le a$, $\overline{S} \cap T \subseteq Q$. Thus $\|x_Q\|_1 \ge \|x_{\overline{S}\cap T}\|_1 \ge \tau|\overline{S} \cap T|$. Therefore,

$$\begin{aligned}\|x_{-k}\|_H &\ge \frac{1}{\tau}\|x_P\|_2^2 + \tau|\overline{S} \cap T| \\ &\ge 2\sqrt{\frac{1}{\tau}\|x_P\|_2^2 \cdot \tau|\overline{S} \cap T|} = 2\sqrt{|\overline{S} \cap T|} \cdot \|x_P\|_2.\end{aligned}$$

So we have

$$\|x_P\|_2 \le \frac{1}{2\sqrt{|\overline{S} \cap T|}}\|x_{-k}\|_H.$$

In order to bound $\|(x_Q)_{-63k}\|_2$, by Lemma 15.5.2,

$$\begin{aligned}\|(x_Q)_{-63k}\|_2 &\le \frac{1}{\sqrt{63k}}\|x_Q\|_1 \le \frac{1}{7\sqrt{|\overline{S} \cap T|}}\|x_Q\|_1 \\ &\le \frac{1}{7\sqrt{|\overline{S} \cap T|}}\|x_{-k}\|_H.\end{aligned}$$

Therefore,

$$\begin{aligned}& H(b) - H(a) \\ =\ & b - a \\ \le\ & \frac{\epsilon}{4\sqrt{|\overline{S} \cap T|}}\|x_{-64k}\|_2 \\ \le\ & \frac{\epsilon}{4\sqrt{|\overline{S} \cap T|}}(\|x_P\|_2 + \|(x_Q)_{-63k}\|_2) \\ \le\ & \frac{\epsilon}{4\sqrt{|\overline{S} \cap T|}}\left(\frac{1}{2\sqrt{|\overline{S} \cap T|}}\|x_{-k}\|_H + \frac{1}{7\sqrt{|\overline{S} \cap T|}}\|x_{-k}\|_H\right) \\ \le\ & \frac{\epsilon\|x_{-k}\|_H}{4|\overline{S} \cap T|},\end{aligned}$$

where the first line follows from definition of Huber function and $a \geq \tau$, the second line follows by Eq. (15.7), the third follows by Eq. (15.9), the forth line follows by the bounds of $\|x_P\|_2$ and $\|(x_Q)_{-63k}\|_2$ we just obtained. □

Claim 15.7.3. *If $a < \tau < b$, then $H(b) - H(a) \leq \frac{7\epsilon}{8} \frac{\|x_{-k}\|_H}{|\overline{S} \cap T|}$.*

Proof. Let $P = \{i \mid |x_i| \leq \tau, i \in \overline{S}\}, Q = \{i \mid |x_i| > \tau, i \in \overline{S}\}$. We discuss in two cases depending on whether the size of Q is greater than $9k$ or not.

(1) If $|Q| \leq 9k$. We first bound $H(b) - H(a)$ as follows

$$\begin{aligned}
& H(b) - H(a) \\
= & \, b - a^2/\tau \\
\leq & \, b^2/\tau - a^2/\tau \\
= & \, \frac{1}{\tau}(b-a)(b+a) \\
\leq & \, \frac{\epsilon \|x_{-10k}\|_2}{4\tau\sqrt{k}} (\frac{\epsilon \|x_{-10k}\|_2}{4\sqrt{k}} + 2a) \\
\leq & \, \frac{\epsilon \|x_{-10k}\|_2}{4\tau\sqrt{k}} (\frac{\epsilon \|x_{-10k}\|_2}{4\sqrt{k}} + 2\frac{\|x_P\|_2}{\sqrt{|\overline{S} \cap T|}}) \\
= & \, \frac{\epsilon^2 \|x_{-10k}\|_2^2}{16\tau k} + \frac{\epsilon \|x_{-10k}\|_2 \|x_P\|_2}{2\sqrt{k}\sqrt{|\overline{S} \cap T|}} \\
\leq & \, \frac{\epsilon}{\tau |\overline{S} \cap T|} (\frac{1}{16}\|x_{-10k}\|_2^2 + \frac{1}{2}\|x_{-10k}\|_2 \cdot \|x_P\|_2),
\end{aligned}$$

where the forth step follows by Eq.(15.7), and the fifth step follows by Eq.(15.8).

Because $|Q| \leq 9k$, $\|x_{-10k}\|_2 \leq \|x_P\|_2$, and thus

$$H(b) - H(a) \leq \frac{\epsilon}{\tau |\overline{S} \cap T|} \cdot \frac{9}{16}\|x_P\|_2^2 \leq \frac{9\epsilon \|x_{-k}\|_H}{16 |\overline{S} \cap T|}.$$

(2) If $|Q| > 9k$. First note that

$$b - a^2/\tau \leq 2(b-a),$$

because

$$b + a^2/\tau \geq 2\sqrt{b \cdot a^2/\tau} \geq 2a.$$

Furthermore, $b - a \leq \frac{\epsilon\|x_{-64}\|_2}{4\sqrt{k}} \leq \frac{\epsilon\|x_{-10}\|_2}{4\sqrt{k}}$, where

$$\|x_{-10k}\|_2 \leq \|(x_Q)_{-9k}\|_2 + \|x_P\|_2 \leq \frac{1}{\sqrt{9k}}\|x_Q\|_1 + \|x_P\|_2.$$

Also note that $\|x_{-k}\|_H = \|x_Q\|_1 + \frac{1}{\tau}\|x_P\|_2^2 \geq 2\sqrt{\|x_Q\|_1 \cdot \|x_P\|_2^2/\tau} \geq 6\sqrt{k}\|x_P\|_2$. Therefore,

$$\begin{aligned} H(b) - H(a) &= b - a^2/\tau \\ &\leq \frac{2\epsilon}{4\sqrt{k}}\|x_{-10}\|_2 \\ &\leq \frac{2\epsilon}{4\sqrt{k}}\left(\frac{1}{\sqrt{9k}}\|x_{-k}\|_H + \frac{1}{6\sqrt{k}}\|x_{-k}\|_H\right) \\ &= \frac{\epsilon\|x_{-k}\|_H}{4k} \leq \frac{\epsilon\|x_{-k}\|_H}{4|\overline{S} \cap T|}. \end{aligned}$$

□

Claim 15.7.4. *If $b \leq \tau$, then $H(b) - H(a) \leq \frac{7\epsilon}{8}\frac{\|x_{-k}\|_H}{|\overline{S}\cap T|}$.*

Proof. The analysis is similar to classical ℓ_2/ℓ_2 sparse recovery case. We have

$$\begin{aligned}
& H(b) - H(a) \\
= & \frac{b^2}{\tau} - \frac{a^2}{\tau} \\
= & \frac{1}{\tau}(b-a)(b+a) \\
= & \frac{1}{\tau}(b-a)(b-a+2a) \\
\leq & \frac{1}{\tau} \cdot \frac{\epsilon}{4\sqrt{k}}\|x_{-k}\|_2 \cdot (\frac{\epsilon}{4\sqrt{k}}\|x_{-k}\|_2 + 2\frac{\|x_{-k}\|_2}{\sqrt{|\overline{S} \cap T|}}) \\
\leq & \frac{1}{\tau} \cdot \frac{\epsilon\|x_{-k}\|_2}{4\sqrt{|\overline{S} \cap T|}} \cdot (\frac{\|x_{-k}\|_2}{4\sqrt{|\overline{S} \cap T|}} + 2\frac{\|x_{-k}\|_2}{\sqrt{|\overline{S} \cap T|}}) \\
= & \frac{1}{\tau} \cdot \frac{9\epsilon\|x_{-k}\|_2^2}{16|\overline{S} \cap T|} \\
= & \frac{9\epsilon\|x_{-k}\|_H}{16|\overline{S} \cap T|},
\end{aligned}$$

where the first step follows by definition of Huber function, the forth step follows by Eq. (15.7) and Eq.(15.8), the fifth step follows by $|\overline{S} \cap T| \leq |T| = k$ and $\epsilon \leq 1$, the last step follows because by definition of $b = \max_{i \in S \cap \overline{T}} |x_i|$, for any $j \in [n] \backslash S$, $|x_j| \leq \min_{i \in S} |x_i| \leq b \leq \tau$ and then by definition of Huber function $\|x_{-k}\|_H = \frac{1}{\tau}\|x_{-k}\|_2^2$. □

Lemma 15.7.5. *Let vector $u, v \in \mathbb{R}^n$ such that $\|u - v\|_\infty \leq \Delta$. Let S be the largest k coordinates in u, and let T be the largest k coordinates in v. Let $b = \max_{i \in S \cap \overline{T}} u_i$ and $a = \min_{i \in \overline{S} \cap T} u_i$. Then $b - a \leq 2\Delta$.*

Proof. Let $\lambda = \max_{i \in \overline{T}} v_i$. We have

$$b = \max_{i \in S \cap \overline{T}} u_i \leq \max_{i \in \overline{T}} u_i \leq \Delta + \max_{i \in \overline{T}} v_i = \Delta + \lambda.$$

On the other hand,

$$a = \min_{i \in \overline{S} \cap T} u_i \geq \min_{i \in T} u_i \geq -\Delta + \min_{i \in T} v_i \geq -\Delta + \lambda.$$

Therefore

$$b - a \leq (\Delta + \lambda) - (-\Delta + \lambda) \leq 2\Delta.$$

□

15.7.2 k-sparse recovery for Tukey function

In this subsection, we present our k-sparse recovery results for Tukey function. Similar to results for Huber function, we use count-sketch to estimate each coordinate and then find the top k ones. Moreover, our proof for Tukey function has a similar structure, where we first decompose estimation error into set query error and displacement error, and then discuss the displacement error in several cases.

Theorem 15.7.6. *Let H denote Tukey function defined in Definition 15.4.2. There exists a distribution of $\Phi \in \mathbb{R}^{m \times n}$ with $m = O(\epsilon^{-2} k \log n)$ and a recovery algorithm, such that for each $x \in \mathbb{R}^n$, $k \in [n]$ and $\epsilon \in (0, 1)$, the recovery algorithm takes Φx as its input and outputs k-sparse vector $x' \in \mathbb{R}^n$ so that with probability $1 - 1/\operatorname{poly}(n)$,*

$$\|x' - x\|_H \leq (1 + \epsilon) \min_{k-\text{sparse } y} \|y - x\|_H,$$

The update time is $O(\log n)$ and the recovery time is $O(n \log n)$.

Proof. Let x_{-k} denotes the vector x with absolute largest k coordinates removed.

We use count sketch to give each x_i (where $i \in [n]$) an estimate. In particular, we choose the number of buckets $B = O(\epsilon^{-2}k)$ and the number of duplicates $R = O(\log n)$ for the count sketch data structure. Let x'_i denote the estimate obtained by the count sketch for x_i. Let T denote the largest k coordinates in absolute value in x'. The recovery algorithm outputs x'_T as a k-sparse approximation to x.

Note that the count sketch data structure takes $B \cdot R = O(\epsilon^{-2}k \log n)$ measurements. Its update time is $O(R) = O(\log n)$ time, and the time to obtain x' is $O(n \cdot R) = O(n \log n)$. Taking largest k coordinates from x' requires $O(n \log n)$ time. Thus the total query time is $O(n \log n)$. In the following, we show that $\|x - x'_T\|_H \leq (1+\epsilon)\|x_{-k}\|_H$.

By Lemma 15.5.1, for each $i \in [n]$, with probability $1 - 1/\operatorname{poly}(n)$,

$$(x_i - x'_i)^2 \leq \frac{\epsilon^2}{10k}\|x_{-10k/\epsilon^2}\|_2^2.$$

Let S denote the largest k coordinates in absolute value in x. The recovery error is

$$\|x - x'_T\|_H = \sum_{i \in T} H(x_i - x'_i) + \sum_{i \in \overline{T}} H(x_i).$$

By point estimation guarantee and results from Tukey set query in Theorem 15.2.1,

$$\sum_{i \in T} H(x_i - x'_i) \leq |T| \cdot H(\frac{\epsilon}{\sqrt{10k}}\|x_{-4k}\|_2) \leq \frac{\epsilon}{3}\|x_{-k}\|_H.$$

Note that

$$\begin{aligned}\sum_{i \in \overline{T}} H(x_i) &= \left(\sum_{i \in \overline{T}} H(x_i) - \sum_{i \in \overline{S}} H(x_i)\right) + \sum_{i \in \overline{S}} H(x_i) \\ &= \left(\sum_{i \in S \cap \overline{T}} H(x_i) - \sum_{i \in \overline{S} \cap T} H(x_i)\right) + \|x_{-k}\|_H,\end{aligned}$$

where the second step follows by canceling out the coordinates in $\overline{S} \cap \overline{T}$. In the following, our goal is to show that $\sum_{i \in S \cap \overline{T}} H(x_i) - \sum_{i \in \overline{S} \cap T} H(x_i) \leq \frac{2}{3}\epsilon \|x_{-k}\|_H$.

If $S \cap \overline{T} = \emptyset$, then $S = T$ and $\sum_{i \in S \cap \overline{T}} H(x_i) - \sum_{i \in \overline{S} \cap T} H(x_i) = 0$. In the following, we assume that $S \cap \overline{T} \neq \emptyset$. Let $b = \max_{i \in S \cap \overline{T}} |x_i|$, and $a = \min_{i \in \overline{S} \cap T} |x_i|$. Because Huber function $H(t)$ is monotonically increasing as $|t|$ increases,

$$\sum_{i \in S \cap \overline{T}} H(x_i) - \sum_{i \in \overline{S} \cap T} H(x_i) \leq |\overline{S} \cap T| \cdot (H(b) - H(a)).$$

By Lemma 15.7.5

$$b - a \leq \frac{\epsilon}{\sqrt{10k}} \|x_{-10k/\epsilon^2}\|_2.$$

In the following, we are going to prove that $H(b) - H(a) \leq \frac{\epsilon}{2} \frac{\|x_{-k}\|_H}{|\overline{S} \cap T|}$. We discuss in three cases: case (1) $\tau \leq a$, case (2) $a < \tau < b$, case (3) $b \leq \tau$.

Case (1) $\tau \leq a$.

In this case $H(a) = H(b) = 1$ and $H(b) - H(a) = 0$.

Case (2) $a < \tau < b$.

Let $P = \{i \in \overline{S} \mid |x_i| < \tau\}$, and $Q = \{i \in \overline{S} \mid |x_i| \geq \tau\}$. We have

$$\|x_{-k}\|_H = |Q| + \|x_P\|_2^2$$

and

$$\begin{aligned} H(b) - H(a) &= 1 - a^2 = (1 - a)(1 + a) \\ &\leq \frac{\epsilon \|x_{-10k/\epsilon^2}\|_2}{\sqrt{k}} \cdot (1 + a) \leq \frac{2\epsilon \|x_{-10k/\epsilon^2}\|_2}{\sqrt{k}}. \end{aligned}$$

If $|Q| > \epsilon^{-2}k$, then we have $H(b) - H(a) \leq 1$ while $\|x_{-k}\|_H \geq |Q| > \epsilon^{-2}k$. Thus $H(b) - H(a) \leq \frac{\epsilon}{k}\|x_{-k}\|_H \leq \frac{\epsilon}{|S\cap\overline{T}|}\|x_{-k}\|_H$.

If $|Q| \leq \epsilon^{-2}k$ and $\|x_{-10k/\epsilon^2}\|_2 > \sqrt{k}$, then $\|x_P\|_2 \geq \|x_{-10k/\epsilon^2}\|_2 > \sqrt{k}$, and so $\|x_{-10k/\epsilon^2}\|_2 \leq \frac{\|x_P\|_2}{\sqrt{k}} \cdot \|x_P\|_2 = \frac{\|x_P\|_2^2}{\sqrt{k}} \leq \frac{1}{\sqrt{k}}\|x_{-k}\|_H$. Thus $H(b) - H(a) \leq \frac{2\epsilon}{k}\|x_{-k}\|_H \leq \frac{2\epsilon}{|S\cap\overline{T}|}\|x_{-k}\|_H$.

If $|Q| \leq \epsilon^{-2}k$ and $\|x_{-10k/\epsilon^2}\|_2 \leq \sqrt{k}$. We have $\|x_P\|_2 \geq \|x_{-10k/\epsilon^2}\|_2$. By definition that $a = \min_{i\in\overline{S}\cap T}|x_i|$, $a \leq \frac{\|x_{\overline{S}\cap T}\|_2}{\sqrt{|\overline{S}\cap T|}} \leq \frac{\|x_P\|_2}{\sqrt{|\overline{S}\cap T|}}$. Therefore,

$$\begin{aligned} H(b) - H(a) &\leq \frac{\epsilon\|x_{-10k/\epsilon^2}\|_2}{\sqrt{k}} \cdot \left(\frac{\epsilon\|x_{-10k/\epsilon^2}\|_2}{\sqrt{k}} + 2\frac{\|x_P\|_2}{\sqrt{|\overline{S}\cap T|}}\right) \\ &\leq \frac{\epsilon\|x_P\|_2}{\sqrt{|\overline{S}\cap T|}} \cdot 3\frac{\|x_P\|_2}{\sqrt{|\overline{S}\cap T|}} \\ &\leq \frac{3\epsilon}{|\overline{S}\cap T|}\|x_P\|_2^2 \\ &\leq \frac{3\epsilon}{|\overline{S}\cap T|}\|x_{-k}\|_H. \end{aligned}$$

Case (3) $b \leq \tau$.

The analysis is similar to classical ℓ_2/ℓ_2 sparse recovery case. We have

$$\begin{aligned}
H(b) - H(a) &= \frac{b^2}{\tau} - \frac{a^2}{\tau} \\
&= \frac{1}{\tau}(b-a)(b+a) \\
&= \frac{1}{\tau}(b-a)(b-a+2a) \\
&\leq \frac{1}{\tau} \cdot \frac{\epsilon}{\sqrt{k}}\|x_{-k}\|_2 \cdot (\frac{\epsilon}{\sqrt{k}}\|x_{-k}\|_2 + 2\frac{\|x_{-k}\|_2}{\sqrt{|\overline{S} \cap T|}}) \\
&\leq \frac{1}{\tau} \cdot \frac{\epsilon\|x_{-k}\|_2}{\sqrt{|\overline{S} \cap T|}} \cdot (\frac{\|x_{-k}\|_2}{\sqrt{|\overline{S} \cap T|}} + 2\frac{\|x_{-k}\|_2}{\sqrt{|\overline{S} \cap T|}}) \\
&= \frac{1}{\tau} \cdot \frac{3\epsilon\|x_{-k}\|_2^2}{|\overline{S} \cap T|} \\
&= \frac{3\epsilon\|x_{-k}\|_H}{|\overline{S} \cap T|},
\end{aligned}$$

where the first step follows by definition of Huber function, the forth step follows by Eq. (15.7) and Eq.(15.8), the fifth step follows by $|\overline{S} \cap T| \leq |T| = k$ and $\epsilon \leq 1$, the last step follows because by definition of $b = \max_{i \in S \cap \overline{T}} |x_i|$, for any $j \in [n] \backslash S$, $|x_j| \leq \min_{i \in S} |x_i| \leq b \leq \tau$ and then by definition of Huber function $\|x_{-k}\|_H = \frac{1}{\tau}\|x_{-k}\|_2^2$. □

Part IV

Matrix Factorizations

Chapter 16

Weighted Low Rank Approximation

The classical low rank approximation problem is: given a matrix A, find a rank-k matrix B such that the Frobenius norm of $A - B$ is minimized. It can be solved efficiently using, for instance, the Singular Value Decomposition (SVD). If one allows randomization and approximation, it can be solved in time proportional to the number of non-zero entries of A with high probability.

Inspired by practical applications, we consider a *weighted* version of low rank approximation: for a non-negative weight matrix W we seek to minimize $\sum_{i,j}(W_{i,j} \cdot (A_{i,j} - B_{i,j}))^2$. The classical problem is a special case of this problem when all weights are 1. Weighted low rank approximation is known to be NP-hard, so we are interested in a meaningful parametrization that would allow efficient algorithms.

In this paper we present several efficient algorithms for the case of small k and under the assumption that the weight matrix W is of low rank, or has a small number of distinct columns. An important feature of our algorithms is that they do not assume anything about the matrix A. We also obtain lower bounds that show that our algorithms are nearly optimal in these parameters. We give several applications in which these parameters are small. To the best of our knowledge, the present paper is the first to provide algorithms for the weighted low rank approximation problem with provable guarantees.

Perhaps even more importantly, our algorithms proceed via a new technique, which we call "guess the sketch". The technique turns out to be general enough to give solutions to several other fundamental problems: adversarial matrix completion, weighted non-negative matrix factorization and tensor completion.

16.1 Introduction

Low rank approximation is arguably one of the most well-studied problems in randomized numerical linear algebra, with diverse applications to clustering [DFK+04, FSS13, LBKW14, CEM+15], data mining [AFK+01], distance matrix completion [Cha12], information retrieval [PRTV00], learning mixtures of distributions [AM05, KSV08], recommendation systems [DKR02], and web search [AFKM01, Kle99]. In practice one often has a low rank matrix which has been corrupted with noise of bounded norm, and low rank approximation allows one to approximately recover the original matrix. Low rank approximation may also help explain a dataset, revealing low dimensional structure in high dimensional data. Given a low rank approximation, one can store a matrix and compute a matrix-vector product much more efficiently by storing the corresponding factorization. It can also be used as a preprocessing step in applications, that is, by first projecting data onto a lower-dimensional subspace one preserves important properties of the input, but can now run subsequent algorithms in the lower-dimensional space. For example, it has been proposed to reduce the data dimension in Non-Negative Matrix Factorization [LS00] (more on this below), and Latent Dirichlet Allocation (LDA) [BNJ01].

The basic low rank approximation problem is: given an $n \times n$ matrix A, find a matrix $\widehat{A}$ of rank at most k for which $\|A - \widehat{A}\|_F$ is minimized, where for a matrix B, $\|B\|_F = \left(\sum_{i,j} B_{i,j}^2\right)^{1/2}$ is its Frobenius norm. This formulation intuitively corresponds to the matrix $\widehat{A}$ capturing as much of the variance of A as possible. It is well-known that the optimal solution is given by A_k, which if $U\Sigma V^\top$ is the singular value decomposition (SVD) of A, where U and V are orthogonal matrices and Σ is a non-negative diagonal matrix with $\Sigma_{1,1} \geq \Sigma_{2,2} \geq \cdots \geq \Sigma_{n,n}$, then $A_k = U\Sigma_k V^\top$, where Σ_k agrees with Σ on its first k

diagonal entries and is 0 otherwise. Although the SVD is computable in polynomial time, it is often acceptable to output a matrix $\widehat{A}$ for which $\|A - \widehat{A}\|_F \leq (1+\epsilon)\|A - A_k\|_F$ with high probability. In the latter case, much more efficient algorithms are known, and it is possible to compute such an $\widehat{A}$ in $\mathrm{nnz}(A) + n \cdot \mathrm{poly}(k/\epsilon)$ time, where $\mathrm{nnz}(A)$ denotes the number of non-zero entries of A [CW13, MM13, NN13a]. We note that for typical applications k and $1/\epsilon$ are assumed to be much smaller than n, e.g., in [Har14] they are treated as absolute constants.

Despite the large body of work on low rank approximation, the *weighted* case is not well understood. In this case one is given an $n \times n$ matrix A and an $n \times n$ matrix W with $W_{i,j} \geq 0$, and one seeks to solve:

$$\min_{\text{rank-}k \text{ matrices } \widehat{A}} \|W \circ (A - \widehat{A})\|_F^2 = \min_{\text{rank-}k \text{ matrices } \widehat{A}} \sum_{i,j} W_{i,j}^2 (A_{i,j} - \widehat{A}_{i,j})^2.$$

The classical low rank approximation is a special case in which $W_{i,j} = 1$ for all i and j. However, in general there may not be a good reason to weight all elements of the approximation error $A - \widehat{A}$ equally, especially if one is given prior knowledge about the distribution of the errors. For example, suppose the columns of A each come from a low-dimensional subspace but one of the columns is then shifted by a fixed large vector so that its mean is different. One may first want to recenter the data by subtracting off the mean from each of the columns. While this is possible without weighted low rank approximation, suppose instead that each of the columns of A comes from a perturbation of columns in a low dimensional subspace but one of the columns has a much larger variance. Then if all weights were equal, it would be enough for $\widehat{A}$ to fit this one single large variance column, which fails to capture the entire low-dimensional subspace. One way of fixing this is to reweight each entry of A

by the inverse of its variance. This is a common technique used in gene expression analysis, where the error model for microarray measurements provides entry-specific noise estimates, or when entries of A represent aggregates of many samples such as in word co-occurrence matrices and non-uniform weights are needed to appropriately capture any differences in the sample sizes; see [SJ03] for a discussion, and also the Wikipedia entry on weighted low rank approximation for a brief introduction.

While the extension of low rank approximation to the weighted case goes back to work of Young in 1940 [You40], its complexity is not well-understood, partly because the weighted case does not admit a solution via the SVD and may have many local minima [SJ03]. Early work by Shpak [Shp90] looked at gradient-based approaches while Lu et al. [LPW97, LA03] looked at alternating minimization methods. These were significantly sped up in practice by the work of Srebro and Jaakkola [SJ03], with success in various applications such as color image restoration [MES08], though there are no provable time bounds and in the worst case the running times could be exponential or worse. In fact, weighted low rank approximation is known to be NP-hard to approximate up to a $(1 \pm 1/\operatorname{poly}(n))$ factor [GG11]. We note that this also follows from the fact that matrix completion, arguably one of the most important special cases of weighted low rank approximation in which case all weights are 0 or 1, which we discuss more below, is also known to be NP-hard [Pee96, HMRW14]. Typically, though, assumptions such as incoherence and randomly sampled entries allow one to circumvent this hardness [CR09, LLR16]. There is some debate as to whether these assumptions are valid, for instance in [SW15] an argument is made why randomly missing entries may not hold for real-world datasets.

Many natural questions are left open from previous work. In particular the main

question as we see it is the following:

- *For which weight matrices W is the problem tractable? More generally, is it possible to identify a natural parameter of W and to obtain parameterized complexity bounds in terms of that parameter?*

16.1.1 Our Results

In this work we provide an answer to the above questions by parameterizing the complexity of the problem in terms of the *rank of the weight matrix* W and the *number of distinct columns of the weight matrix.* Note that we make *no assumptions* about the input matrix A. Let OPT denote the quantity $\min_{\text{rank-}k \text{ matrices } \widehat{A}} \|W \circ (\widehat{A} - A)\|_F^2$. Our main theorems are the following. For a function f, define $\widetilde{f} = f \cdot \text{poly}(\log(f))$.

Theorem 16.1.1 (Algorithm for Weighted Low Rank Approximation)**.** *Let r be the rank of W. There is an algorithm running in time $n^{O(k^2 r/\epsilon)}$ which outputs a factorization (into an $n \times k$ and a $k \times n$ matrix) of a rank-k matrix $\widehat{A}$ for which*

$$\|W \circ (A - \widehat{A})\|_F^2 \leq (1 + \epsilon)\,\text{OPT},$$

with probability at least $9/10$.

We also have the following theorem.

Theorem 16.1.2 (Algorithm for Weighted Distinct Columns)**.** *Let r be the number of distinct columns of W. For every $\epsilon > 0$ and for an arbitrarily small constant $\gamma > 0$, there is an algorithm running in time $O((\text{nnz}(A) + \text{nnz}(W)) \cdot n^{\gamma}) + n \cdot 2^{\widetilde{O}(k^2 r^2/\epsilon)}$ which outputs a factorization (into an $n \times k$ and a $k \times n$ matrix) of a rank-k matrix $\widehat{A}$ for which*

$$\|W \circ (A - \widehat{A})\|_F^2 \leq (1 + \epsilon)\,\text{OPT},$$

with probability at least $9/10$.

Note that Theorem 16.1.2 does not make any assumptions on the number of distinct rows, which is important for the applications described below.

Let us point out two things here:

- Before only the case of W having rank 1 was known to be provably solvable in polynomial time by a direct reduction to the SVD[1];
- The result for at most r distinct columns implies that, with respect to this parametrization, the weighted low rank approximation problem is fixed-parameter tractable.

We complement the above positive results by proving a lower bound, which assumes the Exponential Time Hypothesis for the average case hardness of random 4-SAT. We state the assumption as follows.

Assumption 16.1.3 ("Random Exponential Time Hypothesis")**.** *Let $c > \ln 2$ be a constant. Consider a random 4-SAT formula on n variables in which each clause has 4 literals, and in which each of the $16n^4$ clauses is picked independently with probability c/n^3. Then any algorithm which always outputs 1 when the random formula is satisfiable, and outputs 0 with probability at least $1/2$ when the random formula is unsatisfiable, must run in $2^{c'n}$ time on some input, where $c' > 0$ is an absolute constant.*

We are not aware of prior work using this form of the Exponential Time Hypothesis, though both the Exponential Time Hypothesis and Feige's original assumption [Fei02] that there is no polynomial time algorithm for the problem in Assumption 16.1.3 are commonly

[1]Namely, in that case, we can rewrite the problem as $\min_{\text{rank-}k \text{ matrices } \widehat{A}} \|D(\widehat{A} - A)E\|_F^2$ for diagonal matrices D and E, from which we can replace A with DAE and solve the unweighted low rank approximation, obtaining an $\widehat{A}$ for which we can then output $D^{-1}\widehat{A}E^{-1}$ if the diagonal entries of D and E are non-zero. If they are zero we can first remove rows and columns from A

used. We do not know of any better algorithm for the problem in Assumption 16.1.3 and have consulted several experts[2] about the assumption who do not know a counterexample to it. Instead of calling it "Random-ETH", some experts[3] suggest other names, e.g. average case ETH or Feige-ETH.

Theorem 16.1.4 (Weighted Low Rank Approximation Hardness)**.** *Let r be an upper bound on the number of distinct columns of W. Under Assumption 16.1.3, there is an absolute constant $\epsilon_0 \in (0,1)$ for which any algorithm for solving the weighted low rank approximation algorithm with $\epsilon \leq \epsilon_0$ and for any $k \geq 1$, with constant probability, requires $2^{\Omega(r)}$ time. Further, this holds even if W also only has r distinct rows.*

Note that for constant k and ϵ, and $r \geq C \log n$ for a constant $C > 0$, our upper bound assuming at most r distinct columns is $2^{\widetilde{O}(r^2)}$, which nearly matches our lower bound in Theorem 16.1.4 of $2^{\Omega(r)}$.

There are naturally arising applications in which the rank of the weight matrix or the number of distinct columns is small. Consider a matrix in which the rows correspond to users and the columns correspond to ratings of a movie, such as in the Netflix matrix. Further, suppose for each movie, there are r columns, indicating different aspects of the movie to be rated, such as acting, plot, sound effects, visual effects, etc. For a given user, one can look at the distribution of scores across movies along one of these aspects. These distributions may have different variances for that user and one can renormalize the scores by the reciprocal of their variance. In this case, the weight matrix consists of r distinct columns, one for each

[2]Personal communication with Russell Impagliazzo and Ryan Williams.
[3]Personal communication with Lijie Chen and Aviad Rubinstein.

aspect, each copied n/r times, where n/r is the total number of movies. Each entry of a column is a variance for a certain user for that aspect. This naturally generalizes to other applications for which the columns can be clustered into r groups, such as in stochastic block models or more general latent space models [Cha12]. Also, in some of these applications, one would want a low rank nonnegative factorization, and we remark that we can achieve this below.

Suppose now that A has constant rank k. A consequence of Theorem 16.1.1, which we elaborate on more below, is that even if an adversary deletes up to $O(1)$ entries in each column of A in an arbitrary way, one can still recover a matrix $\widehat{A}$ of rank at most k which agrees with A on all of the remaining entries in $\text{poly}(n)$ time. This is a form of *adversarial* matrix completion, which was studied in [HKZ11, SW15]. In these works the authors had to make an incoherence assumption on the entries of A, which may not always hold, e.g., in the presence of outliers. Without this assumption the prior results would not be able to recover such an $\widehat{A}$ even if a single adversarially chosen entry was deleted in each column.

16.1.2 Other Results

Our techniques can be applied to weighted non-negative factorization, a problem that has been extensively studied both in the unweighted (see, e.g., [AGKM12, Moi13]) and weighted cases (see, e.g., [GVS03, GZD10, KG12, ZZGM15, YC09]). In this problem, one seeks *non-negative* matrices $U \in \mathbb{R}^{n\times k}$ and $V \in \mathbb{R}^{k\times n}$ for which

$$\forall i, j \text{ such that } W_{i,j} > 0,\ A_{i,j} = (U \cdot V)_{i,j},$$

where the unweighted case corresponds to $W_{i,j} = 1$ for all i and j. This problem naturally arises in applications such as topic modeling in which negative entries do not make sense. In a beautiful line of work [AGKM12, Moi13], an upper bound of $n^{O(k^2)}$ was established for the unweighted case. By combining their techniques with ours, we obtain the first results for the weighted case when the weight matrix W has rank at most r. We defer the details to the full version of the paper.

Finally, there is a simple bi-criteria approximation algorithm for weighted low rank approximation when the weight matrix has entries that are all either 0 or 1. In this case it is possible to obtain a rank-rk approximation to A in $\text{nnz}(A) + \text{nnz}(W) + n \cdot \text{poly}(rk/\epsilon)$ time with cost at most $(1+\epsilon)$ times the cost of the best rank-k approximation. Here r denotes the rank of the weight matrix. This has a better dependence on r and could be useful when one does not need to output a matrix of rank exactly k. This is given in the full version of the paper.

16.1.3 Our Techniques

To prove Theorem 16.1.1, we first prove a structural theorem about regression. Then our main new theme is what we call "guessing a sketched matrix".

Structural Regression Theorem Given any number t of multiple response regression instances, i.e., instances of the form $\min_{X^1,X^2,\cdots,X^t} \|A^iX^i - B^i\|_F^2$, where $A^1,\ldots,A^t$ and $B^1,\ldots,B^t$ are matrices and each A^i has rank at most k, if one is interested in simultaneously minimizing the sum of their costs, i.e., the objective function is $\min_{X^1,X^2,\cdots,X^t} \sum_{i=1}^t \|A^iX^i - B^i\|_F^2$, then one can choose a Gaussian matrix S with $O(k/\epsilon)$ rows and if one solves the problem $\min_{Y^1,Y^2,\cdots,Y^t} \sum_{i=1}^t \|SA^iY^i - SB^i\|_F^2$, then the minimizers $Y^1, Y^2, \cdots, Y^t$ for this latter problem satisfy $\sum_{i=1}^t \|A^iY^i - B^i\|_F^2 \leq (1+\epsilon) \min_{X^1,X^2,\cdots,X^t} \sum_{i=1}^t \|A^iX^i - B^i\|_F^2$ with high constant probability. Interestingly, the number of rows of S does not depend on the number t of regression instances, and is optimal already for $t = 1$ and when B^1 only has a single column [CW09]. This also generalizes affine embeddings in [CW13] in which $t = 1$ and B^1 may have multiple columns; we stress that the design matrices A^i may be different. The proof uses a novel observation that a Gaussian S is a *subspace embedding on average over i*. This part uses a tail bound from [CD05] for the condition number for a Gaussian matrix. Having this, we bootstrap the approximation to $(1 \pm \epsilon)$ using the approximate matrix product property of a Gaussian matrix.

Guessing a Sketched Matrix Given our structural result, the main new theme of this paper is to "guess a sketched matrix". Suppose one is given an optimization problem of the

form

$$p_1(x_1, \ldots, x_v) \geq 0, p_2(x_1, \ldots, x_v) \geq 0, \ldots, p_m(x_1, \ldots, x_v) \geq 0,$$

where each p_i is a polynomial of degree at most d, the $x_1, \ldots, x_v$ are indeterminates over the reals, and we are are interested in an assignment to $x_1, \ldots, x_v$ which simultaneously satisfies these c polynomial inequalities. Then this can be solved in time $(md)^{O(v)}$ using generic solvers [Ren92d, Ren92b, Ren92a, BPR96], and such techniques have been used in the context of database theory [EFW10], non-negative matrix factorization [AGKM12, Moi13], learning mixtures of Gaussians [LS15], computing approximate PSD factorizations [BDL16], and solving small-scale mixed-norm low rank approximation instances [CW15a]. There are two kinds of algorithms for semi-algebraic sets: the ones from [Ren92a, Ren92b, BPR96] are able to determine if a given sem-algebraic set is empty or not. The one from [Ren92d] is able to return a δ-approximate solution to a given semi-algebraic formulae by paying an extra factor of $\log(\frac{1}{\delta})$ in the running time. For weighted low-rank approximation, there are two ways to output the matrices. One is using the algorithm from [Ren92d] directly. Another option is to perform binary search using the algorithm from [Ren92a, Ren92b, BPR96] for the entries of $\widehat{A}_{ij}$ one by one.

Our main idea here is to use polynomial optimization for large-scale non-convex optimization by combining it directly with sketching. For example, suppose one is given the multiple response regression problem $\min_V \|UV - A\|_F^2$ in which the number of columns of U is small. The twist though, is that *both U and V are unknown*! Then UV is just a low rank approximation to A, and if we knew U we could solve the sketched optimization problem $\min_X \|SUV - SA\|_F^2$, for a random oblivious sketching matrix S, and our solution V would

be a good solution to the original problem. Since we do not know U, we instead choose a random S and create variables for $S \cdot U$, which is small, and also compute $S \cdot A$, which we know. We then solve a regression problem for V in terms of the variables that we created for $S \cdot U$. Given V, we can then plug it into the original regression problem and solve for U in terms of V which is in turn in terms of our variables for $S \cdot U$. Finally we can verify the solution by requiring that $\|UV - A\|_F^2$ is small, which is now a system in a small number of variables. This verification step is essential because our S only has a probabilistic guarantee that it works for a fixed U with good probability, but crucially, we know there *exists* a U for which it works, and so by doing the verification step we will find such a U. We note there are several issues with this approach which we discuss below.

While this may seem like an unnecessarily complicated way of doing standard low rank approximation, this idea proves crucial for weighted low rank approximation. In this case, using say, the rank constraint on the weight matrix W, and using our structural result on multiple instances of regression, we are able to choose a single sketching matrix S and create variables for only r regression problems, $SD_{W_1}U, \ldots, SD_{W_r}U$, where $D_{W_1}, \ldots, D_{W_r}$ are diagonal matrices with independent columns of W on each diagonal, and U is a fixed optimal solution. We can then try to express all regression solutions $\min_{X^i} \|D_{W_i}UX^i - D_{W_i}A^i\|_2^2$, for $i = 1, \ldots, n$ and where X^i and A^i are the i-th columns of X and A respectively, in terms of these variables, and hope to carry out the procedure above.

Dealing with Linear Dependencies At this point another obstacle arises which is that when the columns of a given $D_{W_i}U$ are not linearly independent, there is no way to write down the pseudoinverse of $D_{W_i}U$ in terms of the variables we have created. We also cannot afford

to create more than $r \cdot \text{poly}(k/\epsilon)$ variables, since our optimization procedure is exponential in this quantity, and so we cannot create new variables for each $i = 1, \ldots, n$. To get around this, we observe that there is a solution $U \cdot V$ for which for all i, the first $\min(|\text{supp}(D_{W_i})|, k)$ columns of U are linearly independent, where $\text{supp}(D_{W_i})$ denotes the set of non-zero entries of W_i, and further the solution cost of $U \cdot V$ is an arbitrarily small amount larger than that of the optimal cost. This follows by a simple perturbation argument applied to the optimal solution. This immediately gives an algorithm with additive error when we parameterize W by its rank.

In order to turn it into a relative error algorithm, we need a lower bound on the cost assuming that the cost is non-zero. The main idea here is that if we correctly guess which subsets of columns are linearly independent for the different matrices D_{W_i}, then we can set up a non-negative polynomial system and provided this system has non-zero cost, we can apply known lower bounds on the cost of polynomial optimazation problems as a function of the degrees, number of variables, number of constraints, and coefficient sizes [Bas14]. While this is not an algorithmic procedure, since we cannot afford to make guess for each D_{W_i} without spending exponential in n time, it suffices for lower bounding the cost. Given such a lower bound, the above perturbation argument can then be used to argue that we achieve relative error.

While this leads to our time bound in the case in which we parameterize the weight matrix by its rank, in the case in which we parameterize by the number of distinct columns of W, this is too slow. The issue is that we have at least n constraints to enforce, namely, that the first $\min(\text{supp}(W_i), k)$ columns of $SD_{W_i}U$ are linearly independent. This would lead to a running time of $n^{\text{poly}(rk/\epsilon)}$ as opposed to the $\text{poly}(n) \cdot 2^{\text{poly}(kr/\epsilon)}$ that we desire. We

notice though that when we have r distinct columns, there are only r constraints to enforce, one for each distinct column. These are "not equal" constraints but can be transformed to a single equality constraint of degree $\text{poly}(rk/\epsilon)$ by introducing a single auxiliary variable. While this ultimately enables us to write down all the entries of V using only a $\text{poly}(rk/\epsilon)$ number of variables, we are then faced with the task of writing down U in terms of such a V. Here a priori we could have many distinct *rows* in W, and may have n constraints to enforce. We observe though that since the entries of W are integers in $\{1, 2, \ldots, \text{poly}(n)\}$, if we round them to the nearest power of $1+\epsilon$, the solution cost changes only by a $(1+\epsilon)$-factor. Moreover, given that we have r distinct columns, for any row it is entirely specified on these r columns and after rounding, there are only $O((\log n)/\epsilon)^r$ choices for entries on these r columns, which upper bounds the number of distinct rows. This ultimately allows us to write down U in terms of V with only $O((\log n)/\epsilon)^r$ not equal constraints. This ultimately yields our improved running time when parameterizing W by its number of distinct columns.

Dealing with Rational Functions There is a subtle problem with the above arguments. When one solves for the i-th column V^i of V in terms of $SD_{W_i}U$, the entries of V^i are *rational functions* rather than polynomials, and we cannot afford to clear the denominators of V^i for every i without blowing up the degree of the polynomials to $\Omega(n)$, which would give a running time of $n^{\text{poly}(kr/\epsilon)}$. While this is not a problem when we parameterize the problem by the rank of W, since we anyway spend this amount of time, this is a problem when we parameterize by the number of distinct columns of W, in which we seek polynomial time even for super-constant r (and constant k/ϵ). Instead, we write V as $V' \cdot D$, where V' has entries which are polynomials, and D is a diagonal matrix whose entries are $\frac{1}{\det(U^\top D_{W_i} S^\top S D_{W_i} U)}$,

given by Cramer's rule. The entries of D are rational functions, and we would like to make them polynomials. Since W has at most r distinct columns, D has at most r distinct entries and we can create r new variables for the entries of D. However, when we try to solve for U in terms of V we face the same problem again: we can write U as $E \cdot U'$, where U' only has polynomial entries, but since we do not assume a small number of distinct rows of W, it follows that E could have n distinct entries $\frac{1}{\det(U^\top D_{W^i} S^\top S D_{W^i} U)}$ for $i = 1, \ldots, n$, where D_{W^i} is the diagonal matrix with the i-th row of W on the diagonal. As mentioned above, we fix this problem by rounding the entries of W to powers of $1 + \epsilon$, and then observing that the number of distinct rows of W can only be $O((\log n)/\epsilon)^r$ after rounding.

Other Methods Our techniques for weighted non-negative matrix factorization and tensor completion largely follow these ideas as well, where we combine our "guessing a sketched matrix" approach with techniques of Arora, Ge, Kannan and Moitra [AGKM12] and Moitra [Moi13]. Our bi-criteria solution directly follows from known sketching results.

16.2 Preliminaries

Notation Let $\mathbb{R}$ denote the real numbers, and $\mathbb{R}_{\geq 0}$ denote the nonnegative real numbers. Let $\|A\|$ (and sometimes $\|A\|_2$) denote the spectral norm of matrix A. Let $\|A\|_F^2 = \sum_{i,j} A_{i,j}^2$ denote the Frobenius norm of A. Let $W \circ A$ denote the entry-wise product of matrices W and A. Let $\|A\|_W^2 = \sum_{i,j} W_{i,j}^2 A_{i,j}^2$ denote the weighted Frobenius norm of A. Let $\text{nnz}(A)$ denote the number of nonzero entries of A. Let $\det(A)$ denote the determinant of a square matrix A. Let $A^\top$ denote the transpose of A. Let $A^\dagger$ denote the Moore-Penrose pseudoinverse of A. Let A^{-1} denote the inverse of a full rank square matrix A.

For the weight matrix W, we always use W_j to denote the j-th column vector of W, and W^i to denote the i-th row of W. Let D_{W_j} denote the diagonal matrix with entries from the column vector W_j and D_{W^i} denote the diagonal matrix with entries from the row vector W^i.

The following real algebraic geometry definitions are needed when proving a lower bound for the minimum nonzero cost of our problem. For a full discussion, we refer the reader to Bochnak *et al.* [BCR87]. Here we use the brief summary by Basu *et al.*[BPR05].

Definition 16.2.1 ([BPR05])**.** Let R be a real closed field.

Given $x = (x_1, \cdots, x_v) \in R^v, r \in R, r > 0$, we denote

$$B_v(x, r) = \{y \in R^v \mid \|y - x\|^2 < r^2\} \qquad \text{(open ball)},$$
$$\bar{B}_v(x, r) = \{y \in R^v \mid \|y - x\|^2 \leq r^2\} \qquad \text{(closed ball)}.$$

A set $S \subset R^v$ is open if it is the union of open balls, i.e., if every point of U is contained in an open ball contained in U.

A set $S \subset R^v$ is closed if its complement is open. Clearly, the arbitrary union of open sets is open and the arbitrary intersection of closed sets is closed.

Semi-algebraic sets are defined by a finite number of polynomial inequalities and equalities.

A semi-algebraic set has a finite number of connected components, each of which is semi-algebraic. Here, we use the topological definition of a connected component, which is a maximal connected subset (ordered by inclusion), where connected means it cannot be divided into two disjoint nonempty closed sets.

A closed and bounded semi-algebraic set is compact.

A semi-algebraic set $S \subset R^v$ is semi-algebraically connected if S is not the disjoint union of two non-empty semi-algebraic sets that are both closed in S. Or, equivalently, S does not contain a non-empty semi-algebraic strict subset which is both open and closed in S.

A semi-algebraically connected component of a semi-algebraic set S is a maximal semi-algebraically connected subset of S.

Renegar [Ren92a, Ren92b] and Basu *et al.* [BPR96] independently provided an algorithm for the decision problem for the existential theory of the reals is to decide the truth or falsity of a sentence $(x_1, \cdots, x_v)F(f_1, \cdots, f_m)$ where F is a quantifier-free Boolean formula with atoms of the form $\text{sign}(f_i) = \sigma$ with $\sigma \in \{0, 1, -1\}$. Note that this problem is equivalent to deciding if a given semi-algebraic set is empty or not. Here we formally state that theorem. For a full discussion of algorithms in real algebraic geometry, we refer reader to [BPR05] and [Bas14].

Theorem 16.2.1 (Decision Problem [Ren92a, Ren92b, BPR96]). *Given a real polynomial system $P(x_1, x_2, \cdots, x_v)$ having v variables and m polynomial constraints*

$$f_i(x_1, x_2, \cdots, x_v)\Delta_i 0, \forall i \in [m],$$

where Δ_i is any of the "standard relations": $\{>, \geq, =, \neq, \leq, <\}$, let d denote the maximum degree of all the polynomial constraints and let H denote the maximum bitsize of the coefficients of all the polynomial constraints. Then in

$$(md)^{O(v)} \operatorname{poly}(H)$$

time one can determine if there exists a solution to the polynomial system P.

The key result we used for proving lower bound is the following bound on the minimum value attained by an integer polynomial restricted to a compact connected component of a basic closed semi-algebraic subset of $\mathbb{R}^v$ defined by polynomials with integer coefficients in terms of the degrees and the bitsizes of the coefficients of the polynomials involved.

Theorem 16.2.2 (Jeronimo, Perrucci and Tsigaridas [JPT13]). *Let $T = \{x \in \mathbb{R}^v \mid f_1(x) \geq 0, \cdots, f_\ell(x) \geq 0, f_{\ell+1}(x) = 0, \cdots, f_m(x) = 0\}$ be defined by polynomials $f_1, \cdots, f_m \in \mathbb{Z}[x_1, \cdots, x_v]$ with $n \geq 2$, degrees bounded by an even integer d and coefficients of absolute value at most H, and let C be a compact connected component of T. Let $g \in \mathbb{Z}[x_1, \cdots, x_v]$ be a polynomial of degree at most d and coefficients of absolute value bounded by H. Then, the minimum value that g takes over C is satisfies that if it is not zero, then its absolute value is greater than or equal to*

$$(2^{4-v/2}\widetilde{H}d^v)^{-v2^v d^v},$$

where $\widetilde{H} = \max\{H, 2v + 2m\}$.

While the above theorem involves notions from topology, we shall apply it in an elementary way. Namely, in our setting T will be bounded and so every connected component, which is by definition closed, will also be bounded and therefore compact. As the connected components partition T the theorem will just be applied to give a global minimum value of g on T provided that it is non-zero.

16.3 Multiple Regression Sketch

Theorem 16.3.1. *Let $A^1, \ldots, A^m \in \mathbb{R}^{n \times k}$ be m matrices of size $n \times k$. Let $b^1, \ldots, b^m \in \mathbb{R}^{n \times 1}$ be m column vectors of dimension n.*

For $1 \leq i \leq m$ denote:

$$x^i = \operatorname*{argmin}_{x \in \mathbb{R}^{k \times 1}} \left\| A^i x - b^i \right\|_2^2$$

the solution of the i-th regression problem.

Let $S \in \mathbb{R}^{t \times n}$ be a random matrix with i.i.d. Gaussian entries with zero mean and standard deviation $1/\sqrt{t}$. For $1 \leq i \leq m$ denote:

$$y^i = \operatorname*{argmin}_{y \in \mathbb{R}^{k \times 1}} \left\| SA^i y - Sb^i \right\|_2^2$$

the solution of the i-th regression problem in the sketch space.

We claim that for every $0 < \epsilon < 1/2$ one can set $t = O(k/\epsilon)$ such that:

$$\sum_{i=1}^{m} \left\| A^i y^i - b^i \right\|_2^2 \leq (1+\epsilon) \cdot \sum_{i=1}^{m} \left\| A^i x^i - b^i \right\|_2^2.$$

The rest of this section is devoted to the proof of this theorem.

For $1 \leq i \leq m$ we let $D^i \geq 1$ denote the smallest number such that for every $x \in \mathbb{R}^{k \times 1}$ and $\lambda \in \mathbb{R}$ one has:

$$\left\| S\left(A^i x + \lambda b^i\right) \right\|_2^2 \in \left[\frac{1}{D^i}; D^i \right] \cdot \left\| A^i x + \lambda b^i \right\|_2^2.$$

Claim 16.3.2. *For every i*

$$\left\| A^i y^i - b^i \right\|_2^2 \leq \left(D^i\right)^2 \cdot \left\| A^i x^i - b^i \right\|_2^2.$$

Proof. This follows from the definition of D^i. □

Claim 16.3.3. *One can set $t = O(k/\epsilon)$ such that for every i*

$$\Pr_S\left[D^i \geq 1.01\right] \leq 2^{-\Omega(1/\epsilon)}.$$

Proof. This follows from Theorem 2.1 from [Woo14b]. □

Claim 16.3.4. *One can set $t = O(k/\epsilon)$ such that for every i*

$$\mathop{\mathbb{E}}_S\left[\left\|A^i y^i - b^i\right\|_2^2 - \left\|A^i x^i - b^i\right\|_2^2 \;\middle|\; D^i \leq 1.01\right] \leq \epsilon \cdot \left\|A^i x^i - b^i\right\|_2^2.$$

Proof. This follows from the proofs of Theorem 2.8 and Theorem 2.16 from [Woo14b] (adapted to Gaussian matrices). □

Claim 16.3.5. *One can set $t = O(k/\epsilon)$ such that for every i*

$$\mathop{\mathbb{E}}_S\left[\left(D^i\right)^2 \;\middle|\; D^i \geq 1.01\right] = O(1).$$

Proof. One can see that D^i is polynomially related to the condition number of a random $O(k/\epsilon) \times (k+1)$ matrix with i.i.d. Gaussian entries; indeed it corresponds to the maximum distortion of S applied to the vectors in the column span of an $n \times (k+1)$ orthonormal matrix U whose columns span the space spanned by the columns of A^i together with b^i. By rotational invariance, $S \cdot U$ also has i.i.d. Gaussian entries. To understand the condition number one can invoke the main result from [CD05] which gives for all sufficiently large t: $\Pr_S\left[D^i \geq t\right] = \frac{1}{t^{\Theta(k/\epsilon)}}$. Thus,

$$\mathop{\mathbb{E}}_S\left[\left(D^i\right)^2 \;\middle|\; D_i \geq 1.01\right] \leq O(1) + \int_{1.01}^{\infty} \frac{t^2}{t^{\Theta(k/\epsilon)}}\, dt = O(1).$$

□

Having these Claims, let us complete the proof. We have for every $1 \leq i \leq m$:

$$\begin{aligned}
& \mathop{\mathbb{E}}_{S}\left[\left\|A^i y^i - b^i\right\|_2^2 - \left\|A^i x^i - b^i\right\|_2^2\right] \\
= & \Pr_S\left[D^i \geq 1.01\right] \cdot \mathop{\mathbb{E}}_{S}\left[\left\|A^i y^i - b^i\right\|_2^2 - \left\|A^i x^i - b^i\right\|_2^2 \;\middle|\; D^i \geq 1.01\right] \\
+ & \Pr_S\left[D^i \leq 1.01\right] \cdot \mathop{\mathbb{E}}_{S}\left[\left\|A^i y^i - b^i\right\|_2^2 - \left\|A^i x^i - b^i\right\|_2^2 \;\middle|\; D^i \leq 1.01\right] \\
\leq & 2^{-\Omega(1/\epsilon)} \cdot \mathop{\mathbb{E}}_{S}\left[\left(D^i\right)^2 - 1 \;\middle|\; D^i \geq 1.01\right] \cdot \left\|A^i x^i - b^i\right\|_2^2 \\
+ & \Pr_S\left[D^i \leq 1.01\right] \cdot \mathop{\mathbb{E}}_{S}\left[\left\|A^i y^i - b^i\right\|_2^2 - \left\|A^i x^i - b^i\right\|_2^2 \;\middle|\; D^i \leq 1.01\right] \\
\leq & 2^{-\Omega(1/\epsilon)} \cdot \mathop{\mathbb{E}}_{S}\left[\left(D^i\right)^2 - 1 \;\middle|\; D^i \geq 1.01\right] \cdot \left\|A^i x^i - b^i\right\|_2^2 \\
+ & \epsilon \cdot \left\|A^i x^i - b^i\right\|_2^2 \\
\leq & O(\epsilon) \cdot \left\|A^i x^i - b^i\right\|_2^2,
\end{aligned}$$

where the second step is by Claim 16.3.2 and Claim 16.3.3, the third step is by Claim 16.3.4, and the fourth step is by Claim 16.3.5.

Summing over i and applying the Markov's inequality, we are done.

While the above result is for Gaussian sketching matrices, one can also combine a Gaussian random matrix with a Count-Sketch matrix [CW13]. This way we are still getting $O(k/\epsilon)$ rows, but now one can perform a matrix-vector multiplication in time proportional to the sparsity of the vector plus $\text{poly}(k\widetilde{r}/\epsilon)$, where $\widetilde{r}$ is the dimension of the union of column spaces of A^i (which is at most km in the worst case, but is much smaller for our applications).

16.4 Additive Approximation

In this Section, to demonstrate the new technique, we prove the following theorem.

Theorem 16.4.1. *Given $A, W \in \mathbb{R}^n$, $1 \leq k \leq n$ and $0 < \epsilon, \tau < 0.1$ such that:*

- $\operatorname{rank}(W) = r$*;*
- *all the non-zero entries of A and W are multiples of $\delta > 0$;*
- *all the entries of A and W are at most $\Delta > 0$ in absolute value,*

one can output a number Λ in time $n^{O(k^2 r/\epsilon)} \cdot \log^{O(1)}(\frac{\Delta}{\delta\tau})$ such that $\mathrm{OPT} \leq \Lambda \leq (1+\epsilon)\,\mathrm{OPT} + \tau$, where

$$\mathrm{OPT} = \min_{U \in \mathbb{R}^{n\times k}, V \in \mathbb{R}^{k\times n}} \|(UV - A) \circ W\|_F^2.$$

We first assume that W has no zero entries; later, we will remove this assumption by being slightly more careful.

Lemma 16.4.2.

$$\mathrm{OPT} \leq \mathrm{poly}(n, \Delta)$$

Proof. Set U and V to be the zero matrices. □

Let us expand the objective function in two ways. On the one hand:

$$\left\|(UV - A) \circ W\right\|_F^2 = \sum_{i=1}^{n} \left\|U^i V D_{W^i} - A^i D_{W^i}\right\|_2^2. \tag{16.1}$$

On the other hand:

$$\left\|(UV - A) \circ W\right\|_F^2 = \sum_{j=1}^{n} \left\|D_{W_j} U V_j - D_{W_j} A_j\right\|_2^2. \tag{16.2}$$

We can sketch (16.1) and (16.2) using Gaussian matrices $S' \in \mathbb{R}^{n \times t}$ and $S'' \in \mathbb{R}^{t \times n}$ as follows:

$$\sum_{i=1}^{n} \left\|U^i V D_{W^i} S' - A^i D_{W^i} S'\right\|_2^2.$$

and

$$\sum_{j=1}^{n} \left\|S'' D_{W_j} U V_j - S'' D_{W_j} A_j\right\|_2^2.$$

Denote, for $1 \leq i \leq n$, $P^i := V D_{W^i} S'$ and for $1 \leq j \leq n$ denote $Q_j := S'' D_{W_j} U$.

The crucial observation is that we can encode all P^i's and Q_j's using linear functions of $2krt$ variables, since W has rank r, and we can represent its rows/columns as linear combinations of r fixed rows/columns.

For fixed P^i's we define:

$$\widehat{U} := \operatorname*{argmin}_{U \in \mathbb{R}^{n \times k}} \sum_{i=1}^{n} \left\|U^i P^i - A^i D_{W^i} S'\right\|_2^2.$$

Similarly, for fixed Q_j's we define:

$$\widehat{V} := \operatorname*{argmin}_{V \in \mathbb{R}^{k \times n}} \sum_{j=1}^{n} \left\|Q_j V_j - S'' D_{W_j} A_j\right\|_2^2.$$

Let us use $\left\|\left(\widehat{U}\widehat{V} - A\right) \circ W\right\|_F^2$ as a proxy for the objective function. We need to argue that we can, in fact, minimize the new objective function efficiently, and that it gives a good approximation to the original objective function, if $t = \Theta(k/\epsilon)$, with high probability.

Optimization We assume that all P^i's and Q_j's have the maximum rank k. Later we will show that, since W has no zero entries, this does not affect the quality of the solution found under this non-degeneracy constraint.

Assuming non-degeneracy of P^i's and Q_j's, we can express $\widehat{U}$ and $\widehat{V}$ as follows:

$$\widehat{U}^i = A^i D_{W^i} S'(P^i)^\top \left(P^i(P^i)^\top\right)^{-1}$$

and

$$\widehat{V}_j = \left(Q_j^\top Q_j\right)^{-1} Q_j^\top S'' D_{W_j} A_j. \tag{16.3}$$

Since the entries of P^i's and Q_j's are linear functions of $2krt$ variables, we can represent the entries of $\widehat{U}$ and $\widehat{V}$ as rational functions over $2krt$ variables and of degree $O(k)$ (we use Cramer's formula for that).

Finally, we can represent $\left\| \left(\widehat{U}\widehat{V} - A\right) \circ W \right\|_F^2$ as a rational function over $2krt$ variables of degree $O(kn)$.

We can minimize the objective function using the algorithm for checking the feasibility of a system of polynomial inequalities from Theorem 16.2.1, together with a binary search over the value of the objective function. Each iteration of the binary search takes time

$$(\#\text{degree of the polynomials})^{O(\#\text{variables})} \cdot \text{poly}(\text{input size}).$$

Since the degree is $O(nk)$, the number of variables is $O(rkt) = O(k^2 r/\epsilon)$, and the input size is $\text{poly}(n, \log(\Delta/\delta))$, the running time of a single iteration of the binary search is

$$n^{O(k^2 r/\epsilon)} \cdot \log^{O(1)}(\Delta/\delta).$$

Finally, to perform the binary search, we need $O(\log(n\Delta/\tau)/\epsilon)$ iterations to check for the existence of a solution with cost at most $(1+\epsilon)\,\mathrm{OPT}+\tau$ (assuming that sketching and non-degeneracy constraints on P^i's and Q_j's increase the cost to at most $(1+\epsilon)\,\mathrm{OPT}+\tau$, which we will show later), since, by Lemma 16.4.2, $\mathrm{OPT} \leq \mathrm{poly}(n, \Delta)$. The overall running time is thus: $n^{O(k^2 r/\epsilon)} \cdot \log^{O(1)}(\frac{\Delta}{\delta\tau})$.

Near-optimality Here we show that one can set $t = O(k/\epsilon)$ so that, with high probability,

$$\min_{P^i, Q_j} \left\| \left(\widehat{U}\widehat{V} - A\right) \circ W \right\|_F^2 \leq (1+\epsilon) \cdot \min_{U,V} \left\| (UV - A) \circ W \right\|_F^2 + \tau'. \tag{16.4}$$

for every $\tau' > 0$. This, together with the above discussion about the optimization procedure, concludes the analysis of the algorithm.

As such, (16.4) follows from Theorem 16.3.1. Indeed, if the optimal solution is U^*V^*, then set $Q_j := S'' D_{W_j} U^*$. Q_j may be degenerate, but, since W has no zero entries, we can perturb U^* by an arbitrarily small amount to make Q_j non-degenerate (with probability one over S''). Then, with high probability over S'', we have

$$\left\| \left(U^*\widehat{V} - A\right) \circ W \right\|_F^2 \leq (1+\epsilon) \cdot \left\| (U^*V^* - A) \circ W \right\|_F^2 + \tau' \tag{16.5}$$

for an arbitrarily small $\tau' > 0$. Similarly, we can set $P^i = \widehat{V} D_{W^i} S'$ (again, it can be degenerate, but the same argument as above for Q_j applies), which gives, with high probability over S',

$$\left\| \left(\widehat{U}\widehat{V} - A\right) \circ W \right\|_F^2 \leq (1+\epsilon) \cdot \left\| \left(U^*\widehat{V} - A\right) \circ W \right\|_F^2 + \tau' \tag{16.6}$$

for an arbitrarily small $\tau' > 0$.

Combining (16.5) and (16.6), we are done.

16.4.1 Handling Weight Matrices With Zero Entries

Here we prove the version of Theorem 16.4.1 for the case when W is allowed to have zero entries. Let us first see what breaks in the previous argument.

What does not work anymore is that (after a small perturbation) $Q_j = S'' D_{W_j} U$ and $P^i = V D_{W^i} S'$ can be assumed to have the maximum possible rank k. Nevertheless, we can assume that every Q_j has rank equal to

$$t_j = \min(k, \text{the number of non-zero entries of } W_j).$$

Moreover, we can assume that the first t_j columns of Q_j are linearly-independent. A similar argument applies to P^i's as well.

The above argument allows us to express $\widehat{U}$ and $\widehat{V}$ as before, but instead of Q_j we use the first t_j columns of Q_j (and, similarly for the P^i's).

16.5 Multiplicative Approximation

In order to get a genuine multiplicative $(1+\epsilon)$-approximation, we need to lower bound OPT–provided that it is not equal to zero–which would allow us to set $\tau \leq \epsilon \cdot \mathrm{OPT}$ in the algorithm from the previous section.

We do this for the following optimization problem:

$$\min_{\substack{U,V: \\ \|UV\|_F^2 \leq (\Delta/\delta)^{\mathrm{poly}(n)}}} \|(UV - A) \circ W\|_F^2.$$

Note that we assume $\|UV\|_F$ has an upper bound, as otherwise we cannot write down U and V using $\mathrm{poly}(n)$ bits.

Using the approach outlined above, one can write down a rational function $p(x_1, \ldots, x_l)/q(x_1, \ldots, x_l)$ such that:

- $l = O(k^2 r/\epsilon)$;
- for every x such that $q(x) \neq 0$, one has $p(x)/q(x) \geq \mathrm{OPT}$;
- for every $\tau' > 0$, there exists x^* such that $p(x^*)/q(x^*) \leq (1+\epsilon)\,\mathrm{OPT} + \tau'$;
- both p and q are homogeneous, and their degrees are $O(kn)$;
- the coefficients of p and q are integers with absolute values at most $(\Delta/\delta)^{\mathrm{poly}(n)}$;
- $q(x) = \prod_{i=1}^{2n} g_i^2(x)$, where every $g_i(x)$ is the determinant polynomial.

Only the fifth item needs an explanation. If sketch matrices S' and S'' were integer, then the last item would hold automatically (by scaling up all the coefficients). But, in

reality, S' and S'' are Gaussian matrices. Fortunately, one can show that it is possible to discretize them up to $\pm 1/\operatorname{poly}(n)$ so that the multiple regression theorem (Theorem 16.3.1) still goes through. Indeed, this just follows from the fact that discretization to $\pm 1/\operatorname{poly}(n)$ preserves condition number and subspace embeddings (since one argues about preserving lengths of unit vectors), and approximate matrix product properties used in that theorem.

Lower Bound We use the same way explained in section 16.4 to create variables, write down the system in a small number of variables, and also create some "$\neq 0$" constraints. It will generate $2n$ determinant polynomials, which are defined in the following way,

$$\begin{aligned} g_i(x) &= \det((SD_{W_i}U)^\top_{P_i}(SD_{W_i}U)_{P_i}), \forall i \in [n] \\ g_{i+n}(x) &= \det((VD_{W^i}S)^{Q_i}((VD_{W^i}S)^{Q_i}))^\top), \forall i \in [n] \end{aligned}$$

where P_i and Q_i are maximal linearly indepdent subsets. Then we can write down the following optimization problem,

$$\begin{aligned} \min_{x\in\mathbb{R}^l} \quad & p(x)/q(x) \\ s.t. \quad & g_i^2(x) \neq 0, \forall i \in [2n], \\ & q(x) = \prod_{i=1}^{2n} g_i^2(x) \end{aligned}$$

To lower bound $p(x)/q(x)$, let us introduce a new variable y. Lower bounding $p(x)/q(x)$ is equivalent to lower bounding $p(x)y$ subject to $q(x)y - 1 = 0$. By assumption $\|U\|_F^2, \|V\|_F^2 \leq (\Delta/\delta)^{\operatorname{poly}(n)}$[4], we know the upper bound of all the variables we created

[4]A priori, we know only that $\|UV\|_F^2$ is bounded. But we can get that each of the matrices is bounded by taking an optimal solution and orthonormalizing one of the matrices.

except for y. In order to give an upper bound for y, it suffices to show a lower bound for $q(x)$. By definition, $q(x)$ is the square of the product of all the determinant polynomials. Thus, for every determinant polynomial $g_i(x)$, we need to show that if $g_i^2(x)$ is nonzero, then it is at least something.

Define B to be an ℓ_2-ball with bounded radius, e.g.,

$$B = \left\{ x \in \mathbb{R}^l \;\middle|\; \sum_{i=1}^{l} x_i^2 \leq (\Delta/\delta)^{\mathrm{poly}(n)} \right\}.$$

By [BPR05], we know that B is a closed and bounded semi-algebraic set. Thus B is also compact. Let x^* denote the optimal solution of the original problem $\min_{x\in\mathbb{R}^l} g_i^2(x)$ when all variables are bounded. Because the radius of the ball B is large enough, $x^* \in B$. Define $T_1 = \{x \in \mathbb{R}^l | g_i(x) \geq 0\}$ and let $T = T_1 \cap B$. By definition (see [BPR05]), T_1 is a basic closed semi-algebraic set. Thus, the intersection of T_1 and B is a semi-algebraic set with a finite number of connected components. Because B is compact and T_1 is closed, each of these connected components is compact. There must exist one compact connected component C which contains the optimal soltuion x^*. Applying Theorem 16.2.2 on system $\{T, C, g_i^2(x)\}$, we conclude that if $g_i^2(x)$ is not zero, then it is at least

$$\begin{aligned} \left((\Delta/\delta)^{\mathrm{poly}(n)}\right)^{-k^{O(l)}} &= (\Delta/\delta)^{-\mathrm{poly}(n)2^{\widetilde{O}(l)}} \\ &= (\Delta/\delta)^{-\mathrm{poly}(n)} \end{aligned}$$

which immediately gives us an upper bound for y,

$$y \leq ((\Delta/\delta)^{\mathrm{poly}(n)})^n = (\Delta/\delta)^{\mathrm{poly}(n)}.$$

Now, we are able to show a lower bound for $p(x)y$. Define

$$T_1 = \left\{ x \in \mathbb{R}^l, y \in \mathbb{R} \;\middle|\; \prod_{i=1}^{m} g_i^2(x) y - 1 = 0 \right\}.$$

Define B to be a bounded ball over $l+1$ variables,

$$B = \left\{ (x, y) \in \mathbb{R}^{l+1} \;\middle|\; \sum_{i=1}^{l} x_i^2 + y^2 \leq (\Delta/\delta)^{\mathrm{poly}(n)} \right\}.$$

By [BPR05], B is a closed and bounded semi-algebraic set. Thus B is also compact. Define $T = T_1 \cap B$. Let (x^*, y^*) denote the optimal solution of $\min_{(x,y)\in T_1} p(x)y$, then (x^*, y^*) is also the optimal solution of $\min_{(x,y)\in T} p(x)y$, because all variables are bounded and the radius of the ball is large enough.

By definition (see [BPR05]), T_1 is a basic closed semi-algebraic set. Thus, the intersection of T_1 and B is a semi-algebraic set with a finite number of connected components. Because B is compact and T_1 is closed, each of the connected components is compact.

There must exist a compact connected component that contains the optimal solution (x^*, y^*). Let C denote that component. Applying Theorem 16.2.2 on system $\{T, C, p(x)y\}$, where the number of constraints is bounded by $O(1)$, the maximum coefficient of absolute value is bounded by $(\Delta/\delta)^{\mathrm{poly}(n)}$, the maximum degree is bounded by $O(nk)$, the number of variables is bounded by $l = O(rk^2/\epsilon)$, we conclude that if the minimum cost is not zero, then it is at least

$$((\Delta/\delta)^{\mathrm{poly}(n)})^{-n^{O(rk^2/\epsilon)}} = \exp\left(-n^{O(k^2 r/\epsilon)} \log^{O(1)}(\Delta/\delta)\right). \tag{16.7}$$

Hence, OPT is at least (16.7) as well.

Plugging $\tau \ll \epsilon \cdot$ (16.7) $\leq \epsilon\,\mathrm{OPT}$ into the algorithm from the previous section with an additional constraint $\|\widehat{U}\widehat{V}\|_F^2 \leq (\Delta/\delta)^{\mathrm{poly}(n)}$, we do binary search to narrow down the range of $[\Lambda^-, \Lambda^+]$ until we reach Λ. During the jth step of binary search, we use Theorem 16.2.1

to check if the following semi-algebraic set is empty or not,

$$S = \{x \in \mathbb{R}^l \mid p(x) \leq \Lambda_j^+ q(x), p(x) \geq \Lambda_j^- q(x), q(x) \neq 0\}$$

where Λ_1^- is initialized to be τ and Λ_1^+ is initialized to be $\text{poly}(n, \Delta)$. We obtain the following theorem.

Theorem 16.5.1. *Given $A, W \in \mathbb{R}^n$, $1 \leq k \leq n$ and $0 < \epsilon, \tau < 0.1$ such that:*

- $\text{rank}(W) = r$*;*
- *all the non-zero entries of A and W are multiples of $\delta > 0$;*
- *all the entries of A and W are at most $\Delta > 0$ in absolute value,*

one can output a number Λ in time $n^{O(k^2 r/\epsilon)} \cdot \log^{O(1)} \frac{\Delta}{\delta\tau}$ such that $\text{OPT} \leq \Lambda \leq (1+\epsilon)\,\text{OPT}$, where

$$\text{OPT} = \min_{\substack{U \in \mathbb{R}^{n\times k}, V \in \mathbb{R}^{k\times n} \\ \|UV\|_F^2 \leq \left(\frac{\Delta}{\delta}\right)^{\text{poly}(n)}}} \|(UV - A) \circ W\|_F^2.$$

16.6 Recovering the Solution Itself

Here we show how to recover an approximate solution, not only the value of OPT.

The idea is to recover the entries of U and V one by one and use the algorithm from the previous section for the corresponding decision problem. We initialize the semi-algebraic set to be

$$S = \{x \in \mathbb{R}^l \mid q(x) \neq 0, p(x) \leq \Lambda q(x)\}$$

We start by recovering the first entry of U. We perform the binary search to localize the entry within an interval of length δ', which takes $\mathrm{poly}(n) \cdot \log\left(\frac{\Delta}{\delta\delta'}\right)$ invocations of the decision algorithm. For each step of binary search, we use 16.2.1 to determine if the following semi-algebraic set S is empty or not,

$$S \cap \{U_{1,1}(x) \geq \widehat{U}^{-}_{1,1}, U_{1,1}(x) \leq \widehat{U}^{+}_{1,1}\}$$

After that, we declare the first entry of U to be any point in this interval. This changes the cost of the solution by at most $\delta' \cdot \left(\frac{\Delta}{\delta}\right)^{\mathrm{poly}(n)}$. Then, we add an equality constraint that fixes the entry of $\widehat{U}$ to this value, and add a new constraint into S permanently, e.g. $S \leftarrow S \cap \{U_{1,1}(x) = \widehat{U}_{1,1}\}$. Next, we repeat the same with the second entry of U and so on.

This allows us to recover a *solution* of cost at most $(1+\epsilon)\,\mathrm{OPT} + \tau$ in time

$$n^{O(k^2 r/\epsilon)} \cdot \log^{O(1)}(\Delta/(\delta\tau)).$$

16.7 Adversarial Matrix Completion

Here we prove the following theorem.

Theorem 16.7.1. *Let $B \in \mathbb{R}^{n \times n}$ be a rank-k matrix with entries that are multiples of $\delta > 0$ bounded by $\Delta > 0$.*

Let $1 \leq r \leq n$ be an integer parameter.

Let C be an $n \times n$ matrix, where in every column there are at most r question marks *and other entries are equal to the corresponding entries of B.*

Then, there is an algorithm that:

- *receives C as an input;*
- *outputs a rank-k matrix that is τ-close to C in Frobenius norm (restricted to the entries that are not replaced with a question mark);*
- *has running time $n^{O(k^2 r)} \cdot \log^{O(1)}\left(\frac{\Delta}{\delta\tau}\right)$.*

A naive attempt is to use the algorithm from the previous section with W equal to 0 for the missing entries and to 1 for the "surviving" entries. Unfortunately, setting W like this does not work, since W may not have small rank.

We fix it by proving the following lemma.

Lemma 16.7.2. *For every set system $z_1, z_2, \ldots, z_n \subseteq [n]$ with $|z_j| \leq r$ there exists a rank-$(2r+1)$ integer matrix W such that, for every $1 \leq i \leq n$, the entry in the i-th column and j-th row,*

$$W_i^j \begin{cases} = 0 & \text{if } j \in z_i \\ > 0 & \text{if } j \in [n] \backslash z_i \end{cases}$$

Moreover, all the entries of W are bounded by $n^{O(r)}$ in absolute value.

Recall that for the i-th column of W, z_i denotes the set of row indices that have zero entry. For the i-th column of W, define a polynomial $p_i(x)$ such that

$$p_i(x) = \prod_{\ell \in z_i} (x-j)^2 = \sum_{\ell=1}^{2|z_i|+1} a_{i\ell} x^{\ell-1}$$

where all $a_{i\ell}$ are integers. Since $2|z_i|+1$ is at most $2r+1$, we can also think of $p_i(x)$ as a degree $2r$ polynomial,

$$p_i(x) = \sum_{\ell=1}^{2r+1} a_{i\ell} x^{\ell-1}$$

Then, we can use $a_{i\ell}$ to create a basis $T \in \mathbb{R}^{n \times (2r+1)}$ which has rank at most $2r+1$. Let T_ℓ^i denote the entry of the i-th row and ℓ-th column, then set $T_\ell^i = a_{i\ell}, \forall i \in [n], \forall \ell \in [2r+1]$. Let T_ℓ denote the ℓ-th column of matrix T, $\forall \ell \in [2r+1]$. To guarantee a linear combination of those $2r+1$ columns always outputs a nonnegative vector, we can just choose coefficients $1, x, x^2, \cdots, x^{2r}$. Let S denote a set of all possible vectors formed by the following linear combination,

$$T(x) = \sum_{\ell=1}^{2r+1} x^{\ell-1} T_\ell \in \mathbb{R}^{n \times 1}$$

Moreover, we have

$$\begin{aligned} T(x) &= \sum_{\ell=1}^{2r+1} x^{\ell-1} T_\ell = \sum_{\ell=1}^{2r+1} x^{\ell-1} \begin{bmatrix} a_{1\ell} & a_{2\ell} & \cdots & a_{n\ell} \end{bmatrix}^\top \\ &= \begin{bmatrix} \sum_{\ell=1}^{2r+1} x^{\ell-1} a_{1\ell} & \sum_{\ell=1}^{2r+1} x^{\ell-1} a_{2\ell} & \cdots & \sum_{\ell=1}^{2r+1} x^{\ell-1} a_{n\ell} \end{bmatrix}^\top \\ &= \begin{bmatrix} p_1(x) & p_2(x) & \cdots & p_n(x) \end{bmatrix}^\top, \end{aligned}$$

where the second equality follows by the definition of T_ℓ and the last equality follows by the definition of $p_i(x)$. Let $T^j(x)$ denote the jth entry of column vector $T(x) \in \mathbb{R}^{n \times 1}$. For any

column vector W_i, we assign $T(i)$ to it,

$$W_i \leftarrow T(i)$$

which has the following property: for every $1 \leq i \leq n$, the entry at the i-th column and $j-$th row satisfies

$$W_i^j = \begin{cases} T^j(i) = p_i(j) = 0 & \text{if } j \in z_i, \\ T^j(i) = p_i(j) > 0 & \text{if } j \in [n] \backslash z_i. \end{cases}$$

Note that for any column vector W_i, we know that $W_i \in S$ and $\text{rank}(S) = 2r + 1$. Thus, $\text{rank}(W) = 2r + 1$.

16.8 Few distinct columns

In this section we show how to improve the running time from $n^{\mathrm{poly}(k,r,1/\epsilon)}$ to $\mathrm{poly}(n) \cdot 2^{\mathrm{poly}(k,r,1/\epsilon)}$ under the following assumptions: (1) $\Delta = \mathrm{poly}(n)\delta$; and (2) $\|UV\|_F^2 \leq (\Delta/\delta)^{n^\gamma}$, for an arbitrarily small constant $\gamma > 0$. In Section 16.8.1, as a warmup we assume that W has r *distinct columns* and r *distinct rows*, while in Section 16.8.2 and 16.8.3 we give our main result assuming only that W has r *distinct columns*.

A crucial observation is that the term $n^{\mathrm{poly}(k,r,1/\epsilon)}$ shows up in the "rank-r" algorithm due to the fact that the degree of polynomials we optimize is $\Omega(n)$. The reason for this is that entries of $\widehat{U}$ and $\widehat{V}$ are rational functions with $\Omega(n)$ potentially different denominators. When we combine them in a single rational function that corresponds to $\|(\widehat{U}\widehat{V} - A) \circ W\|_F^2$, we get a denominator of degree $\Omega(n)$.

16.8.1 Few Distinct Rows and Columns

In this subsection, as a warmup we assume that W has r distinct rows *and* r distinct columns. Then, we get rid of the dependence on n in the degree. Indeed, now we have only $2r$ distinct denominators (w.l.o.g., assume the first r columns are distinct and the first r rows are distinct),

$$\begin{aligned} g_i(x) &= \det((SD_{W_i}U)_{P_i}^{\top}(SD_{W_i}U)_{P_i}), \forall i \in [r] \\ f_i(x) &= \det((VD_{W^i}S)^{Q_i}((VD_{W^i}S)^{Q_i}))^{\top}), \forall i \in [r] \end{aligned}$$

where P_i and Q_i are maximal linearly independent subsets. Then we can write down the following optimization problem,

$$\begin{aligned} \min_{x \in \mathbb{R}^l} \quad & p(x)/q(x) \\ s.t. \quad & g_i^2(x) \neq 0, f_i^2(x) \neq 0, \forall i \in [r], \\ & q(x) = \prod_{i=1}^{r} g_i^2(x) f_i^2(x) \end{aligned}$$

where $q(x)$ has degree $O(rk)$, the maximum coefficient in absolute value is $(\Delta/\delta)^{O(rkn^{\gamma})}$, and the number of variables $O(rk^2/\epsilon)$. Using the same argument as in the rank-r case and applying Theorem 16.2.2, we can achieve the following minimum nonzero cost: $(\Delta/\delta)^{-n^{\gamma}2^{\widetilde{O}(rk^2/\epsilon)}}$. Now, using the approach described in section 16.6, we can find the solution in time

$$(\mathrm{nnz}(A) + \mathrm{nnz}(W))n^{\gamma} + n2^{\widetilde{O}(rk^2/\epsilon)} \log^{O(1)}(\Delta/(\delta\tau))$$

within a multiplicative factor of $1+\epsilon$ and additive factor of τ.

One can adjust the lower bound on OPT accordingly, and conclude that an algorithm for approximating OPT within a multiplicative factor of $1+\epsilon$ can be done in time

$$(\mathrm{nnz}(A) + \mathrm{nnz}(W))n^{\gamma} + n2^{\widetilde{O}(rk^2/\epsilon)} \log^{O(1)}(\Delta/\delta)$$

where

$$\mathrm{OPT} = \min_{\substack{U,V \\ \|UV\|_F^2 \le (\Delta/\delta)^{n^\gamma}}} \|(UV - A) \circ W\|_F^2.$$

16.8.2 Few Distinct Columns When OPT $= 0$

This section, we explain how to find the solution to the weighted low rank approximation problem when W has at most r distinct columns and OPT $= 0$.

The key observation is that for any matrix A and $W \in \mathbb{R}_{\geq 0}^{n \times n}$, if there exists a solution of an $n \times k$ matrix U and a $k \times n$ matrix V such that,

$$\|W \circ (UV - A)\|_F^2 = \text{OPT},$$

then there exists another matrix $W' \in \{0,1\}^{n \times n}$ such that

$$\|W' \circ (UV - A)\|_F^2 = \text{OPT}$$

where $W'_{i,j} = 0$ if $W_{i,j} = 0$ and $W'_{i,j} = 1$ if $W_{i,j} > 0$.

The above observation states that modifying the weight matrix to be Boolean does not change the optimal cost. Since W has r distinct columns, now that it is Boolean it has at most 2^r distinct rows. Indeed, each row of W is completely determined after fixing its values on the r distinct columns, and there are only 2^r possibly fixings. W.l.o.g. we assume that the first r columns are distinct. Instead of having at most $2r$ distinct denominators as in Section 16.8.1, we have at most $r + 2^r$ distinct denominators. We create l variables for $\{SD_{W_1}U, \cdots, SD_{W_r}U\}$. Then we can write down $\widehat{V}$ in the following way,

$$\widehat{V}_j = (SD_{W_j}U)^\dagger \cdot SD_{W_j}A_j = \left(((SD_{W_j}U)_{P_i})^\top \cdot (SD_{W_j}U)_{P_i}\right)^{-1} \cdot ((SD_{W_j}U)_{P_i})^\top SD_{W_j}A_j$$

where P_i denotes a subset of rows. For all D_{W_j}s in the group Z_i, they share the same P_i, where for any $j \in Z_i$, $D_{W_j} = D_{W_i}$.

Thus to express $\widehat{V}$, there are only r distinct denominators $g_i(x)$ which are the determinants of $((SD_{W_j}U)_{P_i})^\top \cdot (SD_{W_j}U)_{P_i}$. In order to remove these denominators, we can create a new variable x_{l+i} and add a new equality constaint $g_i(x)x_{l+i} - 1 = 0$. Therefore, we do not have any denominators in $\widehat{V}$.

W.l.o.g., we assume that the first 2^r rows of W are distinct. Using $\widehat{V}$ we can write down $\widehat{U}$ in the following way,

$$\begin{aligned}\widehat{U}^j &= A^j D_{W^j}(\widehat{V}D_{W^j})^\dagger \\ &= A^j D_{W^j}((\widehat{V}D_{W^j})^{Q_i})^\top \left((\widehat{V}D_{W^j})^{Q_i}((\widehat{V}D_{W^j})^{Q_i})^\top\right)^{-1}\end{aligned}$$

where Q_i denotes a subset of columns, where all D_{W^j}s in the group Z_i' can share the same Q_i, and for any $j \in Z_i'$, $D_{W^j} = D_{W^i}$.

Thus, to express $\widehat{U}$, there are only 2^r distinct denominators $f_j(x)$ which are the determinants of the matrices $(\widehat{V}D_{W^j})^{Q_i}((\widehat{V}D_{W^j})^{Q_i})^\top$.

Finally, we can use a small number of variables to represent all the entries of $\widehat{U}$ and $\widehat{V}$. It allows us to write the following optmization problem,

$$\begin{aligned}\min_{x\in\mathbb{R}^{l+r}} \quad & p(x)/q(x) \\ s.t. \quad & g_i(x)x_{l+i} - 1 = 0, \forall i \in [r] \\ & f_j^2(x) \neq 0, \forall j \in [2^r] \\ & q(x) = \prod_{j=1}^{2^r} f_j^2(x)\end{aligned}$$

where $q(x)$ has degree $O(2^r k^2)$, maximum coefficients bounded in absolute value by $(\Delta/\delta)^{O(2^r k n^\gamma)}$, $l = O(rk^2/\epsilon)$ and the number of variables $O(rk^2/\epsilon)$. Using the same argument as for the

rank-r case and applying Theorem 16.2.2, we have the following minimum nonzero cost: $(\Delta/\delta)^{-n^\gamma 2^{\widetilde{O}(r^2k^2/\epsilon)}}$.

Using the approach described in section 16.6, we can find the solution achieving zero cost in time

$$(\mathrm{nnz}(A) + \mathrm{nnz}(W))n^\gamma + n2^{\widetilde{O}(r^2k^2/\epsilon)} \log^{O(1)}(\Delta/\delta)$$

16.8.3 Few Distinct Columns When OPT $\neq 0$

Lower Bound Let U, V denote the optimal solution A, W, which gives nonzero cost. We can modify W to a new matrix W' in the following sense, $W'_{i,j} = \delta$ if $W_{i,j} \neq 0$ and $W'_{i,j} = 0$ if $W_{i,j} = 0$. Then we know that

$$\|W' \circ (UV - A)\|_F^2 \leq \|W \circ (UV - A)\|_F^2 \neq 0$$

Note that if problem A, W' has a zero cost solution, then problem A, W also has a zero cost solution, which contradicts our assumption in this section. Thus problem A, W' does not have a zero cost solution. It follows from previous sections that the minimum nonzero cost of $\min_{U,V} \|W' \circ (UV - A)\|_F^2$ is at least $(\Delta/\delta)^{-n^\gamma 2^{\tilde{O}(r^2k^2/\epsilon)}}$. Let U', V' denote the optimal solution of problem A, W'. Thus we have

$$\|W \circ (UV - A)\|_F^2 \geq \|W' \circ (UV - A)\|_F^2 \geq \|W' \circ (U'V' - A)\|_F^2$$

which is at least $(\Delta/\delta)^{-n^\gamma 2^{\tilde{O}(r^2k^2/\epsilon)}}$.

Algorithm For notational convenience, let $\delta = 1$. Then each entry of the input weight matrix W' is in $\{0, 1, 2, \cdots, \text{poly}(n)\}$. For each entry $W'_{i,j}$, we round it to the smallest $(1+\epsilon)^x$ such that $W'_{i,j} \leq (1+\epsilon)^x$ where x is an integer. Because W' is bounded, the total number of choices for the power x is $O(\log(n)/\epsilon)$. Define W to be the matrix after rounding. Define OPT to be $\min_{U,V} \|W' \circ (UV - A)\|_F^2$. Then W has the following properties

1. W has r distinct columns,
2. W has $R := (\log(n)/\epsilon)^{O(r)}$ distinct rows,
3. $\text{OPT} \leq \min_{U,V} \|W \circ (UV - A)\|_F^2 \leq (1+\epsilon)^2 \,\text{OPT}$.

We prove the above three properties one by one.

The rounding is a deterministic procedure: if two values are the same in W', then they are the same in W. Hence, Property 1 holds.

To prove Property 2, take the r distinct columns $i_1, ..., i_r$. Then every other column can be labeled j in $\{i_1, ..., i_r\}$. If you fix the values on entries $i_1, ..., i_r$ in a row, this fixes the values on every other column. So the number of distinct rows is the number of fixings to the values on $i_1, ..., i_r$. Each entry has $\log_{1+\epsilon} \text{poly}(n) = O((\log n)/\epsilon)$ possibilities, so there are $O((\log n)/\epsilon)^r$ distinct rows.

Because of the rounding procedure, each $W'_{i,j}$ satisifies that $W'_{i,j} \leq W_{i,j} \leq (1+\epsilon)W'_{i,j}$, which implies Property 3.

We use the same approach as in Section 16.8.2 to create variables, write down the polynomial systems and add not equal constraints. Instead of having $r + 2^r$ distinct denominators, we have $r + R$, where $R = O((\log n)/\epsilon)^r$. We create $l = O(rk^2/\epsilon)$ variables for $\{SD_{W_1}U, \cdots, SD_{W_r}U\}$, then we can write down $\widehat{V}$ with r distinct denominators $g_i(x)$. Each $g_i(x)$ is non-zero in an optimal solution using the perturbation argument in Section 16.4. We create new variables x_{i+l} to remove the denominators $g_i(x)$. Then the entries of $\widehat{V}$ are polynomials as opposed to rational functions. Using $\widehat{V}$ we can express $\widehat{U}$ with $R = (\log(n)/\epsilon)^{O(r)}$ distinct denominators $f_i(x)$, which are also non-zero by using the perturbation argument in 16.4, and using that W has at most this number of distinct rows. Finally we can write the

following optimization problem,

$$\begin{aligned} \min_{x \in \mathbb{R}^{l+r}} \quad & p(x)/q(x) \\ s.t. \quad & g_i(x)x_{l+i} - 1 = 0, \forall i \in [r] \\ & f_j^2(x) \neq 0, \forall j \in [R] \\ & q(x) = \prod_{j=1}^{R} f_j^2(x) \end{aligned}$$

We then determine if there exists a solution to the above semi-algebraic set in time $(k^2R)^{O(rk^2/\epsilon)} = (\log(n)/\epsilon)^{O(r^2k^2/\epsilon)}$. Combining the binary search explained in section 16.5 and 16.6 with the lower bound we obtained, we can find the solution for the original problem in time

$$(\text{nnz}(A) + \text{nnz}(W))n^{\gamma} + n2^{\widetilde{O}(r^2k^2/\epsilon)} \log^{O(1)}(\Delta/\delta).$$

Note that there is no $\log\log n$ in the exponent $2^{\widetilde{O}(r^2k^2/\epsilon)}$ since either $r^2k^2/\epsilon = o(\log n/\log\log n)$, in which case this term is dominated by $n^{1+\gamma}$, or $\log(r^2k^2/\epsilon) = \Omega(\log\log n)$.

16.9 Hardness

The Maximum Edge Biclique problem [AMS11] is defined as:

Input: An n by n bipartite graph G.

Output: A k_1 by k_2 complete bipartite subgraph of G.

Objective Function: Maximize $k_1 \cdot k_2$.

We use the Maximum Edge Biclique problem under the R4SAT assumption in [GL04], which extends the previous work done by Feige [Fei02] under the R3SAT assumption. That hardness result [GL04] shows under the R4SAT assumption there exist two constants $\epsilon_1 > \epsilon_2 > 0$ such that no efficient algorithm is able to distinguish between bipartite graphs $G(U, V, E)$ with $|U| = |V| = n$ which have a clique of size $\geq (n/16)^2(1+\epsilon_1)$ and those in which all bipartite cliques are of size $\leq (n/16)^2(1+\epsilon_2)$. Using the reduction of [GL04], one can show there exists a constant c such that for any instance of R4SAT with $\widetilde{n}$ variables and $c\widetilde{n}$ clauses, the corresponding bipartite graph G created in [GL04] has at least tn^2 edges with large probability, for a constant t, e.g. $t = 9/10$.

To construct a weighted low-rank appoximation problem from a given bipartite graph, for a given bipartite graph $G(U, V, E)$, we generate the matrix A and W as in [GG11]: $A_{ij} = 1$ if edge $(U_i, V_j) \in E$, $A_{ij} = 0$ if edge $(U_i, V_j) \notin E$. $W_{ij} = 1$ if edge $(U_i, V_j) \in E$, $W_{ij} = \text{poly}(n)$ if edge $(U_i, V_j) \notin E$. One can then show if there exists a biclique in G such the number of remaining edges is at most $tn^2 - (n/16)^2(1+\epsilon_1)$, then the solution to $\min \|W \circ \widehat{A} - W \circ A\|_F^2$ has cost at most $tn^2 - (n/16)^2(1+\epsilon_1)$. On the other hand, if there does not exist a biclique that has more than $(n/16)^2(1+\epsilon_2)$ edges, which leads to the number of remaining edges being at least $tn^2 - (n/16)^2(1+\epsilon_2)$, then any solution to $\min \|W \circ \widehat{A} - W \circ A\|_F^2$ has cost at

least $tn^2 - (n/16)^2(1 + \epsilon_2)$.

16.10 Bicriteria Approximation

16.10.1 Rank r and Binary Weights W

This section, we explain how to get a polynomial time bicriteria algorithm when weighted matrix is boolean.

Lemma 16.10.1. *Given matrix $A \in \mathbb{R}^{n\times n}$ and rank-r weighted matrix $W \in \mathbb{R}^{n\times n}_{\geq 0}$. We can find some rank-$rk$ matrix $B \in \mathbb{R}^{n\times n}$ such that*

$$\|B - W \circ A\|_F^2 \leq (1+\epsilon) \min_{\mathrm{rank}(B')=rk} \|B' - W \circ A\|_F^2.$$

in $O(\mathrm{nnz}(A) + \mathrm{nnz}(W) + n \cdot \mathrm{poly}(rk/\epsilon))$ time.

Proof. It directly follows by the input sparsity low-rank approximation result in [CW13]. □

Lemma 16.10.2. *For any matrix $W, A \in \mathbb{R}^{n\times n}$ and $\mathrm{rank}(W) = r$, we have*

$$\min_{\mathrm{rank}(B')=rk} \|B' - W \circ A\|_F^2 \leq \min_{\mathrm{rank}(C')=k} \|W \circ C' - W \circ A\|_F^2.$$

Proof. For any rank-k matrix C', matrix $W \circ C'$ has rank at most rk. Then, for any rank-k matrix C', there must exists a rank-rk matrix B' such that

$$\|B' - W \circ A\|_F^2 \leq \|W \circ C' - W \circ A\|_F^2$$

□

Theorem 16.10.3. *Given two matrices $W \in \mathbb{F}_2^{n\times n}$ and $A \in \mathbb{R}^{n\times n}$, where $\mathrm{rank}(W) = r$. Define $A_k^* = \arg\min_{\mathrm{rank}(A')=k} \|A' - A\|_W$. We can find a matrix B that has rank rk and such that*

$$\|B - A\|_W^2 \leq (1+\epsilon)\|A_k^* - A\|_W^2$$

in $O(\mathrm{nnz}(A) + \mathrm{nnz}(W) + n \cdot \mathrm{poly}(kr/\epsilon))$ time.

Proof. Let matrix B denote the output of Lemma 16.10.1, let Ω denote the support set of weighted matrix W, $\Omega = \{(i,j)|W_{i,j} > 0\}$.

$$
\begin{aligned}
& \|B - A\|_W^2 \\
= \; & \|W \circ B - W \circ A\|_F^2 \\
= \; & \sum_{(i,j)\in\Omega} W_{ij}^2 (B_{ij} - A_{ij})^2 + \sum_{(i,j)\in[n]^2\setminus\Omega} W_{ij}^2 (B_{ij} - A_{ij})^2 \\
= \; & \sum_{(i,j)\in\Omega} 1 \cdot (B_{ij} - A_{ij})^2 + \sum_{(i,j)\in[n]^2\setminus\Omega} 0 \cdot (B_{ij} - A_{ij})^2 \\
\leq \; & \sum_{(i,j)\in\Omega} 1 \cdot (B_{ij} - A_{ij})^2 + \sum_{(i,j)\in[n]^2\setminus\Omega} 1 \cdot (B_{ij})^2 \\
= \; & \|B - W \circ A\|_F^2 \\
\leq \; & (1+\epsilon) \min_{\mathrm{rank}(B')=rk} \|B' - W \circ A\|_F^2 \text{ by Lemma 16.10.1} \\
\leq \; & (1+\epsilon) \min_{\mathrm{rank}(C')=k} \|W \circ C' - W \circ A\|_F^2 \text{ by Lemma 16.10.2}
\end{aligned}
$$

□

16.10.2 Modulo p Integer Weights W

Theorem 16.10.4. *Given two matrices $W \in \mathbb{F}_p^{n\times n}$ and $A \in \mathbb{R}^{n\times n}$, where $\mathrm{rank}(W) = r$. Define $A_k^* = \arg\min_{\mathrm{rank}(A')=k} \|A' - A\|_W$. We can find a matrix A' that has rank $p^r \cdot k$ and satisfies the following*

$$\|A' - A\|_W^2 \leq (1+\epsilon)\|A_k^* - A\|_W^2$$

in $O(\mathrm{nnz}(A) + \mathrm{nnz}(W) + p^r \cdot n \cdot \mathrm{poly}(k/\epsilon))$ time.

Proof. Since each entry of weight matrix W is chosen from $\{0, 1, \cdots, p-1\}$ and has rank r. Thus, for any given r linearly independent basis, weight matrix W has at most p^r distinct columns. For any column vector A_j in A_{Z_i}, its weighted column vector W_j is equal to W_i. Then we can reduce the original problem to

$$\sum_{i=1}^{g} \min_{\mathrm{rank}(A_i')=k} \|D_{W_i} A_i' - D_{W_i} A_{Z_i}\|_F^2 \tag{16.8}$$

where $g \leq p^r$, $A_{Z_i} \in \mathbb{R}^{n\times|Z_i|}$, $\sum_{i=1}^{g} |Z_i| = n$. Since these g problems are independent, we can just solve them in parallel. Let Ω_i denote the support set of rows of D_{W_i}, ignoring the zero rows of $D_{W_i} A_{Z_i}$, we can have a new matrix $B_i \in \mathbb{R}^{|\Omega_i|\times|Z_i|}$. Consider the following low rank approximation problem

$$\min_{\mathrm{rank}(B_i')=k} \|B_i' - B_i\|_F^2$$

By the input sparsity low-rank approximation result in [CW13], in time $O(\mathrm{nnz}(B)+\max(|\Omega_i|, |Z_i|)\cdot \mathrm{poly}(k/\epsilon))$, we can find some rank-$k$ matrix $B_i' \in \mathbb{R}^{|\Omega_i|\times|Z_i|}$ such that

$$\|B_i' - B\|_F^2 \leq (1+\epsilon) \min_{\mathrm{rank}(B_i^*)=k} \|B_i^* - B_i\|_F^2$$

Since D_{W_i} is diagonal matrix, we can recover a rank-k matrix $A_i' \in \mathbb{R}^{n\times|Z_i|}$ by using matrix B_i' and D_{W_i} in the follwoing way, for jth row of $n \times |Z_i|$ matrix A_i', if jth entry of W_i is zero,

then make jth row matrix A'_i zero everywhere; otherwise, copy the corresponding row from B'_i and rescale it by 1 / (jth entry of W_i). Then the $n \times |Z_i|$ matrix A'_i we obtained has the following property

$$\begin{aligned}
& \|D_{W_i} A'_i - D_{W_i} A_{Z_i}\|_F^2 \\
= \; & \|B'_i - B_i\|_F^2 \\
\leq \; & (1+\epsilon) \min_{\mathrm{rank}(B_i^*)=k} \|B_i^* - B_i\|_F^2 \\
\leq \; & (1+\epsilon) \min_{\mathrm{rank}(A_i^*)=k} \|D_{W_i} A_i^* - D_{W_i} A_{Z_i}\|_F^2 \text{ by Lemma 16.10.2}
\end{aligned}$$

Since all the g subproblems are independent, then we can solve them separately and merge all the A'_i to a matrix A' that has rank-$p^r k$. The entire procedure takes $O(\mathrm{nnz}(A)+\mathrm{nnz}(W)+ p^r \cdot n \cdot \mathrm{poly}(k/\epsilon))$. □

16.10.3 r Distinct Columns Weights W

Theorem 16.10.5. *Given two matrices $W \in \mathbb{R}^{n\times n}_{\geq 0}$ and $A \in \mathbb{R}^{n\times n}$, W has r distinct columns. Define $A^*_k = \underset{\mathrm{rank}(A')=k}{\arg\min} \|A' - A\|_W$. We can find a matrix $A' \in \mathbb{R}^{n\times n}$ that has rank $r \cdot k$ and satisfies the following*

$$\|A' - A\|^2_W \leq (1+\epsilon)\|A^*_k - A\|^2_W$$

in $O(\mathrm{nnz}(A) + \mathrm{nnz}(W) + nr \cdot \mathrm{poly}(k/\epsilon))$ time.

Proof. In previous proof, we have p^r distinct columns in W. But now, we only have r distinct columns in W. We still use the same algorithm to solve each subproblem. Since the total numbe of subproblems is r, thus the running time doesn't have exponential dependence for r any more. □

16.11 Weighted Nonnegative Matrix Factorization Problem

Arora, Ge, Moitra and Kannan [AGKM12] provided the first provably result for nonnegative matrix factorization. Later, Moitra [Mad13] improved it to almost optimal. This section explains the details of getting an algorithm for weighted nonnegative matrix factorization. We combine our new technique "guess the sketch" with previous work done by Moitra [Mad13] to write down a low degree polynomial system with some number of inequalities in terms of small number of variables. Next, we can use polynomial solver [Ren92a, Ren92b, Ren92c] to determine if there exists a nonnegative factorization of target matrix A under weighted matrix W. The slover takes $(md)^{O(v)}$ time to run, if m is the number of inequality constraints, d is the maximum degree of polynomials and v is the number of variables. We first show an algorithm that needs try $2^{k^{O(r)}}$ guesses, the running time of which has double exponential dependence in r. Then we expalin how to write down a slightly bigger polynomial systems without $2^{k^{O(r)}}$ guesses, which reduces the double exponential of r down to single exponential dependence.

Notations Let $W \in \mathbb{R}_{\geq 0}^{n \times n}$ denote the nonnegative weight matrix, let A denote the target $n \times n$ matrix, our goal is to factorize A into two low-dimensional nonnegative matrices $B \in \mathbb{R}_{\geq 0}^{n \times k}$ and $C \in \mathbb{R}_{\geq 0}^{k \times n}$. Let A_i denote the ith column of matrix A and A^j denote the jth row of matrix A. Let $A \circ W$ denote the entry-wise product of two matrices A and W, and let $u \circ v$ denote the entry-wise product of two vectors A and W. Let D_{lev} denote the leverage score matrix. Let D_{W_i} denote the diagonal matrix, where each entry on diagonal is from the vector W_i, similarly for D_{W^j}.

Definition 16.11.1. The nonnegative weighted rank $\mathrm{rank}^+(A)$ is defined to be the smallest

k such that A can be written as

$$A_{ij} = \begin{cases} B^i C_j = (BC)_{ij} & \text{if } W_{ij} > 0 \\ \text{arbitrary} & \text{otherwise} \end{cases} \tag{16.9}$$

where B and C are nonnegative and have dimension $n \times k$ and $k \times n$ respectively, W has rank r.

We use the notations $\text{aff}(A)$ and lexicographic ordering from [Mad13] directly, and extend the stable definition and addmissible definition into weighted version. The standard lexicographic ordering is usually for the subsets that have the same size, additionally, we assume the smaller size subset has early order than larger size subset.

Definition 16.11.2 (Affine Hull)**.** Given $n \times n$ matrix A and let $S \subseteq [n]$. Define

$$\text{aff}(A_S) = \{\sum_{i \in S} \alpha_i A_i | \forall i, \alpha_i \geq 0\}.$$

Definition 16.11.3 (Weighted Stable)**.** $\forall i \in [n]$, let $S_i \subseteq [k]$ be the lexicographically first admissible subset (of columns of $B \in \mathbb{R}^{n \times k}$) for A_i. $\forall j \in [n]$, , let $T^j \subseteq [k]$ be the lexicographically first admissible subset (of rows of $C \in \mathbb{R}^{k \times n}$) for A^j. We call $W \circ A = W \circ (BC)$ **weighted stable** if:

1. $\forall i \in [n]$, C_i is supported in $S_i \subseteq [k]$
2. $\forall j \in [n]$, B^j is supported in $T^j \subseteq [k]$

Definition 16.11.4 (Weighted Admissible)**.** Given a weighted vector $w \in \mathbb{R}^n_{\geq 0}$ and a vector $v \in \mathbb{R}^n$, let S denote a subset of columns of A, S is **weighted admissible** if there exists some $u \in \text{aff}(A_S)$ such that $w \circ v = w \circ u$.

Another interpretation of weighted admissible is thinking of we have $*$ entry in vector v instead of thinking there is antoher weighted vector. Given a vector $v \in \{\mathbb{R}, *\}^n$, let $q \subseteq [n]$ denote the subset of coordinates that contain $*$, we will call a subset S of columns of A admissible if there exists some $u \in \text{aff}(A_S)$ such that, $\forall i \notin q$, $v_i = u_i$.

Lemma 16.11.1. *Given matrices $A \in \mathbb{R}^{n\times n}$ and $W \in \mathbb{R}^{n\times n}_{\geq 0}$. If there exist some nonnegative inner-dimension k matrices B and C such that $W \circ A = W \circ (BC)$, then there exist $\widetilde{B}$ and $\widetilde{C}$ such that:*

1. *$W \circ A = W \circ (\widetilde{B}\widetilde{C})$, $\widetilde{B}$ and $\widetilde{C}$ are nonnegative and have inner-dimension k;*
2. *$W \circ A = W \circ (\widetilde{B}\widetilde{C})$ is weighted stable.*

Proof. In this proof, we modify the matrix A according to the zero position of matrix W, $A_{ij} = *$ if $W_{ij} = 0$. At the beginning, we have some B and C such that A and BC are entry-wise consistent except the $*$ positions. Then each update phase will alternately switching between B-updating phase and C-updating phase. In the C-updating phase, for each column vector $A_i \in \{\mathbb{R}, *\}^{n\times 1}$, let S_i be the lexicographically first subset of columns of B that is admissible for A_i. If $S_i \subseteq [k]$ is lexicographically (strictly) earlier than the support of column vector $C_i \in \mathbb{R}^{k\times 1}$, then we can find a column vector $\widetilde{C}_i \in \mathbb{R}^{k\times 1}_{\geq 0}$, $\text{supp}(\widetilde{C}_i) \subseteq S_i$ satisfies that A_i is entry-wise consistent with $B\widetilde{C}_i$ except $*$ position. Otherwise, we choose $\widetilde{C}_i = C_i$. Since, we don't change anything for matrix B and $C_{[n]\setminus i}$, then A is still entry-wise consistent with $B\widetilde{C}$ except the $*$ positions. The last step of this C-updating phase is updating C by $\widetilde{C}$. Similarly, we can define the B-updating phase. During the overall updating, (1) the lexicographically order of support monotonically decrease, if B^j or C_i got updated. (2) the

matrices B and C always satisfy that A is entry-wise consistent with BC except $*$ position, in other words. Thus, the overall updating phases will terminate with $W \circ A = W \circ (BC)$. $\square$

16.11.1 Combining with Our Techniques

Here we explain how to adapt our technique for solving the exact case to weighted non-negative matrix factorization. (Note that there exists a low-rank matrix A' to make the optimal cost 0, but it doesn't mean A is low-rank.) We first do leverage score sampling on the LHS of the problem, there exists some diagonal matrix D_{lev} respect to $\text{span}(D_{W_1}B, D_{W_2}B, \cdots, D_{W_n}B)$ such that

$$\min_{B,C} \sum_{i=1}^{n} \|D_{lev}D_{W_i}BC_i\|_2^2 \leq (1+\epsilon)\min_{B,C} \sum_{i=1}^{n} \|D_{W_i}BC_i\|_2^2$$

If we know matrix B, then we can compute the leverage sampling matrix deterministically and exactly. Here, we don't know the B, thus, we have to either guess those nonzero probabilities or creat variables for nonzero probabilities. Let t denote the number of nonzero probabilities in leverage score sampling matrix, then

Claim 16.11.2. *There are $O(n^t)$ leverage score sampling matrix we need to guess.*

Since D_{lev} is the leverage score sampling matrix that has $t = (kr/\epsilon^2)$ nonzero probability on diagonal. Thus, the nonzero position of D_{lev} will determine the set of rows of B. We create variables for the rows of B which are chosen by D_{lev}. There are t such rows, thus we need tr variables in total. To write down $D_{lev}D_{W_i}B$ by a small number of variables, we also need to create t variables for the nonzero probability of D_{lev}. Thus,

Claim 16.11.3. *$\forall i \in [n]$, $t \times k$ matrix $D_{lev}D_{W_i}B$ can expressed as $O(tr)$ variables, where each entry of that matrix is single monomial that has degree at most 2.*

In order to write down C_i by these $O(tr)$ small number of variables, we need to group C_i by looking at the zero-pattern of $D_{lev}D_{W_i}$. Let g denote the number of different

zero-patterns, let z_i denote the set of indices belong to the same group,

$$\min_{B,C} \sum_{i=1}^{n} \|D_{lev} D_{W_i} B C_i\|_2^2 = \min_{B,C} \sum_{\ell=1}^{g} \sum_{i \in z_\ell} \|D_{lev} D_{W_i} B C_i\|_2^2$$

By the similar result from previous section, there are only $\binom{t}{r} = k^r$ different zero-patterns. Let $D_{lev} D_{W'_\ell}$ denote the zero-pattern of group $z_\ell, \forall \ell \in [g]$. Let $\text{rank}(D_{lev} D_{W'_\ell}) = s_\ell$. For each group, we first guess s_ℓ which is the size of maximal linearly independent of rows of $D_{lev} D_{W'_\ell} B$, it has k possibilities. Second we need to guess the maximal linearly independent set of rows of $D_{lev} D_{W'_\ell} B$, let U_ℓ denote the subset of rows of $D_{lev} D_{W'_\ell} B$. Then we define $S_1, S_2, \cdots, S_p$ to be the full list of sets of s_ℓ linearly independent columns of $D_{lev} D_{W'_\ell} B$ in lexicographic order, where $p \leq \binom{k}{s_\ell} \leq 2^k$. By the definition of U_ℓ and S_i, we have the property that each $s_\ell \times s_\ell$ submatrix $(D_{lev} D_{W'_\ell} B)_{S_i}^{U_\ell}$ is full rank and invertible.

Claim 16.11.4. *If $D_{lev} D_{W_i}$ and $D_{lev} D_{W_j}$ belong to the same zero pattern group z_ℓ, then they must have the same rank. There are k possibilities for the rank of $D_{lev} D_{W'_\ell} B, \forall \ell \in [k^r]$. In total, there are $O(k^{k^r})$ guesses need to try.*

Claim 16.11.5. *If $D_{lev} D_{W_{i_1}}$ and $D_{lev} D_{W_{i_2}}$ belong to the same zero pattern group z_ℓ, then they must have the same U_ℓ. There are $r^{O(k)}$ possibilities for row subset U_ℓ of $D_{lev} D_{W'_\ell} B, \forall \ell \in [k^r]$. In total, there are $2^{k^{O(r)}}$ guesses need to try. Fix a guess of U_ℓ, there are at most 2^k possibilities for column subset S_j of $D_{lev} D_{W'_\ell} B, \forall \ell \in [k^r]$. In total, there are $2^{k^{O(r)}}$ guesses need to try.*

16.11.2 Few Entries Determine B and C

We extend Lemma 2.7 in [Mad13] to the the following weighted version.

Lemma 16.11.6. *For each column vector $A_i \in \mathbb{R}^{n\times 1}$, let $D_{lev}D_{W_i}$ belong to the group z_ℓ, then among the set of vectors*

$$\mathcal{S} = \left\{ \widehat{B}_1^{i,\ell}(D_{lev}D_{W_i}A_i)^{U_\ell}, \widehat{B}_2^{i,\ell}(D_{lev}D_{W_i}A_i)^{U_\ell}, \cdots, \widehat{B}_p^{i,\ell}(D_{lev}D_{W_i}A_i)^{U_\ell} \right\}$$

column vector $C_i \in \mathbb{R}^{k\times 1}$ is the unique vector with lexicographically minimal support among all nonnegative vectors in the above set. Note that $(D_{lev}D_{W_i}B)^{U_\ell}_{S_j} \in \mathbb{R}^{s_\ell \times s_\ell}$, $((D_{lev}D_{W_i}B)^{U_\ell}_{S_j})^\dagger \in \mathbb{R}^{s_\ell\times s_\ell}$, $\widehat{B}_j^{i,\ell} \in \mathbb{R}^{k\times s_\ell}, \forall j \in [p]$,

$$\widehat{B}_j^{i,\ell} = \underbrace{P_{\pi:[s_\ell]\to[k]}}_{k\times s_\ell} \cdot ((D_{lev}D_{W_i}B)^{U_\ell}_{S_j})^\dagger, \forall i \in [n]$$

$P_{\pi:[s]\to[k]}$ denotes a permutation matrix that has exact one 1 in each column, and there are s out of k rows has exact one 1, everywhere else are all 0.

Proof. We already shown that the weighted stable C_i belongs to the set $\mathcal{S}$. Thus, there must exist some $j^* \in [p]$ such that $C_i = \widehat{B}_{j^*}^{i,\ell}(D_{lev}D_{W_i}A_i)^{U_\ell}$. Then, for any $j \neq j^*$, consider the nonnegative vector $\widehat{B}_j^{i,\ell}(D_{lev}D_{W_i}A_i)^{U_\ell} = v$. We need to show that the support of v is lexicographically later than the support of C_i.

First, we claim that if $v \neq C_i$ then the support of C_i is not the same as the support of v. Let's prove it by getting a contradiction. Suppose $v \neq C_i$ and both of them have support R_i. Indeed R_i must correspond to a linearly independent set of columns of $D_{W_i}B$. By Claim 16.11.8, we obtain that $(D_{W_i}B(v-C_i))^{\Omega_i} = (\overrightarrow{0})^{\Omega_i}$. By $v - C_i \neq \overrightarrow{0}$ and support of $v - C_i$ contained in R_i, we get $(D_{W_i}B(v-C_i))^{\Omega_i} \neq (\overrightarrow{0})^{\Omega_i}$. Thus, we get a contradiction.

So the support C_i and v are not identical and one of these must be lexicographically earlier. Suppose (for contradiction) that the support of v is earlier. By Claim 16.11.7, we have that the support of C_i is a weighted admissible set of columns of B for A_i. This contradicts stability (because we could update C_i to v), and so we can conclude that the support of C_i is lexicographically earlier. □

It remains to prove Claim 16.11.7 and Claim 16.11.8, which are the weighted version of Lemma 2.8 and Lemma 2.9 in [Mad13].

Claim 16.11.7. *For each column vector $C_i \in \mathbb{R}^{k\times 1}$, it is contained in the set $\mathcal{S}$.*

Proof. Let R_i denote the support of column vector $C_i \in \mathbb{R}^{k\times 1}$. Then R_i must correspond to a linearly independent set of columns of B - otherwise we could find a nonnegative $\widetilde{C}_i$ whose support is a strict subset of R_i such that $W_i \circ (B\widetilde{C}_i) = W_i \circ A_i$, but this would violate the condition of stability.

Suppose $D_{lev}D_{W_i}$ belongs to zero pattern z_ℓ, let U_ℓ denote maximal linearly indepedent set of rows of $D_{lev}D_{W'_\ell}B$ and s_ℓ is the rank. Because the sets of linearly independent columns of $D_{lev}D_{W_i}B \in \mathbb{R}^{n\times k}$ are a matroid, there is a set $S_j \subset [k]$ of s_ℓ linearly independent columns of $D_{lev}D_{W_i}B$ for which $R_i \subset S_j$, hence

$$\begin{aligned}
& \widehat{B}_j^{i,\ell}(D_{lev}D_{W_i}A_i)^{U_\ell} \\
= \;& \widehat{B}_j^{i,\ell}(D_{lev}D_{W_i}BC_i)^{U_\ell} \text{ by } A_i = BC_i \\
= \;& \widehat{B}_j^{i,\ell}(D_{lev}D_{W_i}B)^{U_\ell}C_i \\
= \;& P_{\pi:[s_\ell]\to[k]} \cdot ((D_{lev}D_{W_i}B)_{S_j}^{U_\ell})^\dagger \cdot (D_{lev}D_{W_i}B)_{S_j}^{U_\ell} \cdot C_i \\
= \;& v
\end{aligned}$$

Note that $\widehat{B}_j^{i,\ell}$ is zero on rows outside the set S_j and support C_i is contained in S_j, then we have $C_i = v$ $\square$

Claim 16.11.8. *For each column vector $\widehat{B}_j^{i,\ell}(D_{lev}D_{W_i}A_i)^{U_\ell} \in \mathbb{R}^{k\times 1}$, we have*

$$W_i \circ (B\widehat{B}_j^{i,\ell}(D_{lev}D_{W_i}A_i)^{U_\ell}) = W_i \circ A_i$$

or equivalently

$$D_{W_i}(B\widehat{B}_j^{i,\ell}(D_{lev}D_{W_i}A_i)^{U_\ell}) = D_{W_i}A_i$$

Proof. Let Ω_i denote the support of W_i. Define $v = B\widehat{B}_j^{i,\ell}(D_{lev}D_{W_i}A_i)^{U_\ell,\Omega_i} \in \mathbb{R}^{n\times 1}$. First, we prove that $v^{U_\ell,\Omega_i} = A_i^{U_\ell,\Omega_i}$. Since $\widehat{B}_j^{i,\ell} \in \mathbb{R}^{k\times|U_\ell\cap\Omega_i|}$ is zero on rows outside the set S_j, we have

$$\underbrace{B}_{n\times k}\ \underbrace{\widehat{B}_j^{i,\ell}}_{k\times|U\cap\Omega_i|} = B_{S_j}(\widehat{B}_j^{i,\ell})^{S_j} = B_{S_j}((D_{lev}D_{W_i}B)_{S_j}^{U_\ell})^\dagger$$

Thus

$$\begin{aligned}
v^{U_\ell,\Omega_i} &= (B\widehat{B}_j^{i,\ell}(D_{lev}D_{W_i}A_i)^{U_\ell,\Omega_i})^{U_\ell,\Omega_i} \\
&= (B\widehat{B}_j^{i,\ell})^{U_\ell,\Omega_i}(D_{lev}D_{W_i}A_i)^{U_\ell,\Omega_i} \\
&= B^{U_\ell,\Omega_i}\widehat{B}_j^{i,\ell}(D_{lev}D_{W_i}A_i)^{U_\ell,\Omega_i} \\
&= B_{S_j}^{U_\ell,\Omega_i}((D_{lev}D_{W_i}B)_{S_j}^{U_\ell})^\dagger(D_{lev}D_{W_i}A_i)^{U_\ell,\Omega_i} \\
&= A_i^{U_\ell,\Omega_i}
\end{aligned}$$

Second, we prove that $v^{\overline{U_\ell},\Omega_i} = A_i^{\overline{U_\ell},\Omega_i}$. Since U_ℓ is the maximal linearly independent set of rows of $D_{lev}D_{W_i}B$ and D_{lev} preserves the rank of $D_{W_i}B$. For any $t \in \overline{U_\ell}\cap\Omega_i$, the row vector $B^t \in \mathbb{R}^{1\times k}$ can be written as a linear combination of rows in B in the set $U_\ell \cap \Omega_i$:

$$B^t = \sum_{t'\in U_\ell\cap\Omega_i} \alpha_{t,t'}B^{t'}$$

Note that $W_i \circ (BC_i) = W_i \circ A_i$ implies $(BC_i)^{\Omega_i} = (A_i)^{\Omega_i}$. Then we have for $t \in \overline{U_\ell} \cap \Omega_i$, $A_i^t = B^t C_i = \sum_{t' \in U_\ell \cap \Omega_i} \alpha_{t,t'} B^{t'} C_i = \sum_{t' \in U_\ell \cap \Omega_i} \alpha_{t,t'} A_i^{t'}$ and hence:

$$\begin{aligned} v^t &= (B\widehat{B}_j^{i,\ell}(D_{lev} D_{W_i} A_i)^{U_\ell, \Omega_i})^t \\ &= B^t \widehat{B}_j^{i,\ell}(D_{lev} D_{W_i} A_i)^{U_\ell, \Omega_i} \\ &= \sum_{t' \in U_\ell \cap \Omega_i} \alpha_{t,t'} B^{t'} \widehat{B}_j^{i,\ell}(D_{lev} D_{W_i} A_i)^{U_\ell, \Omega_i} \\ &= \sum_{t' \in U_\ell \cap \Omega_i} \alpha_{t,t'} v^{t'} \\ &= \sum_{t' \in U_\ell \cap \Omega_i} \alpha_{t,t'} A_i^{t'} \\ &= A_i^t \end{aligned}$$

□

We still need to show that leverage score sampling matrix doesn't destroy first lexicographically order property.

Claim 16.11.9. *If $\widetilde{C} \in \mathbb{R}^{k \times n}$ is the first lexicographically order solution of the original problem $A = BC$ under W, then it is also a solution to the problem $D_{lev} A = D_{lev} BC$ under W.*

Proof. Subspace embedding preserves the equality. For each $i \in [n]$, $D_{W_i} A_i = D_{W_i} B\widetilde{C}_i$ implies that $D_{lev} D_{W_i} A_i = D_{lev} D_{W_i} B\widetilde{C}_i$. □

Claim 16.11.10. *If $C' \in \mathbb{R}^{k \times n}$ is the first lexicographically order solution of the new problem $D_{lev} A = D_{lev} BC$ under W, then the C' has to be equal to $\widetilde{C}$.*

Proof. Let's prove it by contradiction. Suppose the order of C' is early than $\widetilde{C}$. For each $i \in [n]$, we know that $D_{lev}D_{W_i}A_i = D_{lev}D_{W_i}BC'_i$. Since we also know that $D_{lev}D_{W_i}A_i = D_{lev}D_{W_i}B\widetilde{C}_i$. Combing the above equation gives that $D_{lev}D_{W_i}BC'_i = D_{lev}D_{W_i}B\widetilde{C}_i$, since $C' \neq \widetilde{C}$, then there must exist one i such that $C'_i \neq \widetilde{C}_i$, which means $D_{W_i}BC'_i \neq D_{W_i}B\widetilde{C}_i$.

By leverage score sampling, if $D_{W_i}BC'_i \neq D_{W_i}B\widetilde{C}_i$, then $D_{lev}D_{W_i}BC'_i \neq D_{lev}D_{W_i}B\widetilde{C}_i$

□

16.11.3 A Semi-Algebraic Set with Double Exponential Depence in r

Here, we explain the details of constraints should be added into the polynomials system solver.

Constraints 1 Define first($\mathcal{S}$) to be the function that takes a collection of vectors as input and outputs the vector with lexicographically minimal support among all nonnegative vectors in $\mathcal{S}$. For each k-dimensional column vector $C_i, i \in [n]$, we need to add nonnegative constraints to it,

$$C_i \geq \overrightarrow{0}, \forall i \in [n]$$

where $C_i = \text{first}(\{\widehat{B}_1^{i,\ell}(D_{lev}D_{W_i}A_i)^{U_\ell}, \widehat{B}_2^{i,\ell}(D_{lev}D_{W_i}A_i)^{U_\ell}, \cdots, \widehat{B}_p^{i,\ell}(D_{lev}D_{W_i}A_i)^{U_\ell}\})$ and $D_{lev}D_{W_i}$ belong to the zero-pattern z_ℓ.

Similarly, we can choose a leverage score sampling matrix $D_{lev'}$ on the RHS of the original problem and create variables for $C^{\text{supp}(D_{lev'})}$. Thus, for each k-dimensional row vector $B^j, j \in [n]$, we also need to add nonnegative constraints to it,

$$B^j \geq \overrightarrow{0}, \forall j \in [n]$$

where $B^j = \text{first}(\{(A^jD_{W^j}D_{lev'})^{U_\ell}\widehat{C}_1^{j,\ell}, (A^jD_{W^j}D_{lev'})^{U_\ell}\widehat{C}_2^{j,\ell}, \cdots, (A^jD_{W^j}D_{lev'})^{U_\ell}\widehat{C}_p^{j,\ell}\})$ and $D_{W^j}D_{lev'}$ belong to the zero-pattern z_ℓ.

Constraints 2 Since we know the representation of C_i, B^j and the exact value of A_i^j, then we need to add n^2 equality constraints,

$$B^jC_i = A_i^j, \text{ if } W_i^j \neq 0, \forall i \in [n], \forall j \in [n]$$

where we need to clean up the denominator of B^j and C_i when writing down this constraint.

Constraints 3 We also need to write down the inequality constraints for the denominators of every $\widehat{B}_j^{i,\ell}$, by definition, the denominator is from $((D_{lev}D_{W_i}B)_{S_j}^{U_\ell})^\dagger$. To guarantee the denominator is nonzero, we need to add this constraint,

$$\text{the denominator of } ((D_{lev}D_{W_i}B)_{S_j}^{U_\ell})^\dagger \neq 0.$$

where the number of such constraints is

$$\#D_{W_i} \cdot \#S_j \leq n \cdot 2^k$$

which is same for the other side C.

Polynomial system Claim 2.12 in [Mad13] is able to write the explicit Boolean fucntion $\mathbb{P}$ for the unweighted case of nonnegative matrix factorization problem. Note that $\mathbb{P}$ is a function of sign constraints on polynomials. Here, we extend that the Boolean function to a weighted case of nonnegative matrix factorization.

$$\bigwedge_{\ell\in[g]}\bigwedge_{\ell'\in[g']}\left(\bigwedge_{i\in[z_\ell]}\bigwedge_{j\in[z_{\ell'}]}\left(\bigvee_{i_1\in[\binom{k}{s_\ell}]}\bigvee_{j_1\in[\binom{k}{\omega_{\ell'}}]}(f_1\wedge f_2\wedge f_3\wedge f_4\wedge f_5\wedge f_6\wedge f_7)\right)\right)$$

where

$$f_1 : \left((A^j D_{W^j} D_{lev'})_{V_{\ell'}} \widehat{C}_{j_1}^{j,\ell'} \cdot \widehat{B}_{i_1}^{i,\ell}(D_{lev}D_{W_i}A_i)^{U_\ell,} - A_j^i\right) = 0$$

$$f_2 := \left(\widehat{B}_{i_1}^{i,\ell}(D_{lev}D_{W_i}A_i)^{U_\ell}\right) \geq 0$$

$$f_3 := \left((A^j D_{W^j} D_{lev'})_{V_{\ell'}} \widehat{C}_{j_1}^{j,\ell'}\right) \geq 0$$

$$f_4 := \left(\widehat{B}_{i_1}^{i,\ell}(D_{lev}D_{W_i}A_i)^{U_\ell} \text{ is the first out of } \mathcal{S}^i\right)$$

$$f_5 := \left((A^j D_{W^j} D_{lev'})_{V_{\ell'}} \widehat{C}_{j_1}^{j,\ell'} \text{ is the first out of } \mathcal{T}^j\right)$$

$$f_6 := \left(\text{denominator polynomial of } \widehat{B}_{i_1}^{i,\ell}\right) \neq 0$$

$$f_7 := \left(\text{denominator polynomial of } \widehat{C}_{j_1}^{j,\ell'}\right) \neq 0$$

where the sign of the polynomial in each constraint can be recovered as a Boolean function of signs of degree at most poly(kr).

Theorem 16.11.11. *Given matrix $A \in \mathbb{R}^{n\times n}$ and rank-r weighted matrix $W \in \mathbb{R}_{\geq 0}^{n\times n}$. It takes $n^{O(rk^2)}2^{2^{O(r\log k)}}$ time to determine if there exists some matrices $B \in \mathbb{R}_{\geq 0}^{n\times k}$ and $C \in \mathbb{R}_{\geq 0}^{k\times n}$ such that $W \circ (BC) = W \circ A$, which means existing a weighted nonnegative matrix factorization.*

Proof. The final running time comes from two parts, (1) solving the polynomial system, it is known that the running time is

$$(\text{degree} \cdot \#\text{constraints})^{\#\text{variables}}$$

The number of variables is $O(tk) = O(rk^2)$. The degree is increasing by using Cramer rule to compute the inverse of full rank square matrix, the degree is at most poly(kr). The number of constraints is at most $n^2 \cdot 2^{\text{poly}(kr)}$. Thus, running time of polynomial system solver is at most $n^{O(rk^2)}2^{\text{poly}(kr)}$. (2) The number of guesses for leverage score sampling matrix is $n^{O(t)}$ and the number of guesses for each group's maximum linearly independent rows in B is $2^{k^{O(r)}}$. Thus, we need to rerun the solver for $n^{O(t)}2^{k^{O(r)}}$ different polynomial systems. In summary, the entire running is $n^{O(rk^2)}2^{k^{O(r)}}$ by plugging $t = O(kr)$. □

16.11.4 A Semi-Algebraic Set with Single Exponential Dependence in r

The intuition of improving double exponential dependence to single exponential dependence is, instead of guessing a possibility of rank and maximum linearly independent rows for each group at a time, we can write down all the possibilities into a slightly bigger polynomial system.

We define

$$\begin{aligned}
&\mathcal{S}^{i,s_\ell,p_{s_\ell}} = \\
&\{\widehat{B}^{i,\ell}_{1,1}(D_{lev}D_{W_i}A_i)^{U_{s_\ell,1}}, \cdots, \widehat{B}^{i,\ell}_{1,\binom{k}{s_\ell}}(D_{lev}D_{W_i}A_i)^{U_{s_\ell,1}} \\
&\widehat{B}^{i,\ell}_{2,1}(D_{lev}D_{W_i}A_i)^{U_{s_\ell,2}}, \cdots, \widehat{B}^{i,\ell}_{2,\binom{k}{s_\ell}}(D_{lev}D_{W_i}A_i)^{U_{s_\ell,2}}, \\
&\cdots, \\
&\widehat{B}^{i,\ell}_{p_{s_\ell},1}(D_{lev}D_{W_i}A_i)^{U_{s_\ell,p_{s_\ell}}}, \cdots, \widehat{B}^{i,\ell}_{p_{s_\ell},\binom{k}{s_\ell}}(D_{lev}D_{W_i}A_i)^{U_{s_\ell,p_{s_\ell}}}\}
\end{aligned}$$

where z_ℓ is corresponding to the zero-pattern of $D_{lev}D_{W_i}$, s_ℓ is the rank of $D_{lev}D_{W'_\ell}$, $U_{s_\ell,i}, \forall i \in [p_{s_\ell}]$ is a set of indices of s_ℓ rows of submatrix $D_{lev}D_{W'_\ell}$, $p_{s_\ell} = \binom{t}{s_\ell} \leq \binom{t}{k} = O(r^k)$. Then we define

$$\mathcal{S}^i = \bigcup_{s_\ell=1}^{k} \mathcal{S}^{i,s_\ell,p_{s_\ell}}$$

where the number of entries in set S^i is at most $k \cdot r^k \cdot 2^k$.

Then the Constraints 3 is slightly changing to the following,

Constraints 3' We also need to write down the inequality constraints for the denominators of every $\widehat{B}^{i,\ell}_{i_1,i_2}$, by definition, the denominator is from $((D_{lev}D_{W_i}B)^{U_{s_\ell,i_1}}_{S_{i_2}})^\dagger$. To guarantee the

denominator is nonzero, we need to add this constraint,

$$\text{the denominator of } ((D_{lev} D_{W_i} B)^{U_{s_\ell, i_1}}_{S_{i_2}})^\dagger \neq 0.$$

where the number of such constraints is

$$\#D_{W_i} \cdot \#s_\ell \cdot \#i_1 \cdot \#S_{i_2} \leq n \cdot k \cdot r^{O(k)} \cdot 2^k = n \cdot 2^{O(k \log r)}$$

which is same for the other side C.

Polynomial System Finally, we can write down the boolean function $\mathbb{P}$ over the polynomial inequality constraints,

$$\bigwedge_{\ell \in [g]} \bigwedge_{\ell' \in [g']} \left(\bigvee_{s_\ell \in [k]} \bigvee_{\omega_{\ell'} \in [k]} \bigvee_{i_1 \in [p_{s_\ell}]} \bigvee_{j_1 \in [q_{\omega_{\ell'}}]} \left(\bigwedge_{i \in [z_\ell]} \bigwedge_{j \in [z_{\ell'}]} \left(\bigvee_{i_2 \in [\binom{k}{s_\ell}]} \bigvee_{j_2 \in [\binom{k}{\omega_{\ell'}}]} (f_1 \wedge f_2 \wedge \cdots \wedge f_6 \wedge f_7) \right) \right) \right)$$

where

$$f_1 := \left((A^j D_{W^j} D_{lev'})_{V_{\omega_{\ell'}, j_1}} \widehat{C}^{j, \ell'}_{j_1, j_2} \cdot \widehat{B}^{i, \ell}_{i_1, i_2} (D_{lev} D_{W_i} A_i)^{U_{s_\ell}, i_1} - A^i_j \right) = 0$$

$$f_2 := \left(\widehat{B}^{i, \ell}_{i_1, i_2} (D_{lev} D_{W_i} A_i)^{U_{s_\ell}, i_1} \right) \geq 0$$

$$f_3 := \left((A^j D_{W^j} D_{lev'})_{V_{\omega_{\ell'}, j_1}} \widehat{C}^{j, \ell'}_{j_1, j_2} \right) \geq 0$$

$$f_4 := \left(\widehat{B}^{i, \ell}_{i_1, i_2} (D_{lev} D_{W_i} A_i)^{U_{s_\ell}, i_1} \text{ is the first out of } \mathfrak{S}^i \right)$$

$$f_5 := \left((A^j D_{W^j} D_{lev'})_{V_{\omega_{\ell'}, j_1}} \widehat{C}^{j, \ell'}_{j_1, j_2} \text{ is the first out of } \mathfrak{T}^j \right)$$

$$f_6 := \left(\text{denominator polynomial of } \widehat{B}^{i,\ell}_{i_1,i_2}\right) \neq 0$$

$$f_7 := \left(\text{denominator polynomial of } \widehat{C}^{j,\ell'}_{j_1,j_2}\right) \neq 0$$

Note that we only need to adding the functions f_1, f_2, f_3, f_4, f_5 into the this big polynomial system when $W_{ji} \neq 0$.

Theorem 16.11.12. *Given matrix $A \in \mathbb{R}^{n\times n}$ and rank-r weighted matrix $W \in \mathbb{R}^{n\times n}_{\geq 0}$. It takes $n^{O(rk^2)}2^{\widetilde{O}(rk^3)}$ time to determine if there exists some matrices $B \in \mathbb{R}^{n\times k}_{\geq 0}$ and $C \in \mathbb{R}^{k\times n}_{\geq 0}$ such that $W \circ (BC) = W \circ A$, which means existing a weighted nonnegative matrix factorization.*

Algorithm 16.1 Bicriteria Approximation Algorithm — Section 16.10

procedure LowRankApproximation(A, n, k, ϵ)
    Use [CW13] to find rank-k matrix A' such that:
    $\|A' - A\|_F^2 \leq (1+\epsilon) \min_{\text{rank}(B)=k} \|B - A\|_F^2$.
    return A'.
end procedure
procedure BicriteriaLowRankBooleanW$(A, W, n, r, k, \epsilon)$ — Section 16.10.1
    Comment rank$(W) = r$ and $W \in \mathbb{F}_2^{n \times n}$.
    $B \leftarrow$ LowRankApproximation$(A \circ W, n, rk, \epsilon)$.
    return B.
end procedure
procedure BicriteriaLowRankFiniteFieldW$(A, W, n, r, k, p, \epsilon)$ — Section 16.10.2
    Comment rank$(W) = r$ and $W \in \mathbb{F}_p^{n \times n}$.
    $g \leftarrow$ # distinct columns of W.
    for $i = 1 \rightarrow g$ **do**
        Let Z_i denote the subset of columns of A such that, all the columns have the same weight column vector.
        $\Omega_i \leftarrow \text{supp}(D_{W_i})$.
        $B_i \leftarrow (D_{W_i} A_{Z_i})^{\Omega_i}$.
        $B_i' \leftarrow$ LowRankApproximation$(B_i, (|\Omega_i|, |Z_i|), k, \epsilon)$.
    end for
    Reconstruct rank $k \cdot p^r$ matrix B by merging all B_i'.
    return B.
end procedure
procedure BicriteriaRDistinctColumnsW$(A, W, n, r, k, \epsilon)$ — Section 16.10.3
    Comment W has r distinct columns and $W \in \mathbb{R}_{\geq 0}^{n \times n}$
    for $i = 1 \rightarrow r$ **do**
        Let Z_i denote the subset of columns of A such that, all the columns have the same weight column vector.
        $\Omega_i \leftarrow \text{supp}(D_{W_i})$.
        $B_i \leftarrow (D_{W_i} A_{Z_i})^{\Omega_i}$.
        $B_i' \leftarrow$ LowRankApproximation$(B_i, (|\Omega_i|, |Z_i|), k, \epsilon)$.
    end for
    Reconstruct rank rk matrix B by merging all B_i'.
    return B
end procedure

Algorithm 16.2 Weighted Nonnegative Matrix Factorization

procedure WeightedNonnegativeMatrixFactorization(A, W, n, r, k)
  $t \leftarrow O(kr/\epsilon^2)$
  Guess leverage sampling matrix D_{lev} and $D_{lev'}$
  Create t variables for the nonzero probablities on diagonal of D_{lev}.
  Create t variables for the nonzero probablities on diagonal of $D_{lev'}$.
  Create tr variables for r rows of B based on the nonzero rows of D_{lev}.
  Create rt variables for r columns of C based on the nonzero columns of $D_{lev'}$.
  for $i \in [n]$ **do**
    Write down $D_{lev} D_{W_i} B$ and $C D_{W^i} D_{lev'}$.
  end for
  $g \leftarrow$ # zero-patterns of $\{D_{lev} D_{W_i}\}_{i \in [n]}$
  $g' \leftarrow$ # zero-patterns of $\{D_{W^i} D_{lev'}\}_{i \in [n]}$
  for $\ell \in [g], \ell' \in [g']$ **do**
    Guess a rank $s_\ell \in [k]$ for zero-pattern $D_{lev} D_{W'_\ell}$
    Guess a maximal linearly independent set of rows U_ℓ for $D_{lev} D_{W'_\ell} B$, $|U_\ell| = s_\ell$
    Guess a rank $s_{\ell'} \in [k]$ for zero-pattern $D_{lev} D_{W'_{\ell'}}$
    Guess a maximal linearly independent set of columns $V_{\ell'}$ for $C D_{W'_{\ell'}} D_{lev'}$, $|U_\ell| = s_\ell$
  end for
  for $i \in [n], j \in [p]$ **do**
    Write down $\widehat{B}_j^{i,\ell}$ in terms of $O(rt)$ variables created early
    Write down $\widehat{C}_j^{i,\ell'}$ in terms of $O(rt)$ variables created early
  end for
  Write down semi-algebraic set
  Run an algorithm for deciding if the semi-algebraic set is non-empty.
end procedure

Algorithm 16.3 Weighted Nonnegative Matrix Factorization

procedure FasterWeightedNonnegativeMatrixFactorization(A, W, n, r, k)
  $t \leftarrow O(kr/\epsilon^2)$
  Guess leverage sampling matrix D_{lev} and $D_{lev'}$
  Create t variables for the nonzero probablities on diagonal of D_{lev}.
  Create t variables for the nonzero probablities on diagonal of $D_{lev'}$.
  Create tr variables for r rows of B based on the nonzero rows of D_{lev}.
  Create rt variables for r columns of C based on the nonzero columns of $D_{lev'}$.
  for $i \in [n]$ **do**
    Write down $D_{lev}D_{W_i}B$ and $CD_{W^i}D_{lev'}$.
  end for
  $g \leftarrow$ # zero-patterns of $\{D_{lev}D_{W_i}\}_{i\in[n]}$
  $g' \leftarrow$ # zero-patterns of $\{D_{W^i}D_{lev'}\}_{i\in[n]}$
  for $\ell \in [g]$ **do**
    for $s_\ell \in [k]$ **do**
      rank $\leftarrow s_\ell$, $p_{s_\ell} \leftarrow$ # size s_ℓ subset rows of $D_{lev}D_{W'_\ell}B$
      for $i_1 \in [p_{s_\ell}]$ **do**
        for $i \in [z_\ell]$ **do**
          for $i_2 \in [\binom{k}{s_\ell}]$ **do**
            Write down $\widehat{B}^{i,\ell}_{i_1,i_2}$ in terms of $O(rt)$ variables created early, by Cramer rule
          end for
        end for
      end for
    end for
  end for
  for $\ell' \in [g']$ **do**
    for $\omega_{\ell'} \in [k]$ **do**
      rank $\leftarrow \omega_{\ell'}$, $q_{\omega_{\ell'}} \leftarrow$ # size $\omega_{\ell'}$ subset columns of $CD_{W'_{\ell'}}D_{lev'}$
      for $j_1 \in [q_{\omega_\ell}]$ **do**
        for $j \in [z_{\ell'}]$ **do**
          for $j_2 \in [\binom{k}{\omega_{\ell'}}]$ **do**
            Write down $\widehat{C}^{j,\ell}_{j_1,j_2}$ in terms of $O(rt)$ variables created early, by Cramer rule
          end for
        end for
      end for
    end for
  end for
  Write down semi-algebraic set
  Run an algorithm for deciding if the semi-algebraic set is non-empty.
end procedure

Chapter 17

Entry-wise L1 Low Rank Approximation

We study the ℓ_1-low rank approximation problem, where for a given $n \times d$ matrix A and approximation factor $\alpha \geq 1$, the goal is to output a rank-k matrix $\widehat{A}$ for which

$$\|A - \widehat{A}\|_1 \leq \alpha \cdot \min_{\text{rank-}k \text{ matrices } A'} \|A - A'\|_1,$$

where for an $n \times d$ matrix C, we let $\|C\|_1 = \sum_{i=1}^{n} \sum_{j=1}^{d} |C_{i,j}|$. This error measure is known to be more robust than the Frobenius norm in the presence of outliers and is indicated in models where Gaussian assumptions on the noise may not apply. The problem was shown to be NP-hard by Gillis and Vavasis and a number of heuristics have been proposed. It was asked in multiple places if there are any approximation algorithms.

We give the first provable approximation algorithms for ℓ_1-low rank approximation, showing that it is possible to achieve approximation factor $\alpha = (\log d) \cdot \text{poly}(k)$ in $\text{nnz}(A) + (n + d)\,\text{poly}(k)$ time, where $\text{nnz}(A)$ denotes the number of non-zero entries of A. If k is constant, we further improve the approximation ratio to $O(1)$ with a $\text{poly}(nd)$-time algorithm. Under the Exponential Time Hypothesis, we show there is no $\text{poly}(nd)$-time algorithm achieving a $(1 + \frac{1}{\log^{1+\gamma}(nd)})$-approximation, for $\gamma > 0$ an arbitrarily small constant, even when $k = 1$.

17.1 Introduction

Two well-studied problems in numerical linear algebra are regression and low rank approximation. In regression, one is given an $n \times d$ matrix A, and an $n \times 1$ vector b, and one seeks an $x \in \mathbb{R}^d$ which minimizes $\|Ax - b\|$ under some norm. For example, for least squares regression one minimizes $\|Ax - b\|_2$. In low rank approximation, one is given an $n \times d$ matrix A, and one seeks a rank-k matrix $\widehat{A}$ which minimizes $\|A - \widehat{A}\|$ under some norm. For example, in Frobenius norm low rank approximation, one minimizes $\|A - \widehat{A}\|_F = \left(\sum_{i,j}(A_{i,j} - \widehat{A}_{i,j})^2\right)^{1/2}$. Algorithms for regression are often used as subroutines for low rank approximation. Indeed, one of the main insights of [DMM06c, DMM06b, Sar06, DMM08, CW09] was to use results for generalized least squares regression for Frobenius norm low rank approximation. Algorithms for ℓ_1-regression, in which one minimizes $\|Ax - b\|_1 = \sum_i |(Ax)_i - b_i|$, were also used [BD13, SW11] to fit a set of points to a hyperplane, which is a special case of entrywise ℓ_1-low rank approximation, the more general problem being to find a rank-k matrix $\widehat{A}$ minimizing $\sum_{i,j} |A_{i,j} - \widehat{A}_{i,j}|$.

Randomization and approximation were introduced to significantly speed up algorithms for these problems, resulting in algorithms achieving relative error approximation with high probability. Such algorithms are based on sketching and sampling techniques; we refer to [Woo14b] for a survey. For least squares regression, a sequence of work [Sar06, CW13, MM13, NN13a, LMP13, BDN15, Coh16a] shows how to achieve algorithms running in $\text{nnz}(A) + \text{poly}(d)$ time. For Frobenius norm low rank approximation, using the advances for regression this resulted in $\text{nnz}(A) + (n + d)\,\text{poly}(k)$ time algorithms. For ℓ_1-regression, sketching and sampling-based methods [Cla05, SW11, CDMI+13, CW13, MM13, LMP13, WZ13, CW15b, CP15] led to an $\text{nnz}(A) + \text{poly}(d)$ time algorithm.

Just like Frobenius norm low rank approximation is the analogue of least squares regression, entrywise ℓ_1-low rank approximation is the analogue of ℓ_1-regression. Despite this analogy, *no non-trivial upper bounds with provable guarantees* are known for ℓ_1-low rank approximation. Unlike Frobenius norm low rank approximation, which can be solved exactly using the singular value decomposition, no such algorithm or closed-form solution is known for ℓ_1-low rank approximation. Moreover, the problem was recently shown to be NP-hard [GV15]. A major open question is whether there exist approximation algorithms, sketching-based or otherwise, for ℓ_1-low rank approximation. Indeed, the question of obtaining betters algorithms was posed in section 6 of [GV15], in [Exc13], and as the second part of open question 2 in [Woo14b], among other places. The earlier question of NP-hardness was posed in Section 1.4 of [KV09], for which the question of obtaining approximation algorithms is a natural followup. The goal of our work is to answer this question.

We now formally define the ℓ_1-low rank approximation problem: we are given an $n \times d$ matrix A and approximation factor $\alpha \geq 1$, and we would like, with large constant probability, to output a rank-k matrix $\widehat{A}$ for which

$$\|A - \widehat{A}\|_1 \leq \alpha \cdot \min_{\text{rank-}k \text{ matrices } A'} \|A - A'\|_1,$$

where for an $n \times d$ matrix C, we let $\|C\|_1 = \sum_{i=1}^{n} \sum_{j=1}^{d} |C_{i,j}|$. This notion of low rank approximation has been proposed as a more robust alternative to Frobenius norm low rank approximation [KK03, KK05, KLC+15, Kwa08, ZLS+12, BJ12, BD13, BDB13, MXZZ13, MKP13, MKP14, MKCP16, PK16], and is sometimes referred to as ℓ_1-matrix factorization or robust PCA. ℓ_1-low rank approximation gives improved results over Frobenius norm low rank approximation since outliers are less exaggerated, as one does not square their contribution

in the objective. The outlier values are often erroneous values that are far away from the nominal data, appear only a few times in the data matrix, and would not appear again under normal system operation. These works also argue ℓ_1-low rank approximation can better handle missing data, is appropriate in noise models for which the noise is not Gaussian, e.g., it produces the maximum likelihood estimator for Laplacian noise [Gao08, KAC+08, VT01], and can be used in image processing to prevent image occlusion [YZD12].

To see that ℓ_1-low rank approximation and Frobenius norm low rank approximation can give very different results, consider the $n \times n$ matrix $A = \left[\begin{smallmatrix} n & 0 \\ 0 & B \end{smallmatrix}\right]$, where B is *any* $(n-1) \times (n-1)$ matrix with $\|B\|_F < n$. The best rank-1 approximation with Frobenius norm error is given by $\widehat{A} = n \cdot e_1 e_1^\top$, where e_1 is the first standard unit vector. Here $\widehat{A}$ ignores all but the first row and column of A, which may be undesirable in the case that this row and column represent an outlier. Note $\|A - \widehat{A}\|_1 = \|B\|_1$. If, for example, B is the all 1s matrix, then $\widehat{A} = [0, 0; 0, B]$ is a rank-1 approximation for which $\|A - \widehat{A}\|_1 = n$, and therefore this solution is a much better solution to the ℓ_1-low rank approximation problem than $n \cdot e_1 e_1^\top$, for which $\|A - n \cdot e_1 e_1^\top\|_1 = (n-1)^2$.

Despite the advantages of ℓ_1-low rank approximation, its main disadvantage is its computationally intractability. It is not rotationally invariant and most tools for Frobenius low rank approximation do not apply. To the best of our knowledge, all previous works only provide heuristics. Using that for an $n \times d$ matrix C, $\|C\|_F \leq \|C\|_1 \leq \sqrt{nd}\|C\|_F$, a Frobenius norm low rank approximation gives a $\sqrt{nd}$ approximation for ℓ_1-low rank approximation. A bit better is to use algorithms for low rank approximation with respect to the sum of distances, i.e., to find a rank-k matrix $\widehat{A}$ minimizing $\|A - \widehat{A}\|_{1,2}$, where for an $n \times d$ matrix C, $\|C\|_{1,2} = \sum_{i=1}^n \|C_i\|_2$, where C_i is the i-th row of C. A sequence of work

[DV07, FMSW10, FL11, SV12a, CW15a] shows how to obtain an $O(1)$-approximation to this problem in $\text{nnz}(A)+(n+d)\,\text{poly}(k)+\exp(k)$ time, and using that $\|C\|_{1,2} \leq \|C\|_1 \leq \sqrt{d}\|C\|_{1,2}$ results in an $O(\sqrt{d})$-approximation.

17.1.1 Our Results

We give the first efficient algorithms for ℓ_1-low rank approximation with provable approximation guarantees. By symmetry of the problem, we can assume $d \leq n$. We first give an algorithm which runs in $O(\text{nnz}(A)) + n \cdot \text{poly}(k)$ time and solves the ℓ_1-low rank approximation problem with approximation factor $(\log d) \cdot \text{poly}(k)$. This is an exponential improvement over the previous approximation factor of $O(\sqrt{d})$, provided k is not too large, and is polynomial time for every k. Moreover, provided $\text{nnz}(A) \geq n \cdot \text{poly}(k)$, our time is optimal up to a constant factor as any relative error algorithm must spend $\text{nnz}(A)$ time. We also give a hard instance for our algorithm ruling out $\frac{\log d}{k \log k} + k^{1/2-\gamma}$ approximation for arbitrarily small constant $\gamma > 0$, and hard instances for a general class of algorithms based on linear sketches, ruling out $k^{1/2-\gamma}$ approximation.

Via a different algorithm, we show how to achieve an $\widetilde{O}(k)$-approximation factor in $\text{poly}(n) d^{\widetilde{O}(k)} 2^{\widetilde{O}(k^2)}$ time. This is useful for constant k, for which it gives an $O(1)$-approximation in $\text{poly}(n)$ time, improving the $O(\log d)$-approximation for constant k of our earlier algorithm. The approximation ratio of this algorithm, although $O(1)$ for constant k, depends on k. We also show one can find a rank-$2k$ matrix $\widehat{A}$ in $\text{poly}(n)$ time for constant k for which $\|A - \widehat{A}\|_1 \leq C \min_{\text{rank-}k \text{ matrices } A'} \|A - A'\|_1$, where $C > 1$ is an absolute constant independent of k. We refer to this as a bicriteria algorithm. Finally, one can output a rank-k matrix $\widehat{A}$, instead of a rank-$2k$ matrix $\widehat{A}$, in $\text{poly}(n)$ time with the same absolute constant C approximation factor, under an additional assumption that the entries of $\widehat{A}$ are integers in the range $\{-b, -b+1, \ldots, b\}$ for an integer $b \leq \text{poly}(n)$. Unlike our previous algorithms, this very last algorithm has a bit complexity assumption, and runs in $\text{poly}(b)$ time instead of $\text{poly}(\log(b))$ time.

We also give a number of results for variants of ℓ_1-low rank approximation which are studied for Frobenius norm low rank approximation; prior to our work nothing was known about these problems.

Column Subset Selection and CUR Decomposition: In the column subset selection problem, one seeks a small subset C of columns of A for which there is a matrix X for which $\|CX - A\|$ is small, under some norm. The matrix CX provides a low rank approximation to A which is often more interpretable, since it stores actual columns of A, preserves sparsity, etc. These have been extensively studied when the norm is the Frobenius or operator norm (see, e.g., [BMD09, DR10, BDM11] and the references therein). We initiate the study of this problem with respect to the ℓ_1-norm. We first prove an existence result, namely, that there exist matrices A for which any subset C of $\mathrm{poly}(k)$ columns satisfies $\min_X \|CX - A\|_1 \geq k^{1/2-\gamma} \cdot \min_{\text{rank-}k \text{ matrices } A'} \|A - A'\|_1$, where $\gamma > 0$ is an arbitrarily small constant. This result is in stark contrast to the Frobenius norm for which for every matrix there exist $O(\frac{k}{\epsilon})$ columns for which the approximation factor is $1+\epsilon$. We also show that our bound is nearly optimal in this regime, by showing for every matrix there exists a subset of $O(k \log k)$ columns providing an $O(\sqrt{k \log k})$-approximation. One can find such columns in $\mathrm{poly}(n) d^{O(k \log k)}$ time by enumerating and evaluating the cost of each subset. Although this is exponential in k, we show it is possible to find $O(k \log k)$ columns providing an $O(\sqrt{k \log k} \log d)$-approximation in polynomial time for every k.

We extend these results to the CUR decomposition problem (see, e.g., [DMM08, BW14]), in which one seeks a factorization CUR for which C is a subset of columns of A, R is a subset of rows of A, and $\|CUR - A\|$ is as small as possible. In the case of Frobenius norm, one can choose $O(k/\epsilon)$ columns and rows, have $\mathrm{rank}(U) = k$, have $\|CUR - A\|_F$ be at most

$(1+\epsilon)$ times the optimal cost, and find the factorization in $\text{nnz}(A)\log n + n \cdot \text{poly}((\log n)k/\epsilon)$ time [BW14]. Using our column subset selection results, we give an $\text{nnz}(A) + n \cdot \text{poly}(k)$ time algorithm choosing $O(k \log k)$ columns and rows, for which $\text{rank}(U) = k$, and for which $\|CUR - A\|_1$ is $\text{poly}(k) \log d$ times the cost of any rank-k approximation to A.

17.1.2 Technical Overview

Initial Algorithm and Optimizations: Let A^* be a rank-k matrix for which $\|A - A^*\|_1 = \min_{\text{rank-}k \text{ matrices } A'} \|A - A'\|_1$. Let $A^* = U^*V^*$ be a factorization for which U^* is $n \times k$ and V^* is $k \times d$. Suppose we somehow knew U^* and consider the multi-response ℓ_1-regression problem $\min_V \|U^*V - A\|_1 = \min_V \sum_{i=1}^d \|U^*V_i - A_i\|_1$, where V_i, A_i denote the i-th columns of V and A, respectively. We could solve this with linear programming though this is not helpful for our argument here.

Instead, inspired by recent advances in sketching for linear algebra (see, e.g., [Woo14b] for a survey), we could choose a random matrix S and solve $\min_V \|SU^*V - SA\|_1 = \min_V \sum_{i=1}^d \|(SU^*)V_i - SA_i\|_1$. If V is an approximate minimizer of the latter problem, we could hope V is an approximate minimizer of the former problem. If also S has a small number t of rows, then we could instead solve $\min_V \sum_{i=1}^d \|(SU^*)V_i - SA_i\|_2$, that is, minimize the sum of Euclidean norms rather than the sum of ℓ_1-norms. Since $t^{-1/2}\|(SU^*)V_i - SA_i\|_1 \leq \|(SU^*)V_i - SA_i\|_2 \leq \|(SU^*)V_i - SA_i\|_1$, we would obtain a $\sqrt{t}$-approximation to the problem $\min_V \|SU^*V - SA\|_1$. A crucial observation is that the solution to $\min_V \sum_{i=1}^d \|(SU^*)V_i - SA_i\|_2$ is given by $V = (SU^*)^\dagger SA$, which implies that V is in the row span of SA. If also S were *oblivious* to U^*, then we could compute SA without ever knowing U^*. Having a low-dimensional space containing a good solution in its span is our starting point.

For this to work, we need a distribution on oblivious matrices S with a small number of rows, for which an approximate minimizer V to $\min_V \|SU^*V - SA\|_1$ is also an approximate minimizer to $\min_V \|U^*V - A\|_1$. It is *unknown* if there exists a distribution on S with this property. What is known is that if S has $O(d \log d)$ rows, then the Lewis weights (see, e.g., [CP15] and references therein) of the concatenated matrix $[U^*, A]$ give a distribution for

which the optimal V for the latter problem is a $(1+\epsilon)$-approximation to the former problem; see also earlier work on ℓ_1-leverage scores [Cla05, DDH+09] which have poly(d) rows and the same $(1+\epsilon)$-approximation guarantee. Such distributions are not helpful here as (1) they are not oblivious, and (2) the number $O(d \log d)$ of rows gives an $O(\sqrt{d \log d})$ approximation factor, which is much larger than what we want.

There are a few oblivious distributions S which are useful for *single-response* ℓ_1-regression $\min \|U^*v - a\|_1$ for column vectors $v, a \in \mathbb{R}^k$ [SW11, CDMI+13, WZ13]. In particular, if S is an $O(k \log k) \times n$ matrix of i.i.d. Cauchy random variables, then the solution v to $\min \|SU^*v - Sa\|_1$ is an $O(k \log k)$-approximation to $\min \|U^*v - a\|_1$ [SW11]. The important property of Cauchy random variables is that if X and Y are independent Cauchy random variables, then $\alpha X + \beta Y$ is distributed as a Cauchy random variable times $|\alpha| + |\beta|$, for any scalars $\alpha, \beta \in \mathbb{R}$. The $O(k \log k)$ approximation arises because all possible regression solutions are in the column span of $[U^*, a]$ which is $(k+1)$-dimensional, and the sketch S gives an approximation factor of $O(k \log k)$ to preserve every vector norm in this subspace. If we instead had a multi-response regression problem $\min \|SU^*V^* - SA\|_1$ the dimension of the column span of $[U^*, A]$ would be $d + k$, and this approach would give an $O(d \log d)$-approximation. Unlike Frobenius norm multi-response regression $\min \|SU^*V^* - SA\|_F$, which can be bounded if S is a subspace embedding for U^* and satisfies an approximate matrix product theorem [Sar06], there is no convenient linear-algebraic analogue for the ℓ_1-norm.

We first note that since regression is a minimization problem, to obtain an $O(\alpha)$-approximation by solving the sketched version of the problem, it suffices that (1) for the optimal V^*, we have $\|SU^*V^* - SA\|_1 \leq O(\alpha)\|U^*V^* - A\|_1$, and (2) for all V, we have $\|SU^*V - SA\|_1 \geq \Omega(1) \cdot \|U^*V - A\|_1$.

We show (1) holds for $\alpha = O(\log d)$ and any number of rows of S. Our analysis follows by truncating the Cauchy random variables $(SU^*V_j^* - SA_j)_i$ for $i \in [O(k \log k)]$ and $j \in [d]$, so that their expectation exists, and applying linearity of expectation across the d columns. This is inspired from an argument of Indyk [Ind06] for embedding a vector into a lower-dimensional vector while preserving its ℓ_1-norm; for *single-response* regression this is the statement that $\|SU^*v^* - Sa\|_1 = \Theta(1)\|U^*v - a\|_1$, implied by [Ind06]. However, for *multi-response* regression we have to work entirely with expectations, rather than the tail bounds in [Ind06], since the Cauchy random variables $(SU^*V_j - SA_j)_i$, while independent across i, are dependent across j. Moreover, our $O(\log d)$-approximation factor is not an artifact of our analysis - we show in Section 17.6 that there is an $n \times d$ input matrix A for which with probability $1 - 1/\operatorname{poly}(k)$, *there is no k-dimensional space in the span of SA achieving a $\left(\frac{\log d}{t \log t} + k^{1/2-\gamma}\right)$-approximation*, for S a Cauchy matrix with t rows, where $\gamma > 0$ is an arbitrarily small constant. This shows $(k \log d)^{\Omega(1)}$-inapproximability. Thus, the fact that we achieve $O(\log d)$-approximation instead of $O(1)$ is fundamental for a matrix S of Cauchy random variables or any scaling of it.

While we cannot show (2), we instead show for all V, $\|SU^*V - SA\|_1 \geq \|U^*V - A\|_1/2 - O(\log d)\|U^*V^* - A\|_1$ if S has $O(k \log k)$ rows. This suffices for regression, since the only matrices V for which the cost is much smaller in the sketch space are those providing an $O(\log d)$ approximation in the original space. The guarantee follows from the triangle inequality: $\|SU^*V - SA\|_1 \geq \|SU^*V - SU^*V^*\|_1 - \|SU^*V^* - SA\|_1$ and the fact that S is known to not contract any vector in the column span of U^* if S has $O(k \log k)$ rows [SW11]. Because of this, we have $\|SU^*V - SU^*V^*\|_1 = \Omega(1)\|U^*V - U^*V^*\|_1 = \Omega(1)(\|U^*V - A\|_1 - \|U^*V^* - A\|_1)$, where we again use the triangle inequality. We also bound the additive term

$\|SU^*V^* - SA\|_1$ by $O(\log d)\|U^*V^* - A\|_1$ using (1) above.

Given that SA contains a good rank-k approximation in its row span, our algorithm with a slightly worse poly(n) time and poly$(k\log(n))$-approximation can be completely described here. Let S and T_1 be independent $O(k\log k)\times n$ matrices of i.i.d. Cauchy random variables, and let R and T_2 be independent $d\times O(k\log k)$ matrices of i.i.d. Cauchy random variables. Let

$$X = (T_1AR)^\dagger((T_1AR)(T_1AR)^\dagger(T_1AT_2)(SAT_2)(SAT_2)^\dagger)_k(SAT_2)^\dagger,$$

which is the rank-k matrix minimizing $\|T_1ARXSAT_2 - T_1AT_2\|_F$, where for a matrix C, C_k is its best rank-k approximation in Frobenius norm. Output $\widehat{A} = ARXSA$ as the solution to ℓ_1-low rank approximation of A. We show with constant probability that $\widehat{A}$ is a poly$(k\log(n))$-approximation.

To improve the approximation factor, after computing SA, we ℓ_1-project each of the rows of A onto SA using linear programming or fast algorithms for ℓ_1-regression [CW13, MM13], obtaining an $n\times d$ matrix B of rank $O(k\log k)$. We then apply the algorithm in the previous paragraph with A replaced by B. This ultimately leads to a $\log d\cdot$ poly(k)-approximation.

To improve the running time from poly(n) to nnz$(A) + n\cdot$ poly(k), we show a similar analysis holds for the sparse Cauchy matrices of [MM13]; see also the matrices in [WZ13].

CUR Decompositions: To obtain a CUR decomposition, we first find a $\log d\cdot$ poly(k)-approximate rank-k approximation $\widehat{A}$ as above. Let B_1 be an $n\times k$ matrix whose columns span those of $\widehat{A}$, and consider the regression $\min_V \|B_1V - A\|_1$. Unlike the problem $\min_V \|U^*V -$

$A\|_1$ where U^* was unknown, we *know* B_1 so can compute its Lewis weights efficiently, sample by them, and obtain a regression problem $\min_V \|D_1(B_1V - A)\|_1$ where D_1 is a sampling and rescaling matrix. Since

$$\|D_1(B_1V - A)\|_1 \leq \|D_1(B_1V - B_1V^*)\|_1 + \|D_1(B_1V^* - A)\|_1,$$

where $V^* = \text{argmin}_V\|B_1V - A\|_1$, we can bound the first term by $O(\|B_1V - B_1V^*\|_1)$ using that D_1 is a subspace embedding if it has $O(k \log k)$ rows, while the second term is $O(1)\|B_1V^* - A\|_1$ by a Markov bound. Note that

$$\|B_1V^* - A\|_1 \leq (\log d) \cdot \text{poly}(k) \min_{\text{rank-}k \text{ matrices } A'} \|A - A'\|_1.$$

By switching to ℓ_2 as before, we see that $\widehat{V} = (D_1B_1)^\dagger D_1A$ contains a $(\log d)\,\text{poly}(k)$-approximation in its span. Here D_1A is an actual subset of rows of A, as required in a CUR decomposition. Moreover the subset size is $O(k \log k)$. We can sample by the Lewis weights of $\widehat{V}$ to obtain a subset C of $O(k \log k)$ rescaled columns of A, together with a rank-k matrix U for which

$$\|CUR - A\|_1 \leq (\log d)\,\text{poly}(k) \min_{\text{rank-}k \text{ matrices } A'} \|A - A'\|_1.$$

Algorithm for Small k: Our CUR decomposition shows how we might obtain an $O(1)$-approximation for constant k in $\text{poly}(n)$ time. If we knew the Lewis weights of U^*, an α-approximate solution to the problem $\min_V \|D_1(U^*V - A)\|_1$ would be an $O(\alpha)$-approximate solution to the problem $\min_V \|U^*V - A\|_1$, where D_1 is a sampling and rescaling matrix of $O(k \log k)$ rows of A. Moreover, an $O(\sqrt{k \log k})$-approximate solution to $\min_V \|D_1(U^*V -$

$A)\|_1$ is given by $V = (D_1U^*)^\dagger D_1A$, which implies the $O(k \log k)$ rows of D_1A contain an $O(\sqrt{k \log k})$-approximation. For small k, we can guess every subset of $O(k \log k)$ rows of A in $n^{O(k \log k)}$ time (if $d \ll n$, by taking transposes at the beginning one can replace this with $d^{O(k \log k)}$ time). For each guess, we set up the problem $\min_{\text{rank-}k\ U} \|U(D_1A) - A\|_1$. If D_2 is a sampling and rescaling matrix according to the Lewis weights of D_1A, then by a similar triangle inequality argument as for our CUR decomposition, minimizing $\|U(D_1A)D_2 - AD_2\|_1$ gives an $O(\sqrt{k \log k})$ approximation. By switching to ℓ_2, this implies there is an $O(k \log k)$-approximation of the form AD_2WD_1A, where W is an $O(k \log^2 k) \times O(k \log k)$ matrix of rank k. By setting up the problem $\min_{\text{rank-}k\ W} \|AD_2WD_1A - A\|_1$, one can sample from Lewis weights on the left and right to reduce this to a problem independent of n and d, after which one can use polynomial optimization to solve it in $\exp(\text{poly}(k))$ time. One of our guesses D_1A will be correct, and for this guess we obtain an $\widetilde{O}(k)$-approximation. For each guess we can compute its cost and take the best one found. This gives an $O(1)$-approximation for constant k, removing the $O(\log d)$-factor from the approximation of our earlier algorithm.

Existential Results for Subset Selection: In our algorithm for small k, the first step was to show there exist $O(k \log k)$ rows of A which contain a rank-k space which is an $O(\sqrt{k \log k})$-approximation.

While for Frobenius norm one can find $O(k)$ rows with an $O(1)$-approximation in their span, one of our main negative results for ℓ_1-low rank approximation is that this is impossible, showing that the best approximation one can obtain with $\text{poly}(k)$ rows is $k^{1/2-\gamma}$ for an arbitrarily small constant $\gamma > 0$. Our hard instance is an $r \times (r + k)$ matrix A in which the first k columns are i.i.d. Gaussian, and the remaining r columns are an identity

matrix. Here, r can be twice the number of rows one is choosing. The optimal ℓ_1-low rank approximation has cost at most r, obtained by choosing the first k columns.

Let $R \in \mathbb{R}^{r/2 \times k}$ denote the first k entries of the $r/2$ chosen rows, and let y denote the first k entries of an unchosen row. For $r/2 > k$, there exist many solutions $x \in \mathbb{R}^{r/2}$ for which $x^\top R = y$. However, we can show the following tradeoff:

$$\textit{whenever } \|x^\top R - y\|_1 < \tfrac{\sqrt{k}}{\mathrm{poly}(\log k)}, \textit{ then } \|x\|_1 > \tfrac{\sqrt{k}}{\mathrm{poly}(\log k)}.$$

Then no matter which linear combination $x^\top$ of the rows of R one chooses to approximate y by, either one incurs a $\frac{\sqrt{k}}{\mathrm{poly}(\log k)}$ cost on the first k coordinates, or since A contains an identity matrix, one incurs cost $\|x\|_1 > \frac{\sqrt{k}}{\mathrm{poly}(\log k)}$ on the last r coordinates of $x^\top R$.

To show the tradeoff, consider an $x \in \mathbb{R}^{r/2}$. We decompose $x = x^0 + \sum_{j \geq 1} x^j$, where x^j agrees with x on coordinates which have absolute value in the range $\frac{1}{\sqrt{k}\log^c k} \cdot [2^{-j}, 2^{-j+1}]$, and is zero otherwise. Here, $c > 0$ is a constant, and x^0 denotes the restriction of x to all coordinates of absolute value at least $\frac{1}{\sqrt{k}\log^c k}$. Then $\|x\|_1 < \frac{\sqrt{k}}{\log^c k}$, as otherwise we are done. Hence, x^0 *has small support.* Thus, one can build a small net for all x^0 vectors by choosing the support, then placing a net on it. For x^j for $j > 0$, the support sizes are increasing so the net size needed for all x^j vectors is larger. However, since x^j has all coordinates of roughly the same magnitude on its support, its ℓ_2*-norm is decreasing in* j. Since $(x^j)^\top R \sim N(0, \|x^j\|_2^2 I_k)$, this makes it much less likely that individual coordinates of $(x^j)^\top R$ can be large. Since this probability goes down rapidly, we can afford to union bound over the larger net size. What we show is that for any sum of the form $\sum_{j \geq 1} x^j$, at most $\frac{k}{10}$ of its coordinates are at least $\frac{1}{\log k}$ in magnitude.

For $\|x^\top R - y\|_1$ to be at most $\frac{\sqrt{k}}{\log^c k}$, for at least $\frac{k}{2}$ coordinates i, we must have $|(x^\top R - y)_i| < \frac{2}{\sqrt{k}\log^c k}$. With probability $1 - 2^{-\Omega(k)}$, $|y_i| \geq \frac{1}{100}$ on at least $\frac{2k}{3}$ coordinates. From the previous paragraph, it follows there are at least $\frac{k}{2} - \frac{k}{10} - \frac{k}{3} = \Omega(k)$ coordinates i of x for which (1) $|(x^\top R - y)_i| < \frac{2}{\sqrt{k}\log^c k}$, (2) $|\sum_{j\geq 1} x_i^j| < \frac{1}{\log k}$, and (3) $|y_i| \geq \frac{1}{100}$. On these i, $(x^0)^\top R_i$ must be in an interval of width $\frac{1}{\log k}$ at distance at least $\frac{1}{100}$ from the origin. Since $(x^0)^\top R \sim N(0, \|x^0\|_2^2 I_k)$, for any value of $\|x^0\|_2^2$ the probability this happens on $\Omega(k)$ coordinates is at most $2^{-\Theta(k)}$. Since the net size for x^0 is small, we can union bound over every sequence $x^0, x^1, \ldots,$ coming from our nets.

Some care is needed to union bound over all possible subsets R of rows which can be chosen. We handle this by conditioning on a few events of A itself, which imply corresponding events *for every subset of rows*. These events are such that if R is the chosen set of half the rows, and S the remaining set of rows of A, then the event that a constant fraction of rows in S are close to the row span of R is $2^{-\Theta(kr)}$, which is small enough to union bound over all choices of R.

Curiously, we also show there are some matrices $A \in \mathbb{R}^{n\times d}$ for which any ℓ_1 rank-k approximation in the entire row span of A cannot achieve better than a $(2 - \Theta(1/d))$-approximation.

Bicriteria Algorithm: Our algorithm for small k gives an $O(1)$-approximation in $\mathrm{poly}(n)$ time for constant k, but the approximation factor depends on k. We show how one can find a rank-$2k$ matrix $\widehat{A}$ for which $\|A - \widehat{A}\|_1 \leq C \cdot \mathrm{OPT}$, where C is an absolute constant, and $\mathrm{OPT} = \min_{\text{rank-}k \text{ matrices } A'} \|A - A'\|_1$. We first find a rank-$k$ matrix B^1 for which $\|A - B^1\|_1 \leq p \cdot \mathrm{OPT}$ for a factor $1 \leq p \leq \mathrm{poly}(n)$. We can use any of our algorithms above

for this.

Next consider the problem $\min_{V \in \mathbb{R}^{2k \times d}} \|U^*V - (A - B^1)\|_1$, and let U^*V^* be a best ℓ_1-low rank-$2k$ approximation to $A - B^1$; we later explain why we look at this problem. We can assume V^* is an ℓ_1 *well-conditioned basis* [Cla05, DDH+09], since we can replace U^* with U^*R^{-1} and V^* with RV^* for any invertible linear transformation R. For any vector x we then have $\frac{\|x\|_1}{f} \leq \|x^\top V^*\|_1 \leq e\|x\|_1$, where $1 \leq e, f \leq \text{poly}(k)$. This implies all entries of U^* are at most $2f\|A - B\|_1$, as otherwise one could replace U^* with $0^{n \times 2k}$ and reduce the cost. Also, any entry of U^* smaller than $\frac{\|A-B\|_1}{100enkp}$ can be replaced with 0 as this incurs additive error $\frac{\text{OPT}}{100}$. If we round the entries of U^* to integer multiples of $\frac{\|A-B\|_1}{100enkp}$, then we only have $O(enkpf)$ possibilities for each entry of U^*, and still obtain an $O(1)$-approximation. We refer to the rounded U^* as U^*, abusing notation.

Let D be a sampling and rescaling matrix with $O(k \log k)$ non-zero diagonal entries, corresponding to sampling by the Lewis weights of U^*. We do not know D, but handle this below. By the triangle inequality, for any V,

$$\begin{aligned} \|D(U^*V - (A - B^1))\|_1 &= \|D(U^*V - U^*V^*)\|_1 \pm \|D(U^*V^* - (A - B^1))\|_1 \\ &= \Theta(1)\|U^*V - U^*V^*\|_1 \pm O(1)\|U^*V^* - (A - B^1)\|_1, \end{aligned}$$

where the Lewis weights give $\|D(U^*V - U^*V^*)\|_1 = \Theta(1)\|U^*V - U^*V^*\|_1$ and a Markov bound gives $\|D(U^*V^* - (A - B^1))\|_1 = O(1)\|U^*V^* - (A - B^1)\|_1$. Thus, minimizing $\|DU^*V - D(A - B^1)\|_1$ gives a fixed constant factor approximation to the problem $\min_{V \in \mathbb{R}^{2k \times d}} \|U^*V - (A - B^1)\|_1$. The non-zero diagonal entries of D can be assumed to be integers between 1 and n^2.

We guess the entries of DU^* and note for each entry there are only $O(enkpf \log(n^2))$

possibilities. One of our guesses corresponds to Lewis weight sampling by U^*. We solve for V and by the guarantees of Lewis weights, the row span of this V provides an $O(1)$-approximation. We can find the corresponding U via linear programming. As mentioned above, we do not know D, but can enumerate over all D and all possible DU^*. The total time is $n^{\text{poly}(k)}$.

After finding U, which has $2k$ columns, we output the rank-$3k$ space formed by the column span of $[U, B^1]$. By including the column span of B^1, we ensure our original transformation of the problem $\min_{V\in\mathbb{R}^{k\times d}} \|U^*\cdot V - A\|_1$ to the problem $\min_{V\in\mathbb{R}^{2k\times d}} \|U^*\cdot V - (A - B^1)\|_1$ is valid, since we can first use the column span of B^1 to replace A with $A - B^1$. Replacing A with $A - B^1$ ultimately results in a rank-$3k$ output. Had we used A instead of $A - B^1$ our output would have been rank k but would have additive error $\frac{\|A\|_1}{\text{poly}(k/\epsilon)}$. If we assume the entries of A are in $\{-b, -b+1, \ldots, b\}$, then we can lower bound the cost $\|U^*V - A\|_1$, given that it is non-zero, by $(ndb)^{-O(k)}$ (if it is zero then we output A) using Lemma 4.1 in [CW09] and relating entrywise ℓ_1-norm to Frobenius norm. We can go through the same arguments above with $A - B$ replaced by A and our running time will now be $(ndb)^{\text{poly}(k)}$.

Hard Instances for Cauchy Matrices and More General Sketches: We consider a $d \times d$ matrix $A = I_d + (\log d)e_1^\top e$, where $e_1 = (1, 0, \ldots, 0)$ and $e = (1, 1, \ldots, 1)$ and I_d is the $d \times d$ identity. For an $O(k \log k) \times d$ matrix S of i.i.d. Cauchy random variables, $SA = S + (\log d)S_1^\top e$, where S_1 is the first column of S. For a typical column of SA, all entries are at most $\text{poly}(k) \log d$ in magnitude. Thus, in order to approximate the first row of A, which is $(\log d)e$, by $x^\top SA$ for an $x \in \mathbb{R}^{k \log k}$, we need $\|x\|_1 \geq \frac{1}{\text{poly}(k)}$. Also $\|x^\top S\|_1 = \Omega(\|x\|_1 d \log d)$ with $1 - \exp(-k \log k)$ probability, for d large enough, so by a net

argument $\|x\|_1 \leq \text{poly}(k)$ for all x.

However, *there are* entries of SA that are very large, i.e., about one which is $r = \Theta(dk \log k)$ in magnitude, and in general about 2^i entries about $r2^{-i}$ in magnitude. These entries typically occur in columns C_j of SA for which all other entries in the column are bounded by $\text{poly}(k)$ in magnitude. Thus, $|x^\top C_j| \approx r2^{-i}$ for about 2^i columns j. For each such column, if $r2^{-i} \gg \log d$, then we incur cost $\frac{r2^{-i}}{\text{poly}(k)}$ in approximating the first row of A. In total the cost is $\frac{r \log r}{\text{poly}(k)} = \frac{d \log d}{\text{poly}(k)}$, but the optimal cost is at most d, giving a $\frac{\log d}{\text{poly}(k)}$ lower bound. We optimize this to a $\frac{\log d}{k \log^2 k}$ lower bound.

When k is large this bound deteriorates, but we also show a $k^{1/2-\gamma}$ lower bound for arbitrarily small constant $\gamma > 0$. This bound applies to any oblivious sketching matrix. The idea is similar to our row subset selection lower bound. Let A be as in our row subset selection lower bound, consider SA, and write $S = U\Sigma V^\top$ in its full SVD. Then SA is in the row span of the top $O(k \log k)$ rows of $V^\top A$, since Σ only has $O(k \log k)$ non-zero singular values. Since the first k columns of A are rotationally invariant, $V^\top A$ has first k columns i.i.d. Gaussian and remaining columns equal to $V^\top$. Call the first $O(k \log k)$ rows of $V^\top A$ the matrix B. We now try to approximate a row of A by a vector in the row span of B. There are two issues that make this setting different from row subset selection: (1) B no longer contains an identity submatrix, and (2) the rows of B depend on the rows of A. We handle the first issue by building nets for subsets of coordinates of $x^\top V^\top$ rather than x as before; since $\|x^\top V^\top\|_2 = \|x\|_2$ similar arguments can be applied. We handle the second issue by observing that if the number of rows of B is considerably smaller than that of A, then the distribution of B had we replaced a random row of A with zeros would be statistically close to i.i.d. Gaussian. Hence, typical rows of A can be regarded as being independent of

B.

17.1.3 Several Theorem Statements, an Algorithm, and a Roadmap

Algorithm 17.1 Main Meta-Algorithm

procedure L1LowRankApprox(A, n, d, k) ▷ Theorem 17.1.1
  Choose sketching matrix S (a Cauchy matrix or a sparse Cauchy matrix.)
  Compute SA, form C by $C^i \leftarrow \arg\min_x \|xSA - A^i\|_1$. Form $B = C \cdot SA$.
  Choose sketching matrices T_1, R, D, T_2 (Cauchy matrices or sparse Cauchy matrices.)
  Solve $\min_{X,Y} \|T_1 BRXYDBT_2 - T_1 BT_2\|_F$.
  return BRX,YDB.
end procedure

Theorem 17.1.1 (Informal Version of Theorem 17.4.6)**.** *Given $A \in \mathbb{R}^{n\times d}$, there is an algorithm which in $\mathrm{nnz}(A) + (n+d)\cdot \mathrm{poly}(k)$ time, outputs a (factorization of a) rank-k matrix A' such that with probability $9/10$, $\|A' - A\|_1 \leq (\log d)\,\mathrm{poly}(k) \min_{U\in\mathbb{R}^{n\times k}, V\in\mathbb{R}^{k\times d}} \|UV - A\|_1$.*

Theorem 17.1.2 (Informal Version of Theorem 17.4.7)**.** *Given $A \in \mathbb{R}^{n\times d}$, there is an algorithm that takes $\mathrm{poly}(n) d^{\widetilde{O}(k)} 2^{\widetilde{O}(k^2)}$ time and outputs a rank-k matrix A' such that, with probability $9/10$, $\|A' - A\|_1 \leq \widetilde{O}(k) \min_{U\in\mathbb{R}^{n\times k}, V\in\mathbb{R}^{k\times d}} \|UV - A\|_1$. In addition, A' is a CUR decomposition.*

Theorem 17.1.3 (Informal Version of Theorem 17.6.26)**.** *For any $k \geq 1$, and any constant $c \geq 1$, let $n = k^c$. There exists a matrix A such that for any matrix A' in the span of $n/2$ rows of A, $\|A' - A\|_1 = \Omega(k^{0.5-\alpha}) \min_{U\in\mathbb{R}^{n\times k}, V\in\mathbb{R}^{k\times d}} \|UV - A\|_1$, where $\alpha > 0$ is an arbitrarily small constant.*

Road map Section 17.2 introduces some notation and definitions. Section 17.3 includes several useful tools. We provide several ℓ_1-low rank approximation algorithms in Section 17.4. Section 17.5 contains the no contraction and no dilation analysis for our main algorithm.

We provide our existential hardness results for Cauchy matrices, row subset selection and oblivious subspace embeddings in Section 17.6.

17.2 Notation

Let $\mathbb{N}_+$ denote the set of positive integers. For any $n \in \mathbb{N}_+$, let $[n]$ denote the set $\{1, 2, \cdots, n\}$. For any $p \in [1, 2]$, the ℓ_p-norm of a vector $x \in \mathbb{R}^d$ is defined as

$$\|x\|_p = \big(\sum_{i=1}^{d} |x_i|^p\big)^{1/p}.$$

For any $p \in [1, 2)$, the ℓ_p-norm of a matrix $A \in \mathbb{R}^{n \times d}$ is defined as

$$\|A\|_p = \big(\sum_{i=1}^{n} \sum_{j=1}^{d} |A_{ij}|^p\big)^{1/p}.$$

Let $\|A\|_F$ denote the Frobenius norm of matrix A. Let $\mathrm{nnz}(A)$ denote the number of nonzero entries of A. Let $\det(A)$ denote the determinant of a square matrix A. Let $A^\top$ denote the transpose of A. Let $A^\dagger$ denote the Moore-Penrose pseudoinverse of A. Let A^{-1} denote the inverse of a full rank square matrix. We use A_j to denote the j^{th} column of A, and A^i to denote the i^{th} row of A. For an $n \times d$ matrix A, for S a subset of $[n]$ and T a subset of $[d]$, we let A^S denote the $|S| \times d$ submatrix of A with rows indexed by S, while A_T denotes the $n \times |T|$ submatrix of A with columns indexed by T, and A_T^S denote the $|S| \times |T|$ submatrix A with rows in S and columns in T.

For any function f, we define $\widetilde{O}(f)$ to be $f \cdot \log^{O(1)}(f)$. In addition to $O(\cdot)$ notation, for two functions f, g, we use the shorthand $f \lesssim g$ (resp. $\gtrsim$) to indicate that $f \leq Cg$ (resp. $\geq$) for an absolute constant C. We use $f \eqsim g$ to mean $cf \leq g \leq Cf$ for constants c, C. We use OPT to denote $\min_{\mathrm{rank}-k\ A_k} \|A_k - A\|_1$, unless otherwise specified.

17.3 Preliminaries

17.3.1 Polynomial system verifier

Renegar [Ren92a, Ren92b] and Basu *et al.* [BPR96] independently provided an algorithm for the decision problem for the existential theory of the reals, which is to decide the truth or falsity of a sentence $(x_1, \cdots, x_v)F(f_1, \cdots, f_m)$ where F is a quantifier-free Boolean formula with atoms of the form $\text{sign}(f_i) = \sigma$ with $\sigma \in \{0, 1, -1\}$. Note that this problem is equivalent to deciding if a given semi-algebraic set is empty or not. Here we formally state that theorem. For a full discussion of algorithms in real algebraic geometry, we refer the reader to [BPR05] and [Bas14].

Theorem 17.3.1 (Decision Problem [Ren92a, Ren92b, BPR96])**.** *Given a real polynomial system $P(x_1, x_2, \cdots, x_v)$ having v variables and m polynomial constraints*

$$f_i(x_1, x_2, \cdots, x_v)\Delta_i 0, \forall i \in [m],$$

where Δ_i is any of the "standard relations": $\{>, \geq, =, \neq, \leq, <\}$, let d denote the maximum degree of all the polynomial constraints and let H denote the maximum bitsize of the coefficients of all the polynomial constraints. Then in

$$(md)^{O(v)}\,\text{poly}(H),$$

time one can determine if there exists a solution to the polynomial system P.

Recently, this technique has been used to solve a number of low-rank approximation and matrix factorization problems [AGKM12, Mad13, CW15a, BDL16, RSW16].

17.3.2 Cauchy and p-stable transform

Definition 17.3.1 (Dense Cauchy transform)**.** Let $S = \sigma \cdot C \in \mathbb{R}^{m \times n}$ where σ is a scalar, and each entry of $C \in \mathbb{R}^{m \times n}$ is chosen independently from the standard Cauchy distribution. For any matrix $A \in \mathbb{R}^{n \times d}$, SA can be computed in $O(m \cdot \text{nnz}(A))$ time.

Definition 17.3.2 (Sparse Cauchy transform)**.** Let $\Pi = \sigma \cdot SC \in \mathbb{R}^{m \times n}$, where σ is a scalar, $S \in \mathbb{R}^{m \times n}$ has each column chosen independently and uniformly from the m standard basis vectors of $\mathbb{R}^m$, and $C \in \mathbb{R}^{n \times n}$ is a diagonal matrix with diagonals chosen independently from the standard Cauchy distribution. For any matrix $A \in \mathbb{R}^{n \times d}$, ΠA can be computed in $O(\text{nnz}(A))$ time.

Definition 17.3.3 (Dense p-stable transform)**.** Let $p \in (1, 2)$. Let $S = \sigma \cdot C \in \mathbb{R}^{m \times n}$ where σ is a scalar, and each entry of $C \in \mathbb{R}^{m \times n}$ is chosen independently from the standard p-stable distribution. For any matrix $A \in \mathbb{R}^{n \times d}$, SA can be computed in $O(m \,\text{nnz}(A))$ time.

Definition 17.3.4 (Sparse p-stable transform)**.** Let $p \in (1, 2)$. Let $\Pi = \sigma \cdot SC \in \mathbb{R}^{m \times n}$, where σ is a scalar, $S \in \mathbb{R}^{m \times n}$ has each column chosen independently and uniformly from the m standard basis vectors of $\mathbb{R}^m$, and $C \in \mathbb{R}^{n \times n}$ is a diagonal matrix with diagonals chosen independently from the standard p-stable distribution. For any matrix $A \in \mathbb{R}^{n \times d}$, ΠA can be computed in $O(\text{nnz}(A))$ time.

17.3.3 Lewis weights

We follow the exposition of Lewis weights from [CP15].

Definition 17.3.5. For a matrix A, let a_i denote i^{th} row of A, and $a_i (= (A^i)^\top)$ is a column vector. The statistical leverage score of a row a_i is

$$\tau_i(A) \stackrel{\text{def}}{=} a_i^\top (A^\top A)^{-1} a_i = \|(A^\top A)^{-1/2} a_i\|_2^2.$$

For a matrix A and norm p, the ℓ_p Lewis weights w are the unique weights such that for each row i we have

$$w_i = \tau_i(W^{1/2-1/p} A).$$

or equivalently

$$a_i^\top (A^\top W^{1-2/p} A)^{-1} a_i = w_i^{2/p}.$$

Lemma 17.3.2 (Lemma 2.4 of [CP15] and Lemma 7 of [CLM+15]). *Given a matrix $A \in \mathbb{R}^{n \times d}$, $n \geq d$, for any constant $C > 0, 4 > p \geq 1$, there is an algorithm which can compute C-approximate ℓ_p Lewis weights for every row i of A in $O((nnz(A) + d^\omega \log d) \log n)$ time, where $\omega < 2.373$ is the matrix multiplication exponent[Str69, CW87, Wil12].*

Lemma 17.3.3 (Theorem 7.1 of [CP15]). *Given matrix $A \in \mathbb{R}^{n \times d}$ ($n \geq d$) with ℓ_p ($4 > p \geq 1$) Lewis weights w, for any set of sampling probabilities p_i, $\sum_i p_i = N$,*

$$p_i \geq f(d, p) w_i,$$

if $S \in \mathbb{R}^{N \times n}$ has each row chosen independently as the i^{th} standard basis vector, times $1/p_i^{1/p}$, with probability p_i/N, then with probability at least 0.999,

$$\forall x \in \mathbb{R}^d, \frac{1}{2}\|Ax\|_p^p \leq \|SAx\|_p^p \leq 2\|Ax\|_p^p$$

Furthermore, if $p = 1$, $N = O(d \log d)$. *If* $1 < p < 2$, $N = O(d \log d \log \log d)$. *If* $2 \leq p < 4$, $N = O(d^{p/2} \log d)$.

Given a matrix $A \in \mathbb{R}^{n \times d}$ $(n \geq d)$, by Lemma 17.3.3 and Lemma 17.3.2, we are able to compute a sampling/rescaling matrix S in $O((nnz(A) + d^{\omega} \log d) \log n)$ with $\widetilde{O}(d)$ nonzero entries such that

$$\forall x \in \mathbb{R}^d, \frac{1}{2}\|Ax\|_p^p \leq \|SAx\|_p^p \leq 2\|Ax\|_p^p.$$

Sometimes, $\mathrm{poly}(d)$ is much smaller than $\log n$. In this case, we are able to compute the such sampling/rescaling matrix S in $n\,\mathrm{poly}(d)$ time in the following way: basically we can run one of the input sparsity ℓ_p embedding algorithm (see e.g. [MM13]) to compute a well conditioned basis U of column span of A in $n\,\mathrm{poly}(d)$ time. By sampling according to the well conditioned basis (see e.g. [Cla05, DDH+09, Woo14b]), we can compute a sampling/rescaling matrix S_1 such that $(1-\epsilon)\|Ax\|_p^p \leq \|S_1 Ax\|_p^p \leq (1+\epsilon)\|Ax\|_p^p$ where $\epsilon \in (0,1)$ is an arbitrary constant. Notice that S_1 has $\mathrm{poly}(d)$ nonzero entries, thus $S_1 A$ has size $\mathrm{poly}(d)$. Now, we apply Lewis weights sampling according to $S_1 A$, we can get a sampling/rescaling matrix S such that

$$\forall x \in \mathbb{R}^d, (1 - \frac{1}{3})\|S_1 Ax\|_p^p \leq \|SS_1 Ax\|_p^p \leq (1 + \frac{1}{3})\|S_1 Ax\|_p^p.$$

It means that

$$\forall x \in \mathbb{R}^d, \frac{1}{2}\|Ax\|_p^p \leq \|SS_1 Ax\|_p^p \leq 2\|Ax\|_p^p.$$

Note that SS_1 is still a sampling/rescaling matrix according to A, and the number of non-zero entries is $\widetilde{O}(d)$. And the total running time is thus $n\,\mathrm{poly}(d)$.

17.3.4 Frobenius norm and ℓ_2 relaxation

Theorem 17.3.4 (Generalized rank-constrained matrix approximations, Theorem 2 in [FT07])**.** *Given matrices $A \in \mathbb{R}^{n\times d}$, $B \in \mathbb{R}^{n\times p}$, and $C \in \mathbb{R}^{q\times d}$, let the SVD of B be $B = U_B\Sigma_B V_B^\top$ and the SVD of C be $C = U_C\Sigma_C V_C^\top$. Then,*

$$B^\dagger(U_B U_B^\top A V_C C_C^\top)_k C^\dagger = \underset{\text{rank}-k\ X\in\mathbb{R}^{p\times q}}{\arg\min} \|A - BXC\|_F,$$

where $(U_B U_B^\top A V_C V_C^\top)_k \in \mathbb{R}^{p\times q}$ is of rank at most k and denotes the best rank-k approximation to $U_B U_B^\top A V_C V_C^\top \in \mathbb{R}^{p\times d}$ in Frobenius norm.

Claim 17.3.5 (ℓ_2 relaxation of ℓ_p-regression)**.** *Let $p \in [1, 2)$. For any $A \in \mathbb{R}^{n\times d}$ and $b \in \mathbb{R}^n$, define $x^* = \arg\min_{x\in\mathbb{R}^d} \|Ax - b\|_p$ and $x' = \arg\min_{x\in\mathbb{R}^d} \|Ax - b\|_2$. Then,*

$$\|Ax^* - b\|_p \le \|Ax' - b\|_p \le n^{1/p-1/2} \cdot \|Ax^* - b\|_p.$$

Proof. The lower bound trivially holds by definition; we will focus on proving the upper bound. Because $Ax - b$ is an n-dimensional vector, $\forall x$,

$$\frac{1}{n^{1/p-1/2}} \|Ax - b\|_p \le \|Ax - b\|_2 \le \|Ax - b\|_p. \tag{17.1}$$

Then,

$$\begin{aligned}
&\|Ax' - b\|_p \\
&\le \sqrt{n}\|Ax' - b\|_2 && \text{by LHS of Equation (17.1)} \\
&\le \sqrt{n}\|Ax^* - b\|_2 && \text{by } x' = \arg\min_x \|Ax - b\|_2 \\
&\le \sqrt{n}\|Ax^* - b\|_p && \text{by RHS of Equation (17.1)}
\end{aligned}$$

This completes the proof. □

Claim 17.3.6 (Frobenius norm relaxation of ℓ_p-low rank approximation)**.** *Let $p \in [1, 2)$ and for any matrix $A \in \mathbb{R}^{n\times d}$, define $A^* = \arg\min_{\text{rank}-k\ B\in\mathbb{R}^{n\times d}} \|B - A\|_p$ and $A' = \arg\min_{\text{rank}-k\ B\in\mathbb{R}^{n\times d}} \|B - A\|_F$. Then*

$$\|A^* - A\|_p \le \|A' - A\|_p \le (nd)^{1/p-1/2}\|A^* - A\|_p. \tag{17.2}$$

Proof. The lower bound of $\|A' - A\|_p$ trivially holds by definition. We show an upper bound of $\|A' - A\|_p$ in the rest of the proof. For any $A' - A \in \mathbb{R}^{n\times d}$, we have

$$\frac{1}{(nd)^{1/p-1/2}}\|A' - A\|_p \le \|A' - A\|_F \le \|A' - A\|_p. \tag{17.3}$$

Then,

$$\begin{aligned}
&\|A' - A\|_p \\
&\le (nd)^{1/p-1/2}\|A' - A\|_F && \text{by LHS of Equation (17.3)} \\
&\le (nd)^{1/p-1/2}\|A^* - A\|_F && \text{by } A' = \arg\min_{\text{rank}-k\ B}\|B - A\|_p \\
&\le (nd)^{1/p-1/2}\|A^* - A\|_p. && \text{by RHS of Equation (17.3)}
\end{aligned}$$

□

17.3.5 Converting entry-wise ℓ_1 and ℓ_p objective functions into polynomials

Claim 17.3.7 (Converting absolute value constraints into variables)**.** *Given m polynomials $f_1(x), f_2(x), \cdots, f_m(x)$ where $x \in \mathbb{R}^v$, solving the problem*

$$\min_{x \in \mathbb{R}^v} \sum_{i=1}^{m} |f_i(x)|, \tag{17.4}$$

is equivalent to solving another minimization problem with $O(m)$ extra constraints and m extra variables,

$$\begin{aligned} \min_{x \in \mathbb{R}^v, \sigma \in \mathbb{R}^m} & \sum_{i=1}^{m} \sigma_i f_i(x) \\ \text{s.t. } & \sigma_i^2 = 1, \forall i \in [m] \\ & f_i(x)\sigma_i \geq 0, \forall i \in [m]. \end{aligned}$$

Claim 17.3.8. *(Handling ℓ_p) Given m polynomials $f_1(x), f_2(x), \cdots, f_m(x)$ where $x \in \mathbb{R}^v$ and $p = a/b$ for positive integers a and b, solving the problem*

$$\min_{x \in \mathbb{R}^v} \sum_{i=1}^{m} |f_i(x)|^p, \tag{17.5}$$

is equivalent to solving another minimization problem with $O(m)$ extra constraints and $O(m)$ extra variables,

$$\begin{aligned} \min_{x \in \mathbb{R}^v, \sigma \in \mathbb{R}^m} & \sum_{i=1}^{m} y_i \\ \text{s.t. } & \sigma_i^2 = 1, \forall i \in [m] \\ & f_i(x)\sigma_i \geq 0, \forall i \in [m] \\ & (\sigma_i f_i(x))^a = y_i^b, \forall i \in [m] \\ & y_i \geq 0, \forall i \in [m]. \end{aligned}$$

17.3.6 Converting entry-wise ℓ_1 objective function into a linear program

Claim 17.3.9. *Given any matrix $A \in \mathbb{R}^{n \times d}$ and matrix $B \in \mathbb{R}^{k \times d}$, the problem $\min_{U \in \mathbb{R}^{n \times k}} \|UB - A\|_1$ can be solved by solving the following linear program,*

$$\begin{aligned}
&\min_{U \in \mathbb{R}^{n \times k}, x \in \mathbb{R}^{n \times d}} \sum_{i=1}^{n} \sum_{j=1}^{m} x_{i,j} \\
&U_i B^j - A_{i,j} \leq x_{i,j}, \forall i \in [n], j \in [d] \\
&U_i B^j - A_{i,j} \geq -x_{i,j}, \forall i \in [n], j \in [d] \\
&x_{i,j} \geq 0, \forall i \in [n], j \in [d],
\end{aligned}$$

where the number of constraints is $O(nd)$ and the number of variables is $O(nd)$.

17.4 ℓ_1-Low Rank Approximation

This section presents our main ℓ_1-low rank approximation algorithms. Section 17.4.1 provides our three existence results. Section 17.4.2 shows an input sparsity algorithm with $\text{poly}(k)\log^2 d \log n$-approximation ratio. Section 17.4.3 improves the approximation ratio to $\text{poly}(k)\log d$. Section 17.4.4 explains how to obtain $\widetilde{O}(k)$ approximation ratio. Section 17.4.5 improves the approximation ratio to $O(1)$ by outputting a rank-$3k$ solution. Section 17.4.6 presents our algorithm for CUR decomposition. Section 17.4.7 includes some useful properties. Our ℓ_1-low rank approximation algorithm for a rank-r (where $k \leq r \leq (n,d)$) matrix is used as a black box (by setting $r = \text{poly}(k)$) in several other algorithms.

17.4.1 Existence results via dense Cauchy transforms, sparse Cauchy transforms, Lewis weights

The goal of this section is to present the existence results in Corollary 17.4.2. We first provide some bicriteria algorithms in Theorem 17.4.1 which can be viewed as a "warmup". Then the proof of our bicriteria algorithm actually implies the existence results.

Theorem 17.4.1. *Given matrix $A \in \mathbb{R}^{n\times d}$, for any $k \geq 1$, there exist bicriteria algorithms with running time T (specified below), which output two matrices $U \in \mathbb{R}^{n\times m}$, $V \in \mathbb{R}^{m\times d}$ such that, with probability* $9/10$*,*

$$\|UV - A\|_1 \leq \alpha \min_{\text{rank}-k\ A_k} \|A_k - A\|_1.$$

(I). Using a dense Cauchy transform,

$T = \text{poly}(n, d, k)$, $m = O(k \log k)$, $\alpha = O(\sqrt{k \log k} \log d)$.

(II). Using a sparse Cauchy transform,

$T = \text{poly}(n, d, k)$,$m = O(k^5 \log^5 k)$, $\alpha = O(k^{4.5} \log^{4.5} k \log d)$.

(III). Sampling by Lewis weights,

$T = (nd)^{\widetilde{O}(k)}$, $m = O(k \log k)$, $\alpha = O(\sqrt{k \log k})$.

The matrices in (I), (II), (III) here, are the same as those in (I), (II), (III), (IV) of Lemma 17.5.5. Thus, they have the properties shown in Section 17.5.2.

Proof. We define

$$\text{OPT} := \min_{\text{rank}-k\ A_k} \|A_k - A\|_1.$$

We define $U^* \in \mathbb{R}^{n \times k}, V^* \in \mathbb{R}^{k \times d}$ to be the optimal solution such that $\|U^* V^* - A\|_1 = \mathrm{OPT}$.

Part (I). Apply the dense Cauchy transform $S \in \mathbb{R}^{m \times n}$ with $m = O(k \log k)$ rows, and $\beta = O(\log d)$.

Part (II). Apply the sparse Cauchy transform $S (= \Pi \in \mathbb{R}^{m \times n})$ with $m = O(k^5 \log^5 k)$ rows, and $\beta = O(\sigma \log d) = O(k^2 \log^2 k \log d)$.

Part (III). Use S $(= D \in \mathbb{R}^{n \times k})$ to denote an $n \times n$ matrix which is a sampling and rescaling diagonal matrix according to the Lewis weights of matrix U^*. It has $m = O(k \log k)$ rows, and $\beta = O(1)$. Sometimes we abuse notation, and should regard D as a matrix which has size $m \times n$, where $m = O(k \log k)$.

We can just replace M in Lemma 17.5.5 with $U^* V^* - A$, replace U in Lemma 17.5.5 with U^*, and replace $c_1 c_2$ with $O(\beta)$. So, we can apply Lemma 17.5.5 for S. Then we can plug it in Lemma 17.5.2, we have: with constant probability, for any $c \geq 1$, for any $V' \in \mathbb{R}^{k \times d}$ which satisfies

$$\|SU^* V' - SA\|_1 \leq c \cdot \min_{V \in \mathbb{R}^{k \times d}} \|SU^* V - SA\|_1, \tag{17.6}$$

it has

$$\|U^* V' - A\|_1 \leq c \cdot O(\beta) \|U^* V^* - A\|_1. \tag{17.7}$$

Define $\widehat{V}_i = \arg\min_{V_i \in \mathbb{R}^k} \|SU^* V_i - SA_i\|_2$ for each $i \in [d]$. By using Claim 17.3.5 with

$n = m$ and $d = k$, it shows

$$\begin{aligned}\|SU^*\widehat{V} - SA\|_1 &= \sum_{i=1}^{d} \|SU^*\widehat{V}_i - SA_i\|_1 \\ &\leq \sum_{i=1}^{d} \sqrt{m}\|SU^*\widetilde{V}_i - SA_i\|_1 \\ &= \sqrt{m} \min_{V \in \mathbb{R}^{k\times d}} \|SU^*V - SA\|_1.\end{aligned}$$

which means $\widehat{V}$ is a $\sqrt{m}$-approximation solution to problem, $\min_{V\in\mathbb{R}^{k\times d}}\|SU^*V - SA\|_1$.

Now, let us look into Equation (17.6) and Equation (17.7), we can obtain that

$$\|U^*\widehat{V} - A\|_1 \leq \sqrt{m}O(\beta)\,\mathrm{OPT}\,.$$

Because $\widehat{V}_i$ is the optimal solution of the ℓ_2 regression problem, we have

$$\widehat{V}_i = (SU^*)^\dagger SA_i \in \mathbb{R}^k, \forall i \in [d], \text{ which means } \widehat{V} = (SU^*)^\dagger SA \in \mathbb{R}^{k\times d}.$$

Plugging $\widehat{V}$ into original problem, we obtain

$$\|U^*(SU^*)^\dagger \cdot SA - A\|_1 \leq \sqrt{m}O(\beta)\,\mathrm{OPT}\,.$$

It means

$$\min_{\mathrm{rank}-k\ X\in\mathbb{R}^{n\times m}} \|XSA - A\|_1 \leq \sqrt{m}O(\beta)\,\mathrm{OPT}\,. \tag{17.8}$$

If we ignore the constraint on the rank of X, we can get a bicriteria solution:

For part (I), notice that X is an $n \times m$ matrix which can be found by using a linear program, because matrices $SA \in \mathbb{R}^{m\times d}$ and $A \in \mathbb{R}^{n\times d}$ are known.

For part (II), notice that X is an $n \times m$ matrix which can be found by using a linear program, because matrices $SA \in \mathbb{R}^{m \times d}$ and $A \in \mathbb{R}^{n \times d}$ are known.

For part (III), notice that X is an $n \times m$ matrix which can be found by using a linear program, when the span of rows of $DA \in \mathbb{R}^{m \times d}$ is known. We assume that D is known in all the above discussions. But D is actually unknown. So we need to try all the possible choices of the row span of DA. Since D samples at most $m = O(k \log k)$ rows of A, then the total number of choices of selecting m rows from n rows is $\binom{n}{m} = n^{O(k \log k)}$. This completes the proof.

□

Equation (17.8) in the proof of our bicriteria solution implies the following result,

Corollary 17.4.2. *Given $A \in \mathbb{R}^{n \times d}$, there exists a* rank-$k$ *matrix $A' \in \mathbb{R}^{n \times d}$ such that $A' \in \text{rowspan}(S'A) \subseteq \text{rowspan}(A)$ and $\|A' - A\|_1 \leq \alpha \cdot \min_{\text{rank}-k\ A_k} \|A - A_k\|_1$, where $S' \in \mathbb{R}^{m \times n}$ is a sketching matrix. If S'*

(I). indicates the dense Cauchy transform, then $\alpha = O(\sqrt{k \log k} \log d)$.

(II). indicates the sparse Cauchy transform, then $\alpha = O(k^{4.5} \log^{4.5} k \log d)$.

(III). indicates sampling by Lewis weights, then $\alpha = O(\sqrt{k \log k})$.

Proof. Define $\text{OPT} = \min_{U \in \mathbb{R}^{n \times k}, V \in \mathbb{R}^{k \times d}} \|UV - A\|_1$.

Proof of (I). Choose S to be a dense Cauchy transform matrix with m rows, then

$$\min_{U \in \mathbb{R}^{n \times k}, Z \in \mathbb{R}^{k \times m}} \|UZSA - A\|_1 \leq O(\sqrt{m} \log d)\, \text{OPT},$$

where $m = O(k \log k)$. Choosing $A' = UZSA$ completes the proof.

Proof of (II). Choose $\Pi = SD \in \mathbb{R}^{m\times n}$ where $S \in \mathbb{R}^{m\times n}$ has each column chosen independently and uniformly from the m standard basis vectors of $\mathbb{R}^m$, and where D is a diagonal matrix with diagonals chosen independently from the standard Cauchy distribution, then

$$\min_{U\in\mathbb{R}^{n\times k}, Z\in\mathbb{R}^{k\times m}} \|UZ\Pi A - A\|_1 \leq O(\sqrt{m}\sigma \log d)\,\mathrm{OPT},$$

where $m = O(k^5 \log^5 k)$ and $\sigma = O(k^2 \log^2 k)$. Choosing $A' = UZ\Pi A$ completes the proof.

Proof of (III).

Choose D to be the sampling and rescaling matrix corresponding to the Lewis weights of U^*, and let it have $m = O(k \log k)$ nonzero entries on the diagonal, then

$$\min_{U\in\mathbb{R}^{n\times k}, Z\in\mathbb{R}^{k\times m}} \|UZDA - A\|_1 \leq O(\sqrt{m})\,\mathrm{OPT}\,.$$

Choosing $A' = UZDA$ completes the proof. □

17.4.2 Input sparsity time, $\mathrm{poly}(k, \log n, \log d)$-approximation for an arbitrary matrix A

The algorithm described in this section is actually worse than the algorithm described in the next section. But this algorithm is easy to extend to the distributed and streaming settings (See full version of [SWZ17]).

Algorithm 17.2 Input Sparsity Time Algorithm

procedure L1LowRankApproxInputSparsity(A, n, d, k) ▷ Theorem 17.4.3
  Set $s \leftarrow r \leftarrow t_1 \leftarrow \widetilde{O}(k^5)$, $t_2 \leftarrow \widetilde{O}(k)$.
  Choose sparse Cauchy matrices $S \in \mathbb{R}^{s \times n}$, $R \in \mathbb{R}^{d \times r}$, $T_1 \in \mathbb{R}^{t_1 \times n}$.
  Choose dense Cauchy matrices $T_2 \in \mathbb{R}^{d \times t_2}$.
  Compute $S \cdot A$, $A \cdot R$ and $T_1 \cdot A \cdot T_2$.
  Compute $XY = \arg\min_{X,Y} \|T_1 ARXYSAT_2 - T_1 AT_2\|_F$.
  return ARX, YSA.
end procedure

Theorem 17.4.3. *Given matrix $A \in \mathbb{R}^{n \times d}$, for any $k \geq 1$, there exists an algorithm which takes $O(\mathrm{nnz}(A)) + (n+d) \cdot \mathrm{poly}(k)$ time and outputs two matrices $U \in \mathbb{R}^{n \times k}$, $V \in \mathbb{R}^{k \times d}$ such that*

$$\|UV - A\|_1 \leq O(\mathrm{poly}(k) \log n \log^2 d) \min_{\mathrm{rank}-k \ A_k} \|A_k - A\|_1$$

holds with probability $9/10$.

Proof. Choose a Cauchy matrix $S \in \mathbb{R}^{s \times n}$ (notice that S can be either a dense Cauchy transform matrix or a sparse Cauchy transform matrix). Using Corollary 17.4.2, we have

$$\min_{U \in \mathbb{R}^{n \times k}, Z \in \mathbb{R}^{k \times s}} \|UZSA - A\|_1 \leq \alpha_s \, \mathrm{OPT},$$

where α_s is the approximation by using matrix S. If S is a dense Cauchy transform matrix, then due to Part (I) of Corollary 17.4.2, $\alpha_s = O(\sqrt{k \log k} \log d)$, $s = O(k \log k)$, and computing SA takes $O(s \operatorname{nnz}(A))$ time. If S is a sparse Cauchy transform matrix, then due to Part II of Corollary 17.4.2, $\alpha_s = \widetilde{O}(k^{4.5} \log d)$, $s = \widetilde{O}(k^5)$, and computing SA takes $\operatorname{nnz}(A)$ time.

We define $U^*, Z^* = \arg\min_{U,Z} \|UZSA - A\|_1$. For the fixed $Z^* \in \mathbb{R}^{k \times s}$, choose a Cauchy matrix $R \in \mathbb{R}^{d \times r}$ (note that R can be either a dense Cauchy transform matrix or a sparse Cauchy transform matrix) and sketch on the right of $(UZSA - A)$. If R is a dense Cauchy transform matrix, then $\alpha_r = O(\log n)$, $r = O(k \log k)$, computing AR takes $O(r \cdot \operatorname{nnz}(A))$ time. If R is a sparse Cauchy transform matrix, then $\alpha_r = \widetilde{O}(k^2) \log n$, $r = \widetilde{O}(k^5)$, computing AR takes $O(\operatorname{nnz}(A))$ time.

Define a row vector $\widehat{U}^j = A^j R((Z^*SA)R)^\dagger \in \mathbb{R}^k$. Then

$$\forall j \in [n], \|\widehat{U}^j Z^* SAR - A^j R\|_2 = \min_{x \in \mathbb{R}^k} \|x^\top Z^* SAR - A^j R\|_2.$$

Recall that r is the number of columns of R. Due to Claim 17.3.5,

$$\sum_{j=1}^n \|A^j R((Z^*SA)R)^\dagger Z^* SAR - A^j R\|_1 \leq O(\sqrt{r}) \sum_{j=1}^n \min_{U^j \in \mathbb{R}^k} \|U^j Z^* SAR - A^j R\|_1,$$

which is equivalent to

$$\|AR((Z^*SA)R)^\dagger Z^* SAR - AR\|_1 \leq O(\sqrt{r}) \min_{U \in \mathbb{R}^{n \times k}} \|UZ^* SAR - AR\|_1,$$

where AR is an $n \times r$ matrix and SA is an $s \times d$ matrix.

Using Lemma 17.5.2, we obtain,

$$\|AR((Z^*SA)R)^\dagger Z^* SA - A\|_1 \leq O(\sqrt{r}\alpha_r) \min_{U \in \mathbb{R}^{n \times k}} \|UZ^* SA - A\|_1.$$

We define $X^* \in \mathbb{R}^{r\times k}$, $Y^* \in \mathbb{R}^{k\times s}$,

$$X^*, Y^* = \arg\min_{X\in\mathbb{R}^{r\times k}, Y\in\mathbb{R}^{k\times s}} \|ARXYSA - A\|_1.$$

Then,

$$\begin{aligned}
\|ARX^*Y^*SA - A\|_1 &\leq \|AR((Z^*SA)R)^\dagger Z^*SA - A\|_1 \\
&\leq O(\sqrt{r}\alpha_r) \min_{U\in\mathbb{R}^{n\times k}} \|UZ^*SA - A\|_1 \\
&= O(\sqrt{r}\alpha_r) \min_{U\in\mathbb{R}^{n\times k}, Z\in\mathbb{R}^{k\times s}} \|UZSA - A\|_1 \\
&\leq O(\sqrt{r}\alpha_r\alpha_s)\,\mathrm{OPT}\,.
\end{aligned}$$

It means that ARX^*, Y^*SA gives an $O(\alpha_r\alpha_s\sqrt{r})$-approximation to the original problem.

Thus it suffices to use Lemma 17.4.4 to solve

$$\min_{X\in\mathbb{R}^{r\times k}, Y\in\mathbb{R}^{k\times s}} \|ARXYSA - A\|_1,$$

by losing an extra $\mathrm{poly}(k)\log d$ factor in the approximation ratio.

By using a sparse Cauchy transform (for the place discussing the two options), combining the approximation ratios and running times all together, we can get $\mathrm{poly}(k)\log(n)\log^2(d)$-approximation ratio with $O(\mathrm{nnz}(A)) + (n+d)\,\mathrm{poly}(k)$ running time. This completes the proof. □

Lemma 17.4.4. *Given matrices $A \in \mathbb{R}^{n\times d}$, $SA \in \mathbb{R}^{s\times n}$,$RA \in \mathbb{R}^{r\times d}$ where $S \in \mathbb{R}^{s\times n}$, $R \in \mathbb{R}^{d\times r}$ with $\min(n,d) \geq \max(r,s)$. For any $1 \leq k \leq \min(r,s)$, there exists an algorithm that takes $O(\mathrm{nnz}(A)) + (n+d)\,\mathrm{poly}(s,r,k)$ time to output two matrices $X' \in \mathbb{R}^{r\times k}, Y' \in \mathbb{R}^{k\times s}$ such that*

$$\|ARX' \cdot Y'SA - A\|_1 \leq \mathrm{poly}(r,s)\log(d) \min_{X\in\mathbb{R}^{r\times k}, Y\in\mathbb{R}^{k\times s}} \|ARXYSA - A\|_1$$

holds with probability at least .999.

Proof. Choose sketching matrices $T_1 \in \mathbb{R}^{t_1 \times n}$ to sketch on the left of $(ARXYSA - A)$ (note that S can be either a dense Cauchy transform matrix or a sparse Cauchy transform matrix). If T_1 is a dense Cauchy transform matrix, then $t_1 = O(r \log r)$, $\alpha_{t_1} = O(\log d)$, and computing $T_1 A$ takes $O(t_1 \cdot \text{nnz}(A))$ time. If T_1 is a sparse Cauchy transform matrix, then $t_1 = \widetilde{O}(r^5)$, $\alpha_{t_1} = \widetilde{O}(r^2) \log d$, and computing $T_1 A$ takes $\text{nnz}(A)$ time.

Choose dense Cauchy matrices $T_2^\top \in \mathbb{R}^{t_2 \times d}$ to sketch on the right of $T_1(ARXYSA - A)$ with $t_2 = O((t_1 + s)\log(t_1 + s))$. We get the following minimization problem,

$$\min_{X \in \mathbb{R}^{r \times k}, Y \in \mathbb{R}^{k \times s}} \|T_1 ARXYSAT_2 - T_1 AT_2\|_1. \tag{17.9}$$

Define X', Y' to be the optimal solution of

$$\min_{X \in \mathbb{R}^{r \times k}, Y \in \mathbb{R}^{k \times s}} \|T_1 ARXYSAT_2 - T_1 AT_2\|_F.$$

Due to Claim 17.3.6,

$$\|T_1 ARX'Y'SAT_2 - T_1 AT_2\|_1 \leq \sqrt{t_1 t_2} \min_{X \in \mathbb{R}^{r \times k}, Y \in \mathbb{R}^{k \times s}} \|T_1 ARXYSAT_2 - T_1 AT_2\|_1.$$

Due to Lemma 17.5.4

$$\|ARX'Y'SA - A\|_1 \leq \sqrt{t_1 t_2}\alpha_{t_1} \log t_1 \min_{X \in \mathbb{R}^{r \times k}, Y \in \mathbb{R}^{k \times s}} \|ARXYSA - A\|_1.$$

It remains to solve

$$\min_{X \in \mathbb{R}^{r \times k}, Y \in \mathbb{R}^{k \times s}} \|T_1 ARXYSAT_2 - T_1 AT_2\|_F.$$

By using Theorem 17.3.4 and choosing T_1 to be a sparse Cauchy transform matrix, we have that the optimal rank-k solution $X'Y'$ is $(T_1AR)^\dagger(U_BU_B^\top(T_1AT_2)V_CV_C^\top)_k(SAT_2)$ which can be computed in $O(\mathrm{nnz}(A)) + (n+d)\,\mathrm{poly}(s,r,k)$ time. Here, U_B are the left singular vectors of T_1AR. V_C are the right singular vectors of SAT_2. □

An alternative way of solving Equation (17.9) is using a polynomial system verifier. Note that a polynomial system verifier does not allow absolute value constraints. Using Claim 17.3.7, we are able to remove these absolute value constraints by introducing new constraints and variables. Thus, we can get a better approximation ratio but by spending exponential running time in k. In the previous step, we should always use a dense Cauchy transform to optimize the approximation ratio.

Corollary 17.4.5. *Given $A \in \mathbb{R}^{n\times d}$, there exists an algorithm which takes $nd \cdot \mathrm{poly}(k) + (n+d)\cdot 2^{\widetilde{O}(k^2)}$ time and outputs two matrices $U \in \mathbb{R}^{n\times k}$, $V \in \mathbb{R}^{k\times d}$ such that*

$$\|UV - A\|_1 \leq O(\mathrm{poly}(k)\log n \log^2 d) \min_{\mathrm{rank}-k\ A_k} \|A_k - A\|_1$$

holds with probability $9/10$.

The $\mathrm{poly}(k)$ factor in the above corollary is much smaller than that in Theorem 17.4.3.

17.4.3 $\mathrm{poly}(k, \log d)$-approximation for an arbitrary matrix A

In this section, we explain how to get an $O(\log d) \cdot \mathrm{poly}(k)$ approximation.

Algorithm 17.3 $\mathrm{poly}(k) \log d$-approximation Algorithm

procedure L1LowRankApproxPolykLogd(A, n, d, k) ▷ Theorem 17.4.6
  Set $s \leftarrow \widetilde{O}(k^5)$.
  Choose sparse Cauchy matrices $S \in \mathbb{R}^{s \times n}$ and compute $S \cdot A$.
  Implicitly obtain $B = U_B V_B$ by finding $V_B = SA \in \mathbb{R}^{s \times d}$ and $U_B \in \mathbb{R}^{n \times s}$ where $\forall i \in [n]$, row vector $(U_B)^i$ gives an $O(1)$ approximation to $\min_{x \in \mathbb{R}^{1 \times s}} \|xSA - A^i\|_1$.
  $U, V \leftarrow$L1LowRankApproxB(U_B, V_B, n, d, k, s). ▷ Theorem 17.4.19
  return U, V.
end procedure

Intuitively, our algorithm has two stages. In the first stage, we just want to find a low rank matrix B which is a good approximation to A. Then, we can try to find a rank-k approximation to B. Since now B is a low rank matrix, it is much easier to find a rank-k approximation to B. The procedure L1LowRankApproxB(U_B, V_B, n, d, k, s) corresponds to Theorem 17.4.19.

Theorem 17.4.6. *Given matrix $A \in \mathbb{R}^{n \times d}$, for any $k \geq 1$, there exists an algorithm which takes $\mathrm{nnz}(A) + (n + d) \cdot \mathrm{poly}(k)$ time to output two matrices $U \in \mathbb{R}^{n \times k}$, $V \in \mathbb{R}^{k \times d}$ such that*

$$\|UV - A\|_1 \leq \mathrm{poly}(k) \log d \min_{\mathrm{rank}-k \ A_k} \|A_k - A\|_1$$

holds with probability $9/10$.

Proof. We define

$$\mathrm{OPT} := \min_{\mathrm{rank}-k \ A_k} \|A_k - A\|_1.$$

The main idea is to replace the given $n \times d$ matrix A with another low rank matrix B which also has size $n \times d$. Choose $S \in \mathbb{R}^{s \times n}$ to be a Cauchy matrix, where $s \leq \text{poly}(k)$ (note that, if S is a dense Cauchy transform matrix, computing SA takes $O(s \,\text{nnz}(A))$ time, while if S is a sparse Cauchy transform matrix, computing SA takes $O(\text{nnz}(A))$ time). Then B is obtained by taking each row of A and replacing it with its closest point (in ℓ_1-distance) in the row span of SA. By using Part II of Corollary 17.4.2, we have,

$$\min_{U \in \mathbb{R}^{n \times k}, Z \in \mathbb{R}^{k \times s}} \|UZSA - A\|_1 \leq O(\sqrt{s}\,\text{poly}(k) \log d)\,\text{OPT}\,.$$

We define B to be the product of two matrices $U_B \in \mathbb{R}^{n \times s}$ and $V_B \in \mathbb{R}^{s \times d}$. We define V_B to be SA and U_B to be such that for any $i \in [n]$, $(U_B)^i$ gives an $O(1)$-approximation to problem $\min_{x \in \mathbb{R}^{1 \times s}} \|xSA - A^i\|_1$, i.e.,

$$\|(U_B)^i SA - A^i\|_1 \leq O(1) \min_{x \in \mathbb{R}^{1 \times s}} \|xSA - A^i\|_1, \forall i \in [n],$$

which means

$$\|U_B SA - A\|_1 \leq O(1) \min_{X \in \mathbb{R}^{n \times s}} \|XSA - A\|_1.$$

For a fixed $SA \in \mathbb{R}^{s \times d}$, we can compute $D \in \mathbb{R}^{d \times d}$, which is a sampling and rescaling matrix corresponding to Lewis weights of $(SA)^\top$, and let $m = O(s \log s)$ be the number of nonzero entries on the diagonal of D.

Define $\widehat{X} = \arg\min_{X \in \mathbb{R}^{n \times s}} \|XSAD - AD\|_1$, thus by Lemma 17.5.5 and Lemma 17.5.2, we have

$$\|\widehat{X}SA - A\|_1 \leq O(1) \min_{X \in \mathbb{R}^{n \times s}} \|XSA - A\|_1.$$

Notice that computing Lewis weights takes $d\,\mathrm{poly}(s)$ time. We can use ℓ_1-regression solver and linear programming to find $\widehat{X} \in \mathbb{R}^{n\times s}$ in $(n+d)\,\mathrm{poly}(s)$ time. Thus U_B can be found in $O(\mathrm{nnz}(A)) + (n+d)\,\mathrm{poly}(s)$ time.

By the definition of B, it is an $n \times d$ matrix. Naïvely we can write down B after finding U_B and V_B. The time for writing down B is $O(nd)$. To avoid this, we can just keep a factorization U_B and V_B. We are still able to run algorithm L1LowRankApproxB. Because $s = \mathrm{poly}(k)$, the running time of algorithm L1LowRankApproxB is still $O(\mathrm{nnz}(A)) + (n+d)\,\mathrm{poly}(k)$.

By the definition of B, we have that B has rank at most s. Suppose we then solve ℓ_1-low rank approximation problem for rank-s matrix B, finding a rank-k g-approximation matrix UV. Due to Lemma 17.4.15, we have that if B is an f-approximation solution to A, then UV is also an $O(fg)$-approximation solution to A,

$$\|UV - A\|_1 \leq O(\log d) \cdot \mathrm{poly}(k) \cdot g\,\mathrm{OPT}\,.$$

Using Theorem 17.4.19 we have that $g = \mathrm{poly}(s)$, which completes the proof.

□

17.4.4 $\widetilde{O}(k)$-approximation for an arbitrary matrix A

Algorithm 17.4 $\widetilde{O}(k)$-approximation Algorithm

procedure L1LowRankApproxK(A, n, d, k) ▷ Theorem 17.4.7
  $r \leftarrow O(k \log k), m \leftarrow t_1 \leftarrow O(r \log r), t_2 \leftarrow O(m \log m)$.
  Guess a diagonal matrix $R \in \mathbb{R}^{d \times d}$ with only r 1s. ▷ R selects r columns of $A \in \mathbb{R}^{n \times d}$.
  Compute a sampling and rescaling matrix $D \in \mathbb{R}^{n \times n}, T_1 \in \mathbb{R}^{n \times n}$ corresponding to the Lewis weights of AR, and let them have m, t_1 nonzero entries on the diagonals, respectively.
  Compute a sampling and rescaling matrix $T_2^\top \in \mathbb{R}^{d \times d}$ according to the Lewis weights of $(DA)^\top$, and let it have t_2 nonzero entries on the diagonal.
  Solve $\min_{X,Y} \|T_1 ARXYDAT_2 - T_1 AT_2\|_1$.
  Take the best solution X, Y over all guesses of R.
  return ARX, YDA.
end procedure

Theorem 17.4.7. *Given matrix $A \in \mathbb{R}^{n \times d}$, there exists an algorithm that takes $\mathrm{poly}(n) \cdot d^{\widetilde{O}(k)} \cdot 2^{\widetilde{O}(k^2)}$ time and outputs two matrices $U \in \mathbb{R}^{n \times k}$, $V \in \mathbb{R}^{k \times d}$ such that*

$$\|UV - A\|_1 \leq \widetilde{O}(k) \min_{\mathrm{rank}-k\ A_k} \|A_k - A\|_1$$

holds with probability $9/10$.

Proof. We define

$$\mathrm{OPT} := \min_{\mathrm{rank}-k\ A_k} \|A_k - A\|_1.$$

Let $U^* \in \mathbb{R}^{n \times k}, V^* \in \mathbb{R}^{k \times d}$ satisfy

$$\|U^* V^* - A\|_1 = \mathrm{OPT}\,.$$

Let $S^\top \in \mathbb{R}^{d\times d}$ denote the sampling and rescaling matrix corresponding to the Lewis weights of $(V^*)^\top$, where the number of nonzero entries on the diagonal of S is $s = r = O(k \log k)$. Let $R^\top \in \mathbb{R}^{d\times d}$ denote a diagonal matrix such that $\forall i \in [d]$, if $S_{i,i} \neq 0$, then $R_{i,i} = 1$, and if $S_{i,i} = 0$, then $R_{i,i} = 0$. Since $\mathrm{rowspan}(R^\top A^\top) = \mathrm{rowspan}(S^\top A^\top)$,

$$\min_{Z\in\mathbb{R}^{m\times k}, V\in\mathbb{R}^{k\times d}} \|ASZV - A\|_1 = \min_{Z\in\mathbb{R}^{m\times k}, V\in\mathbb{R}^{k\times d}} \|ARZV - A\|_1.$$

Combining with Part III of Corollary 17.4.2, there exists a rank-k solution in the column span of AR, which means,

$$\min_{Z\in\mathbb{R}^{m\times k}, V\in\mathbb{R}^{k\times d}} \|ARZV - A\|_1 \leq O(\sqrt{r})\,\mathrm{OPT}\,. \tag{17.10}$$

Because the number of 1s of R is r, and the size of the matrix is $d \times d$, there are $\binom{d}{r} = d^{\widetilde{O}(k)}$ different choices for locations of 1s on the diagonal of R. We cannot compute R directly, but we can guess all the choices of locations of 1s. Regarding R as selecting r columns of A, then there are $d^{\widetilde{O}(k)}$ choices. There must exist a "correct" way of selecting a subset of columns over all all choices. After trying all of them, we will have chosen the right one.

For a fixed guess R, we can compute $D \in \mathbb{R}^{n\times n}$, which is a sampling and rescaling matrix corresponding to the Lewis weights of AR, and let $m = O(k \log^2 k)$ be the number of nonzero entries on the diagonal of D.

By Equation (17.10), there exists a $W \in \mathbb{R}^{r\times k}$ such that,

$$\min_{V\in\mathbb{R}^{k\times d}} \|ARWV - A\|_1 \leq O(\sqrt{r})\,\mathrm{OPT}\,. \tag{17.11}$$

We define $\widehat{V}_i = \arg\min_{V_i\in\mathbb{R}^{k\times d}} \|DARWV_i - DA_i\|_2$, $\forall i \in [d]$, which means $\widehat{V}_i = (DARW)^\dagger DA_i \in \mathbb{R}^k$. Then $\widehat{V} = (DARW)^\dagger DA \in \mathbb{R}^{k\times d}$. We define $V^* = \arg\min_{V\in\mathbb{R}^{k\times d}} \|ARWV - A\|_1$. Then, by

Claim 17.3.5, it has

$$\|DARW\widehat{V} - DA\|_1 \leq O(\sqrt{m}) \min_{V \in \mathbb{R}^{k \times d}} \|DARWV - DA\|_1.$$

By applying Lemma 17.5.5, Lemma 17.5.2 and Equation (17.11), we can show

$$\|ARW\widehat{V} - A\|_1 \leq O(\sqrt{m}) \|ARWV^* - A\|_1 \leq O(\sqrt{mr})\,\mathrm{OPT} \leq \widetilde{O}(k)\,\mathrm{OPT}\,.$$

Plugging $\widehat{V} = (DARW)^\dagger DA$ into $\|ARW\widehat{V} - A\|_1$, we obtain that

$$\|ARW(DARW)^\dagger DA - A\|_1 \leq \widetilde{O}(k)\,\mathrm{OPT}\,.$$

and it is clear that,

$$\min_{X \in \mathbb{R}^{r \times k}, Y \in \mathbb{R}^{k \times m}} \|ARXYDA - A\|_1 \leq \|ARW(DARW)^\dagger DA - A\|_1 \leq \widetilde{O}(k)\,\mathrm{OPT}\,.$$

Recall that we guessed R, so it is known. We can compute $T_1 \in \mathbb{R}^{n \times n}$, which is a sampling and rescaling diagonal matrix corresponding to the Lewis weights of AR, and $t_1 = O(r \log r)$ is the number of nonzero entries on the diagonal of T_1.

Also, DA is known, and the number of nonzero entries in D is $m = O(k \log^2 k)$. We can compute $T_2^\top \in \mathbb{R}^{d \times d}$, which is a sampling and rescaling matrix corresponding to the Lewis weights of $(DA)^\top$, and $t_2 = O(m \log m)$ is the number of nonzero entries on the diagonal of T_2.

Define $X^*, Y^* = \arg\min_{X \in \mathbb{R}^{r \times k}, Y \in \mathbb{R}^{k \times m}} \|T_1(AR)XY(DA)T_2 - T_1AT_2\|_1$. Thus, using Lemma 17.5.4, we have

$$\|(AR)X^*Y^*(DA) - A\|_1 \leq \widetilde{O}(k)\,\mathrm{OPT}\,.$$

To find X^*, Y^*, we need to solve this minimization problem

$$\min_{X \in \mathbb{R}^{r \times k}, Y \in \mathbb{R}^{k \times m}} \|T_1(AR)XY(DA)T_2 - T_1AT_2\|_1,$$

which can be solved by a polynomial system verifier (see more discussion in Section 17.3.1 and 17.3.5).

In the next paragraphs, we explain how to solve the above problem by using a polynomial system verifier. Notice that $T_1(AR)$ is known and $(DA)T_2$ is also known. First, we create $r \times k$ variables for the matrix X, i.e., one variable for each entry of X. Second, we create $k \times m$ variables for matrix Y, i.e., one variable for each entry of Y. Putting it all together and creating $t_1 \times t_2$ variables $\sigma_{i,j}, \forall i \in [t_1], j \in [t_2]$ for handling the unknown signs, we write down the following optimization problem

$$\begin{aligned} \min_{X,Y} \ & \sum_{i=1}^{t_1} \sum_{j=1}^{t_2} \sigma_{i,j} (T_1(AR)XY(DA)T_2)_{i,j} \\ \text{s.t } & \sigma_{i,j}^2 = 1, \forall i \in [t_1], j \in [t_2] \\ & \sigma_{i,j} \cdot (T_1(AR)XY(DA)T_2)_{i,j} \geq 0, \forall i \in [t_1], j \in [t_2]. \end{aligned}$$

Notice that the number of constraints is $O(t_1 t_2) = \widetilde{O}(k^2)$, the maximum degree is $O(1)$, and the number of variables $O(t_1 t_2 + km + rk) = \widetilde{O}(k^2)$. Thus the running time is,

$$(\#\,\text{constraints} \cdot \#\text{degree})^{O(\#\,\text{variables})} = 2^{\widetilde{O}(k^2)}.$$

To use a polynomial system verifier, we need to discuss the bit complexity. Suppose that all entries are multiples of δ, and the maximum is Δ, i.e., each entry

$$\in \{-\Delta, \cdots, -2\delta, -\delta, 0, \delta, 2\delta, \cdots, \Delta\},$$

and $\Delta/\delta = 2^{\text{poly}(nd)}$. Then the running time is $O(\text{poly}(\Delta/\delta)) \cdot 2^{\widetilde{O}(k^2)} = \text{poly}(nd)2^{\widetilde{O}(k^2)}$.

Also, a polynomial system verifier is able to tell us whether there exists a solution in a semi-algebraic set. In order to find the solution, we need to do a binary search over the cost C. In each step of the binary search we use a polynomial system verifier to determine if there exists a solution in,

$$\begin{aligned}
&\sum_{i=1}^{t_1}\sum_{j=1}^{t_2}\sigma_{i,j}(T_1(AR)XY(DA)T_2)_{i,j} \leq C \\
&\sigma_{i,j}^2 = 1, \forall i \in [t_1], j \in [t_2] \\
&\sigma_{i,j} \cdot (T_1(AR)XY(DA)T_2)_{i,j} \geq 0, \forall i \in [t_1], j \in [t_2].
\end{aligned}$$

In order to do binary search over the cost, we need to know an upper bound on the cost and also a lower bound on the minimum nonzero cost. The upper bound on the cost is $C_{\max} = O(nd\Delta)$, and the minimum nonzero cost is $C_{\min} = 2^{-\Omega(\text{poly}(nd))}$. Thus, the total number of steps for binary search is $O(\log(C_{\max}/C_{\min}))$. Overall, the running time is

$$nd^{\widetilde{O}(k)} \cdot 2^{\widetilde{O}(k^2)} \cdot \log(\Delta/\delta) \cdot \log(C_{\max}/C_{\min}) = \text{poly}(n)d^{\widetilde{O}(k)}2^{\widetilde{O}(k^2)}.$$

This completes the proof. □

Instead of solving an ℓ_1 problem at the last step of L1LowRankApproxK by using a polynomial system verifier, we can just solve a Frobenius norm minimization problem. This slightly improves the running time and pays an extra $\text{poly}(k)$ factor in the approximation ratio. Thus, we obtain the following corollary,

Corollary 17.4.8. *Given matrix $A \in \mathbb{R}^{n \times d}$, there exists an algorithm that takes $\mathrm{poly}(n) \cdot d^{\widetilde{O}(k)}$ time which outputs two matrices $U \in \mathbb{R}^{n \times k}$, $V \in \mathbb{R}^{k \times d}$ such that*

$$|UV - A\|_1 \leq \mathrm{poly}(k) \min_{\mathrm{rank}-k \ A_k} \|A_k - A\|_1$$

holds with probability $9/10$.

17.4.5 Rank-$3k$ and $O(1)$-approximation algorithm for an arbitrary matrix A

In this section, we show how to output a rank-$3k$ solution that is able to achieve an $O(1)$-approximation.

Algorithm 17.5 Bicriteria $O(1)$-approximation Algorithm

procedure L1LowRankApproxBicriteria(A, n, d, k) ▷ Theorem 17.4.9
  $U_B, V_B \leftarrow \min_{U \in \mathbb{R}^{n \times k}, V \in \mathbb{R}^{k \times d}} \|UV - A\|_F$.
  $r \leftarrow O(k \log k)$.
  Guess a diagonal matrix $D \in \mathbb{R}^{n \times n}$ with r nonzero entries.
  Guess matrix $DU \in \mathbb{R}^{r \times 2k}$.
  Find V_A by solving $\min_V \|DUV - D(A - B)\|_1$.
  Find U_A by solving $\min_U \|UV_A - (A - B)\|_1$.
  Take the best solution $\begin{bmatrix} U_A & U_B \end{bmatrix}, \begin{bmatrix} V_A \\ V_B \end{bmatrix}$ over all guesses.
  return $\begin{bmatrix} U_A & U_B \end{bmatrix}, \begin{bmatrix} V_A \\ V_B \end{bmatrix}$.
end procedure

Theorem 17.4.9. *Given matrix $A \in \mathbb{R}^{n \times d}$, for any $k \geq 1$, there exists an algorithm which takes $(nd)^{\widetilde{O}(k^2)}$ time to output two matrices $U \in \mathbb{R}^{n \times 3k}$, $V \in \mathbb{R}^{3k \times d}$,*

$$\|UV - A\|_1 \lesssim \min_{\text{rank}-k \ A_k} \|A_k - A\|_1,$$

holds with probability $9/10$.

Proof. We define OPT to be

$$\arg\min_{U \in \mathbb{R}^{n \times k}, V \in \mathbb{R}^{k \times d}} \|UV - A\|_1$$

Solving the Frobenius norm problem, we can find a factorization of a rank-k matrix $B = U_B V_B$ where $U_B \in \mathbb{R}^{n \times k}, V_B \in \mathbb{R}^{k \times d}$, and B satisfies $\|A - B\|_1 \leq \alpha_B \, \mathrm{OPT}$ for an $\alpha_B = \sqrt{nd}$.

We define $U^* \in \mathbb{R}^{n\times 2k}, V^* \in \mathbb{R}^{2k\times d}$ to be the optimal solution, i.e.,

$$U^*V^* = \arg\min_{U\in\mathbb{R}^{n\times 2k}, V\in\mathbb{R}^{2k\times d}} \|UV - (A-B)\|_1.$$

Let D be a sampling and rescaling diagonal matrix corresponding to the Lewis weights of U^*, and let the number of nonzero entries on the diagonal be $t = O(k\log k)$.

By Lemma 17.5.5 and Lemma 17.5.2, the solution of $\min_{V\in\mathbb{R}^{k\times d}} \|DU^*V - D(A-B)\|_1$ together with U^* gives an $O(1)$-approximation to $A-B$. In order to compute D we need to know U^*. Although we do not know U^*, there still exists a way to figure out the Lewis weights. The idea is the same as in the previous discussion Lemma 17.4.10 "Guessing Lewis weights". By Claim 17.4.11 and Claim 17.4.12, the total number of possible D is $n^{O(t)}$.

Lemma 17.4.10 tries to find a rank-k solution when all the entries in A are integers at most $\mathrm{poly}(nd)$. Here we focus on a bicriteria algorithm that outputs a rank-$3k$ matrix without such a strong bit complexity assumption. We can show a better claim 17.4.13, which is that the total number of possible DU^* is $N^{\tilde{O}(k^2)}$ where $N = \mathrm{poly}(n)$ is the number of choices for a single entry in U^*.

We explain how to obtain an upper bound on N. Consider the optimum $\|U^*V^* - (A-B)\|_1$. We can always change the basis so assume V^* is an Auerbach basis (i.e., an ℓ_1-well-conditioned basis discussed in Section 17.1), so $e\|x\|_1 \geq \|xV^*\|_1 \geq \|x\|_1/f$, where $e, f = \mathrm{poly}(k)$. Then no entry of U^* is larger than $2f\|A-B\|_1$, otherwise we could replace U^* with 0 and get a better solution. Also any entry smaller than $\|A-B\|_1/(enk\alpha_B 100)$ can be replaced with 0 as this will incur additive error at most $\mathrm{OPT}/100$. So if we round to integer multiples of $\|A-B\|_1/(enk\alpha_B 100)$ we only have $O(enk\alpha_B f)$ possibilities for each

entry of U^* and still have an $O(1)$-approximation. We will just refer to this rounded U^* as U^*, abusing notation.

Let $\mathcal{U}$ denote the set of all the matrices U that we guess. From the above discussion, we conclude that, there exists a $U \in \mathcal{U}$ such that $\|UV^* - (A-B)\|_1 \leq O(\mathrm{OPT})$.

For each guess of DU^* and D, we find V_A, U_A in the following way. We find V_A by using a linear program to solve,

$$\min_{V \in \mathbb{R}^{k \times d}} \|DU^*V - D(A-B)\|_1.$$

Given V_A and A, we write down a linear program to solve this problem,

$$\min_{U \in \mathbb{R}^{n \times k}} \|UV_A - (A-B)\|_1,$$

which takes $\mathrm{poly}(ndk)$ time. Then we obtain U_A.

Recall that V_B, U_B are the two factors of B and it is a rank-k, α_B-approximation solution to $\min_{U,V} \|UV - A\|_1$. Then we have

$$\left\| \begin{bmatrix} U_A & U_B \end{bmatrix} \begin{bmatrix} V_A \\ V_B \end{bmatrix} - A \right\|_1 = \left\| U_A V_A - (A - U_B V_B) \right\|_1 = \left\| U_A V_A - (A-B) \right\|_1.$$

Because there must exist a pair U_A, V_A satisfying $\|U_A V_A - (A-B)\|_1 \leq O(\mathrm{OPT})$, it follows that by taking the best solution $\begin{bmatrix} U_A & U_B \end{bmatrix} \begin{bmatrix} V_A \\ V_B \end{bmatrix}$ over all guesses, we obtain an $O(1)$-approximation solution.

Overall, the running time is $(nd)^{\widetilde{O}(k^2)}$. □

Lemma 17.4.10. *Given an $n \times d$ matrix A with integers bounded by $\mathrm{poly}(n)$, for any $k \geq 1$, there exists an algorithm which takes $(nd)^{\widetilde{O}(k^3)}$ time to output two matrices $U \in \mathbb{R}^{n \times k}$,*

$V \in \mathbb{R}^{k \times d}$, *such that*

$$\|UV - A\|_1 \lesssim \min_{\text{rank}-k \ A_k} \|A_k - A\|_1,$$

Proof. We define $U^* \in \mathbb{R}^{n \times k}, V^* \in \mathbb{R}^{k \times d}$ to be the optimal solution, i.e.,

$$U^*, V^* = \arg\min_{U \in \mathbb{R}^{n \times k}, V \in \mathbb{R}^{k \times d}} \|UV - A\|_1,$$

and define $\mathrm{OPT} = \|U^* V^* - A\|_1$.

Let D denote a sampling and rescaling diagonal matrix corresponding to the Lewis weights of U^*, and let the number of nonzero entries on the diagonal be $t = O(k \log k)$.

By Lemma 17.5.5 and Lemma 17.5.2, the solution to $\min_{V \in \mathbb{R}^{2k \times d}} \|DU^* V - DA\|_1$ together with U^* gives an $O(1)$-approximation to A. In order to compute D, we need to know U^*. Although we do not know U^*, there still exists a way to figure out the Lewis weights. We call the idea "Guessing Lewis weights". We will explain this idea in the next few paragraphs.

First, we can guess the nonzero entries on the diagonal, because the number of choices is small.

Claim 17.4.11. *The number of possible choice of* $\mathrm{supp}(D)$ *is at most* $n^{O(t)}$.

Proof. The matrix has dimension $n \times n$ and the number of nonzero entries is t. Thus the number of possible choices is at most $\sum_{i=1}^{t} \binom{n}{t} = n^{O(t)}$. □

Second, we can guess the value of each probability. For each probability, it is trivially at most 1. If the probability is less than $1/(\mathrm{poly}(n) k \log k)$, then we will never sample that row with high probability. It means that we can truncate the probability if it is below that

threshold. We can also round each probability to 2^{-i} which only loses another constant factor in the approximation ratio. Thus, we have:

Claim 17.4.12. *The total number of possible D is $n^{O(t)} = n^{\widetilde{O}(k)}$.*

Since $\Delta/\delta \leq \text{poly}(n)$ and the entries of A are in

$$\{-\Delta, -\Delta+\delta, \ldots, -2\delta, -\delta, 0, \delta, 2\delta, \cdots, \Delta-\delta, \Delta\},$$

we can lower bound the cost of $\|U^*V - A\|_1$ given that it is non-zero by $(nd\Delta/\delta)^{-O(k)}$ (if it is zero then A has rank at most k and we output A) using Lemma 4.1 in [CW09] and relating entrywise ℓ_1-norm to Frobenius norm. We can assume V is an ℓ_1 well-conditioned basis, since we can replace U^* with U^*R^{-1} and V with RV for any invertible linear transformation R. By properties of such basis, we can discretize the entries of U^* to integer multiples of $(nd\Delta/\delta)^{-O(k)}$ while preserving relative error. Hence we can correctly guess each entry of DU^* in $\left(n^{O(k)}\right)$ time.

Claim 17.4.13. *The total number of possible DU^* is $n^{\widetilde{O}(k^3)}$.*

In the following, let DU denote a guess of DU^*. Now the problem remaining is to solve $\min_{V \in \mathbb{R}^{k\times d}} \|DUV - DA\|_1$. Since we already know DA can be computed, and we know DU, we can solve this multiple regression problem by running linear programming. Thus the running time of this step is in $\text{poly}(nd)$. After we get such a solution V, we use a linear program to solve $\min_{U \in \mathbb{R}^{n\times k}} \|DUV - DA\|_1$. Then we can get U.

After we guess all the choices of D and DU^*, we must find a solution U, V which gives an $O(1)$ approximation. The total running time is $n^{\widetilde{O}(k)} \cdot n^{\widetilde{O}(k^3)} \cdot \text{poly}(nd) = (nd)^{\widetilde{O}(k^3)}$. □

17.4.6 CUR decomposition for an arbitrary matrix A

Algorithm 17.6 CUR Decomposition Algorithm

procedure L1LowRankApproxCUR(A, n, d, k) ▷ Theorem 17.4.14
  $U_B, V_B \leftarrow$ L1LowRankApproxPolykLogd(A, n, d, k).
  Let $D_1 \in \mathbb{R}^{n\times n}$ be the sampling and rescaling diagonal matrix corresponding to the Lewis weights of $B_1 = U_B \in \mathbb{R}^{n\times k}$, and let D_1 have $d_1 = O(k \log k)$ nonzero entries.
  Let $D_2^\top \in \mathbb{R}^{d\times d}$ be the sampling and rescaling diagonal matrix corresponding to the Lewis weights of $B_2^\top = \left((D_1 B_1)^\dagger D_1 A\right)^\top \in \mathbb{R}^{d\times k}$, and let D_2 have $d_2 = O(k \log k)$ nonzero entries.
  $C \leftarrow AD_2$, $U \leftarrow (B_2 D_2)^\dagger (D_1 B_1)^\dagger$, and $R \leftarrow D_1 A$.
  return C, U, R.
end procedure

Theorem 17.4.14. *Given matrix $A \in \mathbb{R}^{n\times d}$, for any $k \geq 1$, there exists an algorithm which takes $O(\mathrm{nnz}(A)) + (n+d)\,\mathrm{poly}(k)$ time to output three matrices $C \in \mathbb{R}^{n\times c}$ with columns from A, $U \in \mathbb{R}^{c\times r}$, and $R \in \mathbb{R}^{r\times d}$ with rows from A, such that $\mathrm{rank}(CUR) = k$, $c = O(k \log k)$, $r = O(k \log k)$, and*

$$\|CUR - A\|_1 \leq \mathrm{poly}(k) \log d \min_{\mathrm{rank}-k\ A_k} \|A_k - A\|_1,$$

holds with probability $9/10$.

Proof. We define

$$\mathrm{OPT} := \min_{\mathrm{rank}-k\ A_k} \|A_k - A\|_1.$$

Due to Theorem 17.4.6, we can output two matrices $U_B \in \mathbb{R}^{n\times k}$, $V_B \in \mathbb{R}^{k\times d}$ such that $U_B V_B$ gives a rank-k, and $\mathrm{poly}(k) \log d$-approximation solution to A, i.e.,

$$\|U_B V_B - A\|_1 \leq \mathrm{poly}(k) \log d\, \mathrm{OPT}\,. \tag{17.12}$$

By Section 17.3.3, we can compute $D_1 \in \mathbb{R}^{n\times n}$ which is a sampling and rescaling matrix corresponding to the Lewis weights of $B_1 = U_B$ in $O(n\,\mathrm{poly}(k))$ time, and there are $d_1 = O(k\log k)$ nonzero entries on the diagonal of D_1.

Define $V^* \in \mathbb{R}^{k\times d}$ to be the optimal solution of $\min_{V\in\mathbb{R}^{k\times d}} \|B_1V - A\|_1$, $\widehat{V} = (D_1B_1)^\dagger D_1A \in \mathbb{R}^{k\times d}$, $U_1 \in \mathbb{R}^{n\times k}$ to be the optimal solution of $\min_{U\in\mathbb{R}^{n\times k}} \|U\widehat{V} - A\|_1$, and V' to be the optimal solution of $\min_{V\in\mathbb{R}^{k\times d}} \|D_1A - D_1B_1V\|_1$.

By Claim 17.3.5, we have

$$\|D_1B_1\widehat{V} - D_1A\|_1 \leq \sqrt{d_1}\|D_1B_1V' - D_1A\|_1.$$

Due to Lemma 17.5.5 and Lemma 17.5.2, with constant probability, we have

$$\|B_1\widehat{V} - A\|_1 \leq \sqrt{d_1}\alpha_{D_1}\|B_1V^* - A\|_1,$$

where $\alpha_{D_1} = O(1)$.

Now, we can show,

$$\begin{aligned}
\|U_1\widehat{V} - A\|_1 \leq\ & \|B_1\widehat{V} - A\|_1 && \text{by } U_1 = \arg\min_{U\in\mathbb{R}^{n\times k}}\|U\widehat{V} - A\|_1 \\
\lesssim\ & \sqrt{d_1}\|B_1V^* - A\|_1 \\
\leq\ & \sqrt{d_1}\|U_BV_B - A\|_1 \\
\leq\ & \mathrm{poly}(k)\log d\,\mathrm{OPT}\,. && \text{by Equation (17.12)} \qquad (17.13)
\end{aligned}$$

We define $B_2 = \widehat{V}$, then we replace $\widehat{V}$ by $B_2 \in \mathbb{R}^{k\times d}$ and look at this objective function,

$$\min_{U\in\mathbb{R}^{n\times k}} \|UB_2 - A\|_1,$$

where U^* denotes the optimal solution. We use a sketching matrix to sketch the RHS of matrix $UB_2 - A \in \mathbb{R}^{n\times d}$. Let $D_2^\top \in \mathbb{R}^{d\times d}$ denote a sampling and rescaling diagonal matrix corresponding to the Lewis weights of $B_2^\top \in \mathbb{R}^{d\times k}$, and let the number of nonzero entries on the diagonal of D_2 be $d_2 = O(k \log k)$. We define $\widehat{U} = AD_2(B_2D_2)^\dagger \in \mathbb{R}^{n\times k}$, $U' \in \mathbb{R}^{n\times k}$ to be the optimal solution of $\min_{U\in\mathbb{R}^{n\times k}} \|(UB_2 - A)D_2\|_1$. Recall that $U_1 \in \mathbb{R}^{n\times k}$ is the optimal of $\min_{U\in\mathbb{R}^{n\times k}} \|UB_2 - A\|_1$.

By Claim 17.3.5, we have

$$\|\widehat{U}B_2D_2 - AD_2\|_1 \leq \sqrt{d_2}\|U'B_2D_2 - AD_2\|_1.$$

According to Lemma 17.5.2 and Lemma 17.5.5, with constant probability,

$$\|\widehat{U}B_2 - A\|_1 \leq \alpha_{D_2}\sqrt{d_2}\|U_1B_2 - A\|_1,$$

where $\alpha_{B_2} = O(1)$.

We have

$$\begin{aligned}
& \|\widehat{U}B_2 - A\|_1 \\
\leq\ & \sqrt{d_2}\alpha_{D_2}\|U_1B_2 - A\|_1 \\
=\ & \sqrt{d_2}\alpha_{D_2}\|U_1\widehat{V} - A\|_1 && \text{by } B_2 = \widehat{V} \\
\leq\ & \mathrm{poly}(k)\log(d)\,\mathrm{OPT}\,. && \text{by Equation (17.13)}
\end{aligned}$$

Notice that $\widehat{U}B_2 = AD_2(B_2D_2)^\dagger(D_1B_1)^\dagger D_1A$. Setting

$$C = AD_2 \in \mathbb{R}^{n\times d_2}, U = (B_2D_2)^\dagger(D_1B_1)^\dagger \in \mathbb{R}^{d_2\times d_1}, \text{ and } R = D_1A \in \mathbb{R}^{d_1\times d},$$

we get the desired CUR decomposition,

$$\| \underbrace{AD_2}_{C} \cdot \underbrace{(B_2D_2)^\dagger (D_1B_1)^\dagger}_{U} \cdot \underbrace{D_1A}_{R} - A\|_1 \leq \mathrm{poly}(k) \log(d)\, \mathrm{OPT}\,.$$

with $\mathrm{rank}(CUR) = k$. Overall, the running time is $O(\mathrm{nnz}(A)) + (n+d)\,\mathrm{poly}(k)$.

□

17.4.7 Rank-r matrix B

17.4.7.1 Properties

Lemma 17.4.15. *Given matrix $A \in \mathbb{R}^{n\times d}$, let $\mathrm{OPT} = \min_{\mathrm{rank}-k\ A_k} \|A - A_k\|_1$. For any $r \geq k$, if rank-r matrix $B \in \mathbb{R}^{n\times d}$ is an f-approximation to A, i.e.,*

$$\|B - A\|_1 \leq f \cdot \mathrm{OPT},$$

and $U \in \mathbb{R}^{n\times k}, V \in \mathbb{R}^{k\times d}$ is a g-approximation to B, i.e.,

$$\|UV - B\|_1 \leq g \cdot \min_{\mathrm{rank}-k\ B_k} \|B_k - B\|_1,$$

then,

$$\|UV - A\|_1 \lesssim gf \cdot \mathrm{OPT}\,.$$

Proof. We define $\widetilde{U} \in \mathbb{R}^{n\times k}, \widetilde{V} \in \mathbb{R}^{k\times d}$ to be two matrices, such that

$$\|\widetilde{U}\widetilde{V} - B\|_1 \leq g \min_{\mathrm{rank}-k\ B_k} \|B_k - B\|_1,$$

and also define,

$$\widehat{U}, \widehat{V} = \operatorname*{arg\,min}_{U\in\mathbb{R}^{n\times k}, V\in\mathbb{R}^{k\times d}} \|UV - B\|_1 \text{ and } U^*, V^* = \operatorname*{arg\,min}_{U\in\mathbb{R}^{n\times k}, V\in\mathbb{R}^{k\times d}} \|UV - A\|_1,$$

Then,

$$\begin{aligned}
\|\widetilde{U}\widetilde{V} - A\|_1 &\leq \|\widetilde{U}\widetilde{V} - B\|_1 + \|B - A\|_1 && \text{by triangle inequality} \\
&\leq g\|\widehat{U}\widehat{V} - B\|_1 + \|B - A\|_1 && \text{by definition} \\
&\leq g\|U^* V^* - B\|_1 + \|B - A\|_1 && \text{by } \|\widehat{U}\widehat{V} - B\|_1 \leq \|U^* V^* - B\|_1 \\
&\leq g\|U^* V^* - A\|_1 + g\|B - A\|_1 + \|B - A\|_1 && \text{by triangle inequality} \\
&= g\,\mathrm{OPT} + (g+1)\|B - A\|_1 && \text{by definition of OPT} \\
&\leq g\,\mathrm{OPT} + (g+1)f \cdot \mathrm{OPT} && \text{by } B \text{ is } f\text{-approximation to } A \\
&\lesssim gf\,\mathrm{OPT}\,.
\end{aligned}$$

This completes the proof. □

Lemma 17.4.16. *Given a matrix $B \in \mathbb{R}^{n \times d}$ with rank r, for any $1 \leq k < r$, for any fixed $U^* \in \mathbb{R}^{n \times k}$, choose a Cauchy matrix S with $m = O(r \log r)$ rows and rescaled by $\Theta(1/m)$. With probability .999 for all $V \in \mathbb{R}^{k \times d}$, we have*

$$\|SU^*V - SB\|_1 \geq \|U^*V - B\|_1.$$

Proof. This follows by definitions in Section 17.5 and Lemma 17.5.15. □

Lemma 17.4.17. *Given a matrix $B \in \mathbb{R}^{n \times d}$ with rank r, for any $1 \leq k < r$, for any fixed $U^* \in \mathbb{R}^{n \times k}, V^* \in \mathbb{R}^{k \times d}$, choose a Cauchy matrix S with m rows and rescaled by $\Theta(1/m)$. We have*

$$\|SU^*V^* - SB\|_1 \leq O(r \log r)\|U^*V^* - B\|_1,$$

with probability .999.

Proof. Let U denote a well-conditioned basis of $[U^*V^*, B]$, then U has $\widetilde{d} = O(r)$ columns. We have

$$\begin{aligned}
\|SU\|_1 &= \sum_{i=1}^{\widetilde{d}} \sum_{j=1}^{m} |(SU_i)_j| \\
&= \sum_{i=1}^{\widetilde{d}} \sum_{j=1}^{m} |\frac{1}{m} \sum_{l=1}^{n} S_{j,l} U_{l,i}| && \text{by } S_{j,l} \sim C(0,1) \\
&= \frac{1}{m} \sum_{i=1}^{\widetilde{d}} \sum_{j=1}^{m} |c_{i,j}| && \text{by } c_{i,j} \sim C(0, \|U_i\|_1) \\
&= \frac{1}{m} \sum_{i=1}^{\widetilde{d}} \sum_{j=1}^{m} \|U_i\|_1 \cdot w_{i+(j-1)d}, && \text{by } w_{i,j} \sim |C(0,1)|
\end{aligned}$$

where the last step follows since each w_i can be thought of as a clipped half-Cauchy random variable. Define $d' = m\widetilde{d}$. Define event ξ_i to be the situation when $w_i < D$ (we will choose D later), and define event $\xi = \xi_1 \cap \xi_2 \cap \cdots \cap \xi_{d'}$. Using a similar proof as Lemma 17.5.6, which is also similar to previous work [Ind06, SW11, CDMI+13], we obtain that

$$\Pr\left[\sum_{i=1}^{m} \|SU_i\|_1 \geq \sum_{i=1}^{m} \|U_i\|_1 t\right] \lesssim \frac{\log d'}{t} + \frac{d'}{D}.$$

Choosing $t = \Theta(\log d')$ and $D = \Theta(d')$, we have

$$\Pr\left[\sum_{i=1}^{m} \|SU_i\|_1 \geq \sum_{i=1}^{m} \|U_i\|_1 O(\log d')\right] \leq \frac{1}{C},$$

for a constant C. Condition on the above event. Let $y = Ux$, for some $x \in \mathbb{R}^d$. Then for

any y,

$$\begin{aligned}
\|Sy\|_1 = &\ \|SUx\|_1 \\
\leq &\ \sum_{j=1}^{d'} \|SU_j x_j\|_1 && \text{by triangle inequality} \\
= &\ \sum_{j=1}^{d'} |x_j| \cdot \|SU_j\|_1 \\
\lesssim &\ \|x\|_\infty \log(d') \sum_{j=1}^{d'} \|U_j\|_1 \\
\lesssim &\ r \log r \|y\|_1,
\end{aligned}$$

where the last step follows by $\sum_{j=1}^{d'} \|U_j\|_1 \leq d'$ and $\|x\|_\infty \leq \|Ux\|_1 = \|y\|_1$. Choosing $C = 1000$ completes the proof. □

Lemma 17.4.18. *Given a matrix $M \in \mathbb{R}^{n\times d}$ with rank $O(r)$, choose a random matrix $S \in \mathbb{R}^{m\times n}$ with each entry drawn from a standard Cauchy distribution and scaled by $\Theta(1/m)$. We have that*

$$\|SM\|_1 \leq O(r \log r)\|M\|_1,$$

holds with probability .999.

Proof. Let $U \in \mathbb{R}^{O(r)}$ be the well-conditioned basis of M. Then each column of M can be expressed by Ux for some x. We then follow the same proof as that of Lemma 17.4.17. □

17.4.7.2 $\mathrm{poly}(k, r)$-approximation for rank-r matrix B

Theorem 17.4.19. *Given a factorization of a* rank-r *matrix $B = U_B V_B \in \mathbb{R}^{n\times d}$, where $U_B \in \mathbb{R}^{n\times r}, V_B \in \mathbb{R}^{r\times d}$, for any $1 \leq k \leq r$ there exists an algorithm which takes $(n + d) \cdot \mathrm{poly}(k)$*

Algorithm 17.7 poly(r,k)-approximation Algorithm for Rank-r matrix B

procedure L1LowRankApproxB(U_B, V_B, n, d, k, r) ▷ Theorem 17.4.19
  Set $s \leftarrow \widetilde{O}(r), r' \leftarrow \widetilde{O}(r)$, $t_1 \leftarrow \widetilde{O}(r)$, $t_2 \leftarrow \widetilde{O}(r)$.
  Choose dense Cauchy matrices $S \in \mathbb{R}^{s\times n}$, $R \in \mathbb{R}^{d\times r'}$, $T_1 \in \mathbb{R}^{t_1\times n}$, $T_2 \in \mathbb{R}^{d\times t_2}$.
  Compute $S \cdot U_B \cdot V_B$, $U_B \cdot V_B \cdot R$ and $T_1 \cdot U_B \cdot V_B \cdot T_2$.
  Compute $XY = \arg\min_{X,Y} \|T_1 U_B V_B R X Y S U_B V_B T_2 - T_1 U_B V_B T_2\|_F$.
  return $U_B V_B R X, Y S U_B V_B$.
end procedure

time to output two matrices $U \in \mathbb{R}^{n\times k}, V \in \mathbb{R}^{k\times d}$ *such that*

$$\|UV - B\|_1 \leq \mathrm{poly}(r) \min_{\mathrm{rank}-k\ B_k} \|B_k - B\|_1,$$

holds with probability $9/10$.

Proof. We define

$$\mathrm{OPT} = \min_{\mathrm{rank}-k\ B_k} \|B_k - B\|_1.$$

Choose $S \in \mathbb{R}^{s\times n}$ to be a dense Cauchy transform matrix with $s = O(r \log r)$. Using Lemma 17.4.18, Lemma 17.5.15, and combining with Equation (17.8), we have

$$\min_{U\in\mathbb{R}^{n\times k}, Z\in\mathbb{R}^{k\times s}} \|UZSB - B\|_1 \leq \sqrt{s} O(r\log r)\,\mathrm{OPT} = O(r^{1.5}\log^{1.5} r)\,\mathrm{OPT}\,.$$

Let $\alpha_s = O(r^{1.5}\log^{1.5} r)$.

We define $U^*, Z^* = \arg\min_{U\in\mathbb{R}^{n\times k}, Z\in\mathbb{R}^{k\times d}} \|UZSB - B\|_1$. For the fixed $Z^* \in \mathbb{R}^{k\times s}$, choose a dense Cauchy transform matrix $R \in \mathbb{R}^{d\times r'}$ with $r' = O(r\log r)$ and sketch on the right of $(UZSB - B)$. We obtain the minimization problem, $\min_{U\in\mathbb{R}^{n\times k}} \|UZ^*SBR - BR\|_1$.

Define $\widehat{U}^j = B^j R((Z^*SB)R)^\dagger \in \mathbb{R}^k, \forall j \in [n]$. Then $\widehat{U} = BR((Z^*SB)R)^\dagger \in \mathbb{R}^{n\times k}$. Due to Claim 17.3.5,

$$\sum_{j=1}^n \|B^j R((Z^*SB)R)^\dagger Z^*SBR - B^j R\|_1 \leq O(\sqrt{r'}) \sum_{j=1}^n \min_{U^j\in\mathbb{R}^k} \|U^j Z^*SBR - B^j R\|_1,$$

which is equivalent to

$$\|BR((Z^*SB)R)^\dagger Z^*SBR - BR\|_1 \leq O(\sqrt{r'}) \min_{U\in\mathbb{R}^{n\times k}} \|UZ^*SBR - BR\|_1,$$

where BR is an $n \times r'$ matrix and SB is an $s \times d$ matrix. Both of them can be computed in $(n+d)\,\mathrm{poly}(r)$ time.

Using Lemma 17.5.2, Lemma 17.5.15, Lemma 17.4.18, we obtain,

$$\|BR((Z^*SB)R)^\dagger Z^*SB - B\|_1 \leq O(\sqrt{r'}\alpha_{r'}) \min_{U\in\mathbb{R}^{n\times k}} \|UZ^*SB - B\|_1$$

where $\alpha_{r'} = \widetilde{O}(r)$.

We define $X^* \in \mathbb{R}^{r\times k}$, $Y^* \in \mathbb{R}^{k\times s}$,

$$X^*, Y^* = \arg\min_{X\in\mathbb{R}^{r'\times k}, Y\in\mathbb{R}^{k\times s}} \|BRXYSB - B\|_1.$$

Then,

$$\begin{aligned}
\|BRX^*Y^*SB - B\|_1 &\leq \|BR((Z^*SB)R)^\dagger Z^*SB - B\|_1 \\
&\leq O(\sqrt{r'}\alpha_{r'}) \min_{U\in\mathbb{R}^{n\times k}} \|UZ^*SB - B\|_1 \\
&= O(\sqrt{r'}\alpha_{r'}) \min_{U\in\mathbb{R}^{n\times k}, Z\in\mathbb{R}^{k\times s}} \|UZSB - B\|_1 \\
&\leq O(\sqrt{r'}\alpha_{r'}\alpha_s)\,\mathrm{OPT}\,.
\end{aligned}$$

It means that BRX, YSB gives an $O(\alpha_{r'}\alpha_s\sqrt{r'})$-approximation to the original problem.

Thus it suffices to use Lemma 17.4.20 to solve

$$\min_{X\in\mathbb{R}^{r\times k},Y\in\mathbb{R}^{k\times s}} \|BRXYSB - B\|_1,$$

by losing an extra $\mathrm{poly}(r)$ approximation ratio. Therefore, we finish the proof. □

Lemma 17.4.20. *Suppose we are given $S \in \mathbb{R}^{s\times n}$, $R \in \mathbb{R}^{d\times r'}$, and a factorization of a* rank-r *matrix $B = U_B V_B \in \mathbb{R}^{n\times d}$, where $U_B \in \mathbb{R}^{n\times r}, V_B \in \mathbb{R}^{r\times d}$. Then for any $1 \le k \le r$, there exists an algorithm which takes $(n+d)\,\mathrm{poly}(r, r', s)$ time to output two matrices $X' \in \mathbb{R}^{r'\times k}, Y' \in \mathbb{R}^{k\times s}$ such that*

$$\|BRX' \cdot Y'SB - B\|_1 \le \mathrm{poly}(r) \min_{X\in\mathbb{R}^{r'\times k},Y\in\mathbb{R}^{k\times s}} \|BRXYSB - B\|_1$$

holds with probability at least .999.

Proof. Choosing dense Cauchy matrices $T_1 \in \mathbb{R}^{t_1\times n}, T_2^\top \in \mathbb{R}^{t_2\times d}$ to sketch on both sides, we get the problem

$$\min_{X\in\mathbb{R}^{r'\times k},Y\in\mathbb{R}^{k\times s}} \|T_1BRXYSBT_2 - T_1BT_2\|_1, \tag{17.14}$$

where $t_1 = \widetilde{O}(r)$ and $t_2 = \widetilde{O}(r)$.

Define X', Y' to be the optimal solution of

$$\min_{X\in\mathbb{R}^{r'\times k},Y\in\mathbb{R}^{k\times s}} \|T_1BRXYSBT_2 - T_1BT_2\|_F.$$

Define $\widetilde{X}, \widetilde{Y}$ to the be the optimal solution of

$$\min_{X\in\mathbb{R}^{r'\times k},Y\in\mathbb{R}^{k\times s}} \|BRXYSB - B\|_1.$$

By Claim 17.3.6,

$$\|T_1BRX'Y'SBT_2 - T_1BT_2\|_1 \leq \sqrt{t_1t_2} \min_{X\in\mathbb{R}^{r'\times k}, Y\in\mathbb{R}^{k\times s}} \|T_1BRXYSBT_2 - T_1BT_2\|_1.$$

By Lemma 17.5.4, Lemma 17.4.18 and Lemma 17.5.15, we have

$$\|BRX' \cdot Y'SB - B\|_1 \leq \sqrt{t_1t_2} \cdot \widetilde{O}(r^2)\|BR\widetilde{X} \cdot \widetilde{Y}SB - B\|_1.$$

This completes the proof. □

17.5 Contraction and Dilation Bound for ℓ_1

This section presents the essential lemmas for ℓ_1-low rank approximation. Section 17.5.1 gives some basic definitions. Section 17.5.2 shows some properties implied by contraction and dilation bounds. Section 17.5.3 presents the no dilation lemma for a dense Cauchy transform. Section 17.5.4 and 17.5.5 presents the no contraction lemma for dense Cauchy transforms. Section 17.5.6 and 17.5.7 contains the results for sparse Cauchy transforms and Lewis weights.

17.5.1 Definitions

Definition 17.5.1. Given a matrix $M \in \mathbb{R}^{n\times d}$, if matrix $S \in \mathbb{R}^{m\times n}$ satisfies

$$\|SM\|_1 \leq c_1\|M\|_1,$$

then S has at most c_1-dilation on M.

Definition 17.5.2. Given a matrix $U \in \mathbb{R}^{n\times k}$, if matrix $S \in \mathbb{R}^{m\times n}$ satisfies

$$\forall x \in \mathbb{R}^k, \|SUx\|_1 \geq \frac{1}{c_2}\|Ux\|_1,$$

then S has at most c_2-contraction on U.

Definition 17.5.3. Given matrices $U \in \mathbb{R}^{n\times k}, A \in \mathbb{R}^{n\times d}$, let $V^* = \arg\min_{V\in\mathbb{R}^{k\times d}} \|UV-A\|_1$. If matrix $S \in \mathbb{R}^{m\times n}$ satisfies

$$\forall V \in \mathbb{R}^{k\times d}, \|SUV - SA\|_1 \geq \frac{1}{c_3}\|UV - A\|_1 - c_4\|UV^* - A\|_1,$$

then S has at most (c_3, c_4)-contraction on (U, A).

Definition 17.5.4. A (c_5, c_6) ℓ_1-subspace embedding for the column space of an $n\times k$ matrix U is a matrix $S \in \mathbb{R}^{m\times n}$ for which all $x \in \mathbb{R}^k$

$$\frac{1}{c_5}\|Ux\|_1 \leq \|SUx\|_1 \leq c_6\|Ux\|_1.$$

Definition 17.5.5. Given matrices $U \in \mathbb{R}^{n\times k}, A \in \mathbb{R}^{n\times d}$, let $V^* = \arg\min_{V\in\mathbb{R}^{k\times d}} \|UV-A\|_1$. Let $S \in \mathbb{R}^{m\times n}$. If for all $c \geq 1$, and if for any $\widehat{V} \in \mathbb{R}^{k\times d}$ which satisfies

$$\|SU\widehat{V} - SA\|_1 \leq c \cdot \min_{V\in\mathbb{R}^{k\times d}} \|SUV - SA\|_1,$$

it holds that

$$\|U\widehat{V} - A\|_1 \leq c \cdot c_7 \cdot \|UV^* - A\|_1,$$

then S provides a c_7-multiple-regression-cost preserving sketch of (U, A).

Definition 17.5.6. Given matrices $L \in \mathbb{R}^{n \times m_1}, N \in \mathbb{R}^{m_2 \times d}, A \in \mathbb{R}^{n \times d}, k \geq 1$, let

$$X^* = \arg \min_{\text{rank}-k \ X} \|LXN - A\|_1.$$

Let $S \in \mathbb{R}^{m \times n}$. If for all $c \geq 1$, and if for any rank $-k$ $\widehat{X} \in \mathbb{R}^{m_1 \times m_2}$ which satisfies

$$\|SL\widehat{X}N - SA\|_1 \leq c \cdot \min_{\text{rank}-k \ X} \|SLXN - SA\|_1,$$

it holds that

$$\|L\widehat{X}N - A\|_1 \leq c \cdot c_8 \cdot \|LX^*N - A\|_1,$$

then S provides a c_8-restricted-multiple-regression-cost preserving sketch of (L, N, A, k).

17.5.2 Properties

Lemma 17.5.1. *Given matrices $A \in \mathbb{R}^{n\times d}, U \in \mathbb{R}^{n\times k}$, let $V^* = \arg\min_{V\in\mathbb{R}^{k\times d}} \|UV - A\|_1$. If $S \in \mathbb{R}^{m\times n}$ has at most c_1-dilation on $UV^* - A$, i.e.,*

$$\|S(UV^* - A)\|_1 \leq c_1\|UV^* - A\|_1,$$

and it has at most c_2-contraction on U, i.e.,

$$\forall x \in \mathbb{R}^k, \|SUx\|_1 \geq \frac{1}{c_2}\|Ux\|_1,$$

then S has at most $(c_2, c_1 + \frac{1}{c_2})$-contraction on (U, A), i.e.,

$$\forall V \in \mathbb{R}^{k\times d}, \|SUV - SA\|_1 \geq \frac{1}{c_2}\|UV - A\|_1 - (c_1 + \frac{1}{c_2})\|UV^* - A\|_1,$$

Proof. Let $A \in \mathbb{R}^{n\times d}, U \in \mathbb{R}^{n\times k}, S \in \mathbb{R}^{m\times n}$ be the same as that described in the lemma. Then $\forall V \in \mathbb{R}^{k\times d}$

$$\begin{aligned}
\|SUV - SA\|_1 &\geq \|SUV - SUV^*\|_1 - \|SUV^* - SA\|_1 \\
&\geq \|SUV - SUV^*\|_1 - c_1\|UV^* - A\|_1 \\
&= \|SU(V - V^*)\|_1 - c_1\|UV^* - A\|_1 \\
&= \sum_{j=1}^{d} \|SU(V - V^*)_j\|_1 - c_1\|UV^* - A\|_1 \\
&\geq \sum_{j=1}^{d} \frac{1}{c_2}\|U(V - V^*)_j\|_1 - c_1\|UV^* - A\|_1 \\
&= \frac{1}{c_2}\|UV - UV^*\|_1 - c_1\|UV^* - A\|_1 \\
&\geq \frac{1}{c_2}\|UV - A\|_1 - \frac{1}{c_2}\|UV^* - A\|_1 - c_1\|UV^* - A\|_1 \\
&= \frac{1}{c_2}\|UV - A\|_1 - \left((\frac{1}{c_2} + c_1)\|UV^* - A\|_1\right).
\end{aligned}$$

The first inequality follows by the triangle inequality. The second inequality follows since S has at most c_1 dilation on $UV^* - A$. The third inequality follows since S has at most c_2 contraction on U. The fourth inequality follows by the triangle inequality. □

Lemma 17.5.2. *Given matrices $A \in \mathbb{R}^{n\times d}, U \in \mathbb{R}^{n\times k}$, let $V^* = \arg\min_{V\in\mathbb{R}^{k\times d}} \|UV - A\|_1$. If $S \in \mathbb{R}^{m\times n}$ has at most c_1-dilation on $UV^* - A$, i.e.,*

$$\|S(UV^* - A)\|_1 \leq c_1\|UV^* - A\|_1,$$

and has at most c_2-contraction on U, i.e.,

$$\forall x \in \mathbb{R}^k, \|SUx\|_1 \geq \frac{1}{c_2}\|Ux\|_1,$$

then S provides a $(2c_1c_2+1)$-multiple-regression-cost preserving sketch of (U, A), i.e., for all $c \geq 1$, for any $\widehat{V} \in \mathbb{R}^{k\times d}$ which satisfies

$$\|SU\widehat{V} - SA\|_1 \leq c \cdot \min_{V\in\mathbb{R}^{k\times d}} \|SUV - SA\|_1,$$

it has

$$\|U\widehat{V} - A\|_1 \leq c \cdot (2c_1c_2 + 1) \cdot \|UV^* - A\|_1,$$

Proof. Let $S \in \mathbb{R}^{m\times n}, A \in \mathbb{R}^{n\times d}, U \in \mathbb{R}^{n\times k}, V^*, \widehat{V} \in \mathbb{R}^{k\times d}$, and c be the same as stated in the lemma.

$$\begin{aligned}
\|U\widehat{V} - A\|_1 &\leq c_2\|SU\widehat{V} - SA\|_1 + (1 + c_1c_2)\|UV^* - A\|_1 \\
&\leq c_2c \min_{V\in\mathbb{R}^{k\times d}} \|SUV - SA\|_1 + (1 + c_1c_2)\|UV^* - A\|_1 \\
&\leq c_2c\|SUV^* - SA\|_1 + (1 + c_1c_2)\|UV^* - A\|_1 \\
&\leq c_1c_2c\|UV^* - A\|_1 + (1 + c_1c_2)\|UV^* - A\|_1 \\
&\leq c \cdot (1 + 2c_1c_2)\|UV^* - A\|_1.
\end{aligned}$$

The first inequality follows by Lemma 17.5.1. The second inequality follows by the guarantee of $\widehat{V}$. The fourth inequality follows since S has at most c_1-dilation on $UV^* - A$. The fifth inequality follows since $c \geq 1$. □

Lemma 17.5.3. *Given matrices $L \in \mathbb{R}^{n \times m_1}, N \in \mathbb{R}^{m_2 \times d}, A \in \mathbb{R}^{n \times d}, k \geq 1$, let*

$$X^* = \arg \min_{\text{rank}-k \ X} \|LXN - A\|_1.$$

If $S \in \mathbb{R}^{m \times n}$ has at most c_1-dilation on $LX^*N - A$, i.e.,

$$\|S(LX^*N - A)\|_1 \leq c_1 \|LX^*N - A\|_1,$$

and has at most c_2-contraction on L, i.e.,

$$\forall x \in \mathbb{R}^{m_1} \|SLx\|_1 \geq \|Lx\|_1,$$

then S provides a $(2c_1c_2+1)$-restricted-multiple-regression-cost preserving sketch of (L, N, A, k), i.e., for all $c \geq 1$, for any rank $-k$ *$\widehat{X} \in \mathbb{R}^{m_1 \times m_2}$ which satisfies*

$$\|SL\widehat{X}N - SA\|_1 \leq c \cdot \min_{\text{rank}-k \ X} \|SLXN - SA\|_1,$$

it holds that

$$\|L\widehat{X}N - A\|_1 \leq c \cdot (2c_1c_2 + 1) \cdot \|LX^*N - A\|_1.$$

Proof. Let $S \in \mathbb{R}^{m \times n}, L \in \mathbb{R}^{n \times m_1}, \widehat{X} \in \mathbb{R}^{m_1 \times m_2}, X^* \in \mathbb{R}^{m_1 \times m_2}, N \in \mathbb{R}^{m_2 \times d}, A \in \mathbb{R}^{n \times d}$, and

$c \geq 1$ be the same as stated in the lemma.

$$\begin{aligned}
\|SL\widehat{X}N - SA\|_1 &\geq \|SL\widehat{X}N - SLX^*N\|_1 - \|SLX^*N - SA\|_1 \\
&\geq \frac{1}{c_2}\|L(\widehat{X}N - X^*N)\|_1 - c_1\|LX^*N - A\|_1 \\
&\geq \frac{1}{c_2}\|L\widehat{X}N - A\|_1 - \frac{1}{c_2}\|LX^*N - A\|_1 - c_1\|LX^*N - A\|_1 \\
&= \frac{1}{c_2}\|L\widehat{X}N - A\|_1 - (\frac{1}{c_2} + c_1)\|LX^*N - A\|_1.
\end{aligned}$$

The inequality follows by the triangle inequality. The second inequality follows since S has at most c_2-contraction on L, and it has at most c_1-dilation on $LX^*N - A$. The third inequality follows by the triangle inequality.

It follows that

$$\begin{aligned}
\|L\widehat{X}N - A\|_1 &\leq c_2\|SL\widehat{X}N - SA\|_1 + (1 + c_1c_2)\|LX^*N - A\|_1 \\
&\leq c_2c \cdot \min_{\text{rank}-k\ X} \|SLXN - SA\|_1 + (1 + c_1c_2)\|LX^*N - A\|_1 \\
&\leq c_2c \cdot \|SLX^*N - SA\|_1 + (1 + c_1c_2)\|LX^*N - A\|_1 \\
&\leq cc_1c_2 \cdot \|LX^*N - A\|_1 + (1 + c_1c_2)\|LX^*N - A\|_1 \\
&\leq c \cdot (1 + 2c_1c_2)\|LX^*N - A\|_1.
\end{aligned}$$

The first inequality directly follows from the previous one. The second inequality follows from the guarantee of $\widehat{X}$. The fourth inequality follows since S has at most c_1 dilation on $LX^*N - A$. The fifth inequality follows since $c \geq 1$. □

Lemma 17.5.4. *Given matrices $L \in \mathbb{R}^{n\times m_1}, N \in \mathbb{R}^{m_2\times d}, A \in \mathbb{R}^{n\times d}, k \geq 1$, let*

$$X^* = \arg\min_{\text{rank}-k\ X} \|LXN - A\|_1.$$

Let $T_1 \in \mathbb{R}^{t_1 \times n}$ have at most c_1-dilation on $LX^*N - A$, and at most c_2-contraction on L. Let

$$\widetilde{X} = \arg \min_{\text{rank}-k \ X} \|T_1 LXN - T_1 A\|_1.$$

Let $T_2^\top \in \mathbb{R}^{t_2 \times d}$ have at most c_1'-dilation on $(T_1 L\widetilde{X}N - T_1 A)^\top$, and at most c_2'-contraction on $N^\top$. Then, for all $c \geq 1$, for any rank $-k$ $\widehat{X} \in \mathbb{R}^{m_1 \times m_2}$ *which satisfies*

$$\|T_1(L\widehat{X}N - SA)T_2\|_1 \leq c \cdot \min_{\text{rank}-k \ X} \|T_1(LXN - A)T_2\|_1,$$

it has

$$\|L\widehat{X}N - A\|_1 \leq c \cdot (2c_1c_2 + 1)(2c_1'c_2' + 1) \cdot \|LX^*N - A\|_1.$$

Proof. Apply Lemma 17.5.3 for sketch matrix T_2. Then for any $c \geq 1$, any rank $-k$ $\widehat{X} \in \mathbb{R}^{m_1 \times m_2}$ which satisfies

$$\|T_1(L\widehat{X}N - A)T_2\|_1 \leq c \cdot \min_{\text{rank}-k \ X} \|T_1(LXN - A)T_2\|_1,$$

has

$$\|T_1(L\widehat{X}N - A)\|_1 \leq c \cdot (2c_1'c_2' + 1) \cdot \|T_1(L\widetilde{X}N - A)\|_1.$$

Apply Lemma 17.5.3 for sketch matrix T_1. Then for any $c \geq 1$, any rank $-k$ $\widehat{X} \in \mathbb{R}^{m_1 \times m_2}$ which satisfies

$$\|T_1(L\widehat{X}N - A)\|_1 \leq c(2c_1'c_2' + 1) \cdot \min_{\text{rank}-k \ X} \|T_1(L\widetilde{X}N - A)\|_1,$$

has

$$\|L\widehat{X}N - A\|_1 \leq c \cdot (2c_1c_2 + 1)(2c_1'c_2' + 1) \cdot \|LX^*N - A\|_1.$$

□

Lemma 17.5.5. *Given matrices $M \in \mathbb{R}^{n\times d}, U \in \mathbb{R}^{n\times t}$, $d \geq t = \text{rank}(U), n \geq d \geq r = \text{rank}(M)$, if sketching matrix $S \in \mathbb{R}^{m\times n}$ is drawn from any of the following probability distributions of matrices, with .99 probability S has at most c_1-dilation on M, i.e.,*

$$\|SM\|_1 \leq c_1\|M\|_1,$$

and S has at most c_2-contraction on U, i.e.,

$$\forall x \in \mathbb{R}^t,\ \|SUx\|_1 \geq \frac{1}{c_2}\|Ux\|_1,$$

where $c_1,\ c_2$ are parameters depend on the distribution over S.

(I) *$S \in \mathbb{R}^{m\times n}$ is a dense Cauchy matrix: a matrix with i.i.d. entries from the standard Cauchy distribution. If $m = O(t\log t)$, then $c_1c_2 = O(\log d)$. If $m = O((t+r)\log(t+r))$, then $c_1c_2 = O(\min(\log d, r\log r))$.*

(II) *$S \in \mathbb{R}^{m\times n}$ is a sparse Cauchy matrix: $S = TD$, where $T \in \mathbb{R}^{m\times n}$ has each column i.i.d. from the uniform distribution on standard basis vectors of $\mathbb{R}^m$, and $D \in \mathbb{R}^{n\times n}$ is a diagonal matrix with i.i.d. diagonal entries following a standard Cauchy distribution. If $m = O(t^5\log^5 t)$, then $c_1c_2 = O(t^2\log^2 t\log d)$. If $m = O((t+r)^5\log^5(t+r))$, then $c_1c_2 = O(\min(t^2\log^2 t\log d, r^3\log^3 r))$.*

(III) *$S \in \mathbb{R}^{m\times n}$ is a sampling and rescaling matrix (notation $S \in \mathbb{R}^{n\times n}$ denotes a diagonal sampling and rescaling matrix with m non-zero entries): If S samples and reweights $m = O(t\log t)$ rows of U, selecting each with probability proportional to the i^{th} row's ℓ_1 Lewis weight and reweighting by the inverse probability, then $c_1c_2 = O(1)$.*

In the above, if we replace S with $\sigma \cdot S$ where $\sigma \in \mathbb{R}\backslash\{0\}$ is any scalar, then the relation between m and $c_1 c_2$ can be preserved.

For (I), if $m = O(t \log t)$, then $c_1 c_2 = O(\log d)$ is implied by Lemma 17.5.15 and Lemma 17.5.7. If $m = O((t+r)\log(t+r))$, $c_1 c_2 = O(r \log r)$ is implied by [SW11].

For (II), if $m = O(t^5 \log^5 t)$, then $c_1 c_2 = O(t^2 \log^2 t \log d)$ is implied by Corollary 17.5.19 and Lemma 17.5.17. If $m = O((t+r)^5 \log^5(t+r))$, $c_1 c_2 = O(r^3 \log^3 r)$ is implied by [MM13].

For (III), it is implied by [CP15] and Lemma 17.5.21.

17.5.3 Cauchy embeddings, no dilation

Lemma 17.5.6. *Define $U^* \in \mathbb{R}^{n\times k}, V^* \in \mathbb{R}^{k\times d}$ to be the optimal solution of $\min_{U\in\mathbb{R}^{n\times k}, V\in\mathbb{R}^{k\times d}} \|UV - A\|_1$. Choose a Cauchy matrix S with m rows and rescaled by $\Theta(1/m)$. We have that*

$$\|SU^*V^* - SA\|_1 \leq O(\log d)\|U^*V^* - A\|_1$$

holds with probability at least 99/100.

Proof. The proof technique has been used in [Ind06] and [CDMI+13]. Fix the optimal U^* and V^*, then

$$\begin{aligned}
\|SU^*V^* - SA\|_1 &= \sum_{i=1}^{d} \|S(U^*V_i^* - A_i)\|_1 \\
&= \sum_{i=1}^{d}\sum_{j=1}^{m} |\sum_{l=1}^{n} \frac{1}{m} S_{j,l} \cdot (U^*V_i^* - A_i)_l| && \text{where } S_{j,l} \sim C(0,1) \\
&= \sum_{i=1}^{d}\sum_{j=1}^{m} \frac{1}{m}|c_{i,j}| && \text{where } c_{i,j} \sim C(0, \|U^*V_i^* - A_i\|_1) \\
&= \frac{1}{m}\sum_{i=1}^{d}\sum_{j=1}^{m} \|U^*V_i^* - A_i\|_1 \cdot w_{i+d(j-1)}. && \text{where } w_{i+d(j-1)} \sim |C(0,1)|
\end{aligned} \tag{17.15}$$

where the last step follows since each w_i can be thought of as a clipped half-Cauchy random variable. Define $d' = md$. Define event ξ_i to be the situation in which $w_i < D$ (we will decide upon D later), and define event $\xi = \xi_1 \cap \xi_2 \cap \cdots \cap \xi_{d'}$. Then it is clear that $\xi \cap \xi_i = \xi, \forall i \in [d']$.

Using the probability density function (pdf) of a Cauchy and because $\tan^{-1} x \leq x$, we can lower bound Pr[event ξ_i holds] in the following sense,

$$\Pr[\xi_i] = \frac{2}{\pi}\tan^{-1}(D) = 1 - \frac{2}{\pi}\tan^{-1}(1/D) \geq 1 - \frac{2}{\pi D}.$$

By a union bound over all $i \in [d']$, we can lower bound Pr[event ξ holds],

$$\Pr[\xi] \geq 1 - \sum_{i=1}^{d'} \Pr[\overline{\xi}_i] \geq 1 - \frac{2d'}{\pi D}. \tag{17.16}$$

By Bayes rule and $\xi = \xi \cap \xi_i$, $\Pr[\xi|\xi_i] \Pr[\xi_i] = \Pr[\xi \cap \xi_i] = \Pr[\xi]$, which implies that $\Pr[\xi|\xi_i] = \Pr[\xi] / \Pr[\xi_i]$. First, we can lower bound $\mathbb{E}[w_i|\xi_i]$,

$$\begin{aligned}
\mathbb{E}[w_i|\xi_i] =& \ \mathbb{E}[w_i|\xi_i \cap \xi] \Pr[\xi|\xi_i] + \mathbb{E}[w_i|\xi \cap \overline{\xi}] \Pr[\overline{\xi}|\xi_i] \\
\geq& \ \mathbb{E}[w_i|\xi_i \cap \xi] \Pr[\xi|\xi_i] && \text{by } w_i \geq 0 \text{ and } \Pr[] \geq 0 \\
=& \ \mathbb{E}[w_i|\xi] \Pr[\xi|\xi_i] && \text{by } \xi = \xi \cap \xi_i.
\end{aligned}$$

The above equation implies that

$$\begin{aligned}
\mathbb{E}[w_i|\xi] \leq& \frac{\mathbb{E}[w_i|\xi_i]}{\Pr[\xi|\xi_i]} \\
=& \frac{\mathbb{E}[w_i|\xi_i] \Pr[\xi_i]}{\Pr[\xi \cap \xi_i]} && \text{by Bayes rule } \Pr[\xi|\xi_i] \Pr[\xi_i] = \Pr[\xi \cap \xi_i] \\
=& \frac{\mathbb{E}[w_i|\xi_i] \Pr[\xi_i]}{\Pr[\xi]} && \text{by } \xi = \xi \cap \xi_i.
\end{aligned}$$

Using the pdf of a Cauchy, $\mathbb{E}[w_i|\xi_i] = \frac{1}{\pi} \log(1 + D^2) / \Pr[\xi_i]$ and plugging it into the lower bound of $\mathbb{E}[w_i|\xi]$,

$$\mathbb{E}[w_i|\xi] \leq \frac{\mathbb{E}[w_i|\xi_i] \Pr[\xi_i]}{\Pr[\xi]} = \frac{\frac{1}{\pi} \log(1 + D^2)}{\Pr[\xi]} \leq \frac{\frac{1}{\pi} \log(1 + D^2)}{1 - \frac{2d}{\pi D}} \lesssim \log(D),$$

where the third step follows since $\Pr[\xi] \geq 1 - \frac{2d'}{\pi D}$ and the last step follows by choosing $D = \Theta(d')$.

We can conclude

$$\begin{aligned}
\mathbb{E}[\|SU^*V^* - SA\|_1|\xi] =& \frac{1}{m} \sum_{i=1}^{d} \sum_{j=1}^{m} \|U^*V_i^* - A_i\|_1 \cdot \mathbb{E}[w_{i+d(j-1)}|\xi] \\
\lesssim& (\log d') \cdot \|U^*V^* - A\|_1.
\end{aligned} \tag{17.17}$$

For simplicity, define $X = \|SU^*V^* - SA\|_1$ and $\gamma = \|U^*V^* - A\|_1$. By Markov's inequality and because $\Pr[X \geq \gamma t|\bar{\xi}] \leq 1$, we have

$$\begin{aligned}
& \Pr[X \geq \gamma t] \\
= \; & \Pr[X \geq \gamma t|\xi] \Pr[\xi] + \Pr[X \geq \gamma t|\bar{\xi}] \Pr[\bar{\xi}] \\
\leq \; & \Pr[X \geq \gamma t|\xi] + \Pr[\bar{\xi}] \\
\leq \; & \frac{\mathbb{E}[X|\xi]}{\gamma t} + \Pr[\bar{\xi}] && \text{by Markov's inequality} \\
\leq \; & \frac{\mathbb{E}[X|\xi]}{\gamma t} + \frac{2d'}{\pi D} && \text{by Equation (17.16)} \\
\lesssim \; & \frac{\log d'}{t} + \frac{2d'}{\pi D} && \text{by Equation (17.17)} \\
\leq \; & .01,
\end{aligned}$$

where choosing $t = \Theta(\log d')$ and $D = \Theta(d')$. Since $k \leq d$ and $m = \text{poly}(k)$, we have $t = \Theta(\log d)$, which completes the proof.

□

Lemma 17.5.7. *Given any matrix $M \in \mathbb{R}^{n \times d}$, if matrix $S \in \mathbb{R}^{m \times n}$ has each entry drawn from an i.i.d. standard Cauchy distribution and is rescaled by $\Theta(1/m)$, then*

$$\|SM\|_1 \leq O(\log d)\|M\|_1$$

holds with probability at least 99/100.

Proof. Just replace the matrix $U^*V^* - A$ in the proof of Lemma 17.5.6 with M. Then we can get the result directly. □

17.5.4 Cauchy embeddings, no contraction

We prove that if we choose a Cauchy matrix S, then for a fixed optimal solution U^* of $\min_{U,V} \|UV - A\|_1$, and for all V, we have that with high probability $\|SU^*V - SA\|_1$ is lower bounded by $\|U^*V - A\|_1$ up to some constant.

Lemma 17.5.8. *Define $U^* \in \mathbb{R}^{n\times k}, V^* \in \mathbb{R}^{k\times d}$ to be the optimal solution of $\min_{U\in\mathbb{R}^{n\times k} V\in\mathbb{R}^{k\times d}} \|UV - A\|_1$, where $A \in \mathbb{R}^{n\times d}$. Let $m = O(k \log k), S \in \mathbb{R}^{m\times n}$ be a random matrix with each entry an i.i.d. standard Cauchy random variable, scaled by $\Theta(1/m)$. Then with probability at least 0.95,*

$$\forall V \in \mathbb{R}^{k\times d}, \|U^*V - A\|_1 \lesssim \|SU^*V - SA\|_1 + O(\log(d))\|U^*V^* - A\|_1.$$

Proof. This follows by Lemmas 17.5.6, 17.5.15, 17.5.1.

□

17.5.5 Cauchy embeddings, k-dimensional subspace

The goal of this section is to prove Lemma 17.5.15.

Before getting into the details, we first give the formal definition of an (α, β) ℓ_1 well-conditioned basis and ϵ-net.

Definition 17.5.7 (ℓ_1 Well-conditioned basis)**.** [DDH+09] A basis U for the range of A is (α, β)-conditioned if $\|U\|_1 \leq \alpha$ and for all $x \in \mathbb{R}^k$, $\|x\|_\infty \leq \beta \|Ux\|_1$. We will say U is well-conditioned if α and β are low-degree polynomials in k, independent of n.

Note that a well-conditioned basis implies the following result.

Fact 17.5.9. *There exist $\alpha, \beta \geq 1$ such that,*

$$\forall x \in \mathbb{R}^k, \frac{1}{k\beta}\|x\|_1 \leq \|Ux\|_1 \leq \alpha \|x\|_1.$$

Proof. The lower bound can be proved in the following sense,

$$\|Ux\|_1 \geq \frac{1}{\beta}\|x\|_\infty \geq \frac{1}{\beta}\frac{1}{k}\|x\|_1,$$

where the first step follows by the properties of a well-conditioned basis, and the second step follows since $k\|x\|_\infty \geq \|x\|_1$. Then we can show an upper bound,

$$\|Ux\|_1 \leq \|U\|_1 \cdot \|x\|_1 \leq \alpha \|x\|_1,$$

where the first step follows by $\|Ux\|_1 \leq \|U\|_1 \|x\|_1$, and the second step follows using $\|U\|_1 \leq \alpha$. □

Definition 17.5.8 (ϵ-net)**.** Define $\mathcal{N}$ to an ϵ-net where, for all the $x \in \mathbb{R}^k$ that $\|x\|_1 = 1$, for any vectors two $x, x' \in \mathcal{N}$, $\|x - x'\|_1 \geq \epsilon$, for any vector $y \notin \mathcal{N}$, there exists an $x \in \mathcal{N}$ such that $\|x - y\|_1 \leq \epsilon$. Then the size of $\mathcal{N}$ is $(\frac{1}{\epsilon})^{O(k)} = 2^{O(k \log(1/\epsilon))}$.

Lemma 17.5.10 (Lemma 6 in [SW11])**.** *There is a constant $c_0 > 0$ such that for any $t \geq 1$ and any constant $c > c_0$, if S is a $t \times n$ matrix whose entries are i.i.d. standard Cauchy random variables scaled by c/t, then, for any fixed $y \in \mathbb{R}^n$,*

$$\Pr[\|Sy\|_1 < \|y\|_1] \leq 1/2^t$$

Lemma 17.5.11. *Suppose we are given a well-conditioned basis $U \in \mathbb{R}^{n \times k}$. If S is a $t \times n$ matrix whose entries are i.i.d. standard Cauchy random variables scaled by $\Theta(1/t)$, then with probability $1 - 2^{-\Omega(t)}$, for all vectors $x \in \mathcal{N}$ we have that $\|SUx\|_1 \geq \|Ux\|_1$.*

Proof. First, using Lemma 17.5.10, we have for any fixed vector $y \in \mathbb{R}^n$, $\Pr[\|Sy\| < \|y\|_1] \leq 1/2^t$. Second, we can rewrite $y = Ux$. Then for any fixed $x \in \mathcal{N}$, $\Pr[\|SUx\| < \|Ux\|_1] \leq 1/2^t$. Third, choosing $t \gtrsim k \log(1/\epsilon)$ and taking a union bound over all the vectors in the ϵ-net $\mathcal{N}$ completes the proof. □

Lemma 17.5.12 (Lemma 7 in [SW11])**.** *Let S be a $t \times n$ matrix whose entries are i.i.d standard Cauchy random variables, scaled by c/t for a constant c, and $t \geq 1$. Then there is a constant $c' = c'(c) > 0$ such that for any fixed set of $\{y_1, y_2, \cdots, y_k\}$ of d vectors in $\{y \in \mathbb{R}^n : x \in \mathbb{R}^k, Ux = y\}$,*

$$\Pr\left[\sum_{i=1}^{k} \|Sy_i\|_1 \geq c' \log(tk) \sum_{i=1}^{k} \|y_i\|_1\right] \leq \frac{1}{1000}.$$

Using Lemma 17.5.12 and the definition of an (α, β) well-conditioned basis, we can show the following corollary.

Corollary 17.5.13. *Suppose we are given an (α, β) ℓ_1 well-conditioned basis $U \in \mathbb{R}^{n \times k}$. Choose S to be an i.i.d. Cauchy matrix with $t = O(k \log k)$ rows, and rescale each entry by $\Theta(1/t)$. Then with probability 99/100, for all vectors $x \in \mathbb{R}^k$,*

$$\|SUx\|_1 \leq O(\alpha\beta \log(k)) \cdot \|Ux\|_1,$$

Proof. Define event E to be the situation when

$$\sum_{i=1}^{k} \|SU_i\|_1 < c' \log(tk) \sum_{i=1}^{k} \|U_i\|_1$$

holds, and U is a well-conditioned basis. Using Lemma 17.5.12, we can show that event E holds with probability 999/1000. We condition on Event E holding. Then for any $y = Ux$ for an $x \in \mathbb{R}^k$, we have

$$\begin{aligned}
\|Sy\|_1 &= \|SUx\|_1 \\
&\leq \sum_{j=1}^{k} \|SU_j x_j\|_1 && \text{by triangle inequality} \\
&= \sum_{j=1}^{k} |x_j| \cdot \|SU_j\|_1 \\
&\leq \|x\|_\infty c' \log(tk) \sum_{j=1}^{k} \|U_j\|_1 && \text{by Lemma 17.5.12} \\
&= \|x\|_\infty c' \log(tk)\alpha && \text{by } \|U\|_1 = \sum_{j=1}^{k} \|U_j\|_1 \leq \alpha \\
&\leq \beta \|Ux\|_1 c' \log(tk)\alpha && \text{by } \|x\|_\infty \leq \beta \|Ux\|_1 \\
&\leq \beta \|y\|_1 c' \log(tk)\alpha && \text{by } Ux = y.
\end{aligned}$$

This completes the proof. □

Lemma 17.5.14. *Given an (α, β) ℓ_1 well-conditioned basis, condition on the following two events,*

1. *For all $x \in \mathcal{N}$, $\|SUx\|_1 \geq \|Ux\|_1$. (Lemma 17.5.10)*
2. *For all $x \in \mathbb{R}^k$, $\|SUx\|_1 \leq O(\alpha\beta \log k)\|Ux\|_1$. (Corollary 17.5.13)*

Then, for all $w \in \mathbb{R}^k$, $\|SUw\|_1 \gtrsim \|Uw\|_1$.

Proof. For any $w \in \mathbb{R}^k$ we can write it as $w = \ell \cdot z$ where ℓ is some scalar and z has $\|z\|_1 = 1$. Define $y = \arg\min_{y' \in \mathcal{N}} \|y' - z\|_1$.

We first show that if U is an (α, β) well-conditioned basis for ℓ_1, then $\|U(y-z)\|_1 \leq \alpha\beta k\epsilon\|Uy\|_1$,

$$\begin{aligned}
&\|U(y-z)\|_1 && \\
&\leq \alpha\|y-z\|_1 && \text{by } \|U(y-z)\|_1 \leq \alpha\|y-z\|_1 \\
&\leq \alpha\epsilon && \text{by } \|y-z\|_1 \leq \epsilon \\
&= \alpha\epsilon\|y\|_1 && \text{by } \|y\|_1 = 1 \\
&\leq \alpha\epsilon\|Uy\|_1\beta k && \text{by } \|Uy\|_1 \geq \frac{1}{\beta k}\|y\|_1.
\end{aligned}$$

Because $\epsilon < 1/(\alpha\beta k^{c+1})$, we have

$$\|U(y-z)\|_1 \leq \frac{1}{k^c}\|Uy\|_1. \tag{17.18}$$

Using the triangle inequality, we can lower bound $\|Uy\|_1$ by $\|Uz\|_1$ up to some constant,

$$\|Uy\|_1 \geq \|Uz\|_1 - \|U(y-z)\|_1 \geq \|Uz\|_1 - \frac{1}{k^c}\|Uy\|_1, \tag{17.19}$$

which implies

$$\|Uy\|_1 \geq .99\|Uz\|_1. \tag{17.20}$$

Thus,

$$\begin{aligned}
&\|SUz\|_1 \\
&\geq \|SUy\|_1 - \|SU(z-y)\|_1 && \text{by triangle inequality} \\
&\geq \|Uy\|_1 - \|SU(z-y)\|_1 && \text{by Lemma 17.5.10} \\
&\geq \|Uy\|_1 - \alpha\beta\log(k)\cdot\|U(z-y)\|_1 && \text{by Corollary 17.5.13} \\
&\geq \|Uy\|_1 - \alpha\beta\log(k)\cdot\frac{1}{k^c}\|Uy\|_1 && \text{by Equation (17.18)} \\
&\gtrsim \|Uy\|_1 && \text{by } k^c \gtrsim \alpha\beta\log(k) \\
&\gtrsim \|Uz\|_1, && \text{by Equation (17.20)}
\end{aligned}$$

by rescaling z to w, we complete the proof. □

Lemma 17.5.15. *Given matrix $U \in \mathbb{R}^{n\times k}$, let $t = O(k\log k)$, and let $S \in \mathbb{R}^{t\times n}$ be a random matrix with entries drawn i.i.d. from a standard Cauchy distribution, where each entry is rescaled by $\Theta(1/t)$. With probability .99,*

$$\forall x \in \mathbb{R}^k, \|SUx\|_1 \gtrsim \|Ux\|_1.$$

Proof. We can compute a well-conditioned basis for U, and denote it U'. Then, $\forall x \in \mathbb{R}^k$, there exists $y \in \mathbb{R}^k$ such that $Ux = U'y$. Due to Lemma 17.5.14, with probability .99, we have

$$\forall y \in \mathbb{R}^k, \|SU'y\|_1 \gtrsim \|U'y\|_1.$$

□

17.5.6 Sparse Cauchy transform

This section presents the proof of two lemmas related to the sparse Cauchy transform. We first prove the no dilation result in Lemma 17.5.16. Then we show how to get the no contraction result in Lemma 17.5.18.

Lemma 17.5.16. *Given matrix $A \in \mathbb{R}^{n \times d}$, let U^*, V^* be the optimal solutions of $\min_{U,V} \|UV - A\|_1$. Let $\Pi = \sigma \cdot SC \in \mathbb{R}^{m \times n}$, where $S \in \mathbb{R}^{m \times n}$ has each column vector chosen independently and uniformly from the m standard basis vectors of $\mathbb{R}^m$, where C is a diagonal matrix with diagonals chosen independently from the standard Cauchy distribution, and σ is a scalar. Then*

$$\|\Pi U^* V^* - \Pi A\|_1 \lesssim \sigma \cdot \log(md) \cdot \|U^* V^* - A\|_1$$

holds with probability at least .999.

Proof. We define $\Pi = \sigma \cdot SC \in \mathbb{R}^{m \times n}$, as in the statement of the lemma. Then,

$$
\begin{aligned}
& \|\Pi(U^*V^* - A)\|_1 \\
= & \sum_{i=1}^{d} \|SD(U^*V_i^* - A_i)\|_1 \\
= & \sum_{i=1}^{d} \left\| \begin{bmatrix} S_{11} & S_{12} & \cdots & S_{1n} \\ S_{21} & S_{22} & \cdots & S_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ S_{m1} & S_{m2} & \cdots & S_{mn} \end{bmatrix} \cdot \begin{bmatrix} c_1 & 0 & 0 & 0 \\ 0 & c_2 & 0 & 0 \\ 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & c_n \end{bmatrix} \cdot (U^*V_i^* - A_i) \right\|_1 \\
= & \sum_{i=1}^{d} \left\| \begin{bmatrix} c_1S_{11} & c_2S_{12} & \cdots & c_nS_{1n} \\ c_1S_{21} & c_2S_{22} & \cdots & c_nS_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ c_1S_{m1} & c_2S_{m2} & \cdots & c_nS_{mn} \end{bmatrix} \cdot (U^*V_i^* - A_i) \right\|_1 \\
= & \sum_{i=1}^{d} \left\| \sum_{l=1}^{n} c_l S_{1l} \cdot (U^*V_i^* - A_i)_l, \sum_{l=1}^{n} c_l S_{2l} \cdot (U^*V_i^* - A_i)_l, \cdots, \sum_{l=1}^{n} c_l S_{ml} \cdot (U^*V_i^* - A_i)_l \right\|_1 \\
= & \sum_{i=1}^{d} \sum_{j=1}^{m} |\sum_{l=1}^{n} c_l S_{jl} \cdot (U^*V_i^* - A_i)_l| \\
= & \sum_{i=1}^{d} \sum_{j=1}^{m} |\widetilde{w}_{ij} \cdot \sum_{l=1}^{n} |S_{jl}(U^*V_i^* - A_i)_l|| \text{ where } \widetilde{w}_{ij} \sim C(0,1) \\
= & \sum_{i=1}^{d} \sum_{j=1}^{m} \sum_{l=1}^{n} |S_{jl}(U^*V_i^* - A_i)_l| \cdot |\widetilde{w}_{ij}| \\
= & \sum_{i=1}^{d} \sum_{j=1}^{m} \sum_{l=1}^{n} |S_{jl}(U^*V_i^* - A_i)_l| \cdot w_{i+(j-1)d} \text{ where } w_{i+(j-1)d} \sim |C(0,1)|,
\end{aligned}
$$

where the last step follows since each w_i can be thought of as a clipped half-Cauchy random variable. Define $d' = md$. Define event ξ_i to be the situation when $w_i < D$ (we will decide upon D later). Define event $\xi = \xi_1 \cap \xi_2 \cap \cdots \cap \xi_{d'}$. By choosing $D = \Theta(d')$, we can conclude

that,

$$\begin{aligned}\mathbb{E}[\|\Pi U^*V^* - \Pi A\|_1|\xi] &= \sum_{i=1}^{d}\sum_{j=1}^{m}\sum_{l=1}^{n}|S_{jl}(U^*V_i^* - A_i)_l| \cdot \mathbb{E}[w_{i+(j-1)d}|\xi] \\ &\lesssim \sum_{i=1}^{d}\sum_{j=1}^{m}\sum_{l=1}^{n}|S_{jl}(U^*V_i^* - A_i)_l| \cdot \log(d') \\ &= \sum_{i=1}^{d}\sum_{l=1}^{n}|(U^*V_i^* - A_i)_l|_1 \cdot \log(d') \\ &= \log(d') \cdot \|U^*V^* - A\|_1.\end{aligned}$$

where the third step follows from $\sum_{j=1}^{m}|S_{jl}| = 1, \forall l \in [n]$.

Thus, we can show that

$$\Pr[\|\Pi U^*V^* - \Pi A\|_1 \lesssim \log(d')\|U^*V^* - A\|_1] \geq 0.999.$$

□

Lemma 17.5.17. *Given any matrix $M \in \mathbb{R}^{n\times d}$. Let $\Pi = \sigma \cdot SC \in \mathbb{R}^{m\times n}$, where $S \in \mathbb{R}^{m\times n}$ has each column vector chosen independently and uniformly from the m standard basis vectors of $\mathbb{R}^m$, where C is a diagonal matrix with diagonals chosen independently from the standard Cauchy distribution, and σ is a scalar. Then*

$$\|\Pi M\|_1 \lesssim \sigma \cdot \log(md) \cdot \|M\|_1$$

holds with probability at least .999.

Proof. Just replace the matrix $U^*V^* - A$ in the proof of Lemma 17.5.16 with M. Then we can get the result directly. □

We already provided the proof of Lemma 17.5.16. It remains to prove Lemma 17.5.18.

Lemma 17.5.18. *Given matrix $A \in \mathbb{R}^{n\times d}$, let U^*, V^* be the optimal solution of $\min_{U,V} \|UV - A\|_1$. Let $\Pi = \sigma \cdot SC \in \mathbb{R}^{m\times n}$, where $S \in \mathbb{R}^{m\times n}$ has each column vector chosen independently and uniformly from the m standard basis vectors of $\mathbb{R}^m$, and where C is a diagonal matrix with diagonals chosen independently from the standard Cauchy distribution. Then with probability at least .999, for all $V \in \mathbb{R}^{k\times d}$,*

$$\|\Pi U^* V - \Pi A\|_1 \geq \|U^* V - A\|_1 - O(\sigma \cdot \log(md))\|U^* V^* - A\|_1.$$

Notice that $m = O(k^5 \log^5 k)$ and $\sigma = O(k^2 \log^2 k)$ according to Theorem 2 in [MM13].

We start by using Theorem 2 in [MM13] to generate the following Corollary.

Corollary 17.5.19. *Given $U \in \mathbb{R}^{n\times k}$ with full column rank, let $\Pi = \sigma \cdot SC \in \mathbb{R}^{m\times n}$ where $S \in \mathbb{R}^{m\times n}$ has each column chosen independently and uniformly from the m standard basis vectors of $\mathbb{R}^m$, and where $C \in \mathbb{R}^{n\times n}$ is a diagonal matrix with diagonals chosen independently from the standard Cauchy distribution. Then with probability .999, for all $x \in \mathbb{R}^k$, we have*

$$\|\Pi U x\|_1 \geq \|U x\|_1.$$

Notice that $m = O(k^5 \log^5 k)$ and $\sigma = O(k^2 \log^2 k)$ according to Theorem 2 in [MM13].

We give the proof of Lemma 17.5.18.

Proof. Follows by Corollary 17.5.19, Lemma 17.5.16, Lemma 17.5.1. □

17.5.7 ℓ_1-Lewis weights

In this section we show how to use Lewis weights to get a better dilation bound. The goal of this section is to prove Lemma 17.5.24 and Lemma 17.5.23. Notice that our algorithms in Section 17.4 use Lewis weights in several different ways. The first way is using Lewis weights to show an existence result, which means we only need an existential result for Lewis weights. The second way is only guessing the nonzero locations on the diagonal of a sampling and rescaling matrix according to the Lewis weights. The third way is guessing the values on the diagonal of a sampling and rescaling matrix according to the Lewis weights. The fourth way is computing the Lewis weights for a known low dimensional matrix($n \times \mathrm{poly}(k)$ or $\mathrm{poly}(k) \times d$). We usually do not need to optimize the running time of computing Lewis weights for a low-rank matrix to have input-sparsity time.

Claim 17.5.20. *Given matrix $A \in \mathbb{R}^{n \times d}$, let $B = U^* V^* - A$. For any distribution $p = (p_1, p_2, \ldots, p_n)$ define random variable X such that $X = \|B_i\|_1 / p_i$ with probability p_i. Then take any m independent samples $X^1, X^2, \ldots, X^m$, let $Y = \frac{1}{m} \sum_{j=1}^m X^j$. We have*

$$\Pr[Y \leq 1000 \|B\|_1] \geq .999$$

Proof. We can compute the expectation of X^j, for any $j \in [m]$

$$\mathbb{E}[X^j] = \sum_{i=1}^n \frac{\|B_i\|_1}{p_i} \cdot p_i = \|B\|_1.$$

Then, $\mathbb{E}[Y] = \frac{1}{m} \sum_{j=1}^m \mathbb{E}[X^j] = \|B\|_1$. Using Markov's inequality, we have

$$\Pr[Y \geq 1000 \|B\|_1] \leq .001$$

□

Lemma 17.5.21. *Given matrix $M \in \mathbb{R}^{n\times d}$, let $S \in \mathbb{R}^{n\times n}$ be any sampling and rescaling diagonal matrix. Then with probability at least .999,*

$$\|SM\|_1 \lesssim \|M\|_1.$$

Proof. Just replace the matrix B in the proof of Claim 17.5.20 with M. Then we can get the result directly. □

Using Theorem 1.1 of [CP15], we have the following result,

Claim 17.5.22. *Given matrix $A \in \mathbb{R}^{n\times d}$, for any fixed $U^* \in \mathbb{R}^{n\times k}$ and $V^* \in \mathbb{R}^{k\times d}$, choose $D \in \mathbb{R}^{n\times n}$ to be the sampling and rescaling diagonal matrix with $m = O(k \log k)$ nonzeros according to the Lewis weights of U^*. Then with probability .999, for all V,*

$$\|U^*V^* - U^*V\|_1 \leq \|DU^*V^* - DU^*V\|_1 \lesssim \|U^*V^* - U^*V\|_1.$$

Lemma 17.5.23. *Given matrix $A \in \mathbb{R}^{n\times d}, U^* \in \mathbb{R}^{n\times k}$, define $V^* \in \mathbb{R}^{k\times d}$ to be the optimal solution of $\min_{V\in\mathbb{R}^{k\times d}} \|U^*V - A\|_1$. Choose a sampling and rescaling diagonal matrix $D \in \mathbb{R}^{n\times n}$ with $m = O(k \log k)$ non-zero entries according to the Lewis weights of U^*. Then with probability at least .99, we have: for all $V \in \mathbb{R}^{k\times d}$,*

$$\|DU^*V - DA\|_1 \lesssim \|U^*V^* - U^*V\|_1 + O(1)\|U^*V^* - A\|_1 \lesssim \|U^*V - A\|_1,$$

holds with probability at least .99.

Proof. Using the above two claims, we have with probability at least .99, for all $V \in \mathbb{R}^{k\times d}$,

$$\begin{aligned}
\|DU^*V - DA\|_1 \leq & \|DU^*V - DU^*V^*\|_1 + \|DU^*V^* - DA\|_1 \\
\lesssim & \|DU^*V - DU^*V^*\|_1 + O(1)\|U^*V^* - A\|_1 \\
\lesssim & \|U^*V - U^*V^*\|_1 + O(1)\|U^*V^* - A\|_1 \\
\leq & \|U^*V - A\|_1 + \|U^*V^* - A\|_1 + O(1)\|U^*V^* - A\|_1 \\
\lesssim & \|U^*V - A\|_1,
\end{aligned}$$

where the first step follows from triangle inequality, the second step follows from Claim 17.5.20, the third step follows from Claim 17.5.22, the forth step follows from triangle inequality. □

Lemma 17.5.24. *Given matrix $A \in \mathbb{R}^{n\times d}$, define $U^* \in \mathbb{R}^{n\times k}, V^* \in \mathbb{R}^{k\times d}$ to be the optimal solution of $\min_{U\in\mathbb{R}^{n\times k}, V\in\mathbb{R}^{k\times d}} \|UV - A\|_1$. Choose a sampling and rescaling diagonal matrix $D \in \mathbb{R}^{n\times n}$ with $m = O(k \log k)$ non-zero entries according to the Lewis weights of U^*. For all $V \in \mathbb{R}^{k\times d}$ we have*

$$\|U^*V - A\|_1 \lesssim \|DU^*V - DA\|_1 + O(1)\|U^*V^* - A\|_1,$$

holds with probability at least .99.

Proof. Follows by Claim 17.5.20, Lemma 17.5.23, Lemma 17.5.1.

□

17.6 Hardness Results for Cauchy Matrices, Row Subset Selection, OSE

Section 17.6.1 presents some inapproximability results by using random Cauchy matrices. Section 17.6.2 is a warmup for inapproximability results for row subset selection problems. Section 17.6.3 shows the inapproximability results by using any linear oblivious subspace embedding(OSE), and also shows inapproximability results for row subset selection.

17.6.1 Hard instance for Cauchy matrices

The goal of this section is to prove Theorem 17.6.2. Before stating the result, we first introduce some useful tools in our analysis.

Lemma 17.6.1 (Cauchy Upper Tail Inequality, Lemma 3 of [CDMI+13]). *For $i \in [m]$, let C_i be m random Cauchy variables from $C(0,1)$ (not necessarily independent), and $\gamma_i > 0$ with $\gamma = \sum_{i\in[m]} \gamma_i$. Let $X = \sum_{i\in[m]} \gamma_i |C_i|$. Then, for any $t \geq 1$,*

$$\Pr[X > \gamma t] \leq O(\log(mt)/t).$$

Theorem 17.6.2. *Let $k \geq 1$. There exist matrices $A \in \mathbb{R}^{d\times d}$ such that for any $o(\log d) \geq t \geq 1$, where c can be any constant smaller than $1/3$, for random Cauchy matrices $S \in \mathbb{R}^{t\times d}$ where each entry is sampled from an i.i.d. Cauchy distribution $C(0,\gamma)$ where γ is an arbitrary real number, with probability .99 we have*

$$\min_{U\in\mathbb{R}^{d\times t}} \|USA - A\|_1 \geq \Omega(\log d/(t\log t)) \min_{\text{rank}-k\ A'} \|A' - A\|_1.$$

Proof. We define matrix $A \in \mathbb{R}^{d\times d}$

$$A = B + I = \alpha \cdot \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix},$$

where $\alpha = \Theta(\log d)$.So, if we only fit the first row of A, we can get approximation cost at most $O(d)$.

For $t > 0$, let $S \in \mathbb{R}^{t\times d}$ denote a random Cauchy matrix where $S_{i,j}$ denotes the entry

in the ith row and jth column. Then SA is

$$SA = SB + SI = \alpha \cdot \begin{bmatrix} S_{1,1} & S_{1,1} & S_{1,1} & \cdots & S_{1,1} \\ S_{2,1} & S_{2,1} & S_{2,1} & \cdots & S_{2,1} \\ S_{3,1} & S_{3,1} & S_{3,1} & \cdots & S_{3,1} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ S_{t,1} & S_{t,1} & S_{t,1} & \cdots & S_{t,1} \end{bmatrix} + \begin{bmatrix} S_{1,1} & S_{1,2} & S_{1,3} & \cdots & S_{1,d} \\ S_{2,1} & S_{2,2} & S_{2,3} & \cdots & S_{2,d} \\ S_{3,1} & S_{3,2} & S_{3,3} & \cdots & S_{3,d} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ S_{t,1} & S_{t,2} & S_{t,3} & \cdots & S_{t,d} \end{bmatrix}.$$

Since $\forall \gamma$,

$$\min_{U \in \mathbb{R}^{d \times t}} \|USA - A\|_1 = \min_{U \in \mathbb{R}^{d \times t}} \|\gamma USA - A\|_1,$$

without loss of generality, we can let $S_{i,j} \sim C(0,1)$. Then we want to argue that, if we want to use SA to fit the first row of A, with high probability, the cost will be $\Omega(d \log d)$.

Let $b \in \mathbb{R}^d$ denote the first row of A. Then, we want to use SA to fit the first row of A, which is a d-dimensional vector that has entry $\Theta(\log d)$ on each position. The problem is equivalent to

$$\min_{x \in \mathbb{R}^t} \|(SA)^\top x - b\|_1.$$

First, we want to show that for any x in $\mathbb{R}^t$, if $x^\top SA$ fits the first row of A very well, then the ℓ_1 and ℓ_2 norm of vector x must have reasonable size.

Claim 17.6.3. *Define A^1 to be the first row of matrix A. With probability .999, for any column vector $x \in \mathbb{R}^{t \times 1}$, if $\|x^\top SA - A^1\|_1 \leq o(d \log d)$, then*

Property (I) : $\|x\|_2 \geq \Omega(1/t \log t)$;

Property (II) : $\|x\|_1 \leq O(\log d)$.

Proof. Consider the absolute value of the i-th coordinate of $x^\top SA$. We can rewrite $|\langle (SA)_i, x\rangle|$ in the following sense,

$$\begin{aligned} |\langle (SA)_i, x\rangle| &= |\langle (SB)_i, x\rangle + \langle (SI)_i, x\rangle| \\ &= |\langle (SB)_1, x\rangle + \langle (SI)_i, x\rangle| \\ &= |\langle (SB)_1, x\rangle + \langle S_i, x\rangle|, \end{aligned} \tag{17.21}$$

where the second step follows because $(SB)_i = (SB)_1, \forall i \in [n]$, and the last step follows because $(SI)_i = S_i, \forall i \in [n]$.

We start by proving Property I. Using the triangle inequality and Equation (17.21),

$$\begin{aligned} & |\langle (SA)_i, x\rangle| \\ \leq\ & |\langle (SB)_1, x\rangle| + |\langle S_i, x\rangle| \\ \leq\ & \|(SB)_1\|_2 \|x\|_2 + \|S_i\|_2 \|x\|_2 && \text{by Cauchy-Schwarz inequality} \\ \leq\ & \|(SB)_1\|_1 \|x\|_2 + \|S_i\|_1 \|x\|_2. && \text{by } \|\cdot\|_2 \leq \|\cdot\|_1 \end{aligned}$$

Then, according to Lemma 17.6.1, with probability .99999, we have $\|(SB)_1\|_1 \leq O(t \log t \log d)$ and for a fixed $i \in [d]$, $\|S_i\|_1 \leq O(t \log t)$. Applying the Chernoff bound, with probability $1 - 2^{-\Omega(d)}$, there are a constant fraction of i such that $\|S_i\|_1 = O(t \log t)$. Taking the union bound, with probability .9999, there exists a constant fraction of i such that $|\langle (SA)_i, x\rangle| \leq O(t \log t \log d)\|x\|_2$. Because $A_{1,i} \geq \alpha$, $\forall i \in [d]$ where $\alpha = \Theta(\log d)$, we need $\|x\|_2 \geq \Omega(1/t \log t)$. Otherwise, the total cost on this constant fraction of coordinates will be at least $\Omega(d \log d)$.

For Property II, for a fixed $x \in \mathbb{R}^t$, we have

$$\begin{aligned}
&|\langle (SA)_i, x\rangle| \\
=&|\langle (SB)_1, x\rangle + \langle S_i, x\rangle| && \text{by Equation (17.21)} \\
=&\left|\Theta(\log d)\|x\|_1 w_1'(x) + \|x\|_1 w_i'(x)\right|.
\end{aligned}$$

where $w_i'(x) \sim C(0,1)$, and for different x, $w_i'(x)$ are different. Then for a fixed $x \in \mathbb{R}^t$ with probability at least 0.9, $|w_i'(x)| = \Omega(1)$, and with probability 0.5, $w_1'(x)$ and $w_i'(x)$ have the same sign. Since these two events are independent, with probability at least 0.45, we have

$$|\langle (SA)_i, x\rangle| \geq \|x\|_1 \cdot \Omega(1).$$

Applying the Chernoff bound, with probability at least $1 - 2^{\Theta(d)}$, there exists a 3/10 fraction of i such that $|\langle (SA)_i, x\rangle| \geq \|x\|_1 \Omega(1)$.

We build an ϵ-net $\mathcal{N}$ for $x \in \mathbb{R}^t$ on an ℓ_1-norm unit ball, where $\epsilon = 1/(t^2 \log^2 d)$. Thus the size of the net is $|\mathcal{N}| = 2^{\widetilde{\Theta}(t)}$. Consider y to be an arbitrary vector, let $y/\|y\|_1 = x + \delta$, where x is the closest point to $y/\|y\|_1$ and $x \in \mathcal{N}$. For any $\delta \in \mathbb{R}^t$,

$$\begin{aligned}
|\langle (SA)_i, \delta\rangle| &= |\langle (SB)_1, \delta\rangle + \langle (SI)_i, \delta\rangle| \\
&\leq |\langle (SB)_1, \delta\rangle| + |\langle (SI)_i, \delta\rangle| && \text{by triangle inequality} \\
&\leq \|(SB)_1\|_2 \|\delta\|_2 + \|S_i\|_2 \|\delta\|_2 && \text{by Cauchy-Schwarz inequality} \\
&\leq \|(SB)_1\|_2 \|\delta\|_1 + \|S_i\|_2 \|\delta\|_1. && \text{by } \|\cdot\|_2 \leq \|\cdot\|_1
\end{aligned}$$

As we argued before, With probability .99999, we have

$$\|(SB)_1\|_2 \leq \|(SB)_1\|_1 \leq O(t \log t \log d), \tag{17.22}$$

and with probability $1-2^{-\Theta(d)}$, there is a $9/10$ fraction of $i \in [d]$, $\|S_i\|_2 \leq \|S_i\|_1 = O(t \log t)$. Therefore, with probability .999, for any $\delta \in \mathbb{R}^t$, there exists a $9/10$ fraction of i such that $|\langle (SA)_i, \delta \rangle| \leq \Theta(t \log t \log d)\|\delta\|_1$.

Therefore, with probability .99, $\forall y \in \mathbb{R}^t$, due to the pigeonhole principle, there is a $3/10 + 9/10 - 1 = 1/5$ fraction of i such that

$$
\begin{aligned}
& |\langle (SA)_i, y \rangle| \\
= & \|y\|_1 \cdot |\langle (SA)_i, y/\|y\|_1 \rangle| \\
= & \|y\|_1 \cdot |\langle (SA)_i, x + \delta \rangle| && \text{by } y/\|y\|_1 = x + \delta \\
\geq & \|y\|_1 \cdot \big(|\langle (SB)_i, x \rangle + \langle (SI)_i, x \rangle| - |\langle (SB)_1, \delta \rangle + \langle (SI)_i, \delta \rangle| \big) && \text{by triangle inequality} \\
\geq & \|y\|_1 \big(\Omega(1) - \epsilon O(t \log t \log d) \big) \\
\geq & \|y\|_1 \Omega(1).
\end{aligned}
$$

So $\|y\|_1$ should be $O(\log d)$. Otherwise, the total cost on this $1/5$ fraction of coordinates is at least $\Omega(d \log d)$.

Combining Property (I) and (II) completes the proof. □

Next, we need to show the following claim is true,

Claim 17.6.4. *For any d independent Cauchy random variables $x_1, x_2, \cdots, x_d$ from $C(0,1)$, with probability $1 - 1/\operatorname{poly}(t \log d)$, for any $j \in [1, 2, \cdots, \log d - \Theta(\log \log(t \log d))]$, there are $\Omega(d/2^j)$ variables belonging to $(2^j, 2^{j+1}]$.*

Proof. For each Cauchy random variable x_i, we have for any $j \in [1, 2, \cdots, \Theta(\log d)]$,

$$
\Pr\Big[|x_i| \in (2^j, 2^{j+1}] \Big] = \Theta(1/2^j).
$$

We define the indicator random variable $z_{i,j}$

$$z_{i,j} = \begin{cases} 1 & \text{if } |x_i| \in (2^j, 2^{j+1}], \\ 0 & \text{otherwise.} \end{cases}$$

We define $z_j = \sum_{i=1}^{d} z_{i,j}$. It is clear that $\mathbb{E}[z_{i,j}] = \Theta(1/2^j)$. We use a Chernoff bound,

$$\Pr[X < (1-\delta)\mu] < \left(\frac{e^{-\delta}}{(1-\delta)^{1-\delta}} \right)^{\mu},$$

and set $X = z_j$, $\delta = 1/2$, $\mu = \mathbb{E}[z_j]$. Then, this probability is at most $2^{-\Omega(\mu)} = 2^{-\Omega(\mathbb{E}[z_j])}$. For any $j \in [1, \log d - \Theta(\log\log(t \log d))]$, we have $\mathbb{E}[z_j] = d\,\mathbb{E}[z_{ij}] = \Omega(d/2^j) = \Omega(\log(t \log d))$. Thus, this probability is at most $1/\operatorname{poly}(t \log d)$. Overall, we have

$$\Pr\left[z_j \gtrsim d/2^j \right] \geq 1 - 1/\operatorname{poly}(t \log d).$$

Taking a union bound over $\Theta(\log d)$ such j, we complete the proof.

□

Claim 17.6.5. *Let $1 > c_1 > c_2 > 1/3 > 0$ be three arbitrary constants. We fix the first column of $(SI)^\top \in \mathbb{R}^{d \times t}$. All the rows are grouped together according to the value in the first column. Let R_j be the set of rows for which the entry in the first column is $\in (2^j, 2^{j+1}]$. With probability at least $1 - O(1/\operatorname{poly}(t \log(d)))$, for any $j \in [c_2 \log d, c_1 \log d]$, the following event holds. There exist $\Omega(d/2^j)$ rows such that the first coordinate $\in (2^j, 2^{j+1}]$ and all the other coordinates are at most $O(d^{1/3})$.*

Proof. Let R_j be a subset of rows of $S^\top$, such that for any row in R_j, the first coordinate is $\in (2^j, 2^{j+1}]$. By Claim 17.6.4, we have that with probability $1 - 1/\operatorname{poly}(t \log d)$, for any $j \in [c_2 \log d, c_1 \log d]$, $|R_j| \geq \Omega(d/2^j)$. We want to show that for any $j \in [c_2 \log d, c_1 \log d]$,

there exists a constant fraction of rows in R_j such that, the remaining coordinates are at most $O(d^{1/3})$.

For a fixed row in R_j and a fixed coordinate in that row, the probability that the absolute value of this entry is at least $\Omega(d^{1/3})$ is at most $O(1/d^{1/3})$. By taking a union bound, with probability at least $1 - O(t/d^{1/3})$, the row has every coordinate of absolute value at most $O(d^{1/3})$.

By applying a Chernoff bound, for a fixed subset R_j of rows, the probability that there is a constant fraction of these rows for which every coordinate except the first one in absolute value is at most $O(d^{1/3})$, is at least $1 - 2^{-\Theta(|R_j|)} \geq 1 - 2^{-\Theta(d^{1-c_1})} \geq 1 - O(1/\operatorname{poly}(t \log(d)))$.

After taking a union over all the j, we complete the proof. □

Claim 17.6.6. *With probability at least $1 - O(1/t)$, the absolute value of any coordinate in column vector $(SB)_1 \in \mathbb{R}^{t \times 1}$ is at most $O(t^2 \log d)$.*

Proof. Because each entry is sampled from a Cauchy distribution $C(0, 1)$ scaled by $O(\log d)$, then with probability at most $O(1/t^2)$, the absolute value of one coordinate is at least $\Omega(t^2 \log d)$. Taking a union over all the coordinates, we complete the proof. □

Because $\|x\|_1 \leq O(\log d), \|x\|_2 \geq \Omega(1/t \log t)$, there exists one coordinate i such that the absolute value of it is at least $\Omega(1/t \log t)$ and all the other coordinates are most $O(\log d)$. We can assume that $i = 1$ for now. (To remove this assumption, we can take a union bound over all the possibilities for $i \in [t]$.) According to Claim 17.6.5 and 17.6.6, let $\widehat{R}_j$ denote a subset of R_j, which is a "good" constant fraction of R_j. Considering the ℓ_1-norm of all

coordinates $l \in \widehat{R}_j \subset R_j \subset [d]$, we have

$$\begin{aligned}
&\sum_{l\in\widehat{R}_j} |((SA)^\top x)_l - O(\log d)| \\
\geq &\sum_{l\in\widehat{R}_j} |\langle (SA)_l, x\rangle - O(\log d)| \\
\geq &\sum_{l\in\widehat{R}_j} \Big(|(SI)_{1,l}\cdot x_1| - \sum_{j=2}^{t} |(SI)_{j,l}\cdot x_j| - |\langle (SB)_l, x\rangle| - O(\log d) \Big) \\
\geq &\sum_{l\in\widehat{R}_j} \Big(\Omega(\frac{2^j}{t\log t}) - O(td^{1/3}\log d) - |\langle (SB)_1, x\rangle| - O(\log d) \Big) \\
\geq &\sum_{l\in\widehat{R}_j} \Big(\Omega(\frac{2^j}{t\log t}) - O(td^{1/3}\log d) - \|(SB)_1\|_1 \|x\|_1 - O(\log d) \Big) \\
\geq &\sum_{l\in\widehat{R}_j} \Big(\Omega(\frac{2^j}{t\log t}) - O(td^{1/3}\log d) - O(t\log t\log^2 d) - O(\log d) \Big) \\
\gtrsim &\sum_{l\in\widehat{R}_j} 2^j/(t\log t) \\
\gtrsim &\, d/2^j \cdot 2^j/(t\log t) \\
\gtrsim &\, d/(t\log t).
\end{aligned}$$

The second inequality follows by the triangle inequality. The third inequality follows by $(SB)_1 = (SB)_l$, $|x_1| = \Omega(1/t\log t), (SI)_{1,l} \in [2^j, 2^{j+1})$, and $\forall j \neq 1, |x_j| < O(\log d), (SI)_{j,l} \leq O(d^{1/3})$. The fourth inequality follows by Cauchy-Schwarz and $\|\cdot\|_2 \leq \|\cdot\|_1$. The fifth inequality follows by Equation (17.22) and Claim 17.6.3. The sixth inequality follows by $t = o(\log d)$ where c is a constant smaller than $1/3$ and $2^j \geq d^{c_2} > \mathrm{poly}(t)$. The seventh inequality follows from $|\widehat{R}_j| \geq \Omega(d/2^j)$.

Since there are $c1 - c2$ different j, the total cost is $\Omega(d\log d/(t\log t))$. The gap then is $\Omega(\log d/(t\log t))$. This completes the proof of Theorem 17.6.2. □

17.6.2 Hard instance for row subset selection

Theorem 17.6.7. *Let $\epsilon \in (0,1)$. There exists a value $k \geq 1$ and matrix $A \in \mathbb{R}^{(d-1)\times d}$ such that, for any subset R of rows of A, letting B be the ℓ_1-projection of each row of A onto R, we have*

$$\|A - B\|_1 > (2-\epsilon) \min_{\text{rank}-k \ A'} \|A' - A\|_1,$$

unless $|R| \gtrsim \epsilon d$.

Proof. We construct the $(d-1) \times d$ matrix A in the following way. The first column is all 1s, and then the remaining $(d-1) \times (d-1)$ matrix is the identity matrix.

$$A = \begin{bmatrix} 1 & 1 & 0 & 0 & \cdots & 0 \\ 1 & 0 & 1 & 0 & \cdots & 0 \\ 1 & 0 & 0 & 1 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & 0 & 0 & 0 & \cdots & 1 \end{bmatrix}.$$

Note that $\min_{\text{rank}-1 \ A'} \|A - A'\|_1 = \text{OPT} \leq d-1$, since one could choose A' to have the first column all 1s and remaining entries 0. On the other hand, consider any subset of r rows. We can permute the columns and preserve the entrywise ℓ_1-norm so w.l.o.g., we can take the first r rows for R.

Because we are taking the first r rows, we do not pay any cost on the first rows. To minimize the cost on the i-th row, where $i \in \{r+1, r+2, \ldots, d\}$, let v_i denote the row vector we use for the i-th row. Then v_i can be written as a linear combination of the first r rows $\{A^1, A^2, \ldots, A^r\}$,

$$v_i = \sum_{j=1}^{r} \alpha_{i,j} A^j.$$

Then the cost of using v_i to approximate the i-th row of A is:

$$\begin{aligned}
\|v_i - A^i\|_1 = & \text{ (cost in 1st col) + (cost in 2nd,3rd, } \ldots, r+1\text{th cols) + (cost in } i+1\text{th col)} \\
= & \ |\sum_{j=1}^{r} \alpha_{i,j} - 1| + \sum_{j=1}^{r} |\alpha_{i,j}| + 1 \\
\geq & \ |\sum_{j=1}^{r} \alpha_{i,j} - 1 - \sum_{j=1}^{r} \alpha_{i,j}| + 1 \text{ by triangle inequality} \\
= & \ 2.
\end{aligned}$$

Hence, the cost of using v_i to approximate the i-th row of A is at least 2. So in total, across these $(d-1-r)$ rows, the algorithm pays at least $2(d-1-r)$ cost, which needs to be at most $C(d-1)$, and therefore $r \geq (d-1)(1-C/2) = \Omega(\epsilon d)$. Choosing $C = 2-\epsilon$ completes the proof.

□

Theorem 17.6.8. *There exists a value $k \geq 1$ and matrix $A \in \mathbb{R}^{(d-1)\times d}$ such that, there is no algorithm that is able to output a* rank $-k$ *matrix B in the row span of A satisfying*

$$\|A - B\|_1 < 2(1 - \Theta(\frac{1}{d})) \min_{\text{rank}-k \ A'} \|A' - A\|_1.$$

Proof. We use the same matrix A as in the previous theorem.

$$A = \begin{bmatrix} 1 & 1 & 0 & 0 & \cdots & 0 \\ 1 & 0 & 1 & 0 & \cdots & 0 \\ 1 & 0 & 0 & 1 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & 0 & 0 & 0 & \cdots & 1 \end{bmatrix}.$$

Let vector v be $(\sum_{i=1}^{d-1} \beta_i, \beta_1, \ldots, \beta_{d-1})$. For the i-th row of A, we use $\alpha_j \cdot v$ to cancel the cost of that row, where α_j is a scalar. Then for any $\beta_1, \ldots, \beta_{d-1}, \alpha_1, \ldots, \alpha_{d-1}$, we can

compute the entire residual cost,

$$f(\alpha,\beta) = \sum_{j=1}^{d-1} \|A^j - v\alpha_j\|_1$$
$$= \sum_{j=1}^{d-1} \left(|1-\alpha_j(\sum_{i=1}^{d-1}\beta_i)| + |1-\alpha_j\beta_j| + \sum_{i\neq j}|\alpha_j\beta_i| \right).$$

In the next few paragraphs, we will show that optimizing $f(\alpha,\beta)$ over some extra constraints for α,β does not change the optimality. Without loss of generality, we can assume that $\sum_{i=1}^{d-1}\beta_i = 1$. If not we can just rescale all the α_i. Consider a fixed index j. All the terms related to α_j and β_j are,

$$\left|1-\alpha_j(\sum_{i=1}^{d-1}\beta_i)\right| + |1-\alpha_j\beta_j| + \sum_{i\neq j}|\alpha_j\beta_i| + |\alpha_i\beta_j|.$$

We first show a simple proof by assuming $\beta_j \geq 0$. Later, we will prove the general version which does not have any assumptions.

Handling a special case We can optimize $f(\alpha,\beta)$ in the following sense,

$$\begin{aligned} &\min\ f(\alpha,\beta) \\ &\text{s.t.}\ \sum_{j=1}^{d-1}\beta_j = 1 \\ &\quad \alpha_j \geq 0, \beta_j \geq 0, \forall j \in [d-1]. \end{aligned}$$

For each j, we consider three cases. Case I, if $\alpha_j \geq \frac{1}{\beta_j}$, then

$$|1-\alpha_j| + |1-\alpha_j\beta_j| + \sum_{i\neq j}\alpha_j\beta_i = \qquad \alpha_j - 1 + \alpha_j\beta_j - 1 + \sum_{i\neq j}\alpha_j\beta_i$$
$$= 2\alpha_j - 2 \geq 2(1/\beta_j - 1) \geq 2(1-\beta_j),$$

where the last step follows by $\beta_j + 1/\beta_j \geq 2$. Case II, if $1 \leq \alpha_j < 1/\beta_j$, then

$$|1-\alpha_j| + |1-\alpha_j\beta_j| + \sum_{i\neq j} \alpha_j\beta_i = \qquad \alpha_j - 1 + 1 - \alpha_j\beta_j + \sum_{i\neq j} \alpha_j\beta_i$$
$$= 2\alpha_j(1-\beta_j) \geq 2(1-\beta_j).$$

Case III, if $\alpha_j < 1$, then

$$|1-\alpha_j| + |1-\alpha_j\beta_j| + \sum_{i\neq j} \alpha_j\beta_i = \qquad 1 - \alpha_j + 1 - \alpha_j\beta_j + \sum_{i\neq j} \alpha_j\beta_i$$
$$= 2(1-\alpha_j\beta_j) \geq 2(1-\beta_j).$$

Putting it all together, we have

$$f(\alpha,\beta) \geq \sum_{j=1}^{d-1} 2(1-\beta_j) = 2(d-1) - 2\sum_{j=1}^{d-1}\beta_j = 2(d-2).$$

To handle the case where β_i can be negative Without loss of generality, we can assume that $\sum_{i=1}^{d-1}\beta_i = 1$. Notice that we can also assume $\beta_i \neq 0$, otherwise it means we do not choose that row. We split all β_i into two disjoint sets S and $\overline{S}$. For any $i \in S$, $\beta_i > 0$ and for any $i \in \overline{S}$, $\beta_i < 0$.

As a first step, we discuss the case when all the j are in set S. Case I, if $1 - \alpha_j\sum_{i=1}^{d-1}\beta_i < 0$ and $1-\alpha_j\beta_j < 0$, then it means $\alpha_j \geq \max(\frac{1}{\sum_{i=1}^{d-1}\beta_i}, \frac{1}{\beta_j})$. The cost of that row is,

$$= \alpha_j \sum_{i=1}^{d-1}\beta_i - 1 + \alpha_j\beta_j - 1 + \sum_{i\neq j}|\alpha_j\beta_i|$$
$$= 2\alpha_j\beta_j + 2\alpha_j \sum_{i\in S\setminus j}\beta_i - 2.$$

If $\beta_j \geq 1$, then $\alpha_j \geq 1$. The cost is at least $2\sum_{i\in S\setminus j}\beta_i$. If $\beta_j < 1$, then $\alpha_j \geq 1/\beta_j$, and the cost is at least $2\frac{1}{\beta_j}\sum_{i\in S\setminus j}\beta_i$. If there are C such j in Case I, then the total cost of Case I is at least 0 if $C = 1$, and $2(C-1)$ if $C \geq 2$.

Case II, if $1 - \alpha_j \sum_{i=1}^{d-1}\beta_i < 0$ and $1 - \alpha_j\beta_j > 0$, then it means $1 < \alpha_j < 1/\beta_j$. The cost of that row is,

$$\begin{aligned}
&= \alpha_j \sum_{i=1}^{d-1}\beta_i - 1 + 1 - \alpha_j\beta_j + |\alpha_j| \sum_{i\neq j}|\beta_j| \\
&= 2\alpha_j \sum_{i\in S\setminus j}\beta_i \\
&\geq 2\sum_{i\in S\setminus j}\beta_i.
\end{aligned}$$

Similarly to before, if there are C such j in Case II, then the total cost of Case II is at least 0 if $C = 1$, and $2(C-1)$ if $C \geq 2$.

Case III, if $1 - \alpha_j \sum_{i=1}^{d-1}\beta_i > 0$ and $1 - \alpha_j\beta_j < 0$, then it means $1/\beta_j < \alpha_j < \frac{1}{\sum_{i=1}^{d-1}\beta_i}$. The cost of the row is,

$$\begin{aligned}
&= 1 - \alpha_j \sum_{i=1}^{d-1}\beta_i + \alpha_j\beta_j - 1 + |\alpha_j| \sum_{i\neq j}|\beta_j| \\
&= 2|\alpha_j| \sum_{i\in\overline{S}}|\beta_i| \\
&= 2|\alpha_j|(\sum_{i\in S}|\beta_i| - 1) \\
&\geq 2\frac{1}{\beta_j}\sum_{i\in S\setminus j}\beta_i.
\end{aligned}$$

If there are C such j in Case III, then the total cost of Case III is at least 0 if $C = 1$, and $2(C\cdot(C-1))$ if $C \geq 2$.

Case IV, if $1-\alpha_j \sum_{i=1}^{d-1} \beta_i > 0$ and $1-\alpha_j\beta_j > 0$, then it means $\alpha_j \leq \min(1/\beta_j, \frac{1}{\sum_{i=1}^{d-1}\beta_i})$. The cost for the case $\alpha_j < 0$ is larger than the cost for case $\alpha > 0$. Thus we can ignore the case $\alpha_j < 0$. The cost of the row is,

$$\begin{aligned}
&= 1 - \alpha_j \sum_{i=1}^{d-1} \beta_i + 1 - \alpha_j\beta_j + |\alpha_j| \sum_{i \neq j} |\beta_j| \\
&= 2 - 2\alpha_j\beta_j + 2\alpha_j \sum_{i \in \overline{S}} |\beta_i|.
\end{aligned}$$

If $\beta_j < \sum_{i\in\overline{S}} |\beta_i|$, we know that the cost is at least 2. Otherwise, using $\alpha_j \leq 1$, we have the cost is at least $2 - 2\beta_j + \sum_{i\in\overline{S}} |\beta_i|$. Let T denote the set of those j with $\beta_j \geq \sum_{i\in\overline{S}} |\beta_i|$. If $|T| = 1$, we know that cost is at least 0. If $|T| \geq 2$, then the cost is at least

$$\begin{aligned}
&= \sum_{j\in T} (2 - 2\beta_j + 2\sum_{i\in\overline{S}} |\beta_i|) \\
&\geq 2|T| - 2\sum_{j\in S} \beta_j + 2|T| \sum_{i\in\overline{S}} |\beta_i| \\
&\geq 2|T| - 2 + 2(|T|-1) \sum_{i\in\overline{S}} |\beta_i| \\
&\geq 2(C-1).
\end{aligned}$$

Now we discuss the case where $j \in \overline{S}$, which means $\beta_j < 0$.

Case V if $1 - \alpha_j \sum_{i=1}^{d-1} \beta_i < 0$ and $1 - \alpha_j\beta_j < 0$, then it means $\alpha_j > 1/\sum_{i=1}^{d-1} \beta_i$ and $\alpha_j < 1/\beta_j$. Notice that this case will never happen, because $\sum_{i=1}^{d-1} \beta_i = 1$ and $\alpha_j < 0$.

Case VI if $1-\alpha_j \sum_{i=1}^{d-1} \beta_i < 0$ and $1-\alpha_j\beta_j < 0$, then it means $\alpha_j \geq \max(1/\sum_{i=1}^{d-1} \beta_i, 1/\beta_j)$.

Because of $\beta_j < 0$, then $\alpha_j \geq 1$. The cost of that is,

$$
\begin{aligned}
&= \alpha_j \sum_{i=1}^{d-1} \beta_i - 1 + 1 - \alpha_j \beta_j + |\alpha_j| \sum_{i \neq j} |\beta_i| \\
&= 2\alpha_j \sum_{i \in S} |\beta_i| \\
&\geq 2. && \text{by } \alpha_j \geq 1 \text{ and } \sum_{i \in S} |\beta_i| \geq 1.
\end{aligned}
$$

Case VII if $1 - \alpha_j \sum_{i=1}^{d-1} \beta_i > 0$ and $1 - \alpha_j \beta_j < 0$, then it means $\alpha_j < \min(1/\beta_j, 1/\sum_{i=1}^{d-1} \beta_i)$. Because $\beta_j < 0$ and $\sum_{i=1}^{d-1} \beta_i = 1$, thus $\alpha_j < 1/\beta_j$. The cost of that row is,

$$
\begin{aligned}
&= 1 - \alpha_j \sum_{i=1}^{d-1} \beta_i + \alpha_j \beta_j - 1 + |\alpha_j| \sum_{i \neq j} |\beta_i| \\
&= 2|\alpha_j| \sum_{i \in \overline{S} \setminus j} |\beta_i| \\
&\geq 2 \frac{1}{|\beta_j|} \sum_{i \in \overline{S} \setminus j} |\beta_i|.
\end{aligned}
$$

If there are C such j in Case VII, then the total cost of Case VII is at least 0 if $C = 1$, and $2(C \cdot (C-1))$ if $C \geq 2$.

Case VIII if $1 - \alpha_j \sum_{i=1}^{d-1} \beta_i > 0$ and $1 - \alpha_j \beta_j > 0$, then it means $1/\beta_j < \alpha_j < 1/\sum_{i=1}^{d-1} \beta_i$. The cost of that row,

$$
\begin{aligned}
&= 1 - \alpha_j \sum_{i=1}^{d-1} \beta_i + 1 - \alpha_j \beta_j + |\alpha_j| \sum_{i \neq j} |\beta_i| \\
&= 2 - 2\alpha_j \beta_j + 2|\alpha_j| \sum_{i \in \overline{S} \setminus j} |\beta_i|.
\end{aligned}
$$

If $\alpha_j > 0$, then the cost is always at least 2. If $\alpha_j < 0$, the cost is at least,

$$
2 - 2|\alpha_j \beta_j| + 2|\alpha_j| \sum_{i \in \overline{S} \setminus j} |\beta_i|.
$$

If $\sum_{i\in\overline{S}\setminus j}|\beta_i| > |\beta_j|$, we also have cost at least 2. Otherwise, we have

$$2-2|\alpha_j|(|\beta_j|-\sum_{i\in\overline{S}\setminus j}|\beta_i|)=\begin{cases}2-2|\beta_j|+2\sum_{i\in\overline{S}\setminus j}|\beta_i| & \text{if } 1/|\beta_j|\leq 1,\\ 2\frac{1}{|\beta_j|}\sum_{i\in\overline{S}\setminus j}|\beta_i| & \text{if } 1/|\beta_j|>1.\end{cases}$$

Let C denote the number of such j with $1/|\beta_j|\leq 1$. If $C=1$, the cost is at least 0. If $C\geq 2$, the cost is at least $2C$. Let C' denote the number of such j with $1/|\beta_j|>1$. If $C'=1$, the cost is at least 0, if $C'\geq 2$, the cost is at least $2(C'(C'-1))$. Overall, putting all the eight cases together, we complete the proof. □

17.6.3 Hard instance for oblivious subspace embedding and more row subset selection

The goal in this section is to prove Theorem 17.6.28. By applying Yao's minmax principle, it suffices to prove Theorem 17.6.25.

17.6.3.1 Definitions

We first give the definition of total variation distance and Kullback-Leibler divergence.

Definition 17.6.1. [LPW09, Ver14] The total variation distance between two probability measures P and Q on the measurable space $(\mathcal{X}, \mathcal{F})$ is defined as,

$$D_{\mathrm{TV}}(P, Q) = \sup_{\mathcal{A} \in \mathcal{F}} |P(\mathcal{A}) - Q(\mathcal{A})|.$$

The Kullback-Leibler(KL) divergence of P and Q is defined as,

$$D_{\mathrm{KL}}(P||Q) = \mathop{\mathbb{E}}_{P}\left(\log \frac{\mathrm{d}P}{\mathrm{d}Q}\right) = \int_{\mathcal{X}} \left(\log \frac{\mathrm{d}P}{\mathrm{d}Q}\right) \mathrm{d}P.$$

Lemma 17.6.9. *[Pin60, Tsy09, CK11] Pinsker's inequality states that, if P and Q are two probability distributions on a measurable $(\mathcal{X}, \mathcal{F})$, then*

$$D_{\mathrm{TV}}(P, Q) \leq \sqrt{\frac{1}{2} D_{\mathrm{KL}}(P||Q)}.$$

Lemma 17.6.10. *For any Gaussian random variable $x \sim N(0, \sigma^2)$, we have, for any $a > 0$*

$$\Pr[|x| < a] \lesssim \frac{a}{\sigma}.$$

Proof.

$$\begin{aligned}
\Pr[|x| < a] &= \operatorname{erf}(\frac{a}{\sigma\sqrt{2}}) \\
&\leq (1 - e^{-\frac{-4a^2}{2\sigma^2\pi}})^{\frac{1}{2}} && \text{by } 1 - \exp(-4x^2/\pi) \geq \operatorname{erf}(x)^2 \\
&\leq (\frac{4a^2}{2\sigma^2\pi})^{\frac{1}{2}} && \text{by } 1 - e^{-x} \leq x \\
&\lesssim \frac{a}{\sigma}.
\end{aligned}$$

□

Lemma 17.6.11. *Let $V \in \mathbb{R}^{n\times m}$ be a matrix with orthonormal columns. Let $\mathcal{H}'$ be a distribution over $V^\top A$, where $A \in \mathbb{R}^{n\times k}$ is a random matrix with each entry i.i.d. Gaussian $N(0,1)$. Denote $\mathcal{H}$ as a distribution over $H \in \mathbb{R}^{m\times k}$, where each entry of H is drawn from i.i.d. Gaussian $N(0,1)$. Then, $\mathcal{H}$ and $\mathcal{H}'$ are the same distribution.*

Proof. It is clear that each entry of $V^\top A$ is a random Gaussian variable from $N(0,1)$, so our goal is to prove the entries of $V^\top A$ are fully independent. Since the the j^{th} column of $V^\top A$ only depends on A_j, the variables from different columns are fully independent. Now, we look at one column of $x = V^\top A_j$. The density function is

$$f(x) = \frac{1}{\sqrt{(2\pi)^m |V^\top V|}} \exp\left(-\frac{1}{2} x^\top V^\top V x\right) = \frac{1}{\sqrt{(2\pi)^m}} \exp\left(-\frac{1}{2} x^\top x\right),$$

which is exactly the density function of $N(0, I_m)$. Thus $x \in N(0, I_m)$. Therefore, all the entries of $V^\top A$ are fully independent. □

Lemma 17.6.12 (Matrix Determinant Lemma)**.** *Suppose $A \in \mathbb{R}^{n\times n}$ is an invertible matrix, and $u, v \in \mathbb{R}^n$. Then,*

$$|A + uv^\top| = (1 + v^\top A^{-1} u)|A|.$$

Lemma 17.6.13 (KL divergence between two multivariate Gaussians [PP+08][1])**.** *Given two d-dimensional multivariate Gaussian distribution $N(\mu_1, \Sigma_1)$ and $N(\mu_2, \Sigma_2)$, then*

$$D_{\mathrm{KL}}(N(\mu_1||\Sigma_1), N(\mu_2, \Sigma_2)) = \frac{1}{2}\left(\log\frac{|\Sigma_2|}{|\Sigma_1|} - d + \mathrm{tr}(\Sigma_2^{-1}\Sigma_1) + (\mu_2 - \mu_1)^\top \Sigma_2^{-1}(\mu_2 - \mu_1)\right).$$

Lemma 17.6.14 (Sherman-Morrison formula)**.** *Suppose $A \in \mathbb{R}^{n\times n}$ is an invertible matrix and $u, v \in \mathbb{R}^n$. Suppose furthermore that $1 + v^\top A^{-1}u \neq 0$. Then the Sherman-Morrison formula states that*

$$(A + uv^\top)^{-1} = A^{-1} - \frac{A^{-1}uv^\top A^{-1}}{1 + v^\top A^{-1}u}. \tag{17.23}$$

17.6.3.2 Main results

Lemma 17.6.15. *Let $V \in \mathbb{R}^{n\times m}$ be a matrix with orthonormal columns, and let $A \in \mathbb{R}^{n\times k}$ be a random matrix with each entry drawn from i.i.d. Gaussian $N(0, 1)$. We denote the distribution $\mathcal{D}_i$ over $D_i \in \mathbb{R}^{(m+1)\times k}$ where*

$$D_i = \begin{bmatrix} V^\top A \\ A^i \end{bmatrix}.$$

If $\|(V^\top)_i\|_2^2 < \frac{1}{2}$, then

$$D_{\mathrm{TV}}(\mathcal{D}_i, \mathcal{G}) \leq O(k\|V^i\|_2) + 2^{-\Theta(k)},$$

where $\mathcal{G}$ is a distribution over $G \in \mathbb{R}^{(m+1)\times k}$, where each entry of G is drawn from the i.i.d. Gaussian $N(0, 1)$.

[1] http://stats.stackexchange.com/questions/60680/kl-divergence-between-two-multivariate-gaussians

Proof. Let $\mathcal{H}$ be a distribution over $H \in \mathbb{R}^{m\times k}$, where each entry of H is an i.i.d. Gaussian $N(0,1)$. Let $\mathcal{H}'$ be a distribution over $V^\top A$. According to Lemma 17.6.11, $\mathcal{H}$ and $\mathcal{H}'$ are the same distribution.

We define matrix $V^{\neg i} \in \mathbb{R}^{n\times m}$ as

$$V^{\neg i} = \begin{bmatrix} (V^1)^\top & (V^2)^\top & \cdots & (V^{i-1})^\top & 0 & (V^{i+1})^\top & \cdots & (V^n)^\top \end{bmatrix}^\top,$$

where V^i denotes the i^{th} row of $V \in \mathbb{R}^{n\times m}, \forall i \in [n]$.

Let $\mathcal{G}$ be a distribution over $G \in \mathbb{R}^{(m+1)\times k}$, where each entry of G is an i.i.d. Gaussian $N(0,1)$. Let $\mathcal{P}_i$ be a distribution over $P_i \in \mathbb{R}^{(m+1)\times k}$, where

$$P_i = \begin{bmatrix} (V^{\neg i})^\top A \\ A^i \end{bmatrix}.$$

Let $\widehat{\mathcal{P}}_i$ be a distribution over $\widehat{P}_i \in \mathbb{R}^{m\times k}$, where $\widehat{P}_i = (V^{\neg i})^\top A$. Then we have:

$$D_{\mathrm{TV}}(\mathcal{P}_i, \mathcal{G}) = D_{\mathrm{TV}}(\widehat{\mathcal{P}}_i, \mathcal{H}') = D_{\mathrm{TV}}(\widehat{\mathcal{P}}_i, \mathcal{H}). \tag{17.24}$$

The first equality is because $(V^{\neg i})^\top A$ is independent from A^i. The second equality follows the Lemma 17.6.11.

Claim 17.6.16. *If* $\|V^i\|_2^2 \leq \frac{1}{2}$,

$$D_{\mathrm{KL}}(\widehat{\mathcal{P}}_i || \mathcal{H}) = -\frac{k}{2}\left(\log(1 - \|V^i\|_2^2) + \|V^i\|_2^2\right) \leq k\|V^i\|_2^2.$$

Proof. Let $\widehat{P}_i \sim \widehat{\mathcal{P}}_i, H \sim \mathcal{H}$. Notice that different columns of $\widehat{P}_i$ are i.i.d, and all entries of H are fully independent. We can look at one column of $\widehat{P}_i$ and H. Since $\widehat{P}_i = (V^{\neg i})^\top A$, it is easy to see that its column is drawn from $N(0, \Sigma_1)$ where $\Sigma_1 = I_m - (V^i)^\top V^i$. Since H is fully independent, each column of H is drawn from $N(0, \Sigma_2)$ where $\Sigma_2 = I_m$. Let $p(x)$

be the pdf of the column of $\widehat{P}_i$, and let $q(x)$ be the pdf of the column of H. We have the following calculation [PP+08][2],

$$\begin{aligned}
D_{\mathrm{KL}}(\widehat{\mathcal{P}}_i||\mathcal{H}) &= \frac{k}{2}\left(\log\frac{|\Sigma_2|}{|\Sigma_1|} - m + \mathrm{tr}(\Sigma_2^{-1}\Sigma_1)\right) \\
&= \frac{k}{2}\left(\log\frac{|I_m|}{|I_m - (V^i)^\top V^i|} - m + \mathrm{tr}(I_m - (V^i)^\top V^i)\right) \\
&= \frac{k}{2}\left(-\log|I_m - (V^i)^\top V^i| - m + m - \|V^i\|_2^2\right) \\
&= \frac{k}{2}\left(-\log(1 - \|V^i\|_2^2) - m + m - \|V^i\|_2^2\right) \\
&= -\frac{k}{2}\left(\log(1 - \|V^i\|_2^2) + \|V^i\|_2^2\right) \\
&\leq -\frac{k}{2}\cdot 2\|V^i\|_2^2 \\
&= k\|V^i\|_2^2.
\end{aligned}$$

The first equality is due to Lemma 17.6.13. The sixth equality follows by $\Sigma_2 = I_m$ and $\Sigma_1 = I_m - (V^i)^\top V^i$. The eighth equality follows by Lemma 17.6.12. The first inequality follows by $\log(1-x) + x \geq -2x$, when $0 < x < 1/2$.

□

According to Lemma 17.6.9, we have

$$D_{\mathrm{TV}}(\widehat{\mathcal{P}}_i, \mathcal{H}) \leq \sqrt{\frac{1}{2}D_{\mathrm{KL}}(\widehat{\mathcal{P}}_i||\mathcal{H})} \leq \sqrt{k}\|V^i\|_2. \tag{17.25}$$

Now, we want to argue that $D_{\mathrm{TV}}(\mathcal{D}_i, \mathcal{P}_i)$ is small, where $\mathcal{D}_i$ is a distribution over $D_i \in \mathbb{R}^{(m+1)\times k}$ that

$$D_i = \begin{bmatrix} V^\top A \\ A^i \end{bmatrix}.$$

[2] http://stats.stackexchange.com/questions/60680/kl-divergence-between-two-multivariate-gaussians

For a fixed $x \in \mathbb{R}^k$, let $\widehat{\mathcal{D}}_i(x)$ be a distribution over $\widehat{D}_i(x) \in \mathbb{R}^{m \times k}$, where $\widehat{D}_i(x) = (V^{\neg i})^\top A + (V^i)^\top x$. Let $p(x)$ be the pdf of $(A^i)^\top$, then

$$D_{\mathrm{TV}}(\mathcal{D}_i, \mathcal{P}_i) = \int D_{\mathrm{TV}}\left(\widehat{\mathcal{D}}_i(x), \widehat{\mathcal{P}}_i\right) p(x) \mathrm{d}x. \tag{17.26}$$

Now we look at the j^{th} column of $\widehat{D}_i(x)$ and the j^{th} column of $\widehat{P}_i$. The distribution of the previous one is over $N((V^i)^\top x_j, \Sigma_2)$, and the latter distribution as we said before is $N(0, \Sigma_2)$, where $\Sigma_2 = I_m - (V^i)^\top V^i$. Now we can argue that the KL divergence between them is bounded:

$$\begin{aligned}
& D_{\mathrm{KL}}(N((V^i)^\top x_j, \Sigma_2) || N(0, \Sigma_2)) \\
=& \frac{1}{2}\left(\log \frac{|\Sigma_2|}{|\Sigma_2|} - m + \operatorname{tr}(\Sigma_2^{-1}\Sigma_2) + x_j^2 V^i \Sigma_2^{-1} (V^i)^\top\right) \\
=& \frac{1}{2}\left(-m + \operatorname{tr}(I_m) + x_j^2 V^i \Sigma_2^{-1} (V^i)^\top\right) \\
=& \frac{1}{2} x_j^2 V^i \Sigma_2^{-1} (V^i)^\top \\
=& \frac{1}{2} x_j^2 V^i (I_m - (V^i)^\top V^i)^{-1} (V^i)^\top \\
=& \frac{1}{2} x_j^2 V^i \left(I_m + \frac{(V^i)^\top V^i}{1 - V^i (V^i)^\top}\right) (V^i)^\top \\
=& \frac{1}{2} x_j^2 \left(\|V^i\|_2^2 + \frac{\|V^i\|_2^2}{1 - \|V^i\|_2^2}\right) \\
=& \frac{1}{2} x_j^2 \frac{\|V^i\|_2^2}{1 - \|V^i\|_2^2}.
\end{aligned}$$

The first equality is due to Lemma 17.6.13. The fourth equality follows by $\Sigma_2 = I_m - (V^i)^\top V^i$. The fifth equality follows by Lemma 17.6.14.

By summing the KL divergence on all the columns up,

$$D_{\mathrm{KL}}(\widehat{\mathcal{D}}_i(x) || \widehat{\mathcal{P}}_i) = \frac{1}{2}\|x\|_2^2 \frac{\|V^i\|_2^2}{1 - \|V^i\|_2^2}.$$

Applying Lemma 17.6.9 again, we get

$$D_{\mathrm{TV}}(\widehat{\mathcal{D}}_i(x), \widehat{\mathcal{P}}_i) \leq \sqrt{\frac{1}{2} D_{\mathrm{KL}}(\widehat{\mathcal{D}}_i(x) || \widehat{\mathcal{P}}_i)} = \frac{\|x\|_2 \|V^i\|_2}{2\sqrt{1 - \|V^i\|_2^2}}.$$

Plugging it into Equation (17.26), we get

$$\begin{aligned}
D_{\mathrm{TV}}(\mathcal{D}_i, \mathcal{P}_i) &= \int D_{\mathrm{TV}}\left(\widehat{\mathcal{D}}_i(x), \widehat{\mathcal{P}}_i\right) p(x) \mathrm{d}x \\
&= \int_{\|x\|_2 \leq 10k} D_{\mathrm{TV}}\left(\widehat{\mathcal{D}}_i(x), \widehat{\mathcal{P}}_i\right) p(x) \mathrm{d}x + \int_{\|x\|_2 > 10k} D_{\mathrm{TV}}\left(\widehat{\mathcal{D}}_i(x), \widehat{\mathcal{P}}_i\right) p(x) \mathrm{d}x \\
&\leq \int_{\|x\|_2 \leq 10k} D_{\mathrm{TV}}\left(\widehat{\mathcal{D}}_i(x), \widehat{\mathcal{P}}_i\right) p(x) \mathrm{d}x + \int_{\|x\|_2 > 10k} p(x) \mathrm{d}x \\
&\leq \int_{\|x\|_2 \leq 10k} \frac{\|x\|_2 \|V^i\|_2}{2\sqrt{1 - \|V^i\|_2^2}} p(x) \mathrm{d}x + \int_{\|x\|_2 > 10k} p(x) \mathrm{d}x \\
&\leq \frac{10k\|V^i\|_2}{2\sqrt{1 - \|V^i\|_2^2}} + \int_{\|x\|_2 > 10k} p(x) \mathrm{d}x \\
&\leq \frac{10k\|V^i\|_2}{2\sqrt{1 - \|V^i\|_2^2}} + 2^{-\Theta(k)}.
\end{aligned}$$

The first inequality just follows from the fact that total variation distance is smaller than 1. The second inequality is what we plugged in. The last inequality follows from the fact that $x \sim N(0, I_k)$ and from the tail bounds of a Gaussian.

Together with Equation (17.24) and Equation (17.25), we can get

$$\begin{aligned}
D_{\mathrm{TV}}(\mathcal{D}_i, \mathcal{G}) &\leq D_{\mathrm{TV}}(\mathcal{D}_i, \mathcal{P}_i) + D_{\mathrm{TV}}(\mathcal{G}, \mathcal{P}_i) \\
&= D_{\mathrm{TV}}(\mathcal{D}_i, \mathcal{P}_i) + D_{\mathrm{TV}}(\widehat{\mathcal{P}}_i, \mathcal{H}) \\
&\leq \frac{10k\|V^i\|_2}{2\sqrt{1 - \|V^i\|_2^2}} + 2^{-\Theta(k)} + \sqrt{k}\|V^i\|_2 \\
&\leq 10k\|V^i\|_2 + 2^{-\Theta(k)} + \sqrt{k}\|V^i\|_2.
\end{aligned}$$

The last inequality follows by $\|V^i\|_2^2 \leq \frac{1}{2}$. Then, we have completed the proof.

□

Lemma 17.6.17. *$A \in \mathbb{R}^{r \times k}$ $(r \geq k)$ is a random matrix for which each entry is i.i.d. $N(0,1)$. With probability at least $1 - e^{-\Theta(r)}$, the maximum singular value $\|A\|_2$ is at most $O(\sqrt{r})$.*

Proof. Since $A \in \mathbb{R}^{k \times r}$ is a random matrix with each entry i.i.d. $N(0,1)$, this follows by standard arguments (Proposition 2.4 in [RV10]). Since $r \geq k$, with probability at least $1 - e^{-\Theta(r)}$, $\|A\|_2$ is at most $O(\sqrt{r})$. □

Definition 17.6.2. Let $V \in \mathbb{R}^{n \times r}$ be a matrix with orthonormal columns, and let each entry of $A \in \mathbb{R}^{k \times r}$ be a random variable drawn from an i.i.d. Gaussian $N(0,1)$. Define event $\widehat{\mathcal{E}}(A, V, \beta, \gamma)$ to be: $\forall y \in \mathbb{R}^n$, $\|y\|_1 \leq O(k^\gamma)$ and each coordinate of y has absolute value at most $1/k^\beta$, and also $AV^\top y$ has at most $O(k/\log k)$ coordinates with absolute value at least $\Omega(1/\log k)$, and $\|A\|_2 \leq O(\sqrt{r})$.

Lemma 17.6.18. *For any $k \geq 1$, and any constants $c_2 \geq c_1 \geq 1$, let $k \leq r = O(k^{c_1}), r \leq n = O(k^{c_2})$, let $V \in \mathbb{R}^{n \times r}$ be a matrix with orthonormal columns, and let each entry of $A \in \mathbb{R}^{k \times r}$ be a random variable drawn from i.i.d. Gaussian $N(0,1)$. Furthermore, if β and γ are two constants which satisfy $\beta > \gamma > 0$ and $\beta + \gamma < 1$,*

$$\Pr\left[\widehat{\mathcal{E}}(A, V, \beta, \gamma)\right] \geq 1 - 2^{-\Theta(k)}.$$

Proof. Due to Lemma 17.6.17, with probability at least $1 - 2^{-\Theta(r)}$, $\|A\|_2 \leq O(\sqrt{r})$, so we can restrict attention to $\|A\|_2 \leq O(\sqrt{r})$ in the following proof.

Take any $y \in \mathbb{R}^n$ which has each non-zero coordinate with absolute value at most $1/k^\beta$, and write it as $y = \sum_{j=j_0}^{+\infty} y^j$, where the coordinates in y^j have absolute value in the

range $[2^{-j-1}, 2^{-j})$, and the supports of different y^j are disjoint. Since each coordinate of y is at most $1/k^\beta$, $2^{-j_0} = 1/k^\beta$. Since $\|y\|_1 \leq O(k^\gamma)$, the support size of $y^j \in \mathbb{R}^n$ is at most $s_j \leq O(2^{j+1}k^\gamma)$, so it follows that

$$\|y^j\|_2 \leq \sqrt{s_j \cdot 2^{-2j}} \leq \sqrt{2^{-j+1}k^\gamma}.$$

We also know $\|y^j\|_2 \geq \sqrt{s_j \cdot 2^{-2j-2}}$. Then we can conclude that y^j has 2-norm $\Theta(2^{-j}\sqrt{s_j})$. Now we state an ϵ-net for all the possible y^j: let $\epsilon = O(1/(nrk^3)) = O(1/(k^{c_1+c_2+3}))$. Let $\mathcal{N}_j \subset \mathbb{R}^n$ be the following:

$$\mathcal{N}_j = \left\{p \in \mathbb{R}^n \mid \exists q \in \mathbb{Z}^n, \text{ s.t. } p = \epsilon q, \|p\|_1 \leq O(k^\gamma), \forall i \in [n], \text{ either } |p_i| \in [2^{-j-1}, 2^{-j}) \text{ or } p_i = 0\right\}.$$

Obviously, for any y^j, there exists $p \in \mathcal{N}_j$ such that $\|y^j - p\|_\infty \leq \epsilon = O(1/(k^{c_1+c_2+3}))$, since $n \leq O(k^{c_2})$, $\|y^j - p\|_2 \leq \|y^j - p\|_1 \leq n\|y^j - p\|_\infty \leq O(1/k^{c_1+3})$. Now let us consider the size of $\mathcal{N}_j$. If $p \in \mathcal{N}_j$, the choice of one coordinate of p is at most $2/\epsilon+1$. And since $\|p\|_1 \leq O(k^\gamma)$ and each coordinate of p has absolute value at least 2^{-j-1}, the number of supports of p is at most $O(2^{j+1}k^\gamma)$. Therefore,

$$\begin{aligned}
|\mathcal{N}_j| &\leq (n+1)^{O(2^{j+1}k^\gamma)} \cdot (2/\epsilon+1)^{O(2^{j+1}k^\gamma)} \\
&\leq 2^{O(2^{j+1}k^\gamma(\log n + \log(2/\epsilon+1))} \\
&\leq 2^{O(2^{j+1}k^\gamma \log k)}.
\end{aligned}$$

The last inequality follows from $n \leq O(k^{c_2}), r \leq O(k^{c_1}), \epsilon = O(1/(rnk^3))$.

We define an event $\mathcal{E}(y^j)$ to be: $AV^\top y^j$ has $k/\log^2 k$ coordinates which are each at least $2/\log^2 k$ in absolute value. Now, we want to show,

Claim 17.6.19. *For any $j \geq j_0$, for a fixed $y^j \in \mathbb{R}^n$,*

$$\Pr\left[\mathcal{E}(y^j) \text{ happens }\right] \leq 2^k e^{-\Theta(k/(\|y^j\|_2^2 \log^6 k))} \leq e^{-\Theta(2^{j-1}k^{1-\gamma}/\log^6 k)}.$$

Proof. We let p be the probability that the absolute value of a single coordinate of $AV^\top y^j$ is at least $1/\log^2 k$. Notice that each coordinate of $AV^\top y^j$ is i.i.d. Gaussian $N(0, \|V^\top y^j\|_2^2)$ and because for any Gaussian random variable g, $\Pr[|g| \geq t] \leq \exp(-\Theta(t^2/\sigma^2))$, then $p \leq \exp(-1/(\|V^\top y^j\|_2^2 \log^4 k))$, by plugging $\sigma^2 = \|V^\top y^j\|_2^2$ and $t = 1/\log^2 k$. So the probability $AV^\top y^j$ has $k/\log^2 k$ coordinates which are each at least $1/\log^2 k$ in absolute value is,

$$\begin{aligned}
\sum_{i=k/\log^2 k}^{k} p^i (1-p)^{k-i} \binom{k}{i} \leq & \sum_{i=k/\log^2 k}^{k} p^i \binom{k}{k/2} \leq 2p^{k/\log^2 k} 2^k \\
\leq & \; e^{-\Theta(kt^2/(\sigma^2 \log^2 k))} 2^k \\
\leq & \; e^{-\Theta(k/(\|V^\top y^j\|_2^2 \log^6 k))} 2^k \\
\leq & \; e^{-\Theta(k/(\|y^j\|_2^2 \log^6 k))} 2^k \\
\leq & \; e^{-\Theta(k/(2^{-2j} s_j \log^6 k))} 2^k \\
\leq & \; e^{-\Theta(k/(2^{-j+1} k^\gamma \log^6 k))} 2^k \\
= & \; e^{-\Theta(k/(2^{-j+1} k^\gamma \log^6 k))} \\
\leq & \; e^{-\Theta(2^{j-1} k^{1-\gamma}/\log^6 k)},
\end{aligned}$$

where the second step follows from $p \leq \exp(-\Theta(t^2/\sigma^2))$, the forth step follows from $\|V^\top y^j\|_2^2 \leq \|V\|_2^2 \|y^j\|_2^2 \leq \|y^j\|_2^2$, the fifth step follows from $\|y^j\|_2^2 \leq 2^{-2j} s_j$, the sixth step follows from $s_j \leq O(2^{j+1} k^\gamma)$, the seventh step follows from $2^{-j+1} k^\gamma \log^6 k \leq 2^{-j_0+1} k^\gamma \log^6 k \leq 2k^{\gamma-\beta} \log^6 k = o(1)$. □

For $j_1 = \lceil 100(c_1+c_2+1)\log k \rceil = \Theta(\log k)$, consider $j \in [j_1, \infty)$. We have $\| \sum_{j=j_1}^{\infty} y^j \|_2$ is at most $\Theta(2^{-j_1}\sqrt{n}) \leq 1/k^{100(1+c_1)}$, and so

$$\begin{aligned}
\|AV^\top \sum_{j=j_1}^{\infty} y^j\|_1 &\leq \sqrt{k}\|AV^\top \sum_{j=j_1}^{\infty} y^j\|_2 \\
&\leq \sqrt{k}\|A\|_2\|V\|_2\| \sum_{j=j_1}^{\infty} y^j\|_2 \\
&\leq \sqrt{k} \cdot \sqrt{r} \cdot 1/k^{100(1+c_1)} \leq 1/k^{50}.
\end{aligned}$$

The last inequality follows from $r = O(k^{c_1})$. So the contribution of y^j to $\|AV^\top y^j\|_1$ for all $j \geq j_1$ is at most $1/k^{50}$. Thus, if we only consider those j which contribute, i.e., $j_0 \leq j \leq j_1$, we have $O(\log k)$ values of j. Then we can only construct $O(\log k)$ nets $\mathcal{N}_{j_0}, \mathcal{N}_{j_0+1}, \cdots, \mathcal{N}_{j_1}$. Since the size of net $\mathcal{N}_j$ is $2^{\Theta(2^{j+1}k^\gamma \log k)}$, by combining Claim 17.6.19 and taking a union bound, we have

$$\begin{aligned}
\Pr\left[\exists y^j \in \bigcup_{j=j_0}^{j_1} \mathcal{N}_j,\ \mathcal{E}(y^j) \text{ happens}\right] &\leq \sum_{j=j_0}^{j_1} 2^{\Theta(2^{j+1}k^\gamma \log k)} \cdot e^{-\Theta(2^{j-1}k^{1-\gamma}/\log^6 k)} \\
&\leq e^{-\Theta(2^{j_0-1}k^{1-\gamma}/\log^6 k)} \\
&\leq e^{-\Theta(k^{1+\beta-\gamma}/\log^6 k)} \\
&\leq 2^{-\Theta(k)}.
\end{aligned}$$

The second inequality follows since $k^\gamma = o(k^{1-\gamma})$. The third inequality follows since $2^{j_0} \geq k^\beta$. The fourth inequality follows since $1 + \beta - \gamma > 1$.

Then, $\forall j_0 \leq j \leq j_1, \forall \widetilde{y}^j \notin \mathcal{N}_j$, there exists a vector $\widehat{y}^j$ in $\mathcal{N}_j$, such that $\|\widehat{y}^j - \widetilde{y}^j\|_2 \leq$

$1/k^{3+c_1}$. We can upper bound the ℓ_∞ norm of $AV^\top \widehat{y}^j - AV^\top \widetilde{y}^j$ in the following sense,

$$\begin{aligned}
\|AV^\top \widehat{y}^j - AV^\top \widetilde{y}^j\|_\infty &\leq \|AV^\top \widehat{y}^j - AV^\top \widetilde{y}^j\|_2 && \text{by } \|\cdot\|_\infty \leq \|\cdot\|_2 \\
&\leq \|A\|_2 \cdot \|\widehat{y}^j - \widetilde{y}^j\|_2 \\
&\leq \sqrt{r}/k^{3+c_1} && \text{by Claim 17.6.17 and } \|\widehat{y}^j - \widetilde{y}^j\|_2 \leq 1/k^{3+c_1} \\
&= 1/k^2. && \text{by } r \leq O(k^{c_1})
\end{aligned}$$

We let $Y = \{y \in \mathbb{R}^n \mid \|y\|_1 \leq O(k^\gamma) \text{ and each coordinate of } y \leq 1/k^\beta\}$. Since $1/k^2 < 1/\log^2 k$ we can conclude that,

$$\Pr\left[\exists y \in Y, j \geq j_0, \mathcal{E}(y^j) \text{ happens }\right] \leq 2^{-\Theta(k)}. \tag{17.27}$$

Recalling $y = \sum_j y^j$, by Equation (17.27), for any y, with probability at most $2^{-\Theta(k)}$, there are at most $O(\log k) \cdot k/\log^2 k \leq O(k/\log k)$ coordinates for which $AV^\top \sum_{j=j_0}^{j_1} y^j$ (the same statement also holds for $AV^\top \sum_{j=j_0}^{\infty} y^j = AV^\top y$ since we argued that there is negligible "contribution" for those $j > j_1$) is at least $O(\log k)/\log^2 k = O(1/\log k)$ on that coordinate. Summarizing,

$$\Pr\left[\widehat{\mathcal{E}}(A, V, \beta, \gamma)\right] \geq 1 - 2^{-\Theta(k)}.$$

□

Lemma 17.6.20. *For any $t, k \geq 1$, and any constants $c_2 \geq c_1 \geq 1$, let $k \leq r = O(k^{c_1}), r \leq n = O(k^{c_2})$, let $V \in \mathbb{R}^{n\times r}$ be a matrix with orthonormal columns, and let each entry of $A \in \mathbb{R}^{k\times r}, v_1, v_2, \cdots, v_t \in \mathbb{R}^k$ be an i.i.d. Gaussian $N(0,1)$ random variable. For a constant $\alpha \in (0, 0.5)$ which can be arbitrarily small, if $\widehat{\mathcal{E}}(A, V, 0.5 + \alpha/2, 0.5 - \alpha)$ happens, then with probability at least $1 - 2^{-\Theta(tk)}$, there are at least $\lceil t/10 \rceil$ such $j \in [t]$ that $\forall x \in \mathbb{R}^r$ either $\|Ax - v_j\|_1 \geq \Omega(k^{0.5-\alpha})$ or $\|Vx\|_1 \geq \Omega(k^{0.5-\alpha})$ holds.*

Proof. For convenience, we define $\gamma = 0.5 - \alpha$ which can be an arbitrary constant in $(0, 0.5)$. We let constant $\beta = 0.5 + \alpha/2$. Then we have $\beta + \gamma < 1$ and $\beta > \gamma$. Let $v \in \mathbb{R}^k$ be a random vector with each entry drawn from i.i.d. Gaussian $N(0, 1)$. Suppose $\widehat{\mathcal{E}}(A, V, \beta, \gamma)$ happens.

For any $x \in \mathbb{R}^r$, we can find a $y \in \mathbb{R}^n$ such that $y = Vx$. Since $V \in \mathbb{R}^{n\times r}$ has orthonormal columns, $x = V^\top Vx = V^\top y$. Then our goal is to argue that with high probability if $\|AV^\top y - v\|_1 \le O(k^\gamma)$, then $\|y\|_1 > \Omega(k^\gamma)$. Take any $y \in \mathbb{R}^n$ which can be expressed as Vx, and decompose it as $y = y_0 + y_1$, where $y_0 \in \mathbb{R}^n$ and $y_1 \in \mathbb{R}^n$ have disjoint supports, y_0 has each coordinate with absolute value greater than $1/k^\beta$, and y_1 has each coordinate with absolute value at most $1/k^\beta$.

Now we create an ϵ-net for y_0: let $\epsilon = O(1/(rnk^3)) = O(1/k^{c_1+c_2+3})$, we denote $\mathcal{N} \subset \mathbb{R}^n$ as follows:

$$\mathcal{N} = \left\{ p \in \mathbb{R}^n \mid \exists q \in \mathbb{Z}^n, \text{ s.t. } p = \epsilon q, \|p\|_1 \le O(k^\gamma), \forall i \in [n], \text{ either } |p_i| > 1/k^\beta \text{ or } p_i = 0 \right\}.$$

Obviously, for any y_0, there exists $p \in \mathcal{N}$ such that $\|y_0 - p\|_\infty \le \epsilon = O(1/(k^{c_1+c_2+3}))$, since $n \le O(k^{c_2})$, $\|y_0 - p\|_2 \le \|y_0 - p\|_1 \le n\|y_0 - p\|_\infty \le O(1/k^{c_1+3})$. Now let us consider the size of $\mathcal{N}$. If $p \in \mathcal{N}$, the number of choices of one coordinate of p is at most $O(k^\gamma/\epsilon)$. And since $\|p\|_1 \le O(k^\gamma)$ and each coordinate of p has absolute value at least $1/k^\beta$, the number of supports of p is at most $O(k^{\gamma+\beta})$. Therefore,

$$\begin{aligned} |\mathcal{N}| &\le (n+1)^{O(k^{\gamma+\beta})} \cdot O(k^\gamma/\epsilon)^{O(k^{\gamma+\beta})} \\ &\le 2^{O(k^{\gamma+\beta}\log k)}. \end{aligned}$$

The last inequality follows from $n \le O(k^{c_2}), r \le O(k^{c_1}), \epsilon = O(1/(rnk^3))$.

For $y_0 \in \mathbb{R}^n$, we define event $\mathcal{E}_1(y_0)$ as: $\exists$ valid $y_1 \in \mathbb{R}^n$, $\|AV^\top y - v\|_1 \le O(k^\gamma)$, where $y = y_0 + y_1$ is the decomposition of a possible $y \in \mathbb{R}^n$. Here y_1 is valid means that there

exists $y \in \mathbb{R}^n$ such that $\|y\|_1 \leq O(k^\gamma)$ and the decomposition of y is $y = y_0 + y_1$. We define event $\mathcal{E}_2(y_0)$ as: $\exists$ valid y_1, the absolute value of $AV^\top y - v$ is at most $O(1/k^\gamma)$ for at least $k - O(k^{2\gamma})$ coordinates i in $[k]$. We define event $\mathcal{E}_3(y_0)$ as: at least $k - O(k^{2\gamma}) - O(k/\log k)$ coordinates of $Ay_0 - v$ have absolute value at most $O(1/\log k)$.

Claim 17.6.21. *For $y_0 \in \mathbb{R}^n$,*

$$\Pr[\mathcal{E}_3(y_0) \text{ happens }] \geq \Pr[\mathcal{E}_2(y_0) \text{ happens }] \geq \Pr[\mathcal{E}_1(y_0) \text{ happens }].$$

Proof. If $\mathcal{E}_1(y_0)$ happens, then there exists a valid $y_1 \in \mathbb{R}^n$ such that $y = y_0 + y_1$ and $\|AV^\top y - v\|_1 \leq O(k^\gamma)$. For this y, there are at least $k - O(1/k^{2\gamma})$ coordinates of $AV^\top y - v$ with absolute value at most $O(1/k^\gamma)$. Otherwise, $\|AV^\top y - v\|_1 > \Omega(k^\gamma)$. Thus, $\mathcal{E}_1(y_0)$ implies $\mathcal{E}_2(y_0)$.

Now we want to show $\mathcal{E}_2(y_0)$ implies $\mathcal{E}_3(y_0)$. We suppose $\mathcal{E}_2(y_0)$ happens. Then there is a valid y_1 such that there are at least $k - O(1/k^{2\gamma})$ coordinates of $AV^\top y - v$ with absolute value at most $O(1/k^\gamma)$, where $y = y_0 + y_1$. Recall that the event $\widehat{\mathcal{E}}(A, V, \beta, \gamma)$ happens, for any valid y_1 there are at most $O(k/\log k)$ coordinates of $AV^\top y_1$ are at least $\Omega(1/\log k)$. Therefore,

$$AV^\top y_0 - v = \underbrace{AV^\top y - v}_{\substack{\geq k - O(k^{2\gamma}) \text{ coordinates} \\ \text{each } \leq O(1/k^\gamma)}} - \underbrace{AV^\top y_1}_{\substack{\leq O(k/\log k) \text{ coordinates} \\ \text{each} \geq \Omega(1/\log k)}}.$$

Therefore, at least $k - O(k^{2\gamma}) - O(k/\log k)$ coordinates of $Ay_0 - v$ in absolute value is at most $O(1/\log k) + O(1/k^\gamma) = O(1/\log k)$ □

Claim 17.6.22. *Define event $\mathcal{F}_1$ to be the situation for which there exists $1/2$ of the coordinates of $v \in \mathbb{R}^k$ that are at least $1/100$. The probability this event $\mathcal{F}_1$ holds is at least $1 - 2^{-\Theta(k)}$.*

Proof. Note that $\overline{\mathcal{F}}_1$ means there exist $k/2$ of the coordinates of $v \in \mathbb{R}^k$ which are at most $1/100$. Using Lemma 17.6.10, for each single coordinate the probability it is smaller than $1/100$ is $1/200$. Then the probability more than half the coordinates are no more than $1/100$ is

$$\sum_{i=k/2}^{k} p^i (1-p)^{k-i} \binom{k}{i} \leq \sum_{i=k/2}^{k} p^i \binom{k}{k/2} \leq 2p^{k/2} 2^k \leq (1/200)^{\Theta(k)} = 2^{-\Theta(k)}.$$

□

Conditioned on $\mathcal{F}_1$ happening, if $\mathcal{E}_3(y_0)$ happens, then due to the pigeonhole principle, there are at least $\frac{k}{2} - O(k^{2\gamma}) - O(k/\log k)$ coordinates of $AV^\top y_0 - v$ that are at most $O(1/\log k)$ and the corresponding coordinate of v is larger than $1/100$. Now, let us look at the probability on a single coordinate of $AV^\top y_0 - v$.

Claim 17.6.23. *If the i^{th} coordinate of $v \in \mathbb{R}^k$ is at least $1/100$. Then $\Pr[|(AV^\top y_0)_i - (v)_i| \leq O(1/\log k)] \leq O(1/\log k)$.*

Proof. Since $|(v)_i| > 1/100$ and v is independent from $A \in \mathbb{R}^{k \times r}$, for $0 < \eta < 1/100$, $\Pr[|(AV^\top y_0)_i - (v)_i| \leq \eta]$ is always upper bounded by $\Pr[(Ax_0)_i \in [1/100 - \eta, 1/100 + \eta]$. Thus, it suffices to prove an upper bound for $\Pr[(AV^\top x_0)_i \in [1/100 - \eta, 1/100 + \eta]$. Let $f(x)$

be the pdf of $N(0, \sigma^2)$, where $\sigma^2 = \|V^\top y_0\|_2^2, V \in \mathbb{R}^{n \times r}, y_0 \in \mathbb{R}^n$. Then

$$\begin{aligned} & \Pr[(AV^\top y_0)_i \in [1/100 - \eta, 1/100 + \eta]] \\ = & \int_{1/100-\eta}^{1/100+\eta} f(x) \mathrm{d}x \\ \leq & f(1/200) \int_{1/100-\eta}^{1/100+\eta} \mathrm{d}x \\ \leq & O(\eta). \end{aligned}$$

where the last step follows since $f(1/200) \leq 200$. We set $\eta = O(1/\log k)$, then we get the statement. □

Claim 17.6.24. *Conditioned on $\mathcal{F}_1$, for a fixed $y_0 \in \mathbb{R}^n$, with probability at most $2^{-\Theta(k)}$, there are at least $\frac{k}{10}$ coordinates of $AV^\top y_0 - v \in \mathbb{R}^k$ which are at most $O(1/\log k)$ and the corresponding coordinate of v is larger than $1/100$.*

Proof. We look at the coordinate $i \in [k]$ which has $|(v)_i| > 1/100$. The probability $\|(AV^\top y_0)_i - (v)_i\| \leq O(1/\log k)$ is at most $O(1/\log k)$. Due to the independence between different coordinates of v, since there are at least $k/2$ coordinates of v satisfying that they have absolute value greater than $1/100$, with probability at most $2^{-\Theta(k)}$, there are at least $\frac{1}{5} \cdot \frac{k}{2} = k/10$ coordinates of $AV^\top y_0 - v$ which are at most $O(1/\log k)$.

□

Because $\mathcal{E}_3(y_0)$ implies the event described in the above claim when conditioning on $\mathcal{F}_1$, for a fixed y_0, the probability that $\mathcal{E}_3(y_0)$ holds is at most $2^{-\Theta(k)}$. Since $\gamma + \beta < 1$, the $|\mathcal{N}| \leq 2^{k^{o(1)}}$, we can take a union bound over the $\mathcal{N}$:

$$\Pr[\exists y_0 \in \mathcal{N}, \mathcal{E}_1(y_0) \text{ happens }] \leq 2^{-\Theta(k)}.$$

It means that with probability at least $1 - 2^{-\Theta(k)}$, for any $y = y_0 + y_1$ with $y_0 \in \mathcal{N}$, $\|AV^\top y - v\|_1 > \Omega(k^\gamma)$. Let $\widetilde{y} = \widetilde{y}_0 + \sum \widetilde{y}^j$, where $\widetilde{y}_0 \notin \mathcal{N}$. We can find $\widehat{y}_0 \in \mathcal{N}$ which is the closest to $\widetilde{y}_0$. Denote $\widehat{y} = \widehat{y} + \sum \widetilde{y}^j$. Then,

$$\begin{aligned}
\|AV^\top \widehat{y} - v\|_1 &\leq \|AV^\top \widetilde{y} - v\|_1 + \|AV^\top \widetilde{y} - AV^\top \widehat{y}\|_1 && \text{by triangle inequality} \\
&\leq \|AV^\top \widetilde{y} - v\|_1 + \sqrt{k}\|AV^\top \widetilde{y} - AV^\top \widehat{y}\|_2 && \text{by } \|\cdot\|_1 \leq \sqrt{\dim}\|\cdot\|_2 \\
&\leq \|AV^\top \widetilde{y} - v\|_1 + \sqrt{k}\|A\|_2\|\widetilde{y} - \widehat{y}\|_2 \\
&\leq \|AV^\top \widetilde{y} - v\|_1 + \sqrt{k}\cdot\sqrt{r}\cdot\|\widetilde{y}_0 - \widehat{y}_0\|_2 && \text{by } \|A\|_2 \leq \sqrt{r} \\
&\leq \|AV^\top \widetilde{y} - v\|_1 + \sqrt{k}\cdot\sqrt{r}\cdot\epsilon && \text{by } \|\widetilde{y}_0 - \widehat{y}_0\|_2 \leq \epsilon \\
&= \|AV^\top \widetilde{y} - v\|_1 + \sqrt{k}\cdot\sqrt{r}\cdot 1/k^{c_1+3} && \text{by } \epsilon' = 1/k^{c_1+3} \\
&= \|AV^\top \widetilde{y} - v\|_1 + 1/k. && \text{by } r = O(k^{c_1})
\end{aligned}$$

and so if $\|AV^\top \widetilde{y} - v\|_1$ is at most $O(k^\gamma)$ then $\|A\widehat{y} - v\|_1$ is at most $O(k^\gamma)$.

For $j \in [t]$, we now use notation $\mathcal{E}_4(v_j)$ to denote the event: $\exists y_0 \in \mathbb{R}^n$ with $\|y_0\|_1 \leq O(k^\gamma)$ and each non-zero coordinate of y_0 is greater than $1/k^\beta$, at least $k - O(k^{2\gamma}) - O(k/\log k)$ coordinates of $AV^\top y_0 - v_j$ in absolute value are at most $O(1/\log k)$. Based on the previous argument, conditioned on $\widehat{\mathcal{E}}(A, V, \beta, \gamma)$,

$$\Pr[\mathcal{E}_4(v_j)] \leq 2^{-\Theta(k)}.$$

Also notice that, conditioned on $\widehat{\mathcal{E}}(A, V, \beta, \gamma)$, $\forall j \in [t]$, $\mathcal{E}_4(v_j)$ are independent. Thus, conditioned on $\widehat{\mathcal{E}}(A, V, \beta, \gamma)$, due to the Chernoff bound, the probability that there are $\lceil t/10 \rceil$ such j that $\mathcal{E}_4(v_j)$ happens is at most $2^{-\Theta(kt)}$. We define $\mathcal{E}_5(v_j)$ to be the event: $\exists y \in \mathbb{R}^n, \|AV^\top y - v_j\| \leq O(k^\gamma)$ with $\|y\|_1 \leq O(k^\gamma)$. Similar to the proof of Claim 17.6.21, conditioned on $\widehat{\mathcal{E}}(A, V, \beta, \gamma)$, $\mathcal{E}_5(v_j)$ implies $\mathcal{E}_4(v_j)$. Thus, conditioned on $\widehat{\mathcal{E}}(A, V, \beta, \gamma)$, the

probability that there are $\lceil t/10 \rceil$ such j that $\mathcal{E}_5(v_j)$ happens is at most $2^{-\Theta(tk)}$. Then, we complete the proof.

□

Theorem 17.6.25. *For any $k \geq 1$, and any constants c_1, c_2 which satisfy $c_2 - 2 > c_1 > 1$, let $r = \Theta(k^{c_1}), n = \Theta(k^{c_2})$, and let $\mathcal{A}(k, n)$ denote a distribution over $n \times (k + n)$ matrices where each entry of the first $n \times k$ matrix is i.i.d. Gaussian $N(0, 1)$ and the next $n \times n$ matrix is an identity matrix. For any fixed $r \times n$ matrix S and a random matrix $\widehat{A} \sim \mathcal{A}(k, n)$, with probability at least $1 - O(k^{1+\frac{c_1-c_2}{2}}) - 2^{-\Theta(k)}$, there is no algorithm that is able to output a matrix $B \in \mathbb{R}^{n \times r}$ such that*

$$\|BS\widehat{A} - \widehat{A}\|_1 \leq O(k^{0.5-\epsilon}) \min_{\text{rank}-k \ A'} \|A' - \widehat{A}\|_1,$$

where $\epsilon > 0$ is a constant which can be arbitrarily small.

Proof. For convenience, we let $\gamma = 0.5 - \epsilon$ be a constant which can be arbitrarily close to 0.5. Since the last n columns of $\widehat{A}$ is an identity matrix, we can fit the first k columns of $\widehat{A}$, so we have

$$\min_{\text{rank}-k \ A'} \|A' - \widehat{A}\|_1 \leq n.$$

Now, we want to argue that, for a fixed S, with high probability, for any rank-k $n \times r$ matrix B, the cost

$$\|BS\widehat{A} - \widehat{A}\|_1 \geq \Omega(n \cdot k^{\gamma}).$$

Thus, the approximation gap will be at least $\Omega(k^{\gamma})$.

We denote the SVD of $S = U_S \Sigma_S V_S^\top$ where $U_S \in \mathbb{R}^{r\times r}, \Sigma_S \in \mathbb{R}^{r\times r}, V_S \in \mathbb{R}^{n\times r}$. Then, we can rewrite

$$\begin{aligned}\|BS\widehat{A} - \widehat{A}\|_1 &= \|BU_S\Sigma_S V_S^\top \widehat{A} - \widehat{A}\|_1 \\ &= \sum_{l=1}^{n} \|(BU_S\Sigma_S)^l (V_S^\top \widehat{A}) - \widehat{A}^l\|_1 \\ &\geq \sum_{l:\|V_S^l\|_2^2 \leq 2r/n} \|(BU_S\Sigma_S)^l (V_S^\top \widehat{A}) - \widehat{A}^l\|_1. \end{aligned} \tag{17.28}$$

The first equality follows from the SVD of S. The second equality follows from the fact that the ℓ_1-norm of a matrix is the sum of ℓ_1-norms of rows. The third inequality follows since we just look at the cost on a part of the rows.

We use β_l to denote $(BU_S\Sigma_S)^l$. We look at a fixed row l, then the cost on this row is:

$$\begin{aligned}&\|\beta_l (V_S^\top \widehat{A}) - \widehat{A}^l\|_1 \\ =&\|\beta_l (V_S^\top \widehat{A})_{[1:k]} - (\widehat{A}^l)_{[1:k]}\|_1 + \|\beta_l (V_S^\top \widehat{A})_{[k+1:k+n]} - (\widehat{A}^l)_{[k+1:n+k]}\|_1 \\ \geq&\|\beta_l (V_S^\top \widehat{A})_{[1:k]} - (\widehat{A}^l)_{[1:k]}\|_1 + \|\beta_l (V_S^\top \widehat{A})_{[k+1:k+n]}\|_1 - \|(\widehat{A}^l)_{[k+1:n+k]}\|_1 \\ \geq&\|\beta_l (V_S^\top \widehat{A})_{[1:k]} - (\widehat{A}^l)_{[1:k]}\|_1 + \|\beta_l (V_S^\top \widehat{A})_{[k+1:k+n]}\|_1 - 1 \\ \geq&\|\beta_l (V_S^\top \widehat{A})_{[1:k]} - (\widehat{A}^l)_{[1:k]}\|_1 + \|\beta_l V_S^\top\|_1 - 1. \end{aligned} \tag{17.29}$$

where $(V_S^\top \widehat{A})_{[1:k]}$ denotes the first k columns of $(V_S^\top \widehat{A})$, and similarly, $(V_S^\top \widehat{A})_{[k+1:k+n]}$ denotes the last n columns of $(V_S^\top \widehat{A})$. The first equality is because we can compute the sum of ℓ_1 norms on the first k coordinates and ℓ_1 norm on the last n coordinates. The first inequality follows from the triangle inequality. The second inequality follows since the last n columns of $\widehat{A}$ form an identity, so there is exactly one 1 on the last n columns in each row. The third

inequality follows since the last n columns of $\widehat{A}$ form an identity. Let $\mathcal{D}_l$ be a distribution over $D_l \in \mathbb{R}^{(r+1)\times k}$, where

$$D_l = \begin{bmatrix} (V_S^\top \widehat{A})_{[1:k]} \\ (\widehat{A}^l)_{[1:k]} \end{bmatrix}.$$

Let $\mathcal{G}$ be a distribution over $G \in \mathbb{R}^{(r+1)\times k}$ where each entry of G is drawn from i.i.d. $N(0,1)$. According to Lemma 17.6.15, we have

$$D_{\mathrm{TV}}(\mathcal{D}_l, \mathcal{G}) \leq O(k\|(V_S)^l\|_2) + 2^{-\Theta(k)}. \tag{17.30}$$

Let $A = G^{[1:r]}, v = G^{r+1}$. Due to Lemma 17.6.18, with probability at least $1 - 2^{-\Theta(k)}$, $\widehat{\mathcal{E}}(A^\top, V_S, 0.75 - \gamma/2, \gamma)$ happens. Then conditioned on $\widehat{\mathcal{E}}(A^\top, V_S, 0.75 - \gamma/2, \gamma)$, due to Lemma 17.6.20, with probability at most $2^{-\Theta(k)}$, there exists β_l such that

$$\|\beta_l A - v\|_1 + \|\beta_l V_S^\top\|_1 = o(k^\gamma).$$

Combined with Equation (17.30), we can get that for a fixed l, with probability at most $O(k\|(V_S)^l\|_2) + 2^{-\Theta(k)}$, there exists β_l such that

$$\|\beta_l (V_S^\top \widehat{A})_{[1:k]} - (\widehat{A}^l)_{[1:k]}\|_1 + \|\beta_l V_S^\top\|_1 = o(k^\gamma).$$

When $\|(V_S)^l\|_2^2 \leq 2r/n = \Theta(k^{c_1-c_2})$, this probability is at most $\Theta(k^{1+(c_1-c_2)/2}) + 2^{-\Theta(k)}$. Since $\sum_{l=1}^n \|(V_S)^l\|_2^2 = r$, there are at most $n/2$ such l that $\|(V_S)^l\|_2^2 > 2r/n$ which means that there are at least $n/2$ such l that $\|(V_S)^l\|_2^2 \leq 2r/n$. Let s be the number of l such that $\|(V_S)^l\|_2^2 \leq 2r/n$, then $s > n/2$. Let t be a random variable which denotes that the number of l which satisfies $\|(V_S)^l\|_2^2 \leq 2r/n$ and achieve

$$\|\beta_l (V_S^\top \widehat{A})_{[1:k]} - (\widehat{A}^l)_{[1:k]}\|_1 + \|\beta_l V_S^\top\|_1 = o(k^\gamma),$$

at the same time. Then,

$$\mathbb{E}[t] \leq (O(k\|(V_S)^l\|_2) + 2^{-\Theta(k)})s = (O(k\sqrt{2r/n}) + 2^{-\Theta(k)})s.$$

Due to a Markov inequality,

$$\Pr[t > s/2 > n/4] \leq O(k\sqrt{2r/n}) + 2^{-\Theta(k)} = O(k^{1+(c_1-c_2)/2}) + 2^{-\Theta(k)}.$$

The equality follows since $r = \Theta(k^{c_1}), n = \Theta(k^{c_2})$. Plugging it into Equation (17.28), now we can conclude, with probability at least $1 - O(k^{1+(c_1-c_2)/2}) - 2^{-\Theta(k)}$, $\forall B \in \mathbb{R}^{n\times r}$

$$\|BS\widehat{A} - \widehat{A}\|_1 \geq \sum_{l:\|V_S^l\|_2^2 \leq 2r/n} \|(BU_S\Sigma_S)^l(V_S^\top \widehat{A}) - \widehat{A}^l\|_1 \geq n/4 \cdot \Omega(k^\gamma) = \Omega(n \cdot k^\gamma).$$

□

Theorem 17.6.26 (Hardness for row subset selection)**.** *For any $k \geq 1$, any constant $c \geq 1$, let $n = O(k^c)$, and let $\mathcal{A}(k, n)$ denote the same distribution stated in Theorem 17.6.25. For matrix $\widehat{A} \sim \mathcal{A}(k, n)$, with positive probability, there is no algorithm that is able to output $B \in \mathbb{R}^{n\times(n+k)}$ in the row span of any $r = n/2$ rows of $\widehat{A}$ such that*

$$\|\widehat{A} - B\|_1 \leq O(k^{0.5-\alpha}) \min_{\text{rank}-k\ A'} \|A' - \widehat{A}\|_1,$$

where $\alpha \in (0, 0.5)$ is a constant which can be arbitrarily small.

Proof. For convenience, we define $\gamma = 0.5 - \alpha$ which can be an arbitrary constant in $(0, 0.5)$. Since the last n columns of $\widehat{A}$ is an identity matrix, we can fit the first k columns of $\widehat{A}$, so we have

$$\min_{\text{rank}-k\ A'} \|A' - \widehat{A}\|_1 \leq n.$$

We want to argue that $\forall B \in \mathbb{R}^{n\times(k+n)}$ in the row span of any $r = n/2$ rows of $\widehat{A}$,

$$\|\widehat{A} - B\|_1 \geq \Omega(n \cdot k^\gamma).$$

Let $A^\top \in \mathbb{R}^{n\times k}$ be the first k columns of $\widehat{A}$, and let $S \subset [n]$ be a set of indices of chosen rows of $\widehat{A}$ with $k \leq |S| \leq r$. Let $M^S \in \mathbb{R}^{k\times n}$ with the i^{th} column $M^S_i = A_i$ if $i \in S$ and $M^S_i = 0$ otherwise. We use $\widehat{M}^S \in \mathbb{R}^{k\times r}$ to be M^S without those columns of zeros, so it is a random matrix with each entry i.i.d. Gaussian $N(0,1)$. Then the minimum cost of using a matrix in the span of rows of $\widehat{A}$ with index in S to fit $\widehat{A}$ is at least:

$$\sum_{l\notin S} \min_{x_l\in\mathbb{R}^n} \left(\|M^S x_l - A_l\|_1 + \|x_l\|_1 - 1\right).$$

The part of $\|M^S x_l - A_l\|_1$ is just the cost on the l^{th} row of the first k columns of $\widehat{A}$, and the part of $\|x_l\|_1 - 1$ is just the lower bound of the cost on the l^{th} row of the last n columns of $\widehat{A}$.

Claim 17.6.27. *$A, \widehat{M}^S \in \mathbb{R}^{k\times n}, \gamma \in (0, 0.5)$,*

$$\Pr\left[\widehat{\mathcal{E}}(\widehat{M}^S, I_r, 0.75 - \gamma/2, \gamma) \;\middle|\; \widehat{\mathcal{E}}(A, I_n, 0.75 - \gamma/2, \gamma)\right] = 1.$$

Proof. Suppose $\widehat{\mathcal{E}}(A, I_n, 0.75 - \gamma/2, \gamma)$ happens. Since M^S has just a subset of columns of A, $\|M^S\|_2 \leq \|A\|_2 \leq \sqrt{n}$. Notice that $\forall x \in \mathbb{R}^n$ with $\|x\|_1 \leq O(k^\gamma)$ and each non-zero coordinate of x is at most $O(1/k^{0.75-\gamma/2})$, $M^S x \equiv M^S x^S \equiv A x^S$, where $x^S \in \mathbb{R}^n$ has ${x^S}_i = x_i$ if $i \in S$ and ${x^S}_i = 0$ otherwise. Because $\|x^S\|_1 \leq O(k^\gamma)$ and x^S has each coordinate in absolute value at most $O(1/k^{0.75-\gamma/2})$, Ax^S has at most $O(k/\log k)$ coordinates in absolute value at least $\Omega(1/\log k)$. So, $M^S x = Ax^S$ has at most $O(k/\log k)$ coordinates in absolute value at least $\Omega(1/\log k)$. □

We denote $\text{cost}(S,l) = \min_{x_l \in \mathbb{R}^n} \left(\|M^S x_l - A_l\|_1 + \|x_l\|_1 - 1 \right)$. Since $\forall l \notin S$, A_l are independent, and they are independent from M^S, due to Lemma 17.6.20,

$$\Pr\left[\sum_{l \notin S} \text{cost}(S,l) \leq O(n \cdot k^\gamma) \;\middle|\; \widehat{\mathcal{E}}(\widehat{M}^S, I_r, 0.75 - \gamma/2, \gamma) \right] \leq 2^{-\Theta(rk)}. \tag{17.31}$$

Now we just want to upper bound the following:

$$\begin{aligned}
& \Pr\left[\exists S \subset [n], |S| \leq r, \sum_{l \notin S} \text{cost}(S,l) \leq O(n \cdot k^\gamma) \right] \\
\leq & \Pr\left[\exists S \subset [n], |S| \leq r, \sum_{l \notin S} \text{cost}(S,l) \leq O(n \cdot k^\gamma) \;\middle|\; \widehat{\mathcal{E}}(A, I_n, 0.75 - \gamma/2, \gamma) \right] \\
+ & \Pr\left[\neg \widehat{\mathcal{E}}(A, I_n, 0.75 - \gamma/2, \gamma) \right] \\
\leq & \sum_{S \subset [n], |S| \leq r} \Pr\left[\sum_{l \notin S} \text{cost}(S,l) \leq O(n \cdot k^\gamma) \;\middle|\; \widehat{\mathcal{E}}(A, I_n, 0.75 - \gamma/2, \gamma) \right] \\
+ & \Pr\left[\neg \widehat{\mathcal{E}}(A, I_n, 0.75 - \gamma/2, \gamma) \right] \\
\leq & \sum_{S \subset [n], |S| \leq r} \Pr\left[\sum_{l \notin S} \text{cost}(S,l) \leq O(n \cdot k^\gamma) \;\middle|\; \widehat{\mathcal{E}}(A, I_n, 0.75 - \gamma/2, \gamma) \right] + 2^{-\Theta(k)} \\
\leq & \sum_{S \subset [n], |S| \leq r} \Pr\left[\sum_{l \notin S} \text{cost}(S,l) \leq O(n \cdot k^\gamma) \;\middle|\; \widehat{\mathcal{E}}(\widehat{M}^S, I_r, 0.75 - \gamma/2, \gamma) \right] + 2^{-\Theta(k)} \\
\leq & (n+1)^r 2^{-\Theta(rk)} + 2^{-\Theta(k)} \\
\leq & 2^{-\Theta(rk)} + 2^{-\Theta(k)} \\
\leq & 2^{-\Theta(k)}.
\end{aligned}$$

The second inequality follows by a union bound. The third inequality follows by Lemma 17.6.18. The fourth inequality follows by Claim 17.6.27. The fifth inequality is due to Equation (17.31). The sixth inequality follows by $n \leq O(k^c), r = n/2$. Thus, with probability at least $1 - 2^{-\Theta(k)}$,

$\forall B \in \mathbb{R}^{n \times (n+k)}$ which is in the span of any $r \leq n/2$ rows of $\widehat{A}$,

$$\|B - \widehat{A}\|_1 \geq \Omega(n \cdot k^\gamma).$$

Then, we have completed the proof. □

Definition 17.6.3. Given a matrix $A \in \mathbb{R}^{n \times d}$, a matrix $S \in \mathbb{R}^{r \times n}$, $k \geq 1$ and $\gamma \in (0, \frac{1}{2})$, we say that an algorithm $\mathcal{M}(A, S, k, \gamma)$ which outputs a matrix $B \in \mathbb{R}^{n \times r}$ "succeeds", if

$$\|BSA - A\|_1 \leq k^\gamma \cdot \min_{\mathrm{rank}-k \ A'} \|A' - A\|_1,$$

holds.

Theorem 17.6.28 (Hardness for oblivious embedding)**.** *Let Π denote a distribution over matrices $S \in \mathbb{R}^{r \times n}$. For any $k \geq 1$, any constant $\gamma \in (0, \frac{1}{2})$, arbitrary constants $c_1, c_2 > 0$ and $\min(n, d) \geq \Omega(k^{c_2})$, if for all $A \in \mathbb{R}^{n \times d}$, it holds that*

$$\Pr_{S \sim \Pi}[\mathcal{M}(A, S, k, \gamma) \text{ succeeds }] \geq \Omega(1/k^{c_1}).$$

Then r must be at least $\Omega(k^{c_2 - 2c_1 - 2})$.

Proof. We borrow the idea from [NN14, PSW17]. We use Yao's minimax principle [Yao77] here. Let $\mathcal{D}$ be an arbitrary distribution over $\mathbb{R}^{n \times d}$, then

$$\Pr_{A \sim \mathcal{D}, S \sim \Pi}[\mathcal{M}(A, S, k, \gamma)] \geq 1 - \delta.$$

It means that there is a fixed S_0 such that

$$\Pr_{A \sim \mathcal{D}}[\mathcal{M}(A, S_0, k, \gamma)] \geq 1 - \delta.$$

Therefore, we want to find a hard distribution $\mathcal{D}_{\text{hard}}$ that if

$$\Pr_{A\sim\mathcal{D}_{\text{hard}}}[\mathcal{M}(A, S_0, k, \gamma)] \geq 1 - \delta,$$

S_0 must have at least some larger $\text{poly}(k)$ rows.

Here, we just use the distribution $\mathcal{A}(k, \Omega(k^{c_2}))$ described in Theorem 17.6.25 as our hard distribution. We can just fill zeros to expand the size of matrix to $n \times d$. We can complete the proof by using Theorem 17.6.25.

□

Remark 17.6.1. Actually, in Lemma 17.6.18 and Lemma 17.6.20, the reason we need $\beta > \gamma > 0$ is that we want $k^{\beta-\gamma} = \omega(\text{poly}(\log k))$, and the reason we need $\beta + \gamma < 1$ is that we want $k^{\beta+\gamma}\,\text{poly}(\log k) = o(k)$. Thus we can replace all the k^γ by $\sqrt{k}/\,\text{poly}(\log k)$, e.g., $\sqrt{k}/\log^{20} k$, and replace all the k^β by $\sqrt{k}\,\text{poly}(\log k)$ with a smaller $\text{poly}(\log k)$, e.g., $\sqrt{k}\log^{10} k$. Our proofs still work. Therefore, if we replace the approximation ratio in Theorem 17.6.25, Theorem 17.6.26, and Theorem 17.6.28 to be $\sqrt{k}/\log^c k$ where c is a sufficiently large constant, the statements are still correct.

17.7 ℓ_p-Low Rank Approximation

This section presents some fundamental lemmas for ℓ_p-low rank approximation problems. Using these lemmas, all the algorithms described for ℓ_1-low rank approximation problems can be extended to ℓ_p-low rank approximation directly. We only state the important Lemmas in this section, due to most of the proofs in this section being identical to the proofs in Section 17.5.

17.7.1 Definitions

This section is just a generalization of Section 17.5.1 to the ℓ_p setting when $1 < p < 2$.

Definition 17.7.1. Given a matrix $M \in \mathbb{R}^{n\times d}$, if matrix $S \in \mathbb{R}^{m\times n}$ satisfies

$$\|SM\|_p^p \leq c_1\|M\|_p^p,$$

then S has at most c_1-dilation on M.

Definition 17.7.2. Given a matrix $U \in \mathbb{R}^{n\times k}$, if matrix $S \in \mathbb{R}^{m\times n}$ satisfies

$$\forall x \in \mathbb{R}^k, \|SUx\|_p^p \geq \frac{1}{c_2}\|Ux\|_p^p,$$

then S has at most c_2-contraction on U.

Definition 17.7.3. Given matrices $U \in \mathbb{R}^{n\times k}, A \in \mathbb{R}^{n\times d}$, denote $V^* = \arg\min_{V\in\mathbb{R}^{k\times d}} \|UV - A\|_p^p$. If matrix $S \in \mathbb{R}^{m\times n}$ satisfies

$$\forall V \in \mathbb{R}^{k\times d}, \|SUV - SA\|_p^p \geq \frac{1}{c_3}\|UV - A\|_p^p - c_4\|UV^* - A\|_p^p,$$

then S has at most (c_3, c_4)-contraction on (U, A).

Definition 17.7.4. A (c_5, c_6) ℓ_p-subspace embedding for the column space of an $n\times k$ matrix U is a matrix $S \in \mathbb{R}^{m\times n}$ for which all $x \in \mathbb{R}^k$

$$\frac{1}{c_5}\|Ux\|_p^p \leq \|SUx\|_p^p \leq c_6\|Ux\|_p^p.$$

Definition 17.7.5. Given matrices $U \in \mathbb{R}^{n\times k}, A \in \mathbb{R}^{n\times d}$, denote $V^* = \arg\min_{V\in\mathbb{R}^{k\times d}} \|UV - A\|_p^p$. Let $S \in \mathbb{R}^{m\times n}$. If for all $c \geq 1$, and if for any $\widehat{V} \in \mathbb{R}^{k\times d}$ which satisfies

$$\|SU\widehat{V} - SA\|_p^p \leq c \cdot \min_{V\in\mathbb{R}^{k\times d}} \|SUV - SA\|_p^p,$$

it holds that

$$\|U\widehat{V} - A\|_p^p \leq c \cdot c_7 \cdot \|UV^* - A\|_p^p,$$

then S provides a c_7-multiple-regression-cost preserving sketch of (U, A).

Definition 17.7.6. Given matrices $L \in \mathbb{R}^{n \times m_1}, N \in \mathbb{R}^{m_2 \times d}, A \in \mathbb{R}^{n \times d}, k \geq 1$, let

$$X^* = \arg \min_{\text{rank}-k \ X} \|LXN - A\|_p^p.$$

Let $S \in \mathbb{R}^{m \times n}$. If for all $c \geq 1$, and if for any rank $-k$ $\widehat{X} \in \mathbb{R}^{m_1 \times m_2}$ which satisfies

$$\|SL\widehat{X}N - SA\|_p^p \leq c \cdot \min_{\text{rank}-k \ X} \|SLXN - SA\|_p^p,$$

it holds that

$$\|L\widehat{X}N - A\|_p^p \leq c \cdot c_8 \cdot \|LX^*N - A\|_p^p,$$

then S provides a c_8-restricted-multiple-regression-cost preserving sketch of (L, N, A, k).

17.7.2 Properties

Lemma 17.7.1. *Given matrices $A \in \mathbb{R}^{n\times d}, U \in \mathbb{R}^{n\times k}$, let $V^* = \arg\min_{V\in\mathbb{R}^{k\times d}} \|UV - A\|_p^p$. If $S \in \mathbb{R}^{m\times n}$ has at most c_1-dilation on $UV^* - A$, i.e.,*

$$\|S(UV^* - A)\|_p^p \le c_1\|UV^* - A\|_p^p,$$

and it has at most c_2-contraction on U, i.e.,

$$\forall x \in \mathbb{R}^k, \|SUx\|_p^p \ge \frac{1}{c_2}\|Ux\|_p^p,$$

then S has at most $(2^{2p-2}c_2, c_1 + 2^{1-p}\frac{1}{c_2})$-contraction on (U, A), i.e.,

$$\forall V \in \mathbb{R}^{k\times d}, \|SUV - SA\|_p^p \ge \frac{1}{c_2}\|UV - A\|_p^p - (c_1 + \frac{1}{c_2})\|UV^* - A\|_p^p,$$

Proof. Let $A \in \mathbb{R}^{n\times d}, U \in \mathbb{R}^{n\times k}$, and $S \in \mathbb{R}^{m\times n}$ be the same as that described in the lemma. Then $\forall V \in \mathbb{R}^{k\times d}$

$$\begin{aligned}
\|SUV - SA\|_p^p &\ge 2^{1-p}\|SUV - SUV^*\|_p^p - \|SUV^* - SA\|_p^p \\
&\ge 2^{1-p}\|SUV - SUV^*\|_p^p - c_1\|UV^* - A\|_p^p \\
&= 2^{1-p}\|SU(V - V^*)\|_p^p - c_1\|UV^* - A\|_p^p \\
&= 2^{1-p}\sum_{j=1}^{d}\|SU(V - V^*)_j\|_p^p - c_1\|UV^* - A\|_p^p \\
&\ge 2^{1-p}\sum_{j=1}^{d}\frac{1}{c_2}\|U(V - V^*)_j\|_p^p - c_1\|UV^* - A\|_p^p \\
&= 2^{1-p}\frac{1}{c_2}\|UV - UV^*\|_p^p - c_1\|UV^* - A\|_p^p \\
&\ge 2^{2-2p}\frac{1}{c_2}\|UV - A\|_p^p - 2^{1-p}\frac{1}{c_2}\|UV^* - A\|_p^p - c_1\|UV^* - A\|_p^p \\
&= 2^{2-2p}\frac{1}{c_2}\|UV - A\|_p^p - \left((2^{1-p}\frac{1}{c_2} + c_1)\|UV^* - A\|_p^p\right).
\end{aligned}$$

The first inequality follows by Fact 17.7.8. The second inequality follows since S has at most c_1 dilation on $UV^* - A$. The third inequality follows since S has at most c_2 contraction on U. The fourth inequality follows by Fact 17.7.8. □

Lemma 17.7.2. *Given matrices $A \in \mathbb{R}^{n\times d}, U \in \mathbb{R}^{n\times k}$, let $V^* = \arg\min_{V\in\mathbb{R}^{k\times d}} \|UV - A\|_p^p$. If $S \in \mathbb{R}^{m\times n}$ has at most c_1-dilation on $UV^* - A$, i.e.,*

$$\|S(UV^* - A)\|_p^p \leq c_1\|UV^* - A\|_p^p,$$

and has at most c_2-contraction on U, i.e.,

$$\forall x \in \mathbb{R}^k, \|SUx\|_p^p \geq \frac{1}{c_2}\|Ux\|_p^p,$$

then S provides a $2^{p-1}(2c_1c_2 + 1)$-multiple-regression-cost preserving sketch of (U, A), i.e., for all $c \geq 1$, for any $\widehat{V} \in \mathbb{R}^{k\times d}$ which satisfies

$$\|SU\widehat{V} - SA\|_p^p \leq c \cdot \min_{V\in\mathbb{R}^{k\times d}} \|SUV - SA\|_p^p,$$

it has

$$\|U\widehat{V} - A\|_p^p \leq c \cdot 2^{p-1}(2c_1c_2 + 1) \cdot \|UV^* - A\|_p^p,$$

Proof. Let $S \in \mathbb{R}^{m\times n}, A \in \mathbb{R}^{n\times d}, U \in \mathbb{R}^{n\times k}, V^*, \widehat{V} \in \mathbb{R}^{k\times d}$, and c be the same as stated in the lemma.

$$\begin{aligned}
\|U\widehat{V} - A\|_p^p &\leq 2^{2p-2}c_2\|SU\widehat{V} - SA\|_p^p + (2^{p-1} + 2^{2p-2}c_1c_2)\|UV^* - A\|_p^p \\
&\leq 2^{2p-2}c_2c \min_{V\in\mathbb{R}^{k\times d}} \|SUV - SA\|_p^p + (2^{p-1} + 2^{2p-2}c_1c_2)\|UV^* - A\|_p^p \\
&\leq 2^{2p-2}c_2c\|SUV^* - SA\|_p^p + (2^{p-1} + 2^{2p-2}c_1c_2)\|UV^* - A\|_p^p \\
&\leq 2^{2p-2}c_1c_2c\|UV^* - A\|_p^p + (2^{p-1} + 2^{2p-2}c_1c_2)\|UV^* - A\|_p^p \\
&\leq c \cdot 2^{p-1}(1 + 2c_1c_2)\|UV^* - A\|_p^p.
\end{aligned}$$

The first inequality follows by Lemma 17.7.1. The second inequality follows by the guarantee of $\widehat{V}$. The fourth inequality follows since S has at most c_1-dilation on $UV^* - A$. The fifth inequality follows since $c \geq 1$. □

Lemma 17.7.3. *Given matrices $L \in \mathbb{R}^{n \times m_1}, N \in \mathbb{R}^{m_2 \times d}, A \in \mathbb{R}^{n \times d}, k \geq 1$, let*

$$X^* = \arg \min_{\text{rank}-k \ X} \|LXN - A\|_p^p.$$

If $S \in \mathbb{R}^{m \times n}$ has at most c_1-dilation on $LX^*N - A$, i.e.,

$$\|S(LX^*N - A)\|_p^p \leq c_1 \|LX^*N - A\|_p^p,$$

and has at most c_2-contraction on L, i.e.,

$$\forall x \in \mathbb{R}^{m_1} \|SLx\|_p^p \geq \|Lx\|_p^p,$$

then S provides a $2^{p-1}(2c_1c_2+1)$-restricted-multiple-regression-cost preserving sketch of (L, N, A, k), i.e., for all $c \geq 1$, for any rank $-k$ *$\widehat{X} \in \mathbb{R}^{m_1 \times m_2}$ which satisfies*

$$\|SL\widehat{X}N - SA\|_p^p \leq c \cdot \min_{\text{rank}-k \ X} \|SLXN - SA\|_p^p,$$

it has

$$\|L\widehat{X}N - A\|_p^p \leq c \cdot 2^{p-1}(2c_1c_2 + 1) \cdot \|LX^*N - A\|_p^p.$$

Proof. Let $S \in \mathbb{R}^{m \times n}, L \in \mathbb{R}^{n \times m_1}, \widehat{X} \in \mathbb{R}^{m_1 \times m_2}, X^* \in \mathbb{R}^{m_1 \times m_2}, N \in \mathbb{R}^{m_2 \times d}, A \in \mathbb{R}^{n \times d}$ and

$c \geq 1$ be the same as stated in the lemma.

$$\begin{aligned}
\|SL\widehat{X}N - SA\|_p^p &\geq 2^{1-p}\|SL\widehat{X}N - SLX^*N\|_p^p - \|SLX^*N - SA\|_p^p \\
&\geq 2^{1-p}\frac{1}{c_2}\|L(\widehat{X}N - X^*N)\|_p^p - c_1\|LX^*N - A\|_p^p \\
&\geq 2^{2-2p}\frac{1}{c_2}\|L\widehat{X}N - A\|_p^p - 2^{1-p}\frac{1}{c_2}\|LX^*N - A\|_1 - c_1\|LX^*N - A\|_p^p \\
&= 2^{2-2p}\frac{1}{c_2}\|L\widehat{X}N - A\|_p^p - (2^{1-p}\frac{1}{c_2} + c_1)\|LX^*N - A\|_p^p.
\end{aligned}$$

The inequality follows from the Fact 17.7.8. The second inequality follows since S has at most c_2-contraction on L, and it has at most c_1-dilation on $LX^*N - A$. The third inequality follows by Fact 17.7.8.

It follows that

$$\begin{aligned}
\|L\widehat{X}N - A\|_p^p &\leq 2^{2p-2}c_2\|SL\widehat{X}N - SA\|_p^p + (2^{p-1} + 2^{2p-2}c_1c_2)\|LX^*N - A\|_p^p \\
&\leq 2^{2p-2}c_2c \cdot \min_{\text{rank}-k\ X} \|SLXN - SA\|_p^p + (2^{p-1} + 2^{2p-2}c_1c_2)\|LX^*N - A\|_p^p \\
&\leq 2^{2p-2}c_2c \cdot \|SLX^*N - SA\|_p^p + (2^{p-1} + 2^{2p-2}c_1c_2)\|LX^*N - A\|_p^p \\
&\leq 2^{2p-2}cc_1c_2 \cdot \|LX^*N - A\|_p^p + (2^{p-1} + 2^{2p-2}c_1c_2)\|LX^*N - A\|_p^p \\
&\leq c \cdot 2^{p-1}(1 + 2c_1c_2)\|LX^*N - A\|_p^p.
\end{aligned}$$

The first inequality directly follows from the previous one. The second inequality follows from the guarantee of $\widehat{X}$. The fourth inequality follows since S has at most c_1 dilation on $LX^*N - A$. The fifth inequality follows since $c \geq 1$. □

Lemma 17.7.4. *Given matrices $L \in \mathbb{R}^{n\times m_1}, N \in \mathbb{R}^{m_2\times d}, A \in \mathbb{R}^{n\times d}, k \geq 1$, let*

$$X^* = \arg\min_{\text{rank}-k\ X} \|LXN - A\|_p^p.$$

Let $T_1 \in \mathbb{R}^{t_1 \times n}$ have at most c_1-dilation on $LX^*N - A$, and have at most c_2-contraction on L. Let

$$\widetilde{X} = \arg \min_{\text{rank}-k \ X} \|T_1 LXN - T_1 A\|_p^p.$$

Let $T_2^\top \in \mathbb{R}^{t_2 \times d}$ have at most c_1'-dilation on $(T_1 L\widetilde{X}N - T_1 A)^\top$, and at most c_2'-contraction on $N^\top$. Then, for all $c \geq 1$, for any $\text{rank}-k$ $\widehat{X} \in \mathbb{R}^{m_1 \times m_2}$ which satisfies

$$\|T_1(L\widehat{X}N - SA)T_2\|_p^p \leq c \cdot \min_{\text{rank}-k \ X} \|T_1(LXN - A)T_2\|_p^p,$$

it holds that

$$\|L\widehat{X}N - A\|_p^p \leq c \cdot 2^{2p-2}(2c_1c_2 + 1)(2c_1'c_2' + 1) \cdot \|LX^*N - A\|_p^p.$$

Proof. Apply Lemma 17.5.3 for sketching matrix T_2. Then for any $c \geq 1$, any $\text{rank}-k$ $\widehat{X} \in \mathbb{R}^{m_1 \times m_2}$ which satisfies

$$\|T_1(L\widehat{X}N - A)T_2\|_p^p \leq c \cdot \min_{\text{rank}-k \ X} \|T_1(LXN - A)T_2\|_p^p,$$

it has

$$\|T_1(L\widehat{X}N - A)\|_p^p \leq c \cdot 2^{p-1}(2c_1'c_2' + 1) \cdot \|T_1(L\widetilde{X}N - A)\|_p^p.$$

Apply Lemma 17.5.3 for sketch matrix T_1. Then for any $c \geq 1$, any $\text{rank}-k$ $\widehat{X} \in \mathbb{R}^{m_1 \times m_2}$ which satisfies

$$\|T_1(L\widehat{X}N - A)\|_p^p \leq c2^{p-1}(2c_1'c_2' + 1) \cdot \min_{\text{rank}-k \ X} \|T_1(L\widetilde{X}N - A)\|_p^p,$$

it has

$$\|L\widehat{X}N - A\|_p^p \leq c \cdot 2^{2p-2}(2c_1c_2 + 1)(2c_1'c_2' + 1) \cdot \|LX^*N - A\|_p^p.$$

□

Lemma 17.7.5. *Given matrices $M \in \mathbb{R}^{n\times d}, U \in \mathbb{R}^{n\times t}$, $d \geq t = \text{rank}(U), n \geq d \geq r = \text{rank}(M)$. If sketching matrix $S \in \mathbb{R}^{m\times n}$ is drawn from any of the following probability distributions on matrices, with .99 probability, S has at most c_1-dilation on M, i.e.,*

$$\|SM\|_p^p \leq c_1 \|M\|_p^p,$$

and S has at most c_2-contraction on U, i.e.,

$$\forall x \in \mathbb{R}^t, \ \|SUx\|_p^p \geq \frac{1}{c_2}\|Ux\|_p^p,$$

where c_1, c_2 are parameters depend on the distribution over S.

(I) *$S \in \mathbb{R}^{m\times n}$ is a dense matrix with entries drawn from a p-stable distribution: a matrix with i.i.d. standard p-stable random variables. If $m = O(t \log t)$, then $c_1 c_2 = O(\log d)$.*

(II) *$S \in \mathbb{R}^{m\times n}$ is a sparse matrix with some entries drawn from a p-stable distribution: $S = TD$, where $T \in \mathbb{R}^{m\times n}$ has each column drawn i.i.d. from the uniform distribution over standard basis vectors of $\mathbb{R}^m$, and $D \in \mathbb{R}^{n\times n}$ is a diagonal matrix with each diagonal entry drawn from i.i.d. from the standard p-stable distribution. If $m = O(t^5 \log^5 t)$, then $c_1 c_2 = O(t^{2/p} \log^{2/p} t \log d)$. If $m = O((t+r)^5 \log^5(t+r))$, then $c_1 c_2 = O(\min(t^{2/p} \log^{2/p} t \log d, r^{3/p} \log^{3/p} r))$.*

(III) *$S \in \mathbb{R}^{m\times n}$ is a sampling and rescaling matrix (notation $S \in \mathbb{R}^{n\times n}$ denotes a diagonal sampling and rescaling matrix with m non-zero entries): If S samples and reweights $m = O(t \log t \log\log t)$ rows of U, selecting each with probability proportional to the i^{th} row's ℓ_p Lewis weight and reweighting by the inverse probability, then $c_1 c_2 = O(1)$.*

In the above, if we replace S with $\sigma \cdot S$ where $\sigma \in \mathbb{R}\backslash\{0\}$ is any scalar, then the relation between m and $c_1 c_2$ can be preserved.

For (I), it is implied by Lemma E.17, Lemma E.19. Also see from [SW11][3].

For (II), if $m = O(t^5 \log^5 t)$, then $c_1 c_2 = O(t^{2/p} \log^{2/p} t \log d)$ is implied by Corollary 17.5.19 and Lemma 17.7.13 and Theorem 4 in [MM13]. If $m = O((t+r)^5 \log^5(t+r))$, $c_1 c_2 = O(r^{3/p} \log^{3/p} r)$ is implied by [MM13].

For (III), it is implied by [CP15] and Lemma 17.5.21.

[3]Full version.

17.7.3 Tools and inequalities

Lemma 17.7.6 (Lemma 9 in [MM13], Upper Tail Inequality for p-stable Distributions)**.** *Let $p \in (1,2)$ and $m \geq 3$. $\forall i \in [m]$, let X_i be m(not necessarily independent) random variables sampled from D_p, and let $\gamma_i > 0$ with $\gamma = \sum_{i=1}^{m} \gamma_i$. Let $X = \sum_{i=1}^{m} \gamma_i |X_i|^p$. Then for any $t \geq 1$,*

$$\Pr[X \geq t\alpha_p \gamma] \leq \frac{2\log(mt)}{t}.$$

We first review some facts about the p-norm and q-norm,

Fact 17.7.7. *For any $p \geq q > 0$ and any $x \in \mathbb{R}^k$,*

$$\|x\|_p \leq \|x\|_q \leq k^{\frac{1}{q}-\frac{1}{p}} \|x\|_p.$$

We provide the triangle inequality for the p-norm,

Fact 17.7.8. *For any $p \in (1,2)$, for any $x, y \in \mathbb{R}^k$,*

$$\|x+y\|_p \leq \|x\|_p + \|y\|_p, \text{ and } \|x+y\|_p^p \leq 2^{p-1}(\|x\|_p^p + \|y\|_p^p).$$

Fact 17.7.9 (Hölder's inequality)**.** *For any $x, y \in \mathbb{R}^k$, if $\frac{1}{p} + \frac{1}{q} = 1$, then $|x^\top y| \leq \|x\|_p \|y\|_q$.*

We give the definition of a well-conditioned basis for ℓ_p,

Definition 17.7.7. Let $p \in (1,2)$. A basis U for the range of A is (α, β, p)-conditioned if $\|U\|_p \leq \alpha$ and for all $x \in \mathbb{R}^k$, $\|x\|_q \leq \beta \|Ux\|_p$. We will say U is well-conditioned if α and β are low-degree polynomials in k, independent of n.

Proof. We first show an upper bound,

$$\|Ux\|_p \leq \|U\|_p \cdot \|x\|_p \leq \alpha \|x\|_p$$

Then we show a lower bound,

$$\|Ux\|_p \geq \frac{1}{\beta} \|x\|_q$$

For any p and q with $1/p + 1/q = 1$, by Hölder's inequality we have

$$|x^\top y| \leq \|x\|_p \cdot \|y\|_q$$

choosing y to be the vector that has 1 everywhere, $\|x\|_1 \leq \|x\|_p k^{1/q}$ □

17.7.4 Dense p-stable transform

This section states the main tools for the dense p-stable transform. The proof is identical to that for the dense Cauchy transform.

Lemma 17.7.10. *Given matrix $A \in \mathbb{R}^{n\times d}$ and $p \in (1,2)$, define $U^* \in \mathbb{R}^{n\times k}$, $V^* \in \mathbb{R}^{k\times d}$ to be an optimal solution of $\min_{U\in\mathbb{R}^{n\times k}, V\in\mathbb{R}^{k\times d}} \|UV - A\|_p$. Choose a p-stable distribution matrix $S \in \mathbb{R}^{m\times n}$, rescaled by $\Theta(1/m^{1/p})$. Then we have*

$$\|SU^*V^* - SA\|_p^p \lesssim \log(md)\|U^*V^* - A\|_p^p$$

with probability at least $99/100$.

Proof. Let $P(0,1)$ denote the p-stable distribution. Then,

$$\begin{aligned}
\|SU^*V^* - SA\|_p^p \leq & \sum_{i=1}^{d} \|S(U^*V_i^* - A_i)\|_p^p \\
= & \sum_{i=1}^{d}\sum_{j=1}^{m} |\sum_{l=1}^{n} \frac{1}{m} S_{j,l}(U^*V_i^* - A_i)_l|^p && \text{where } S_{j,l} \sim P(0,1) \\
= & \frac{1}{m}\sum_{i=1}^{d}\sum_{j=1}^{m} |\widetilde{w}_{ij}(\sum_{l=1}^{n} |(U^*V_i^* - A_i)_l|^p)^{1/p}|^p && \text{where } \widetilde{w}_{ij} \sim P(0,1) \\
= & \frac{1}{m}\sum_{i=1}^{d}\sum_{j=1}^{m}\sum_{l=1}^{n} |(U^*V_i^* - A_i)_l|^p \cdot |\widetilde{w}_{ij}|^p \\
= & \frac{1}{m}\sum_{i=1}^{d}\sum_{j=1}^{m} \|U^*V_i^* - A_i\|_p^p \cdot w_{i+(j-1)d}^p, && \text{where } w_{i+(j-1)d} \sim |P(0,1)|
\end{aligned}$$

where the last step follows since each w_i can be thought of as a clipped half-p-stable random variable. Define X to be $\sum_{i=1}^{d}\sum_{j=1}^{m} \|U^*V_i^* - A_i\|_p^p \cdot w_{i+(j-1)d}^p$ and γ to be $\sum_{i=1}^{d}\sum_{j=1}^{m} \|U^*V_i^* -$

$A_i\|_p^p$. Then applying Lemma 17.7.6,

$$\Pr[X \geq t\alpha_p\gamma] \leq \frac{2\log(mdt)}{t}.$$

Choosing $t = \Theta(\log(md))$, we have with probability .999,

$$X \lesssim \log(md)\alpha_p\gamma = \log(md)\alpha_p \sum_{i=1}^{d} \|U^*V_i^* - A_i\|_p^p,$$

where the last step follows by definition of γ. Thus, we can conclude that with probability .999, $\|\Pi(U^*V^* - A)\|_p^p \lesssim \log(md)\|U^*V^* - A\|_p^p$. □

Lemma 17.7.11. *Given matrix $A \in \mathbb{R}^{n\times d}$ and $p \in (1,2)$, define $U^* \in \mathbb{R}^{n\times k}$ to be the optimal solution of $\min_{U\in\mathbb{R}^{n\times k}, V\in\mathbb{R}^{k\times d}} \|UV - A\|_p$. Choose a matrix of i.i.d. p-stable random variables $S \in \mathbb{R}^{m\times n}$. Then for all $V \in \mathbb{R}^{k\times n}$, we have*

$$\|SU^*V - SA\|_p^p \gtrsim \|U^*V - A\|_p^p - O(\log(md))\|U^*V^* - A\|_p^p.$$

Lemma 17.7.12. *Let $p \in (1,2)$. Given an (α,β) ℓ_p well-conditioned basis, condition on the following two events,*

1. *For all $x \in \mathcal{N}$, $\|SUx\|_p \gtrsim \|Ux\|_p$.*
2. *For all $x \in \mathbb{R}^k$, $\|SUx\|_p \leq \text{poly}(k)\|Ux\|_p$.*

Then for all $x \in \mathbb{R}^k$, $\|SUx\|_p \gtrsim \|Ux\|_p$.

Proof. The proof is identical to Lemma 17.5.14 in Section 17.5. □

17.7.5 Sparse p-stable transform

This section states the main tools for the sparse p-stable transform. The proof is identical to that of the sparse Cauchy transform.

Lemma 17.7.13. *Let $p \in (1,2)$. Given matrix $A \in \mathbb{R}^{n \times d}$ with U^*, V^* an optimal solution of $\min_{U,V} \|UV - A\|_p$, let $\Pi = \sigma \cdot SC \in \mathbb{R}^{m \times n}$, where $S \in \mathbb{R}^{m \times n}$ has each column vector chosen independently and uniformly from the m standard basis vectors of $\mathbb{R}^m$, where C is a diagonal matrix with diagonals chosen independently from the standard p-stable distribution, and σ is a scalar. Then*

$$\|\Pi U^* V^* - \Pi A\|_p^p \lesssim \sigma \cdot \log(md) \cdot \|U^* V^* - A\|_p^p$$

holds with probability at least .999.

Proof. We define $\Pi = \sigma \cdot SC \in \mathbb{R}^{m \times n}$ as in the statement of the lemma. Then by the

definition of Π, we have,

$$
\begin{aligned}
&\|\Pi(U^*V^* - A)\|_p^p \\
&= \sum_{i=1}^{d} \|SC(U^*V_i^* - A_i)\|_p^p \\
&= \sum_{i=1}^{d} \left\| \begin{bmatrix} S_{11} & S_{12} & \cdots & S_{1n} \\ S_{21} & S_{22} & \cdots & S_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ S_{m1} & S_{m2} & \cdots & S_{mn} \end{bmatrix} \cdot \begin{bmatrix} c_1 & 0 & 0 & 0 \\ 0 & c_2 & 0 & 0 \\ 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & c_n \end{bmatrix} \cdot (U^*V_i^* - A_i) \right\|_p^p \\
&= \sum_{i=1}^{d} \left\| \begin{bmatrix} c_1S_{11} & c_2S_{12} & \cdots & c_nS_{1n} \\ c_1S_{21} & c_2S_{22} & \cdots & c_nS_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ c_1S_{m1} & c_2S_{m2} & \cdots & c_nS_{mn} \end{bmatrix} \cdot (U^*V_i^* - A_i) \right\|_p^p \\
&= \sum_{i=1}^{d} \left\| \sum_{l=1}^{n} c_l S_{1l} \cdot (U^*V_i^* - A_i)_l, \sum_{l=1}^{n} c_l S_{2l} \cdot (U^*V_i^* - A_i)_l, \cdots, \sum_{l=1}^{n} c_l S_{ml} \cdot (U^*V_i^* - A_i)_l \right\|_p^p \\
&= \sum_{i=1}^{d} \sum_{j=1}^{m} \left| \sum_{l=1}^{n} c_l S_{jl} \cdot (U^*V_i^* - A_i)_l \right|^p \text{ by } aX + bY \text{ and } (|a|^p + |b|^p)^{1/p} Z \text{ are identically distributed} \\
&= \sum_{i=1}^{d} \sum_{j=1}^{m} \left| \widetilde{w}_{ij} \cdot (\sum_{l=1}^{n} |S_{jl}(U^*V_i^* - A_i)_l|^p)^{1/p} \right|^p \text{ where } \widetilde{w}_{ij} \sim P(0,1) \\
&= \sum_{i=1}^{d} \sum_{j=1}^{m} \sum_{l=1}^{n} |S_{jl}(U^*V_i^* - A_i)_l|^p \cdot |\widetilde{w}_{ij}|^p \\
&= \sum_{i=1}^{d} \sum_{j=1}^{m} \sum_{l=1}^{n} |S_{jl}(U^*V_i^* - A_i)_l|^p \cdot w_{i+(j-1)d}^p, \text{ where } w_{i+(j-1)d} \sim |P(0,1)|
\end{aligned}
$$

where the last step follows since each w_i can be thought of as a clipped half-p-stable random variable. Define X to be $\sum_{i=1}^{d} \sum_{j=1}^{m} \sum_{l=1}^{n} |S_{jl}(U^*V_i^* - A_i)_l|^p \cdot w_{i+(j-1)d}^p$ and γ to be $\sum_{i=1}^{d} \sum_{j=1}^{m} \sum_{l=1}^{n} |S_{jl}(U^*V_i^* - A_i)_l|^p$. Then applying Lemma 17.7.6,

$$\Pr[X \geq t\alpha_p\gamma] \leq \frac{2\log(mdt)}{t}.$$

Choosing $t = \Theta(\log(md))$, we have with probability .999,

$$X \lesssim \log(md)\alpha_p\gamma = \log(md)\alpha_p \sum_{i=1}^{d}\sum_{l=1}^{n} |(U^*V_i^* - A_i)_l|^p,$$

where the last steps follows by

$$\begin{aligned}\gamma &= \sum_{i=1}^{d}\sum_{j=1}^{m}\sum_{l=1}^{n} |S_{jl}(U^*V_i^* - A_i)_l|^p \\ &= \sum_{i=1}^{d}\sum_{j=1}^{m}\sum_{l=1}^{n} |S_{jl}|^p|(U^*V_i^* - A_i)_l|^p \\ &= \sum_{i=1}^{d}\sum_{l=1}^{n} |(U^*V_i^* - A_i)_l|^p.\end{aligned}$$

Thus, we can conclude that with probability .999,

$$\|\Pi(U^*V^* - A)\|_p^p \lesssim \log(md)\|U^*V^* - A\|_p^p.$$

□

Lemma 17.7.14. *Given matrix $A \in \mathbb{R}^{n\times d}$ with U^*, V^* an optimal solution of $\min_{U,V} \|UV - A\|_p$, let $\Pi = \sigma \cdot SC \in \mathbb{R}^{m\times n}$, where $S \in \mathbb{R}^{m\times n}$ has each column vector chosen independently and uniformly from the m standard basis vectors of $\mathbb{R}^m$, and where C is a diagonal matrix with diagonals chosen independently from the standard p-stable distribution. Then with probability at least .999, for all $V \in \mathbb{R}^{k\times d}$,*

$$\|\Pi U^*V - \Pi A\|_p^p \geq \|U^*V - A\|_p^p - O(\sigma \log(md))\|U^*V^* - A\|_p^p.$$

Notice that $m = O(k^5 \log^5 k)$ and $\sigma = O((k \log k)^{2/p})$ according to Theorem 4 in [MM13].

17.7.6 ℓ_p-Lewis weights

This section states the main tools for ℓ_p-Lewis weights. The proof is identical to ℓ_1-Lewis weights.

Lemma 17.7.15. *For any $p \in (1,2)$. Given matrix $A \in \mathbb{R}^{n\times d}$, define $U^* \in \mathbb{R}^{n\times k}, V^* \in \mathbb{R}^{k\times d}$ to be an optimal solution of $\min_{U\in\mathbb{R}^{n\times k}, V\in\mathbb{R}^{k\times d}} \|UV - A\|_p$. Choose a diagonal matrix $D \in \mathbb{R}^{n\times n}$ according to the Lewis weights of U^*. We have that*

$$\|DU^*V^* - DA\|_p^p \lesssim \|U^*V^* - A\|_p^p,$$

holds with probability at least .99.

Lemma 17.7.16. *Let $p \in (1,2)$. Given matrix $A \in \mathbb{R}^{n\times d}$, define $U^* \in \mathbb{R}^{n\times k}, V^* \in \mathbb{R}^{k\times d}$ to be an optimal solution of $\min_{U\in\mathbb{R}^{n\times k}, V\in\mathbb{R}^{k\times d}} \|UV - A\|_p$. Choose a sampling and rescaling matrix $D \in \mathbb{R}^{n\times n}$ according to the Lewis weights of U^*. For all $V \in \mathbb{R}^{k\times d}$ we have*

$$\|DU^*V - DA\|_p^p \gtrsim \|U^*V - A\|_p^p - O(1)\|U^*V^* - A\|_p^p,$$

holds with probability at least .99.

17.8 EMD-Low Rank Approximation

In this section we explain how to embed EMD to ℓ_1. For more detailed background on the Earth-Mover Distance(EMD) problem, we refer the reader to [IT03, AIK08, ABIW09, IP11, BIRW16] and [SL09, LOG16]. Section 17.8.1 introduces some necessary notation and definitions for Earth-Mover Distance. Section 17.8.2 presents the main result for the Earth-Mover distance low rank approximation problem.

17.8.1 Definitions

Consider any two non-negative vectors $x, y \in \mathbb{R}_+^{[\Delta]^2}$ such that $\|x\|_1 = \|y\|_1$. Let $\Gamma(x, y)$ be the set of functions $\gamma : [\Delta]^2 \times [\Delta]^2 \to \mathbb{R}_+$, such that for any $i, j \in [\Delta]^2$ we have $\sum_l \gamma(i, l) = x_i$ and $\sum_l \gamma(l, j) = y_j$; that is, Γ is the set of possible "flows" from x to y. Then we define

$$\mathrm{EMD}(x, y) = \min_{\gamma \in \Gamma} \sum_{i,j \in [\Delta]^2} \gamma(i, j) \|i - j\|_1$$

to be the min cost flow from x to y, where the cost of an edge is its ℓ_1 distance.

Using the $\mathrm{EMD}(\cdot, \cdot)$ metric, for general vectors w, we define $\|\cdot\|_{\mathrm{EEMD}}$ distance (which is the same as [SL09]),

$$\|w\|_{\mathrm{EEMD}} = \min_{\substack{x - y + z = w \\ \|x\|_1 = \|y\|_1 \\ x, y \geq 0}} \mathrm{EMD}(x, y) + 2\Delta \|z\|_1.$$

Using $\|\cdot\|_{\mathrm{EEMD}}$ distance, for general matrices $X \in \mathbb{R}^{n \times d}$, we define the $\|\cdot\|_{1,\mathrm{EEMD}}$ distance,

$$\|X\|_{1,\mathrm{EEMD}} = \sum_{i=1}^{d} \|X_i\|_{\mathrm{EEMD}},$$

where X_i denotes the j-th column of matrix X.

17.8.2 Analysis of no contraction and no dilation bound

Lemma 17.8.1. *Given matrix* $A \in \mathbb{R}^{n\times d}$ *and* $U^*, V^* = \underset{U\in\mathbb{R}^{n\times k}, V\in\mathbb{R}^{k\times d}}{\arg\min} \|UV - A\|_{1,\text{EEMD}}$*, there exist sketching matrices* $S \in \mathbb{R}^{m\times n}$ *such that, with probability* .999*, for all* $V \in \mathbb{R}^{k\times d}$*,*

$$\|S(U^*V - A)\|_1 \geq \|U^*V - A\|_{1,\text{EEMD}}$$

holds.

Proof. Using Lemma 1 in [IT03], there exists a constant $C > 0$ such that for all $i \in [d]$,

$$C\|SU^*V_i - SA_i\|_1 \geq \|U^*V_i - A_i\|_{\text{EEMD}}. \tag{17.32}$$

Then taking a summation over all d terms and rescaling the matrix S, we obtain,

$$\sum_{i=1}^{d} \|S(U^*V_i - A_i)\|_1 \geq \|U^*V_i - A_i\|_{\text{EEMD}}$$

which completes the proof. □

Lemma 17.8.2. *Given matrix* $A \in \mathbb{R}^{n\times d}$ *and* $U^*, V^* = \underset{U\in\mathbb{R}^{n\times k}, V\in\mathbb{R}^{k\times d}}{\arg\min} \|UV - A\|_{1,\text{EEMD}}$*, there exist sketching matrices* $S \in \mathbb{R}^{m\times n}$ *such that*

$$\|S(U^*V^* - A)\|_1 \leq O(\log n)\|U^*V^* - A\|_{1,\text{EEMD}}$$

holds with probability at least .999*.*

Proof. Using Lemma 2 in [IT03], we have for any $i \in [d]$,

$$\mathbb{E}[\|SU^*V_i^* - SA_i\|_1] \leq O(\log n)\|U^*V_i^* - A_i\|_{\text{EEMD}}. \tag{17.33}$$

Then using that the expectation is linear, we have

$$\begin{aligned}\mathbb{E}[\|SU^*V^* - SA\|_1] = \ & \mathbb{E}[\sum_{i=1}^{d} \|SU^*V_i^* - SA_i\|_1] \\ = & \sum_{i=1}^{d} \mathbb{E}[\|SU^*V_i^* - SA_i\|_1] \\ \leq & \sum_{i=1}^{d} O(\log n)\|U^*V_i^* - A_i\|_{\mathrm{EEMD}} \qquad \text{by Equation (17.33)} \\ = & O(\log n)\|U^*V^* - A\|_{1,\mathrm{EEMD}}.\end{aligned}$$

Using Markov's inequality, we can complete the proof. □

Theorem 17.8.3. *Given a matrix $A \in \mathbb{R}^{n\times d}$, there exists an algorithm running in* $\mathrm{poly}(k, n, d)$ *time that is able to output $U \in \mathbb{R}^{n\times k}$ and $V \in \mathbb{R}^{k\times d}$ such that*

$$\|UV - A\|_{1,\mathrm{EEMD}} \leq \mathrm{poly}(k) \cdot \log d \cdot \log n \min_{\mathrm{rank}-k \ A_k} \|A_k - A\|_{1,\mathrm{EEMD}}$$

holds with probability .99.

Proof. First using Lemma 17.8.1 and Lemma 17.8.2, we can reduce the original problem into an ℓ_1-low rank approximation problem by choosing $m = \mathrm{poly}(n)$. Second, we can use our ℓ_1-low rank approximation algorithm to solve it. Notice that all of our ℓ_1-low rank approximation algorithms can be applied here. If we apply Theorem 17.4.6, we complete the proof.

□

Our current $\|\cdot\|_{1,\mathrm{EEMD}}$ is column-based. We can also define it to be row-based. Then we get a slightly better result by applying the ℓ_1-low rank algorithm.

Corollary 17.8.4. *Given a matrix $A \in \mathbb{R}^{n \times d}$, there exists an algorithm running in* $\text{poly}(k, n, d)$ *time that is able to output $U \in \mathbb{R}^{n \times k}$ and $V \in \mathbb{R}^{k \times d}$ such that*

$$\|UV - A\|_{1,\text{EEMD}} \leq \text{poly}(k) \cdot \log^2 d \min_{\text{rank}-k \ A_k} \|A_k - A\|_{1,\text{EEMD}}$$

holds with probability .99.

17.9 Hardness

This section presents our hardness results. Section 17.9.1 states several useful tools from literature. Section 17.9.2 shows that, it is **NP-hard** to get some multiplicative error. Assuming ETH is true, we provide a stronger hardness result in Section 17.9.3. Section 17.9.4 extends the result from the rank-1 case to the rank-k case.

17.9.1 Previous results

Definition 17.9.1 ($\|A\|_{\infty\to 1}$,[GV15]). Given matrix $A \in \mathbb{R}^{n\times d}$,

$$\|A\|_{\infty\to 1} = \min_{x\in\{-1,+1\}^n, y\in\{-1,+1\}^d} x^\top A y.$$

The following lemma says that computing $\|A\|_{\infty\to 1}$ for matrix A with entries in $\{-1,+1\}$ is equivalent to computing a best $\{-1,+1\}$ matrix which is an ℓ_1 norm rank-1 approximation to A.

Lemma 17.9.1 (Lemma 3 of [GV15]). *Given matrix $A \in \{-1,+1\}^{n\times d}$,*

$$\|A\|_{\infty\to 1} + \min_{x\in\{-1,+1\}^n, y\in\{-1,+1\}^d} \|A - xy^\top\|_1 = nd.$$

Lemma 17.9.2 (Theorem 2 of [GV15]). *Given $A \in \{-1,+1\}^{n\times d}$, we have*

$$\min_{x\in\{-1,+1\}^n, y\in\{-1,+1\}^d} \|A - xy^\top\|_1 = \min_{x\in\mathbb{R}^n, y\in\mathbb{R}^d} \|A - xy^\top\|_1.$$

Combining with Lemma 17.9.1 and Lemma 17.9.2, it implies that computing $\|A\|_{\infty\to 1}$ for $A \in \{-1,+1\}^{n\times d}$ is equivalent to computing the best ℓ_1 norm rank-1 approximation to the matrix A:

$$\|A\|_{\infty\to 1} + \min_{x\in\mathbb{R}^n, y\in\mathbb{R}^d} \|A - xy^\top\|_1 = nd.$$

Theorem 17.9.3 (**NP-hard** result, Theorem 1 of [GV15]). *Computing $\|A\|_{\infty\to 1}$ for matrix $A \in \{-1,+1\}^{n\times d}$ is* **NP-hard**.

The proof of the above theorem in [GV15] is based on the reduction from MAX-CUT problem. The above theorem implies that computing $\min_{x\in\mathbb{R}^n, y\in\mathbb{R}^d} \|A - xy^\top\|_1$ is also **NP-hard**.

17.9.2 Extension to multiplicative error ℓ_1-low rank approximation

The previous result only shows that solving the exact problem is **NP-hard**. This section presents a stronger hardness result, which says that, it is still **NP-hard** even if the goal is to find a solution that is able to achieve some multiplicative error. The proof in this section and the next section are based on the reduction from the MAX-CUT problem. For recent progress on MAX-CUT problem, we refer the readers to [GW95, BGS98, TSSW00, Hås01, KKMO07, FLP15].

Theorem 17.9.4. *Given* $A \in \{-1, +1\}^{n \times d}$, *computing an* $\widehat{x} \in \mathbb{R}^n, \widehat{y} \in \mathbb{R}^d$ *s.t.*

$$\|A - \widehat{x}^\top \widehat{y}\|_1 \leq (1 + \frac{1}{nd}) \min_{x \in \mathbb{R}^n, y \in \mathbb{R}^d} \|A - x^\top y\|_1$$

is **NP-hard**.

MAX-CUT decision problem: Given a positive integer c^* and an unweighted graph $G = (V, E)$ where V is the set of vertices of G and E is the set of edges of G, the goal is to determine whether there is a cut of G has at least c^* edges.

Lemma 17.9.5. MAX-CUT *decision problem is* **NP-hard**.

We give the definition for the Hadamard matrix,

Definition 17.9.2. The Hadamard matrix H_p of size $p \times p$ is defined recursively : $\begin{bmatrix} H_{p/2} & H_{p/2} \\ H_{p/2} & -H_{p/2} \end{bmatrix}$ with $H_2 = \begin{bmatrix} +1 & +1 \\ +1 & -1 \end{bmatrix}$.

For simplicity, we use H to denote H_p in the rest of the proof.

Recall the reduction shown in [GV15] which is from MAX-CUT to computing $\|\cdot\|_{\infty\to 1}$ for $\{-1,+1\}$ matrices. We do the same thing: for a given graph $G=(V,E)$, we construct a matrix $A\in\{-1,+1\}^{n\times d}$ where $n=p|E|$ and $d=p|V|$. Notice that $p=\text{poly}(|E|,|V|)$ is a parameter which will be determined later, and also p is a power of 2.

We divide the matrix A into $|E|\times|V|$ blocks, and each block has size $p\times p$. For $e\in[|E|]$, if the e^{th} edge has endpoints $i\in[|V|], j\in[|V|]$ and $i<j$, let all the $p\times p$ elements of (e,i) block of A be 1, all the $p\times p$ elements of (e,j) block of A be -1, and all the (e,l) block of A be $p\times p$ Hadamard matrix H for $l\neq i,j$.

Claim 17.9.6 (Lower bound of $\|A\|_{\infty\to 1}$, proof of Theorem 1 of [GV15])**.** *If there is a cut of G with cut size at least c,*

$$\|A\|_{\infty\to 1}\geq 2p^2c-|E||V|p^{3/2}.$$

Claim 17.9.7 (Upper bound of $\|A\|_{\infty\to 1}$, proof of Theorem 1 of [GV15])**.** *If the max cut of G has fewer than c edges,*

$$\|A\|_{\infty\to 1}\leq 2p^2(c-1)+|E||V|p^{3/2}.$$

Remark 17.9.1. In [GV15], they set p as a power of 2 and $p>|E|^2|V|^2$. This implies

$$\forall c\in[|E|], 2p^2(c-1)+|E||V|p^{3/2}<2p^2c-|E||V|p^{3/2}.$$

Therefore, according to Claim 17.9.6 and Claim 17.9.7, if we can know the precise value of $\|A\|_{\infty\to 1}$, we can decide whether G has a cut with cut size at least c^*.

For convenience, we use T^* to denote $\|A\|_{\infty\to 1}$ and use L^* to denote

$$\min_{x\in\mathbb{R}^n, y\in\mathbb{R}^d}\|A-x^\top y\|_1.$$

Also, we use L to denote a $(1+\frac{1}{nd})$ relative error approximation to L^*, which means:

$$L^* \leq L \leq (1+\frac{1}{nd})L^*.$$

We denote T as $nd - L$.

Proof of Theorem 17.9.4. Because $L^* \leq L \leq (1+\frac{1}{nd})L^*$, we have:

$$nd - L^* \geq nd - L \geq nd - (1+\frac{1}{nd})L^*.$$

Due to Lemma 17.9.1 and the definition of T, it has:

$$T^* \geq T \geq T^* - \frac{1}{nd}L^*.$$

Notice that A is a $\{-1,+1\}$ matrix, we have

$$L^* \leq \|A\|_1 \leq 2nd.$$

Thus,

$$T^* \geq T \geq T^* - 2.$$

It means

$$T + 2 \geq T^* \geq T.$$

According to Claim 17.9.11, if G has a cut with cut size at least c, we have:

$$T + 2 \geq T^* \geq 2p^2c - |E||V|p^{3/2}.$$

That is

$$T \geq 2p^2c - |E||V|p^{3/2} - 2.$$

According to Claim 17.9.12, if the max cut of G has fewer than c edges,

$$T \leq T^* \leq 2p^2(c-1) + |E||V|p^{3/2}.$$

Let p be a power of 2 and $p > |E|^3|V|^3$, we have

$$2p^2(c-1) + |E||V|p^{3/2} < 2p^2c - |E||V|p^{3/2} - 2.$$

Therefore, we can decide whether G has a cut with size at least c based on the value of T.

Thus, if we can compute $\widehat{x} \in \mathbb{R}^n, \widehat{y} \in \mathbb{R}^d$ s.t.

$$\|A - \widehat{x}^\top \widehat{y}\|_1 \leq (1 + \frac{1}{nd}) \min_{x \in \mathbb{R}^n, y \in \mathbb{R}^d} \|A - x^\top y\|_1,$$

in polynomial time, it means we can compute L and T in polynomial time, and we can solve MAX-CUT decision problem via the value of T, which leads to a contradiction. □

17.9.3 Using the ETH assumption

The goal of this section is to prove Theorem 17.9.9. We first introduce the definition of 3SAT and Exponential Time Hypothesis(ETH). For the details and background of 3SAT problem, we refer the readers to [AB09].

Definition 17.9.3 (3SAT problem)**.** Given an r variables and m clauses conjunctive normal form CNF formula with size of each clause at most 3, the goal is to decide whether there exists an assignment for the r boolean variables to make the CNF formula be satisfied.

Hypothesis 17.9.8 (Exponential Time Hypothesis (ETH) [IPZ98])**.** *There is a $\delta > 0$ such that* 3SAT *problem defined in Definition 17.9.3 cannot be solved in $O(2^{\delta r})$ running time.*

The main lower bound is stated as follows:

Theorem 17.9.9. *Unless* ETH *(see Hypothesis 17.9.8) fails, for arbitrarily small constant $\gamma > 0$, given some matrix $A \in \{-1, +1\}^{n \times d}$, there is no algorithm can compute $\widehat{x} \in \mathbb{R}^n, \widehat{y} \in \mathbb{R}^d$ s.t.*

$$\|A - \widehat{x}^\top \widehat{y}\|_1 \leq (1 + \frac{1}{\log^{1+\gamma} nd}) \min_{x \in \mathbb{R}^n, y \in \mathbb{R}^d} \|A - x^\top y\|_1,$$

in $(nd)^{O(1)}$ running time.

Before we prove our lower bound, we introduce the following theorem which is used in our proof.

Definition 17.9.4 (MAX-CUT decision problem)**.** Given a positive integer c^* and an unweighted graph $G = (V, E)$ where V is the set of vertices of G and E is the set of edges of G, the goal is to determine whether there is a cut of G has at least c^* edges.

Theorem 17.9.10 (Theorem 6.1 in [FLP15]). *There exist constants $a, b \in (0,1)$ and $a > b$, such that, for a given* MAX-CUT *(see Definition 17.9.4) instance graph $G = (E, V)$ which is an n-vertices 5-regular graph, if there is an algorithm in time $2^{o(n)}$ which can distinguish the following two cases:*

1. *At least one cut of the instance has at least $a|E|$ edges,*
2. *All cuts of the instance have at most $b|E|$ edges,*

then ETH *(see Hypothesis 17.9.8) fails.*

Proof of Theorem 17.9.9. We prove it by contradiction. We assume, for any given $A \in \{-1,+1\}^{n\times d}$, there is an algorithm can compute $\widehat{x} \in \mathbb{R}^n, \widehat{y} \in \mathbb{R}^d$ s.t.

$$\|A - \widehat{x}^\top \widehat{y}\|_1 \leq (1 + \frac{1}{W}) \min_{x\in\mathbb{R}^n, y\in\mathbb{R}^d} \|A - x^\top y\|_1,$$

in time $\mathrm{poly}(nd)$, where $W = \log^{1+\gamma} d$ for arbitrarily small constant $\gamma > 0$. Then, we show the following. There exist constants $a, b \in [0,1], a > b$, for a given MAX-CUT instance $G = (V, E)$ with $|E| = O(|V|)$, such that we can distinguish whether G has a cut with size at least $a|E|$ or all the cuts of G have size at most $b|E|$ in $2^{o(|V|)}$ time, which leads to a contradiction to Theorem 17.9.10.

Recall the reduction shown in [GV15] which is from MAX-CUT to computing $\|\cdot\|_{\infty\to 1}$ for $\{-1,+1\}$ matrices. We do similar things here: for a given graph $G = (V, E)$ where $|E| = O(|V|)$, we construct a matrix $A \in \{-1,+1\}^{n\times d}$ where $n = p|E|$ and $d = p|V|$. Notice that p is a parameter which will be determined later, and also p is a power of 2.

We divide the matrix A into $|E| \times |V|$ blocks, and each block has size $p \times p$. For $e \in [|E|]$, if the e^{th} edge has endpoints $i \in [|V|], j \in [|V|]$ and $i < j$, let all the $p \times p$ elements of (e, i) block of A be 1, all the $p \times p$ elements of (e, j) block of A be -1, and all the (e, l) block of A be $p \times p$ Hadamard matrix H for $l \neq i, j$.

We can construct the matrix in nd time, which is $p^2|E||V|$. We choose p to be the smallest number of power of 2 which is larger than $2^{\frac{2}{a-b}|V|^{1-\frac{\gamma}{10}}}$ for some $\gamma > 0$. Thus, the time for construction of the matrix A is $O(nd) = 2^{O(|V|^{1-\frac{\gamma}{10}})}$.

We will show, if we can compute a $(1 + 1/W)$-approximation to A, we can decide whether G has a cut with size at least $a|E|$ or has no cut with size larger than $b|E|$. For convenience, we use T^* to denote $\|A\|_{\infty \to 1}$ and use L^* to denote

$$\min_{x \in \mathbb{R}^n, y \in \mathbb{R}^d} \|A - x^\top y\|_1.$$

Also, we use L to denote a $(1 + \frac{1}{W})$ relative error approximation to L^*, which means:

$$L^* \leq L \leq (1 + \frac{1}{W})L^*.$$

We denote T as $nd - L$.

Because $L^* \leq L \leq (1 + \frac{1}{W})L^*$, we have:

$$nd - L^* \geq nd - L \geq nd - (1 + \frac{1}{W})L^*.$$

Due to Lemma 17.9.1 and the definition of T, it has:

$$T^* \geq T \geq T^* - \frac{1}{W}L^*.$$

Notice that A is a $\{-1, +1\}$ matrix, we have

$$L^* \leq \|A\|_1 \leq 2nd.$$

Thus,

$$T^* \geq T \geq T^* - 2nd/W.$$

It means

$$T + 2nd/W \geq T^* \geq T.$$

Claim 17.9.11 (Lower bound of $\|A\|_{\infty\to 1}$, proof of Theorem 1 of [GV15])**.** *If there is a cut of G with cut size at least c,*

$$\|A\|_{\infty\to 1} \geq 2p^2c - |E||V|p^{3/2}.$$

Claim 17.9.12 (Upper bound of $\|A\|_{\infty\to 1}$, proof of Theorem 1 of [GV15])**.** *If the max cut of G has fewer than c edges,*

$$\|A\|_{\infty\to 1} \leq 2p^2(c-1) + |E||V|p^{3/2}.$$

According to Claim 17.9.11, if G has a cut with cut size at least $a|E|$, we have:

$$T + 2nd/W \geq T^* \geq 2p^2a|E| - |E||V|p^{3/2}.$$

That is

$$T \geq 2p^2a|E| - |E||V|p^{3/2} - 2nd/W. \tag{17.34}$$

According to Claim 17.9.12, if the max cut of G has fewer than $b|E|$ edges,

$$T \leq T^* \leq 2p^2b|E| + |E||V|p^{3/2}. \tag{17.35}$$

Using these conditions $p \geq 2^{\frac{2}{a-b}|V|^{1-\frac{\gamma}{10}}}, d = p|V|, W \geq \log^{1+\gamma} d$, we can lower bound $|W|$ by $|V|$ up to some constant,

$$\begin{aligned}
W \geq\ & \log^{1+\gamma} d && \text{by } W \geq \log^{1+\gamma} d \\
=\ & \log^{1+\gamma}(p|V|) && \text{by } d = p|V| \\
=\ & (\log|V| + \log p)^{1+\gamma} \\
\geq\ & (\log|V| + \frac{2}{a-b}|V|^{1-\frac{\gamma}{10}})^{1+\gamma} && \text{by } p \geq 2^{\frac{2}{a-b}|V|^{1-\frac{\gamma}{10}}} \\
\geq\ & \frac{2}{a-b}|V|. && \text{by } (1-\gamma/10)(1+\gamma) > 1 \text{ for } \gamma \text{ small enough} \qquad (17.36)
\end{aligned}$$

Thus, we can upper bound $1/W$ in the following sense,

$$\frac{1}{W} \leq \frac{a-b}{2|V|} \leq \frac{a-b}{|V|} - p^{-\frac{1}{2}}, \qquad (17.37)$$

where the first inequality follows by Equation (17.36) and the second inequality follows by $p \geq 2^{\frac{2}{a-b}|V|^{1-\frac{\gamma}{10}}}$, γ is sufficient small, and $|V|$ is large enough.

Now, we can conclude,

$$\begin{aligned}
& \frac{1}{W} \leq \frac{a-b}{|V|} - p^{-\frac{1}{2}} \\
\iff\ & p^2|E||V|/W \leq (a-b)p^2|E| - |E||V|p^{\frac{3}{2}} \\
& \text{by multiplying } p^2|E||V| \text{ on both sides} \\
\iff\ & 2nd/W \leq 2(a-b)p^2|E| - 2|E||V|p^{\frac{3}{2}} \\
& \text{by multiplying 2 on both sides and } p^2|E||V| = nd \\
\iff\ & 2p^2b|E| + |E||V|p^{3/2} < 2p^2a|E| - |E||V|p^{3/2} - 2nd/W, \qquad (17.38) \\
& \text{by adding } 2p^2b|E| + |E||V|p^{3/2} - 2nd/W \text{ on both sides}
\end{aligned}$$

which implies that Equation (17.38) is equivalent to Equation (17.37). Notice that the LHS of (17.38) is exactly the RHS of (17.35) and the RHS of (17.38) is exactly the RHS of (17.34). Therefore, we can decide whether G has a cut with size larger than $a|E|$ or has no cut with size larger than $b|E|$.

Thus, if we can compute $\widehat{x} \in \mathbb{R}^n, \widehat{y} \in \mathbb{R}^d$ s.t.

$$\|A - \widehat{x}^\top \widehat{y}\|_1 \leq (1 + \frac{1}{\log^{1+\gamma} d}) \min_{x \in \mathbb{R}^n, y \in \mathbb{R}^d} \|A - x^\top y\|_1,$$

in poly(nd) time, which means we can compute L and T in poly(nd) time. Notice that $nd = p^2|E||V|, |E| = O(|V|), p \geq 2^{\frac{2}{a-b}|V|^{1-\frac{\gamma}{10}}}$, it means poly$(nd) = 2^{O(|V|^{1-\frac{\gamma}{10}})}$. Because we decide whether G has a cut with size larger than $a|E|$ or has no cut with size larger than $b|E|$ via the value of T, we can solve it in $2^{O(|V|^{1-\frac{\gamma}{10}})}$ time which leads to a contradiction to Theorem 17.9.10. □

17.9.4 Extension to the rank-k case

This section presents a way of reducing the rank-k case to the rank-1 case. Thus, we can obtain a lower bound for general $k \geq 1$ under ETH.

Theorem 17.9.13. *For any constants $c_1 > 0, c_2 > 0$ and $c_3 > 0$, and any constant $c_4 \geq 10(c_1 + c_2 + c_3 + 1)$, given any matrix $A \in \mathbb{R}^{n \times n}$ with absolute value of each entry bounded by n^{c_1}, we define a block diagonal matrix $\widetilde{A} \in \mathbb{R}^{(n+k-1)\times(n+k-1)}$ as*

$$\widetilde{A} = \begin{bmatrix} A & 0 & 0 & \cdots & 0 \\ 0 & B & 0 & \cdots & 0 \\ 0 & 0 & B & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & B \end{bmatrix},$$

where $B = n^{c_4}$. If $\widehat{A}$ is an ℓ_1-norm rank-k C-approximation solution to $\widetilde{A}$, i.e.,

$$\|\widehat{A} - \widetilde{A}\|_1 \leq C \cdot \min_{\mathrm{rank}-k\ \widehat{A}'} \|\widehat{A}' - \widetilde{A}\|_1,$$

where $C \in [1, n^{c_3}]$, then there must exist $j^* \in [n]$ such that

$$\min_{v \in \mathbb{R}^n} \|\widehat{A}_{j^*}^{[1:n]} v^\top - A\|_1 \leq C \cdot \min_{u,v \in \mathbb{R}^n} \|uv^\top - A\|_1 + 1/n^{c_2},$$

i.e., the first n coordinates of the column j^* of $\widehat{A}$ can give an ℓ_1-norm rank-1 C-approximation to A.

Proof. The first observation is that because we can use a rank-1 matrix to fit A and use a rank-$(k-1)$ matrix to fit other Bs, we have

$$\min_{\mathrm{rank}-k\ \widehat{A}'} \|\widehat{A}' - \widetilde{A}\|_1 \leq \min_{u,v \in \mathbb{R}^n} \|uv^\top - A\|_1 \leq \|A\|_1. \tag{17.39}$$

Claim 17.9.14. *Let $\widehat{A}$ denote the* rank-k *C-approximate solution to $\widetilde{A}$. Let $Z \in \mathbb{R}^{(n+k-1)\times(k-1)}$ denote the rightmost $k-1$ columns of $\widehat{A}$, then,* $\text{rank}(Z) = k-1$.

Proof. Consider the $(k-1)\times(k-1)$ submatrix $Z^{[n+1:n+k-1]}$ of Z. Each element on the diagonal of this submatrix should be at least $B - C\|A\|_1$, and each element not on the diagonal of the submatrix should be at most $C\|A\|_1$. Otherwise $\|\widehat{A} - \widetilde{A}\|_1 > C\|A\|_1$ which will lead to a contradiction. Since $B = n^{c_4}$ is sufficiently large, $Z^{[n+1:n+k-1]}$ is diagonally dominant. Thus $\text{rank}(Z) \geq \text{rank}(Z^{[n+1:n+k-1]}) = k-1$. Because Z only has $k-1$ columns, $\text{rank}(Z) = k-1$. □

Claim 17.9.15. $\forall x \in \mathbb{R}^{k-1}, i \in [n], \exists j \in \{n+1, n+2, \cdots, n+k-1\}$ *such that,*

$$\frac{|(Zx)_j|}{|(Zx)_i|} \geq \frac{B}{2(k-1)C\|A\|_1}.$$

Proof. Without loss of generality, we can let $\|x\|_1 = 1$. Thus, there exists j such that $|x_j| \geq \frac{1}{k-1}$. So we have

$$\begin{aligned}
|(Zx)_{n+j}| &= |\sum_{i=1}^{k-1} Z_{n+j,i}x_i| \\
&\geq |Z_{n+j,j}x_j| - \sum_{i\neq j} |Z_{n+j,i}x_i| \\
&\geq (B - C\|A\|_1)|x_j| - \sum_{i\neq j} |x_i|C\|A\|_1 \\
&\geq (B - C\|A\|_1)/(k-1) - C\|A\|_1 \\
&\geq \frac{B}{2(k-1)}.
\end{aligned}$$

The second inequality follows because $|Z_{n+j,j}| \geq B - C\|A\|_1$ and $\forall i \neq j, |Z_{n+j,i}| \leq C\|A\|_1$ (otherwise $\|\widehat{A} - \widetilde{A}\|_1 > C\|A\|_1$ which leads to a contradiction.) The third inequality follows

from $|x_j| > 1/(k-1)$ and $\|x\|_1 = 1$. The fourth inequality follows since B is large enough such that $\frac{B}{2(k-1)} \geq \frac{C\|A\|_1}{k-1} + C\|A\|_1$.

Now we consider any $q \in [n]$. We have

$$|(Zx)_q| = \sum_{i=1}^{k-1} |Z_{q,i}x_i| \leq \max_{i\in[k-1]} |Z_{q,i}| \cdot \sum_{i=1}^{k-1} |x_i| \leq C\|A\|_1.$$

The last inequality follows that $\|x\|_1 = 1$ and $\forall i \in [k-1], Z_{q,i} \leq C\|A\|_1$. Otherwise, $\|\widehat{A} - \widetilde{A}\|_1 > C\|A\|_1$ which will lead to a contradiction.

Look at $|(Zx)_{n+j}|/|(Zx)_q|$, it is greater than $\frac{B}{2(k-1)C\|A\|_1}$.

□

We now look at the submatrix $\widehat{A}^{[1:n]}_{[1:n]}$, we choose $i^*, j^* \in [n]$ such that

$$|\widehat{A}_{i^*,j^*}| \geq 1/n^{c_2-2}.$$

If there is no such (i^*, j^*), it means that we already found a good rank-1 approximation to A

$$\begin{aligned}
\|\mathbf{0} - A\|_1 &\leq \|\mathbf{0} - \widehat{A}^{[1:n]}_{[1:n]}\|_1 + \|\widehat{A}^{[1:n]}_{[1:n]} - A\|_1 \\
&\leq \|\mathbf{0} - \widehat{A}^{[1:n]}_{[1:n]}\|_1 + \|\widehat{A} - \widetilde{A}\|_1 \\
&\leq 1/n^{c_2} + \|\widehat{A} - \widetilde{A}\|_1 \\
&\leq 1/n^{c_2} + C \min_{\text{rank}-k\ \widehat{A}'} \|\widehat{A}' - \widetilde{A}\|_1 \\
&\leq 1/n^{c_2} + C \min_{u,v\in\mathbb{R}^n} \|uv^\top - A\|_1,
\end{aligned}$$

where $\mathbf{0}$ is an $n \times n$ all zeros matrix. The second inequality follows since $\widehat{A}^{[1:n]}_{[1:n]} - A$ is a submatrix of $\widehat{A} - \widetilde{A}$. The third inequality follows since each entry of $\widehat{A}^{[1:n]}_{[1:n]}$ should be

no greater than $1/n^{c_2-2}$ (otherwise, we can find (i^*, j^*)). The last inequality follows from equation 17.39.

Claim 17.9.16. *$\widehat{A}_{j^*}$ is not in the column span of Z, i.e.,*

$$\forall x \in \mathbb{R}^{k-1}, \widehat{A}_{j^*} \neq Zx.$$

Proof. If there is an x such that $\widehat{A}_{j^*} = Zx$, it means $(Zx)_{i^*} = \widehat{A}_{i^*,j^*} \geq 1/n^{c_2-2}$. Due to Claim 17.9.15, there must exist $i' \in \{n+1, n+2, \cdots, n+k-1\}$ such that $(Zx)_{i'} \geq \frac{B}{2(k-1)C\|A\|_1 n^{c_2-2}}$. Since B is sufficiently large, $\widehat{A}_{i',j^*} = (Zx)_{i'} > C\|A\|_1$ which implies that $\|\widehat{A} - \widetilde{A}\|_1 > C\|A\|_1$, and so leads to a contradiction. □

Due to Claim 17.9.14 and Claim 17.9.16, the dimension of the subspace spanned by $\widehat{A}_{j^*}$ and the column space of Z is k. Since $\widehat{A}$ has rank at most k, it means that each column of $\widehat{A}$ can be written as a linear combination of $\widehat{A}_{j^*}$ and the columns of Z.

Now consider the j^{th} column $\widehat{A}_j$ of $\widehat{A}$ for $j \in [n]$. We write it as

$$\widehat{A}_j = \alpha_j \cdot \widehat{A}_{j^*} + Zx^j.$$

Claim 17.9.17. $\forall j \in [n], \alpha_j \leq 2C\|A\|_1 n^{c_2+2}$.

Proof. Otherwise, suppose $\alpha_j > 2C\|A\|_1 n^{c_2+2}$. We have

$$\begin{aligned} |(Zx^j)_{i^*}| &\geq \alpha_j \cdot |\widehat{A}_{i^*,j^*}| - |\widehat{A}_{i^*,j}| \\ &\geq \alpha_j \cdot \frac{1}{n^{c_2-2}} - |\widehat{A}_{i^*,j}| \\ &\geq \frac{1}{2}\alpha_j \cdot \frac{1}{n^{c_2-2}}. \end{aligned}$$

The second inequality follows from $|\widehat{A}_{i^*,j^*}| \geq 1/n^{c2-2}$. The third inequality follows from $|\widehat{A}_{i^*,j}| \leq \|A\|_1$ and $\frac{1}{2}\alpha_j \cdot \frac{1}{n^{c_2-2}} \geq C\|A\|_1 \geq \|A\|_1$.

Due to Claim 17.9.15, there exists $i \in \{n+1, n+2, \cdots, n+k-1\}$ such that $|(Zx^j)_i| \geq \frac{B}{2(k-1)C\|A\|_1} \cdot \frac{1}{2}\alpha_j \cdot \frac{1}{n^{c_2-2}}$. For sufficiently large B, we can have $|(Zx^j)_i| \geq \alpha_j B^{1/2}$. Then we look at

$$\begin{aligned}
|\widehat{A}_{i,j}| &\geq |(Zx^j)_i| - \alpha_j |\widehat{A}_{i,j^*}| \\
&\geq \alpha_j (B^{1/2} - C\|A\|_1) \\
&\geq \alpha_j \frac{1}{2} B^{1/2}.
\end{aligned}$$

The second inequality follows by $|\widehat{A}_{i,j^*}| \leq C\|A\|_1$, otherwise $\|\widehat{A} - \widetilde{A}\|_1 > C\|A\|_1$ which will lead to a contradiction. The third inequality follows that B is sufficient large that $\frac{1}{2}B^{1/2} > C\|A\|_1$.

Since $|\widehat{A}_{i,j}| \geq \alpha_j \frac{1}{2} B^{1/2} > C\|A\|_1$, it contradicts to the fact $\|\widehat{A} - \widetilde{A}\|_1 \leq C\|A\|_1$.

Therefore, $\forall j \in [n], \alpha_j \leq 2C\|A\|_1 n^{c_2+2}$. □

Claim 17.9.18. $\forall j \in [n], i \in \{n+1, n+2, \cdots, n+k-1\}, |(Zx^j)_i| \leq 4C^2\|A\|_1^2 n^{c_2+2}$

Proof. Consider $j \in [n], i \in \{n+1, n+2, \cdots, n+k-1\}$, we have

$$\begin{aligned}
|(Zx^j)_i| &\leq |\widehat{A}_{i,j}| + \alpha_j |\widehat{A}_{i,j^*}| \\
&\leq C\|A\|_1 + \alpha_j \cdot C\|A\|_1 \\
&\leq 4C^2\|A\|_1^2 n^{c_2+2}.
\end{aligned}$$

The second inequality follows by $|\widehat{A}_{i,j}| \leq C\|A\|_1$ and $|\widehat{A}_{i,j^*}| \leq C\|A\|_1$, otherwise the $\|\widehat{A} - \widetilde{A}\|_1$ will be too large and leads to a contradiction. The third inequality is due to $\alpha_j + 1 \leq 4C\|A\|_1 n^{c_2+2}$ via Claim 17.9.17. □

Claim 17.9.19. $\forall j \in [n], \|\widehat{A}_j^{[1:n]} - \alpha_j \cdot \widehat{A}_{j^*}^{1:n}\|_\infty \le 1/n^{c_2-2}$

Proof. Due to Claim 17.9.18 and Claim 17.9.15, $\forall i, j \in [n]$, we have

$$|(Zx^j)_i| \le \frac{4C^2\|A\|_1^2 n^{c_2+2}}{B/(2(k-1)C\|A\|_1)} \le 1/B^{1/2}.$$

The second inequality follows for a large enough B.

Therefore, $\forall i, j \in [n]$,

$$|\widehat{A}_{i,j} - \alpha_j \cdot \widehat{A}_{i,j^*}| \le |(Zx^j)_i| \le 1/B^{1/2} \le 1/n^{c_2-2}.$$

The last inequality follows since B is large enough.

□

Now, let us show that $\widehat{A}_{j^*}^{1:n}$ can provide a good rank-1 approximation to A:

$$\begin{aligned}
\|\widehat{A}_{j^*}^{1:n}\alpha^\top - A\|_1 &\le \|\widehat{A}_{j^*}^{1:n}\alpha^\top - \widehat{A}_{[1:n]}^{[1:n]}\|_1 + \|\widehat{A}_{[1:n]}^{[1:n]} - A\|_1 \\
&= \sum_{j=1}^{n} \|\alpha_j \widehat{A}_{j^*} - \widehat{A}_j^{[1:n]}\|_1 + \|\widehat{A}_{[1:n]}^{[1:n]} - A\|_1 \\
&\le n^2 \cdot 1/n^{c_2-2} + \|\widehat{A}_{[1:n]}^{[1:n]} - A\|_1 \\
&\le 1/n^{c_2} + \|\widehat{A} - \widetilde{A}\|_1 \\
&\le 1/n^{c_2} + C \min_{u,v\in\mathbb{R}^n} \|uv^\top - A\|_1.
\end{aligned}$$

The first inequality follows by triangle inequality. The first equality is due to the linearity of ℓ_1 norm. The second inequality is due to Claim 17.9.19. The third inequality follows since $\widehat{A}_{[1:n]}^{[1:n]} - A$ is a submatrix of $\widehat{A} - \widetilde{A}$. The fourth inequality is due to the equation 17.39.

□

17.10 Experiments and Discussions

In this section, we provide some counterexamples for the other heuristic algorithms such that, for those examples, the heuristic algorithms can output a solution with a very "bad" approximation ratio, i.e., n^c, where $c > 0$ and the input matrix has size $n \times n$. We not only observe that heuristic algorithms sometimes have very bad performance in practice, but also give a proof in theory.

17.10.1 Setup

We provide some details of our experimental setup. We obtained the R package of [KK05, Kwa08, BDB13] from https://cran.r-project.org/web/packages/pcaL1/index.html. We also implemented our algorithm and the r1-pca algorithm [DZHZ06] using the R language. The version of the R language is 3.0.2. We ran experiments on a machine with Intel X5550@2.67GHz CPU and 24G memory. The operating system of that machine is Linux Ubuntu 14.04.5 LTS. All the experiments were done in single-threaded mode.

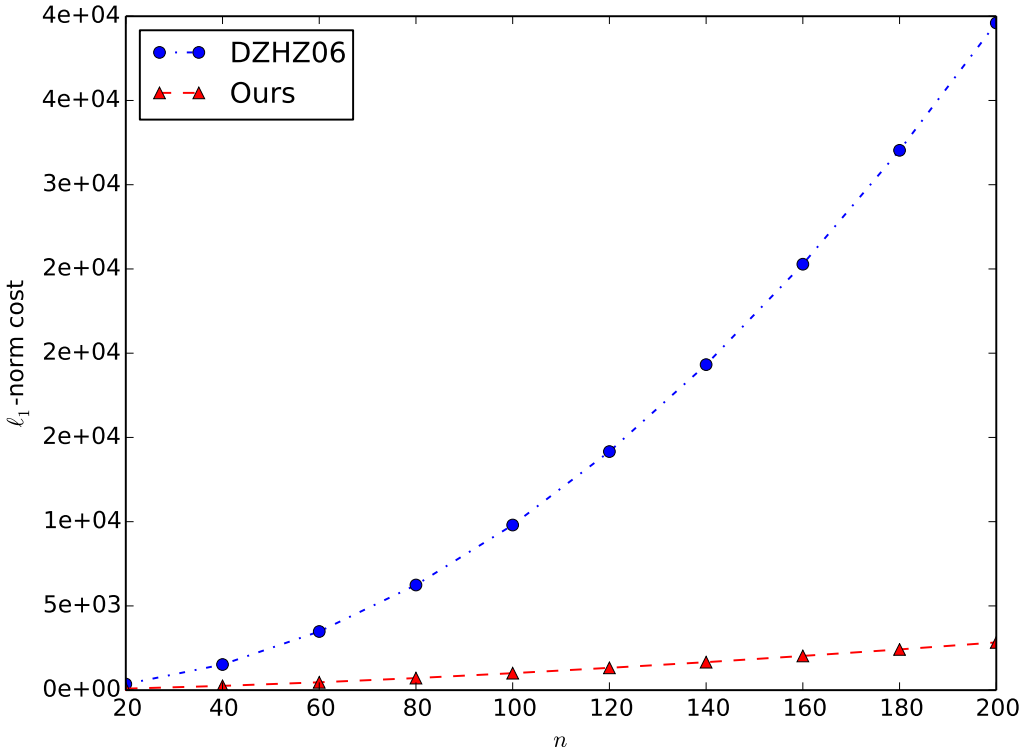


Figure 17.1: The x-axis is n where $A \in \mathbb{R}^{n \times n}$, and the y-axis is $\|A' - A\|_1$ where $\text{rank}(A') = k$. This figure shows the performance of both our algorithm and [DZHZ06] on input matrix A defined as Equation (17.40).

17.10.2 Counterexample for [DZHZ06]

The goal is to find a rank $k = 1$ approximation for matrix A. For any $\epsilon \in [0, 0.5)$, we define $A \in \mathbb{R}^{n \times n}$ as

$$A = \begin{bmatrix} n^{1.5+\epsilon} & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & B \end{bmatrix}, \tag{17.40}$$

where $B \in \mathbb{R}^{(n-1) \times (n-1)}$ is all 1s matrix. It is immediate that the optimal cost is at most $n^{1.5+\epsilon}$. However, using the algorithm in [DZHZ06], the cost is at least $\Omega(n^2)$. Thus, we

can conclude, using algorithm [DZHZ06] to solve ℓ_1 low rank approximation problem on A cannot achieve an approximation ratio better than $n^{0.5-\epsilon}$.

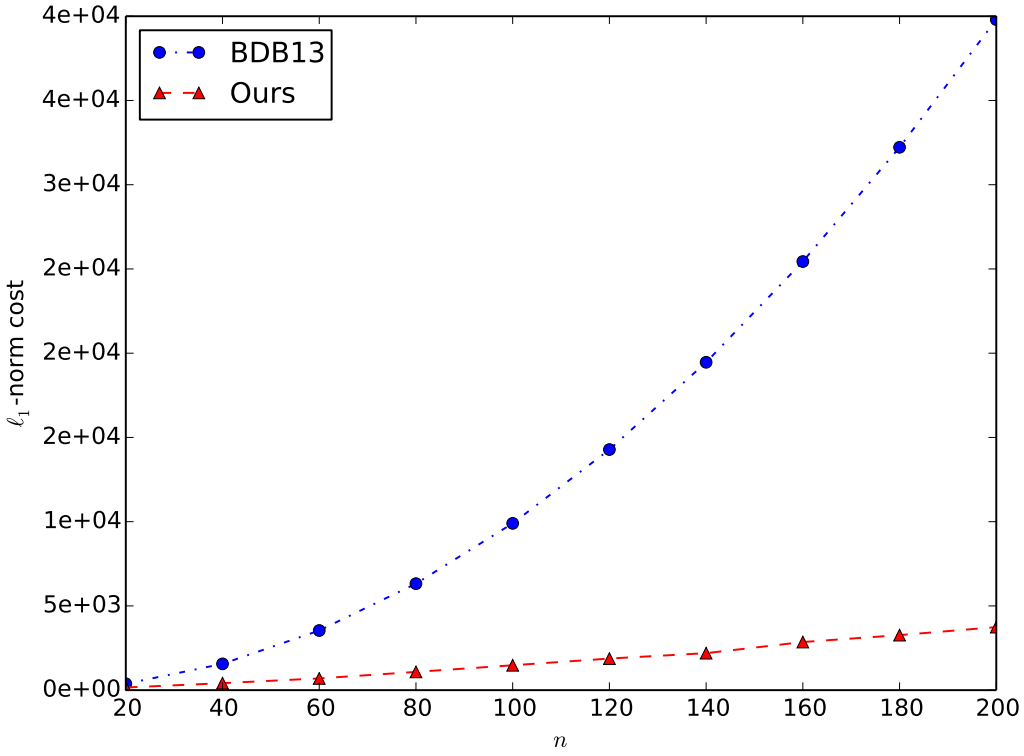


Figure 17.2: The x-axis is n where $A \in \mathbb{R}^{n \times n}$, and the y-axis is $\|A' - A\|_1$ where $\text{rank}(A') = k$. This figure shows the performance of both our algorithm and [BDB13] on input matrix A defined as Equation (17.41).

17.10.3 Counterexample for [BDB13]

The goal is to find a rank $k = 1$ approximation for matrix A. The input matrix $A \in \mathbb{R}^{d \times d}$ for algorithm [BDB13] is defined to be,

$$A = \begin{bmatrix} n^{1.5} & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & B \end{bmatrix}, \tag{17.41}$$

where $B \in \mathbb{R}^{(n-1) \times (n-1)}$ is an all 1s matrix. It is immediate that the optimal cost is at most $n^{1.5}$. Now, let us look at the procedure of [BDB13]. Basically, the algorithm described

in [BDB13] is that they first find a rank $n-1$ approximation via a best ℓ_1-fit hyperplane algorithm, then they rotate it based on the right singular vectors of the rank $n-1$ approximation matrix, and next they recursively do the same thing for the rotated matrix which has only $n-1$ columns.

When running their algorithm on A, they will fit an arbitrary column except the first column of A. Without loss of generality, it just fits the last column of A. After the rotation, the matrix will be an $n \times (n-1)$ matrix:

$$\begin{bmatrix} n^{1.5} & 0 & 0 & \cdots & 0 \\ 0 & \sqrt{n-2} & 0 & \cdots & 0 \\ 0 & \sqrt{n-2} & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & \sqrt{n-2} & 0 & \cdots & 0 \end{bmatrix}.$$

Then, after the t^{th} iteration for $t < (n-1)$, they will get an $n \times (n-t)$ matrix:

$$\begin{bmatrix} n^{1.5} & 0 & 0 & \cdots & 0 \\ 0 & \sqrt{n-2} & 0 & \cdots & 0 \\ 0 & \sqrt{n-2} & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & \sqrt{n-2} & 0 & \cdots & 0 \end{bmatrix}.$$

This means that their algorithm will run on an $n \times 2$ matrix:

$$\begin{bmatrix} n^{1.5} & 0 \\ 0 & \sqrt{n-2} \\ 0 & \sqrt{n-2} \\ \cdots & \cdots \\ 0 & \sqrt{n-2} \end{bmatrix},$$

in the last iteration. Notice that $n \times \sqrt{n-2} < n^{1.5}$. This means that their algorithm will fit the first column which implies that their algorithm will output a rank-1 approximation to A by just fitting the first column of A. But the cost of this rank-1 solution is $(n-1)^2$. Since the optimal cost is at most $n^{1.5}$, their algorithm cannot achieve an approximation ratio better than $n^{0.5}$.

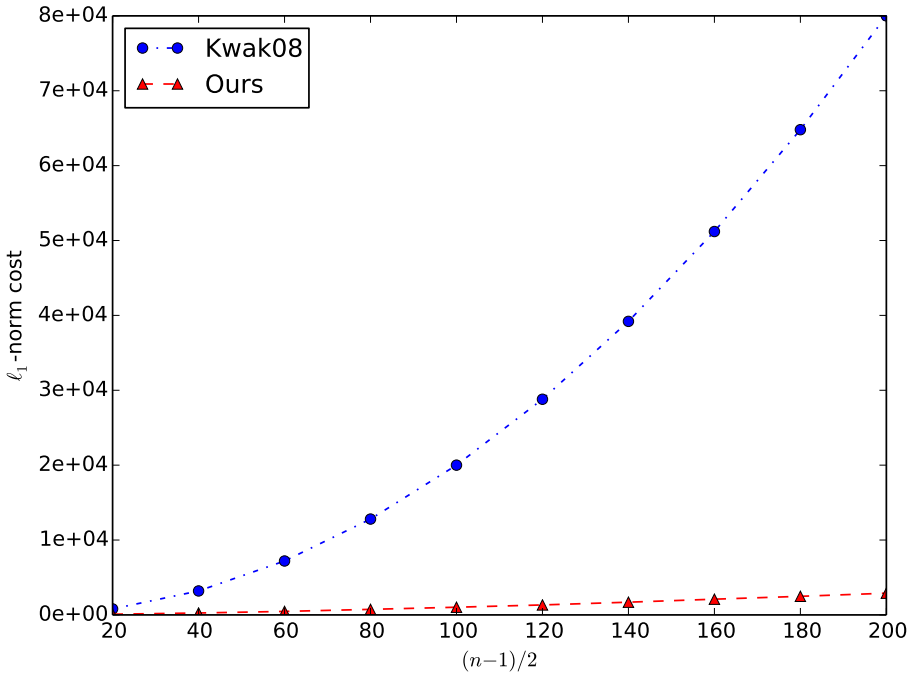


Figure 17.3: The x-axis is n where $A \in \mathbb{R}^{n \times n}$, and the y-axis is $\|A' - A\|_1$ where $\text{rank}(A') = k$. This figure shows the performance of both our algorithm and [Kwa08] on input matrix A defined as Equation (17.42). Algorithm [Kwa08] has two ways of initialization, which have similar performance on matrix A.

17.10.4 Counterexample for [Kwa08]

We show that the algorithm [Kwa08] cannot achieve an approximation ratio better than $\Theta(n)$ on the matrix $A \in \mathbb{R}^{(2n+1)\times(2n+1)}$ defined as,

$$A = \begin{bmatrix} n^{1.5} & 0 & 0 \\ 0 & B & 0 \\ 0 & 0 & B \end{bmatrix}, \tag{17.42}$$

where B is an $n \times n$ matrix that contains all 1s. We consider the rank-2 approximation problem for this input matrix A. The optimal cost is at most $n^{1.5}$.

We run their algorithm. Let x_0 denote the initial random unit vector. Consider the sign vector $s \in \{\pm 1\}^{2n+1}$ where each entry is the sign of the inner product between x_0 and each column of A. There are only three possibilities,

$$s = \begin{cases} (1, \{1\}^n, \{1\}^n) \\ (1, \{1\}^n, \{-1\}^n) \\ (1, \{-1\}^n, \{1\}^n) \end{cases}.$$

Case I, $s = (1, \{1\}^n, \{1\}^n)$. Define $\widehat{u} = \sum_{i=1}^{2n+1} s_i \cdot A_i = (n^{1.5}, n, \cdots, n)$. Let $u = \widehat{u}/\|u\|_2 = \widehat{u}/(\sqrt{3}n^{1.5}) = (1/\sqrt{3}, 1/\sqrt{3n}, \cdots, 1/\sqrt{3n})$. Define matrix D to be $A - uu^\top A$. Then, we can compute D,

$$\begin{aligned} D &= A - uu^\top A \\ &= A - \begin{bmatrix} 1/3 & \frac{1}{3\sqrt{n}}\mathbf{1}^\top & \frac{1}{3\sqrt{n}}\mathbf{1}^\top \\ \frac{1}{3\sqrt{n}}\mathbf{1} & \frac{1}{3n}B & \frac{1}{3n}B \\ \frac{1}{3\sqrt{n}}\mathbf{1} & \frac{1}{3n}B & \frac{1}{3n}B \end{bmatrix} A \\ &= \begin{bmatrix} n^{1.5} & 0 & 0 \\ 0 & B & 0 \\ 0 & 0 & B \end{bmatrix} - \begin{bmatrix} \frac{n^{1.5}}{3} & \frac{\sqrt{n}}{3}\mathbf{1}^\top & \frac{\sqrt{n}}{3}\mathbf{1}^\top \\ \frac{n}{3}\mathbf{1} & \frac{1}{3}B & \frac{1}{3}B \\ \frac{n}{3}\mathbf{1} & \frac{1}{3}B & \frac{1}{3}B \end{bmatrix} \\ &= \begin{bmatrix} 2n^{1.5}/3 & -\frac{\sqrt{n}}{3}\mathbf{1}^\top & -\frac{\sqrt{n}}{3}\mathbf{1}^\top \\ -\frac{n}{3}\mathbf{1} & \frac{2}{3}B & -\frac{1}{3}B \\ -\frac{n}{3}\mathbf{1} & -\frac{1}{3}B & \frac{2}{3}B \end{bmatrix}. \end{aligned}$$

Now we need to take the linear combination of columns $D = A - uu^\top A$. Let w denote another sign vector $\{-1, +1\}^{2n+1}$. Then let v denote the basis vector, $v = \sum_{i=1}^{2n+1} D_i$. There are three possibilities, Case I(a), if $w = (1, \{1\}^n, \{1\}^n)$, then v is the all 0 vector. Using vector v to interpolate each column of D, the cost we obtain is at least $2n^2$. Case I(b), if $w = (1, \{1\}^n, \{-1\}^n)$, then $v = (2n^{1.5}/3, \{2/3\}^n, \{-4/3\}^n)$. We also obtain at least $2n^2$ cost if we use that v to interpolate each column of D. Case I(c), if $w = (1, \{-1\}^n, \{-1\}^n)$, then $v = (0, \{-2/3\}^n, \{-2/3\}^n)$. The cost is also at least $2n^2$.

Case II, $s = (1, \{1\}^n, \{-1\}^n)$. Define $\mathbf{1}$ to be a length n all 1s column vector. Define $\widehat{u} = \sum_{i=1}^{2n+1} s_i \cdot A_i = (n^{1.5}, \{n\}^n, \{-n\}^n)$. Let

$$u = \widehat{u}/\|u\|_2 = \widehat{u}/(\sqrt{3}n^{1.5}) = (1/\sqrt{3}, \{1/\sqrt{3n}\}^n, \{-1/\sqrt{3n}\}^n).$$

Define $(2n+1) \times (2n+1)$ matrix D to be $A - uu^\top A$. Then, we can compute D,

$$\begin{aligned}
D &= A - uu^\top A \\
&= A - \begin{bmatrix} 1/3 & \frac{1}{3\sqrt{n}}\mathbf{1}^\top & -\frac{1}{3\sqrt{n}}\mathbf{1}^\top \\ \frac{1}{3\sqrt{n}}\mathbf{1} & \frac{1}{3n}B & -\frac{1}{3n}B \\ -\frac{1}{3\sqrt{n}}\mathbf{1} & -\frac{1}{3n}B & \frac{1}{3n}B \end{bmatrix} A \\
&= \begin{bmatrix} n^{1.5} & 0 & 0 \\ 0 & B & 0 \\ 0 & 0 & B \end{bmatrix} - \begin{bmatrix} n^{1.5}/3 & \frac{\sqrt{n}}{3}\mathbf{1}^\top & -\frac{\sqrt{n}}{3}\mathbf{1}^\top \\ \frac{n}{3}\mathbf{1} & \frac{1}{3}B & -\frac{1}{3}B \\ -\frac{n}{3}\mathbf{1} & -\frac{1}{3}B & \frac{1}{3}B \end{bmatrix} \\
&= \begin{bmatrix} 2n^{1.5}/3 & -\frac{\sqrt{n}}{3}\mathbf{1}^\top & \frac{\sqrt{n}}{3}\mathbf{1}^\top \\ -\frac{n}{3}\mathbf{1} & \frac{2}{3}B & \frac{1}{3}B \\ \frac{n}{3}\mathbf{1} & \frac{1}{3}B & \frac{2}{3}B \end{bmatrix}.
\end{aligned}$$

Similarly to the previous case, we can also discuss three cases.

Case III, $s = (1, \{1\}^n, \{-1\}^n)$. Define $\mathbf{1}$ to be a length n all 1s column vector. Define $\widehat{u} = \sum_{i=1}^{2n+1} s_i \cdot A_i = (n^{1.5}, \{-n\}^n, \{-n\}^n)$. Let $u = \widehat{u}/\|u\|_2 = \widehat{u}/(\sqrt{3}n^{1.5}) = (1/\sqrt{3}, \{-1/\sqrt{3n}\}^n, \{-1/\sqrt{3n}\}^n)$. Define matrix D to be $A - uu^\top A$. Then, we can compute D,

$$
\begin{aligned}
D &= A - uu^\top A \\
&= A - \begin{bmatrix} 1/3 & -\frac{1}{3\sqrt{n}})\mathbf{1}^\top & -\frac{1}{3\sqrt{n}}\mathbf{1}^\top \\ -\frac{1}{3\sqrt{n}}\mathbf{1} & \frac{1}{3n}B & \frac{1}{3n}B \\ -\frac{1}{3\sqrt{n}}\mathbf{1} & \frac{1}{3n}B & \frac{1}{3n}B \end{bmatrix} A \\
&= \begin{bmatrix} n^{1.5} & 0 & 0 \\ 0 & B & 0 \\ 0 & 0 & B \end{bmatrix} - \begin{bmatrix} n^{1.5}/3 & -\frac{\sqrt{n}}{3}\mathbf{1}^\top & -\frac{\sqrt{n}}{3}\mathbf{1}^\top \\ -\frac{n}{3}\mathbf{1} & \frac{1}{3}B & \frac{1}{3}B \\ -\frac{n}{3}\mathbf{1} & \frac{1}{3}B & \frac{1}{3}B \end{bmatrix} \\
&= \begin{bmatrix} 2n^{1.5}/3 & \frac{\sqrt{n}}{3}\mathbf{1}^\top & \frac{\sqrt{n}}{3}\mathbf{1}^\top \\ \frac{n}{3}\mathbf{1} & \frac{2}{3}B & \frac{1}{3}B \\ \frac{n}{3}\mathbf{1} & \frac{1}{3}B & \frac{2}{3}B \end{bmatrix}.
\end{aligned}
$$

Similarly to the previous case, we can also discuss three cases.

17.10.5 Counterexample for [KK05]

We show that there exist matrices such that the algorithm of [KK05] cannot achieve an approximation ratio better than $\Theta(n)$. Their algorithm has two different ways of initialization. We provide counterexamples for each of the initialization separately.

Random vector initialization We provide a counterexample matrix $A \in \mathbb{R}^{n \times n}$ defined as,

$$A = \begin{bmatrix} n^c & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 0 \end{bmatrix} + I, \tag{17.43}$$

where $c \geq 2$. Consider the rank-1 approximation problem for this matrix A. The optimal cost is at most $n - 1$.

Run their algorithm. The starting vectors are $u(0) \sim N(0, I)$ and $v(0) \sim N(0, 1)$. We define two properties for a given vector $y \in \mathbb{R}^n$. Property I is for all $i \in [n]$, $|y_i| \leq n/8$, and Property II is there exist half of the i such that $|y_i| \geq 1/2$. We can show that with probability $1 - 2^{-\Omega(n)}$, both $u(0)$ and $v(0)$ satisfy Property I and II. After 1 iteration, we can show that $u(1)_1 = v(1)_1 = 0$

Now let us use column vector $u(0) \in \mathbb{R}^n$ to interpolate the first column of matrix A. We simplify $u(0)$ to be u. Let u_i denote the i-th coordinate of vector u, $\forall i \in [n]$. Let A_1 denote the first column of matrix A. We define $\alpha = v(1)_1 = \arg\min_\alpha \|\alpha u(0) - A_1\|_1$. For any scalar α, the cost we pay on the first column of matrix A is,

$$|\alpha \cdot u_1 - n^c| + \sum_{i=2}^{n} |\alpha u_i| \geq |\alpha \cdot u_1 - n^c| + \frac{n}{2} |\alpha \cdot \frac{1}{2}| \geq |\alpha \cdot u_1 - n^c| + \frac{n}{4} |\alpha|. \tag{17.44}$$

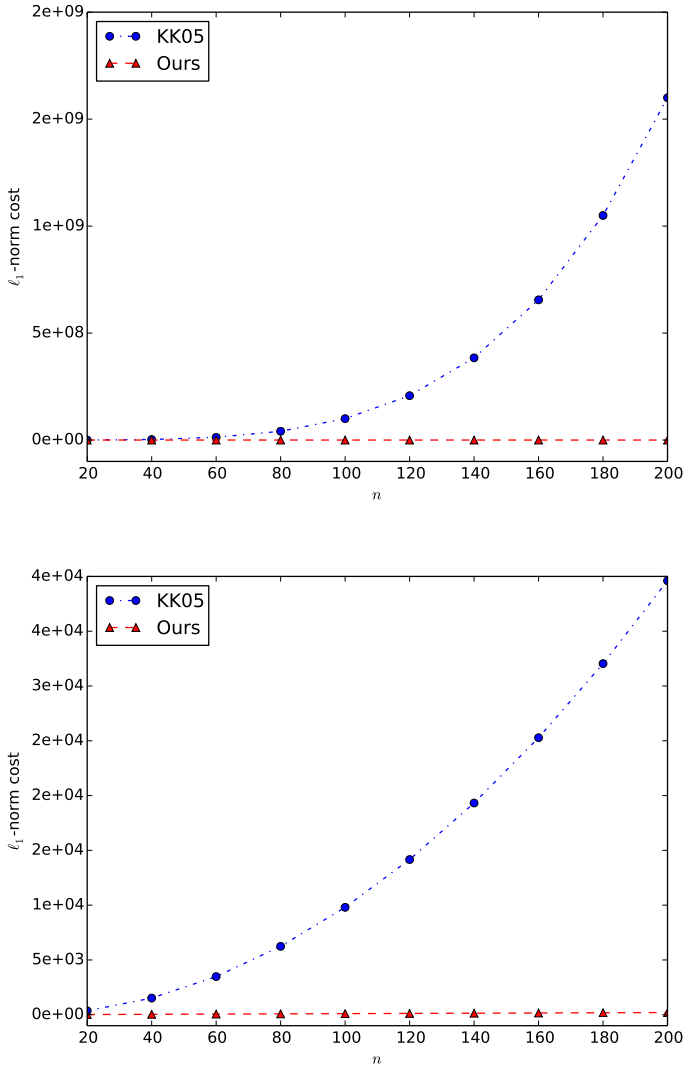


Figure 17.4: The x-axis is n where $A \in \mathbb{R}^{n \times n}$, and the y-axis is $\|A' - A\|_1$ where $\text{rank}(A') = k$. Algorithm [KK05] has two ways of initialization. The left figure shows the performance of both our algorithm and [KK05] with random vector initialization on input matrix A defined as Equation (17.43). The right figure shows the performance of both our algorithm and [KK05] with top singular vector initialization on input matrix A defined as Equation (17.45).

Notice that $c \geq 2$. Then $|\alpha \cdot u_1 - n^c| + \frac{n}{4}|\alpha| \geq |\alpha \cdot \frac{n}{8} - n^c| + \frac{n}{4}|\alpha|$. If $\alpha \leq 0$, then the cost is minimized when $\alpha = 0$. If $\alpha \in [0, 8n^{c-1}]$, the cost is minimized when $\alpha = 0$. If $\alpha \geq 8n^{c-1}$, the cost is minimized when $\alpha = 8n^{c-1}$. Putting it all together, to achieve the minimum cost, there is only one choice for α, which is $\alpha = 0$. The optimal cost is at least n^c.

Then after T iterations(for any $T \geq 1$), $u(T)_1 = v(T)_1 = 0$. Thus, we always pay at least n^c cost on the first entry.

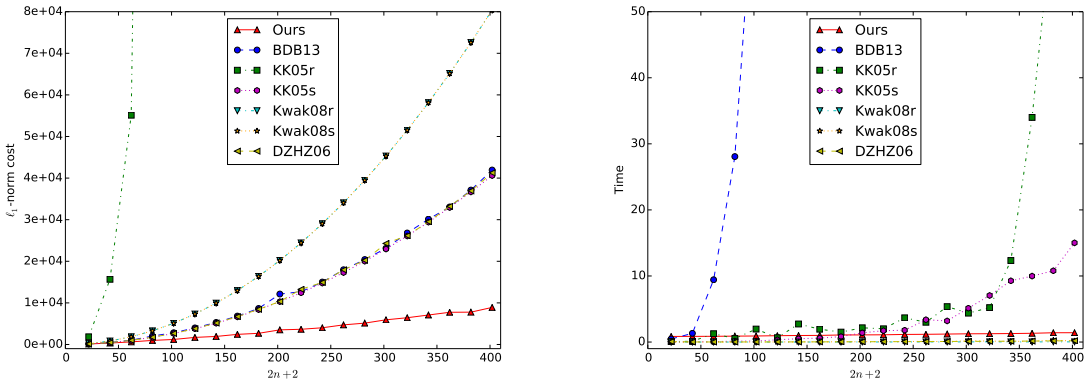
Therefore, their algorithm cannot achieve any approximation ratio better than n^{c-1}. Because $c \geq 2$, we complete the proof.

Top singular vector initialization The counterexample input matrix $A \in \mathbb{R}^{n \times n}$ is defined as,

$$A = \begin{bmatrix} n & 0 \\ 0 & B \end{bmatrix}, \tag{17.45}$$

where matrix $B \in \mathbb{R}^{(n-1)\times(n-1)}$ contains all 1s. Consider the rank-1 approximation problem for this matrix A. The optimal cost is at most n. Run their algorithm. The starting vectors $u(0)$ and $v(0)$ will be set to $(1, 0, \cdots, 0) \in \mathbb{R}^n$. After T iterations(for any $T > 0$), the support of $u(T)$(resp. $v(T)$) is the same as $u(0)$(resp. $v(T)$). Thus, the cost is at least $\|B\|_1 = (n-1)^2$. Therefore, we can conclude that their algorithm cannot achieve any approximation ratio better than $(n-1)^2/n = \Theta(n)$.

17.10.6 Counterexample for all



(a) (b)

Figure 17.5: Let A be $(2n+2) \times (2n+2)$ input matirx. (a) shows the performance of all the algorithms when the matrix dimension is growing. The x-axis is n, and the y-axis is $\|A' - A\|_1$ where A' is the rank-3 solution output by all the heuristic algorithms and also ours. The ℓ_1 residual cost of all the other algorithms is growing much faster than ours, which is consistent with our theoretical results. (b) shows the running time (in seconds) of all the algorithms when the matrix dimension n is growing. The x-axis is n and the y-axis is time (seconds). The running time of some of the algorithms is longer than 3 seconds. For most of the algorithms (including ours), the running time is always less than 3 seconds.

For any $\epsilon \in (0, 0.5)$ and $\gamma > 0$, we construct the input matrix $A \in \mathbb{R}^{(2n+2)\times(2n+2)}$ as follows

$$A = \begin{bmatrix} n^{2+\gamma} & 0 & 0 & 0 \\ 0 & n^{1.5+\epsilon} & 0 & 0 \\ 0 & 0 & B & 0 \\ 0 & 0 & 0 & B \end{bmatrix},$$

where B is $n \times n$ all 1s matrix. We want to find a rank $k = 3$ solution for A. Then any of those four heuristic algorithms [KK05, DZHZ06, Kwa08, BDB13] is not able to achieve better than $n^{\min(\gamma, 0.5-\epsilon)}$ approximation ratio. We present our main experimental results in Figure 17.5.

Both [KK05] and [Kwa08] have two different ways of initialization. In Figure 17.5 we use KK05r(resp. Kwak08r) to denote the way that uses random vector as initialization, and use KK05s(resp. Kwak08s) to denote the way that uses top singular vector as initialization. Figure 17.5(a) shows the performance of all the algorithms and Figure 17.5(b) presents the running time. The ℓ_1 residual cost of all the other algorithm is growing much faster than our algorithm. Most of the algorithms (including ours) are pretty efficient, i.e., the running time is always below 3 seconds. The running time of [BDB13, KK05] is increasing very fast when the matrix dimension n is growing.

In Figure 17.5(a), the cost of KK05r at $\{82, \cdots, 142\}$ is in $[10^5, 10^6]$, at $\{162, \cdots, 302\}$ is in $[10^6, 10^7]$, and at $\{322, \cdots, 402\}$ is in $[10^7, 10^8]$. In Figure 17.5(b), the time of KK05r at $\{382, 482\}$ is 64s and 160s. The running time of BDB13 at $\{82, \cdots, 222\}$ is between 1 minute and 1 hour. The running time of BDB13 at $\{242, \cdots, 322\}$ is between 1 hour and 20 hours. The running time of BDB13 at $\{342, \cdots, 402\}$ is more than 20 hours.

17.10.7 Discussion for Robust PCA [CLMW11]

A popular method is robust PCA [CLMW11], which given a matrix A, tries to find a matrix L for which $\lambda\|A-L\|_1+\|L\|_*$ is minimized, where $\lambda>0$ is a tuning parameter and $\|L\|_*$ is the nuclear norm of L. This is a convex program, but it need not return a low rank matrix L with relative error. As a simple example, suppose $\lambda=1$, and the $n\times n$ matrix A is a block-diagonal matrix of rank k and $n=\frac{k}{2}(b+1)$. Further, the first $k/2$ blocks are $b\times b$ matrices of all 1s, while the next $k/2$ blocks are just a single value b on the diagonal.

Then the solution to the above problem may return L to be the first $k/2$ blocks of A. The total cost of $\lambda\|A-L\|_1+\|L\|_*$is $(k/2)b+(k/2)b=kb$.

Also, the solution to the above problem may return L to be A, which has cost $0+\|A\|_*=kb$. Because this solution has the same cost, it means that their algorithm might output a rank-k solution, and also might output a rank-$k/2$ solution.

Therefore, the relative error of the output matrix may be arbitrarily bad for ℓ_1-low rank approximation.

We also consider the following example. Suppose $\lambda=1/\sqrt{n}$, and let $n\times n$ matrix A denote the Hadamard matrix H_n. Recall that the Hadamard matrix H_p of size $p\times p$ is defined recursively : $\begin{bmatrix}H_{p/2} & H_{p/2}\\ H_{p/2} & -H_{p/2}\end{bmatrix}$ with $H_2=\begin{bmatrix}+1 & +1\\ +1 & -1\end{bmatrix}$. Notice that every singular values of A is $\sqrt{n}$. We consider the objective function $\lambda\|A-L\|_1+\|L\|_*$.

Then the solution to the above problem may return L to be the first $n/2$ rows of A. The total cost of $\lambda\|A-L\|_1+\|L\|_*$ is $(1/\sqrt{n})n^2/2+(n/2)\sqrt{n}=n^{1.5}$. Also, the solution to the above problem may return L to be A, which has cost $0+\|A\|_*=n\sqrt{n}=n^{1.5}$. For any i, if the solution takes i rows of A, the cost is $(1/\sqrt{n})(n-i)n+i\sqrt{n}=n^{1.5}$. Because this

solution has the same cost, it means that their algorithm might output a rank-n solution, and also might output a rank-$n/2$ solution. Therefore, the relative error of the output matrix may be arbitrarily bad for ℓ_1-low rank approximation.

17.11 Limited Independent Cauchy Random Variables

This section presents the fundamental lemmas with limited independent Cauchy variables, which will be used in Section 17.12 and 17.13. In Section 17.11.1, we provide some notation, definitions and tools from previous work. Section 17.11.2 includes the main result.

17.11.1 Notations and tools

For a function $f : \mathbb{R} \to \mathbb{R}$ and nonnegative integer ℓ, $f^{(\ell)}$ denotes the ℓth derivative of f, with $f^{(0)} = f$. We also often use $x \approx_\epsilon y$ to state that $|x - y| = O(\epsilon)$. We use $I_{[a,b]}$ to denote the indicator function of the interval $[a, b]$.

To optimize the communication complexity of our distributed algorithm, we show that instead of using fully independent Cauchy variables, poly(k, d)-wise independent Cauchy variables suffice.

We start by stating two useful Lemmas from previous work [KNW10a].

Lemma 17.11.1 (Lemma 2.2 in [KNW10a])**.** *There exists an $\epsilon_0 > 0$ such that the following holds. Let n be a positive integer and $0 < \epsilon < \epsilon_0$, $0 < p < 2$ be given. Let $f : \mathbb{R} \to \mathbb{R}$ satisfy $\|f^{(\ell)}\|_\infty = O(\alpha^\ell)$ for all $\ell \geq 0$, for some α satisfying $\alpha^p \geq \log(1/\epsilon)$. Let $k = \alpha^p$. Let $a \in \mathbb{R}^n$ satisfy $\|a\|_p = O(1)$. Let X_i be a $3Ck$-independent family of p-stable random variables. Let $X = \sum_i a_i X_i$ and $Y = \sum_i a_i Y_i$. Then $\mathbb{E}[f(x)] = \mathbb{E}[f(Y)] + O(\epsilon)$.*

Lemma 17.11.2 (Lemma 2.5 in [KNW10a])**.** *There exist constants $c', \epsilon_0 > 0$ such that for all $c > 0$ and $0 < \epsilon < \epsilon_0$, and for all $[a, b] \subseteq \mathbb{R}$, there exists a function $J^c_{[a,b]} : \mathbb{R} \to \mathbb{R}$ satisfying:*

i. $\|(J^c_{[a,b]})^{(\ell)}\|_\infty = O(c^\ell)$ for all $\ell \geq 0$.

ii. For all x such that $a, b \notin [x - \epsilon, x + \epsilon]$, and as long as $c > c'\epsilon^{-1} \log^3(1/\epsilon)$, $|J^c_{[a,b]}(x) - I_{[a,b]}(x)| < \epsilon$.

17.11.2 Analysis of limited independent random Cauchy variables

Lemma 17.11.3. *Given a vector $y \in \mathbb{R}^n$, choose Z to be the $t \times n$ random Cauchy matrices with $1/t$ rescaling and $t = O(k \log k)$. The variables from different rows are fully independent, and the variables from the same rows are $O(1)$-wise independent. Then, we have*

$$\|Zy\|_1 \gtrsim \|y\|_1$$

holds with probability at least $1 - 2^{-\Omega(t)}$.

Proof. Let S denote the original fully independent matrix and Z denote the matrix for which the entries in the same row are w-wise independent, and the entries from different rows are fully independent. Notice we define the random matrices without rescaling by $1/t$ and it will be added back at the end. (We will decide w later)

We define random variable X such that $X = 1$ if $|(Zy)_i| \leq \frac{1}{50}$ and $X = 0$ otherwise. We also define random variable Y such that $Y = 1$ if $|(Sy)_i| \leq \frac{1}{50}$ and $Y = 0$ otherwise. Then, we have

$$\mathbb{E}[X] = \Pr\left[|(Zy)_i| \leq \frac{1}{50}\right] = \mathbb{E}\left[I_{[-\frac{1}{50},\frac{1}{50}]}((Zy)_i)\right]$$
$$\mathbb{E}[Y] = \Pr\left[|(Sy)_i| \leq \frac{1}{50}\right] = \mathbb{E}\left[I_{[-\frac{1}{50},\frac{1}{50}]}((Sy)_i)\right]$$

The goal is to show that $\mathbb{E}[X] \approx_\epsilon \mathbb{E}[Y]$. Following the same idea from [KNW10a], we need to argue this chain of inequalities,

$$\mathbb{E}[I_{[a,b]}(X)] \approx_\epsilon \mathbb{E}[J^c_{[a,b]}(X)] \approx_\epsilon \mathbb{E}[J^c_{[a,b]}(Y)] \approx_\epsilon \mathbb{E}[I_{[a,b]}(Y)]$$

Using Lemma 2.2 and Lemma 2.5 from [KNW10a], choosing sufficiently small constant ϵ(which implies $w = O(1)$), it follows that for each $i \in [t]$, we still have

$$\Pr\left[|(Zy)_i| > \frac{1}{50}\|y\|_1\right] \gtrsim \Pr\left[|(Sy)_i| > \frac{1}{50}\|y\|_1\right] \geq 0.9$$

Because all rows of Z are fully independent, using the Chernoff bound we can get that

$$\Pr\left[\|Zy\|_1 \lesssim t\|y\|_1\right] \leq \exp(-\Omega(t))$$

as we needed for the "no contraction" part of the net argument. □

For the no dilation, we need to argue that

Lemma 17.11.4. *Given a set of vectors $\{y_1, y_2, \ldots, y_d\}$ where $y_i \in \mathbb{R}^n, \forall i \in [d]$, choose Z to be the $t \times n$ random Cauchy matrices with $1/t$ rescaling and $t = O(k \log k)$, where the variables from different rows are fully independent, and the variables from the same rows are w-wise independent.*

I. *If $w = \widetilde{O}(dk)$, we have*

$$\sum_{i=1}^{d} \|Zy_i\|_1 \leq O(\log d) \sum_{i=1}^{d} \|y_i\|_1$$

holds with probability at least .999.

II. *If If $w = \widetilde{O}(d)$, we have*

$$\sum_{i=1}^{d} \|Zy_i\|_1 \leq O(k \log d) \sum_{i=1}^{d} \|y_i\|_1$$

holds with probability at least .999.

Proof. Let $m = t$. Let $S \in \mathbb{R}^{m \times n}$ denote the original fully independent matrix and Z denote the matrix that for each entry in the same row are w-wise independent, where the entries from different rows are fully independent. (We will decide on$w$ later)

Applying matrix S to those fixed set of vectors, we have

$$\sum_{i=1}^{d} \|Sy_i\|_1 = \sum_{i=1}^{d} \sum_{j=1}^{m} |\sum_{l=1}^{n} \frac{1}{m} S_{j,l} \cdot (y_i)_l| = \frac{1}{m} \sum_{i=1}^{d} \sum_{j=1}^{m} |\sum_{l=1}^{n} S_{j,l} \cdot (y_i)_l|$$

Applying matrix Z to those fixed set of vectors, we have a similar thing,

$$\sum_{i=1}^{d} \|Zy_i\|_1 = \sum_{i=1}^{d} \sum_{j=1}^{m} |\sum_{l=1}^{n} \frac{1}{m} Z_{j,l} \cdot (y_i)_l| = \frac{1}{m} \sum_{i=1}^{d} \sum_{j=1}^{m} |\sum_{l=1}^{n} Z_{j,l} \cdot (y_i)_l|$$

The goal is to argue that, for any $i \in [d], j \in [m]$,

$$\mathbb{E}\Big[|\sum_{l=1}^{n} Z_{j,l} \cdot (y_i)_l| \Big| \xi \Big] \lesssim \mathbb{E}\Big[|\sum_{l=1}^{n} S_{j,l} \cdot (y_i)_l| \Big| \xi \Big] + \delta$$

As long as δ is small enough, we are in a good shape.

Let $X = \frac{1}{\|y_i\|_1} \sum_{l=1}^{n} Z_{j,l}(y_i)_l$, and $Y = \frac{1}{\|y_i\|_1} \sum_{l=1}^{m} S_{j,l}(y_i)_l$. Let D be the truncating threshold of each Cauchy random variable. Define $T = O(\log D)$. On one hand, we have

$$\begin{aligned}
\mathbb{E}[|X||\xi] &\leq 2 \left(\Pr[X \in [0,1]|\xi] \cdot 1 + \sum_{j=0}^{T} \Pr[X \in (2^j, 2^{j+1}]|\xi] \cdot 2^{j+1} \right) \\
&\leq 2 \left(1 + \sum_{j=0}^{T} \Pr[I_{(2^j, 2^{j+1})}(X) = 1|\xi] \cdot 2^{j+1} \right) \\
&= 2 \left(1 + \sum_{j=0}^{T} \mathbb{E}[I_{(2^j, 2^{j+1})}(X)|\xi] \cdot 2^{j+1} \right) \qquad (17.46)
\end{aligned}$$

On the other hand, we can show

$$\begin{aligned}\mathbb{E}[|Y||\xi] &\geq 2\left(\Pr[X \in [0,1]|\xi] \cdot 0 + \sum_{j=0}^{T} \Pr[Y \in (2^j, 2^{j+1}]|\xi] \cdot 2^j\right) \\ &\geq 2\sum_{j=0}^{T} \Pr[I_{(2^j,2^{j+1}]}(Y) = 1|\xi] \cdot 2^j \\ &\geq 2\sum_{j=0}^{T} \mathbb{E}[I_{(2^j,2^{j+1}]}(Y)|\xi] \cdot 2^j \end{aligned} \tag{17.47}$$

Thus, we need to show that, for each j,

$$\mathbb{E}[I_{(2^j,2^{j+1})}(X)|\xi] \approx_\epsilon \mathbb{E}[I_{(2^j,2^{j+1}]}(Y)|\xi] \tag{17.48}$$

Following the same idea from [KNW10a], we need to argue this chain of inequalities,

$$\mathbb{E}[I_{[a,b]}(X)] \approx_\epsilon \mathbb{E}[J^c_{[a,b]}(X)] \approx_\epsilon \mathbb{E}[J^c_{[a,b]}(Y)] \approx_\epsilon \mathbb{E}[I_{[a,b]}(Y)]$$

We first show $\mathbb{E}[I_{[a,b]}(X)] \approx_\epsilon \mathbb{E}[J^c_{[a,b]}(X)]$. Notice that $I_{[a,b]}$ and $J_{[a,b]}$ are within ϵ everywhere except for two intervals of length $O(\epsilon)$. Also the Cauchy distribution is anticoncentrated (any length-$O(\epsilon)$ interval contains $O(\epsilon)$ probability mass) and $\|I_{[a,b]}\|_\infty, \|J^c_{[a,b]}\|_\infty = O(1)$, these intervals contribute $O(\epsilon)$ to the difference.

Second, we show $\mathbb{E}[J^c_{[a,b]}(X)] \approx_\epsilon \mathbb{E}[J^c_{[a,b]}(Y)]$. This directly follows by Lemma 17.11.1 by choosing $\alpha = O(\epsilon^{-1}\log^3(1/\epsilon))$.

Third, we show $\mathbb{E}[J^c_{[a,b]}(Y)] \approx_\epsilon \mathbb{E}[I_{[a,b]}(Y)]$. The argument is similar as the first step, but we need to show anticoncentration of Y. Suppose for any $t \in \mathbb{R}$ we had a nonnegative

function $f_{\epsilon,t} : \mathbb{R} \to \mathbb{R}$ symmetric about t satisfying:

I. $\|f_{t,\epsilon}^{(\ell)}\|_\infty = O(\alpha^\ell)$ for all $\ell \geq 0$, with $\alpha = O(1/\epsilon)$

II. $\mathbb{E}[f_{t,\epsilon}(z)] = O(\epsilon)$ for $z \sim \mathcal{D}_1$

III. $f_{t,\epsilon}(t+\epsilon) = \Omega(1)$

IV. $f_{t,\epsilon}(x)$ is strictly decreasing as $|x-t| \to \infty$

By I, II and Lemma 17.11.1 we could have $\mathbb{E}[f_{\epsilon,t}(Y)] \approx_\epsilon \mathbb{E}[f_{t,\epsilon}(z)] = O(\epsilon)$. Then, $\mathbb{E}[f_{t,\epsilon}(Y)] \geq f_{t,\epsilon}(t+\epsilon) \cdot \Pr[Y \in [t-\epsilon, t+\epsilon]] = \Omega(\Pr[Y \in [t-\epsilon, t+\epsilon]])$ by III and IV, implying anticoncentration in $[t-\epsilon, t+\epsilon]$ as desired. For the details of function $f_{t,\epsilon}$, we refer the readers to Section A.4 in [KNW10a].

Now, combining Equation (17.46), (17.47) and (17.48) gives

$$\begin{aligned}\mathbb{E}\left[|\sum_{l=1}^{n} Z_{j,l} \cdot (y_i)_l| \middle| \xi\right] &\lesssim \mathbb{E}\left[|\sum_{l=1}^{n} S_{j,l} \cdot (y_i)_l| \middle| \xi\right] + \sum_{j=0}^{T} 2^j \cdot \epsilon \cdot \|y_i\|_1 \\ &\lesssim \mathbb{E}\left[|\sum_{l=1}^{n} S_{j,l} \cdot (y_i)_l| \middle| \xi\right] + D \cdot \epsilon \cdot \|y_i\|_1\end{aligned}$$

Overall, for the fixed j, $S_{j,l}$ is $\widetilde{O}(1/\epsilon)$-independent family of Cauchy random variable. Choosing $D = O(dk)$ and $\epsilon = O(1/D)$, we can show

$$\frac{1}{m}\sum_{i=1}^{d}\sum_{j=1}^{m}\mathbb{E}\left[|\sum_{l=1}^{n} S_{j,l} \cdot (y_i)_l| \middle| \xi\right] \leq O(\log d)\sum_{i=1}^{d}\|y_i\|_1$$

as before. Notice that $D\epsilon\|y_i\|_1 = O(\|y_i\|_1)$. Thus, we complete the proof of the first result.

Choosing $D = O(dk)$ and $\epsilon = O(1/d)$, the dominant term becomes $D\epsilon\|y_i\|_1 = O(k\|y_i\|_1)$. Thus, we complete the proof of second result. □

Corollary 17.11.5. *Given $U \in \mathbb{R}^{n\times k}$, let $Z \in \mathbb{R}^{t\times n}$ be the same as the matrix stated in the Lemma 17.11.3, then with probability at least .95,*

$$\forall x \in \mathbb{R}^k, \|ZUx\|_1 \gtrsim \|Ux\|_1.$$

The proof is very similar to the proof of Lemma 17.5.14. Without loss of generality, we can suppose U is a well-conditioned basis. Due to Lemma 17.11.4, with arbitrarily high constant probability $\|ZU\|_1$ is bounded by $\text{poly}(t, k)$. By simply applying the net argument and using Lemma 17.11.3 to take a union bound over net points, we can get the above corollary.

17.12 Streaming Setting

Section 17.12.1 provides some notation and definitions about row-update streaming model and the turnstile streaming model. For some recent developments of row-update streaming and turnstile streaming models, we refer the readers to [CW09, KL11, GP14, Lib13, KLM+14a, BWZ16] and the references therein. Section 17.12.2 presents our turnstile streaming algorithm. Section 17.12.3 presents our row-update streaming algorithm.

17.12.1 Definitions

Definition 17.12.1 (Row-update model)**.** Let matrix $A \in \mathbb{R}^{n \times d}$ be a set of rows $A_1, \cdots, A_n$. In the row-update streaming model, each row of A will occur in the stream exactly once. But the rows can be in arbitrary order. An algorithm in this model is only allowed a single pass over these rows. At the end of the stream, the algorithm stores some information of A. The space of the algorithm is the total number of words required to store this information during the stream. Here, each word is $O(\log(nd))$ bits.

Definition 17.12.2 (Turnstile model)**.** At the beginning, let matrix $A \in \mathbb{R}^{n \times d}$ be a zero matrix. In the turnstile streaming model, there is a stream of update operations, and the i^{th} operation has the form (x_i, y_i, c_i) which means that A_{x_i, y_i} should be incremented by c_i. An algorithm in this model is only allowed a single pass over the stream. At the end of the stream, the algorithm stores some information of A. The space complexity of the algorithm is the total number of words required to store this information during the stream. Here, each word is $O(\log(nd))$ bits.

17.12.2 Turnstile model, $\mathrm{poly}(k, \log(d), \log(n))$ approximation

Definition 17.12.3 (Turnstile model ℓ_1-low rank approximation - rank-k subspace version)**.** Given matrix $A \in \mathbb{R}^{n \times d}$ and $k \in \mathbb{N}_+$, the goal is to propose an algorithm in the streaming model of Definition 17.12.2 such that

1. Upon termination, the algorithm outputs a matrix $V^* \in \mathbb{R}^{k \times d}$.
2. V^* satisfies that

$$\min_{U \in \mathbb{R}^{n \times k}} \|A - UV^*\|_1 \leq \mathrm{poly}(k, \log(d), \log(n)) \cdot \min_{U \in \mathbb{R}^{n \times k}, V \in \mathbb{R}^{k \times d}} \|A - UV\|_1.$$

3. The space complexity is as small as possible

Theorem 17.12.1. *Suppose $A \in \mathbb{R}^{n \times d}$ is given in the turnstile streaming model (See Definition 17.12.2), there is an algorithm(in Algorithm 17.8 without decomposition) which solves the problem in Definition 17.12.3 with constant probability. Further, the space complexity of the algorithm is $\mathrm{poly}(k) + \widetilde{O}(kd)$ words.*

Proof. **Correctness.** The correctness is implied by (IV) of Lemma 17.5.5, and the proof of Theorem 17.4.3. Notice that $L = T_1 AR, N = SAT_2, M = T_1 AT_2$, so $\widehat{X} \in \mathbb{R}^{O(k \log k) \times O(k \log k)}$ minimizes

$$\min_{\mathrm{rank}-k \ X} \|T_1 ARXSAT_2 - T_1 AT_2\|_F.$$

According to the proof of Theorem 17.4.3, $AR\widehat{X}SA$ gives a ℓ_1 rank-k $\mathrm{poly}(k, \log(d), \log(n))$-approximation to A. Because $\widehat{X} = \widehat{U}\widehat{\Sigma}\widehat{V}^\top$, $V^* = \widehat{\Sigma}\widehat{V}^\top SA$ satisfies:

$$\min_{U \in \mathbb{R}^{n \times k}} \|A - UV^*\|_1 \leq \mathrm{poly}(k, \log(d), \log(n)) \cdot \min_{U \in \mathbb{R}^{n \times k}, V \in \mathbb{R}^{k \times d}} \|A - UV\|_1.$$

Space complexity. Generating $\widetilde{O}(kd)$-wise independent random Cauchy variables needs $\widetilde{O}(kd)$ bits. The size of L, N and M are $k^2 \log^2 k, k^2 \log^3 k$ and $k^2 \log^3 k$ words separately. So the space of maintaining them is $O(k^2 \log^3 k)$ words. The size of D is $O(k \log k) \times d$, so maintaining it needs $O(kd \log k)$ words. Therefore, the total space complexity of the algorithm is $\mathrm{poly}(k) + \widetilde{O}(kd)$ words.

□

It is easy to extend our algorithm to output a decomposition. The formal definition of the decomposition problem is as the following:

Definition 17.12.4 (Turnstile model ℓ_1-low rank approximation - rank-k decomposition version)**.** Given matrix $A \in \mathbb{R}^{n \times d}$ and $k \in \mathbb{N}_+$, the goal is to propose an algorithm in the streaming model of Definition 17.12.2 such that

1. Upon termination, the algorithm outputs a matrix $U^* \in \mathbb{R}^{n \times k}, V^* \in \mathbb{R}^{k \times d}$.
2. U^*, V^* satisfies
$$\|A - U^* V^*\|_1 \leq \mathrm{poly}(k, \log(d), \log(n)) \cdot \min_{U \in \mathbb{R}^{n \times k}, V \in \mathbb{R}^{k \times d}} \|A - UV\|_1.$$
3. The space complexity is as small as possible

Theorem 17.12.2. *Suppose $A \in \mathbb{R}^{n \times d}$ is given by the turnstile streaming model (See Definition 17.12.2). There is an algorithm(in Algorithm 17.8 with decomposition) which solves the problem in Definition 17.12.4 with constant probability. Further, the space complexity of the algorithm is* $\mathrm{poly}(k) + \widetilde{O}(k(d+n))$ *words.*

Proof. **Correctness.** The only difference from the Algorithm 17.8 (without decomposition) is that the algorithm maintains C. Thus, finally it can compute $U^* = AR\widehat{U}$. Notice that $U^*V^* = AR\widehat{X}SA$, according to the proof of Theorem 17.4.3, U^*V^* gives a ℓ_1 rank-k $\mathrm{poly}(k, \log(d), \log(n))$-approximation to A.

Space complexity. Since the size of C is $O(nk \log k)$ words, the total space is $\mathrm{poly}(k) + \widetilde{O}(k(d+n))$ words.

□

17.12.3 Row-update model, $\mathrm{poly}(k)\log d$ approximation

Definition 17.12.5 (Row-update model ℓ_1-low rank approximation - rank-k subspace version)**.** Given matrix $A \in \mathbb{R}^{n\times d}$ and $k \in \mathbb{N}_+$, the goal is to propose an algorithm in the streaming model of Definition 17.12.1 such that

1. Upon termination, the algorithm outputs a matrix $V^* \in \mathbb{R}^{k\times d}$.
2. V^* satisfies that

$$\min_{U\in\mathbb{R}^{n\times k}} \|A - UV^*\|_1 \leq \mathrm{poly}(k)\log(d) \cdot \min_{U\in\mathbb{R}^{n\times k}, V\in\mathbb{R}^{k\times d}} \|A - UV\|_1.$$

3. The space complexity is as small as possible

Theorem 17.12.3. *Suppose $A \in \mathbb{R}^{n\times d}$ is given by the row-update streaming model (See Definition 17.12.1), there is an algorithm(in Algorithm 17.9 without decomposition) which solves the problem in Definition 17.12.5 with constant probability. Further, the space complexity of the algorithm is $\mathrm{poly}(k) + \widetilde{O}(kd)$ words.*

Proof. **Correctness.** Notice that $L = T_1BR, N = SBT_2, M = T_1BT_2$. Thus, $\widehat{X} \in \mathbb{R}^{O(k\log k)\times O(k\log k)}$ actually minimizes

$$\min_{\mathrm{rank}-k\ X} \|T_1BRXSBT_2 - T_1BT_2\|_F.$$

Also notice that B is just taking each row of A and replacing it with its nearest point in the row span of $S'A$. According to the proof of Theorem 17.4.6 and (IV) of Lemma 17.5.5 $BR\widehat{X}SB$ gives a $\mathrm{poly}(k)\log d$ ℓ_1 norm rank-k approximation to A. Since $\widehat{X} = \widehat{U}\widehat{\Sigma}\widehat{V}^\top$, $V^* = \widehat{\Sigma}\widehat{V}^\top SB$ satisfies:

$$\min_{U\in\mathbb{R}^{n\times k}} \|A - UV^*\|_1 \leq \mathrm{poly}(k, \log(d), \log(n)) \cdot \min_{U\in\mathbb{R}^{n\times k}, V\in\mathbb{R}^{k\times d}} \|A - UV\|_1.$$

Space complexity. Constructing sketching matrices needs $\widetilde{O}(kd)$ bits to store random seeds. Maintaining L, N, M needs $O(k^2 \log^3 k)$ words. The cost of storing $\widehat{X}$ is also $O(k^2 \log^2 k)$ words. Maintaining D needs $O(kd \log k)$ words. Therefore, the total space cost of the algorithm is $\mathrm{poly}(k) + \widetilde{O}(kd)$ words. □

It is easy to extend our algorithm to output a decomposition. The formal definition of the decomposition problem is as the following:

Definition 17.12.6 (Row-update model ℓ_1-low rank approximation - rank-k decomposition version)**.** Given matrix $A \in \mathbb{R}^{n \times d}$ and $k \in \mathbb{N}_+$, the goal is to propose an algorithm in the streaming model of Definition 17.12.1 such that

1. Upon termination, the algorithm outputs matrices $U^* \in \mathbb{R}^{n \times k}, V^* \in \mathbb{R}^{k \times d}$.
2. U^*, V^* satisfies that
$$\|A - U^* V^*\|_1 \leq \mathrm{poly}(k) \log d \cdot \min_{U \in \mathbb{R}^{n \times k}, V \in \mathbb{R}^{k \times d}} \|A - UV\|_1.$$
3. The space complexity is as small as possible.

Theorem 17.12.4. *Suppose $A \in \mathbb{R}^{n \times d}$ is given by the row-update streaming model (See Definition 17.12.1), there is an algorithm(in Algorithm 17.9 with decomposition) which solves the problem in Definition 17.12.6 with constant probability. Further, the space complexity of the algorithm is* $\mathrm{poly}(k) + \widetilde{O}(k(n+d))$ *words.*

Proof. **Correctness.** The only difference is that the above algorithm maintains C. Thus, We can compute $U^* = C\widehat{U}$ in the end. Notice that $U^* V^* = BR\widehat{X}SB$, according to the proof of Theorem 17.4.6, $U^* V^*$ gives a $\mathrm{poly}(k) \log d$ ℓ_1 norm rank-k approximation to A.

Space complexity. Since the size of C is $nk \log k$ words, the total space is $\mathrm{poly}(k) + \widetilde{O}(k(n+d))$ words. $\square$

17.13 Distributed Setting

Section 17.13.1 provides some notation and definitions for the Row-partition distributed model and the Arbitrary-partition model. These two models were recently studied in a line of works such as [TD99, QOSG02, BCL05, BRB08, MBZ10, FEGK13, PMvdG+13, KVW14, BKLW14, BLS+16b, BWZ16, WZ16]. Section 17.13.2 and 17.13.3 presents our distributed protocols for the Arbitrary-partition distributed model. Section 17.13.4 and 17.13.5 presents our distributed protocols for the Row-partition distributed model.

17.13.1 Definitions

Definition 17.13.1 (Row-partition model [BWZ16])**.** There are s machines, and the i^{th} machine has a matrix $A_i \in \mathbb{R}^{n_i \times d}$ as input. Suppose $n = \sum_{i=1}^{s} n_i$, and the global data matrix $A \in \mathbb{R}^{n \times d}$ is denoted as

$$\begin{pmatrix} A_1 \\ A_2 \\ \cdots \\ A_s \end{pmatrix},$$

we say A is row-partitioned into these s matrices distributed in s machines respectively. Furthermore, there is a machine which is a coordinator. The model only allows communication between the machines and the coordinator. The communication cost in this model is the total number of words transferred between machines and the coordinator. Each word is $O(\log(snd))$ bits.

Definition 17.13.2 (Arbitrary-partition model [BWZ16])**.** There are s machines, and the i^{th} machine has a matrix $A_i \in \mathbb{R}^{n \times d}$ as input. Suppose the global data matrix $A \in \mathbb{R}^{n \times d}$ is denoted as $A = \sum_{i=1}^{s} A_i$. We say A is arbitrarily partitioned into these s matrices distributed in s machines respectively. Furthermore, there is a machine which is a coordinator. The model only allows communication between the machines and the coordinator. The communication cost in this model is the total number of words transferred between machines and the coordinator. Each word is $O(\log(snd))$ bits.

17.13.2 Arbitrary-partition model, subspace, $\mathrm{poly}(k, \log(d), \log(n))$ approximation

Definition 17.13.3 (Arbitrary-partition model ℓ_1-low rank approximation - rank-k subspace version)**.** Given matrix $A \in \mathbb{R}^{n \times d}$ arbitrarily partitioned into s matrices $A_1, A_2, \cdots, A_s$ distributed in s machines respectively, and $k \in \mathbb{N}_+$, the goal is to propose a protocol in the model of Definition 17.13.2 such that

1. Upon termination, the protocol leaves a matrix $V^* \in \mathbb{R}^{k \times d}$ on the coordinator.
2. V^* satisfies that

$$\min_{U \in \mathbb{R}^{n \times k}} \|A - UV^*\|_1 \leq \mathrm{poly}(k, \log(d), \log(n)) \cdot \min_{U \in \mathbb{R}^{n \times k}, V \in \mathbb{R}^{k \times d}} \|A - UV\|_1.$$

3. The communication cost is as small as possible

Theorem 17.13.1. *Suppose $A \in \mathbb{R}^{n \times d}$ is partitioned in the arbitrary partition model (See Definition 17.13.2). There is a protocol(in Algorithm 17.10) which solves the problem in Definition 17.13.3 with constant probability. Further, the communication complexity of the protocol is $s(\mathrm{poly}(k) + \widetilde{O}(kd))$ words.*

Proof. **Correctness.** The correctness is shown by the proof of Theorem 17.4.3 and (IV) of Lemma 17.5.5. Notice that $\widehat{X} \in \mathbb{R}^{O(k \log k) \times O(k \log k)}$ minimizes

$$\min_{\mathrm{rank}-k \ X} \|LXN - M\|_F.$$

which is

$$\min_{\mathrm{rank}-k \ X} \|T_1 ARXSAT_2 - T_1 AT_2\|_F.$$

According to the proof of Theorem 17.4.3, $AR\widehat{X}SA$ gives an ℓ_1 rank-k $\mathrm{poly}(k, \log(d), \log(n))$-approximation to A. Because $\widehat{X} = \widehat{U}\widehat{\Sigma}\widehat{V}^\top$, $V^* = \widehat{\Sigma}\widehat{V}^\top SA$ satisfies:

$$\min_{U \in \mathbb{R}^{n \times k}} \|A - UV^*\|_1 \leq \mathrm{poly}(k, \log(d), \log(n)) \cdot \min_{U \in \mathbb{R}^{n \times k}, V \in \mathbb{R}^{k \times d}} \|A - UV\|_1.$$

Communication complexity. Since the random seed generates $\widetilde{O}(kd)$-wise independent random Cauchy variables, the cost of line 5 is $\widetilde{O}(skd)$ bits. The size of L_i, N_i and M_i are $k^2 \log^2 k, k^2 \log^3 k$ and $k^2 \log^3 k$ words separately. So the cost of line 15 is $O(sk^2 \log^3 k)$ words. Because the size of $\widehat{X}$ is $O(k^2 \log^2 k)$, the cost of line 22 is $O(sk^2 \log^2 k)$ words. line 30 needs skd words of communication. Therefore, the total communication of the protocol is $s(\mathrm{poly}(k) + \widetilde{O}(kd))$ words.

□

17.13.3 Arbitrary-partition model, decomposition, $\mathrm{poly}(k, \log(d), \log(n))$ approximation

Definition 17.13.4 (Arbitrary-partition model ℓ_1-low rank approximation - rank-k decomposition version)**.** Given matrix $A \in \mathbb{R}^{n \times d}$ arbitrarily partitioned into s matrices $A_1, A_2, \cdots, A_s$ distributed in s machines respectively, and $k \in \mathbb{N}_+$, the goal is to propose a protocol in the model of Definition 17.13.2 such that

1. Upon termination, the protocol leave matrices $U^* \in \mathbb{R}^{n \times k}, V^* \in \mathbb{R}^{k \times d}$ on the coordinator.

2. U^*, V^* satisfies that

$$\|A - U^* V^*\|_1 \leq \mathrm{poly}(k, \log(d), \log(n)) \cdot \min_{U \in \mathbb{R}^{n \times k}, V \in \mathbb{R}^{k \times d}} \|A - UV\|_1.$$

3. The communication cost is as small as possible.

Theorem 17.13.2. *Suppose $A \in \mathbb{R}^{n \times d}$ is partitioned in the arbitrary partition model (See Definition 17.13.2). There is a protocol(in Algorithm 17.10 with decomposition) which solves the problem in Definition 17.13.4 with constant probability. Further, the communication complexity of the protocol is $s(\mathrm{poly}(k) + \widetilde{O}(k(d+n)))$ words.*

Proof. **Correctness.** The only difference from the protocol (without decomposition) in Section 17.13.2 is that the protocol sends U_i. Thus, the coordinator can compute $U^* = AR\widehat{U}$. Notice that $U^* V^* = AR\widehat{X}SA$. According to the proof of Theorem 17.4.3, $U^* V^*$ gives a ℓ_1 rank-k $\mathrm{poly}(k, \log(d), \log(n))$-approximation to A.

Communication complexity. Since the size of U_i is kn words, the total communication is $s(\mathrm{poly}(k) + \widetilde{O}(k(d+n)))$ words.

□

17.13.4 Row-partition model, subspace, $\mathrm{poly}(k)\log d$ approximation

Definition 17.13.5 (Row-partition model ℓ_1-low rank approximation - rank-k subspace version)**.** Given matrix $A \in \mathbb{R}^{n\times d}$ row-partitioned into s matrices $A_1, A_2, \cdots, A_s$ distributed in s machines respectively, and $k \in \mathbb{N}_+$, the goal is to propose a protocol in the model of Definition 17.13.1 such that

1. Upon termination, the protocol leaves a matrix $V^* \in \mathbb{R}^{k\times d}$ on the coordinator.
2. V^* satisfies that
$$\min_{U\in\mathbb{R}^{n\times k}} \|A - UV^*\|_1 \leq \mathrm{poly}(k)\log d \cdot \min_{U\in\mathbb{R}^{n\times k}, V\in\mathbb{R}^{k\times d}} \|A - UV\|_1.$$
3. The communication cost is as small as possible

Theorem 17.13.3. *Suppose $A \in \mathbb{R}^{n\times d}$ is partitioned in the row partition model (See Definition 17.13.1). There is a protocol(in Algorithm 17.11 without decomposition) which solves the problem in Definition 17.13.5 with constant probability. Further, the communication complexity of the protocol is $s(\mathrm{poly}(k) + \widetilde{O}(kd))$ words.*

Proof. **Correctness.** For convenience, we denote matrices $B \in \mathbb{R}^{n\times d}, S \in \mathbb{R}^{O(k\log^2 k)\times n}$ and $T_1 \in \mathbb{R}^{O(k\log^2 k)\times n}$ as
$$B = \begin{pmatrix} B_1 \\ B_2 \\ \cdots \\ B_s \end{pmatrix} \quad S = \begin{pmatrix} S_1 & S_2 & \cdots & S_s \end{pmatrix} \quad T_1 = \begin{pmatrix} T_{11} & T_{12} & \cdots & T_{1s} \end{pmatrix}.$$
Notice that $L = T_1BR, N = SBT_2, M = T_1BT_2$. Thus, $\widehat{X} \in \mathbb{R}^{O(k\log k)\times O(k\log k)}$ actually minimizes
$$\min_{\mathrm{rank}-k\ X} \|T_1BRXSBT_2 - T_1BT_2\|_F.$$

Also notice that B is just taking each row of A and replacing it with its nearest point in the row span of $S'A$. According to the proof of Theorem 17.4.6, $BR\widehat{X}SB$ gives a $\mathrm{poly}(k)\log d$ ℓ_1 norm rank-k approximation to A. Since $\widehat{X} = \widehat{U}\widehat{\Sigma}\widehat{V}^\top$, $V^* = \widehat{\Sigma}\widehat{V}^\top SB$ satisfies:

$$\min_{U\in\mathbb{R}^{n\times k}} \|A - UV^*\|_1 \leq \mathrm{poly}(k, \log(d), \log(n)) \cdot \min_{U\in\mathbb{R}^{n\times k}, V\in\mathbb{R}^{k\times d}} \|A - UV\|_1.$$

Communication complexity. Since the R and T_2 are $\widetilde{O}(kd)$-wise independent, line 6 needs $O(sW)$ bits of communication. Line 16 needs $O(sk^2\log^3 k)$ words. The cost of line 23 is $O(sk^2\log^2 k)$ words. Line 32 needs skd words of communication. Therefore, the total communication of the protocol is $s(\mathrm{poly}(k) + \widetilde{O}(kd))$ words.

□

17.13.5 Row-partition model, decomposition, $\mathrm{poly}(k)\log d$ approximation

Definition 17.13.6 (Row-partition model ℓ_1-low rank approximation - rank-k decomposition version)**.** Given matrix $A \in \mathbb{R}^{n\times d}$ row partitioned into s matrices $A_1, A_2, \cdots, A_s$ distributed in s machines respectively, and a positive integer $k < \mathrm{rank}(A)$, the goal is to propose a protocol in the model of Definition 17.13.1 such that

1. Upon termination, the protocol leaves matrices $U^* \in \mathbb{R}^{n\times k}, V^* \in \mathbb{R}^{k\times d}$ on the coordinator.

2. U^*, V^* satisfies that

$$\|A - U^*V^*\|_1 \leq \mathrm{poly}(k)\log d \cdot \min_{U\in\mathbb{R}^{n\times k}, V\in\mathbb{R}^{k\times d}} \|A - UV\|_1.$$

3. The communication cost is as small as possible.

Theorem 17.13.4. *Suppose $A \in \mathbb{R}^{n\times d}$ is partitioned in the row partition model (See Definition 17.13.1). There is a protocol(in Algorithm 17.11 with decomposition) which solves the problem in Definition 17.13.6 with constant probability. Further, the communication complexity of the protocol is $s(\mathrm{poly}(k) + \widetilde{O}(k(n+d)))$ words.*

Proof. **Correctness.** The only difference is that the above protocol sends U_i. Thus, the coordinator can compute $U^* = BR\widehat{U}$. Notice that $U^*V^* = BR\widehat{X}SB$, according to the proof of Theorem 17.4.6, U^*V^* gives a $\mathrm{poly}(k)\log d$ ℓ_1 norm rank-k approximation to A.

Communication complexity. Since the size of U_i is kn words, the total communication is $s(\mathrm{poly}(k) + \widetilde{O}(k(n+d)))$ words.

□

Algorithm 17.8 Turnstile Streaming Algorithm

procedure TURNSTILESTREAMING(k,$\mathcal{S}$)
  Construct sketching matrices $S \in \mathbb{R}^{O(k\log k)\times n}, R \in \mathbb{R}^{d\times O(k\log k)}, T_1 \in \mathbb{R}^{O(k\log k)\times n}, T_2 \in \mathbb{R}^{d\times O(k\log^2 k)}$ where $R,\ T_2$ are fully independent random Cauchy matrices, and $S,\ T_1$ are random Cauchy matrices with fully independent variables across different rows and $\widetilde{O}(d)$-wise independent variables from the same row.
  Initialize matrices:
  $L \leftarrow \{0\}^{O(k\log k)\times O(k\log k)}, N \leftarrow \{0\}^{O(k\log k)\times O(k\log^2 k)}$.
  $M \leftarrow \{0\}^{O(k\log k)\times O(k\log^2 k)}, D \leftarrow \{0\}^{O(k\log k)\times d}$.
  if need decomposition **then**
    $C \leftarrow \{0\}^{n\times O(k\log k)}$.
  end if
  for $i \in [l]$ **do**
    Receive update operation (x_i, y_i, c_i) from the data stream $\mathcal{S}$.
    for $r = 1 \to O(k\log k), s = 1 \to O(k\log k)$ **do**
      $L_{r,s} \leftarrow L_{r,s} + T_{1r,x_i} \cdot c_i \cdot R_{y_i,s}$.
    end for
    for $r = 1 \to O(k\log k), s = 1 \to O(k\log^2 k)$ **do**
      $N_{r,s} \leftarrow N_{r,s} + S_{r,x_i} \cdot c_i \cdot T_{2y_i,s}$.
    end for
    for $r = 1 \to O(k\log k), s = 1 \to O(k\log^2 k)$ **do**
      $M_{r,s} \leftarrow M_{r,s} + T_{1r,x_i} \cdot c_i \cdot T_{2y_i,s}$.
    end for
    for $r = 1 \to O(k\log k)$ **do**
      $D_{r,y_i} \leftarrow D_{r,s} + S_{r,x_i} \cdot c_i$.
    end for
    if need decomposition **then**
      for $s = 1 \to O(k\log k)$ **do**
        $C_{x_i,s} \leftarrow C_{x_i,s} + c_i \cdot R_{y_i,s}$.
      end for
    end if
  end for
  Compute the SVD of $L = U_L\Sigma_L V_L^\top$.
  Compute the SVD of $N = U_N\Sigma_N V_N^\top$.
  Compute $\widehat{X} = L^\dagger(U_L U_L^\top M V_N V_N^\top)_k N^\dagger$.
  Compute the SVD of $\widehat{X} = \widehat{U}\widehat{\Sigma}\widehat{V}^\top$.
  if need decomposition **then**
    return $V^* = \widehat{\Sigma}\widehat{V}^\top D, U^* = C\widehat{U}$.
  else
    return $V^* = \widehat{\Sigma}\widehat{V}^\top D$.
  end if
end procedure

Algorithm 17.9 Row Update Streaming Algorithm

procedure RowUpdateStreaming($k, \mathcal{S}$)
  Construct sketching matrices $S' \in \mathbb{R}^{O(k\log k)\times n}, S \in \mathbb{R}^{O(k\log k)\times n}, R \in \mathbb{R}^{d\times O(k\log k)}, T_1 \in \mathbb{R}^{O(k\log k)\times n}, T_2 \in \mathbb{R}^{d\times O(k\log^2 k)}$ where $R,\ T_2$ are fully independent random Cauchy variables, and $S\ , S'\ , T_1$ are random Cauchy matrices with fully independent random variables from different rows and $\widetilde{O}(d)$-wise independent in the same row.
  Initialize matrices:
  $L \leftarrow \{0\}^{O(k\log k)\times O(k\log k)}, N \leftarrow \{0\}^{O(k\log k)\times O(k\log^2 k)}$.
  $M \leftarrow \{0\}^{O(k\log k)\times O(k\log^2 k)}, D \leftarrow \{0\}^{O(k\log k)\times d}$.
  if need decomposition **then**
    $C \leftarrow \{0\}^{n\times O(k\log k)}$.
  end if
  for $i \in [n]$ **do**
    Receive a row update (i, A_i) from the data stream $\mathcal{S}$.
    Compute $Y_i^* \in \mathbb{R}^{1\times O(k\log k)}$ which minimizes $\min_{Y\in\mathbb{R}^{1\times O(k\log k)}} \|YS'_{:,i}A_i - A_i\|_1$.
    Compute $B_i = Y_i^* S'_{:,i} A_i$.
    for $r = 1 \to O(k\log k), s = 1 \to O(k\log k), j = 1 \to d$ **do**
      $L_{r,s} \leftarrow L_{r,s} + T_{1r,i} \cdot B_{i,j} \cdot R_{j,s}$.
    end for
    for $r = 1 \to O(k\log k), s = 1 \to O(k\log^2 k), j = 1 \to d$ **do**
      $N_{r,s} \leftarrow N_{r,s} + S_{r,i} \cdot B_{i,j} \cdot T_{2j,s}$.
    end for
    for $r = 1 \to O(k\log k), s = 1 \to O(k\log^2 k), j = 1 \to d$ **do**
      $M_{r,s} \leftarrow M_{r,s} + T_{1r,i} \cdot B_{i,j} \cdot T_{2j,s}$.
    end for
    for $r = 1 \to O(k\log k), j = 1 \to d$ **do**
      $D_{r,j} \leftarrow D_{r,j} + S_{r,i} \cdot B_{i,j}$.
    end for
    if need decomposition **then**
      for $s = 1 \to O(k\log k), j = 1 \to d$ **do**
        $C_{i,s} := C_{i,s} + B_{i,j} \cdot R_{j,s}$.
      end for
    end if
  end for
  Compute the SVD of $L = U_L \Sigma_L V_L^\top$.
  Compute the SVD of $N = U_N \Sigma_N V_N^\top$.
  Compute $\widehat{X} = L^\dagger (U_L U_L^\top M V_N V_N^\top)_k N^\dagger$.
  Compute the SVD of $\widehat{X} = \widehat{U}\widehat{\Sigma}\widehat{V}^\top$.
  if need decomposition **then**
    return $V^* = \widehat{\Sigma}\widehat{V}^\top D, U^* = C\widehat{U}$. 1328
  else
    return $V^* = \widehat{\Sigma}\widehat{V}^\top D$.
  end if
end procedure

Algorithm 17.10 Arbitrary Partition Distributed Protocol

procedure ARBITRARYPARTITIONDISTRIBUTEDPROTOCOL(k,s,A)
    $A \in \mathbb{R}^{n\times d}$ was arbitrarily partitioned into s matrices $A_1, \cdots, A_s \in \mathbb{R}^{n\times d}$ distributed in s machines.

Coordinator	**Machines** i
Chooses a random seed.	
Sends it to all machines.	
$- - - - - - - - - - >$	
	Agrees on R, T_2 which are fully independent random Cauchy matrices.
	Agrees on S, T_1 which are random Cauchy matrices with fully independent entries from different rows, and $\widetilde{O}(d)$-wise independent variables from the same row.
	Computes $L_i = T_1 A_i R, N_i = S A_i T_2$.
	Computes $M_i = T_1 A_i T_2$.
	Sends L_i, N_i, M_i to the coordinator.
$< - - - - - - - - - -$	
Computes $L = \sum_{i=1}^{s} L_i, N = \sum_{i=1}^{s} N_i$.	
Computes $M = \sum_{i=1}^{s} M_i$.	
Computes the SVD of $L = U_L \Sigma_L V_L^\top$.	
Computes the SVD of $N = U_N \Sigma_N V_N^\top$.	
Computes $\widehat{X} = L^\dagger (U_L U_L^\top M V_N V_N^\top)_k N^\dagger$.	
Sends $\widehat{X}$ to machines.	
$- - - - - - - - - - >$	
	Computes the SVD of $\widehat{X} = \widehat{U}\widehat{\Sigma}\widehat{V}^\top$.
	Computes $V_i^* = \widehat{\Sigma}\widehat{V}^\top S A_i$.
	If need decomposition
	$U_i^* = A_i R \widehat{U}$.
	Sends U_i^*, V_i^* to the coordinator.
	Else
	Sends V_i^* to the coordinator.
	Endif
$< - - - - - - - - - -$	
If need decomposition,	
return $V^* = \sum_{i=1}^{s} V_i^*$, $U^* = \sum_{i=1}^{s} U_i^*$.	
Else	
return $V^* = \sum_{i=1}^{s} V_i^*$.	
Endif	

end procedure

Algorithm 17.11 Row Partition Distributed Protocol

procedure RowPartitionDistributedProtocol(k,s,A)
    $A \in \mathbb{R}^{n \times d}$ was row partitioned into s matrices $A_1 \in \mathbb{R}^{n_1 \times d}, \cdots, A_s \in \mathbb{R}^{n_s \times d}$ distributed in s machines.

Coordinator	**Machines** i
Chooses a random seed.	
Sends it to all machines.	
$- - - - - - - - - - >$	
	Agrees on $R,\ T_2$ which are fully independent random Cauchy variables. Generates random Cauchy matrices $S_i' \in \mathbb{R}^{O(k \log k) \times n_i}$, $S_i, T_{1i} \in \mathbb{R}^{O(k \log^2 k) \times n_i}$.
	Computes $Y_i^* = \underset{Y \in \mathbb{R}^{n_i \times O(k \log k)}}{\arg\min} \|Y S_i' A_i - A_i\|_1$.
	Computes $B_i = Y_i^* S_i' A_i$.
	Computes $L_i = T_{1i} B_i R, N_i = S_i B_i T_2$.
	Computes $M_i = T_{1i} B_i T_2$.
	Sends L_i, N_i, M_i to the coordinator.
$< - - - - - - - - - -$	
Computes $L = \sum_{i=1}^s L_i, N = \sum_{i=1}^s N_i$.	
Computes $M = \sum_{i=1}^s M_i$.	
Computes the SVD of $L = U_L \Sigma_L V_L^\top$.	
Computes the SVD of $N = U_N \Sigma_N V_N^\top$.	
Computes $\widehat{X} = L^\dagger (U_L U_L^\top M V_N V_N^\top)_k N^\dagger$.	
Sends $\widehat{X}$ to machines.	
$- - - - - - - - - - >$	
	Computes the SVD of $\widehat{X} = \widehat{U} \widehat{\Sigma} \widehat{V}^\top$.
	Computes $V_i^* = \widehat{\Sigma} \widehat{V}^\top S_i B_i$.
	If need decomposition
	Computes $U_i^* = B_i R \widehat{U}$.
	Sends U_i^*, V_i^* to the coordinator.
	Else
	Sends V_i^* to the coordinator.
	Endif
$< - - - - - - - - - -$	
If need decomposition	
return $V^* = \sum_{i=1}^s V_i^*$, $U^* = \sum_{i=1}^s U_i^*$.	
Else	
return $V^* = \sum_{i=1}^s V_i^*$.	
Endif	

end procedure

Chapter 18

L1 Low Rank Approximation with Regularization

Low rank matrix approximation with respect to the entrywise ℓ_1 norm error is a fundamental problem in both theoretical computer science and machine learning community. At the same time, ℓ_1 regularization (a.k.a. the LASSO) has been used for many statistical learning techniques on finding solutions with good performance and interpretability. In this paper, we consider how to combine the two techniques and study the ℓ_1 regularized version of ℓ_1-low rank approximation problem. In particular, given matrix $A \in \mathbb{R}^{n\times n}$, parameter $k \geq 1, \lambda > 0$, the problem we interested in is

$$\min_{U\in\mathbb{R}^{n\times k}, V\in\mathbb{R}^{k\times n}} \|UV - A\|_1 + \lambda(\|U\|_1 + \|V\|_1),$$

where $\|B\|_1 := \sum_{i=1}^{n}\sum_{j=1}^{n}|B_{i,j}|$ for any matrix B.

We introduce the first algorithm for solving this problem with probable guarantee. Precisely, we compute two matrices $U \in \mathbb{R}^{n\times k}$ and $V \in \mathbb{R}^{k\times n}$ such that

$$\|UV - A\|_1 + \lambda(\|U\|_1 + \|V\|_1) \leq \alpha\,\mathrm{OPT}$$

holds with probability at least 9/10, where OPT is the cost of the optimal solution, and α is approximation ratio.

In addition, our algorithm shows a trade-off between running time and approximation ratio:

- $O(\mathrm{nnz}(A)) + n\,\mathrm{poly}(\log n, k)$ time with $\alpha = \mathrm{poly}(k, \log n)$.
- $\widetilde{O}(k)\,\mathrm{nnz}(A) + n\,\mathrm{poly}(\log n, k)$ time with $\alpha = \widetilde{O}(k^2)\log^3 n$.

where $\mathrm{nnz}(A)$ denotes the number of non-zero entries in A.

18.1 Introduction

The low-rank matrix approximation is a core problem in machine learning and data mining. It approximating a matrix by one whose rank is less than that of the original matrix. The goal of low-rank approximation is to obtain more condensed representations of the data with limited loss of information. Given a matrix $A \in \mathbb{R}^{m\times n}$ and a desired rank k, the low-rank matrix approximation problem can be stated as

$$\min_{\widehat{A}\in\mathbb{R}^{m\times n}} \|\widehat{A} - A\| \quad \text{subject to} \quad \text{rank}(\widehat{A}) \leq k, \tag{18.1}$$

where $\|\cdot\|$ is a matrix norm measuring the approximation error.

Generally speaking, a rank-k matrix $\widehat{A} \in \mathbb{R}^{n\times n}$ can be expressed by the matrix product of two matrices U, V, where $U \in \mathbb{R}^{n\times k}$, $V \in \mathbb{R}^{k\times n}$. The above equation can be rewritten as

$$\min_{U\in\mathbb{R}^{n\times k}, V\in\mathbb{R}^{k\times n}} \|UV - A\|. \tag{18.2}$$

The key point in low-rank approximation problem is how to choose the matrix norm $\|\cdot\|$. The most widely used norm is the Frobenius norm, which is defined as $\|B\|_F = (\sum_{i=1}^{n}\sum_{j=1}^{n} B_{i,j}^2)^{\frac{1}{2}}$ for matrix $B \in \mathbb{R}^{n\times n}$, The analytical optimal solution for Frobenius norm can be obtained by using the singular value decomposition (SVD) of B.

The Frobenius norm fits well with Gaussian noise, but it is sensitive to sparse outliers. Thus, an interesting extension for solving (18.2) is when the noise is sparse. This problem can be written as

$$\min_{U\in\mathbb{R}^{n\times k}, V\in\mathbb{R}^{k\times n}} \|UV - A\|_0, \tag{18.3}$$

where $\|\cdot\|_0$ is ℓ_0 norm which measures the sparsity of the matrix.

Nevertheless, solving ℓ_0 low-rank approximation has been proven to be NP-hard [GG11]. In practice, instead of solving (18.3), a more feasible way is to solve

$$\min_{U \in \mathbb{R}^{n \times k}, V \in \mathbb{R}^{k \times n}} \|UV - A\|_1, \tag{18.4}$$

where we replace the ℓ_0 by ℓ_1, and $\|B\|_1 := \sum_{i=1}^{n} \sum_{j=1}^{n} |B_{i,j}|$ for matrix $B \in \mathbb{R}^{n \times n}$. This problem sometimes is referred to as robust PCA [CLMW11]. Finding solution to (18.4) has been widely studied by machine learning and theoretical computer science community for a long time. It has been validated both empirically and theoretically that ℓ_1-norm is usually more robust than ℓ_2-norm and tends to preserve the sparsity of the original matrix [CLMW11, XCS10, WY13, XCS11]. Also, ℓ_1 norm has been extensively studied in many other problems such as sparse recovery [PW11], Fourier transform [BCG+12], furthest pair [Yao82, GBT84], closest pair [PS85, DHKP97] and so on.

Although solving (18.4) has been proven to be NP-hard [GV15], there are a lot of efforts for the ℓ_1 low-rank approximation problem in a heuristic way. [KK05] introduced a method based on alternating convex minimization. [WYWY12] devise a parallelizable expectation-maximization (EM) algorithm which can potentially be applied to large-scale applications. [BDB13] exploit the efficient calculation of the optimal solution of the ℓ_1-norm best-fit hyperplane problem. Unfortunately, those heuristic method do not come with any performance guarantees. Recently, [SWZ17, CGK+17b] introduce efficient algorithms for ℓ_1-low rank approximation with provable approximation guarantees.

ℓ_1 regularization can be very effective for many statistical learning techniques [LZWW16, Tib96]. However, none of the previous ℓ_1 low rank approximation algorithms can handle the regularized version of ℓ_1 low rank approximation problem. Though [LZWW16] studies ℓ_1

principal component analysis problem with ℓ_1 regularization, their algorithm does not have any provable guarantee. [ACW17] studied regularized low rank approximation problem, but they can only deal with the regularized Frobenious norm low rank approximation, i.e. $\min_{U\in\mathbb{R}^{n\times k},V\in\mathbb{R}^{k\times n}} \|UV-A\|_F^2+\lambda(\|U\|_F^2+\|V\|_F^2)$, and their techniques cannot be extended to regularized ℓ_1 low rank approximation case. It is natural to ask whether there is an efficient algorithm for solving ℓ_1-regularized ℓ_1 low-rank approximation with a good approximation ratio.

In this work, we study the ℓ_1 regularized version of ℓ_1 low rank approximation problem. The problem can be formulated as

$$\min_{U\in\mathbb{R}^{n\times k},V\in\mathbb{R}^{k\times n}} \|UV-A\|_1+\lambda(\|U\|_1+\|V\|_1), \tag{18.5}$$

where λ is a non-negative tunable parameter which controls the weight between approximation error and the sparsity of U,V. When $\lambda=0$, (18.5) degenerates to (18.4). By solving (18.5), our model provides the flexibility that can find the balance between the reconstruction error and the sparsity of each low-rank component. Notably, when $\lambda=0$, our results exactly match the results shown in [SWZ17].

18.1.1 Our results

We give the first efficient algorithms for ℓ_1 low-rank approximation with ℓ_1 regularization problem with both provable running time and approximation ratio guarantees. A crucial tool in our algorithms is sketching, i.e. compressing the large input data into a small size summary which is enough to provide a good solution to the original input. In particular, we use the Cauchy transformation as our sketching tool. The Cauchy transformation is widely used in ℓ_1 numerical linear algebra problems such as subspace embedding

and linear regression (see e.g. [SW11, MM13]). However, people did not know how to use the Cauchy transformation technique to solve low-rank matrix approximation problem until recent work [SWZ17]. In our work, we show how to extend this technique to handle ℓ_1 regularization. But the analysis of the algorithms is very different from the previous works [SWZ17, SW11, MM13]. Algorithm 18.1 shows the framework of our algorithms.

There are different kind of the Cauchy transformations given by previous works [SW11, MM13, CDMI+13]. They have different trade-offs between distortion ratio and the running time of applying the transformation. By using different Cauchy transformation, we show algorithms with different running time/approximation ratio tradeoffs.

By using sparse Cauchy transform [MM13] (see Section 18.2.2), we can get the following result,

Theorem 18.1.1 (Input sparsity time algorithm)**.** *Given matrix $A \in \mathbb{R}^{n \times n}$, parameters $k \geq 1$, $\lambda > 0$, let* OPT *denote* $\min_{U' \in \mathbb{R}^{n \times k}, V' \in \mathbb{R}^{k \times n}} \|U'V' - A\|_1 + \lambda(\|U'\|_1 + \|V'\|_1)$*, there is an algorithm that runs in $O(\mathrm{nnz}(A) + n\,\mathrm{poly}(k, \log n))$ time to output two matrices $U \in \mathbb{R}^{n \times k}$, $V \in \mathbb{R}^{k \times n}$ such that,*

$$\|UV - A\|_1 + \lambda(\|U\|_1 + \|V\|_1) \leq \mathrm{poly}(k, \log n) \cdot \mathrm{OPT}$$

holds with probability at least $9/10$, where $\mathrm{nnz}(A)$ is the number of nonzeros in matrix A.

By using dense Cauchy transform [SW11] (see Section 18.2.2), we can get a different trade-off between approximation ration and running time. Theorem 18.1.1 provides a $\mathrm{nnz}(A)$ running time with $\mathrm{poly}(k, \log n)$-approximation. Here we provide a better approximation ratio under polynomial running time.

Theorem 18.1.2 ($\widetilde{O}(k^2)\,\mathrm{poly}(\log n)$-approximation)**.** *Given matrix $A \in \mathbb{R}^{n\times n}$, parameters $k \geq 1$, $\lambda > 0$, let* OPT *denote* $\min_{U'\in\mathbb{R}^{n\times k}, V'\in\mathbb{R}^{k\times n}} \|U'V' - A\|_1 + \lambda(\|U'\|_1 + \|V'\|_1)$*, there is an algorithm that runs in $\widetilde{O}(k)\cdot \mathrm{nnz}(A) + n\cdot \mathrm{poly}(k, \log n)$ time to output two matrices $U \in \mathbb{R}^{n\times k}$, $V \in \mathbb{R}^{k\times n}$ such that,*

$$\|UV - A\|_1 + \lambda(\|U\|_1 + \|V\|_1) \leq \widetilde{O}(k^2)\log^3 n \cdot \mathrm{OPT}$$

holds with probability at least $9/10$*.*

Notice that when $\lambda = 0$, all of our results exactly match the ℓ_1 low-rank approximation results shown by [SWZ17]. Since [SWZ17] also showed $k^{\Omega(1)}$ factor is necessary in the approximation ratio, our results are almost tight.

Algorithm 18.1 A Meta Algorithm

procedure L1RegularizedL1LowRankApprox(A,n,k,λ) ▷ Theorem 18.1.1 and 18.1.2
  Choose sketching matrices $S \in \mathbb{R}^{s\times n}, R \in \mathbb{R}^{n\times r}, T_1 \in \mathbb{R}^{t_1\times n}, T_2 \in \mathbb{R}^{n\times t_2}$ to be random Cauchy transformation matrices
  Compute SA and AR
  Compute SAT_2, T_1AR
  Compute T_1AT_2
  Solve $\min_{X,Y} \|T_1ARXYSAT_2 - T_1AT_2\|_F^2 + \lambda^2(\|T_1ARX\|_F^2 + \|YSAT_2\|_F^2)$
  $U \leftarrow ARX$, $V \leftarrow YSA$
  return U, V
end procedure

18.1.2 Technique overview

Sketching is used widely in many important numerical linear algebra problems such as overconstrained regression [CW13, DMMS11, MM13, NN13a, Sar06], low rank matrix

approximation [AM07, CW13, DV06, DKM06, MM13, NN13a, Sar06, CW15b, SWZ17, BWZ16] computation of leverage scores [CW13, DMIMW12, MM13, NN13a], regularized data fitting [ACW17]. Roughly speaking, the main idea of sketching is to highly compress the original data into a small size summary, and then to recover properties of the original data from the such small summary.

[SWZ17] proposed the first sketching based algorithm for solving ℓ_1 norm low rank matrix approximation problem. In that paper, they focused on the problem that given $A \in \mathbb{R}^{n \times n}, k > 0$, the goal is to solve

$$\min_{U \in \mathbb{R}^{n \times k}, V \in \mathbb{R}^{k \times n}} \|UV - A\|_1.$$

This problem is ℓ_1 norm low rank approximation problem, and it is a special case of our problem, i.e. when $\lambda = 0$. The main idea of [SWZ17] is as the following: for a given matrix $A \in \mathbb{R}^{n \times n}$, and a rank parameter k, they firstly chose two small random sketching matrices $S \in \mathbb{R}^{s \times n}$ and $R \in \mathbb{R}^{n \times r}$ and argued that there is a rank-k matrix B in the both small column span of AR and small row span of SA, and $\|B - A\|_1$ has a very good approximation ratio to the optimal cost, i.e. $\mathrm{poly}(k, \log n)$. Notice that B is in the both column space of AR and the row space of SA, it means that

$$\min_{X \in \mathbb{R}^{r \times k}, Y \in \mathbb{R}^{k \times s}} \|ARX \cdot YSA - A\|_1$$

is a good approximation of $\min_{\mathrm{rank}-k\ UV} \|UV - A\|_1$, and since the dimension of the column space of AR and the row space of SA are very small, then they further sketched down the problem by using oblivious ℓ_1 subspace embedding [MM13, SW11] or Lewis weights sampling [CP15]. Precisely, they chose additional two small random sketching matrices

$T_1 \in \mathbb{R}^{t_1 \times n}, T_2 \in \mathbb{R}^{n \times t_2}$, $(t_1, t_2 \leq \mathrm{poly}(k))$ and further sketched down the optimization problem to get a new minimization problem i.e.,

$$\min_{X \in \mathbb{R}^{r \times k}, Y \in \mathbb{R}^{k \times s}} \|T_1 AR \cdot X \cdot Y \cdot SAT_2 - T_1 AT_2\|_1.$$

Since the new optimization problem is very small, then it can be solved by relaxing to ℓ_2 minimization problem with losing mild additional $\mathrm{poly}(k)$ factors in approximation ratio or solved by using exponential running time optimizer with losing mild additional $2^{\mathrm{poly}(k)}$ running time, e.g. polynomial system verifier [Ren92a, Ren92b, BPR96, RSW16, SWZ17]. Now come back to our regularized ℓ_1 norm low rank approximation problem. We can find that we cannot extend the technique of [SWZ17] directly. Because our objective function is

$$\min_{\mathrm{rank}-k\ UV} \|UV - A\|_1 + \lambda(\|U\|_1 + \|V\|_1),$$

if λ is large, then the cost induced by $\|U\|_1 + \|V\|_1$ may dominate the cost induced by $\|UV - A\|_1$. Thus, the main issue we are facing is that it is not clear whether there is still a $\widehat{U}$ in the column space of AR such that

$$\min_{\mathrm{rank}-k\ V} \|\widehat{U}V - A\|_1 + \lambda(\|\widehat{U}\|_1 + \|V\|_1)$$

has a good approximation ratio to the optimal cost (same issue appears for V, i.e. we are not sure whether there is a good $\widehat{V}$ in the row space of SA.) If we want to sketch $\|UV - A\|_1$, $\|U\|_1$ and $\|V\|_1$ separately, we can not get a $\mathrm{poly}(k)$ size sketch, since the freedom of U and V itself is too large, i.e. we cannot restrict them on any small subspace. Thus, it motivated us to look at the cost induced by $\|UV - A\|_1$, $\|U\|_1$ and $\|V\|_1$ at the same time. Suppose U^* is the optimal solution of

$$\min_{\mathrm{rank}-k\ UV} \|UV - A\|_1 + \lambda(\|U\|_1 + \|V\|_1).$$

Then we have $\min_{\text{rank}-k\ V} \|U^*V - A\|_1 + \lambda(\|U^*\|_1 + \|V\|_1) = \text{OPT}$. Notice that it is actually an ℓ_1 regression problem:

$$\min_{\text{rank}-k\ V} \left\| \begin{bmatrix} U^* \\ \lambda I \end{bmatrix} V - \begin{bmatrix} A \\ 0 \end{bmatrix} \right\|_1 + \lambda \|U^*\|_1.$$

We choose a small random sketching matrix S, and we proved that a good approximation for

$$\min_{\text{rank}-k\ V} \left\| \begin{bmatrix} SU^* \\ \lambda I \end{bmatrix} V - \begin{bmatrix} SA \\ 0 \end{bmatrix} \right\|_1 \tag{18.6}$$

is a good approximation for

$$\min_{\text{rank}-k\ V} \left\| \begin{bmatrix} U^* \\ \lambda I \end{bmatrix} V - \begin{bmatrix} A \\ 0 \end{bmatrix} \right\|_1 .$$

Notice that since S has only $\text{poly}(k)$ number of rows, the number of rows of

$$\begin{bmatrix} SU^* \\ \lambda I \end{bmatrix} V - \begin{bmatrix} SA \\ 0 \end{bmatrix}$$

is at most $\text{poly}(k)$. Thus, we can relax optimization problem (18.6) to

$$\min_{\text{rank}-k\ V} \sum_{i=1}^{n} \left\| \begin{bmatrix} SU^* \\ \lambda I \end{bmatrix} V_i - \begin{bmatrix} SA \\ 0 \end{bmatrix}_i \right\|_2$$

with at most $\text{poly}(k)$ factor loss in approximation ratio. For the above optimization problem, we can get the optimal solution which has form

$$\widehat{V} = \begin{bmatrix} SU^* \\ \lambda I \end{bmatrix}^\dagger \cdot \begin{bmatrix} SA \\ 0 \end{bmatrix} = \begin{bmatrix} SU^* \\ \lambda I \end{bmatrix}^\dagger \cdot \begin{bmatrix} I \\ 0 \end{bmatrix} \cdot SA.$$

Thus, we can prove that

$$\min_{\text{rank}-k\ U} \|U\widehat{V} - A\|_1 + \lambda(\|U\|_1 + \|\widehat{V}\|_1).$$

is a good approximation to OPT . Then, we choose a small random sketching matrix R, we can apply the "iterative existential proof" [SWZ19b] argument to show that there is a matrix $\widehat{U}$ in the column space of AR such that

$$\|\widehat{U}\widehat{V} - A\|_1 + \lambda(\|\widehat{U}\|_1 + \|\widehat{V}\|_1)$$

is also a good approximation to OPT . Since $\widehat{U}$ is in the column space of AR and $\widehat{V}$ is in the row space of SA, we can get that

$$\min_{\mathrm{rank}-k\ XY} \|ARXYSA - A\|_1 + \lambda(\|ARX\|_1 + \|YSA\|_1)$$

is a good approximation to OPT . Because we shrink the solution space to a very small subspace, then we can use the oblivious subspace embedding [MM13, SW11] or Lewis weights sampling [CP15] to further choose two random sketching T_1, T_2 to sketch down the optimization problem to get

$$\min_{\mathrm{rank}-k\ XY} \|T_1ARXYSAT_2 - T_1AT_2\|_1 + \lambda(\|T_1ARX\|_1 + \|YSAT_2\|_1).$$

It is equivalent to solve

$$\min_{\mathrm{rank}-k\ XY} \left\| \begin{bmatrix} T_1ARX \\ \lambda I \end{bmatrix} \begin{bmatrix} YSAT_2 & \lambda I \end{bmatrix} - \begin{bmatrix} T_1AT_2 & 0 \\ 0 & \lambda^2 I \end{bmatrix} \right\|_1 .$$

To solve the above small problem, we can relax it into ℓ_2 version by losing $\mathrm{poly}(k)$ approximation factor or use some powerful optimizers by losing $2^{\mathrm{poly}(k)}$ running time. If we relax it into ℓ_2 version, then we need to solve

$$\min_{\mathrm{rank}-k\ XY} \left\| \begin{bmatrix} T_1ARX \\ \lambda I \end{bmatrix} \begin{bmatrix} YSAT_2 & \lambda I \end{bmatrix} - \begin{bmatrix} T_1AT_2 & 0 \\ 0 & \lambda^2 I \end{bmatrix} \right\|_F^2$$

which is

$$\min_{\text{rank}-k \ XY} \|T_1ARXYSAT_2 - T_1AT_2\|_F^2 + \lambda^2(\|T_1ARX\|_F^2 + \|YSAT_2\|_F^2).$$

It is exactly a generalized version of the Frobenius norm ridge low rank matrix approximation problem (see Section 18.2.1 for more details), and we can directly get the closed form of the optimal solution of this problem [ACW17, UHZ+16].

Surprisingly, we further showed that all the sketching matrices T_1, S, R, T_2 used in [SWZ17] satisfy the property we needed our problem. It means that we can actually fully generalize almost all the algorithms[1] in [SWZ17] to the ℓ_1 regularizer version.

[1]We can also generalize its CUR decomposition algorithms by using our techniques, but it is not the main focus in this work.

18.2 Preliminaries

In this Section, we mainly introduce the notations and some background knowledge.

For an $n \in \mathbb{N}_{\geq 0}$, let $[n]$ denote the set $\{1, 2, \cdots, n\}$.

For any function f, we define $\widetilde{O}(f)$ to be $f \cdot \log^{O(1)}(f)$. In addition to $O(\cdot)$ notation, for two functions f, g, we use the shorthand $f \lesssim g$ (resp. $\gtrsim$) to indicate that $f \leq Cg$ (resp. $\geq$) for an absolute constant C. We use $f \eqsim g$ to mean $cf \leq g \leq Cf$ for constants c, C.

Given a matrix $A \in \mathbb{R}^{n \times d}$, we say the set $\{y \mid x \in \mathbb{R}^d, y = Ax\}$ is the column space of A. Similarly, we say the set $\{y \mid x \in \mathbb{R}^n, y = x^\top A\}$ is the row space of A.

Let $\mathrm{nnz}(A)$ denote the number of nonzero entries of A. Let $\det(A)$ denote the determinant of a square matrix A. Let $A^\top$ denote the transpose of A. Let $A^\dagger$ denote the Moore-Penrose pseudoinverse of A. Let A^{-1} denote the inverse of a full rank square matrix.

Let $\|A\|_F$ denote the Frobenius norm of matrix A and $\|A\|_1$ denote the entry-wise 1-norm of matrix A.

18.2.1 Frobenius norm ridge low-rank approximation and ℓ_2 relaxation

In this Section, we introduce a highly related problem called Frobenius norm ridge low rank approximation problem. It is just an ℓ_2 variation of the problem we studied in this work:

$$\min_{U \in \mathbb{R}^{n \times k}, V \in \mathbb{R}^{k \times n}} \|UV - A\|_F^2 + \lambda(\|U\|_F^2 + \|V\|_F^2). \tag{18.7}$$

In [ACW17], they have an efficient algorithm for more generalized problem:

Theorem 18.2.1 (Lemma 27 in [ACW17]). *For $T_1 \in \mathbb{R}^{t_1 \times n}$, $T_2 \in \mathbb{R}^{n \times t_2}$, $B \in \mathbb{R}^{t_1 \times t_2}$, the problem of finding*

$$\min_{\widehat{X} \in \mathbb{R}^{n \times k}, \widehat{Y} \in \mathbb{R}^{k \times n}} \|T_1 \widehat{X} \widehat{Y} T_2 - B\|_F^2 + \lambda(\|T_1 \widehat{X}\|_F^2 + \|\widehat{Y} T_2\|_F^2)$$

and the minimizing $T_1 \widehat{X} \in \mathbb{R}^{t_1 \times k}$ and $\widehat{Y} T_2 \in \mathbb{R}^{k \times t_2}$, can be solved in $O(t_1 n \operatorname{rank}(T_1) + t_2 n \operatorname{rank}(T_2) + \operatorname{rank}(T_2) t_1 (t_2 + \operatorname{rank}(T_1)))$ time.

The following Claim shows the relation between the solution of ℓ_2 regression and the solution of ℓ_1 regression.

Claim 18.2.2 (ℓ_2 relaxation). *Let $A \in \mathbb{R}^{n \times d}$. We have $\|A\|_F \leq \|A\|_1 \leq \sqrt{nd} \|A\|_F$.*

Roughly speaking, our algorithms firstly sketch down the size of the input, and then relax the original problem to ℓ_2 case. The above lemma and the claim are fundamental building blocks in the reduction from the original problem to the ℓ_2 relaxation.

18.2.2 Cauchy transforms

In this section, we introduce the main sketching tool used in this work: the Cauchy transformation. We mainly introduce two kinds of Cauchy transformations: Dense Cauchy transform and Sparse Cauchy transform where each of them is introduced by [SW11] and [MM13] separately.

Definition 18.2.1 (Dense Cauchy transform [SW11]). Let $S = \sigma \cdot C \in \mathbb{R}^{m \times n}$ where σ is a scalar, and each entry of $C \in \mathbb{R}^{m \times n}$ is chosen independently from the standard Cauchy distribution. S is called Dense Cauchy transform matrix.

It is easy to see that for any $A \in \mathbb{R}^{n \times d}$, SA can be computed in $O(m \cdot \mathrm{nnz}(A))$ time [SW11].

Definition 18.2.2 (Sparse Cauchy transform [MM13])**.** Let $\Pi = \sigma \cdot SC \in \mathbb{R}^{m \times n}$, where σ is a scalar, $S \in \mathbb{R}^{m \times n}$ has each column chosen independently and uniformly from the m standard basis vectors of $\mathbb{R}^m$, and $C \in \mathbb{R}^{n \times n}$ is a diagonal matrix with diagonals chosen independently from the standard Cauchy distribution. Π is called Sparse Cauchy transform matrix.

For any matrix $A \in \mathbb{R}^{n \times d}$, ΠA can be computed in $O(\mathrm{nnz}(A))$ time. For any matrix $A \in \mathbb{R}^{n \times d_1 \times d_2}$, ΠA can be computed in $O(\mathrm{nnz}(A))$ time [MM13].

The properties of Cauchy transformation needed in this work is the following Lemma:

Lemma 18.2.3 (Lemma D.11 in [SWZ17])**.** *Given $M \in \mathbb{R}^{n \times d}, U \in \mathbb{R}^{n \times t}, d \geq t = \mathrm{rank}(U), n \geq d \geq r = \mathrm{rank}(M)$, if sketching matrix $S \in \mathbb{R}^{m \times n}$ is drawn from any of the following probability distributions of matrices, with .99 probability S satisfies the following properties:*

1. $\|SM\|_1 \leq c_1 \|M\|_1$,
2. $\forall x \in \mathbb{R}^t, \|SUx\|_1 \geq \frac{1}{c_2} \|Ux\|_1$,

where c_1, c_2 are parameters which depend on the distribution over S.

- *$S \in \mathbb{R}^{m \times n}$ is a dense Cauchy transform matrix: a matrix with i.i.d. entries drawn from the standard Cauchy distribution. If $m = O(t \log t)$, then $c_1 c_2 = O(\log d)$. If $m = O((t + r) \log(t + r))$, then $c_1 c_2 = O(\min(\log d, r \log r))$.*

- $S \in \mathbb{R}^{m \times n}$ *is a sparse Cauchy transform matrix:* $S = TD$*, where* $T \in \mathbb{R}^{m \times n}$ *has each column i.i.d. from the uniform distribution on standard basis vector of* $\mathbb{R}^m$*, and* $D \in \mathbb{R}^{n \times n}$ *is a diagonal matrix with i.i.d. diagonal entries from standard Cauchy distribution. If* $m = O(t^5 \log^5 t)$*, then* $c_1 c_2 = O(t^2 \log^2 t \log d)$*. If* $m = O((t+r)^5 \log^5(t+r))$*, then* $c_1 c_2 = O(t^2 \log^2 t \log d, r^3 \log^3 r)$*.*

In the above, if we replace S *with* $\sigma \cdot S$ *where* $\sigma \in \mathbb{R} \setminus \{0\}$ *is any scalar, then the relation between* m *and* $c_1 c_2$ *is still preserved.*

18.3 The Algorithm

Our algorithm is shown in Algorithm 18.1. If we choose all the sketching matrices S, R, T_1, T_2 as dense Cauchy transformation matrices (see Definition 18.2.1), then we are able to finally get Theorem 18.1.2. If we choose all the sketching matrices S, R, T_1, T_2 as sparse Cauchy transformation matrices (see Definition 18.2.2), then we are able to finally get Theorem 18.1.1.

For convenience, we define $\mathrm{OPT}(A, k, \lambda)$ as follows

$$\mathrm{OPT}(A, k, \lambda) = \min_{U \in \mathbb{R}^{n \times k}, V \in \mathbb{R}^{k \times n}} \|UV - A\|_1 + \lambda(\|U\|_1 + \|V\|_1).$$

If A, k, λ is clear from the context, we just use OPT to denote $\mathrm{OPT}(A, k, \lambda)$.

Then we present the analysis of our algorithm in the following way:

1. In Section 18.3.1, we show how to reduce the original problem to the following problem,

$$\min_{X \in \mathbb{R}^{r \times k}, Y \in \mathbb{R}^{k \times s}} \|ARXYSA - A\|_1 + \lambda(\|ARX\|_1 + \|YSA\|_1).$$

2. In Section 18.3.2, we show how to further reduce the above problem to:

$$\min_{X \in \mathbb{R}^{r \times k}, Y \in \mathbb{R}^{k \times s}} \|T_1 ARXYSAT_2 - T_1 AT_2\|_1 + \lambda(\|T_1 ARX\|_1 + \|YSAT_2\|_1).$$

3. Finally, in Section 18.3.3, we show how to solve the above problem with a small approximation ratio.

18.3.1 Sketching Lemmas

In this section, we show how to use sketching techniques to prove Corollary 18.3.2 which means that there is a good approximate solution whose columns are in the column space of AR and rows are in the row space of SA where S, A are two sketching matrices (shown in the algorithm 18.1) which satisfy desired properties. Before proving Corollary 18.3.2, we first prove the following lemma which is a "one-side" version of the corollary: it shows that there is a good approximate solution in the row space of SA.

Lemma 18.3.1. (Existence of a good solution in a small column subspace)**.** Given matrix $A \in \mathbb{R}^{n\times n}$, $\alpha \geq 1, \beta \geq 1, \gamma \geq 1, \lambda > 0$ and $k \geq 1$. Let $\widehat{U} \in \mathbb{R}^{n\times k}$ satisfy

$$\min_{V\in\mathbb{R}^{k\times n}} \|\widehat{U}V - A\|_1 + \lambda(\|\widehat{U}\|_1 + \|V\|_1) \leq \gamma\,\mathrm{OPT}\,.$$

Let $V^* \in \mathbb{R}^{k\times n}$ denote the optimal solution of

$$\min_{V\in\mathbb{R}^{k\times n}} \|\widehat{U}V - A\|_1 + \lambda(\|\widehat{U}\|_1 + \|V\|_1).$$

Let matrix $S \in \mathbb{R}^{m\times n}$ satisfy:

$$\|S\widehat{U}x\|_1 \geq \frac{1}{\alpha}\|\widehat{U}x\|_1 \text{ for all } x \in \mathbb{R}^k,$$
$$\text{and } \|S\widehat{U}V^* - SA\|_1 \leq \beta\|\widehat{U}V^* - A\|_1.$$

Let $\widehat{V} \in \mathbb{R}^{k\times n}$ denote $\begin{bmatrix} S\widehat{U} \\ \lambda I \end{bmatrix}^\dagger \cdot \begin{bmatrix} I \\ 0 \end{bmatrix} \cdot SA$. Then

$$\|\widehat{U}\widehat{V} - A\|_1 + \lambda(\|\widehat{U}\|_1 + \|\widehat{V}\|_1) \leq \alpha\beta\gamma(\sqrt{m+k}+2)\,\mathrm{OPT}\,.$$

Proof. Due to the space limitation, we provide the proofs in Appendix 18.4. □

Corollary 18.3.2 (Existence of a good solution in both row and column subspace)**.** *Given matrix $A \in \mathbb{R}^{n\times n}$, $\alpha_1, \beta_1, \alpha_2, \beta_2 \geq 1, \lambda > 0$ and $k \geq 1$. Let $U^* \in \mathbb{R}^{n\times k}, V^* \in \mathbb{R}^{k\times n}$ denote the optimal solution of the problem,*

$$\min_{U,V} \|UV - A\|_1 + \lambda(\|U\|_1 + \|V\|_1).$$

Let matrix $S \in \mathbb{R}^{s\times n}$ satisfy :

$$\begin{aligned} &\|SU^*x\|_1 \geq \frac{1}{\alpha_1}\|U^*x\|_1 \text{ for all } x \in \mathbb{R}^k \\ &\text{and } \|SU^*V^* - SA\|_1 \leq \beta_1\|U^*V^* - A\|_1. \end{aligned}$$

Let $\widehat{V} \in \mathbb{R}^{k\times n}$ denote $\begin{bmatrix} SU^* \\ \lambda I \end{bmatrix}^\dagger \cdot \begin{bmatrix} I \\ 0 \end{bmatrix} \cdot SA$. Let $\widetilde{U} \in \mathbb{R}^{n\times k}$ denote the optimal solution of

$$\min_{U\in\mathbb{R}^{n\times k}} \|U\widehat{V} - A\|_1 + \lambda(\|U\|_1 + \|\widehat{V}\|_1).$$

Let $R^\top \in \mathbb{R}^{r\times n}$ satisfy:

$$\begin{aligned} &\|R^\top\widehat{V}^\top x\|_1 \geq \frac{1}{\alpha_2}\|\widehat{V}^\top x\|_1 \text{ for all } x \in \mathbb{R}^k, \\ &\text{and } \|R^\top\widehat{V}^\top\widetilde{U}^\top - R^\top A^\top\|_1 \leq \beta_2\|\widehat{V}^\top\widetilde{U}^\top - A^\top\|_1. \end{aligned}$$

Let $\widehat{U}^\top \in \mathbb{R}^{k\times n}$ denote $\begin{bmatrix} R^\top\widehat{V}^\top \\ \lambda I \end{bmatrix}^\dagger \cdot \begin{bmatrix} I \\ 0 \end{bmatrix} \cdot R^\top A^\top$. Then

$$\begin{aligned} &\|\widehat{U}\widehat{V} - A\|_1 + \lambda(\|\widehat{U}\|_1 + \|\widehat{V}\|_1) \\ &\leq 4(\alpha_1\beta_1\alpha_2\beta_2)(\sqrt{s+k}+2)(\sqrt{r+k}+2)\,\mathrm{OPT}, \end{aligned}$$

and there exists $X \in \mathbb{R}^{r\times k}, Y \in \mathbb{R}^{k\times s}$ such that

$$\|ARXYSA - A\|_1 + \lambda(\|ARX\|_1 + \|YSA\|_1) \leq \|\widehat{U}\widehat{V} - A\|_1 + \lambda(\|\widehat{U}\|_1 + \|\widehat{V}\|_1).$$

Proof. It is actually applying Lemma 18.3.1 twice. Due to the property of sketching matrix S, we can apply Lemma 18.3.1 to get

$$\begin{aligned}
&\min_{U\in\mathbb{R}^{n\times k}} \|U\widehat{V} - A\|_1 + \lambda(\|U\|_1 + \|\widehat{V}\|_1) \\
\leq\ & \|U^*\widehat{V} - A\|_1 + \lambda(\|U^*\|_1 + \|\widehat{V}\|_1) \\
\leq\ & 2\alpha_1\beta_1(\sqrt{s+k}+2)\,\mathrm{OPT}\,.
\end{aligned}$$

We set $\gamma = 2\alpha_1\beta_1(\sqrt{s+k}+2)$. Due to the property of sketching matrix R, we can apply Lemma 18.3.1 again to get

$$\begin{aligned}
&\|\widehat{U}\widehat{V} - A\|_1 + \lambda(\|\widehat{U}\|_1 + \|\widehat{V}\|_1) \\
\leq\ & 2\alpha_2\beta_2\gamma(\sqrt{r+k}+2)\,\mathrm{OPT} \\
=\ & 4\alpha_1\beta_1\alpha_2\beta_2(\sqrt{s+k}+2)(\sqrt{r+k}+2)\,\mathrm{OPT}\,.
\end{aligned}$$

Notice that $\widehat{U}$ is in the column space of AR, $\widehat{V}$ is in the row space of SA, thus,

$$\begin{aligned}
&\min_{X\in\mathbb{R}^{r\times k}, Y\in\mathbb{R}^{k\times s}} \|ARXYSA - A\|_1 \\
&\quad + \lambda(\|ARX\|_1 + \|YSA\|_1) \\
\leq\ & \|\widehat{U}\widehat{V} - A\|_1 + \lambda(\|\widehat{U}\|_1 + \|\widehat{V}\|_1) \\
\leq\ & 4\alpha_1\beta_1\alpha_2\beta_2(\sqrt{s+k}+2)(\sqrt{r+k}+2)\,\mathrm{OPT}\,.
\end{aligned}$$

□

The above corollary shows that if we can find a good approximate solution $\widehat{X}, \widehat{Y}$ to

$$\min_{X\in\mathbb{R}^{r\times k}, Y\in\mathbb{R}^{k\times s}} \|ARXYSA - A\|_1 + \lambda(\|ARX\|_1 + \|YSA\|_1),$$

then we can have a good approximation $U = AR\widehat{X}, V = \widehat{Y}SA$ to the original problem.

18.3.2 Reducing to small problems

In the last section, we already restrict the solution in a low dimensional column space and in a low dimensional row space. But the problem size is till too large since the size of A is still n-by-n. In this section, we further sketch down the size of the input. The Corollary 18.3.4 shows how to further reduce the problem to solving

$$\min_{X\in\mathbb{R}^{r\times k},Y\in\mathbb{R}^{k\times s}} \|T_1ARXYSAT_2 - T_1AT_2\|_1 + \lambda(\|T_1ARX\|_1 + \|YSAT_2\|_1)$$

where T_1, T_2 are two sketching matrices with some desired properties. Before proving Corollary 18.3.4, let us look at the following lemma which provides a "one-side" sketch version of that corollary:

Lemma 18.3.3. (Reducing the problem size by sketching on one side)**.** Given three matrices $L \in \mathbb{R}^{d_1\times l_1}$, $M^\top \in \mathbb{R}^{d_2\times l_2}$, $N \in \mathbb{R}^{d_1\times d_2}$ and $\alpha \geq 1, \beta \geq 1, \gamma \geq 1, \lambda > 0$. Let $\mathrm{OPT} = \min_{X,Y} \|LXYM - N\|_1 + \lambda(\|LX\|_1 + \|YM\|_1)$. Let X^*, Y^* denote the optimal solution of

$$\min_{X,Y} \|LXYM - N\|_1 + \lambda(\|LX\|_1 + \|YM\|_1).$$

Let $T \in \mathbb{R}^{t\times d_1}$ satisfy :

$$\|TLx\|_1 \geq \frac{1}{\alpha}\|Lx\|_1, \text{ for all } x \in \mathbb{R}^{l_1}$$

and

$$\|TLX^*Y^*M - TN\|_1 + \lambda \cdot \|TLX^*\|_1$$
$$\leq \beta(\|LX^*Y^*M - N\|_1 + \lambda \cdot \|LX^*\|_1)$$

Let $\widehat{X}, \widehat{Y}$ satisfy

$$\begin{aligned}&\|TL\widehat{X}\widehat{Y}M - TN\|_1 + \lambda \cdot (\|TL\widehat{X}\|_1 + \|\widehat{Y}M\|_1) \\ \leq\ & \gamma \min_{X,Y} \|TLXYM - TN\|_1 + \lambda \cdot (\|TLX\|_1 + \|YM\|_1)\end{aligned}$$

Then

$$\begin{aligned}&\|L\widehat{X}\widehat{Y}M - N\|_1 + \lambda(\|L\widehat{X}\|_1 + \|\widehat{Y}M\|_1) \\ \leq\ & (\alpha\beta\gamma + \alpha\beta + 1)\,\mathrm{OPT}\,.\end{aligned}$$

Proof. Due to space limitation, we provide the proofs in Appendix 18.5. □

Corollary 18.3.4 (Reducing the problem size by sketching on both sides)**.** *Given three matrices* $L \in \mathbb{R}^{d_1 \times l_1}$, $M^\top \in \mathbb{R}^{d_2 \times l_2}$, $N \in \mathbb{R}^{d_1 \times d_2}$ *and* $\alpha_1 \geq 1, \alpha_2 \geq 1, \beta_1 \geq 1, \beta_2 \geq 1, \gamma \geq 1, \lambda > 0$. *Let* $\mathrm{OPT} = \min_{X,Y} \|LXYM - N\|_1 + \lambda(\|LX\|_1 + \|YM\|_1)$. *Let* X^*, Y^* *denote the optimal solution of*

$$\min_{X,Y} \|LXYM - N\|_1 + \lambda(\|LX\|_1 + \|YM\|_1).$$

Let $T_1 \in \mathbb{R}^{t_1 \times d_1}$ *satisfy : for all* $x \in \mathbb{R}^{l_1}$

$$\|T_1 Lx\|_1 \geq \frac{1}{\alpha_1}\|Lx\|_1$$

and

$$\begin{aligned}&\|T_1 LX^*Y^*M - T_1 N\|_1 + \lambda \cdot \|T_1 LX^*\|_1 \\ \leq\ & \beta_1(\|LX^*Y^*M - N\|_1 + \lambda \cdot \|LX^*\|_1)\end{aligned}$$

Let $\widehat{X}, \widehat{Y}$ the optimal solution

$$\min_{X,Y} \|T_1 L X Y M - T_1 N\|_1 + \lambda \cdot (\|T_1 L X\|_1 + \|Y M\|_1).$$

Let $T_2^\top \in \mathbb{R}^{t_2 \times d_2}$ satisfy : for all $x \in \mathbb{R}^{l_2}$

$$\|T_2^\top M^\top x\|_1 \geq \frac{1}{\alpha_2} \|M^\top x\|_1$$

and

$$\begin{aligned} &\|T_1 L \widehat{X} \widehat{Y} M T_2 - T_1 N T_2\|_1 + \lambda \cdot \|\widehat{Y} M T_2\|_1 \\ \leq\ & \beta_2 (\|T_1 L \widehat{X} \widehat{Y} M - T_1 N\|_1 + \lambda \cdot \|\widehat{Y} M\|_1) \end{aligned}$$

Let $\widetilde{X}, \widetilde{Y}$ satisfy

$$\begin{aligned} &\|T_1 L \widetilde{X} \widetilde{Y} M L_2 - T_1 N T_2\|_1 + \lambda (\|T_1 L \widetilde{X}\|_1 + \|M \widetilde{Y} T_2\|_1) \\ \leq\ & \gamma \min_{X,Y} \|T_1 L X Y M L_2 - T_1 N T_2\|_1 + \lambda (\|T_1 L X\|_1 + \|M Y T_2\|_1) \end{aligned}$$

Then

$$\begin{aligned} &\|L \widetilde{X} \widetilde{Y} M - N\|_1 + \lambda (\|L \widetilde{X}\|_1 + \|M \widetilde{Y}\|_1) \\ \leq\ & 9 \alpha_1 \alpha_2 \beta_1 \beta_2 \gamma \, \mathrm{OPT} \, . \end{aligned}$$

Proof. This follows by applying Lemma 18.3.3 twice. Due to the property of T_2 and the definition of $\widetilde{X}, \widetilde{Y}$, by applying Lemma 18.3.3, we have that

$$\begin{aligned} &\|T_1 L \widetilde{X} \widetilde{Y} M - T_1 N\|_1 + \lambda \cdot (\|T_1 L \widetilde{X}\|_1 + \|\widetilde{Y} M\|_1) \\ \leq\ & (\alpha_2 \beta_2 \gamma + \alpha_2 \beta_2 + 1) \Big(\|T_1 L \widehat{X} \widehat{Y} M - T_1 N\|_1 + \lambda \cdot (\|T_1 L \widehat{X}\|_1 + \|\widehat{Y} M\|_1) \Big). \end{aligned}$$

Now we set the new $\gamma' = (\alpha_2\beta_2\gamma + \alpha_2\beta_2 + 1)$. Then, due to the property of T_1 and the definition of $\widehat{X}, \widehat{Y}$, we can apply Lemma 18.3.3 again, we then have

$$\begin{aligned} & \|L\widetilde{X}\widetilde{Y}M - N\|_1 + \lambda(\|L\widetilde{X}\|_1 + \|M\widetilde{Y}\|_1) \\ \leq & (\alpha_1\beta_1\gamma' + \alpha_1\beta_1 + 1)\,\mathrm{OPT}\,. \end{aligned}$$

Since $\alpha_1, \alpha_2, \beta_1, \beta_2 \geq 1$, we then have $(\alpha_1\beta_1\gamma' + \alpha_1\beta_1 + 1) \leq 3\alpha_1\beta_1\gamma' \leq 9\alpha_1\beta_1\alpha_2\beta_2\gamma$. □

Let us set $L = AR, M = SA, N = A$, then due to the above corollary, we show that once we get a good approximation $\widetilde{X}, \widetilde{Y}$ to $\min_{X,Y} \|T_1ARXYSAT_2 - T_1AT_2\|_1 + \lambda(\|T_1ARX\|_1 + \|YSAT_2\|_1)$, we get a good approximation to $\min_{X,Y} \|ARXYSA - A\|_1 + \lambda(\|ARX\|_1 + \|YSA\|_1)$. Since the number of rows of T_1 is small and the number of columns of T_2 is also small, the problem size becomes very small. In the next section, we will see how to use the ℓ_2 relaxation to solve the small problem.

18.3.3 Solving small problems

In this section, we focus on solving

$$\min_{X,Y} \|T_1ARXYSAT_2 - T_1AT_2\|_1 + \lambda(\|T_1ARX\|_1 + \|YSAT_2\|_1).$$

Firstly, we have the following easy fact.

Fact 18.3.5. *Given five matrices L, X, Y, M and N with matching dimensions, we have*

$$\begin{aligned} & \|LXYM - N\|_1 + \lambda(\|LX\|_1 + \|YM\|_1) \\ = & \left\| \begin{bmatrix} LX \\ \lambda I \end{bmatrix} \cdot \begin{bmatrix} YM & \lambda I \end{bmatrix} - \begin{bmatrix} N & 0 \\ 0 & \lambda^2 I \end{bmatrix} \right\|_1 . \end{aligned}$$

We define $f(X,Y)$ as follows

$$f(X,Y) = \begin{bmatrix} T_1 ARX \\ \lambda I \end{bmatrix} \cdot \begin{bmatrix} YSAT_2 & \lambda I \end{bmatrix} - \begin{bmatrix} T_1 AT_2 & 0 \\ 0 & \lambda^2 I \end{bmatrix}. \tag{18.8}$$

Due to the above fact, we can relax the ℓ_1-norm minimization problem

$$\min_{X \in \mathbb{R}^{r \times k}, Y \in \mathbb{R}^{k \times s}} \|f(X,Y)\|_1 \tag{18.9}$$

to Frobenius norm minimization problem

$$\min_{X \in \mathbb{R}^{r \times k}, Y \in \mathbb{R}^{k \times s}} \|f(X,Y)\|_F ,$$

which is equivalent to

$$\min_{X \in \mathbb{R}^{r \times k}, Y \in \mathbb{R}^{k \times s}} \|f(X,Y)\|_F^2 . \tag{18.10}$$

Lemma 18.3.6 (Relaxing ℓ_1 norm to Frobenius norm)**.** *Given $A \in \mathbb{R}^{n \times n}$, $T_1 \in \mathbb{R}^{t_1 \times n}$, $T_2 \in \mathbb{R}^{n \times t_2}$, $R \in \mathbb{R}^{n \times r}$ and $S \in \mathbb{R}^{s \times n}$. Let $f_{T_1,T_2,R,S,A,\lambda}(X,Y) \in \mathbb{R}^{(t_1+k)\times(t_2+k)}$ be defined in Eq. (18.8), if X', Y' satisfies that*

$$\|f(X',Y')\|_F^2 \leq (1+\epsilon) \min_{X,Y} \|f(X,Y)\|_F^2$$

for some $\epsilon > 0$, then we have

$$\|f(X',Y')\|_1 \leq \sqrt{t_1+k} \cdot \sqrt{t_2+k} \cdot \sqrt{1+\epsilon} \min_{X,Y} \|f(X,Y)\|_1.$$

Proof. Let X^*, Y^* satisfy $\|f(X^*, Y^*)\|_1 = \min_{X,Y} \|f(X,Y)\|_1$. We have:

$$\begin{aligned}
& \|f(X', Y')\|_1 \\
\leq & \sqrt{t_1 + k} \cdot \sqrt{t_2 + k} \cdot \|f(X', Y')\|_F \\
\leq & \sqrt{t_1 + k} \cdot \sqrt{t_2 + k} \cdot \sqrt{1+\epsilon} \cdot \min_{X,Y} \|f(X,Y)\|_F \\
\leq & \sqrt{t_1 + k} \cdot \sqrt{t_2 + k} \cdot \sqrt{1+\epsilon} \cdot \|f(X^*, Y^*)\|_F \\
\leq & \sqrt{t_1 + k} \cdot \sqrt{t_2 + k} \cdot \sqrt{1+\epsilon} \cdot \|f(X^*, Y^*)\|_1
\end{aligned}$$

where the first inequality and the last inequality follow the Claim 18.2.2. □

Therefore, we can reduce the entry-wise ℓ_1 problem (18.9) to Frobenius norm problem (18.10).

Using the Theorem 18.2.1, we can solve the Frobenius norm problem (18.10). Now, we are ready to present our main results.

Theorem 18.3.7 (Restatement of Theorem 18.1.1)**.** *Given matrix $A \in \mathbb{R}^{n \times n}$, parameters $k \geq 1$, $\lambda > 0$, let* OPT *denote* $\min_{U' \in \mathbb{R}^{n \times k}, V' \in \mathbb{R}^{k \times n}} \|U'V' - A\|_1 + \lambda(\|U'\|_1 + \|V'\|_1)$*, there is an algorithm that runs in $O(\mathrm{nnz}(A) + n\,\mathrm{poly}(k, \log n))$ time to output two matrices $U \in \mathbb{R}^{n \times k}$, $V \in \mathbb{R}^{k \times n}$ such that,*

$$\|UV - A\|_1 + \lambda(\|U\|_1 + \|V\|_1) \leq \mathrm{poly}(k, \log n) \cdot \mathrm{OPT},$$

holds with probability at least $9/10$.

Proof. Due to the space limits, we put the whole proof into Appendix 18.6. □

Theorem 18.3.8 (Restatement of Theorem 18.1.2)**.** *Given matrix $A \in \mathbb{R}^{n \times n}$, parameters $k \geq 1$, $\lambda > 0$, let* OPT *denote* $\min_{U' \in \mathbb{R}^{n \times k}, V' \in \mathbb{R}^{k \times n}} \|U'V' - A\|_1 + \lambda(\|U'\|_1 + \|V'\|_1)$*, there is an algorithm that runs in* $\widetilde{O}(k) \cdot \mathrm{nnz}(A) + n \cdot \mathrm{poly}(k, \log n)$ *time to output two matrices $U \in \mathbb{R}^{n \times k}$, $V \in \mathbb{R}^{k \times n}$ such that,*

$$\|UV - A\|_1 + \lambda(\|U\|_1 + \|V\|_1) \leq \widetilde{O}(k^2) \log^3 n \cdot \mathrm{OPT},$$

holds with probability at least $9/10$*.*

Proof. Due to the space limits, we put the whole proof into Appendix 18.7. □

Remark 18.3.1. Combining our new technique with p-stable transform (e.g. Section E of [SWZ17] and [SWZ19b]), all of our results can be extended to ℓ_p-norm, where $1 < p < 2$.

18.4 Proof of Lemma 18.3.1

Lemma 18.3.1. (Existence of a good solution in a small column subspace)**.** Given matrix $A \in \mathbb{R}^{n\times n}$, $\alpha \geq 1, \beta \geq 1, \gamma \geq 1, \lambda > 0$ and $k \geq 1$. Let $\widehat{U} \in \mathbb{R}^{n\times k}$ satisfy

$$\min_{V\in\mathbb{R}^{k\times n}} \|\widehat{U}V - A\|_1 + \lambda(\|\widehat{U}\|_1 + \|V\|_1) \leq \gamma\,\mathrm{OPT}\,.$$

Let $V^* \in \mathbb{R}^{k\times n}$ denote the optimal solution of

$$\min_{V\in\mathbb{R}^{k\times n}} \|\widehat{U}V - A\|_1 + \lambda(\|\widehat{U}\|_1 + \|V\|_1).$$

Let matrix $S \in \mathbb{R}^{m\times n}$ satisfy:

$$\begin{aligned} &\|S\widehat{U}x\|_1 \geq \frac{1}{\alpha}\|\widehat{U}x\|_1 \text{ for all } x \in \mathbb{R}^k, \\ &\text{and } \|S\widehat{U}V^* - SA\|_1 \leq \beta\|\widehat{U}V^* - A\|_1. \end{aligned}$$

Let $\widehat{V} \in \mathbb{R}^{k\times n}$ denote $\begin{bmatrix} S\widehat{U} \\ \lambda I \end{bmatrix}^\dagger \cdot \begin{bmatrix} I \\ 0 \end{bmatrix} \cdot SA$. Then

$$\|\widehat{U}\widehat{V} - A\|_1 + \lambda(\|\widehat{U}\|_1 + \|\widehat{V}\|_1) \leq \alpha\beta\gamma(\sqrt{m+k}+2)\,\mathrm{OPT}\,.$$

Proof. We can show the following Claim,

Claim 18.4.1.

$$\|\widehat{U}\widehat{V} - A\|_1 + \lambda(\|\widehat{U}\|_1 + \|\widehat{V}\|_1) \leq \alpha \left\| \begin{bmatrix} S\widehat{U} \\ \lambda I \end{bmatrix} \widehat{V} - \begin{bmatrix} SA \\ 0 \end{bmatrix} \right\|_1 + \alpha \left\| \begin{bmatrix} S\widehat{U} \\ \lambda I \end{bmatrix} V^* - \begin{bmatrix} SA \\ 0 \end{bmatrix} \right\|_1 + \gamma\,\mathrm{OPT}\,.$$

Proof.

$$
\begin{aligned}
& \|\widehat{U}\widehat{V} - A\|_1 + \lambda(\|\widehat{U}\|_1 + \|\widehat{V}\|_1) \\
= & \left\| \begin{bmatrix} \widehat{U} \\ \lambda I \end{bmatrix} \widehat{V} - \begin{bmatrix} A \\ 0 \end{bmatrix} \right\|_1 + \lambda \|\widehat{U}\|_1 \\
\leq & \left\| \begin{bmatrix} \widehat{U} \\ \lambda I \end{bmatrix} (\widehat{V} - V^*) \right\|_1 + \left\| \begin{bmatrix} \widehat{U} \\ \lambda I \end{bmatrix} V^* - \begin{bmatrix} A \\ 0 \end{bmatrix} \right\|_1 + \lambda \|\widehat{U}\|_1 \\
\leq & \left\| \begin{bmatrix} \widehat{U} \\ \lambda I \end{bmatrix} (\widehat{V} - V^*) \right\|_1 + \gamma \, \mathrm{OPT} \\
= & \|\widehat{U}(\widehat{V} - V^*)\|_1 + \lambda \|\widehat{V} - V^*\|_1 + \gamma \, \mathrm{OPT} \\
\leq & \alpha \|S\widehat{U}(\widehat{V} - V^*)\|_1 + \lambda \|\widehat{V} - V^*\|_1 + \gamma \, \mathrm{OPT} \\
\leq & \alpha \|S\widehat{U}(\widehat{V} - V^*)\|_1 + \lambda\alpha \|\widehat{V} - V^*\|_1 + \gamma \, \mathrm{OPT} \\
= & \alpha \left\| \begin{bmatrix} S\widehat{U} \\ \lambda I \end{bmatrix} (\widehat{V} - V^*) \right\|_1 + \gamma \, \mathrm{OPT} \\
\leq & \alpha \left\| \begin{bmatrix} S\widehat{U} \\ \lambda I \end{bmatrix} \widehat{V} - \begin{bmatrix} SA \\ 0 \end{bmatrix} \right\|_1 + \alpha \left\| \begin{bmatrix} S\widehat{U} \\ \lambda I \end{bmatrix} V^* - \begin{bmatrix} SA \\ 0 \end{bmatrix} \right\|_1 + \gamma \, \mathrm{OPT}
\end{aligned}
$$

where the second step follows by triangle inequality, the third step follows by

$$
\left\| \begin{bmatrix} \widehat{U} \\ \lambda I \end{bmatrix} V^* - \begin{bmatrix} A \\ 0 \end{bmatrix} \right\|_1 + \lambda \|\widehat{U}\|_1 \leq \gamma \, \mathrm{OPT},
$$

the fifth step follows by $\|\widehat{U}(\widehat{V} - V^*)\|_1 \leq \alpha \|S\widehat{U}(\widehat{V} - V^*)\|_1$, the sixth step follows by $\alpha \geq 1$, the eighth step follows by triangle inequality.

□

We now show how to bound the second term on RHS of the equation shown in Claim 18.4.1.

Claim 18.4.2.

$$\alpha \left\| \begin{bmatrix} S\widehat{U} \\ \lambda I \end{bmatrix} V^* - \begin{bmatrix} SA \\ 0 \end{bmatrix} \right\|_1 \leq \alpha\beta\gamma \, \mathrm{OPT} \, .$$

Proof. We have

$$\begin{aligned} & \alpha \left\| \begin{bmatrix} S\widehat{U} \\ \lambda I \end{bmatrix} V^* - \begin{bmatrix} SA \\ 0 \end{bmatrix} \right\|_1 \\ &= \alpha \| S(\widehat{U}V^* - A) \|_1 + \alpha\lambda \|V^*\|_1 \\ &\leq \alpha\beta \|\widehat{U}V^* - A\|_1 + \alpha\lambda \|V^*\|_1 \\ &\leq \alpha\beta \|\widehat{U}V^* - A\|_1 + \alpha\beta\lambda \|V^*\|_1 + \alpha\beta\lambda \|\widehat{U}\|_1 \\ &\leq \alpha\beta\gamma \, \mathrm{OPT}, \end{aligned}$$

the second step follows by $\|S(\widehat{U}V^* - A)\|_1 \leq \beta \|\widehat{U}V^* - A\|_1$. □

We show how to bound the first term on RHS of Claim 18.4.1,

Claim 18.4.3.

$$\alpha \left\| \begin{bmatrix} S\widehat{U} \\ \lambda I \end{bmatrix} \widehat{V} - \begin{bmatrix} SA \\ 0 \end{bmatrix} \right\|_1 \leq \alpha\beta\sqrt{m+k} \, \mathrm{OPT} \, .$$

Proof. We have

$$
\begin{aligned}
&\alpha \left\| \begin{bmatrix} S\widehat{U} \\ \lambda I \end{bmatrix} \widehat{V} - \begin{bmatrix} SA \\ 0 \end{bmatrix} \right\|_1 \\
&= \alpha \sum_{i=1}^{n} \left\| \begin{bmatrix} S\widehat{U} \\ \lambda I \end{bmatrix} \widehat{V}_i - \begin{bmatrix} SA_i \\ 0 \end{bmatrix} \right\|_1 \\
&\leq \alpha \sqrt{m+k} \sum_{i=1}^{n} \left\| \begin{bmatrix} S\widehat{U} \\ \lambda I \end{bmatrix} \widehat{V}_i - \begin{bmatrix} SA_i \\ 0 \end{bmatrix} \right\|_2 \\
&\leq \alpha \sqrt{m+k} \sum_{i=1}^{n} \left\| \begin{bmatrix} S\widehat{U} \\ \lambda I \end{bmatrix} V_i^* - \begin{bmatrix} SA_i \\ 0 \end{bmatrix} \right\|_2 \\
&\leq \alpha \sqrt{m+k} \sum_{i=1}^{n} \left\| \begin{bmatrix} S\widehat{U} \\ \lambda I \end{bmatrix} V_i^* - \begin{bmatrix} SA_i \\ 0 \end{bmatrix} \right\|_1 \\
&= \alpha \sqrt{m+k} \left\| \begin{bmatrix} S\widehat{U} \\ \lambda I \end{bmatrix} V^* - \begin{bmatrix} SA \\ 0 \end{bmatrix} \right\|_1 \\
&= \alpha \sqrt{m+k} (\|S\widehat{U}V^* - SA\|_1 + \lambda \|V^*\|_1) \\
&\leq \alpha \sqrt{m+k} (\beta \|\widehat{U}V^* - A\|_1 + \lambda \|V^*\|_1) \\
&\leq \alpha\beta \sqrt{m+k}\, \mathrm{OPT}
\end{aligned}
$$

where the second step follows by $\forall d \in \mathbb{Z}_{>0}, x \in \mathbb{R}^d, \|x\|_1 \leq \sqrt{d}\|x\|_2$, the third step follows by definition of $\widehat{V}$, the fourth step follows by $\forall d \in \mathbb{Z}_{>0}, x \in \mathbb{R}^d \ \|x\|_2 \leq \|x\|_1$, the seventh step follows by $\|S\widehat{U}V^* - SA\|_1 \leq \beta \|\widehat{U}V^* - A\|_1$, and the last step follows by $\beta \geq 1, \lambda \|\widehat{U}\|_1 \geq 0$. $\square$

Putting it all together, we

$$
\begin{aligned}
&\|\widehat{U}\widehat{V} - A\|_1 + \lambda(\|\widehat{U}\|_1 + \|\widehat{V}\|_1) \\
&\leq \alpha\beta \sqrt{m+k}\, \mathrm{OPT} + \alpha\beta\gamma\, \mathrm{OPT} + \gamma\, \mathrm{OPT} \\
&\leq \alpha\beta \sqrt{m+k}\, \mathrm{OPT} + 2\alpha\beta\gamma\, \mathrm{OPT} \\
&\leq \alpha\beta\gamma(\sqrt{m+k} + 2)\, \mathrm{OPT},
\end{aligned}
$$

where the first step follows by Claim 18.4.1, Claim 18.4.2 and Claim 18.4.3, the second step follows by $\alpha\beta \geq 1$, the last step follows by $\gamma \geq 1$.

Thus, we complete the proof. □

18.5 Proof of Lemma 18.3.3

Lemma 18.3.3. (Reducing the problem size by sketching on one side)**.** Given three matrices $L \in \mathbb{R}^{d_1 \times l_1}$, $M^\top \in \mathbb{R}^{d_2 \times l_2}$, $N \in \mathbb{R}^{d_1 \times d_2}$ and $\alpha \geq 1, \beta \geq 1, \gamma \geq 1, \lambda > 0$. Let $\mathrm{OPT} = \min_{X,Y} \|LXYM - N\|_1 + \lambda(\|LX\|_1 + \|YM\|_1)$. Let X^*, Y^* denote the optimal solution of

$$\min_{X,Y} \|LXYM - N\|_1 + \lambda(\|LX\|_1 + \|YM\|_1).$$

Let $T \in \mathbb{R}^{t \times d_1}$ satisfy :

$$\|TLx\|_1 \geq \frac{1}{\alpha}\|Lx\|_1, \text{ for all } x \in \mathbb{R}^{l_1}$$

and

$$\begin{aligned} & \|TLX^*Y^*M - TN\|_1 + \lambda \cdot \|TLX^*\|_1 \\ \leq & \beta(\|LX^*Y^*M - N\|_1 + \lambda \cdot \|LX^*\|_1) \end{aligned}$$

Let $\widehat{X}, \widehat{Y}$ satisfy

$$\begin{aligned} & \|TL\widehat{X}\widehat{Y}M - TN\|_1 + \lambda \cdot (\|TL\widehat{X}\|_1 + \|\widehat{Y}M\|_1) \\ \leq & \gamma \min_{X,Y} \|TLXYM - TN\|_1 + \lambda \cdot (\|TLX\|_1 + \|YM\|_1) \end{aligned}$$

Then

$$\begin{aligned} & \|L\widehat{X}\widehat{Y}M - N\|_1 + \lambda(\|L\widehat{X}\|_1 + \|\widehat{Y}M\|_1) \\ \leq & (\alpha\beta\gamma + \alpha\beta + 1)\,\mathrm{OPT}\,. \end{aligned}$$

Proof.

$$
\begin{aligned}
&\|L\widehat{X}\widehat{Y}M - N\|_1 + \lambda(\|L\widehat{X}\|_1 + \|\widehat{Y}M\|_1) \\
\leq\ & \|L(\widehat{X}\widehat{Y} - X^*Y^*)M\|_1 + \|LX^*Y^*M - N\|_1 + \lambda(\|L\widehat{X}\|_1 + \|\widehat{Y}M\|_1) \\
\leq\ & \|L(\widehat{X}\widehat{Y} - X^*Y^*)M\|_1 + \mathrm{OPT} + \lambda(\|L\widehat{X}\|_1 + \|\widehat{Y}M\|_1) \\
\leq\ & \alpha\|TL(\widehat{X}\widehat{Y} - X^*Y^*)M\|_1 + \mathrm{OPT} + \lambda(\alpha\|TL\widehat{X}\|_1 + \|\widehat{Y}M\|_1) \\
\leq\ & \alpha\|TL\widehat{X}\widehat{Y}M - TN\|_1 + \alpha\|TLX^*Y^*M - TN\|_1 + \lambda(\alpha\|TL\widehat{X}\|_1 + \|\widehat{Y}M\|_1) + \mathrm{OPT} \\
\leq\ & \alpha(\|TL\widehat{X}\widehat{Y}M - TN\|_1 + \lambda(\|TL\widehat{X}\|_1 + \|\widehat{Y}M\|_1)) + \alpha\|TLX^*Y^*M - TN\|_1 + \mathrm{OPT} \\
\leq\ & \alpha\gamma(\|TLX^*Y^*M - TN\|_1 + \lambda(\|TLX^*\|_1 + \|Y^*M\|_1)) + \alpha\|TLX^*Y^*M - TN\|_1 + \mathrm{OPT} \\
\leq\ & \alpha\beta\gamma\,\mathrm{OPT} + \alpha\|TLX^*Y^*M - TN\|_1 + \mathrm{OPT} \\
\leq\ & \alpha\beta\gamma\,\mathrm{OPT} + \alpha\beta\|LX^*Y^*M - N\|_1 + \mathrm{OPT} \\
=\ & (\alpha\beta\gamma + \alpha\beta + 1)\,\mathrm{OPT},
\end{aligned}
$$

where the first step follows by triangle inequality, the second step follows by $\|LX^*Y^*M - N\|_1 \leq \mathrm{OPT}$, the third step follows by the property of T, the forth step follows by triangle inequality, the fifth step follows by $\alpha \geq 1$, the sixth step follows by the definition of $\widehat{X}, \widehat{Y}$, the seventh step follows by the property of T, the eighth step follows by the property of T. □

18.6 Proof of Theorem 18.3.7

Proof. We choose all the matrices $S \in \mathbb{R}^{s\times n}, R \in \mathbb{R}^{n\times r}, T_1 \in \mathbb{R}^{t_1\times n}, T_2 \in \mathbb{R}^{n\times t_2}$ in Algorithm 18.1 as sparse Cauchy transformation matrices (see Definition 18.2.2) which have the same scalar $\sigma = 1$. We set $s = O(k^5 \log^5 k), r = O(k^5 \log^5 k), t_1 = O(r^5 \log^5 r), t_2 = O(s^5 \log^5 s)$.

Since computing $T_1 A, AT_2$ can be done in $\mathrm{nnz}(A)$ time, and the size of $T_1 A, AT_2$ is at most $n \cdot \mathrm{poly}(k)$, the total running time of computing $T_1 AR, SAT_2, T_1 AT_2$ is at most $\mathrm{nnz}(A) + n \cdot \mathrm{poly}(k)$. Since the size of the final small Frobenius norm ridge low-rank matrix approximation problem is $\mathrm{poly}(k)$, the running time of the whole algorithm is at most $\mathrm{nnz}(A) + n \cdot \mathrm{poly}(k)$.

Now let us consider the correctness of the algorithm.

Let $U^* \in \mathbb{R}^{n\times k}, V^* \in \mathbb{R}^{k\times n}$ satisfy $\|U^*V^* - A\|_1 + \lambda(\|U^*\|_1 + \|V^*\|_1) = \mathrm{OPT}$. Then due to Lemma 18.2.3, with probability at least 0.99, S satisfies

$$\|SU^*x\|_1 \geq \frac{1}{\alpha_1}\|U^*x\|_1 \text{ for all } x \in \mathbb{R}^k \text{ and } \|SU^*V^* - SA\|_1 \leq \beta_1\|U^*V^* - A\|_1$$

where $\alpha_1\beta_1 = O(k^2 \log^2 k \log n)$. Let $\widehat{V} \in \mathbb{R}^{k\times n}$ denote $\begin{bmatrix} SU^* \\ \lambda I \end{bmatrix}^\dagger \cdot \begin{bmatrix} I \\ 0 \end{bmatrix} \cdot SA$. Let $\widetilde{U} \in \mathbb{R}^{n\times k}$ denote the optimal solution of

$$\min_{U\in\mathbb{R}^{n\times k}} \|U\widehat{V} - A\|_1 + \lambda(\|U\|_1 + \|\widehat{V}\|_1).$$

Due to Lemma 18.2.3, with probability at least 0.99, $R^\top \in \mathbb{R}^{r\times n}$ satisfies:

$$\|R^\top\widehat{V}^\top x\|_1 \geq \frac{1}{\alpha_2}\|\widehat{V}^\top x\|_1 \text{ for all } x \in \mathbb{R}^k, \text{ and}$$
$$\|R^\top\widehat{V}^\top\widetilde{U}^\top - R^\top A^\top\|_1 \leq \beta_2\|\widehat{V}^\top\widetilde{U}^\top - A^\top\|_1$$

where $\alpha_2\beta_2 = O(k^2 \log^2 k \log n)$. Then, by applying Corollary 18.3.2, we have that

$$\min_{X\in\mathbb{R}^{r\times k}, Y\in\mathbb{R}^{k\times s}} \|ARXYSA - A\|_1 + \lambda(\|ARX\|_1 + \|YSA\|_1) \leq \mathrm{poly}(k, \log n)\,\mathrm{OPT}\,.$$

Let $X^* \in \mathbb{R}^{r\times k}, Y^* \in \mathbb{R}^{k\times s}$ denote the optimal solution of

$$\min_{X,Y} \|ARXYSA - A\|_1 + \lambda(\|ARX\|_1 + \|YSA\|_1).$$

Due to Lemma 18.2.3, with probability at least 0.99, T_1 satisfies that $\forall x \in \mathbb{R}^r, \|T_1 ARx\|_1 \geq \frac{1}{\alpha_1'}\|ARx\|_1$, and $\|T_1\left(\begin{bmatrix} ARX^*Y^*SA - A & \lambda ARX^* \end{bmatrix}\right)\|_1 \leq \beta_1' \|\begin{bmatrix} ARX^*Y^*SA - A & \lambda ARX^* \end{bmatrix}\|_1$ where $\alpha_1'\beta_1' = O(r^2 \log^2 r \log n)$. Due to Lemma 18.2.3, with probability at least 0.99, T_2 satisfies that $\forall x \in \mathbb{R}^s, \|T_2^\top A^\top S^\top x\|_1 \geq \frac{1}{\alpha_2'}\|A^\top S^\top x\|_1$ and

$$\begin{aligned}
&\|T_2^\top \left(\begin{bmatrix} A^\top S^\top (Y^*)^\top (X^*)^\top R^\top A^\top T_1^\top - A^\top T_1^\top & \lambda A^\top S^\top (Y^*)^\top \end{bmatrix}\right)\|_1 \\
\leq\ & \beta_2' \|\begin{bmatrix} A^\top S^\top (Y^*)^\top (X^*)^\top R^\top A^\top T_1^\top - A^\top T_1^\top & \lambda A^\top S^\top (Y^*)^\top \end{bmatrix}\|_1
\end{aligned}$$

where $\alpha_2'\beta_2' \leq O(s^2 \log^2 s \log n)$. Thus, by Corollary 18.3.4, for any $\gamma \geq 1$, if $\widetilde{X}, \widetilde{Y}$ can achieve

$$\begin{aligned}
&\|T_1 AR\widetilde{X}\widetilde{Y}SAT_2 - T_1AT_2\|_1 + \lambda(\|T_1AR\widetilde{X}\|_1 + \|\widetilde{Y}SAT_2\|_1) \\
\leq\ & \gamma \min_{X,Y} \left(\|T_1ARXYSAT_2 - T_1AT_2\|_1 + \lambda(\|T_1ARX\|_1 + \|YSAT_2\|_1)\right),
\end{aligned}$$

then

$$\begin{aligned}
&\|AR\widetilde{X}\widetilde{Y}SA - A\|_1 + \lambda(\|AR\widetilde{X}\|_1 + \|\widetilde{Y}SA\|_1) \\
\leq\ & \mathrm{poly}(k, \log n)\gamma \min_{X,Y} \left(\|ARXYSA - A\|_1 + \lambda(\|ARX\|_1 + \|YSA\|_1)\right).
\end{aligned}$$

Finally, due to Lemma 18.3.6, we can find such $\widetilde{X}\widetilde{Y}$ with $\gamma = O(\sqrt{t_1 + k}\sqrt{t_2 + k})$. Thus, we can conclude that $\|AR\widetilde{X}\widetilde{Y}SA - A\|_1 + \lambda(\|AR\widetilde{X}\|_1 + \|\widetilde{Y}SA\|_1) \leq \mathrm{poly}(k, \log n)\cdot\mathrm{OPT}\,.$

By taking union bound over all the success event of S, R, T_1, T_2, the algorithm will succeed with probability at least 9/10. □

18.7 Proof of Theorem 18.3.8

Proof. We choose all the matrices $S \in \mathbb{R}^{s\times n}, R \in \mathbb{R}^{n\times r}, T_1 \in \mathbb{R}^{t_1\times n}, T_2 \in \mathbb{R}^{n\times t_2}$ in Algorithm 18.1 as dense Cauchy transformation matrices (see Definition 18.2.1) which have the same scalar $\sigma = 1$. We set $s = O(k\log k), r = O(k\log k), t_1 = O(r\log r), t_2 = O(s\log s)$.

Since computing T_1A, AT_2 can be done in $t_1\,\mathrm{nnz}(A), t_2\,\mathrm{nnz}(A)$ time respectiviely, and the size of T_1A, AT_2 is at most $n\cdot\mathrm{poly}(k)$, the total running time of computing

$$T_1AR, SAT_2, T_1AT_2$$

is at most $\widetilde{O}(k)\cdot\mathrm{nnz}(A) + n\cdot\mathrm{poly}(k)$. Since the size of the final small Frobenius norm ridge low-rank matrix approximation problem is $\mathrm{poly}(k)$, the running time of the whole algorithm is at most $\widetilde{O}(k)\cdot\mathrm{nnz}(A) + n\cdot\mathrm{poly}(k)$.

Now let us consider the correctness of the algorithm.

Let $U^* \in \mathbb{R}^{n\times k}, V^* \in \mathbb{R}^{k\times n}$ satisfy $\|U^*V^* - A\|_1 + \lambda(\|U^*\|_1 + \|V^*\|_1) = \mathrm{OPT}$. Then due to Lemma 18.2.3, with probability at least 0.99, S satisfies

$$\|SU^*x\|_1 \geq \frac{1}{\alpha_1}\|U^*x\|_1 \text{ for all } x\in\mathbb{R}^k \text{ and } \|SU^*V^* - SA\|_1 \leq \beta_1\|U^*V^* - A\|_1$$

where $\alpha_1\beta_1 = O(\log n)$. Let $\widehat{V} \in \mathbb{R}^{k\times n}$ denote $\begin{bmatrix} SU^* \\ \lambda I \end{bmatrix}^\dagger \cdot \begin{bmatrix} I \\ 0 \end{bmatrix} \cdot SA$. Let $\widetilde{U} \in \mathbb{R}^{n\times k}$ denote the optimal solution of $\min_{U\in\mathbb{R}^{n\times k}} \|U\widehat{V} - A\|_1 + \lambda(\|U\|_1 + \|\widehat{V}\|_1)$. Due to Lemma 18.2.3, with probability at least 0.99, $R^\top \in \mathbb{R}^{r\times n}$ satisfies: $\|R^\top\widehat{V}^\top x\|_1 \geq \frac{1}{\alpha_2}\|\widehat{V}^\top x\|_1$ for all $x \in \mathbb{R}^k$, and $\|R^\top\widehat{V}^\top\widetilde{U}^\top - R^\top A^\top\|_1 \leq \beta_2\|\widehat{V}^\top\widetilde{U}^\top - A^\top\|_1$ where $\alpha_2\beta_2 = O(\log n)$. Then, by applying Corollary 18.3.2, we have that

$$\min_{X\in\mathbb{R}^{r\times k}, Y\in\mathbb{R}^{k\times s}} \|ARXYSA - A\|_1 + \lambda(\|ARX\|_1 + \|YSA\|_1) \leq \widetilde{O}(k)\,\mathrm{poly}(\log n)\,\mathrm{OPT}.$$

Let $X^* \in \mathbb{R}^{r\times k}, Y^* \in \mathbb{R}^{k\times s}$ denote the optimal solution of $\min_{X,Y} \|ARXYSA - A\|_1 + \lambda(\|ARX\|_1 + \|YSA\|_1)$. Due to Lemma 18.2.3, with probability at least 0.99, T_1 satisfies that $\forall x \in \mathbb{R}^r, \|T_1 ARx\|_1 \geq \frac{1}{\alpha_1'}\|ARx\|_1$, and $\|T_1 \left(\begin{bmatrix} ARX^*Y^*SA - A & \lambda ARX^* \end{bmatrix}\right)\|_1 \leq \beta_1' \| \begin{bmatrix} ARX^*Y^*SA - A & \lambda ARX^* \end{bmatrix} \|_1$ where $\alpha_1'\beta_1' = O(\log n)$. Due to Lemma 18.2.3, with probability at least 0.99, T_2 satisfies that

$$\forall x \in \mathbb{R}^s, \|T_2^\top A^\top S^\top x\|_1 \geq \frac{1}{\alpha_2'}\|A^\top S^\top x\|_1$$

and

$$\begin{aligned}
&\|T_2^\top \left(\begin{bmatrix} A^\top S^\top (Y^*)^\top (X^*)^\top R^\top A^\top T_1^\top - A^\top T_1^\top & \lambda A^\top S^\top (Y^*)^\top \end{bmatrix}\right)\|_1 \\
\leq\ & \beta_2' \| \begin{bmatrix} A^\top S^\top (Y^*)^\top (X^*)^\top R^\top A^\top T_1^\top - A^\top T_1^\top & \lambda A^\top S^\top (Y^*)^\top \end{bmatrix} \|_1
\end{aligned}$$

where $\alpha_2'\beta_2' \leq O(\log n)$. Thus, by Corollary 18.3.4, for any $\gamma \geq 1$, if $\widetilde{X}, \widetilde{Y}$ can achieve

$$\begin{aligned}
&\|T_1 AR\widetilde{X}\widetilde{Y}SAT_2 - T_1AT_2\|_1 + \lambda(\|T_1AR\widetilde{X}\|_1 + \|\widetilde{Y}SAT_2\|_1) \\
\leq\ & \gamma \min_{X,Y} \left(\|T_1ARXYSAT_2 - T_1AT_2\|_1 + \lambda(\|T_1ARX\|_1 + \|YSAT_2\|_1)\right),
\end{aligned}$$

then

$$\begin{aligned}
&\|AR\widetilde{X}\widetilde{Y}SA - A\|_1 + \lambda(\|AR\widetilde{X}\|_1 + \|\widetilde{Y}SA\|_1) \\
\leq\ & \widetilde{O}(k) \cdot \mathrm{poly}(\log n)\gamma \min_{X,Y} \left(\|ARXYSA - A\|_1 + \lambda(\|ARX\|_1 + \|YSA\|_1)\right).
\end{aligned}$$

Finally, due to Lemma 18.3.6, we can find such $\widetilde{X}\widetilde{Y}$ with $\gamma = O(\sqrt{t_1 + k}\sqrt{t_2 + k})$. Thus, we can conclude that

$$\|AR\widetilde{X}\widetilde{Y}SA - A\|_1 + \lambda(\|AR\widetilde{X}\|_1 + \|\widetilde{Y}SA\|_1) \leq \widetilde{O}(k^2)\,\mathrm{poly}(\log n) \cdot \mathrm{OPT}\,.$$

By taking union bound over all the success event of S, R, T_1, T_2, the algorithm will succeed with probability at least 9/10. □

Chapter 19

Average Case L1 Low Rank Approximation

We study the column subset selection problem with respect to the entrywise ℓ_1-norm loss. It is known that in the worst case, to obtain a good rank-k approximation to a matrix, one needs an arbitrarily large $n^{\Omega(1)}$ number of columns to obtain a $(1+\epsilon)$-approximation to the best entrywise ℓ_1-norm low rank approximation of an $n \times n$ matrix. Nevertheless, we show that under certain minimal and realistic distributional settings, it is possible to obtain a $(1+\epsilon)$-approximation with a nearly linear running time and $\text{poly}(k/\epsilon) + O(k \log n)$ columns. Namely, we show that if the input matrix A has the form $A = B + E$, where B is an arbitrary rank-k matrix, and E is a matrix with i.i.d. entries drawn from any distribution μ for which the $(1+\gamma)$-th moment exists, for an arbitrarily small constant $\gamma > 0$, then it is possible to obtain a $(1+\epsilon)$-approximate column subset selection to the entrywise ℓ_1-norm in nearly linear time. Conversely we show that if the first moment does not exist, then it is not possible to obtain a $(1+\epsilon)$-approximate subset selection algorithm even if one chooses any $n^{o(1)}$ columns. This is the first algorithm of any kind for achieving a $(1+\epsilon)$-approximation for entrywise ℓ_1-norm loss low rank approximation.

19.1 Introduction

Numerical linear algebra algorithms are fundamental building blocks in many machine learning and data mining tasks. A well-studied problem is low rank matrix approximation. The most common version of the problem is also known as Principal Component Analysis (PCA), in which the goal is to find a low rank matrix to approximate a given matrix such that the Frobenius norm of the error is minimized. The optimal solution of this objective can be obtained via the singular value decomposition (SVD). Hence, the problem can be solved in polynomial time. If approximate solutions are allowed, then the running time can be made almost linear in the number of non-zero entries of the given matrix [Sar06, CW13, MM13, NN13a, BDN15, Coh16a].

An important variant of the PCA problem is the entrywise ℓ_1-norm low rank matrix approximation problem. In this problem, instead of minimizing the Frobenius norm of the error, we seek to minimize the ℓ_1-norm of the error. In particular, given an $n \times n$ input matrix A, and a rank parameter k, we want to find a matrix B with rank at most k such that $\|A - B\|_1$ is minimized, where for a matrix C, $\|C\|_1$ is defined to be $\sum_{i,j} |C_{i,j}|$. There are several reasons for using the ℓ_1-norm as the error measure. For example, solutions with respect to the ℓ_1-norm loss are usually more robust than solutions with Frobenius norm loss [Hub64, CLMW11]. Further, the ℓ_1-norm loss is often used as a relaxation of the ℓ_0-loss, which has wide applications including sparse recovery, matrix completion, and robust PCA; see e.g., [XCS10, CLMW11]. Although a number of algorithms have been proposed for the ℓ_1-norm loss [KK03, KK05, KLC+15, Kwa08, ZLS+12, BJ12, BD13, BDB13, MXZZ13, MKP13, MKP14, MKCP16, PK16], the problem is known to be NP-hard [GV15]. The first ℓ_1-low rank approximation with provable guarantees was proposed by [SWZ17]. To cope

with NP-hardness, the authors gave a solution with a $\mathrm{poly}(k \log n)$-approximation ratio, i.e., their algorithm outputs a rank-k matrix $B' \in \mathbb{R}^{n \times n}$ for which

$$\|A - B'\|_1 \leq \alpha \cdot \min_{\mathrm{rank}-k \ B} \|A - B\|_1$$

for $\alpha = \mathrm{poly}(k \log n)$. The approximation ratio α was further improved to $O(k \log k)$ by allowing B' to have a slightly larger $k' = O(k \log n)$ rank [CGK+17a]. Such B' with larger rank is referred to as a bicriteria solution. However, in high precision applications, such approximation factors are too large. A natural question is if one can compute a $(1+\epsilon)$-approximate solution efficiently for ℓ_1-norm low rank approximation. In fact, a $(1+\epsilon)$-approximation algorithm was given in [BBB+19a], but the running time of their algorithm is a prohibitive $n^{\mathrm{poly}(k/\epsilon)}$. Unfortunately, [BBB+19a] shows in the worst case that a $2^{k^{\Omega(1)}}$ running time is necessary for any constant approximation given a standard conjecture in complexity theory.

19.1.1 Notation

To describe our results, let us first introduce some notation. We will use $[n]$ to denote the set $\{1, 2, \cdots, n\}$. We use A_i to denote the i^{th} column of A. We use A^j to denote the j^{th} row of A. Let $Q \subseteq [n]$. We use A_Q to denote the matrix which is comprised of the columns of A with column indices in Q. Similarly, we use A^Q to denote the matrix which is comprised of the rows of A with row indices in Q. We use $\binom{[n]}{t}$ to denote the set of all the size-t subsets of $[n]$. Let $\|A\|_F$ denote the Frobenius norm of a matrix A, i.e., $\|A\|_F$ is the square root of the sum of squares of all the entries in A. For $1 \leq p < 2$, we use $\|A\|_p$ to denote the entry-wise ℓ_p-norm of a matrix A, i.e., $\|A\|_p$ is the p-th root of the sum of p-th powers of the absolute values of the entries of A. $\|A\|_1$ is an important special case of $\|A\|_p$,

which corresponds to the sum of absolute values of the entries in A. A random variable X has the Cauchy distribution if its probability density function is $f(z) = \frac{1}{\pi(1+z^2)}$.

19.1.2 Our Results

We propose an efficient bicriteria $(1+\epsilon)$-approximate column subset selection algorithm for the ℓ_1-norm. We bypass the running time lower bound mentioned above by making a mild assumption on the input data, and also show that our assumption is necessary in a certain sense.

Our main algorithmic result is described as follows.

Theorem 19.1.1 (Informal version of Theorem 19.2.9)**.** *Suppose we are given a matrix $A = A^* + \Delta \in \mathbb{R}^{n\times n}$, where $\mathrm{rank}(A^*) = k$ for $k = n^{o(1)}$, and Δ is a random matrix for which the $\Delta_{i,j}$ are i.i.d. symmetric random variables with $\mathbb{E}[|\Delta_{i,j}|^p] = O(\mathbb{E}[|\Delta_{i,j}|]^p)$ for some constant $p > 1$. Let $\epsilon \in (0, 1/2)$ satisfy $1/\epsilon = n^{o(1)}$. There is an $\widetilde{O}(n^2 + n\,\mathrm{poly}(k/\epsilon))$[1] time algorithm (Algorithm 19.1) which can output a subset $S \subseteq [n]$ with $|S| \leq \mathrm{poly}(k/\epsilon) + O(k \log n)$ for which*

$$\min_{X\in\mathbb{R}^{|S|\times n}} \|A_S X - A\|_1 \leq (1+\epsilon)\|\Delta\|_1,$$

holds with probability at least 99/100.

Note the running time in Theorem 19.1.1 is nearly linear in the number of non-zero entries of A, since for an $n \times n$ matrix with i.i.d. noise drawn from any continuous distribution, the number of non-zero entries of A will be n^2 with probability 1. We also show the moment assumption of Theorem 19.1.1 is necessary in the following precise sense.

[1] We use the notation $\widetilde{O}(f) := O(f \cdot \log^{O(1)} f)$.

Theorem 19.1.2 (Hardness, informal version of Theorem 19.4.17)**.** *Let $n > 0$ be sufficiently large. Let $A = \eta \cdot \mathbf{1} \cdot \mathbf{1}^\top + \Delta \in \mathbb{R}^{n \times n}$ be a random matrix where $\eta = n^{c_0}$ for some sufficiently large constant c_0, $\mathbf{1} \in \mathbb{R}^n$ is the all-ones vector, and $\forall i, j \in [n], \Delta_{i,j} \sim C(0,1)$ are i.i.d. standard Cauchy random variables. Let $r = n^{o(1)}$. Then with probability at least $1 - O(1/\log\log n)$, $\forall S \subseteq [n]$ with $|S| = r$,*

$$\min_{X \in \mathbb{R}^{r \times n}} \|A_S X - A\|_1 \geq 1.002 \|\Delta\|_1.$$

19.1.3 Our Techniques

For an overview of our hardness result, we refer readers to the supplementary material, namely, Appendix 19.4. In the following, we will outline the main techniques used in our algorithm.

$(1+\epsilon)$-Approximate ℓ_1-Low Rank Approximation. We make the following distributional assumption on the input matrix $A \in \mathbb{R}^{n \times n}$: namely, $A = A^* + \Delta$ where A^* is an arbitrary rank-k matrix and the entries of Δ are i.i.d. from any symmetric distribution with $\mathbb{E}[|\Delta_{i,j}|] = 1$ and $\mathbb{E}[|\Delta_{i,j}|^p] = O(1)$ for any real number p strictly greater than 1, e.g., $p = 1.000001$ would suffice. Note that such an assumption is mild compared to typical noise models which require the noise be Gaussian or have bounded variance; in our case the random variables may even be heavy-tailed with infinite variance. In this setting we show it is possible to obtain a subset of $\text{poly}(k(\epsilon^{-1} + \log n))$ columns spanning a $(1+\epsilon)$-approximation. This provably overcomes the column subset selection lower bound of [SWZ17] which shows for entrywise ℓ_1-low rank approximation that there are matrices for which any subset of $\text{poly}(k)$ columns spans at best a $k^{\Omega(1)}$-approximation.

Consider the following algorithm: sample $\text{poly}(k/\epsilon)$ columns of A, and try to cover as many of the remaining columns as possible. Here, by covering a column i, we mean that if A_I is the subset of columns sampled, then $\min_y \|A_I y - A_i\|_1 \leq (1+O(\epsilon))n$. The reason for this notion of covering is that we are able to show in Lemma 19.2.1 that in this noise model, $\|\Delta\|_1 \geq (1-\epsilon)n^2$ w.h.p., and so if we could cover every column i, our overall cost would be $(1+O(\epsilon))n^2$, which would give a $(1+O(\epsilon))$-approximation to the overall cost.

We will not be able to cover all columns, unfortunately, with our initial sample of $\text{poly}(k/\epsilon)$ columns of A. Instead, though, we will show that we will be able to cover all but a set T of $\epsilon n/(k \log k)$ of the columns. Fortunately, we show in Lemma 19.2.4 another property of the noise matrix Δ is that *all* subsets S of columns of size at most n/r, for $r \geq (1/\gamma)^{1+1/(p-1)}$ satisfy $\sum_{j \in S} \|\Delta_j\|_1 = O(\gamma n^2)$. Thus, for the above set T that we do not cover, we can apply this lemma to it with $\gamma = \epsilon/(k \log k)$, and then we know that $\sum_{j \in T} \|\Delta_j\|_1 = O(\epsilon n^2/(k \log k))$, which then enables us to run a previous $\widetilde{O}(k)$-approximate ℓ_1 low rank approximation algorithm [CGK+17a] on the set T, which will only incur total cost $O(\epsilon n^2)$, and since by Lemma 19.2.1 above the overall cost is at least $(1-\epsilon)n^2$, we can still obtain a $(1+O(\epsilon))$-approximation overall.

The main missing piece of the algorithm to describe is why we are able to cover all but a small fraction of the columns. One thing to note is that our noise distribution may not have a finite variance, and consequently, there *can* be very large entries $\Delta_{i,j}$ in some columns. In Lemma 19.2.3, we show the number of columns in Δ for which there exists an entry larger than $n^{1/2+1/(2p)}$ in magnitude is $O(n^{(2-p)/2})$, which since $p > 1$ is a constant bounded away from 1, is sublinear. Let us call this set with entries larger than $n^{1/2+1/(2p)}$ in magnitude the set H of "heavy" columns; we will not make any guarantees about H,

rather, we will stuff it into the small set T of columns above on which we will run our earlier $O(k \log k)$-approximation.

For the remaining, non-heavy columns, which constitute almost all of our columns, we show in Lemma 19.2.5 that $\|\Delta_i\|_1 \leq (1+\epsilon)n$ w.h.p. The reason this is important is that recall to cover some column i by a sample set I of columns, we need $\min_y \|A_I y - A_i\|_1 \leq (1+O(\epsilon))n$. It turns out, as we now explain, that we will get

$$\min_y \|A_I y - A_i\|_1 \leq \|\Delta_i\|_1 + e_i,$$

where e_i is a quantity which we can control and make $O(\epsilon n)$ by increasing our sample size I. Consequently, since $\|\Delta_i\|_1 \leq (1+\epsilon)n$, overall we will have $\min_y \|A_I y - A_i\|_1 \leq (1+O(\epsilon))n$, which means that i will be covered. We now explain what e_i is, and why $\min_y \|A_I y - A_i\|_1 \leq \|\Delta_i\|_1 + e_i$.

Towards this end, we first explain a key insight in this model. Since the p-th moment exists for some real number $p > 1$ (e.g., $p = 1.000001$ suffices), *averaging* helps reduce the noise of fitting a column A_i by subsets of other columns. Namely, we show in Lemma 19.2.2 that for any t non-heavy column $\Delta_{i_1}, \ldots, \Delta_{i_t}$ of Δ, and any coefficients $\alpha_1, \alpha_2, \ldots, \alpha_t \in [-1, 1]$, $\|\sum_{j=1}^t \alpha_j \Delta_{i_j}\|_1 = O(t^{1/p} n)$, that is, since the individual coordinates of the Δ_{i_j} are zero-mean random variables, their sum *concentrates* as we add up more columns. We do not need bounded variance for this property.

How can we use this averaging property for subset selection? The idea is, instead of sampling a single subset I of $O(k)$ columns and trying to cover each remaining column with this subset as shown in [CGK+17a], we will sample multiple independent subsets $I_1, I_2, \ldots, I_t$. By a similar argument of [CGK+17a], for any given column index $i \in [n]$, for most of these

subset I_j, we have that $A_i^*/\|\Delta_i\|_1$ can be expressed as a linear combination of columns $A_\ell^*/\|\Delta_\ell\|_1, \ell \in I_j$, via coefficients of absolute value at most 1. Note that this is only true for most i and most j; we develop terminology for this in Definitions 19.2.1, 19.2.2, 19.2.3, and 19.2.4, referring to what we call a *good core*. We quantify what we mean by most i and most j having this property in Lemma 19.2.7 and Lemma 19.2.8.

The key though, that drives the analysis, is Lemma 19.2.6, which shows that $\min_y \|A_i y - A_i\|_1 \leq \|\Delta_i\|_1 + e_i$, where $e_i = O(q^{1/p}/t^{1-1/p}n)$, where q is the size of each I_j, and t is the number of different I_j. We need q to be at least k, just as before, so that we can be guaranteed that when we adjoin a column index i to I_j, there is some positive probability that $A_i^*/\|\Delta_i\|_1$ can be expressed as a linear combination of columns $A_\ell^*/\|\Delta_\ell\|_1, \ell \in I_j$, with coefficients of absolute value at most 1. What is different in our noise model though is the division by $t^{1-1/p}$. Since $p > 1$, if we set t to be a large enough poly(k/ϵ), then $e_i = O(\epsilon n)$, and then we will have covered A_i, as desired. This captures the main property that averaging the linear combinations for expression $A_i^*/\|\Delta_i\|_1$ using different subsets I_j gives us better and better approximations to $A_i^*/\|\Delta_i\|_1$. Of course we need to ensure several properties such as not sampling a heavy column (the averaging in Lemma 19.2.2 does not apply when this happens), we need to ensure most of the I_j have small-coefficient linear combinations expressing $A_i^*/\|\Delta_i\|_1$, etc. This is handled in our main theorem, Theorem 19.2.9.

19.2 ℓ_1-Norm Column Subset Selection

We first present two subroutines.

Linear regression with ℓ_1 loss. The first subroutine needed is an approximate ℓ_1 linear regression solver. In particular, given a matrix $M \in \mathbb{R}^{n\times d}$, n vectors $b_1, b_2, \cdots, b_n \in \mathbb{R}^n$, and an error parameter $\epsilon \in (0,1)$, we want to compute $x_1, x_2, \cdots, x_n \in \mathbb{R}^d$ for which $\forall i \in [n]$, we have

$$\|Mx_i - b_i\|_1 \leq (1+\epsilon) \cdot \min_{x\in\mathbb{R}^d} \|Mx - b_i\|_1.$$

Furthermore, we also need an estimate v_i of the regression cost $\|Mx_i - b_i\|_1$ for each $i \in [n]$ such that $\|Mx_i - b_i\|_1 \leq v_i \leq (1+\epsilon)\|Mx_i - b_i\|_1$. Such an ℓ_1-regression problem can be solved efficiently (see [Woo14b] for a survey). The total running time to solve these n regression problems simultaneously is at most $\widetilde{O}(n^2) + n \cdot \mathrm{poly}(d \log n)$, and the success probability is at least 0.999.

ℓ_1 Column subset selection for general matrices. The second subroutine needed is an ℓ_1-low rank approximation solver for general input matrices, though we allow a large approximation ratio. We use the algorithm proposed by [CGK+17a] for this purpose. In particular, given an $n \times d$ ($d \leq n$) matrix M and a rank parameter k, the algorithm can output a small set $S \subset [n]$ with size at most $O(k \log n)$, such that

$$\min_{X\in\mathbb{R}^{|S|\times d}} \|M_S X - M\|_1 \leq O(k \log k) \cdot \min_{\mathrm{rank}-k\ B} \|M - B\|_1.$$

Furthermore, the running time is at most $\widetilde{O}(n^2) + n \cdot \mathrm{poly}(k \log n)$, and the success probability is at least 0.999. Now we can present our algorithm, Algorithm 19.1.

Algorithm 19.1 ℓ_1-Low Rank Approximation with Input Assumption

procedure L1NoisyLowRankApprox($A \in \mathbb{R}^{n\times n}, k, \epsilon$) ▷ Theorem 19.2.9
  Sample a set I from $\binom{[n]}{s}$ uniformly at random, where $s = \mathrm{poly}(k/\epsilon)$.
  Solve the approximate ℓ_1-regression problem $\min_{x\in\mathbb{R}^{|I|}} \|A_I x - A_i\|_1$ for each $i \in [n]$, and let v_i be the estimated regression cost.
  Compute the set $T = \{i \in [n] \mid v_i$ is one of the top l largest values among $v_1, v_2, \cdots, v_n\}$, where $l = n/\mathrm{poly}(k/\epsilon)$.
  Solve ℓ_1-column subset selection for A_T. Let the solution be A_Q.
  Solve the approximate ℓ_1-regression problem $\min_{X\in\mathbb{R}^{(|I|+|Q|)\times n}} \|A_{(I\cup S)}X - A\|_1$, and let $\widehat{X}$ be the solution. Return $A_{(I\cup S)}$ and $\widehat{X}$. ▷ $A_{(I\cup S)}\widehat{X}$ is a good low rank approximation to A
end procedure

Running time. Uniformly sampling a set I can be done in $\mathrm{poly}(k/\epsilon)$ time. According to our ℓ_1-regression subroutine, solving $\min_x \|A_I x - A_i\|_1$ for all $i \in [n]$ can be finished in $\widetilde{O}(n^2) + n \cdot \mathrm{poly}(k\log(n)/\epsilon)$ time. We only need sorting to compute the set T which takes $O(n \log n)$ time. By our second subroutine, the ℓ_1-column subset selection for A_T will take $\widetilde{O}(n^2) + n \cdot \mathrm{poly}(k \log n)$. The last step only needs an ℓ_1-regression solver, which takes $\widetilde{O}(n^2)+n\cdot\mathrm{poly}(k\log(n)/\epsilon)$ time. Thus, the overall running time is $\widetilde{O}(n^2)+n\cdot\mathrm{poly}(k\log(n)/\epsilon)$.

The remaining parts in this section will focus on analyzing the correctness of the algorithm.

19.2.1 Properties of the Noise Matrix

Recall that the input matrix $A \in \mathbb{R}^{n\times n}$ can be decomposed as $A^* + \Delta$, where A^* is the ground truth, and Δ is a random noise matrix. In particular, A^* is an arbitrary rank-k matrix, and Δ is a random matrix where each entry is an i.i.d. sample drawn from an unknown symmetric distribution. The only assumption on Δ is that each entry $\Delta_{i,j}$

satisfies $\mathbb{E}[|\Delta_{i,j}|^p] = O(\mathbb{E}[|\Delta_{i,j}|^p])$ for some constant $p > 1$, i.e., the p-th moment of the noise distribution is bounded. Without loss of generality, we will suppose $\mathbb{E}[|\Delta_{i,j}|] = 1$, $\mathbb{E}[|\Delta_{i,j}|^p] = O(1)$, and $p \in (1, 2)$ throughout the paper. In this section, we will present some key properties of the noise matrix.

The following lemma provides a lower bound on $\|\Delta\|_1$. Once we have the such lower bound, we can focus on finding a solution for which the approximation cost is at most that lower bound.

Lemma 19.2.1 (Lower bound on the noise matrix)**.** *Let $\Delta \in \mathbb{R}^{n \times n}$ be a random matrix where $\Delta_{i,j}$ are i.i.d. samples drawn from a symmetric distribution. Suppose $\mathbb{E}[|\Delta_{i,j}|] = 1$ and $\mathbb{E}[|\Delta_{i,j}|^p] = O(1)$ for some constant $p \in (1, 2)$. Then, $\forall \epsilon \in (0, 1)$ which satisfies $1/\epsilon = n^{o(1)}$, we have*

$$\Pr\left[\|\Delta\|_1 \geq (1 - \epsilon)n^2\right] \geq 1 - e^{-\Theta(n)}.$$

The next lemma shows the main reason why we are able to get a small fitting cost when running regression. Consider a toy example. Suppose we have a target number $a \in \mathbb{R}$, and another t numbers $a + g_1, a + g_2, \cdots, a + g_t \in \mathbb{R}$, where g_i are i.i.d. samples drawn from the standard Gaussian distribution $N(0, 1)$. If we use $a + g_i$ to fit a, then the expected cost is $\mathbb{E}[|a + g_i - a|] = \mathbb{E}[|g_i|] = \sqrt{2/\pi}$. However, if we use the average of $a + g_1, a + g_2, \cdots, a + g_t$ to fit a, then the expected cost is $\mathbb{E}[|\sum_{i=1}^t g_i|/t]$. Since the g_i are independent, $\sum_{i=1}^t g_i$ is a random Gaussian variable with variance t, which means that the above expected cost is $\sqrt{2/\pi}/\sqrt{t}$. Thus the fitting cost is reduced by a factor $\sqrt{t}$. By generalizing the above argument, we obtain the following lemma.

Lemma 19.2.2 (Averaging reduces the noise)**.** *Let $\Delta_1, \Delta_2, \cdots, \Delta_t \in \mathbb{R}^n$ be t random vectors. The $\Delta_{i,j}$ are i.i.d. symmetric random variables with $\mathbb{E}[|\Delta_{i,j}|] = 1$ and $\mathbb{E}[|\Delta_{i,j}|^p] = O(1)$ for some constant $p \in (1,2)$. Let $\alpha_1, \alpha_2, \cdots, \alpha_t \in [-1,1]$ be t real numbers. Conditioned on $\forall i \in [n], j \in [t], |\Delta_{i,j}| \leq n^{1/2+1/(2p)}$, with probability at least $1 - 2^{-n^{\Theta(1)}}$,*

$$\left\| \sum_{i=1}^{t} \alpha_i \Delta_i \right\|_1 \leq O(t^{1/p} n).$$

The above lemma needs a condition that each entry in the noise column should not be too large. Fortunately, we can show that most of the (noise) columns do not have any large entry.

Lemma 19.2.3 (Only a small number of columns have large entries)**.** *Let $\Delta \in \mathbb{R}^{n \times n}$ be a random matrix where the $\Delta_{i,j}$ are i.i.d. symmetric random variables with $\mathbb{E}[|\Delta_{i,j}|] = 1$ and $\mathbb{E}[|\Delta_{i,j}|^p] = O(1)$ for some constant $p \in (1,2)$. Let*

$$H = \{ j \in [n] \mid \exists i \in [n], |\Delta_{i,j}| > n^{1/2+1/(2p)} \}.$$

Then with probability at least 0.999,

$$|H| \leq O(n^{1-(p-1)/2}).$$

The following lemma shows that any small subset of the columns of the noise matrix Δ cannot contribute too much to the overall error. By combining with the previous lemma, the entrywise ℓ_1 cost of all columns containing large entries can be bounded.

Lemma 19.2.4. *Let $\Delta \in \mathbb{R}^{n \times n}$ be a random matrix where $\Delta_{i,j}$ are i.i.d. symmetric random variables with $\mathbb{E}[|\Delta_{i,j}|] = 1$ and $\mathbb{E}[|\Delta_{i,j}|^p] = O(1)$ for some constant $p \in (1,2)$. Let $\epsilon \in (0,1)$*

satisfy $1/\epsilon = n^{o(1)}$. *Let* $r \geq (1/\epsilon)^{1+1/(p-1)}$. *Then, with probability at least* .999, $\forall S \subset [n]$ *with* $|S| \leq n/r$,

$$\sum_{j \in S} \|\Delta_j\|_1 = O(\epsilon n^2).$$

We say a (noise) column is good if it does not have a large entry. We can show that, with high probability, the entry-wise ℓ_1 cost of a good (noise) column is small.

Lemma 19.2.5 (Cost of good noise columns)**.** *Let* $\Delta \in \mathbb{R}^n$ *be a random vector where* Δ_i *are i.i.d. symmetric random variables with* $\mathbb{E}[|\Delta_i|] = 1$ *and* $\mathbb{E}[|\Delta_i|^p] = O(1)$ *for some constant* $p \in (1, 2)$. *Let* $\epsilon \in (0, 1)$ *satisfy* $1/\epsilon = n^{o(1)}$. *If* $\forall i \in [n], |\Delta_i| \leq n^{1/2+1/(2p)}$, *then with probability at least* $1 - 2^{-n^{\Theta(1)}}$,

$$\|\Delta\|_1 \leq (1 + \epsilon)n.$$

19.2.2 Definition of Tuples and Cores

In this section, we provide some basic definitions, e.g., of a tuple, a good tuple, the core of a tuple, and a coefficients tuple. These definitions will be heavily used later when we analyze the correctness of our algorithm.

Before we present the definitions, we introduce a notion $R_{A^*}(S)$. Given a matrix $A^* \in \mathbb{R}^{n_1 \times n_2}$, for a set $S \subseteq [n_2]$, we define

$$R_{A^*}(S) := \arg \max_{P: P \subseteq S} \left\{ \left| \det\left((A^*)_P^Q \right) \right| \;\middle|\; |P| = |Q| = \mathrm{rank}(A^*_S), Q \subseteq [n_1] \right\},$$

where for a squared matrix C, $\det(C)$ denotes the determinant of C. Roughly speaking, by Cramer's rule, if we use the columns of A^* with index in the set $R_{A^*}(S)$ to fit any column

of A^* with index in the set S, the absolute value of any fitting coefficient will be at most 1. Small fitting coefficients are good since they will not increase the noise by too much. For example, suppose $A_i^* = A_S^* x$ and $\|x\|_\infty \leq 1$, i.e., the i-th column can be fit by the columns with indices in the set S and the fitting coefficients $x \in \mathbb{R}^{|S|}$ are small. If we use the noisy columns of $A_S^* + \Delta_S$ to fit the noisy column $A_i^* + \Delta_i$, then the fitting cost is at most

$$\|(A_S^* + \Delta_S)x - (A_i^* + \Delta_i)\|_1 \leq \|\Delta_i\|_1 + \|\Delta_S x\|_1.$$

Since $\|x\|_\infty \leq 1$, it is possible to give a good upper bound for $\|\Delta_S x\|_1$.

Definition 19.2.1 (Tuple)**.** A $(q, t, n)-$tuple is defined to be $(S_1, S_2, \cdots, S_t, i)$, where $\forall j \in [t], S_j \subset [n]$ with $|S_j| = q$. Let $S = \bigcup_{j=1}^t S_j$. Then $|S| = qt$, i.e., $S_1, S_2, \cdots, S_t$ are disjoint. Furthermore, $i \in [n]$ and $i \notin S$. For simplicity, we use $(S_{[t]}, i)$ to denote $(S_1, S_2, \cdots, S_t, i)$.

We next provide the definition of a good tuple.

Definition 19.2.2 (Good tuple)**.** Given a rank-k matrix $A^* \in \mathbb{R}^{n \times n}$, an (A^*, q, t, α)-good tuple is a (q, t, n)-tuple $(S_{[t]}, i)$ which satisfies

$$|\{j \in [t] \mid i \notin R_{A^*}(S_j \cup \{i\})\}| \geq \alpha \cdot t.$$

We need the definition of the core of a tuple.

Definition 19.2.3 (Core of a tuple)**.** The core of $(S_{[t]}, i)$ is defined to be the set

$$\{j \in [t] \mid i \notin R_{A^*}(S_j \cup \{i\})\}.$$

We define a coefficients tuple as follows.

Definition 19.2.4 (Coefficients tuple)**.** Given a rank-k matrix $A^* \in \mathbb{R}^{n\times n}$, let $(S_{[t]}, i)$ be an (A^*, q, t, α)-good tuple. Let C be the core of $(S_{[t]}, i)$. A coefficients tuple corresponding to $(S_{[t]}, i)$ is defined to be $(x_1, x_2, \cdots, x_t)$ where $\forall j \in [t], x_j \in \mathbb{R}^q$. The vector $x_j \in \mathbb{R}^q$ satisfies: $x_j = 0$ if $j \in [t]\backslash C$, while $A^*_{S_j} x_j = A^*_i$ and $\|x_j\|_\infty \leq 1$, if $j \in C$. To guarantee the coefficients tuple is unique, we restrict each vector $x_j \in \mathbb{R}^q$ to be one that has the minimum lexicographic order.

19.2.3 Properties of a Good Tuple and a Coefficients Tuple

Consider a good tuple $(S_1, S_2, \cdots, S_t, i)$. By the definition of a good tuple, the size of the core C of the tuple is large. For each $j \in C$, the coefficients x_j of using $A^*_{S_j}$ to fit A^*_i should have absolute value at most 1. Now consider the noisy setting. As discussed in the previous section, using A_{S_j} to fit A_i has cost at most $\|\Delta_i\|_1 + \|\Delta_{S_j} x_j\|_1$. Although $\|\Delta_{S_j} x_j\|_1$ has a good upper bound, it is not small enough. To further reduce the ℓ_1 fitting cost, we can now apply the averaging argument (Lemma 19.2.2) over all the fitting choices corresponding to C. Formally, we have the following lemma.

Lemma 19.2.6 (Good tuples imply low fitting cost)**.** *Suppose we are given a matrix $A \in \mathbb{R}^{n\times n}$ which satisfies $A = A^* + \Delta$, where $A^* \in \mathbb{R}^{n\times n}$ has rank k. Here $\Delta \in \mathbb{R}^{n\times n}$ is a random matrix where $\Delta_{i,j}$ are i.i.d. symmetric random variables with $\mathbb{E}[|\Delta_{i,j}|] = 1$ and $\mathbb{E}[|\Delta_{i,j}|^p] = O(1)$ for some constant $p \in (1, 2)$. Let $H \subset [n]$ be defined as follows:*

$$H = \left\{ j \in [n] \;\middle|\; \exists i \in [n], |\Delta_{i,j}| > n^{1/2+1/(2p)} \right\}.$$

Let $q, t \leq n^{o(1)}$. Then, with probability at least $1 - 2^{-n^{\Theta(1)}}$, for all $(A^*, q, t, 1/2)$-good tuples

$(S_1, S_2, \cdots, S_t, i)$ *which satisfy* $H \cap \left(\bigcup_{j=1}^{t} S_j\right) = \emptyset$, *we have*

$$\min_{y \in \mathbb{R}^{qt}} \left\| A_{\{\bigcup_{j=1}^{t} S_j\}} y - A_i \right\|_1 \leq \left\| \frac{1}{|C|} \sum_{j=1}^{t} A_{S_j} x_j - A_i \right\|_1 \leq \|\Delta_i\|_1 + O(q^{1/p}/t^{1-1/p} n),$$

where C *is the core of* $(S_1, S_2, \cdots, S_t, i)$, *and* $(x_1, x_2, \cdots, x_t)$ *is the coefficients tuple corresponding to* $(S_1, S_2, \cdots, S_t, i)$.

We next show that if we choose columns randomly, it is easy to find a good tuple.

Lemma 19.2.7. *Given a rank-k matrix* $A^* \in \mathbb{R}^{n \times n}$, *let* $q > 10k, t > 0$. *Let* $I = \{i_1, i_2, \cdots, i_{qt+1}\}$ *be a subset drawn uniformly at random from* $\binom{[n]}{qt+1}$. *Let* $\pi : I \to I$ *be a random permutation of* $qt + 1$ *elements.* $\forall j \in [t]$, *let*

$$S_j = \left\{ i_{\pi((j-1)q+1)}, i_{\pi((j-1)q+2)}, \cdots, i_{\pi((j-1)q+q)} \right\}.$$

We use i *to denote* $i_{\pi(qt+1)}$. *With probability* $\geq 1 - 2k/q$, $(S_1, S_2, \cdots, S_t, i)$ *is an* $(A^*, q, t, 1/2)-$*good tuple.*

Lemma 19.2.7 implies that if we randomly choose $S_1, S_2, \cdots, S_t$, then with high probability, there are many choices of $i \in [n]$, such that $(S_1, S_2, \cdots, S_t, i)$ is a good tuple. Precisely, we can show the following.

Lemma 19.2.8. *Given a rank-k matrix* $A^* \in \mathbb{R}^{n \times n}$, *let* $q > 10k, t > 0$. *Let* $I = \{i_1, i_2, \cdots, i_{qt}\}$ *be a random subset uniformly drawn from* $\binom{[n]}{qt}$. *Let* π *be a random permutation of* qt *elements.* $\forall j \in [t]$, *we define* S_j *as follows:*

$$S_j = \left\{ i_{\pi((j-1)q+1)}, i_{\pi((j-1)q+2)}, \cdots, i_{\pi((j-1)q+q)} \right\}.$$

Then with probability at least $2k/q$,

$$\left| \left\{ i \in [n] \setminus I \;\middle|\; (S_1, S_2, \cdots, S_t, i) \text{ is an } (A^*, q, t, 1/2)-\text{good tuple } \right\} \right| \geq (1 - 4k/q)(n - qt).$$

19.2.4 Main Result

Now we are able to put all ingredients together to prove our main theorem, Theorem 19.2.9.

Theorem 19.2.9 (Formal version of Theorem 19.1.1)**.** *Suppose we are given a matrix* $A = A^* + \Delta \in \mathbb{R}^{n\times n}$*, where* $\mathrm{rank}(A^*) = k$ *for* $k = n^{o(1)}$*, and* Δ *is a random matrix for which the* $\Delta_{i,j}$ *are i.i.d. symmetric random variables with* $\mathbb{E}[|\Delta_{i,j}|] = 1$ *and* $\mathbb{E}[|\Delta_{i,j}|^p] = O(1)$ *for some constant* $p \in (1,2)$*. Let* $\epsilon \in (0, 1/2)$ *satisfy* $1/\epsilon = n^{o(1)}$*. There is an* $\widetilde{O}(n^2 + n\,\mathrm{poly}(k/\epsilon))$ *time algorithm (Algorithm 19.1) which can output a subset* $S \in [n]$ *with* $|S| \leq \mathrm{poly}(k/\epsilon) + O(k \log n)$ *for which*

$$\min_{X\in\mathbb{R}^{|S|\times n}} \|A_S X - A\|_1 \leq (1+\epsilon)\|\Delta\|_1,$$

holds with probability at least $99/100$*.*

Proof. We discussed the running time at the beginning of Section 19.2. Next, we turn to correctness. Let

$$q = \Omega\left(\frac{k(k\log k)^{1+\frac{1}{p-1}}}{\epsilon^{1+\frac{1}{p-1}}}\right), \quad t = \frac{q^{\frac{1}{p-1}}}{\epsilon^{1+\frac{1}{p-1}}}.$$

Let $r = \Theta(q/k)$. Let

$$I_1 = \left\{i_1^{(1)}, i_2^{(1)}, \cdots, i_{qt}^{(1)}\right\}, I_2 = \left\{i_1^{(2)}, i_2^{(2)}, \cdots, i_{qt}^{(2)}\right\}, \cdots, I_r = \left\{i_1^{(r)}, i_2^{(r)}, \cdots, i_{qt}^{(r)}\right\},$$

be r independent subsets drawn uniformly at random from $\binom{[n]}{qt}$. Let $I = \bigcup_{s\in[r]} I_s$, which is the same as that in Algorithm 19.1. Let $\pi_1, \pi_2, \cdots, \pi_r$ be r independent random permutations

of qt elements. Due to Lemma 19.2.8 and a Chernoff bound, with probability at least .999, $\exists s \in [r]$,

$$\left|\left\{i \in [n] \setminus I_s \;\middle|\; (S_1, S_2, \cdots, S_t, i) \text{ is an } (A^*, q, t, 1/2)-\text{good tuple } \right\}\right| \geq (1 - 4k/q)(n - qt)$$

where

$$S_j = \left\{i^{(s)}_{\pi_s((j-1)q+1)}, i^{(s)}_{\pi_s((j-1)q+2)}, \cdots, i^{(s)}_{\pi_s((j-1)q+q)}\right\}, \forall j \in [t].$$

Let set $H \subset [n]$ be defined as follows:

$$H = \{j \in [n] \mid \exists i \in [n], |\Delta_{i,j}| > n^{1/2+1/(2p)}\}.$$

Then due to Lemma 19.2.3, with probability at least 0.999, $|H| \leq O(n^{1-(p-1)/2})$. Thus, for $j \in [r]$, the probability that $H \cap I_j \neq \emptyset$ is at most $O(qt \cdot n^{1-(p-1)/2}/(n - qt)) = 1/n^{\Omega(1)}$. By taking a union bound over all $j \in [r]$, with probability at least $1 - 1/n^{\Omega(1)}$, $\forall j \in [r], I_j \cap H = \emptyset$. Thus, we can condition on $I_s \cap H = \emptyset$. Due to Lemma 19.2.6 and $q^{1/p}/t^{1-1/p} = \epsilon$,

$$\left|\left\{i \in [n] \setminus I_s \;\middle|\; \min_{y \in \mathbb{R}^{qt}} \|A_{I_s} y - A_i\|_1 \leq \|\Delta_i\|_1 + O(\epsilon n)\right\}\right| \geq (1 - 4k/q)(n - qt).$$

Due to Lemma 19.2.5 and a union bound over all $i \in [n] \setminus H$, with probability at least .999, $\forall i \notin H, \|\Delta_i\| \leq (1 + \epsilon)n$. Thus,

$$\left|\left\{i \in [n] \setminus I_s \;\middle|\; \min_{y \in \mathbb{R}^{qt}} \|A_{I_s} y - A_i\|_1 \leq (1 + O(\epsilon))n\right\}\right| \geq (1 - 4k/q)(n - qt) - |H|.$$

Let

$$T' = [n] \setminus \left\{i \in [n] \;\middle|\; \min_{y \in \mathbb{R}^{qt}} \|A_{I_s} y - A_i\|_1 \leq (1 + O(\epsilon))n\right\}.$$

Then $|T'| \leq O(kn/q + n^{1-(p-1)/2}) = O(kn/q) = O((\epsilon/(k \log k))^{1+1/(p-1)} n)$. By our selection of T in algorithm 19.1, T' should be a subset of T. Due to Lemma 19.2.4, with probability at least .999, $\|\Delta_T\|_1 \leq O(\epsilon n^2/(k \log k))$. By our second subroutine mentioned at the beginning of Section 19.2 it can find a set $Q \subset [n]$ with $|Q| = O(k \log n)$ such that

$$\min_{X \in \mathbb{R}^{|Q| \times |T|}} \|A_Q X - A_T\|_1 \leq O(k \log k) \|\Delta_T\|_1 \leq O(\epsilon n^2).$$

Thus, we have

$$\begin{aligned} \min_{X \in \mathbb{R}^{(|Q|+q \cdot t \cdot r) \times n}} \|A_{(Q \cup I)} X - A\|_1 \leq & \min_{X_1 \in \mathbb{R}^{(q \cdot t \cdot r) \times n}} \|A_I X_1 - A_{[n] \setminus T}\|_1 + \min_{X_2 \in \mathbb{R}^{|Q| \times n}} \|A_Q X_2 - A_T\|_1 \\ & \leq (1 + O(\epsilon)) n^2. \end{aligned}$$

Due to Lemma 19.2.1, with probability at least .999, $\|\Delta\|_1 \geq (1 - \epsilon) n^2$, and thus

$$\min_{X \in \mathbb{R}^{(|Q|+q \cdot t \cdot r) \times n}} \|A_{(Q \cup I)} X - A\|_1 \leq (1 + O(\epsilon)) \|\Delta\|_1.$$

□

19.3 Missing Proofs in Section 19.2

19.3.1 Proof of Lemma 19.2.1

Proof. Let $Z \in \mathbb{R}^{n\times n}$ be a random matrix. For each $i, j \in [n]$, define random variable $Z_{i,j}$ as

$$Z_{i,j} = \begin{cases} |\Delta_{i,j}|, & \text{if } |\Delta_{i,j}| \le n; \\ n, & \text{otherwise.} \end{cases}$$

For $i, j \in [n]$, by Markov's inequality, we have

$$\Pr[|\Delta_{i,j}| \ge n] = \Pr[|\Delta_{i,j}|^p \ge n^p] \le \mathbb{E}[|\Delta_{i,j}|^p]/n^p = O(1/n^p). \tag{19.1}$$

Notice that

$$\mathbb{E}[|\Delta_{i,j}|^p] = \int_0^n x^p f(x)\mathrm{d}x + \int_n^\infty x^p f(x)\mathrm{d}x = O(1)$$

where $f(x)$ is the probability density function of $|\Delta_{i,j}|$. Thus we have

$$\int_n^\infty x f(x)\mathrm{d}x \le \int_n^\infty x^p/n^{p-1} \cdot f(x)\mathrm{d}x = O(1/n^{p-1}).$$

Because $\mathbb{E}[|\Delta_{i,j}|] = 1$, we have

$$\int_0^\infty x f(x)\mathrm{d}x = \mathbb{E}[|\Delta_{i,j}|] - \int_n^\infty x f(x)\mathrm{d}x \ge 1 - O(1/n^{p-1}). \tag{19.2}$$

By Equation (19.2), we have

$$\mathbb{E}[Z_{i,j}] = \int_0^n x f(x)dx + n \cdot \Pr[|\Delta_{i,j}| \ge n] \ge \int_0^n x f(x)dx \ge 1 - O(1/n^{p-1}).$$

By Equation (19.1) and $E[|\Delta_{i,j}|^p] \le O(1)$, we have

$$\mathbb{E}[Z_{i,j}^2] = \int_0^n x^2 f(x)\mathrm{d}x + n^2 \Pr[|\Delta_{i,j}| \ge n] \le O(n^{2-p}) + O(n^{2-p}) = O(n^{2-p}).$$

By the inequality of [Mau03],

$$\begin{aligned}\Pr[\mathbb{E}[\|Z\|_1] - \|Z\|_1 \geq \epsilon\,\mathbb{E}[\|Z\|_1]/2] &\leq \exp\left(\frac{-\epsilon^2\,\mathbb{E}[\|Z\|_1]^2/4}{2\sum_{i,j}\mathbb{E}[Z_{i,j}^2]}\right)\\ &\leq \exp\left(\frac{-\epsilon^2(n^2 - O(n^{3-p}))^2/4}{2n^2\cdot O(n^{2-p})}\right)\\ &\leq e^{-\Theta(n)}\end{aligned}$$

Thus with probability at least $1 - e^{-\Theta(n)}$, $\|Z\|_1 \geq (1-\epsilon/2)\,\mathbb{E}[\|Z\|_1] \geq (1-\epsilon)n^2$ where the last inequality follows by $\mathbb{E}[\|Z\|_1 \geq n^2 - O(n^{3-p})]$ and $1/\epsilon = n^{o(1)}$. Since $\|\Delta\|_1 \geq \|Z\|_1$, we complete the proof. □

19.3.2 Proof of Lemma 19.2.2

Proof. Let $Z \in \mathbb{R}^{n\times t}$ be a random matrix where $Z_{i,j}$ are i.i.d. random variables with probability density function:

$$g(x) = \begin{cases} f(x)/\Pr[|\Delta_{1,1}| \leq n^{1/2+1/(2p)}], & \text{if } |x| \leq n^{1/2+1/(2p)};\\ 0, & \text{otherwise.}\end{cases}$$

where $f(x)$ is the probability density function of $\Delta_{1,1}$. (Note that in the above equation, $\Pr[|\Delta_{1,1}| \leq n^{1/2+1/(2p)}] > 0$.) Now, we have $\forall a \geq 0$,

$$\Pr\left[\left\|\sum_{j=1}^{t}\alpha_j\Delta_j\right\|_1 \leq a \;\middle|\; \forall i\in[n], j\in[t], |\Delta_{i,j}| \leq n^{1/2+1/(2p)}\right] = \Pr\left[\left\|\sum_{j=1}^{t}\alpha_j Z_j\right\|_1 \leq a\right].$$

Now we look at the i-th row of $\sum_{j=1}^{t} \alpha_j Z_j$. We have

$$\begin{aligned}
\mathbb{E}\left[\left|\sum_{j=1}^{t} \alpha_j Z_{i,j}\right|\right] &= \left(\mathbb{E}\left[\left|\sum_{j=1}^{t} \alpha_j Z_{i,j}\right|\right]^p\right)^{1/p} \\
&\leq \mathbb{E}\left[\left|\sum_{j=1}^{t} \alpha_j Z_{i,j}\right|^p\right]^{1/p} \\
&\leq \mathbb{E}\left[\left(\left(\sum_{j=1}^{t} \alpha_j^2 Z_{i,j}^2\right)^{1/2}\right)^p\right]^{1/p} \\
&\leq \mathbb{E}\left[\sum_{j=1}^{t} |\alpha_j Z_{i,j}|^p\right]^{1/p} \\
&\leq \left(\sum_{j=1}^{t} \mathbb{E}[|\alpha_j Z_{i,j}|^p]\right)^{1/p} \\
&\leq \left(\sum_{j=1}^{t} \mathbb{E}[|Z_{i,j}|^p]\right)^{1/p} \\
&\leq O(t^{1/p}), \qquad (19.3)
\end{aligned}$$

where the first inequality follows by Jensen's inequality, the second inequality follows by Remark 3 of [Lat97], the third inequality follows by $\|x\|_2 \leq \|x\|_p$ for $p < 2$, the fourth inequality follows by $|\alpha_j| \leq 1$, the fifth inequality follows by $\mathbb{E}[|Z_{i,j}|^p] = \mathbb{E}[|\Delta_{i,j}|^p \mid |\Delta_{1,1}| \leq$

$n^{1/2+1/(2p)}] \leq \mathbb{E}[|\Delta_{i,j}|^p] = O(1)$. For the second moment, we have

$$\begin{aligned}
\mathbb{E}\left[\left|\sum_{j=1}^{t} \alpha_j Z_{i,j}\right|^2\right] &= \sum_{j=1}^{t} \mathbb{E}\left[\alpha_j^2 Z_{i,j}^2\right] + \sum_{j\neq k} \mathbb{E}[\alpha_j \alpha_k Z_{i,j} Z_{i,k}] \\
&= \sum_{j=1}^{t} \alpha_j^2 \,\mathbb{E}\left[Z_{i,j}^2\right] + \sum_{j\neq k} \alpha_j \alpha_k \,\mathbb{E}[Z_{i,j}]\,\mathbb{E}[Z_{i,k}] \\
&\leq \sum_{j=1}^{t} \mathbb{E}\left[Z_{i,j}^2\right] \\
&= t \cdot 2 \int_0^{n^{1/2+1/(2p)}} x^2 f(x) / \Pr\left[|\Delta_{i,j}| \leq n^{1/2+1/(2p)}\right] \mathrm{d}x \\
&\leq 2t / \Pr\left[|\Delta_{i,j}| \leq n^{1/2+1/(2p)}\right] \cdot (n^{1/2+1/(2p)})^{2-p} \int_0^{n^{1/2+1/(2p)}} x^p f(x) \mathrm{d}x \\
&\leq O(tn^{2-p}), \qquad (19.4)
\end{aligned}$$

where the second inequality follows by independence of $Z_{i,j}$ and $Z_{i,k}$. The first inequality follows by $|\alpha_j| \leq 1$ and $\mathbb{E}[Z_{i,j}] = \mathbb{E}[Z_{i,k}] = 0$. The third equality follows by the probability density function of $Z_{i,j}$. The second inequality follows by $x^{2-p} \leq (n^{1/2+1/(2p)})^{2-p}$ when $0 \leq x \leq n^{1/2+1/(2p)}$. The last inequality follows by $\mathbb{E}[|\Delta_{i,j}|^p] = O(1), p > 1$ and $\Pr[|\Delta_{i,j}| \leq n^{1/2+1/(2p)}] \geq 1 - \mathbb{E}[|\Delta_{i,j}|^p]/(n^{1/2+1/(2p)})^p = 1 - O(1/n^{p/2+1/2}) \geq 1/2$.

For $i \in [n]$, define $X_i = |\sum_{j=1}^{t} \alpha_j Z_{i,j}|$. Then, by Bernstein's inequality

$$\begin{aligned}
&\Pr\left[\left\|\sum_{j=1}^{t} \alpha_j Z_j\right\|_1 - \mathbb{E}\left[\left\|\sum_{j=1}^{t} \alpha_j Z_j\right\|_1\right] \geq 0.5t^{1/p}n\right] \\
=\ &\Pr\left[\sum_{i=1}^{n} X_i - \mathbb{E}\left[\sum_{i=1}^{n} X_i\right] \geq 0.5t^{1/p}n\right] \\
\leq\ &\exp\left(-\frac{0.5 \cdot 0.5^2 t^{2/p} n^2}{\sum_{i=1}^{n} \mathbb{E}[X_i^2] + \frac{1}{3} n^{1/2+1/(2p)} \cdot 0.5t^{1/p}n}\right) \\
\leq\ &e^{-n^{\Theta(1)}}.
\end{aligned}$$

The last inequality follows by Equation (19.4). According to Equation (19.3), with probability at least $1 - e^{-n^{\Theta(1)}}$,

$$\left\| \sum_{j=1}^{t} \alpha_j Z_j \right\|_1 \leq \mathbb{E}\left[\left\| \sum_{j=1}^{t} \alpha_j Z_j \right\|_1 \right] + 0.5 t^{1/p} n \leq O(t^{1/p} n).$$

□

19.3.3 Proof of Lemma 19.2.3

Proof. For $i, j \in [n]$, we have

$$\begin{aligned} \Pr\left[|\Delta_{i,j}| > n^{1/2+1/(2p)}\right] &= \Pr\left[|\Delta_{i,j}|^p > n^{p/2+1/2}\right] \\ &\leq \mathbb{E}\left[|\Delta_{i,j}|^p\right] / n^{p/2+1/2} \\ &\leq O(1/n^{p/2+1/2}). \end{aligned}$$

For column j, by taking a union bound,

$$\Pr[j \in H] = \Pr\left[\exists i \in [n], |\Delta_{i,j}| > n^{1/2+1/(2p)}\right] \leq O(1/n^{p/2-1/2}).$$

Thus, $\mathbb{E}[|H|] \leq O(n^{1-(p-1)/2})$. By applying Markov's inequality, we complete the proof. □

19.3.4 Proof of Lemma 19.2.4

Proof. For $l \in \mathbb{N}_{\geq 0}$, define $G_l = \{j \mid \|\Delta_j\|_1 \in (n \cdot 2^l, n \cdot 2^{l+1}]\}$. We have

$$\begin{aligned}
\mathbb{E}[|G_l|] &\leq \sum_{j=1}^{n} \Pr\left[\|\Delta_j\|_1 \geq n \cdot 2^l\right] \\
&= n \Pr\left[\|\Delta_1\|_1 \geq n \cdot 2^l\right] \\
&\leq n \Pr\left[n^{1-1/p}\|\Delta_1\|_p \geq n \cdot 2^l\right] \\
&= n \Pr\left[n^{p-1}\|\Delta_1\|_p^p \geq n^p \cdot 2^{lp}\right] \\
&\leq n \mathbb{E}\left[n^{p-1}\|\Delta_1\|_p^p\right] / (n^p \cdot 2^{lp}) \\
&\leq O(n/2^{lp}).
\end{aligned}$$

The first inequality follows by the definition of G_l. The second inequality follows since $\forall x \in \mathbb{R}^n, \|x\|_1 \leq n^{1-1/p}\|x\|_p$. The third inequality follows by Markov's inequality. The last inequality follows since $\forall i, j \in [n], \mathbb{E}[|\Delta_{i,j}|^p] = O(1)$.

Let $l^* \in \mathbb{N}_{\geq 0}$ satisfy $2^{l^*} < \epsilon r$ and $2^{l^*+1} \geq \epsilon r$. We have

$$\begin{aligned}
\mathbb{E}\left[\sum_{j:\|\Delta_j\|_1 \geq n2^{l^*}} \|\Delta_j\|_1\right] &\leq \mathbb{E}\left[\sum_{l=l^*}^{\infty} |G_l| \cdot n2^{l+1}\right] \\
&= \sum_{l=l^*}^{\infty} \mathbb{E}[|G_l|] \cdot n2^{l+1} \\
&\leq \sum_{l=l^*}^{\infty} O(n/2^{lp}) \cdot n2^{l+1} \\
&= \sum_{l=l^*}^{\infty} O(n^2/2^{l(p-1)}) \\
&= O(n^2/2^{l^*(p-1)}) \\
&= O(n^2/(\epsilon r)^{p-1}) \\
&= O(\epsilon n^2).
\end{aligned}$$

By Markov's inequality, with probability at least .999, $\sum_{j:\|\Delta_j\|_1 \geq n2^{l^*}} \|\Delta_j\|_1 \leq O(\epsilon n^2)$. Conditioned on $\sum_{j:\|\Delta_j\|_1 \geq n2^{l^*}} \|\Delta_j\|_1 \leq O(\epsilon n^2)$, for any $S \subset [n]$ with $|S| \leq n/r$, we have

$$\sum_{j \in S} \|\Delta_j\|_1 \leq |S| \cdot n2^{l^*} + \sum_{j:\|\Delta_j\|_1 \geq n2^{l^*}} \|\Delta_j\|_1 \leq \epsilon n^2 + O(\epsilon n^2) = O(\epsilon n^2).$$

The second inequality follows because $|S| \leq n/r, 2^{l^*} \leq \epsilon r$ and $\sum_{j:\|\Delta_j\|_1 \geq n2^{l^*}} \|\Delta_j\|_1 \leq O(\epsilon n^2)$.

□

19.3.5 Proof of Lemma 19.2.5

Proof. Let $M = n^{1/2+1/(2p)}$. Let $Z \in \mathbb{R}^n$ be a random vector where Z_i are i.i.d. random variables with probability density function

$$g(x) = \begin{cases} f(x)/\Pr[|\Delta_1| \leq M] & \text{if } 0 \leq x \leq M; \\ 0 & \text{otherwise.} \end{cases}$$

where $f(x)$ is the probability density function of $|\Delta_1|$. Then $\forall a > 0$

$$\Pr\left[\|\Delta\|_1 \le a \mid \forall i \in [n], |\Delta_i| \le M\right] = \Pr\left[\|Z\|_1 \le a\right].$$

For $i \in [n]$, because $\mathbb{E}[|\Delta_i|] = 1$, it holds that $\mathbb{E}[Z_i] \le 1$. We have $\mathbb{E}[\sum_{i=1}^{n} Z_i] \le n$. For the second moment, we have

$$\begin{aligned}
\mathbb{E}[Z_i^2] &= \int_0^M x^2 f(x) / \Pr[|\Delta_1| \le M] \mathrm{d}x \\
&\le M^{2-p} / \Pr[|\Delta_1| \le M] \int_0^M x^p f(x) \mathrm{d}x \\
&\le O(M^{2-p}) \\
&\le O(n^{2-p})
\end{aligned}$$

where the second inequality follows by $\mathbb{E}[|\Delta_1|^p] = O(1)$, and

$$\Pr[|\Delta_1| \le M] \ge 1 - \mathbb{E}[|\Delta_1|^p]/M^p \ge 1/2.$$

Then by Bernstein's inequality, we have

$$\begin{aligned}
& \Pr\left[\sum_{i=1}^{n} Z_i - E\left[\sum_{i=1}^{n} Z_i\right] \ge \epsilon n\right] \\
\le\ & \exp\left(\frac{-0.5\epsilon^2 n^2}{\sum_{i=1}^{n} \mathbb{E}[Z_i^2] + \frac{1}{3} M \cdot \epsilon n}\right) \\
\le\ & e^{-n^{\Theta(1)}}.
\end{aligned}$$

Thus,

$$\Pr\left[\|\Delta\|_1 \le (1+\epsilon)n \mid \forall i \in [n], |\Delta_i| \le M\right] = \Pr\left[\|Z\|_1 \le (1+\epsilon)n\right] \ge 1 - e^{-n^{\Theta(1)}}.$$

□

19.3.6 Proof of Lemma 19.2.6

Proof. Recall that $(S_1, S_2, \cdots, S_t, i)$ is equivalent to $(S_{[t]}, i)$. Let $(S_{[t]}, i)$ be an $(A^*, q, t, 1/2)$-good tuple which satisfies $H \cap \left(\bigcup_{j=1}^{t} S_j\right) = \emptyset$. Let C be the core of $(S_{[t]}, i)$. Let $(x_1, x_2, \cdots, x_t)$ be the coefficients tuple corresponding to $(S_{[t]}, i)$. Then we have that

$$
\begin{aligned}
& \left\| \frac{1}{|C|} \sum_{j=1}^{t} A_{S_j} x_j - A_i \right\|_1 \\
= & \left\| \frac{1}{|C|} \sum_{j=1}^{t} \left(A^*_{S_j} + \Delta_{S_j} \right) x_j - (A^*_i + \Delta_i) \right\|_1 \\
\leq & \left\| \frac{1}{|C|} \sum_{j=1}^{t} A^*_{S_j} x_j - A^*_i \right\|_1 + \|\Delta_i\|_1 + \frac{1}{|C|} \left\| \sum_{j=1}^{t} \Delta_{S_j} x_j \right\|_1 \\
= & \|\Delta_i\|_1 + \frac{1}{|C|} \left\| \sum_{j=1}^{t} \Delta_{S_j} x_j \right\|_1 \\
\leq & \|\Delta_i\|_1 + \frac{2}{t} \left\| \sum_{j=1}^{t} \Delta_{S_j} x_j \right\|_1 \\
\leq & \|\Delta_i\|_1 + O\left(\frac{1}{t} \cdot (qt)^{1/p} n \right) \\
= & \|\Delta_i\|_1 + O\left(q^{1/p} / t^{1-1/p} n \right)
\end{aligned}
$$

holds with probability at least $1 - 2^{-n^{\Theta(1)}}$. The first equality follows using $A = A^* + \Delta$. The first inequality follows using the triangle inequality. The second equality follows using the definition of the core and the coefficients tuple (see Definition 19.2.2 and Definition 19.2.4). The second inequality follows using Definition 19.2.2. The third inequality follows by Lemma 19.2.2 and the condition that $H \cap \left(\bigcup_{j=1}^{t} S_j\right) = \emptyset$.

Since the size of $\left|\{i\} \cup \left(\bigcup_{j=1}^{t} S_j\right)\right| = qt + 1$, the total number of $(A^*, q, t, 1/2)-$good tuples is upper bounded by $n^{qt+1} \leq 2^{n^{o(1)}}$. By taking a union bound, we complete the

proof. □

19.3.7 Proof of Lemma 19.2.7

Proof. For $j \in [t]$, by symmetry of the choices of S_j and i, we have $\Pr[i \in R_{A^*}(S_j \cup \{i\})] \leq k/(q+1)$. Thus, by Markov's inequality,

$$\begin{aligned} & \Pr[|\{j \in [t] \mid i \in R_{A^*}(S_j \cup \{i\})\}| > 0.5t] \\ \leq\ & \mathbb{E}[|\{j \in [t] \mid i \in R_{A^*}(S_j \cup \{i\})\}|]/(0.5t) \\ \leq\ & 2k/q. \end{aligned}$$

Thus,

$$\Pr[|\{j \in [t] \mid i \notin R_{A^*}(S_j \cup \{i\})\}| \geq 0.5t] \geq 1 - 2k/q.$$

□

19.3.8 Proof of Lemma 19.2.8

Proof. For $S_1, S_2, \cdots, S_t \in \binom{[n]}{q}$ with $\sum_{j=1}^{t} |S_j| = qt$, define

$$P_{(S_1,S_2,\cdots,S_t)} = \Pr_{i \in [n] \setminus \left(\bigcup_{j=1}^{t} S_j\right)} [(S_1, S_2, \cdots, S_t, i) \text{ is an } (A^*, q, t, 1/2)-\text{good tuple }].$$

Let set T be defined as follows:

$$\left\{ (S_1, S_2, \cdots, S_t) \;\middle|\; S_1, S_2, \cdots, S_t \in \binom{[n]}{q} \text{ with } \sum_{j=1}^{t} |S_j| = qt \right\}.$$

Let G be the set of all the $(A^*, q, t, 1/2)-$good tuples. Then, we have

$$\begin{aligned}
&\Pr_{(S_1,S_2,\cdots,S_t)\sim T}\left[\left|\left\{i \in [n] \setminus \left(\cup_{j=1}^{t} S_j\right) \mid (S_1, S_2, \cdots, S_t, i) \in G\right\}\right| \geq (1-4k/q)(n-qt)\right] \\
&= \frac{1}{|T|}\left|\left\{(S_1, S_2, \cdots, S_t) \mid (S_1, S_2, \cdots, S_t) \in T \text{ and } P_{(S_1,S_2,\cdots,S_t)} \geq 1-4k/q\right\}\right| \\
&= \frac{1}{|T|} \sum_{\substack{(S_1,S_2,\cdots,S_t)\in T \\ P_{(S_1,S_2,\cdots,S_t)}\geq 1-4k/q}} 1 \\
&\geq \frac{1}{|T|} \sum_{\substack{(S_1,S_2,\cdots,S_t)\in T \\ P_{(S_1,S_2,\cdots,S_t)}\geq 1-4k/q}} P_{(S_1,S_2,\cdots,S_t)} \\
&\geq 1 - 2k/q - \frac{1}{|T|} \sum_{\substack{(S_1,S_2,\cdots,S_t)\in T \\ P_{(S_1,S_2,\cdots,S_t)}< 1-4k/q}} P_{(S_1,S_2,\cdots,S_t)} \\
&\geq 1 - 2k/q - (1-4k/q) \\
&\geq 2k/q.
\end{aligned}$$

The second inequality follows from Lemma 19.2.7

$$\frac{1}{|T|} \sum_{\substack{(S_1,S_2,\cdots,S_t)\in T \\ P_{(S_1,S_2,\cdots,S_t)}< 1-4k/q}} P_{(S_1,S_2,\cdots,S_t)} + \frac{1}{|T|} \sum_{\substack{(S_1,S_2,\cdots,S_t)\in T \\ P_{(S_1,S_2,\cdots,S_t)}\geq 1-4k/q}} P_{(S_1,S_2,\cdots,S_t)} \geq 1 - 2k/q.$$

□

19.4 Hardness Result

An overview of the hardness result. Recall that we overcame the column subset selection lower bound of [SWZ17], which shows for entrywise ℓ_1-low rank approximation that there are matrices for which any subset of $\text{poly}(k)$ columns spans at best a $k^{\Omega(1)}$-approximation. Indeed, we came up with a column subset of size $\text{poly}(k(\epsilon^{-1} + \log n))$ spanning a $(1+\epsilon)$-approximation. To do this, we assumed $A = A^* + \Delta$, where A^* is an arbitrary rank-k matrix, and the entries are i.i.d. from a distribution with $\mathbb{E}[|\Delta_{i,j}|] = 1$ and $\mathbb{E}[|\Delta_{i,j}|^p] = O(1)$ for any real number p strictly greater than 1.

Here we show an assumption on the moments is necessary, by showing if instead Δ were drawn from a matrix of i.i.d. Cauchy random variables, for which the p-th moment is undefined or infinite for all $p \geq 1$, then for any subset of $n^{o(1)}$ columns, it spans at best a 1.002 approximation. The input matrix $A = n^C 1 \cdot 1^\top + \Delta$, where $C > 0$ is a constant and we show that $n^{\Omega(1)}$ columns need to be chosen to obtain a 1.001-approximation, even for $k = 1$. Note that this result is stronger than that in [SWZ17] in that it rules out column subset selection even if one were to choose $n^{o(1)}$ columns; the result in [SWZ17] requires at most $\text{poly}(k)$ columns, which for $k = 1$, would just rule out $O(1)$ columns. Our main goal here is to show that a moment assumption on our distribution is necessary, and our result also applies to a symmetric noise distribution which is i.i.d. on all entries, whereas the result of [SWZ17] requires a specific deterministic pattern (namely, the identity matrix) on certain entries.

Our main theorem is given in Theorem 19.4.17. The outline of the proof is as follows. We first condition on the event that $\|\Delta\|_1 \leq \frac{4.0002}{\pi} n^2 \ln n$, which is shown in Lemma 19.4.2 and follows form standard analysis of sums of absolute values of Cauchy random variables.

Thus, it is sufficient to show if we choose any subset S of $r = n^{o(1)}$ columns, denoted by the submatrix A_S, then $\min_{X \in \mathbb{R}^{r \times n}} \|A_S X - A\|_1 \geq \frac{4.01}{\pi} \cdot n^2 \ln n$, as indeed then $\min_{X \in \mathbb{R}^{r \times n}} \|A_S X - A\|_1 \geq 1.002 \|\Delta\|_1$ and we rule out a $(1+\epsilon)$-approximation for ϵ a sufficiently small constant. To this end, we instead show for a fixed S, that $\min_{X \in \mathbb{R}^{r \times n}} \|A_S X - A\|_1 \geq \frac{4.01}{\pi} \cdot n^2 \ln n$ with probability $1 - 2^{-n^{\Theta(1)}}$, and then apply a union bound over all S. To prove this for a single subset S, we argue that for every "coefficient matrix" X, that $\|A_S X - A\|_1 \geq \frac{4.01}{\pi} \cdot n^2 \ln n$.

We show in Lemma 19.4.5, that with probability $1 - (1/n)^{\Theta(n)}$ over Δ, simultaneously for all X, if X has a column X_j with $\|X_j\|_1 \geq n^c$ for a constant $c > 0$, then $\|A_S X_j - A_j\|_1 \geq .9n^3$, which is already too large to provide an $O(1)$-approximation. Note that we need such a high probability bound to later union bound over *all* S. Lemma 19.4.5 is in turn shown via a net argument on all X_j (it suffices to prove this for a single $j \in [n]$, since there are only n different j, so we can union bound over all j). The net bounds are given in Definition 19.4.1 and Definition 19.4.4, and the high probability bound for a given coefficient vector X_j is shown in Lemma 19.4.3, where we use properties of the Cauchy distribution. Thus, we can assume $\|X_j\|_1 < n^c$ for all $j \in [n]$. We also show in Fact 19.4.1, conditioned on the fact that $\|\Delta\|_1 \leq \frac{4.002}{\pi} n^2 \ln n$, it holds that for *any* vector X_j, if $\|X_j\|_1 < n^c$ and $|1 - \mathbf{1}^\top X_j| > 1 - 10^{-20}$, then $\|A_S X - A\|_1 \geq \|A_S X_j - A_j\|_1 > n^3$. The intuition here is $A = n^{c_0} \mathbf{1} \cdot \mathbf{1}^\top + \Delta$ for a large constant c_0, and X_j does not have enough norm ($\|X_j\|_1 \leq n^c$) or correlation with the vector $\mathbf{1}$ ($|1 - \mathbf{1}^\top X_j| > 1 - 10^{-20}$) to make $\|A_S X_j - A_j\|_1$ small.

Given the above, we can assume both that $\|X_j\|_1 \leq n^c$ and $|1 - \mathbf{1}^\top X_j| \leq 1 - 10^{-20}$ for all columns j of our coefficient matrix X. We can also assume that $\|A_S X - A\|_1 \leq 4n^2 \ln n$, as otherwise such an X already satisfies $\|A_S X - A\|_1 \geq \frac{4.01}{\pi} \cdot n^2 \ln n$ and we are done. To analyze $\|A_S X - A\|_1 = \sum_{i,j} |(A_S X - A_{[n] \setminus S})_{i,j}|$ in Theorem 19.4.17, we then split the sum over

"large coordinates" (i,j) for which $|\Delta_{i,j}| > n^{1.0002}$, and "small coordinates" (i,j) for which $|\Delta_{i,j}| < n^{.9999}$, and since we seek to lower bound $\|A_S X - A_{[n]\setminus S}\|_1$, we drop the remaining coordinates (i,j). To handle large coordinates, we observe that since the column span of A_S is only $r = n^{o(1)}$-dimensional, as one ranges over all vectors y in its span of 1-norm, say, $O(n^2 \ln n)$, there is only a small subset T, of size at most $n^{.99999}$ of coordinates $i \in [n]$ for which we could ever have $|y_i| \geq n^{1.0001}$. We show this in Lemma 19.4.7. This uses the property of vectors in low-dimensional subspaces, and has been exploited in earlier works in the context of designing so-called subspace embeddings [CW13, MM13]. We call T the "bad region" for A_S. While the column span of A_S depends on Δ_S, it is independent of $\Delta_{[n]\setminus S}$, and thus it is extremely unlikely that the large coordinate of Δ_S "match up" with the bad region of A_S. This is captured in Lemma 19.4.10, where we show that if $\|A_S X - A_{[n]\setminus S}\|_1 \leq 4n^2 \ln n$ (as we said we could assume above), then $\sum_{\text{large coordinates } i,j} |(A_S X - A_{[n]\setminus S})_{i,j}|$ is at least $\frac{1.996}{\pi} n^2 \ln n$. Intuitively, the heavy coordinates make up about $\frac{2}{\pi} n^2 \ln n$ of the total mass of $\|\Delta\|_1$, by tail bounds of the Cauchy distribution, and for any set S of size $n^{o(1)}$, A_S fits at most a small portion of this, still leaving us left with $\frac{1.996}{\pi} n^2 \ln n$ in cost. Our goal is to show that $\|A_S X - A_{[n]\setminus S}\|_1 \geq \frac{4.01}{\pi} \cdot n^2 \ln n$, so we still have a way to go.

We next analyze $\sum_{\text{small coordinates } i,j} |(A_S X - A_{[n]\setminus S})_{i,j}|$. Via Bernstein's inequality, in Lemma 19.4.11 we argue that for any fixed vector y and random vector Δ_j of i.i.d. Cauchy entries, roughly half of the contribution of coordinates to $\|\Delta_j\|_1$ will come from coordinates j for which $\text{sign}(y_j) = \text{sign}(\Delta_j)$ and $|\Delta_j| \leq n^{.9999}$, giving us a contribution of roughly $\frac{.9998}{\pi} n \ln n$ to the cost. The situation we will actually be in, when analyzing a column of $A_S X - A_{[n]\setminus S}$, is that of taking the sum of two independent Cauchy vectors, shifted by a multiple of $\mathbf{1}^\top$. We analyze this setting in Lemma 19.4.13, after first conditioning on certain level sets having

typical behavior in Lemma 19.4.12. This roughly doubles the contribution, gives us roughly a contribution of $\frac{1.996}{\pi} n^2 \ln n$ from coordinates j for which (i, j) is a small coordinate and we look at coordinates i on which the sum of two independent Cauchy vectors have the same sign. Combined with the contribution from the heavy coordinates, this gives us a cost of roughly $\frac{3.992}{\pi} n^2 \ln n$, which still falls short of the $\frac{4.01}{\pi} \cdot n^2 \ln n$ total cost we are aiming for. Finally, if we sum up two independent Cauchy vectors and look at the contribution to the sum from coordinates which disagree in sign, due to the anti-concentration of the Cauchy distribution we can still "gain a little bit of cost" since the values, although differing in sign, are still likely not to be very close in magnitude. We formalize this in Lemma 19.4.14. We combine all of the costs from small coordinates in Lemma 19.4.15, where we show we obtain a contribution of at least $\frac{2.025}{\pi} n \ln n$. This is enough, when combined with our earlier $\frac{1.996}{\pi} n^2 \ln n$ contribution from the heavy coordinates, to obtain an overall $\frac{4.01}{\pi} \cdot n^2 \ln n$ lower bound on the cost, and conclude the proof of our main theorem in Theorem 19.4.17.

In the remaining sections, we will present detailed proofs.

19.4.1 A Useful Fact

Fact 19.4.1. *Let $c_0 > 0$ be a sufficiently large constant. Let $u = n^{c_0} \cdot \mathbf{1} \in \mathbb{R}^n$ and $\Delta \in \mathbb{R}^{n \times (d+1)}$. If $\sum_{i=1}^{d+1} \|\Delta_i\|_1 \leq n^3$ and if $\alpha \in \mathbb{R}^d$ satisfies $|1 - \mathbf{1}^\top \alpha| > 1/n^{c_1}$ and $\|\alpha\|_1 \leq n^c$, where $0 < c < c_0 - 10$ is a constant and $c_1 > 3$ is another constant depending on c_0, c, then*

$$\|u - u\mathbf{1}^\top \alpha + \Delta_{d+1} - \Delta_{[d]}\alpha\|_1 > n^3.$$

Proof.

$$\begin{aligned}
&\|u - u\mathbf{1}^\top\alpha + \Delta_{d+1} - \Delta_{[d]}\alpha\|_1 \\
&\geq |1 - \mathbf{1}^\top\alpha| \cdot \|u\|_1 - \|\Delta_{d+1}\|_1 - \|\Delta_{[d]}\alpha\|_1 \\
&\geq |1 - \mathbf{1}^\top\alpha| \cdot n \cdot n^{c_0} - n^3 - n^4\|\alpha\|_1 \\
&\geq |1 - \mathbf{1}^\top\alpha| \cdot n \cdot n^{c_0} - n^{5+c} \\
&\geq n^{c_0+1-c_1} - n^{5+c} \\
&\geq n^3.
\end{aligned}$$

The first inequality follows by the triangle inequality. The second inequality follows since $u = n^{c_0} \cdot \mathbf{1} \in \mathbb{R}^n$ and $\sum_{i=1}^{d+1} \|\Delta_i\|_1 \leq n^3$. The third inequality follows since $\|\alpha\|_1 \leq n^c$. The fourth inequality follows since $|1 - \mathbf{1}^\top\alpha| > 1/n^{c_1}$. The last inequality follows since $c_0 - c_1 > c + 5$. □

19.4.2 One-Sided Error Concentration Bound for a Random Cauchy Matrix

Lemma 19.4.2 (Lower bound on the cost)**.** *If n is sufficiently large, then*

$$\Pr_{\Delta \sim \{C(0,1)\}^{n \times n}} \left[\|\Delta\|_1 \leq \frac{4.0002}{\pi} n^2 \ln n \right] \geq 1 - O(1/\log\log n).$$

Proof. Let $\Delta \in \mathbb{R}^{n \times n}$ be a random matrix such that each entry is an i.i.d. $C(0,1)$ random Cauchy variable. Let $B = n^2 \ln\ln n$. Let $Z \in \mathbb{R}^{n \times n}$ and $\forall i, j \in [n]$,

$$Z_{i,j} = \begin{cases} |\Delta_{i,j}| & |\Delta_{i,j}| < B \\ B & \text{Otherwise} \end{cases}.$$

For fixed $i, j \in [n]$, we have

$$\begin{aligned}
\mathbb{E}[Z_{i,j}] &= \frac{2}{\pi}\int_0^B \frac{x}{1+x^2}\mathrm{d}x + \Pr[|\Delta_{i,j}| \geq B] \cdot B \\
&= \frac{1}{\pi}\ln(B^2+1) + \Pr[|\Delta_{i,j}| \geq B] \cdot B \\
&\leq \frac{1}{\pi}\ln(B^2+1) + 1
\end{aligned}$$

where the first inequality follows by the cumulative distribution function of a half Cauchy random variable. We also have $\mathbb{E}[Z_{i,j}] \geq \frac{1}{\pi}\ln(B^2+1)$. For the second moment, we have

$$\begin{aligned}
\mathbb{E}[Z_{i,j}^2] &= \frac{2}{\pi}\int_0^B \frac{x^2}{1+x^2}\mathrm{d}x + \Pr[|\Delta_{i,j}| \geq B] \cdot B^2 \\
&= \frac{2}{\pi}(B - \tan^{-1} B) + \Pr[|\Delta_{i,j}| \geq B] \cdot B^2 \\
&\leq \frac{2}{\pi}B + B \\
&\leq 2B
\end{aligned}$$

where the first inequality follows by the cumulative distribution function of a half Cauchy random variable. By applying Bernstein's inequality, we have

$$\begin{aligned}
&\Pr\left[\|Z\|_1 - \mathbb{E}[\|Z\|_1] > 0.0001\,\mathbb{E}[\|Z\|_1]\right] \\
\leq\ &\exp\left(-\frac{0.5 \cdot 0.0001^2\,\mathbb{E}[\|Z\|_1]^2}{n^2 \cdot 2B + \frac{1}{3}B \cdot 0.0001\,\mathbb{E}[\|Z\|_1]}\right) \\
\leq\ &\exp(-\Omega(\ln n / \ln\ln n)) \\
\leq\ &O(1/\ln n). \qquad (19.5)
\end{aligned}$$

The first inequality follows by the definition of Z and the second moment of $Z_{i,j}$. The second

inequality follows from $\mathbb{E}[\|Z\|_1] = \Theta(n^2 \ln n)$ and $B = \Theta(n^2 \ln \ln n)$. Notice that

$$\begin{aligned}
& \Pr\left[\|\Delta\|_1 > \frac{4.0002}{\pi} n^2 \ln n\right] \\
= \ & \Pr\left[\|\Delta\|_1 > \frac{4.0002}{\pi} n^2 \ln n \mid \forall i,j, |\Delta_{i,j}| < B\right] \Pr\left[\forall i,j, |\Delta_{i,j}| < B\right] \\
& + \Pr\left[\|\Delta\|_1 > \frac{4.0002}{\pi} n^2 \ln n \mid \exists i,j, |\Delta_{i,j}| \geq B\right] \Pr\left[\exists i,j, |\Delta_{i,j}| \geq B\right] \\
\leq \ & \Pr\left[\|\Delta\|_1 > \frac{4.0002}{\pi} n^2 \ln n \mid \forall i,j, |\Delta_{i,j}| < B\right] + \Pr\left[\exists i,j, |\Delta_{i,j}| \geq B\right] \\
\leq \ & \Pr\left[\|Z\|_1 > \frac{4.0002}{\pi} n^2 \ln n\right] + \Pr\left[\exists i,j, |\Delta_{i,j}| \geq B\right] \\
\leq \ & \Pr\left[\|Z\|_1 > \frac{4.0002}{\pi} n^2 \ln n\right] + n^2 \cdot 1/B \\
\leq \ & \Pr\left[\|Z\|_1 > 1.0001\, \mathbb{E}[\|Z\|_1]\right] + n^2 \cdot 1/B \\
\leq \ & O(1/\log(n)) + O(1/\log\log n) \\
\leq \ & O(1/\log\log n)
\end{aligned}$$

The second inequality follows by the definition of Z. The third inequality follows by the union bound and the cumulative distribution function of a half Cauchy random variable. The fourth inequality follows from $\mathbb{E}[\|Z\|_1] \leq n^2(1/\pi \cdot \ln(B^2+1)+1) \leq 4.0000001/\pi \cdot n^2 \ln n$ when n is sufficiently large. □

19.4.3 "For Each" Guarantee

In the following Lemma, we show that, for each fixed coefficient vector α, if the entry of α is too large, the fitting cost cannot be small.

Lemma 19.4.3 (For each fixed α, the entry cannot be too large)**.** *Let $c > 0$ be a sufficiently*

large constant, $n \geq d \geq 1$, $u \in \mathbb{R}^n$ *be any fixed vector and* $\Delta \in \mathbb{R}^{n \times d}$ *be a random matrix where* $\forall i \in [n], j \in [d], \Delta_{i,j} \sim C(0,1)$ *independently. For any fixed* $\alpha \in \mathbb{R}^d$ *with* $\|\alpha\|_1 = n^c$,

$$\Pr_{\Delta \sim \{C(0,1)\}^{n \times d}}[\|(u \cdot \mathbf{1}^\top + \Delta)\alpha\|_1 > n^3] > 1 - (1/n)^{\Theta(n)}.$$

Proof. Let c be a sufficiently large constant. Let $\alpha \in \mathbb{R}^d$ with $\|\alpha\|_1 = n^c$. Let $u \in \mathbb{R}^n$ be any fixed vector. Let $\Delta \in \mathbb{R}^{n \times d}$ be a random matrix where $\forall i \in [n], j \in [d], \Delta_{i,j} \sim C(0,1)$. Then $\Delta\alpha \in \mathbb{R}^n$ is a random vector with each entry drawn independently from $C(0, \|\alpha\|_1)$. Due to the probability density function of standard Cauchy random variables,

$$\Pr[\|\Delta\alpha\|_1 < n^3] \geq \Pr[\|\Delta\alpha + u \cdot \mathbf{1}^\top \alpha\|_1 < n^3].$$

It suffices to upper bound $\Pr[\|\Delta\alpha\|_1 < n^3]$. If $c > 10$, then due to the cumulative distribution function of Cauchy random variables, for a fixed $i \in [n]$, $\Pr[|(\Delta\alpha)_i| < n^3] < 1/n$. Thus, $\Pr[\|\Delta\alpha\|_1 < n^3] < (\frac{1}{n})^n$. Thus,

$$\Pr_{\Delta \sim \{C(0,1)\}^{n \times d}}[\|(u \cdot \mathbf{1}^\top + \Delta)\alpha\|_1 > n^3] > 1 - (1/n)^n.$$

□

19.4.4 From "For Each" to "For All" via an ϵ-Net

Definition 19.4.1 (ϵ-net for the ℓ_1-norm ball)**.** Let $A \in \mathbb{R}^{n \times d}$ have rank d, and let $L = \{y \in \mathbb{R}^n \mid y = Ax, x \in \mathbb{R}^d\}$ be the column space of A. An ϵ-net of the ℓ_1-unit sphere $\mathcal{S}^{d-1} = \{y \mid \|y\|_1 = 1, y \in L\} \subset L$ is a set $N \subset \mathcal{S}^{d-1}$ of points for which $\forall y \in \mathcal{S}^{d-1}, \exists y' \in N$ for which $\|y - y'\| \leq \epsilon$.

[DDH+09] proved an upper bound on the size of an ϵ-net.

Lemma 19.4.4 (See, e.g., the ball B on page 2068 of [DDH+09])**.** *Let $A \in \mathbb{R}^{n \times d}$ have rank d, and let $L = \{y \in \mathbb{R}^n \mid y = Ax, x \in \mathbb{R}^d\}$ be the column space of A. For $\epsilon \in (0,1)$, an ϵ-net (Definition 19.4.1) N of the ℓ_1-unit sphere $\mathcal{S}^{d-1} = \{y \mid \|y\|_1 = 1, y \in L\} \subset L$ exists. Furthermore, the size of N is at most $(3/\epsilon)^d$.*

Lemma 19.4.5 (For all possible α, the entry cannot be too large)**.** *Let $n \geq 1, d = n^{o(1)}$. Let $u = n^{c_0} \cdot \mathbf{1} \in \mathbb{R}^n$ denote a fixed vector where c_0 is a constant. Let $\Delta \in \mathbb{R}^{n \times d}$ be a random matrix where $\forall i \in [n], j \in [d], \Delta_{i,j} \sim C(0,1)$ independently. Let $c > 0$ be a sufficiently large constant. Conditioned on $\|\Delta\|_1 \leq n^3$, with probability at least $1 - (1/n)^{\Theta(n)}$, for all $\alpha \in \mathbb{R}^d$ with $\|\alpha\|_1 \geq n^c$, we have $\|(u \cdot \mathbf{1}^\top + \Delta)\alpha\|_1 > 0.9n^3$.*

Proof. Due to Lemma 19.4.4, there is a set $N \subset \{\alpha \in \mathbb{R}^d \mid \|\alpha\|_1 = n^c\} \subset \mathbb{R}^d$ with $|N| \leq 2^{\Theta(d \log n)}$ such that $\forall \alpha \in \mathbb{R}^d$ with $\|\alpha\|_1 = n^c$, $\exists \alpha' \in N$ such that $\|\alpha - \alpha'\|_1 \leq 1/n^{c'}$ where $c' > c_0 + 100$ is a constant. By applying Lemma 19.4.3 and union bounding over all the points in N, with probability at least $1 - (1/n)^n \cdot |N| \geq 1 - (1/n)^n \cdot 2^{n^{o(1)}} = 1 - (1/n)^{\Theta(n)}$, $\forall \alpha' \in N$, $\|(u \cdot \mathbf{1}^\top + \Delta)\alpha'\|_1 > n^3$. $\forall \alpha \in \mathbb{R}^d$ with $\|\alpha\|_1 = n^c$, we can find $\alpha' \in N$ such that

$\|\alpha - \alpha'\|_1 \leq 1/n^{c'}$. Let $\gamma = \alpha - \alpha'$. Then,

$$\begin{aligned}
&\|(u \cdot \mathbf{1}^\top + \Delta)\alpha\|_1 \\
= &\|(u \cdot \mathbf{1}^\top + \Delta)(\alpha' + \gamma)\|_1 \\
\geq &\|(u \cdot \mathbf{1}^\top + \Delta)\alpha'\|_1 - \|(u \cdot \mathbf{1}^\top + \Delta)\gamma\|_1 \\
\geq &\, n^3 - \sqrt{n}\|(u \cdot \mathbf{1}^\top + \Delta)\gamma\|_2 \\
\geq &\, n^3 - \sqrt{n}(\|u \cdot \mathbf{1}^\top\|_2 + \|\Delta\|_2)\|\gamma\|_2 \\
\geq &\, n^3 - n^{c_0+50}/n^{c'} \\
\geq &\, 0.9n^3.
\end{aligned}$$

The first equality follows from $\alpha = \alpha' + \gamma$. The first inequality follows by the triangle inequality. The second inequality follows by the relaxation from the ℓ_1 norm to the ℓ_2 norm. The third inequality follows from the operator norm and the triangle inequality. The fourth inequality follows using $\|\Delta\|_2 \leq \|\Delta\|_1 \leq n^3, \|u\|_2 \leq n^{c_0+10}, \|\gamma\|_2 \leq \|\gamma\|_1 \leq (1/n)^{c'}$. The last inequality follows since $c' > c_0 + 100$.

For $\alpha \in \mathbb{R}^n$ with $\|\alpha\|_1 > n^c$, let $\alpha' = \alpha/\|\alpha\|_1 \cdot n^c$. Then

$$\|(u \cdot \mathbf{1}^\top + \Delta)\alpha\|_1 \geq \|(u \cdot \mathbf{1}^\top + \Delta)\alpha'\|_1 \geq 0.9n^3.$$

□

19.4.5 Bounding the Cost from the Large-Entry Part via "Bad" Regions

In this section, we will use the concept of *well-conditioned basis* in our analysis.

Definition 19.4.2 (Well-conditioned basis [DDH+09])**.** Let $A \in \mathbb{R}^{n \times m}$ have rank d. Let $p \in [1, \infty)$, and let $\|\cdot\|_q$ be the dual norm of $\|\cdot\|_p$, i.e., $1/p + 1/q = 1$. If $U \in \mathbb{R}^{n \times d}$ satisfies

1. $\|U\|_p \leq \alpha$,

2. $\forall z \in \mathbb{R}^d, \|z\|_q \leq \beta \|Uz\|_p$,

then U is an (α, β, p) well-conditioned basis for the column space of A.

The following theorem gives an existence result of a well-conditioned basis.

Theorem 19.4.6 (ℓ_1 well-conditioned basis [DDH+09])**.** *Let $A \in \mathbb{R}^{n \times m}$ have rank d. There exists $U \in \mathbb{R}^{n \times d}$ such that U is a $(d, 1, 1)$ well-conditioned basis for the column space of A.*

In the following lemma, we consider vectors from low-dimensional subspaces. For a coordinate, if there is a vector from the subspace for which this entry is large, but the norm of the vector is small, then this kind of coordinate is pretty "rare". More formally,

Lemma 19.4.7. *Given a matrix $U \in \mathbb{R}^{n \times r}$ for a sufficiently large $n \geq 1$, let $r = n^{o(1)}$. Let $S = \{y | y = Ux, x \in \mathbb{R}^r\}$. Let the set T denote $\{i \in [n] \mid \exists y \in S, |y_i| \geq n^{1.0001}$ and $\|y\|_1 < 8n^2 \ln n\}$. Then we have*

$$|T| \leq n^{0.99999}.$$

Proof. Due to Theorem 19.4.6, let $U \in \mathbb{R}^{n \times r}$ be the $(r, 1, 1)$ well-conditioned basis of the column space of U. If $i \in T$, then $\exists x \in \mathbb{R}^r$ such that $|(Ux)_i| \geq n^{1.0001}$ and $\|Ux\|_1 < 8n^2 \ln n$. Thus, we have

$$n^{1.0001} \leq |(Ux)_i| \leq \|U^i\|_1 \|x\|_\infty \leq \|U^i\|_1 \|Ux\|_1 \leq \|U^i\|_1 \cdot 8n^2 \ln n.$$

The first inequality follows using $n^{1.0001} \leq |(Ux)_i|$. The second inequality follows by Hölder's inequality. The third inequality follows by the second property of the well-conditioned basis. The fourth inequality follows using $\|Ux\|_1 < 8n^2 \ln n$. Thus, we have

$$\|U^i\|_1 \geq n^{1.0001}/n^{2+o(1)} \geq 1/n^{0.9999-o(1)}.$$

Notice that $\sum_{j=1}^{n} \|U^j\|_1 = \|U\|_1 \leq r$. Thus,

$$|T| \leq r/(1/n^{0.9999-o(1)}) = n^{0.9999+o(1)} \leq n^{0.99999}.$$

□

Definition 19.4.3 (Bad region)**.** Given a matrix $U \in \mathbb{R}^{n \times r}$, we say $\mathcal{B}(U) = \{i \in [n] \mid \exists y \in \text{colspan}(U) \subset \mathbb{R}^n \text{ s.t. } y_i \geq n^{1.0001} \text{ and } \|y\|_1 \leq 8n^2 \ln n\}$ is a bad region for U.

Next we state a lower and an upper bound on the probability that a Cauchy random variable is in a certain range,

Claim 19.4.8. *Let $X \sim C(0,1)$ be a standard Cauchy random variable. Then for any $x > 1549$,*

$$\frac{2}{\pi} \cdot \frac{\ln(1.001)}{x} \geq \Pr[|X| \in (x, 1.001x]] \geq \frac{1.999}{\pi} \cdot \frac{\ln(1.001)}{x}.$$

Proof. When $x > 1549$, $\frac{2}{\pi} \cdot \frac{\ln(1.001)}{x} \geq \frac{2}{\pi} \cdot (\tan^{-1}(1.001x) - \tan^{-1}(x)) \geq \frac{1.999}{\pi} \cdot \frac{\ln(1.001)}{x}$. □

We build a level set for the "large" noise values, and we show the bad region cannot cover much of the large noise. The reason is that the bad region is small, and for each row, there is always some large noise.

Lemma 19.4.9. *Given a matrix $U \in \mathbb{R}^{n\times r}$ with n sufficiently large, let $r = n^{o(1)}$, and consider a random matrix $\Delta \in \mathbb{R}^{n\times(n-r)}$ with $\Delta_{i,j} \sim C(0,1)$ independently. Let $L_t = \{(i,j) \mid (i,j) \in [n]\times[n-r], |\Delta_{i,j}| \in (1.001^t, 1.001^{t+1}]\}$. With probability at least $1-1/2^{n^{\Theta(1)}}$, for all $t \in (\frac{1.0002\ln n}{\ln 1.001}, \frac{1.9999\ln n}{\ln 1.001}) \cap \mathbb{N}$,*

$$|L_t \setminus (\mathcal{B}(U)\times[n-r])| \geq n(n-r)\cdot 1.998\cdot\ln(1.001)/(\pi\cdot 1.001^t).$$

Proof. Let $N = n\cdot(n-r)$. Then according to Claim 19.4.8, $\forall t \in (\frac{1.0002\ln n}{\ln 1.001}, \frac{1.9999\ln n}{\ln 1.001}) \cap \mathbb{N}$, $\mathbb{E}(|L_t|) \geq N\cdot 1.999\cdot\ln(1.001)/(\pi\cdot 1.001^t) \geq n^{\Theta(1)}$. For a fixed t, by a Chernoff bound, with probability at least $1-1/2^{n^{\Theta(1)}}$, $|L_t| \geq N\cdot 1.9989\cdot\ln(1.001)/(\pi\cdot 1.001^t)$. Due to Lemma 19.4.7, $|\mathcal{B}(U)\times[n-r]| \leq n^{0.99999}(n-r) = N/n^{0.00001}$. Due to the Chernoff bound, with probability at least $1-1/2^{n^{\Theta(1)}}$, $|L_t \cap (\mathcal{B}(U)\times[n-r])| < N/n^{0.00001}\cdot 2.0001\cdot\ln(1.001)/(\pi\cdot 1.001^t)$. Thus, with probability at least $1-1/2^{n^{\Theta(1)}}$, $|L_t\setminus(\mathcal{B}(U)\times[n-r])| \geq N\cdot 1.998\cdot\ln(1.001)/(\pi\cdot 1.001^t)$. By taking a union bound over all $t \in (\frac{1.0002\ln n}{\ln 1.001}, \frac{1.9999\ln n}{\ln 1.001}) \cap \mathbb{N}$, we complete the proof. □

Lemma 19.4.10 (The cost of the large noise part)**.** *Let $n \geq 1$ be sufficiently large, and let $r = n^{o(1)}$. Given a matrix $U \in \mathbb{R}^{n\times r}$, and a random matrix $\Delta \in \mathbb{R}^{n\times(n-r)}$ with $\Delta_{i,j} \sim C(0,1)$ independently, let $\mathfrak{I} = \{(i,j) \in [n]\times[n-r] \mid |\Delta_{i,j}| \geq n^{1.0002}\}$. If $\|\Delta\|_1 \leq 4n^2\ln n$, then with probability at least $1-1/2^{n^{\Theta(1)}}$, for all $X \in \mathbb{R}^{r\times n}$, either*

$$\sum_{(i,j)\in\mathfrak{I}} |(UX-\Delta)_{i,j}| > \frac{1.996}{\pi}n^2\ln n,$$

or

$$\|UX-\Delta\|_1 > 4n^2\ln n$$

Proof.

$$\begin{aligned}
&\sum_{(i,j)\in\mathcal{I}} |(UX-\Delta)_{i,j}| \\
\geq &\sum_{(i,j)\in\mathcal{I}\setminus\mathcal{B}(U)} |(UX-\Delta)_{i,j}| \\
\geq &\sum_{(i,j)\in\mathcal{I}\setminus\mathcal{B}(U)} |(\Delta)_{i,j}| - \sum_{(i,j)\in\mathcal{I}\setminus\mathcal{B}(U)} |(UX)_{i,j}| \qquad (19.6)
\end{aligned}$$

Let $N = n(n-r)$. By a Chernoff bound and the cumulative distribution function of a Cauchy random variable, with probability at least $1 - 1/2^{n^{\Theta(1)}}$, $|\mathcal{I}| \leq 1.1 \cdot N/n^{1.0002}$. If $\exists(i,j) \in \mathcal{I} \setminus \mathcal{B}(U)$ which has $|(UX)_{i,j}| > n^{1.0001}$, then according to the definition of $\mathcal{B}(U)$, $\|UX\|_1 \geq \|(UX)_j\|_1 \geq 8n^2 \ln n$. Due to the triangle inequality, $\|UX - \Delta\|_1 \geq \|UX\|_1 - \|\Delta\|_1 \geq 4n^2 \ln n$. If $\forall(i,j) \in \mathcal{I} \setminus \mathcal{B}(U)$ we have $|(UX)_{i,j}| \leq n^{1.0001}$, then

$$\sum_{(i,j)\in\mathcal{I}\setminus\mathcal{B}(U)} |(UX)_{i,j}| \leq |\mathcal{I}| \cdot n^{1.0001} \leq 1.1 \cdot N/n^{0.0001}. \qquad (19.7)$$

Due to Lemma 19.4.9, with probability at least $1 - 1/2^{n^{\Theta(1)}}$,

$$\begin{aligned}
&\sum_{(i,j)\in\mathcal{I}\setminus\mathcal{B}(U)} |(\Delta)_{i,j}| \\
\geq &\sum_{t\in(\frac{1.0002\ln n}{\ln 1.001}, \frac{1.9999\ln n}{\ln 1.001})\cap\mathbb{N}} 1.001^t \cdot N \cdot 1.998 \cdot \ln(1.001)/(\pi \cdot 1.001^t) \\
\geq &\frac{1.997}{\pi} \cdot N \ln n. \qquad (19.8)
\end{aligned}$$

We plug (19.7) and (19.8) into (19.6), from which we have

$$\sum_{(i,j)\in\mathcal{I}} |(UX-\Delta)_{i,j}| \geq \frac{1.996}{\pi} n^2 \ln n.$$

□

19.4.6 Cost from the Sign-Agreement Part of the Small-Entry Part

We use $-y$ to fit Δ (we think of $A_S\alpha = A_S^*\alpha - y$, and want to minimize $\|-y-\Delta\|_1$). If the sign of y_j is the same as the sign of Δ_j, then both coordinate values will collectively contribute.

Lemma 19.4.11 (The contribution from Δ_i when Δ_i and y_i have the same sign)**.** *Suppose we are given a vector $y \in \mathbb{R}^n$ and a random vector $\Delta \in \mathbb{R}^n$ with $\Delta_j \sim C(0,1)$ independently. Then with probability at least $1 - 1/2^{n^{\Theta(1)}}$,*

$$\sum_{j \,:\, \mathrm{sign}(y_j)=\mathrm{sign}(\Delta_j) \text{ and } |\Delta_j| \leq n^{0.9999}} |\Delta_j| > \frac{0.9998}{\pi} n \ln n.$$

Proof. For $j \in [n]$, define the random variable

$$Z_j = \begin{cases} \Delta_j & 0 < \Delta_j \leq n^{0.9999} \\ 0 & \text{otherwise} \end{cases}.$$

Then, we have

$$\Pr\left[\sum_{j \,:\, \mathrm{sign}(y_j)=\mathrm{sign}(\Delta_{i,j}) \text{ and } |\Delta_j| \leq n^{0.9999}} |\Delta_j| > \frac{0.9998}{\pi} n \ln n\right] = \Pr\left[\sum_{j=1}^{n} Z_j > \frac{0.9998}{\pi} n \ln n\right].$$

Let $B = n^{0.9999}$. For $j \in [n]$,

$$\mathbb{E}[Z_j] = \frac{1}{\pi}\int_0^B \frac{x}{1+x^2}\mathrm{d}x = \frac{1}{2\pi}\ln(B^2+1).$$

Also,

$$\mathbb{E}[Z_j^2] = \frac{1}{\pi}\int_0^B \frac{x^2}{1+x^2}\mathrm{d}x = \frac{B - \tan^{-1}(B)}{\pi} \leq B.$$

By Bernstein's inequality,

$$\begin{aligned}
&\Pr\left[\mathbb{E}\left[\sum_{j=1}^{n} Z_j\right] - \sum_{j=1}^{n} Z_j > 10^{-5}\,\mathbb{E}\left[\sum_{j=1}^{n} Z_j\right]\right] \\
\leq\ &\exp\left(-\frac{0.5\cdot\left(10^{-5}\,\mathbb{E}\left[\sum_{j=1}^{n} Z_j\right]\right)^2}{\sum_{j=1}^{n}\mathbb{E}[Z_j^2] + \frac{1}{3}B\cdot 10^{-5}\,\mathbb{E}\left[\sum_{j=1}^{n} Z_j\right]}\right) \\
\leq\ &\exp\left(-\frac{5\cdot 10^{-11} n^2 \ln^2(B^2+1)/(4\pi^2)}{nB + \frac{1}{3}B\cdot 10^{-5} n\ln(B^2+1)/(2\pi)}\right) \\
\leq\ &e^{-n^{\Theta(1)}}.
\end{aligned}$$

The last inequality follows since $B = n^{0.9999}$. Thus, we have

$$\Pr\left[\sum_{j=1}^{n} Z_j < 0.9998/\pi\cdot n\ln n\right] \leq \Pr\left[\sum_{j=1}^{n} Z_j < 0.99999 n\ln(B^2+1)/(2\pi)\right] \leq e^{-n^{\Theta(1)}}.$$

□

Lemma 19.4.12 (Bound on level sets of a Cauchy vector)**.** *Suppose we are given a random vector $y\in\mathbb{R}^n$ with $y_i\sim C(0,1)$ chosen independently. Let*

$$L_t^- = \{i\in[n] \mid -y_i\in(1.001^t, 1.001^{t+1}]\} \text{ and } L_t^+ = \{i\in[n]\mid y_i\in(1.001^t,1.001^{t+1}]\}.$$

With probability at least $1-1/2^{n^{\Theta(1)}}$, for all $t\in(\frac{\ln 1549}{\ln 1.001}, \frac{0.9999\ln n}{\ln 1.001})\cap\mathbb{N}$,

$$\min(|L_t^-|, |L_t^+|) \geq 0.999n\cdot\frac{1}{\pi}\frac{\ln 1.001}{1.001^t}.$$

Proof. For $i\in[n], t\geq\frac{\ln 1549}{\ln 1.001}$, according to Claim 19.4.8, $\Pr[y_i\in(1.001^t,1.001^{t+1}]] \geq 0.9995/\pi\cdot\ln(1.0001)/1.001^t$. Thus, $\mathbb{E}[|L_t^+|] = \mathbb{E}[|L_t^-|] = n\cdot 0.9995/\pi\cdot\ln(1.0001)/1.001^t$.

Since $t \leq \frac{0.9999 \ln n}{\ln 1.001}$, $1.001^t \leq n^{0.9999}$, we have $\mathbb{E}[|L_t^+|] = \mathbb{E}[|L_t^-|] \geq n^{\Theta(1)}$. By applying a Chernoff bound,

$$\Pr[|L_t^+| > 0.999n/\pi \cdot \ln(1.0001)/1.001^t] \geq 1 - 1/2^{n^{\Theta(1)}}.$$

Similarly, we have

$$\Pr[|L_t^-| > 0.999n/\pi \cdot \ln(1.0001)/1.001^t] \geq 1 - 1/2^{n^{\Theta(1)}}.$$

By taking a union bound over all the L_t^+ and L_t^-, we complete the proof. □

Lemma 19.4.13 (The contribution from y_i when Δ_i and y_i have the same sign)**.** *Let $u = \eta \cdot \mathbf{1} \in \mathbb{R}^n$ where $\eta \in \mathbb{R}$ is an arbitrary real number. Let $y \in \mathbb{R}^n$ be a random vector with $y_i \sim C(0, \beta)$ independently for some $\beta > 0$. Let $\Delta \in \mathbb{R}^n$ be a random vector with $\Delta_i \sim C(0, 1)$ independently. With probability at least $1 - 1/2^{n^{\Theta(1)}}$,*

$$\sum_{i \,:\, \text{sign}((u+y)_i)=\text{sign}(\Delta_i) \text{ and } |\Delta_i| \leq n^{0.9999}} |(u+y)_i| \geq \beta \cdot \frac{0.997}{\pi} n \ln n.$$

Proof. For all $t \in (\frac{\ln 1549}{\ln 1.001}, \frac{0.9999 \ln n}{\ln 1.001}) \cap \mathbb{N}$, define

$$L_t^- = \{i \in [n] \mid -y_i \in (\beta \cdot 1.001^t, \beta \cdot 1.001^{t+1}]\},$$
$$L_t^+ = \{i \in [n] \mid +y_i \in (\beta \cdot 1.001^t, \beta \cdot 1.001^{t+1}]\}.$$

Define

$$G = \{i \in [n] \mid \text{sign}((u+y)_i) = \text{sign}(\Delta_i) \text{ and } |\Delta_i| \leq n^{0.9999}\}.$$

Then $\forall i \in [n], \Pr[i \in G] \geq 0.5 - 1/n^{0.9999} \geq 0.4999999999$. Due to Lemma 19.4.12,

$$\min(|L_t^-|, |L_t^+|) \geq 0.999n \cdot \frac{1}{\pi} \frac{\ln 1.001}{1.001^t} \geq n^{\Theta(1)}.$$

By the Chernoff bound and a union bound, with probability at least $1 - 1/2^{n^{\Theta(1)}}$, $\forall t \in (\frac{\ln 1549}{\ln 1.001}, \frac{0.9999 \ln n}{\ln 1.001}) \cap \mathbb{N}$,

$$\min(|L_t^- \cap G|, |L_t^+ \cap G|) \geq 0.499n \cdot \frac{1}{\pi} \frac{\ln 1.001}{1.001^t}. \tag{19.9}$$

Then we have

$$\begin{aligned}
&\sum_{i \in G} |(u+y)_i| \\
&\geq \sum_{t \in (\frac{\ln 1549}{\ln 1.001}, \frac{0.9999 \ln n}{\ln 1.001}) \cap \mathbb{N}} \left(\sum_{i \in L_t^+, i \in G} |y_i + \eta| + \sum_{i \in L_t^-, i \in G} |-y_i - \eta| \right) \\
&\geq \sum_{t \in (\frac{\ln 1549}{\ln 1.001}, \frac{0.9999 \ln n}{\ln 1.001}) \cap \mathbb{N}} 0.499n \cdot \frac{1}{\pi} \frac{\ln 1.001}{1.001^t} \cdot 2 \cdot 1.001^t \cdot \beta \\
&\geq \beta \cdot \frac{0.997}{\pi} n \ln n
\end{aligned}$$

The second inequality follows by Equation (19.9) and the triangle inequality, i.e., $\forall a, b, c \in \mathbb{R}, |a+c| + |b-c| \geq |a+b|$. □

19.4.7 Cost from the Sign-Disagreement Part of the Small-Entry Part

Lemma 19.4.14. *Given a vector $y \in \mathbb{R}^n$ and a random vector $\Delta \in \mathbb{R}^n$ with $\Delta_i \sim C(0,1)$ independently, with probability at least $1 - 1/2^{n^{\Theta(1)}}$,*

$$\sum_{i \ : \ \text{sign}(y_i) \neq \text{sign}(\Delta_i) \ \text{ and } \ |\Delta_i| < n^{0.9999}} |y_i + \Delta_i| > \frac{0.03}{\pi} n \ln n.$$

Proof. For $t \in [0, \frac{0.9999 \ln n}{\ln 4}) \cap \mathbb{N}$ define

$$L_t = \{i \in [n] \mid \text{sign}(y_i) \neq \text{sign}(\Delta_i), |\Delta_i| \in (4^t, 4^{t+1}], |\Delta_i| \notin [|y_i| - 4^t, |y_i| + 4^t]\}.$$

$\forall x \geq 1, y > 0$, we have

$$\begin{aligned}
&\Pr_{X \sim C(0,1)}[|X| \in (x, 4x], |X| \notin [y - x, y + x]] \\
\geq &\Pr_{X \sim C(0,1)}[|X| \in (3x, 4x]] \\
= &\frac{2}{\pi} \cdot (\tan^{-1}(4x) - \tan^{-1}(3x)) \\
\geq &\frac{0.1}{\pi} \cdot \frac{\ln(4)}{x}
\end{aligned}$$

Thus, $\forall i \in [n], t \in [0, \frac{0.9999 \ln n}{\ln 4}) \cap \mathbb{N}$,

$$\Pr[i \in L_t] \geq \frac{0.05}{\pi} \cdot \frac{\ln(4)}{4^t}.$$

Thus, $\forall t \in [0, \frac{0.9999 \ln n}{\ln 4}) \cap \mathbb{N}$,

$$\mathbb{E}[|L_t|] \geq 0.05n/\pi \cdot \ln(4)/4^t \geq n^{\Theta(1)}.$$

By a Chernoff bound and a union bound, with probability at least $1 - 1/2^{n^{\Theta(1)}}$ $\forall t \in [0, \frac{0.9999 \ln n}{\ln 4}) \cap \mathbb{N}$,

$$|L_t| \geq 0.04n/\pi \cdot \ln(4)/4^t.$$

Thus, we have, with probability at least $1 - 1/2^{n^{\Theta(1)}}$,

$$\begin{aligned}
&\sum_{i \,:\, \text{sign}(y_i) \neq \text{sign}(\Delta_i) \text{ and } |\Delta_i| < n^{0.9999}} |y_i + \Delta_i| \\
\geq &\sum_{t \in [0, \frac{0.9999 \ln n}{\ln 4}) \cap \mathbb{N}} |L_t| \cdot 4^t \\
\geq &\frac{0.03}{\pi} n \ln n.
\end{aligned}$$

□

19.4.8 Overall Cost of the Small-Entry Part

Lemma 19.4.15 (For each)**.** *Let $u = \eta \cdot \mathbf{1} \in \mathbb{R}^n$ where $\eta \in \mathbb{R}$ is an arbitrary real number. Let $\alpha \in \mathbb{R}^d$ where $\|\alpha\|_1 \geq 1 - 10^{-20}$. Let $\Delta \in \mathbb{R}^{n \times (d+1)}$ and $\forall (i,j) \in [n] \times [d+1], \Delta_{i,j} \sim C(0,1)$ are i.i.d. standard Cauchy random variables. Then with probability at least $1 - 1/2^{n^{\Theta(1)}}$,*

$$\sum_{j \in [n], |\Delta_{j,d+1}| < n^{0.9999}} |(u + \Delta_{d+1} - (u\mathbf{1}^\top + \Delta_{[d]})\alpha)_j| \geq \frac{2.025}{\pi} n \ln n.$$

Proof. Let G_1 and G_2 be defined as

$$G_1 = \{j \in [n] \mid |\Delta_{j,d+1}| < n^{0.9999}, \mathrm{sign}((u(1 - \mathbf{1}^\top \alpha) - \Delta_{[d]}\alpha)_j) = \mathrm{sign}(\Delta_{d+1})_j)\},$$
$$G_2 = \{j \in [n] \mid |\Delta_{j,d+1}| < n^{0.9999}, \mathrm{sign}((u(1 - \mathbf{1}^\top \alpha) - \Delta_{[d]}\alpha)_j) \neq \mathrm{sign}(\Delta_{d+1})_j)\}.$$

Notice that $\Delta_{[d]}\alpha$ is a random vector with each entry independently drawn from $C(0, \|\alpha\|_1)$. Then with probability at least $1 - 1/2^{n^{\Theta(1)}}$,

$$\begin{aligned}
& \sum_{j \in [n], |\Delta_{j,d+1}| < n^{0.9999}} |(u + \Delta_{d+1} - (u\mathbf{1}^\top + \Delta_{[d]})\alpha)_j| \\
= & \sum_{j \in [n], |\Delta_{j,d+1}| < n^{0.9999}} |(u(1 - \mathbf{1}^\top \alpha) - \Delta_{[d]}\alpha + \Delta_{d+1})_j| \\
= & \sum_{j \in G_1} |(u(1 - \mathbf{1}^\top \alpha) - \Delta_{[d]}\alpha + \Delta_{d+1})_j| + \sum_{j \in G_2} |(u(1 - \mathbf{1}^\top \alpha) - \Delta_{[d]}\alpha + \Delta_{d+1})_j| \\
= & \sum_{j \in G_1} |(u(1 - \mathbf{1}^\top \alpha) - \Delta_{[d]}\alpha)_j| + \sum_{j \in G_1} |(\Delta_{d+1})_j| + \sum_{j \in G_2} |(u(1 - \mathbf{1}^\top \alpha) - \Delta_{[d]}\alpha + \Delta_{d+1})_j| \\
\geq & \|\alpha\|_1 \cdot \frac{0.997}{\pi} \cdot n \ln n + \frac{0.9998}{\pi} n \ln n + \frac{0.03}{\pi} n \ln n \\
\geq & \frac{2.025}{\pi} n \ln n
\end{aligned}$$

The first inequality follows by Lemma 19.4.13, Lemma 19.4.11 and Lemma 19.4.14. The second inequality follows by $\|\alpha\|_1 \geq 1 - 10^{-20}$. □

Lemma 19.4.16 (For all)**.** *Let $c > 0, c_0 > 0$ be two arbitrary constants. Let $u = \eta \cdot \mathbf{1} \in \mathbb{R}^n$ where $\eta \in \mathbb{R}$ satisfies $|\eta| \le n^{c_0}$. Consider a random matrix $\Delta \in \mathbb{R}^{n\times(d+1)}$ with $d = n^{o(1)}$ and $\forall (i,j) \in [n]\times[d+1], \Delta_{i,j} \sim C(0,1)$ are i.i.d. standard Cauchy random variables. Conditioned on $\|\Delta\|_1 \le n^3$, with probability at least $1 - 1/2^{n^{\Theta(1)}}$, $\forall \alpha \in \mathbb{R}^d$ with $1 - 10^{-20} \le \|\alpha\|_1 \le n^c$,*

$$\sum_{j\in[n],|\Delta_{j,d+1}|<n^{0.9999}} |(u + \Delta_{d+1} - (u\mathbf{1}^\top + \Delta_{[d]})\alpha)_j| \ge \frac{2.024}{\pi} n \ln n.$$

Proof. Let $\mathcal{N}$ be a set of points:

$$\mathcal{N} = \left\{ \alpha \in \mathbb{R}^d \mid 1 - 10^{-20} \le \|\alpha\|_1 \le n^c \text{ and } \exists q \in \mathbb{Z}^d, \text{ such that } \alpha = q/n^{c+c_0+1000} \right\}.$$

Since $d = n^{o(1)}$, we have $|\mathcal{N}| \le (n^{2c+c_0+2000})^d = 2^{n^{o(1)}}$. By Lemma 19.4.15 and a union bound, with probability at least $1 - 1/2^{n^{\Theta(1)}} \cdot |\mathcal{N}| \ge 1 - 1/2^{n^{\Theta(1)}}$, $\forall \alpha \in \mathcal{N}$, we have

$$\sum_{j\in[n],|\Delta_{j,d+1}|<n^{0.9999}} |(u + \Delta_{d+1} - (u\mathbf{1}^\top + \Delta_{[d]})\alpha)_j| \ge \frac{2.025}{\pi} n \ln n.$$

Due to the construction of $\mathcal{N}$, we have $\forall \alpha \in \mathbb{R}^d$ with $1 - 10^{-20} \le \|\alpha\|_1 \le n^c$, $\exists \alpha' \in \mathcal{N}$ such that $\|\alpha - \alpha'\|_\infty \le 1/n^{c+c_0+1000}$. Let $\gamma = \alpha - \alpha'$. Then

$$\begin{aligned}
& \sum_{j\in[n],|\Delta_{j,d+1}|<n^{0.9999}} |(u + \Delta_{d+1} - (u\mathbf{1}^\top + \Delta_{[d]})\alpha)_j| \\
= & \sum_{j\in[n],|\Delta_{j,d+1}|<n^{0.9999}} |(u + \Delta_{d+1} - (u\mathbf{1}^\top + \Delta_{[d]})(\alpha' + \gamma))_j| \\
\ge & \sum_{j\in[n],|\Delta_{j,d+1}|<n^{0.9999}} |(u + \Delta_{d+1} - (u\mathbf{1}^\top + \Delta_{[d]})\alpha')_j| - \sum_{j\in[n],|\Delta_{j,d+1}|<n^{0.9999}} |((u\mathbf{1}^\top + \Delta_{[d]})\gamma)_j| \\
\ge & \sum_{j\in[n],|\Delta_{j,d+1}|<n^{0.9999}} |(u + \Delta_{d+1} - (u\mathbf{1}^\top + \Delta_{[d]})\alpha')_j| - \|(u\mathbf{1}^\top + \Delta_{[d]})\gamma\|_1 \\
\ge & \frac{2.025}{\pi} n \ln n - 1/n^{500} \\
\ge & \frac{2.024}{\pi} n \ln n
\end{aligned}$$

The first equality follows from $\alpha = \alpha' + \gamma$. The first inequality follows by the triangle inequality. The third inequality follows from $\|\gamma\|_1 \le 1/n^{c+c_0+800}, \|u\mathbf{1}^\top\|_1 \le n^{c_0+10}, \|\Delta\|_1 \le n^3$, and $\forall \alpha' \in \mathcal{N}$,

$$\sum_{j\in[n],|\Delta_{j,d+1}|<n^{0.9999}} |(u + \Delta_{d+1} - (u\mathbf{1}^\top + \Delta_{[d]})\alpha')_j| \ge \frac{2.025}{\pi} n \ln n.$$

□

19.4.9 Main result

Theorem 19.4.17 (Formal version of Theorem 19.1.2)**.** *Let $n > 0$ be sufficiently large. Let $A = \eta \cdot \mathbf{1} \cdot \mathbf{1}^\top + \Delta \in \mathbb{R}^{n\times n}$ be a random matrix where $\eta = n^{c_0}$ for some sufficiently large constant c_0, and $\forall i, j \in [n], \Delta_{i,j} \sim C(0,1)$ are i.i.d. standard Cauchy random variables. Let $r = n^{o(1)}$. Then with probability at least $1 - O(1/\log\log n)$, $\forall S \subset [n]$ with $|S| = r$,*

$$\min_{X\in\mathbb{R}^{r\times n}} \|A_S X - A\|_1 \ge 1.002\|\Delta\|_1$$

Proof. We first argue that for a fixed set S, conditioned on $\|\Delta\|_1 \le 100n^2 \ln n$, with probability at least $1 - 1/2^{n^{\Theta(1)}}$,

$$\min_{X\in\mathbb{R}^{r\times n}} \|A_S X - A\|_1 \ge 1.002\|\Delta\|_1.$$

Then we can take a union bound over the at most $n^r = 2^{n^{o(1)}}$ possible choices of S. It suffices to show for a fixed set S, $\min_{X\in\mathbb{R}^{r\times n}} \|A_S X - A\|_1$ is not small.

Without loss of generality, let $S = [r]$, and we want to argue the cost

$$\min_{X\in\mathbb{R}^{r\times n}} \|A_S X - A\|_1 \ge \min_{X\in\mathbb{R}^{r\times n}} \|A_S X_{[n]\setminus S} - A_{[n]\setminus S}\|_1 \ge 1.002\|\Delta\|_1.$$

Due to Lemma 19.4.2, with probability at least $1 - O(1/\log\log n)$,

$$\|\Delta\|_1 \leq 4.0002/\pi \cdot n^2 \ln n.$$

Now, we can condition on $\|\Delta\|_1 \leq 4.0002/\pi \cdot n^2 \ln n$.

Consider $j \in [n] \setminus S$. Due to Lemma 19.4.5, with probability at least $1 - (1/n)^{\Theta(n)}$, for all $X_j \in \mathbb{R}^r$ with $\|X_j\|_1 \geq n^c$ for some constant $c > 0$, we have

$$\|A_S X_j - A_j\|_1 = \|(\eta \cdot \mathbf{1} \cdot \mathbf{1}^\top + [\Delta_S \ \Delta_j])[X_j^\top \ -1]^\top\|_1 \geq 0.9n^3.$$

By taking a union bound over all $j \in [n] \setminus S$, with probability at least $1 - (1/n)^{\Theta(n)}$, for all $X \in \mathbb{R}^{r \times n}$ with $\exists j \in [n] \setminus S, \|X_j\|_1 \geq n^c$, we have

$$\|A_S X - A\|_1 \geq 0.9n^3.$$

Thus, we only need to consider the case $\forall j \in [n] \setminus S, \|X_j\|_1 \leq n^c$. Notice that we condition on $\|\Delta\|_1 \leq 4.0002/\pi \cdot n^2 \ln n$. By Fact 19.4.1, we have that if $\|X_j\|_1 \leq n^c$ and $|1 - \mathbf{1}^\top X_j| > 1 - 10^{-20}$, then

$$\|A_S X - A\|_1 \geq \|A_S X_j - A_j\|_1 > n^3.$$

Thus, we only need to consider the case $\forall j \in [n] \setminus S, \|X_j\|_1 \leq n^c, |1 - \mathbf{1}^\top X_j| \leq 1 - 10^{-20}$. $\forall X \in \mathbb{R}^{r \times n}$ with $\forall j \in [n] \setminus S, \|X_j\|_1 \leq n^c, |1 - \mathbf{1}^\top X_j| \leq 1 - 10^{-20}$, if $\|A_S X_{[n] \setminus S} - A_{[n] \setminus S}\|_1 \leq$

$4n^2 \ln n$, then

$$
\begin{aligned}
&\|A_S X_{[n]\setminus S} - A_{[n]\setminus S}\|_1 \\
=\ &\|(\eta \cdot \mathbf{1}\cdot\mathbf{1}^\top + \Delta_S)X_{[n]\setminus S} - (\eta\cdot\mathbf{1}\cdot\mathbf{1}^\top + \Delta_{[n]\setminus S})\|_1 \\
\geq\ & \sum_{i\in[n], j\in[n]\setminus S, |\Delta_{i,j}|\geq n^{1.0002}} |(((\eta\cdot\mathbf{1}\cdot\mathbf{1}^\top + \Delta_S)X_{[n]\setminus S} - \eta\cdot\mathbf{1}\cdot\mathbf{1}^\top) - \Delta_{[n]\setminus S})_{i,j}| \\
&+ \sum_{i\in[n], j\in[n]\setminus S, |\Delta_{i,j}|<n^{0.9999}} |((\eta\cdot\mathbf{1}\cdot\mathbf{1}^\top + \Delta_S)X_{[n]\setminus S} - (\eta\cdot\mathbf{1}\cdot\mathbf{1}^\top + \Delta_{[n]\setminus S}))_{i,j}| \\
\geq\ & \frac{1.996}{\pi}\cdot n^2\ln n + \sum_{i\in[n], j\in[n]\setminus S, |\Delta_{i,j}|<n^{0.9999}} |((\eta\cdot\mathbf{1}\cdot\mathbf{1}^\top + \Delta_S)X_{[n]\setminus S} - (\eta\cdot\mathbf{1}\cdot\mathbf{1}^\top + \Delta_{[n]\setminus S}))_{i,j}| \\
=\ & \frac{1.996}{\pi}\cdot n^2\ln n + \sum_{j\in[n]\setminus S}\sum_{i\in[n], |\Delta_{i,j}|<n^{0.9999}} |((\eta\cdot\mathbf{1}\cdot\mathbf{1}^\top + \Delta_S)X_j - \eta\cdot\mathbf{1} - \Delta_j)_i| \\
\geq\ & \frac{1.996}{\pi}\cdot n^2\ln n + \sum_{j\in[n]\setminus S}\frac{2.024}{\pi} n\ln n \\
\geq\ & \frac{1.996}{\pi}\cdot n^2\ln n + \frac{2.023}{\pi} n^2\ln n \\
\geq\ & \frac{4.01}{\pi}\cdot n^2\ln n
\end{aligned}
$$

holds with probability at least $1 - 1/2^{n^{\Theta(1)}}$. The first equality follows by the definition of A. The first inequality follows by the partition by $|\Delta_{i,j}|$. Notice that $[\mathbf{1}\ \Delta_S]$ has rank at most $r+1 = n^{o(1)}$. Then, due to Lemma 19.4.10, and the condition $\|A_S X_{[n]\setminus S} - A_{[n]\setminus S}\|_1 \leq 4n^2\ln n$, the second inequality holds with probability at least $1 - 1/2^{n^{\Theta(1)}}$. The second equality follows by grouping the cost by each column. The third inequality holds with probability at least $1 - 1/2^{n^{\Theta(1)}}$ by Lemma 19.4.16, and a union bound over all the columns in $[n]\setminus S$. The fourth inequality follows by $n - r = n - n^{o(1)} \geq (1 - 10^{-100})n$.

Thus, conditioned on $\|\Delta\|_1 \leq 4.0002/\pi \cdot n^2\ln n$, with probability at least $1 - 1/2^{n^{\Theta(1)}}$,

we have

$$\min_{X\in\mathbb{R}^{r\times n}} \|A_S X - A\|_1 \geq \frac{4.02}{\pi}\cdot n^2 \ln n.$$

By taking a union bound over all the $\binom{n}{r} = 2^{n^{o(1)}}$ choices of S, we have that conditioned on $\|\Delta\|_1 \leq \frac{4.0002}{\pi} n^2 \ln n$, with probability at least $1 - 1/2^{n^{\Theta(1)}}$, $\forall S \subset [n]$ with $|S| = r = n^{o(1)}$, $\min_{X\in\mathbb{R}^{r\times n}} \|A_S X - A\|_1 \geq \frac{4.02}{\pi}\cdot n^2 \ln n$. Since $4.01/4.0002 > 1.002$,

$$\min_{X\in\mathbb{R}^{r\times n}} \|A_S X - A\|_1 \geq 1.002\|\Delta\|_1.$$

Since $\|\Delta\|_1 \leq \frac{4.0002}{\pi} n^2 \ln n$ happens with probability at least $1 - O(1/\log\log n)$, this completes the proof. □

Chapter 20

Towards a Zero-One Law for Low Rank Approximation

There are a number of approximation algorithms for NP-hard versions of low rank approximation, such as finding a rank-k matrix B minimizing the sum of absolute values of differences to a given n-by-n matrix A, $\min_{\text{rank-}k\ B} \|A - B\|_1$, or more generally finding a rank-k matrix B which minimizes the sum of p-th powers of absolute values of differences, $\min_{\text{rank-}k\ B} \|A - B\|_p^p$. Many of these algorithms are linear time columns subset selection algorithms, returning a subset of $\text{poly}(k \log n)$ columns whose cost is no more than a $\text{poly}(k)$ factor larger than the cost of the best rank-k matrix. The above error measures are special cases of the following general entrywise low rank approximation problem: given an arbitrary function $g : \mathbb{R} \to \mathbb{R}_{\geq 0}$, find a rank-$k$ matrix B which minimizes $\|A-B\|_g = \sum_{i,j} g(A_{i,j}-B_{i,j})$. A natural question is which functions g admit efficient approximation algorithms? Indeed, this is a central question of recent work studying generalized low rank models. In this work we give approximation algorithms for *every* function g which is approximately monotone and satisfies an approximate triangle inequality, and we show both of these conditions are necessary. Further, our algorithm is efficient if the function g admits an efficient approximate regression algorithm. Our approximation algorithms handle functions which are not even scale-invariant, such as the Huber loss function, which we show have very different structural properties than ℓ_p-norms, e.g., one can show the lack of scale-invariance causes any column subset selection algorithm to provably require a $\sqrt{\log n}$ factor larger number of columns

than ℓ_p-norms; nevertheless we design the first efficient column subset selection algorithms for such error measures.

20.1 Introduction

A well-studied problem in machine learning and numerical linear algebra, with applications to recommendation systems, text mining, and computer vision, is that of computing a low-rank approximation of a matrix. Such approximations reveal low-dimensional structure, provide a compact way of storing a matrix, and can quickly be applied to a vector.

A commonly used version of the problem is to compute a near optimal low-rank approximation with respect to the Frobenius norm. That is, given an $n \times n$ input matrix A and an accuracy parameter $\epsilon > 0$, output a rank-k matrix B with large probability so that $\|A - B\|_F^2 \leq (1+\epsilon)\|A - A_k\|_F^2$, where for a matrix C, $\|C\|_F^2 = \sum_{i,j} C_{i,j}^2$ is its squared Frobenius norm, and $A_k = \text{argmin}_{\text{rank-}k\ B}\|A - B\|_F$. A_k can be computed exactly using the singular value decomposition (SVD), but takes $O(n^3)$ time in practice and n^ω time in theory, where $\omega \approx 2.373$ is the exponent of matrix multiplication [Str69, CW87, Wil12, LG14].

Sárlos [Sar06] showed how to achieve the above guarantee with constant probability in $\widetilde{O}(\text{nnz}(A) \cdot k/\epsilon) + n \cdot \text{poly}(k/\epsilon)$ time, where $\text{nnz}(A)$ denotes the number of non-zero entries of A. This was improved in [CW13, MM13, NN13a, BDN15, Coh16a] using sparse random projections in $O(\text{nnz}(A)) + n \cdot \text{poly}(k/\epsilon)$ time. Large sparse datasets in recommendation systems are common, such as the Bookcrossing ($100K \times 300K$ with 10^6 observations) [ZMKL05] and Yelp datasets ($40K \times 10K$ with 10^5 observations) [Yel14], and this is a substantial improvement over the SVD.

Robust Low Rank Approximation. To understand the role of the Frobenius norm in the algorithms above, we recall a standard motivation for this error measure. Suppose one has n data points in a k-dimensional subspace of $\mathbb{R}^d$, where $k \ll d$. We can write these points

as the rows of an $n \times d$ matrix A^* which has rank k. The matrix A^* is often called the *ground truth matrix*. In a number of settings, due to measurement noise or other kinds of noise, we only observe the matrix $A = A^* + \Delta$, where each entry of the *noise matrix* $\Delta \in \mathbb{R}^{n \times n}$ is an i.i.d. random variable from a certain mean-zero noise distribution $\mathcal{D}$. One method for approximately recovering A^* from A is maximum likelihood estimation. Here one tries to find a matrix B maximizing the log-likelihood: $\max_{\text{rank-}k\ B} \sum_{i,j} \log p(A_{i,j} - B_{i,j})$, where $p(\cdot)$ is the probability density function of the underlying noise distribution $\mathcal{D}$. For example, when the noise distribution is Gaussian with mean zero and variance σ^2, denoted by $N(0, \sigma^2)$, then the optimization problem is $\max_{\text{rank-}k\ B} \sum_{i,j} \left(\log(1/\sqrt{2\pi\sigma^2}) - (A_{i,j} - B_{i,j})^2/(2\sigma^2) \right)$, which is equivalent to solving the Frobenius norm loss low rank approximation problem defined above.

The Frobenius norm loss, while having nice statistical properties for Gaussian noise, is well-known to be sensitive to outliers. Applying the same maximum likelihood framework above to other kinds of noise distributions results in minimizing other kinds of loss functions. In general, if the density function of the underlying noise $\mathcal{D}$ is $p(z) = c \cdot e^{-g(z)}$, where c is a normalization constant, then the maximum likelihood estimation problem for this noise distribution becomes the following generalized entry-wise loss low rank approximation problem: $\min_{\text{rank-}k\ B} \sum_{i,j} g(A_{i,j} - B_{i,j}) = \min_{\text{rank-}k\ B} \|A - B\|_g$, which is a central topic of recent work on *generalized low-rank models* [UHZ+16]. For example, when the noise is Laplacian, the entrywise ℓ_1 loss is the maximum likelihood estimation, which is also robust to sparse outliers. A natural setting is when the noise is a mixture of small Gaussian noise and sparse outliers; this noise distribution is referred to as the *Huber density*. In this case the Huber loss function gives the maximum likelihood estimate [UHZ+16], where the Huber

function [Hub64] is defined to be: $g(x) = x^2/(2\tau)$ if $|x| < \tau/2$, and $g(x) = |x| - \tau/2$ if $|x| \geq \tau$. Another nice property of the Huber error measure is that it is differentiable everywhere, unlike the ℓ_1-norm, yet still enjoys the robustness properties as one moves away from the origin, making it less sensitive to outliers than the ℓ_2-norm. There are many other kinds of loss functions, known as M-estimators [Zha97], which are widely used as loss functions in robust statistics [HRRS11].

Although several specific cases have been studied, such as entry-wise ℓ_p loss [CLMW11, SWZ17, CGK+17a, BKW17, BBB+19a], weighted entry-wise ℓ_2 loss [RSW16], and cascaded $\ell_p(\ell_2)$ loss [DVTV09, CW15a], the landscape of general entry-wise loss functions remains elusive. There are no results known for any loss function which is not scale-invariant, much less any kind of characterization of which loss functions admit efficient algorithms. This is despite the importance of these loss functions; we refer the reader to [UHZ+16] for a survey of generalized low rank models. This motivates the main question in our work:

Question 20.1.1 (General Loss Functions)**.** *For a given approximation factor $\alpha > 1$, which functions g allow for efficient low-rank approximation algorithms? Formally, given an $n \times d$ matrix A, can we find a* rank-k *matrix B for which $\|A - B\|_g \leq \alpha \min_{\text{rank}-k\ B'} \|A - B'\|_g$, where for a matrix C, $\|C\|_g = \sum_{i \in [n], j \in [d]} g(C_{i,j})$? What if we also allow B to have rank* $\text{poly}(k \log n)$*?*

For Question 20.1.1, one has $g(x) = |x|^p$ for p-norms, and note the Huber loss function also fits into this framework. Allowing B to have slightly larger rank than k, namely, $\text{poly}(k \log n)$, is often sufficient for applications as it still allows for the space savings and computational gains outlined above. These are referred to as bicriteria approximations and are the focus of our work.

Notation. Before we present our results, let us briefly introduce the notation. For $n \in \mathbb{Z}_{\geq 0}$, let $[n]$ denote the set $\{1, 2, \cdots, n\}$. Let $A \in \mathbb{R}^{n \times m}$. A_i and A^j denote the i^{th} column and the j^{th} row of A respectively. Let $P \subseteq [m], Q \subseteq [n]$. A_P denotes the matrix which is composed by the columns of A with column indices in P. Similarly, A^Q denotes the matrix composed by the rows of A with row indices in Q. Let S be a set and $s \in \mathbb{Z}_{\geq 0}$. We use $\binom{S}{s}$ to denote the set of all the size-s subsets of S.

20.1.1 Our Results

We give necessary and sufficient conditions for low rank approximation with respect to general error measures. Our algorithm is a column subset selection algorithm, returning a small subset of columns which span a good low rank approximation. Column subset selection has the benefit of preserving sparsity and interpretability, as described above. We describe two properties on the function g that we need to obtain our low rank approximation algorithms, and show that if g does not have either property, then there are matrices for which no small subset of columns spanning a good low rank approximation with respect to the error measure g exists. We make these terms precise in the theorem statements below.

Since we obtain column subset selection algorithms for a wide class of functions, our algorithms must necessarily be bicriteria and have approximation factor at least $\text{poly}(k)$. Indeed, a special case of our class of functions includes entrywise ℓ_1-low rank approximation, for which it was shown in Theorem G.27 of [SWZ17] that any subset of $\text{poly}(k)$ columns incurs an approximation error of at least $k^{\Omega(1)}$. We also show that for the entrywise Huber-low rank approximation, already for $k = 1$, $\sqrt{\log n}$ columns are needed to obtain any constant factor approximation, thus showing that for some of the functions we consider, a dependence

on n in our column subset size is necessary.

We note that previously for almost all such functions, it was not known how to obtain any non-trivial approximation factor with any sublinear number of columns.

20.1.1.1 A Zero-One Law

We first state three general properties, the first two of which are structural properties and are necessary and sufficient for obtaining a good approximation from a small subset of columns. The third property is needed for efficient running time.

Approximate triangle inequality. For $t \in \mathbb{Z}_{>0}$, we say a function $g(x) : \mathbb{R} \to \mathbb{R}_{\geq 0}$ satisfies the $\text{ati}_{g,t}$-approximate triangle inequality if for any $x_1, x_2, \cdots, x_t \in \mathbb{R}$, $g\left(\sum x_i\right) \leq \text{ati}_{g,t} \cdot \sum g(x_i)$.

Monotone property. For any parameter $\text{mon}_g \geq 1$, we say function $g(x) : \mathbb{R} \to \mathbb{R}_{\geq 0}$ is mon_g-monotone if for any $x, y \in \mathbb{R}$ with $0 \leq |x| \leq |y|$, we have $g(x) \leq \text{mon}_g \cdot g(y)$.

Many functions including most M-estimators [Zha97] and the quantile function [KBJ78] satisfy the above two properties. See Table 20.1 for several examples. We refer the reader to the later sections for the necessity of these two properties. Our next property is not structural, but rather states that if the loss function has an efficient regression algorithm, then that suffices to efficiently find a small subset of columns spanning a good low rank approximation.

Regression property. We say function $g(x) : \mathbb{R} \to \mathbb{R}_{\geq 0}$ has the $(\text{reg}_{g,d}, \mathcal{T}_{\text{reg},g,n,d,m})$-regression property if the following holds: given two matrices $A \in \mathbb{R}^{n \times d}$ and $B \in \mathbb{R}^{n \times m}$, for each $i \in [m]$, let OPT_i denote $\min_{x \in \mathbb{R}^d} \|Ax - B_i\|_g$. There is an algorithm that runs in

Table 20.1: Example functions satisfying both structural properties.

	$g(x)$	$\text{ati}_{g,t}$	mon_g
HUBER	$\begin{cases} x^2/2 & \lvert x\rvert \le \tau \\ \tau(\lvert x\rvert - \tau/2) & \lvert x\rvert > \tau \end{cases}$	$O(t)$	1
ℓ_p $(p \ge 1)$	$\lvert x\rvert^p/p$	$O(t^{p-1})$	1
$\ell_1 - \ell_2$	$2(\sqrt{1 + x^2/2} - 1)$	$O(t)$	1
GEMAN-MCCLURE	$x^2/(2 + 2x^2)$	$O(t)$	1
"FAIR"	$\tau^2\left(\lvert x\rvert/\tau - \log(1 + \lvert x\rvert/\tau)\right)$	$O(t)$	1
TUKEY	$\begin{cases} \tau^2/6 \cdot (1 - (1 - (x/\tau)^2)^3) & \lvert x\rvert \le \tau \\ \tau^2/6 & \lvert x\rvert > \tau \end{cases}$	$O(t)$	1
CAUCHY	$\tau^2/2 \cdot \log(1 + (x/\tau)^2)$	$O(t)$	1
QUANTILE $(\tau \in (0,1))$	$\begin{cases} \tau x & x \ge 0 \\ (\tau - 1)x & x < 0 \end{cases}$	1	$\max\left(\frac{\tau}{1-\tau}, \frac{1-\tau}{\tau}\right)$

$\mathcal{T}_{\text{reg},g,n,d,m}$ time and outputs a matrix $X' \in \mathbb{R}^{d \times m}$ such that $\|AX'_i - B_i\|_g \le \text{reg}_{g,d} \cdot \text{OPT}_i, \forall i \in [m]$ and outputs a vector of estimated regression costs $v \in \mathbb{R}^d$ such that $\text{OPT}_i \le v_i \le \text{reg}_{g,d} \cdot \text{OPT}_i, \forall i \in [m]$. The success probability is at least $1 - 1/\operatorname{poly}(nm)$.

Some functions for which regression itself is non-trivial are e.g., the ℓ_0-loss function and Tukey function. The ℓ_0-loss function corresponds to the nearest codeword problem over the reals and has slightly better than an $O(d)$-approximation ([BK02, APY09], see also [BKW17]). For the Tukey function, [CWW19] shows that Tukey regression is NP-hard, and it also gives approximation algorithms. For discussion on regression solvers, we refer the reader to Appendix 20.5.

For any function, as long as the above general three properties hold, we can provide an efficient algorithm, as our following main theorem shows.

Theorem 20.1.2. *Given a matrix $A \in \mathbb{R}^{n \times n}$, let $k \ge 1, k' = 2k + 1$. Let $g : \mathbb{R} \to \mathbb{R}_{\ge 0}$*

denote a function satisfying the $\mathrm{ati}_{g,k'}$*-approximate triangle inequality, the* mon_g*-monotone property , and the* $(\mathrm{reg}_{g,k'}, \mathcal{T}_{\mathrm{reg},g,n,k',n})$*-regression property. Let* $\mathrm{OPT} = \min_{\mathrm{rank}-k\ A'} \|A'-A\|_g$*. There is an algorithm that runs in* $\widetilde{O}(n + \mathcal{T}_{\mathrm{reg},g,n,k',n})$ *time and outputs a set* $S \subseteq [n]$ *with* $|S| = O(k \log n)$ *such that with probability at least* 0.99*,*

$$\min_{X \in \mathbb{R}^{|S| \times n}} \|A_S X - A\|_g \leq \mathrm{ati}_{g,k'} \cdot \mathrm{mon}_g \cdot \mathrm{reg}_{g,k'} \cdot O(k \log k) \cdot \mathrm{OPT}\,.$$

Although the input matrix A in the above statement is a square matrix, it is straightforward to extend the result to the rectangular case. By the above theorem, we can obtain a good subset of columns. To further get a low rank matrix B which is a good low rank approximation to A, it is sufficient to take an additional $\mathcal{T}_{\mathrm{reg},g,n,|S|,n}$ time to solve the regression problem.

20.1.1.2 Lower Bound on the Number of Columns

One may wonder if the $\log n$ blowup in rank is necessary in our theorem. We show some dependence on n is necessary by showing that for the important Huber loss function, at least $\sqrt{\log n}$ columns are required in order to obtain a constant factor approximation for $k = 1$:

Theorem 20.1.3. *Let* $H(x)$ *denote the following function:* $H(x) = \begin{cases} x^2, & \text{if } |x| < 1; \\ |x|, & \text{if } |x| \geq 1. \end{cases}$

For any $n \geq 1$*, there is a matrix* $A \in \mathbb{R}^{n \times n}$ *such that, if we select* $o(\sqrt{\log n})$ *columns to fit the entire matrix, there is no* $O(1)$*-approximation, i.e., for any subset* $S \subseteq [n]$ *with* $|S| = o(\sqrt{\log n})$*,*

$$\min_{X \in \mathbb{R}^{|S| \times n}} \|A_S X - A\|_H \geq \omega(1) \cdot \min_{\mathrm{rank}-1\ A'} \|A' - A\|_H.$$

Notice that the above function $H(x)$ is always a constant approximation to the Huber function (see Table 20.1) with $\tau = 1$. Thus, the hardness also holds for the Huber function. For more discussion on our lower bound, we refer the reader to Appendix 20.6.

20.1.2 Overview of our Approach and Related Work

Low Rank Approximation for General Functions. A natural approach to low rank approximation is "column subset selection", which has been extensively studied in numerical linear algebra [DMM06c, DMM06b, DMM08, BMD09, BDM11, FEGK13, BW14, WS15, SWZ17, SWZ19b]. One can take the column subset selection algorithm for ℓ_p-low rank approximation in [CGK+17a] and try to adapt it to general loss functions. Namely, their argument shows that for any matrix $A \in \mathbb{R}^{n\times n}$ there exists a subset S of k columns of A, denoted by $A_S \in \mathbb{R}^{n\times k}$, for which there exists a $k \times n$ matrix V for which $\|A_S V - A\|_p^p \leq (k+1)^p \min_{\text{rank-}k\ B'} \|A - B'\|_p^p$; we refer the reader to Theorem 3 of [CGK+17a]. Given the existence of such a subset S, a natural next idea is to then sample a set T of k columns of A uniformly at random. It is then likely the case that if we look at a random column A_i, (1) with probability $1/(k+1)$, i is not among the subset S of k columns out of the $k+1$ columns $T \cup \{i\}$ defining the optimal rank-k approximation to the submatrix $A_{T\cup\{i\}}$, and (2) with probability at least $1/2$, the best rank-k approximation to $A_{T\cup\{i\}}$ has cost at most

$$(2(k+1)/n) \cdot \min_{\text{rank-}k\ B'} \|A - B'\|_p^p. \tag{20.1}$$

Indeed, (1) follows from $T \cup \{i\}$ being a uniformly random subset of $k+1$ columns, while (2) follows from a Markov bound. The argument in Theorem 7 of [CGK+17a] is then able to "prune" a $1/(k+1)$ fraction of columns (this can be optimized to a constant fraction) in expectation, by "covering" them with the random set T. Recursing on the remaining columns,

this procedure stops after $k \log n$ iterations, giving a column subset of size $O(k^2 \log n)$ (which can be optimized to $O(k \log n)$) and an $O(k)$-approximation.

The proof in [CGK+17a] of the existence of a subset S of k columns of A spanning a $(k+1)$-approximation above is quite general, and one might suspect it generalizes to a large class of error functions. Suppose, for example, that $k = 1$. The idea there is to write $A = A^* + \Delta$, where $A^* = U \cdot V$ is the optimal rank-1 ℓ_p-low rank approximation to A. One then "normalizes" by the error, defining $\widetilde{A}_i^* = A_i^*/\|\Delta_i\|_p$ and letting s be such that $\|\widetilde{A}_s^*\|_p$ is largest. The rank-1 subset S is then just A_s. Note that since $\widetilde{A}^*$ has rank-1 and $\|\widetilde{A}_s^*\|_p$ is largest, one can write $\widetilde{A}_j^*$ for every $j \neq s$ as $\alpha_j \cdot \widetilde{A}_s^*$ for $|\alpha_j| \leq 1$. The fact that $|\alpha_j| \leq 1$ is crucial; indeed, consider what happens when we try to "approximate" A_j by $A_s \cdot \frac{\alpha_j \|\Delta_j\|_p}{\|\Delta_s\|_p}$. Then $\|A_j - A_s \alpha_j \|\Delta_j\|_p/\|\Delta_s\|_p\|_p \leq \|A_j - A_j^*\|_p + \left\|A_j^* - A_s \alpha_j \|\Delta_j\|_p/\|\Delta_s\|_p\right\|_p = \|\Delta_j\|_p + \left\|A_j^* - (A_s^* + \Delta_s)\alpha_j \|\Delta_j\|_p/\|\Delta_s\|_p\right\|_p = \|\Delta_j\|_p + \|\Delta_s \alpha_j \|\Delta_j\|_p/\|\Delta_s\|_p\|_p$, and since the p-norm is monotonically increasing and $\alpha_j \leq 1$, the latter is at most $\|\Delta_j\|_p + \|\Delta_s \frac{\|\Delta_j\|_p}{\|\Delta_s\|_p}\|_p$. So far, all we have used about the p-norm is the monotone increasing property, so one could hope that the argument could be generalized to a much wider class of functions.

However, at this point the proof uses that the p-norm has *scale-invariance*, and so $\|\Delta_s \frac{\|\Delta_j\|_p}{\|\Delta_s\|_p}\|_p = \|\Delta_j\|_p \cdot \|\frac{\Delta_s}{\|\Delta_s\|_p}\|_p = \|\Delta_j\|_p$, and it follows that $\|A_j - A_s \frac{\alpha_j \|\Delta_j\|_p}{\|\Delta_s\|_p}\|_p \leq 2\|\Delta_j\|_p$, giving an overall 2-approximation (recall $k = 1$). But what would happen for a general, not necessarily scale-invariant function g? We need to bound $\|\Delta_s \frac{\|\Delta_j\|_g}{\|\Delta_s\|_g}\|_g$. If we could bound this by $O(\|\Delta_j\|_g)$, we would obtain the same conclusion as before, up to constant factors. Consider, though, the "reverse Huber function": $g(x) = x^2$ if $x \geq 1$ and $g(x) = |x|$ for $x \leq 1$. Suppose that Δ_s and Δ_j were just 1-dimensional vectors, i.e., real numbers, so we need to bound $g(\Delta_s g(\Delta_j)/g(\Delta_s))$ by $O(g(\Delta_j))$. Suppose $\Delta_s = 1$. Then $g(\Delta_s) = 1$ and

$g(\Delta_s g(\Delta_j)/g(\Delta_s)) = g(g(\Delta_j))$ and if $\Delta_j = n$, then $g(g(\Delta_j)) = n^4 = g(\Delta_j)^2$, much larger than the $O(g(\Delta_j))$ we were aiming for.

Maybe the analysis can be slightly changed to correct for these normalization issues? This is not the case, as we show that unlike for ℓ_p-low rank approximation, for the reverse Huber function *there is no subset of* 2 *columns of* A *obtaining better than an* $n^{1/4}$*-approximation factor*. (See Section 20.6.2 for more details). Further, the lack of scale invariance not only breaks the argument in [CGK+17a], it shows that combinatorially such functions g behave very differently than ℓ_p-norms. We show more generally there exist functions, in particular the Huber function, for which one needs to choose $\Omega(\sqrt{\log n})$ columns to obtain a constant factor approximation; we describe this more below. Perhaps more surprisingly, we show a subset of $O(\log n)$ columns suffice to obtain a constant factor approximation to the best rank-1 approximation for any function $g(x)$ which is approximately monotone and has the approximate triangle inequality, the latter implying for any constant $C > 0$ and any $x \in \mathbb{R}_{\geq 0}$, $g(Cx) = O(g(x))$. For $k > 1$, these conditions become: (1) $g(x)$ is monotone non-decreasing in x, (2) $g(x)$ is within a poly(k) factor of $g(-x)$, and (3) for any real number $x \in \mathbb{R}^{\geq 0}$, $g(O(kx)) \leq \text{poly}(k) \cdot g(x)$. We show it is possible to obtain an $O(k^2 \log k)$ approximation with $O(k \log n)$ columns. We give the intuition and main lemma statements for our result in Section 20.2, deferring proofs to the supplementary material.

Even for ℓ_p-low rank approximation, our algorithms slightly improve and correct a minor error in [CGK+17a] which claims in Theorem 7 an $O(k)$-approximation with $O(k \log n)$ columns for ℓ_p-low rank approximation. However, their algorithm actually gives an $O(k \log n)$-approximation with $O(k \log n)$ columns. In [CGK+17a] it was argued that one expects to pay a cost of $O(k/n) \cdot \min_{\text{rank-}k\ B'} \|A - B'\|_p^p$ per column as in (20.1), and since each column is only

counted in one iteration, summing over the columns gives $O(k) \cdot \min_{\text{rank-}k\ B'} \|A - B'\|_p$ total cost. The issue is that the value of n is changing in each iteration, so if in the i-th iteration it is n_i, then we could pay $n_i \cdot O(k/n_i) \cdot \min_{\text{rank-}k\ B'} \|A - B'\|_p = O(k) \cdot \min_{\text{rank-}k\ B'} \|A - B'\|_p$ in each of $O(\log n)$ iterations, giving $O(k \log n)$ approximation ratio. In contrast, our algorithm achieves an $O(k \log k)$ approximation ratio for ℓ_p-low rank approximation as a special case, which gives the first $O(1)$ approximation in nearly linear time for any constant k for ℓ_p norms. Our analysis is finer in that we show not only do we expect to pay a cost of $O(k/n_i) \cdot \min_{\text{rank-}k\ B'} \|A - B'\|_p^p$ per column in iteration i, we pay $O(k/n_i)$ times the cost of the best rank-k approximation to A *after the most costly* n/k columns have been removed; thus we pay $O(k/n_i)$ times a *residual cost* with the top n/k columns removed. This ultimately implies any column's cost can contribute in at most $O(\log k)$ of $O(\log n)$ recursive calls, replacing an $O(\log n)$ factor with an $O(\log k)$ factor in the approximation ratio. This also gives the first poly(k)-approximation for ℓ_0-low rank approximation, studied in [BKW17], improving the $O(k^2 \log(n/k))$-approximation there to $O(k^2 \log k)$ and giving the first constant approximation for constant k.

Algorithm 20.1 Low Rank Approximation Algorithm for General Functions

procedure GENERALFUNCTIONLOWRANKAPPROX($A \in \mathbb{R}^{n\times n}, k \in \mathbb{Z}_{\geq 1}, g : \mathbb{R} \to \mathbb{R}_{\geq 0}$)
  Initialization: $r \leftarrow O(\log n), T_0 \leftarrow [n]$
  for $i = 1 \to r$ **do**
    for $j = 1 \to \log n$ **do**
      Sample $S_i^{(j)}$ from $\binom{T_{i-1}}{2k}$ uniformly at random
      Solve the $\mathrm{reg}_{g,2k}$-approximate regression $\min_{x\in\mathbb{R}^{2k}} \|A_{S_i^{(j)}} x - A_t\|_g$ for each $t \in T_{i-1} \setminus S_i^{(j)}$, and let $v_{i,t}^{(j)}$ be the $\mathrm{reg}_{g,2k}$-estimated regression cost ▷ See Section 20.1.1.1 for regression property
      $R_i^{(j)} \leftarrow \{t \mid v_{i,t}^{(j)}$ is the bottom $|T_{i-1} \setminus S_i^{(j)}|/20$ largest value in $\{v_{i,t'}^{(j)} \mid t' \in T_{i-1} \setminus S_i^{(j)}\}\}$
      $c_i^{(j)} \leftarrow \sum_{t\in R_i^{(j)}} v_{i,t}^{(j)}$
    end for
    $j^* \leftarrow \arg\min_{j\in[\log n]} \left\{c_i^{(j)}\right\}$
    $S_i \leftarrow S_i^{(j^*)}, R_i \leftarrow R_i^{(j^*)}, T_i \leftarrow T_{i-1} \setminus (S_i \cup R_i)$
  end for
  return $S = T_r \cup \bigcup_{i\in[r]} S_i$
end procedure

20.2 Algorithm for General Loss Low Rank Approximation

Our algorithm is presented in Algorithm 20.1. First, let us briefly analyze the running time. Consider fixed $i \in [r], j \in [\log n]$. Sampling $S_i^{(j)}$ takes $O(k)$ time. Solving $\mathrm{reg}_{g,2k}$-approximate regression $\min_x \|A_{S_i^{(j)}} x - A_t\|_g$ for all $t \in T_{i-1} \setminus S_i^{(j)}$ takes $\mathcal{T}_{\mathrm{reg},g,n,2k,|T_{i-1}\setminus S_i^{(j)}|} \leq \mathcal{T}_{\mathrm{reg},g,n,2k+1,n}$ time. Since finding $|T_{i-1} \setminus S_i^{(j)}|/20$ smallest element can be done in $O(n)$ time, $R_i^{(j)}$ can be computed in $O(n)$ time. Thus the inner loop takes $O(n + \mathcal{T}_{\mathrm{reg},g,n,2k+1,n})$ time. Since $r = O(\log n)$, the total running time over all i, j is $O((n + \mathcal{T}_{\mathrm{reg},g,n,2k+1,n}) \log^2 n)$. In the remainder of the section, we will sketch the proof of the correctness. For the missing proofs, we refer the reader to Appendix 20.3.

20.2.1 Properties of Uniform Column Sampling

Let us first introduce some useful notation. Consider a rank-k matrix $M^* \in \mathbb{R}^{n\times m}$. For a set $H \subseteq [m]$, let $R_{M^*}(H) \subseteq H$ be a set such that

$$R_{M^*}(H) = \arg\max_{P:P\subseteq H} \left\{ \det\left((M^*)_P^Q\right) \;\middle|\; |P| = |Q| = \mathrm{rank}(M^*_H), Q \subseteq [n] \right\},$$

where $\det(C)$ denotes the determinant of a square matrix C. By Cramer's rule, if we use a linear combination of the columns of $M^*_{R_{M^*}(H)}$ to express any column of M^*_H, the absolute value of every fitting coefficient will be at most 1.

Consider an arbitrary matrix $M \in \mathbb{R}^{n\times m}$. We can write $M = M^* + N$, where $M^* \in \mathbb{R}^{n\times m}$ is an arbitrary rank-k matrix, and $N \in \mathbb{R}^{n\times m}$ is the residual matrix. The following lemma shows that, if we randomly choose a subset $H \subseteq [m]$ of $2k$ columns, and we randomly look at another column i, then with constant probability, the absolute values of all the coefficients of using a linear combination of the columns of M^*_H to express M^*_i are at most 1, and furthermore, if we use the same coefficients to use columns of M_H to fit M_i, then the fitting cost is proportional to $\|N_H\|_g + \|N_i\|_g$.

Lemma 20.2.1. *Given a matrix $M \in \mathbb{R}^{n\times m}$ and a parameter $k \geq 1$, let $M^* \in \mathbb{R}^{n\times m}$ be an arbitrary rank-k matrix. Let $N = M - M^*$. Let $H \subseteq [m]$ be a uniformly random subset of $[m]$, and let i denote a uniformly random index sampled from $[m]\backslash H$. Then* (I) $\Pr[i \notin R_{M^*}(H \cup \{i\})] \geq 1/2$; (II) *If $i \notin R_{M^*}(H \cup \{i\})$, then there exist $|H|$ coefficients $\alpha_1, \alpha_2, \cdots, \alpha_{|H|}$ for which $M^*_i = \sum_{j=1}^{|H|} \alpha_j (M^*_H)_j, \forall j \in [|H|], |\alpha_j| \leq 1$, and $\min_{x\in\mathbb{R}^{|H|}} \|M_H x - M_i\|_g \leq \mathrm{ati}_{g,|H|+1} \cdot \mathrm{mon}_g \cdot \left(\|N_i\|_g + \sum_{j=1}^{|H|} \|(N_H)_j\|_g\right)$.*

Notice that part (II) of the above lemma does not depend on any randomness of H or i. By applying part (I) of the above lemma, it is enough to prove that if we randomly

choose a subset H of $2k$ columns, there is a constant fraction of columns that each column M_i^* can be expressed by a linear combination of columns in M_H^*, and the absolute values of all the fitting coefficients are at most 1. Because of Cramer's rule, it thus suffices to prove the following lemma.

Lemma 20.2.2.

$$\Pr_{H\sim\binom{[m]}{2k}}\left[\left|\left\{ i \;\middle|\; i\in[m]\setminus H, i\notin R_{M^*}(H\cup\{i\})\right\}\right| \geq (m-2k)/4\right] \geq 1/4.$$

20.2.2 Correctness of the Algorithm

We write the input matrix A as $A^* + \Delta$, where $A^* \in \mathbb{R}^{n\times n}$ is the best rank-k approximation to A, and $\Delta \in \mathbb{R}^{n\times n}$ is the residual matrix with respect to A^*. Then $\|\Delta\|_g = \sum_{i=1}^n \|\Delta_i\|_g$ is the optimal cost. As shown in Algorithm 20.1, our approach iteratively eliminates all the columns. In each iteration, we sample a subset of columns, and use these columns to fit other columns. We drop a constant fraction of columns which have a good fitting cost. Suppose the indices of the columns surviving after the i-th outer iteration are $T_i = \{t_{i,1}, t_{i,2}, \cdots, t_{i,m_i}\} \subseteq [n]$. Without loss of generality, we can assume $\|\Delta_{t_{i,1}}\|_g \geq \|\Delta_{t_{i,2}}\|_g \geq \cdots \geq \|\Delta_{t_{i,m_i}}\|_g$. The following claim shows that if we randomly sample $2k$ column indices H from T_i, then the cost of Δ_H will not be large.

Claim 20.2.3. *If $|T_i| = m_i \geq 1000k$,*

$$\Pr_{H\sim\binom{T_i}{2k}}\left[\sum_{j\in H}\|\Delta_j\|_g \leq 400\frac{k}{m_i}\sum_{j=\frac{m_i}{100k}}^{m_i}\|\Delta_{t_{i,j}}\|_g\right] \geq \frac{19}{20}.$$

By an averaging argument, in the following claim, we can show that there is a constant fraction of columns in T_i whose optimal cost is also small.

Claim 20.2.4. *If* $|T_i| = m_i \geq 1000k$,

$$\left| \left\{ t_{i,j} \;\middle|\; t_{i,j} \in T_i, \|\Delta_{t_{i,j}}\|_g \geq \frac{20}{m_i} \sum_{j'=\frac{m_i}{100k}}^{m_i} \|\Delta_{t_{i,j'}}\|_g \right\} \right| \leq \frac{1}{5} m_i.$$

By combining Lemma 20.2.2, part (II) of Lemma 20.2.1 with the above two claims, it is sufficient to prove the following core lemma. It says that if we randomly choose a subset of $2k$ columns from T_i, then we can fit a constant fraction of the columns from T_i with a small cost.

Lemma 20.2.5. *If* $|T_i| = m_i \geq 1000k$,

$$\Pr_{H \sim \binom{T_i}{2k}} \left[\left| \left\{ j \;\middle|\; j \in T_i, \min_{x \in \mathbb{R}^{|H|}} \|A_H x - A_j\|_g \leq C_1 \cdot \frac{1}{m_i} \cdot \sum_{j'=\frac{m_i}{100k}}^{m_i} \|\Delta_{t_{i,j'}}\|_g \right\} \right| \geq \frac{1}{20} m_i \right] \geq \frac{1}{5},$$

where $C_1 = 500 \cdot k \cdot \mathrm{ati}_{g,|S|+1} \cdot \mathrm{mon}_g$.

Let us briefly explain why the above lemma is enough to prove the correctness of our algorithm. For each column $j \in [m]$, either the column j is in T_r and is selected by the end of the algorithm, or $\exists i < r$ such that $j \in T_i \setminus T_{i+1}$. If $j \in T_i \setminus T_{i+1}$, then by the above lemma, we can show that with high probability, $\min_x \|A_{S_{i+1}} x - A_j\|_g \leq O(C_1 \|\Delta\|_1 / |T_i|)$. Thus, $\min_X \|A_{S_{i+1}} X - A_{T_i \setminus T_{i+1}}\|_g \leq O(C_1 \|\Delta\|_1)$. It directly gives a $O(rC_1) = O(C_1 \log n)$ approximation. For the detailed proof of Theorem 20.1.2, we refer the reader to Appendix 20.3.

20.3 Missing Proofs in Section 20.2

20.3.1 Proof of Lemma 20.2.1

Proof. (I) Since $\mathrm{rank}(M^*_{H\cup\{i\}}) \le \mathrm{rank}(M^*) = k$, $|R_{M^*}(H\cup\{i\})| \le k$. Note that H is sampled from $\binom{[m]}{2k}$ uniformly at random, i is sampled from $[m]\backslash H$ uniformly at random, and $|H\cup\{i\}| = 2k+1$. By symmetry we have

$$\Pr_{H\sim\binom{[m]}{2k}, i\sim[m]\backslash H}[i \notin R_{M^*}(H\cup\{i\})] \ge 1 - \frac{k}{2k+1} \ge 1/2.$$

(II) Since $i \notin R_{M^*}(H\cup\{i\})$, by Cramer's rule, there exist $\alpha_1, \alpha_2, \cdots, \alpha_{|H|}$ such that $M^*_i = \sum_{j=1}^{|H|} \alpha_j \cdot (M^*_H)_j$ and $\forall j \in [|H|], |\alpha_j| \le 1$. Then we have

$$\begin{aligned}
\min_{x\in\mathbb{R}^{|H|}} \|M_H x - M_i\|_g \le & \left\| \sum_{j=1}^{|H|} (M_H)_j \alpha_j - M_i \right\|_g \\
= & \left\| \sum_{j=1}^{|H|} (M^*_H)_j \alpha_j - M^*_i + \sum_{j=1}^{|H|} (N_H)_j \alpha_j - N_i \right\|_g \\
= & \left\| \sum_{j=1}^{|H|} (N_H)_j \alpha_j - N_i \right\|_g \\
\le & \ \mathrm{ati}_{g,|H|+1} \cdot \left(\|-N_i\|_g + \sum_{j=1}^{|H|} \|(N_H)_j \alpha_j\|_g \right) \\
\le & \ \mathrm{ati}_{g,|H|+1} \cdot \left(\|-N_i\|_g + \mathrm{mon}_g \cdot \sum_{j=1}^{|H|} \|(N_H)_j\|_g \right) \\
\le & \ \mathrm{ati}_{g,|H|+1} \cdot \left(\mathrm{mon}_g \cdot \|N_i\|_g + \mathrm{mon}_g \cdot \sum_{j=1}^{|H|} \|(N_H)_j\|_g \right) \\
\le & \ \mathrm{ati}_{g,|H|+1} \cdot \mathrm{mon}_g \cdot \left(\|N_i\|_g + \sum_{j=1}^{|H|} \|(N_H)_j\|_g \right),
\end{aligned}$$

where the second step follows from $M = M^* + N$, the third step follows from $\sum_{j=1}^{|H|} (M_H^*)_j \alpha_j - M_i^* = 0$, the fourth step follows from the approximate triangle inequality, the fifth step follows from the fact that $|\alpha_j| \leq 1$ and g is mon_g-monotone, and the sixth step follows from that g is mon_g-monotone.

□

20.3.2 Proof of Lemma 20.2.2

Proof. Using Part (I) of Lemma 20.2.1, we have

$$\Pr_{H \sim \binom{[m]}{2k}, i \sim [m] \setminus H} [i \notin R_{M^*}(H \cup \{i\})] \geq 1/2.$$

For each set H, we define $P_H = \Pr_{i \sim [m] \setminus H}[i \notin R_{M^*}(H \cup \{i\})]$. We have

$$1 \geq \frac{1}{\binom{m}{2k}} \sum_{H \in \binom{[m]}{2k}} P_H \geq 1/2. \tag{20.2}$$

We can show

$$\begin{aligned}
&\frac{1}{\binom{m}{2k}}\left|\left\{H \;\middle|\; H \in \binom{[m]}{2k}, P_H \geq 1/4\right\}\right| \\
&= \frac{1}{\binom{m}{2k}} \sum_{H \in \binom{[m]}{2k}, P_H \geq 1/4} 1 \\
&\geq \frac{1}{\binom{m}{2k}} \sum_{H \in \binom{[m]}{2k}, P_H \geq 1/4} P_H \\
&\geq \frac{1}{2} - \frac{1}{\binom{m}{2k}} \sum_{H \in \binom{[m]}{2k}, P_H < 1/4} P_H \\
&\geq \frac{1}{2} - \frac{1}{\binom{m}{2k}} \sum_{H \in \binom{[m]}{2k}, P_H < 1/4} \frac{1}{4} \\
&\geq \frac{1}{2} - \frac{1}{\binom{m}{2k}} \binom{m}{2k} \frac{1}{4} \\
&= \frac{1}{4},
\end{aligned}$$

where the second step follows since $1 \geq P_H$, the third step follows since Eq. (20.2), the fourth step follows since $P_H < 1/4$.

Thus, we have

$$\left|\left\{H \;\middle|\; H \in \binom{[m]}{2k}, P_H \geq 1/4\right\}\right| \geq \binom{m}{2k}/4.$$

Recall the definition of P_H, we have

$$\left|\left\{H \;\middle|\; H \in \binom{[m]}{2k}, \Pr_{i \sim [m] \setminus H}[i \notin R_{M^*}(H \cup \{i\})] \geq 1/4\right\}\right| \geq \binom{m}{2k}/4,$$

which implies

$$\Pr_{H \sim \binom{[m]}{2k}}\left[\left|\left\{i \;\middle|\; i \in [m] \setminus H, i \notin R_{M^*}(H \cup \{i\})\right\}\right| \geq (m-2k)/4\right] \geq 1/4.$$

□

20.3.3 Proof of Claim 20.2.3

Proof. For simplicity, we omit i in all the subscripts in this proof.

$$
\begin{aligned}
&\Pr_{H\sim\binom{T}{2k}}\left[\sum_{j\in H}\|\Delta_j\|_g \le 400\frac{k}{m}\sum_{j=\frac{m}{100k}}^{m}\|\Delta_{t_j}\|_g\right]\\
=&\Pr_{H\sim\binom{T}{2k}}\left[\sum_{j\in H}\|\Delta_j\|_g \le 400\frac{k}{m}\sum_{j=\frac{m}{100k}}^{m}\|\Delta_{t_j}\|_g \;\middle|\; \exists j\le\frac{m}{100k}, t_j\in H\right]\cdot\Pr_{H\sim\binom{T}{2k}}\left[\exists j\le\frac{m}{100k}, t_j\in H\right]\\
+&\Pr_{H\sim\binom{T}{2k}}\left[\sum_{j\in H}\|\Delta_j\|_g \le 400\frac{k}{m}\sum_{j=\frac{m}{100k}}^{m}\|\Delta_{t_j}\|_g \;\middle|\; \forall j\le\frac{m}{100k}, t_j\notin H\right]\cdot\Pr_{H\sim\binom{T}{2k}}\left[\forall j\le\frac{m}{100k}, t_j\notin H\right]\\
\le&\underbrace{\Pr_{H\sim\binom{T}{2k}}\left[\exists j\le\frac{m}{100k}, t_j\in H\right]}_{C_1}+\underbrace{\Pr_{H\sim\binom{T}{2k}}\left[\sum_{j\in H}\|\Delta_j\|_g \le 400\frac{k}{m}\sum_{j=\frac{m}{100k}}^{m}\|\Delta_{t_j}\|_g \;\middle|\; \forall j\le\frac{m}{100k}, t_j\notin H\right]}_{C_2}
\end{aligned}
$$

It remains to upper bound the terms C_1 and C_2. We can upper bound C_1:

$$
\begin{aligned}
C_1 &= 1-(1-\frac{m/100k}{m})\cdot(1-\frac{m/100k}{m-1})\cdot\dots\cdot(1-\frac{m/100k}{m-2k+1})\\
&\le 1-(1-\frac{m/100k}{m/2})^{2k}\\
&\le 1-(1-\frac{1}{25})\\
&=\frac{1}{25},
\end{aligned}
$$

where the second step follows since $m\ge 1000k$.

Using Markov's inequality,

$$C_2 \le \frac{\mathop{\mathbb{E}}_{H\sim\binom{T}{2k}}\left[\sum_{j\in H}\|\Delta_j\|_g \le 400\frac{k}{m}\sum_{j=\frac{m}{100k}}^{m}\|\Delta_{t_j}\|_g \;\middle|\; \forall j\le\frac{m}{100k}, t_j\notin H\right]}{400\frac{k}{m}\sum_{j=\frac{m}{100k}}^{m}\|\Delta_{t_j}\|_g} \le 1/100,$$

where the second step follows since

$$\begin{aligned}
&\mathop{\mathbb{E}}_{H\sim\binom{T}{2k}}\left[\sum_{j\in H}\|\Delta_j\|_g \le 400\frac{k}{m}\sum_{j=\frac{m}{100k}}^{m}\|\Delta_{t_j}\|_g \;\middle|\; \forall j\le\frac{m}{100k}, t_j\notin H\right]\\
&\le \frac{2k}{m-m/100k}\sum_{j=\frac{m}{100k}}^{m}\|\Delta_{t_j}\|_g\\
&\le 4\frac{k}{m}\sum_{j=\frac{m}{100k}}^{m}\|\Delta_{t_j}\|_g
\end{aligned}$$

□

20.3.4 Proof of Claim 20.2.4

Proof. For simplicity, we omit i in all the subscripts in this proof.

$$\begin{aligned}
&\left|\left\{t_j \;\middle|\; t_j\in T, \|\Delta_{t_j}\|_g \ge \frac{20}{m}\sum_{j'=\frac{m}{100k}}^{m}\|\Delta_{t_{j'}}\|_g\right\}\right|\\
\le\ &\left|\left\{t_j \;\middle|\; t_j\in T, j\le\frac{m}{100k}\right\}\right| + \left|\left\{t_j \;\middle|\; t_j\in T, j>\frac{m}{100k}, \|\Delta_{t_j}\|_g \ge \frac{20}{m}\sum_{j'=\frac{m}{100k}}^{m}\|\Delta_{t_{j'}}\|_g\right\}\right|\\
\le\ &\left|\left\{t_j \;\middle|\; t_j\in T, j\le\frac{m}{100k}\right\}\right| + \left|\left\{t_j \;\middle|\; t_j\in T, j>\frac{m}{100k}, \|\Delta_{t_j}\|_g \ge \frac{10}{m-\frac{m}{100k}}\sum_{j'=\frac{m}{100k}}^{m}\|\Delta_{t_{j'}}\|_g\right\}\right|\\
\le\ &\frac{m}{100k}+\frac{1}{10}(m-\frac{m}{100k})\\
\le\ &\frac{1}{5}m,
\end{aligned}$$

where the second step follows since $\frac{20}{m} \geq \frac{10}{m-m/100k}$ $\square$

20.3.5 Proof of Lemma 20.2.5

Proof. For simplicity, we omit i in all the subscirptis in this proof.

Let $M = A_T$, $M^* = A_T^*$ and $N = \Delta_T$. Then we can apply Lemma 20.2.2 and part (II) of Lemma 20.2.1:

$$\Pr_{H\sim\binom{T}{2k}}\left[\left|\left\{j \in T \;\middle|\; \min_{x\in\mathbb{R}^{|H|}} \|A_H x - A_j\|_g \leq \mathrm{ati}_{g,|H|+1}\cdot \mathrm{mon}_g \cdot \left(\|\Delta_j\|_g + \sum_{j'=1}^{|H|}\|(\Delta_H)_{j'}\|_g\right)\right\}\right| \geq \frac{m}{4}\right] \geq \frac{1}{4} \tag{20.3}$$

By Claim 20.2.3, we have

$$\Pr_{H\sim\binom{T}{2k}}\left[\sum_{j=1}^{|H|}\|(\Delta_H)_j\|_g \leq 400\frac{k}{m}\sum_{j=\frac{m}{100k}}^{m}\|\Delta_{t_j}\|_g\right] \geq \frac{19}{20} \tag{20.4}$$

Due to Claim 20.2.4,

$$\left|\left\{t_j \;\middle|\; t_j \in T, \|\Delta_{t_j}\|_g \geq \frac{20}{m}\sum_{j'=\frac{m}{100k}}^{m}\|\Delta_{t_{j'}}\|_g\right\}\right| \leq \frac{1}{5}m$$

Combining the above equation with the pigeonhole principle, for any $I \subseteq T$ with $|I| \geq m/4$, we have

$$\left|\left\{t_j \;\middle|\; t_j \in I, \|\Delta_{t_j}\|_g < \frac{20}{m}\sum_{j'=\frac{m}{100k}}^{m}\|\Delta_{t_{j'}}\|_g\right\}\right| \geq \frac{1}{4}m - \frac{1}{5}m = \frac{1}{20}m \tag{20.5}$$

Consider the quantity $\|\Delta_j\|_g + \sum_{j=1}^{|H|}\|(\Delta_H)_{j'}\|_g$ in Eq. (20.3). We use Eq. (20.4) and Eq. (20.5) to provide an upper bound,

$$\|\Delta_j\|_g + \sum_{j'=1}^{|H|}\|(\Delta_H)_{j'}\|_g \leq \left(\frac{20}{m} + \frac{400k}{m}\right)\sum_{j'=\frac{m}{100k}}^{m}\|\Delta_{t_{j'}}\|_g.$$

Eq. (20.4) will decrease the final probability by $(1 - 19/20)$ (from $1/4$ to $1/4 - 1/20$). Eq. (20.5) will decrease the size of this set of j by $\frac{1}{5}m$ (from $\frac{1}{4}m$ to $\frac{1}{4}m - \frac{1}{5}m$).

Putting it all together, we can update Eq. (20.3) in the following sense,

$$\Pr_{H\sim\binom{T}{2k}}\left[\left|\left\{ j \;\middle|\; j \in T, \min_{x\in\mathbb{R}^{|H|}} \|A_H x - A_j\|_g \leq C \cdot \frac{1}{m} \cdot \sum_{j'=\frac{m}{100k}}^{m} \|\Delta_{t_{j'}}\|_g \right\}\right| \geq (\frac{1}{4} - \frac{1}{5})m\right] \geq \frac{1}{4} - \frac{1}{20}.$$

where

$$C = (400 + 20) \cdot k \cdot \mathrm{ati}_{g,|H|+1} \cdot \mathrm{mon}_g \,.$$

□

20.3.6 Proof of Theorem 20.1.2

Proof. The running time is discussed at the beginning of Section 20.2. In the remaining of the proof, we will focus on the correctness of Algorithm 20.1.

Firstly, let us consider the size of the output S. For $i \in \{0\} \cup [r]$, let $m_i = |T_i|$. We set number of rounds r to be the smallest value such that $m_r < 1000k$. By the algorithm, we have $m_i = m_{i-1} - 2k - (m_{i-1} - 2k)/20 \leq 19/20 \cdot m_{i-1}$. Thus, $r = O(\log n)$. In each round i, the size of S_i is $2k$. Then $|S| = |T_r| + \sum_{i=1}^{r} |S_i| \leq 1000k + r \cdot 2k \leq O(k \log n)$.

Next, let us consider the quality of S. Since each regression call has $1 - 1/\operatorname{poly}(n)$ success probability, all the regression calls succeed with probability at least $1 - 1/\operatorname{poly}(n)$. In the remaining of the proof, we condition on that all the regression calls succeed.

Let us fix $i \in [r], j \in [\log n]$. Recall that $T_i = \{t_{i,1}, t_{i,2}, \cdots, t_{i,m_i}\}$ and $\|\Delta_{t_{i,1}}\|_g \geq \|\Delta_{t_{i,2}}\|_g \geq \cdots \geq \|\Delta_{t_{i,m_i}}\|_g$. By regression property and Lemma 20.2.5, with probability at

least $1/4$,

$$\begin{aligned}
&\sum_{q\in R_i^{(j)}} \min_{x\in\mathbb{R}^{2k}} \|A_{S_i^{(j)}}x - A_q\|_g \\
\leq\ &\sum_{q\in R_i^{(j)}} v_{i,q}^{(j)} \\
\leq\ &\mathrm{reg}_{g,2k}\cdot(m_i-2k)\cdot 500\cdot k\cdot \mathrm{ati}_{g,2k+1}\cdot \mathrm{mon}_g\, /m_i \cdot \sum_{j'=\frac{m_i}{100k}}^{m_i} \|\Delta_{t_{i,j'}}\|_g \\
\leq\ &\mathrm{reg}_{g,2k+1}\cdot \mathrm{ati}_{g,2k+1}\cdot \mathrm{mon}_g \cdot O(k)\cdot \sum_{j'=\frac{m_i}{100k}}^{m_i} \|\Delta_{t_{i,j'}}\|_g.
\end{aligned}$$

For each $i\in[r]$, since we repeat $\log(n)$ times, the success probability can be boosted to at least $1-1/\operatorname{poly}(r)$, i.e., with probability at least $1-1/\operatorname{poly}(r)$, we have

$$\sum_{q\in R_i} \min_{x\in\mathbb{R}^{2k}} \|A_{S_i}x - A_q\|_g \leq \mathrm{reg}_{g,2k+1}\cdot \mathrm{ati}_{g,2k+1}\cdot \mathrm{mon}_g \cdot O(k)\cdot \sum_{j'=\frac{m_i}{100k}}^{m_i} \|\Delta_{t_{i,j'}}\|_g. \tag{20.6}$$

In the remaining of the proof, we condition on above inequality for every $i\in[r]$. Without

loss of generality, we suppose $\|\Delta_1\|_g \geq \|\Delta_2\|_g \geq \cdots \geq \|\Delta_n\|_g$. We have

$$
\begin{aligned}
& \sum_{q=1}^{n} \min_{x \in \mathbb{R}^{|S|}} \|A_S x - A_q\|_g \\
\leq & \left(\sum_{q \in T_r} \min_{x \in \mathbb{R}^{|T_r|}} \|A_{T_r} x - A_q\|_g \right) + \left(\sum_{i=1}^{r} \sum_{q \in T_{i-1} \setminus T_i} \min_{x \in \mathbb{R}^{2k}} \|A_{S_i} x - A_q\|_g \right) \\
= & \sum_{i=1}^{r} \sum_{q \in T_{i-1} \setminus T_i} \min_{x \in \mathbb{R}^{2k}} \|A_{S_i} x - A_q\|_g \\
\leq & \sum_{i=1}^{r} \mathrm{reg}_{g,2k+1} \cdot \mathrm{ati}_{g,2k+1} \cdot \mathrm{mon}_g \cdot O(k) \cdot \sum_{j'=\frac{m_i}{100k}}^{m_i} \|\Delta_{t_{i,j'}}\|_g \\
\leq & \sum_{i=1}^{r} \mathrm{reg}_{g,2k+1} \cdot \mathrm{ati}_{g,2k+1} \cdot \mathrm{mon}_g \cdot O(k) \cdot \sum_{j'=\frac{m_i}{100k}}^{m_i} \|\Delta_{j'}\|_g \\
= & \mathrm{reg}_{g,2k+1} \cdot \mathrm{ati}_{g,2k+1} \cdot \mathrm{mon}_g \cdot O(k) \cdot \sum_{j'=1}^{n} \|\Delta_{j'}\|_g \left(\operatorname*{argmin}_{i \in [r]} \{m_i < j'\} - \operatorname*{argmin}_{i \in [r]} \left\{ \frac{m_i}{100k} < j' \right\} + O(1) \right) \\
\leq & \mathrm{reg}_{g,2k+1} \cdot \mathrm{ati}_{g,2k+1} \cdot \mathrm{mon}_g \cdot O(k \log k) \cdot \|\Delta\|_g,
\end{aligned}
$$

where the third step follows from $T_{i-1} \setminus T_i = S_i \cup R_i$ and Equation (20.6), the forth step follows from $\|\Delta_{j'}\|_g \geq \|\Delta_{t_{i,j'}}\|_g$, and the last step follows from

$$
\left(\operatorname*{argmin}_{i \in [r]} \{m_i < j'\} - \operatorname*{argmin}_{i \in [r]} \left\{ \frac{m_i}{100k} < j' \right\} + O(1) \right) \leq O(\log k).
$$

□

20.4 Necessity of the Properties of g

We note that an approximate triangle inequality is necessary to obtain a column subset selection algorithm. An example function not satisfying this is the "jumping function": $g_\tau(x) = |x|$ if $|x| \geq \tau$, and $g_\tau(x) = 0$ otherwise. For the identity matrix I and any $k = \Omega(\log n)$, the Johnson-Lindenstrauss lemma implies one can find a rank-k matrix B for which $\|I - B\|_\infty < 1/2$, that is, all entries of $I - B$ are at most $1/2$. If we set $\tau = 1/2$, then $\|I - B\|_{g_\tau} = 0$, but for any subset I_S of columns of the identity matrix we choose, necessarily $\|I - I_S X\|_\infty \geq 1$, so $\|I - B\|_{g_\tau} > 0$. Consequently, there is no subset of a small number of columns which obtains a $\mathrm{poly}(k \log n)$-approximation with the jumping function loss measure.

While the jumping function does not satisfy the Approximate triangle inequality, it does satisfy our only other required structural property, the Monotone property.

There are interesting examples of functions g which are only approximately monotone in the above sense, such as the quantile function $\rho_\tau(x)$, studied in [YMM14] in the context of regression, where for a given parameter τ, $\rho_\tau(x) = \tau x$ if $x \geq 0$, and $\rho_\tau(x) = (\tau - 1)x$ if $x < 0$. Only when $\tau = 1/2$ is this a monotone function with $\mathrm{mon}_g = 1$ in the above definition, in which case it coincides with the absolute value function up to a factor of $1/2$. For other constant $\tau \in (0, 1)$, mon_g is a constant. The loss function $\rho_\tau(x)$ is also sometimes called the scalene loss, and studied in the context of low rank approximation in [UHZ+16].

When $\tau = 1$ this is the so-called Rectified Linear Unit (ReLU) function in machine learning, i.e., $\rho_1(x) = x$ if $x \geq 0$ and $\rho_1(x) = 0$ if $x < 0$. In this case $\mathrm{mon}_g = \infty$. and the optimal rank-k approximation for any matrix A is 0, since $\|A - \lambda \mathbf{1}\mathbf{1}^\top\|_{\rho_1} = 0$ if one sets λ to

be a large enough positive number, thereby making all entries of $A - \lambda \mathbf{1}\mathbf{1}^\top$ negative and their corresponding cost equal to 0. Notice though, that there are no good column subset selection algorithms for some matrices A, such as the $n \times n$ identity matrix. Indeed, for the identity, if we choose any subset A_S of at most $n-1$ columns of A, then for any matrix X there will be an entry of $A - A_S X$ which is positive, causing the cost to be positive. Since we will restrict ourselves to column subset selection, being approximately monotone with a small value of mon_g in the above definition is in fact necessary to obtain a good approximation with a small number of columns, as the ReLU function illustrates (see also related functions such as the leaky ReLU and squared ReLU [ZSJ+17, BHL18, GKM18]).

Note that the ReLU function is an example which satisfies the triangle inequality, showing that our additional assumption of approximate monotonicity is required.

Thus, if either property fails to hold, there need not be a small subset of columns spanning a relative error approximation. These examples are stated in more detail below.

20.4.1 Functions without Approximate Triangle Inequality

In this section, we show how to construct a function f such that it is not possible to obtain a good entrywise-f low rank approximation by selecting a small subset of columns. Furthermore, f is monotone but does not have the approximate triangle inequality. Theorem 20.4.3 shows this result.

First, we show that a small subset of columns cannot give a good low rank approximation in ℓ_∞ norm. Then we reduce the ℓ_∞ column subset selection problem to the entrywise-f column subset selection problem.

The following is the Johnson-Lindenstrauss lemma.

Lemma 20.4.1 (JL Lemma)**.** *For any $n \geq 1, \epsilon \in (1/\sqrt{n}, 1/2)$, there exists $U \in \mathbb{R}^{n \times k}$ with $k = O(\epsilon^{-2} \log(n))$ such that $\|UU^\top - I_n\|_\infty \leq O(\epsilon)$, where $I_n \in \mathbb{R}^{n \times n}$ is an identity matrix.*

Theorem 20.4.2. *For $n \geq 1$, there is a matrix $A \in \mathbb{R}^{n \times n}$ with the following properties. Let $k = \Theta(\epsilon^{-2} \log(n))$ for an arbitrary $\epsilon \in (1/\sqrt{n}, 1/2)$. Let $D \in \mathbb{R}^{n \times n}$ denote a diagonal matrix with $n - 1$ nonzeros on the diagonal. We have*

$$\min_{X \in \mathbb{R}^{n \times n}} \|XDA - A\|_\infty \geq 1$$

and

$$\min_{\text{rank}-k \ A'} \|A' - A\|_\infty < O(\epsilon).$$

Proof. We choose A to be the identity matrix. By Lemma 20.4.1, we can find a rank-k matrix B for which

$$\|A - B\|_\infty \leq O(\epsilon).$$

Since A is an $n \times n$ identity matrix, even if we can use $n - 1$ columns to fit the other columns, the cost is still at least 1. □

In the following, we state the construction of our function f.

Definition 20.4.1. We define function $f(x)$ to be $f(x) = c$ if $|x| > \tau$ and $f(x) = 0$ if $|x| \leq \tau$. Given matrix A, we define $\|A\|_f = \sum_{i=1}^n \sum_{j=1}^n f(A_{i,j})$.

Theorem 20.4.3 (No good subset of columns)**.** *For any $n \geq 1$, there is a matrix $A \in \mathbb{R}^{n \times n}$ with the following property. Let $k \geq c \log n$ for a sufficiently large constant $c > 0$. Let*

$D \in \mathbb{R}^{n \times n}$ denote an arbitrary diagonal matrix with $n-1$ nonzeros on the diagonal. For f with parameter $\tau = 1/4$, we have

$$\min_{X \in \mathbb{R}^{n \times n}} \|XDA - A\|_f > 0$$

and

$$\min_{\text{rank}-k \ A'} \|A' - A\|_f = 0.$$

Proof. We can set A to be the identity matrix. Due to Theorem 20.4.2, there exists A' for which $\min_{\text{rank}-k \ A'} \|A' - A\|_\infty < 1/4$, which implies that $\min_{\text{rank}-k \ A'} \|A' - A\|_f = 0$. Also due to Theorem 20.4.2, we have $\min_{X \in \mathbb{R}^{n \times n}} \|XDA - A\|_\infty = 1$, and thus, $\min_{X \in \mathbb{R}^{n \times n}} \|XDA - A\|_f > 0$.

□

20.4.2 ReLU Function Low Rank Approximation

In this section, we discuss a function which has the approximate triangle inequality but is not symmetric. The specific function we discuss in this section is ReLU . The definition of ReLU is defined in Definition 20.4.2. First, we show that ReLU low rank approximation has a trivial best rank-k approximation. Second, we show that for some matrices, there is no small subset of columns which can give a good low rank approximation.

Definition 20.4.2. We define function $\text{ReLU}(x)$ to be $\text{ReLU}(x) = \max(0, x)$. Given matrix A, we define $\|A\|_{\text{ReLU}} = \sum_{i=1}^{n} \sum_{j=1}^{n} \text{ReLU}(A_{i,j})$.

In the rank-k approximation problem, given an input matrix A, the goal is to find a rank-k matrix B for which $\|A - B\|_{\text{ReLU}}$ is minimized. A simple observation is that if we set

B to be a matrix with each entry of value $\|A\|_\infty$, then the value of each entry of $A - B$ is at most 0. Thus, $\|A - B\|_{\mathrm{ReLU}} = 0$. Furthermore, the rank of B is 1.

Now, consider the column subset selection problem, let input matrix $A \in \mathbb{R}^{n \times n}$ be an identity matrix. Then even if we can choose $n - 1$ columns, they can never fit the remaining column. Thus, the cost is at least 1. But as discussed, the best rank-k cost is always 0. This implies that any subset of columns cannot give a good rank-k approximation.

20.5 Regression Solvers

In this section, we discuss several regression solvers.

20.5.1 Regression for Convex g

Notice that when the function g is convex, the regression problem $\min_{X\in\mathbb{R}^{d\times m}} \|AX - B\|_g$ for any given matrices $A \in \mathbb{R}^{n\times d}, B \in \mathbb{R}^{n\times m}$ is a convex optimization problem. Thus, it can be solved exactly by convex optimization algorithms.

Fact 20.5.1. *Let g be a convex function. Given $A \in \mathbb{R}^{n\times d}, B \in \mathbb{R}^{n\times m}$, the regression problem $\min_{X\in\mathbb{R}^{d\times m}} \|AX - B\|_g$ can be solved exactly by convex optimization in $\mathrm{poly}(n, d, m)$ time.*

If a function g has additional properties, i.e. g is symmetric, monotone and grows subquadratically, then there is a better running time constant approximation algorithm shown in [CW15b]. Here "grows quadratically" means that there is an $\alpha \in [1, 2]$ and $c_g > 0$ so that for a, a' with $|a| > |a'| > 0$,

$$\left|\frac{a}{a'}\right|^\alpha \geq \frac{g(a)}{g(a')} \geq c_g \left|\frac{a}{a'}\right|.$$

This kind of function g is also called a "sketchable" function. Notice that the Huber function satisfies the above properties.

Theorem 20.5.2 (Modified version of Theorem 3.1 of [CW15b])**.** *Function g is symmetric, monotone and grows subquadratically (g is a G-function defined by [CW15b]). Given a matrix $A \in \mathbb{R}^{n\times d}$ and a matrix $B \in \mathbb{R}^{n\times m}$, there is an algorithm which can output a matrix $\widehat{X} \in \mathbb{R}^{d\times m}$ and a fitting cost vector $y \in \mathbb{R}^m$ such that with probability at least $1-1/\mathrm{poly}(nm)$, $\forall i \in [m], \|A\widehat{X}_i - B_i\|_g \leq O(1)\cdot \min_{x\in\mathbb{R}^d} \|Ax - B_i\|_g$, and $y_i = \Theta(\|A\widehat{X}_i - B_i\|_g)$. Furthermore, the running time is at most $\widetilde{O}(\mathrm{nnz}(A) + \mathrm{nnz}(B) + m \cdot \mathrm{poly}(d \log n))$.*

Proof. We run $O(\log(nm))$ repetitions of the single column regression algorithm shown in Theorem 3.1 of [CW15b] for all columns B_i for $i \in [m]$. For each regression problem $\|Ax - B_i\|_g$, we take the solution whose estimated cost is the median among these $O(\log(nm)$ repetitions as $\widehat{X}_i$. Then by the Chernoff bound, we can boost the success probability of each column to $1 - 1/\operatorname{poly}(nm)$. By taking a union bound over all columns, we complete the proof. □

20.5.2 ℓ_p Regression

One of the most important cases in regression and low rank approximation problems is when the error measure is ℓ_p. For ℓ_p regression, though it can be solved by convex optimization/linear programming exactly, we can get a much faster running time if we allow some approximation ratios. In the following theorem, we show that there is an algorithm which can be used to solve ℓ_p regression for any $p \geq 1$.

Theorem 20.5.3 (Modified version of [WZ13])**.** *Let $p \geq 1, \epsilon \in (0, 1)$. Given a matrix $A \in \mathbb{R}^{n \times d}$ and a matrix $B \in \mathbb{R}^{n \times m}$, there is an algorithm which can output a matrix $\widehat{X} \in \mathbb{R}^{d \times m}$ and a fitting cost vector $y \in \mathbb{R}^m$ such that with probability at least $1 - 1/\operatorname{poly}(nm)$, $\forall i \in [m], \|A\widehat{X}_i - B_i\|_p^p \leq (1+\epsilon) \cdot \min_{x \in \mathbb{R}^d} \|Ax - B_i\|_p^p$, and $y_i = \Theta(\|A\widehat{X}_i - B_i\|_p^p)$. Furthermore, the running time is at most $\widetilde{O}(\operatorname{nnz}(A) + \operatorname{nnz}(B) + mn^{\max(1-2/p,0)} \cdot \operatorname{poly}(d))$.*

Proof. As in the proof of Theorem 20.5.2, we only need to run $O(\log(nm))$ repetitions of the single column regression algorithm shown in [WZ13]. □

20.5.3 ℓ_0 Regression

Definition 20.5.1 (Regular partition)**.** Given a matrix $A \in \mathbb{R}^{n\times k}$, we say $\{S_1, S_2, \cdots, S_h\}$ is a regular partition for $[n]$ with respect to the matrix A if, for each $i \in [h]$,

$$\mathrm{rank}(A^{S_i}) = |S_i|, \text{ and } \mathrm{rowspan}(A^{S_i}) = \mathrm{rowspan}\left(A^{\cup_{j=i}^{h} S_j}\right),$$

where $A^{S_i} \in \mathbb{R}^{|S_i|\times k}$ denotes the matrix that selects a subset S_i of rows of the matrix A.

Algorithm 20.2 ℓ_0 regression [APY09]

procedure L0Regression(A, b, n, k, c) ▷ Theorem 20.5.4
  $x' \leftarrow 0^k$
  $\{S_1, S_2, \cdots, S_h\} \leftarrow$ GenerateRegularPartition(A, n, k)
  $x' \leftarrow 0^k$
  for $i = 1 \to h$ **do**
    Find a $\widetilde{x}$ such that $A^{S_i}\widetilde{x} = b_{S_i}$
    if $\|A\widetilde{x} - b\|_0 < \|Ax' - b\|_0$ **then**
      $x' \leftarrow \widetilde{x}$
    end if
  end for
  return x'
end procedure

[APY09] studied the Nearest Codeword problem over finite fields $\mathbb{F}_2$. Their proof can be extended to the real field and generalized to Theorem 20.5.4. For completeness, we still provide the proof of the following result.

Theorem 20.5.4 (Generalization of [APY09])**.** *Given matrix $A \in \mathbb{R}^{n\times k}$ and vector $\mathbb{R}^n$, for any $c \in [1, k]$, there is an algorithm (Algorithm 20.2) that runs in $n^{O(1)}$ time and outputs a vector $x' \in \mathbb{R}^k$ such that*

$$\|Ax' - b\|_0 \leq k \min_{x\in\mathbb{R}^k} \|Ax - b\|_0.$$

Proof. Let $x^* \in \mathbb{R}^k$ denote the optimal solution to $\min_{x\in\mathbb{R}^k} \|Ax - b\|_0$. We define set E as follows

$$E = \{i \in [n] \mid (Ax^*)_i \neq b_i\}.$$

We create a regular partition $\{S_1, S_2, \cdots, S_h\}$ for $[n]$ with respect to A.

Let i denote the smallest index such that $|S_i \cap E| = 0$, i.e.,

$$i = \min\{j \mid |S_j \cap E| = 0\}.$$

The linear equation we want to solve is $A^{S_i}x = b_{S_i}$. Let $\widetilde{x} \in \mathbb{R}^k$ denote a solution to $A^{S_i}\widetilde{x} = A^{S_i}x^*$ (Note that, by our choice of i, $b_{S_i} = A^{S_i}x^*$). Then we can rewrite $\|A\widetilde{x} - b\|_0$ in the following sense,

$$\|A\widetilde{x} - b\|_0 = \sum_{j=1}^{i-1} \left\|A^{S_j}\widetilde{x} - b_{S_j}\right\|_0 + \sum_{j=i}^{h} \left\|A^{S_j}\widetilde{x} - b_{S_j}\right\|_0. \tag{20.7}$$

For each $j \in \{1, 2, \cdots, i-1\}$, we have

$$\begin{aligned} \|A^{S_j}\widetilde{x} - b_{S_j}\|_0 &\leq k \\ &\leq \|A^{S_j}x^* - b_{S_j}\|_0 \cdot \lceil k \rceil, \end{aligned} \tag{20.8}$$

where the first step follows from $|S_0| \leq k$, and the last step follows from $\|A^{S_j}x^* - b_{S_j}\|_0 \geq 1$, $\forall j \in [i-1]$.

Note that, by our choice of i, we have $A^{S_i}\widetilde{x} = A^{S_i}x^*$. Then for each $j \in \{i, i+1, \cdots, n\}$, using the regular partition property, there always exists a matrix $P_{(j)}$ such that $A^{S_j} = P_{(j)}A^{S_i}$. Then we have

$$A^{S_j}\widetilde{x} = P_{(j)}A^{S_i}\widetilde{x} = P_{(j)}A^{S_i}x^* = A^{S_j}x^*. \tag{20.9}$$

Plugging Eq. (20.8) and (20.9) into Eq. (20.7), we have

$$\begin{aligned}
\|A\widetilde{x} - b\|_0 &= \sum_{j=1}^{i-1} \left\|A^{S_j}\widetilde{x} - b_{S_j}\right\|_0 + \sum_{j=i}^{h} \left\|A^{S_j}\widetilde{x} - b_{S_j}\right\|_0 \\
&\leq k \sum_{j=1}^{i-1} \left\|A^{S_j}x^* - b_{S_j}\right\|_0 + \sum_{j=i}^{h} \left\|A^{S_j}x^* - b_{S_j}\right\|_0 \\
&\leq k\|Ax^* - b\|_0.
\end{aligned}$$

This completes the proof. □

20.6 Hardness

20.6.1 Column Subset Selection for the Huber Function

The rough idea here is to define $k = \Omega(\sqrt{\log n})$ groups of columns, where we carefully choose the i-th group to have $n^{1-2i\epsilon}$ columns, $\epsilon = .2/(1.5k)$, and in the i-th group each column has the form

$$n^{1.5i\epsilon} \cdot 1^n + [\pm n^{-.2+i\epsilon}, \ldots, \pm n^{-.2+i\epsilon}, \pm n^{.5+2i\epsilon}, \ldots, \pm n^{.5+2i\epsilon}],$$

where there are $n - n^{.1}$ coordinates where the perturbation is randomly either $+n^{-.2+i\epsilon}$ or $-n^{-.2+i\epsilon}$, and the remaining $n^{.1}$ coordinates are randomly either $+n^{.5+2i\epsilon}$ or $-n^{.5+2i\epsilon}$. We call the former type of coordinates "small noise", and the latter "large noise". All remaining columns in the matrix are set to 0. Because of the random signs, it is very hard to fit the noise in one column to that of another column. One can show, that to approximate a column in the j-th group by a column in the i-th group, $i < j$, one needs to scale by roughly $n^{1.5(j-i)\epsilon}$, just to cancel out the "mean" $n^{1.5j\epsilon} \cdot 1^n$. But when doing so, since the Huber function is quadratic for small values, the scaled small noise is now magnified more than linearly compared to what it was before, and this causes a column in the i-th group not to be a good approximation of a column in the j-th group. On the other hand, if you want to approximate a column in the j-th group by a column in the i-th group, $i > j$, one again needs to scale by roughly $n^{1.5(j-i)\epsilon}$ just to cancel out the "mean", but now one can show the large noise from the column in the i-th group is too large and remains in the linear regime, causing a poor approximation. The details of this construction are given in the following theorem.

Theorem 20.6.1. *Let $H(x)$ denote the modified Huber function with $\tau = 1$, i.e.,*

$$H(x) = \begin{cases} x^2/\tau, & \text{if } |x| < \tau; \\ |x|, & \text{if } |x| \geq \tau. \end{cases}$$

For any $n \geq 1$, there is a matrix $A \in \mathbb{R}^{n \times n}$ such that, if we select $o(\sqrt{\log n})$ columns to fit the entire matrix, there is no $O(1)$-approximation, i.e., for any subset $S \subseteq [n]$ with $|S| = o(\sqrt{\log n})$,

$$\min_{X \in \mathbb{R}^{|S| \times n}} \|A_S X - A\|_H \geq \omega(1) \cdot \min_{\text{rank}-1 \ A'} \|A' - A\|_H.$$

Proof. Suppose there is an algorithm which only finds a subset with size $k/2 = o(\sqrt{\log n})$. We want to prove a lower bound on its approximation ratio.

Let $\epsilon = 0.2/(1.5k)$. Let A denote a matrix with $k+1$ groups of columns.

For each group $i \in [k]$, I_i has $n^{1-2i\epsilon}$ columns which are

$$\begin{bmatrix} n^{1.5i\epsilon} \\ n^{1.5i\epsilon} \\ n^{1.5i\epsilon} \\ \vdots \\ n^{1.5i\epsilon} \\ n^{1.5i\epsilon} \\ n^{1.5i\epsilon} \\ n^{1.5i\epsilon} \end{bmatrix} + \begin{bmatrix} \pm n^{-0.2+i\epsilon} \\ \pm n^{-0.2+i\epsilon} \\ \vdots \\ \pm n^{-0.2+i\epsilon} \\ \pm n^{0.5+2i\epsilon} \\ \pm n^{0.5+2i\epsilon} \\ \vdots \\ \pm n^{0.5+2i\epsilon} \end{bmatrix} \in \mathbb{R}^n,$$

where $\pm$ indicates i.i.d. random signs. For the error column, the first $n - n^{0.1}$ rows are $n^{-0.2+i\epsilon}$, and the last $n^{0.1}$ rows are $n^{0.5+2i\epsilon}$.

The last group of $n - \sum_{i=1}^{k} n^{1-2i\epsilon}$ columns are

$$\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix} \in \mathbb{R}^n.$$

The optimal cost is at most

$$\begin{aligned} \sum_{i=1}^{k} n^{1-2i\epsilon} \cdot ((n - n^{0.1})H(n^{-0.2+i\epsilon}) + n^{0.1}H(n^{0.5+2i\epsilon})) &\leq \sum_{i=1}^{k} n^{1-2i\epsilon}(n \cdot n^{-0.4+2i\epsilon} + n^{0.6+2i\epsilon}) \\ &\leq O(kn^{1.6}), \end{aligned}$$

where the second step follows since $n^{-0.2+i\epsilon} < 1$ and $n^{0.5+2i\epsilon} \geq 1$. Thus, it implies

$$\min_{\text{rank}-1\ A'} \|A' - A\|_H \leq O(kn^{1.6}).$$

Now let us consider the lower bound for using a subset of columns to fit the matrix. First, we fix a set $S = \{j_1, j_2, \cdots, j_{k/2}\}$ of $k/2$ columns. Since there are k groups, and $|S| \leq k/2$, the number of groups I_i for $i \in [k]$ with $S \cap I_i = \emptyset$ is at least $k/2$. It means that there are at least $k/2$ groups for which S does not have any column from them. Notice that the optimal cost is at most $O(kn^{1.6})$, so it suffices to prove that $\forall i \in [k]$ with $I_i \cap S = \emptyset$, each column $j \in I_i$ will contribute a cost of $\omega(n^{0.6+2i\epsilon})$.

For notation, we use group(j) to denote the index of the group which contains the column j. For each column j, we use Δ_j to denote the noise part, and use A_j^* to denote the rank-1 "ground truth" part. Notice that $A_j = A_j^* + \Delta_j$.

Claim 20.6.2 (Noise cannot be used to fit other vectors)**.** *Let $x_1, x_2, \cdots, x_s \in \mathbb{R}^m$ be s random sign vectors. Then with probability at least $1 - 2^s \cdot 2^{-\Theta(m/2^s)}$, $\forall \alpha_1, \alpha_2, \cdots, \alpha_s \in \mathbb{R}$,*

the size of $Z_{\alpha_1,\alpha_2,\cdots,\alpha_s} = \{i \in [m] \mid \mathrm{sign}((\alpha_1 x_1)_i) = \mathrm{sign}((\alpha_2 x_2)_i) = \cdots = \mathrm{sign}((\alpha_s x_s)_i) = \mathrm{sign}(+1)\}$ *is at least* $\Omega(m/2^s)$.

Proof. For a set of fixed $\alpha_1, \alpha_2, \cdots, \alpha_s$, the claim follows from the Chernoff bound. Since there are 2^s different possibilities of signs of $\alpha_1, \cdots, \alpha_s$, taking a union bound over them completes the proof. □

Now we consider a specific column $j \in I_i$ for some $i \in [k]$, where $I_i \cap S = \emptyset$. Suppose the fitting coefficients are $\alpha_1, \alpha_2, \cdots, \alpha_{k/2}$. Consider the following term

$$\begin{aligned}&(\alpha_1 A_{j_1} + \alpha_2 A_{j_2} + \cdots + \alpha_{k/2} A_{k/2}) - A_j \\ =\ &(\alpha_1 A^*_{j_1} + \alpha_2 A^*_{j_2} + \cdots + \alpha_{k/2} A^*_{k/2} - A^*_j) + (\alpha_1 \Delta_{j_1} + \alpha_2 \Delta_{j_2} + \cdots + \alpha_{k/2} \Delta_{k/2} - \Delta_j)\end{aligned}$$

Let $u^* = (\alpha_1 A^*_{j_1} + \alpha_2 A^*_{j_2} + \cdots + \alpha_{k/2} A^*_{k/2} - A^*_j)$. By Claim 20.6.2, with probability at least $1 - (k/2) \cdot 2^{-\Theta(n/2^{k/2})}$,

$$|\{t \in [n] \mid \mathrm{sign}(u^*_t) = \mathrm{sign}(\alpha_1 \Delta_{j_1,t}) = \cdots = \mathrm{sign}(\alpha_{k/2} \Delta_{j_{k/2},t}) = \mathrm{sign}(-\Delta_{j,t})\}| = \Omega(n/2^{k/2}).$$

Observe that all the coordinates of u^* are the same, and the absolute value of each entry of u^* should be at most $O(n^{1.5i\epsilon})$. Otherwise the column already has $\omega(n/2^{k/2} \cdot n^{1.5i\epsilon}) = \omega(n^{0.6+2i\epsilon})$ cost. Thus, the magnitude of each entry of $\alpha_1 A^*_{j_1} + \alpha_2 A^*_{j_2} + \cdots + \alpha_{k/2} A^*_{k/2}$ is $\Theta(n^{1.5i\epsilon})$. Thus, there exists $t \in [k/2]$, such that the absolute value of each entry of $\alpha_t A^*_{j_t}$ is at least $\Omega(n^{1.5i\epsilon}/k)$. Then there are two cases.

The first case is $\mathrm{group}(t) < \mathrm{group}(j)$. Let $\mathrm{group}(t) = i'$. Then $|\alpha_t| = \Omega(n^{1.5(i-i')\epsilon}/k)$. By Claim 20.6.2 again, with probability at least $1 - (k/2) \cdot 2^{-\Theta(n/2^{k/2})}$, the size of

$$\{z \in [n - n^{0.1}] \mid \mathrm{sign}(u^*_z) = \mathrm{sign}(\alpha_1 \Delta_{j_1,z}) = \cdots = \mathrm{sign}(\alpha_{k/2} \Delta_{j_{k/2},z}) = \mathrm{sign}(-\Delta_{j,z})\}$$

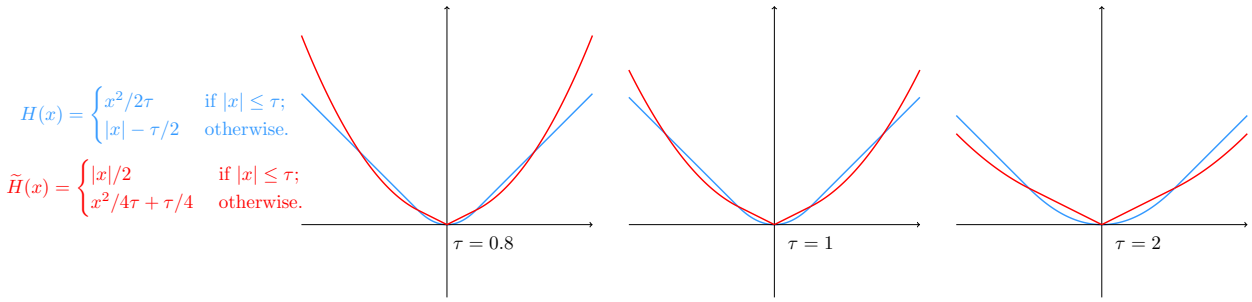


Figure 20.1: The blue curve is the Huber function which combines an ℓ_2-like measure for small x with an ℓ_1-like measure for large x. The red curve is the "reverse" Huber function which combines an ℓ_1-like measure for small x with an ℓ_2-like measure for large x.

is at least $\Omega(n/2^{k/2})$. Thus, the cost to fit is at least $\Omega(n/2^{k/2}) \cdot (n^{-0.2+i'\epsilon} n^{1.5(i-i')\epsilon}/k)^2 = \omega(n^{0.6+2i\epsilon})$.

The second case is $\text{group}(t) > \text{group}(j)$. Let $\text{group}(t) = i'$. Then $|\alpha_t| = \Omega(n^{1.5(i-i')\epsilon}/k)$. By Claim 20.6.2 again, with probability at least $1 - (k/2) \cdot 2^{-\Theta(n^{0.1}/2^{k/2})}$, the size of

$$\{z \in \{n - n^{0.1} + 1, \cdots, n\} \mid \text{sign}(u_z^*) = \text{sign}(\alpha_1 \Delta_{j_1,z}) = \cdots = \text{sign}(\alpha_{k/2} \Delta_{j_{k/2},z}) = \text{sign}(-\Delta_{j,z})\}$$

is at least $\Omega(n^{0.1}/2^{k/2})$. Thus, the fitting cost is at least $\Omega(n^{0.1}/2^{k/2}) \cdot (n^{0.5+2i'\epsilon} n^{1.5(i-i')\epsilon}/k) = \omega(n^{0.6+2i\epsilon})$.

By taking a union bound over all columns j, we have with probability at least $1 - 2^{n^{\Theta(1)}}$, the total cost to fit by a column subset S is at least $\omega(kn^{1.6})$.

Then, by taking a union bound over all the $\binom{n}{k}$ number of sets S, we complete the proof. □

20.6.2 Column Subset Selection for the Reverse Huber Function

In this section, we consider a "reverse Huber function": $g(x) = x^2$ if $x \geq 1$ and $g(x) = |x|$ for $x \leq 1$.

Theorem 20.6.3. *Let $g(x)$ denote the "reverse Huber function" with $\tau = 1$, i.e.,*

$$H(x) = \begin{cases} x^2/\tau, & \text{if } |x| > \tau; \\ |x|, & \text{if } |x| \leq \tau. \end{cases}$$

For any $n \geq 1$, there is a matrix $A \in \mathbb{R}^{n \times n}$ such that, if we select only 1 column to fit the entire matrix, there is no $n^{o(1)}$-approximation to the best rank-1 approximation, i.e., for any subset $S \subseteq [n]$ with $|S| = 1$,

$$\min_{X \in \mathbb{R}^{|S| \times n}} \|A_S X - A\|_g \geq n^{\Omega(1)} \cdot \min_{\text{rank}-1 \ A'} \|A' - A\|_g.$$

Proof. Let $A \in \mathbb{R}^{n \times n}$ have one column that is $a = (n^{1/2}, 0, \ldots, 0)^\top$ and $n-1$ columns that are each equal to $b = (0, 1/n, 1/n, \ldots, 1/n)^\top$. If we choose one column which has the form as a to fit the other columns, the cost is at least $(n-1)^2/n = \Theta(n)$. If we choose one column which has the form as b to fit the other columns, the cost is at least $(n^{1/2})^2 = \Theta(n)$.

Now we consider using a vector $c = (1/n^{1/4}, 1/n, 1/n, \ldots, 1/n)^\top$ to fit all the columns. One can use c to approximate a with cost at most $(n-1) \cdot n^{3/4}/n = \Theta(n^{3/4})$ by matching the first coordinate, while one can use c to approximate b with cost at most $1/n^{1/4}$ by matching the last $n-1$ coordinates, and since there are $n-1$ columns equal to b, the overall total cost of using c to approximate matrix A is $\Theta(n^{3/4})$.

□

Chapter 21

Tensor Low Rank Approximation

We consider relative error low rank approximation of *tensors* with respect to the Frobenius norm. Namely, given an order-q tensor $A \in \mathbb{R}^{\prod_{i=1}^{q} n_i}$, output a rank-$k$ tensor B for which $\|A - B\|_F^2 \leq (1+\epsilon)\,\mathrm{OPT}$, where $\mathrm{OPT} = \inf_{\text{rank-}k\ A'} \|A - A'\|_F^2$. Despite much success on obtaining relative error low rank approximations for matrices, no such results were known for tensors for arbitrary $(1+\epsilon)$-approximations. One structural issue is that there may be no rank-k tensor A_k achieving the above infinum. Another, computational issue, is that an efficient relative error low rank approximation algorithm for tensors would allow one to compute the rank of a tensor, which is NP-hard. We bypass these two issues via (1) bicriteria and (2) parameterized complexity solutions:

1. We give an algorithm which outputs a rank $k' = O((k/\epsilon)^{q-1})$ tensor B for which $\|A - B\|_F^2 \leq (1+\epsilon)\,\mathrm{OPT}$ in $\mathrm{nnz}(A) + n \cdot \mathrm{poly}(k/\epsilon)$ time in the real RAM model, whenever either A_k exists or $\mathrm{OPT} > 0$. Here $\mathrm{nnz}(A)$ denotes the number of non-zero entries in A. If both A_k does not exist and $\mathrm{OPT} = 0$, then B instead satisfies $\|A - B\|_F^2 < \gamma$, where γ is any positive, arbitrarily small function of n.

2. We give an algorithm for any $\delta > 0$ which outputs a rank k tensor B for which $\|A - B\|_F^2 \leq (1+\epsilon)\,\mathrm{OPT}$ and runs in $(\mathrm{nnz}(A) + n\,\mathrm{poly}(k/\epsilon) + \exp(k^2/\epsilon)) \cdot n^{\delta}$ time in the unit cost RAM model, whenever $\mathrm{OPT} > 2^{-O(n^{\delta})}$ and there is a rank-k tensor $B =$

$\sum_{i=1}^{k} u_i \otimes v_i \otimes w_i$ for which $\|A-B\|_F^2 \leq (1+\epsilon/2)\,\mathrm{OPT}$ and $\|u_i\|_2, \|v_i\|_2, \|w_i\|_2 \leq 2^{O(n^\delta)}$. If $\mathrm{OPT} \leq 2^{-\Omega(n^\delta)}$, then B instead satisfies $\|A-B\|_F^2 \leq 2^{-\Omega(n^\delta)}$.

Our first result is polynomial time, and in fact input sparsity time, in n, k, and $1/\epsilon$, for any $k \geq 1$ and any $0 < \epsilon < 1$, while our second result is fixed parameter tractable in k and $1/\epsilon$. For outputting a rank-k tensor, or even a bicriteria solution with rank-Ck for a certain constant $C > 1$, we show a $2^{\Omega(k^{1-o(1)})}$ time lower bound under the Exponential Time Hypothesis.

Our results are based on an "iterative existential argument", and also give the first relative error low rank approximations for tensors for a large number of error measures for which nothing was known. In particular, we give the first relative error approximation algorithms on tensors for: column row and tube subset selection, entrywise ℓ_p-low rank approximation for $1 \leq p < 2$, low rank approximation with respect to sum of Euclidean norms of faces or tubes, weighted low rank approximation, and low rank approximation in distributed and streaming models. We also obtain several new results for matrices, such as $\mathrm{nnz}(A)$-time CUR decompositions, improving the previous $\mathrm{nnz}(A)\log n$-time CUR decompositions, which may be of independent interest.

21.1 Introduction

Low rank approximation of matrices is one of the most well-studied problems in randomized numerical linear algebra. Given an $n \times d$ matrix A with real-valued entries, we want to output a rank-k matrix B for which $\|A - B\|$ is small, under a given norm. While this problem can be solved exactly using the singular value decomposition for some norms like the spectral and Frobenius norms, the time complexity is still $\min(nd^{\omega-1}, dn^{\omega-1})$, where $\omega \approx 2.373$ is the exponent of matrix multiplication [Str69, CW87, Wil12, LG14]. This time complexity is prohibitive when n and d are large. By now there are a number of approximation algorithms for this problem, with the Frobenius norm [1] being one of the most common error measures. Initial solutions [FKV04, AM07] to this problem were based on sampling and achieved additive error in terms of $\epsilon\|A\|_F$, where $\epsilon > 0$ is an approximation parameter, which can be arbitrarily larger than the optimal cost $\mathrm{OPT} = \min_{\text{rank-}k\ B} \|A - B\|_F^2$. Since then a number of solutions based on the technique of oblivious sketching [Sar06, CW13, MM13, NN13a] as well as sampling based on non-uniform distributions [DMM06c, DMM06b, DMM08, DMIMW12], have been proposed which achieve the stronger notion of *relative error*, namely, which output a rank-k matrix B for which $\|A - B\|_F^2 \leq (1+\epsilon)\,\mathrm{OPT}$ with high probability. It is now known how to output a factorization of such a $B = U \cdot V$, where U is $n \times k$ and V is $k \times d$, in $\mathrm{nnz}(A) + (n+d)\,\mathrm{poly}(k/\epsilon)$ time [CW13, MM13, NN13a]. Such an algorithm is optimal, up to the $\mathrm{poly}(k/\epsilon)$ factor, as any algorithm achieving relative error must read almost all of the entries.

Tensors are often more useful than matrices for capturing higher order relations

[1] Recall the Frobenius norm $\|A\|_F$ of a matrix A is $(\sum_{i=1}^{n}\sum_{j=1}^{d} A_{i,j}^2)^{1/2}$.

in data. Computing low rank factorizations of approximations of tensors is the primary task of interest in a number of applications, such as in psychology[Kro83], chemometrics [Paa00, SBG04], neuroscience [AAB+07, KB09, CLK+15], computational biology [CV15, SC15], natural language processing [CYYM14, LZBJ14, LZMB15, BNR+15], computer vision [VT02, WA03, SH05, HPS05, HD08, AFdLGTL09, PLY10, LFC+16, CLZ17], computer graphics [VT04, WWS+05, Vas09], security [AÇKY05, ACY06, KB06], cryptography [FS99, Sch12, KYFD15, SHW+16] data mining [KS08, RST10, KABO10, Mør11], machine learning applications such as learning hidden Markov models, reinforcement learning, community detection, multi-armed bandit, ranking models, neural network, Gaussian mixture models and Latent Dirichlet allocation [MR05, AFH+12, HK13, ALB13, ABSV14, AGH+14, AGHK14, BCV14, JO14a, GHK15, PBLJ15, JSA15, ALA16, AGMR16, ZSJ+17], programming languages [RTP16], signal processing [Wes94, DLDM98, Com09, CMDL+15], and other applications [YCS11, LMWY13, OS14, ZCZJ14, STLS14, YCS16, RNSS16].

Despite the success for matrices, the situation for order-q tensors for $q > 2$ is much less understood. There are a number of works based on alternating minimization [CC70, Har70, FMPS13, FT15, ZG01, BS15] gradient descent or Newton methods [ES09, ZG01], methods based on the Higher-order SVD (HOSVD) [LMV00a] which provably incur $\Omega(\sqrt{n})$-inapproximability for Frobenius norm error [LMV00b], the power method or orthogonal iteration method [LMV00b], additive error guarantees in terms of the flattened (unfolded) tensor rather than the original tensor [MMD08], tensor trains [Ose11], the tree Tucker decomposition [OT09], or methods specialized to orthogonal tensors [KM11, AGH+14, MHG15, WTSA15, WA16, SWZ16]. There are also a number of works on the problem of tensor completion, that is, recovering a low rank tensor from missing entries [WM01, AKDM10,

TSHK11, LMWY13, MHWG14, JO14b, BM16]. There is also another line of work using the sum of squares (SOS) technique to study tensor problems [BKS15a, GM15, HSS15, HSSS16, MSS16, PS17, SS17], other recent work on tensor PCA [All12b, All12a, RM14, JMZ15, ADGM16, ZX17], and work applying smoothed analysis to tensor decomposition [BCMV14]. Several previous works also consider more robust norms than the Frobenius norm for tensors, e.g., the R_1 norm (ℓ_1-ℓ_2-ℓ_2 norm in our work) [HD08], ℓ_1-PCA [PLY10], entry-wise ℓ_1 regularization [GGH14], M-estimator loss [YFS16], weighted approximation [Paa97, TK11, LRHG13], tensor-CUR [OST08, MMD08, CC10, FMMN11, FT15], or robust tensor PCA [GQ14, LFC+16, CLZ17].

Some of the above works, such as ones based on the tensor power method or alternating minimization, require incoherence or orthogonality assumptions. Others, such as those based on the simultaneous SVD, require an assumption on the minimum singular value. See the monograph of Moitra [Moi14a] for further discussion. Unlike the situation for matrices, there is no work for tensors that is able to achieve the following natural relative error guarantee: given a q-th order tensor $A \in \mathbb{R}^{n^{\otimes q}}$ and an arbitrary accuracy parameter $\epsilon > 0$, output a rank-k tensor B for which

$$\|A - B\|_F^2 \leq (1+\epsilon)\,\mathrm{OPT}, \tag{21.1}$$

where $\mathrm{OPT} = \inf_{\text{rank-}k\ B'} \|A - B'\|_F^2$, and where recall the rank of a tensor B is the minimal integer k for which B can be expressed as $\sum_{i=1}^k U_1^i \otimes U_2^i \otimes \cdots \otimes U_q^i$. General speaking, a q-th order tensor can have rank at most n^{q-1}. A third order tensor, for example, has rank which is an integer in $\{0, 1, 2, \ldots, n^2\}$. We note that [BCV14] is able to achieve a relative error 5-approximation for third order tensors, and an $O(q)$-approximation for q-th order tensors,

though it cannot achieve a $(1+\epsilon)$-approximation. We compare our work to [BCV14] in Section 21.1.4 below.

For notational simplicity, we will start by assuming third order tensors with all dimensions of equal size, but we extend all of our main theorems below to tensors of any constant order $q > 3$ and dimensions of different sizes.

The first caveat regarding (21.1) for tensors is that an optimal rank-k solution may not even exist! This is a well-known problem for tensors (see, e.g., [KHL89, Paa00, KDS08, Ste06, Ste08] and more details in section 4 of [DSL08]), for which for any rank-k tensor B, there always exists another rank-k tensor B' for which $\|A-B'\|_F^2 < \|A-B\|_F^2$. If $\mathrm{OPT} = 0$, then in this case for any rank-k tensor B, necessarily $\|A-B\|_F^2 > 0$, and so (21.1) cannot be satisfied. This fact was known to algebraic geometers as early as the 19th century, which they refer to as the fact that the locus of r-th secant planes to a Segre variety may not define a (closed) algebraic variety [DSL08, Lan12]. It is also known as the phenomenon underlying the concept of *border rank*[2][Bin80, Bin86, BCS97, Knu98, Lan06]. In this case it is natural to allow the algorithm to output an arbitrarily small $\gamma > 0$ amount of additive error. Note that unlike several additive error algorithms for matrices, the additive error here can in fact be an arbitrarily small positive function of n. If, however, $\mathrm{OPT} > 0$, then for any $\epsilon > 0$, there exists a rank-k tensor B for which $\|A-B\|_F^2 \leq (1+\epsilon)\,\mathrm{OPT}$, and in this case we should still require the algorithm to output a relative-error solution. If an optimal rank-k solution B exists, then as for matrices, it is natural to require the algorithm to output a relative-error solution.

[2] https://en.wikipedia.org/wiki/Tensor_rank_decomposition#Border_rank

Besides the above definitional issue, a central reason that (21.1) has not been achieved is that computing the rank of a third order tensor is well-known to be NP-hard [Hås90, HL13]. Thus, if one had such a polynomial time procedure for solving the problem above, one could determine the rank of A by running the procedure on each $k \in \{0, 1, 2, \ldots, n^2\}$, and check for the first value of k for which $\|A - B\|_F^2 = 0$, thus determining the rank of A. However, it is unclear if approximating the tensor rank is hard. This question will also be answered in this work.

The main question which we address is how to define a meaningful notion of (21.1) for the case of tensors and whether it is possible to obtain provably efficient algorithms which achieve this guarantee, without any assumptions on the tensor itself. Besides (21.1), there are many other notions of relative error for low rank approximation of matrices for which provable guarantees for tensors are unknown, such as tensor CURT, R_1 norm, and the weighted and ℓ_1 norms mentioned above. Our goal is to provide a general technique to obtain algorithms for many of these variants as well.

21.1.1 Our Results

To state our results, we first consider the case when a rank-k solution A_k exists, that is, there exists a rank-k tensor A_k for which $\|A - A_k\|_F^2 = \text{OPT}$.

We first give a $\text{poly}(n, k, 1/\epsilon)$-time $(1+\epsilon)$-relative error approximation algorithm for any $0 < \epsilon < 1$ and any $k \geq 1$, but allow the output tensor B to be of rank $O((k/\epsilon)^2)$ (for general q-order tensors, the output rank is $O((k/\epsilon)^{q-1})$, whereas we measure the cost of B with respect to rank-k tensors. Formally, $\|A - B\|_F^2 \leq (1+\epsilon)\|A - A_k\|_F^2$. In fact, our algorithm can be implemented in $\text{nnz}(A) + n \cdot \text{poly}(k/\epsilon)$ time in the real-RAM model, where $\text{nnz}(A)$ is the number of non-zero entries of A. Since it is necessary to use $\text{nnz}(A)$ time to avoid missing any (large) entry, such an algorithm is optimal (for running time) for any relative error algorithm, even bicriteria ones.

If A_k does not exist, then our output B instead satisfies $\|A - B\|_F^2 \leq (1+\epsilon)\,\text{OPT} + \gamma$, where γ is an arbitrarily small additive error. Since γ is arbitrarily small, $(1+\epsilon)\,\text{OPT} + \gamma$ is still a relative error whenever $\text{OPT} > 0$. Our theorem is as follows.

Theorem 21.1.1 (A Version of Theorem 21.4.9, bicriteria)**.** *Given a 3rd order tensor $A \in \mathbb{R}^{n \times n \times n}$, if A_k exists then there is a randomized algorithm running in $\text{nnz}(A) + n \cdot \text{poly}(k/\epsilon)$ time which outputs a (factorization of a) rank-$O(k^2/\epsilon^2)$ tensor B for which $\|A - B\|_F^2 \leq (1+\epsilon)\|A - A_k\|_F^2$. If A_k does not exist, then the algorithm outputs a rank-$O(k^2/\epsilon^2)$ tensor B for which $\|A - B\|_F^2 \leq (1+\epsilon)\,\text{OPT} + \gamma$, where $\gamma > 0$ is an arbitrarily small positive function of n. In both cases, the success probability is at least $2/3$.*

One of the main applications of matrix low rank approximation is parameter reduction, as one can store the matrix using fewer parameters in factored form or more quickly

multiply by the matrix if given in factored form, as well as remove directions that correspond to noise. In such applications, it is not essential that the low rank approximation have rank exactly k, since one still has a significant parameter reduction with a matrix of slightly larger rank. This same motivation applies to tensor low rank approximation; we obtain both space and time savings by representing a tensor in factored form, and in such applications bicriteria applications suffice. Moreover, the extremely efficient $\text{nnz}(A) + n \cdot \text{poly}(k/\epsilon)$ time algorithm we obtain may outweigh the need for outputting a tensor of rank exactly k. Bicriteria algorithms are common for coping with hardness; see e.g., results on robust low rank approximation of matrices [DV07, FFSS07, CW15a], sparse recovery [CKPS16], clustering [MMSW15, HT16], and approximation algorithms more generally.

We note that there are other applications, such as unique tensor decomposition in the method of moments, see, e.g., [BCV14], where one may have a hard rank constraint of k for the output. However, in such applications the so-called Tucker decomposition is still a useful dimensionality-reduction analogue of the SVD and our techniques for proving

We next consider the case when the rank parameter k is small, and we try to obtain rank-k solutions which are efficient for small values of k. As before, we first suppose that A_k exists.

If $A_k = \sum_{i=1}^{k} u_i \otimes v_i \otimes w_i$ and the norms $\|u_i\|_2$, $\|v_i\|_2$, and $\|w_i\|_2$ are bounded by $2^{\text{poly}(n)}$, we can return a rank-k solution B for which $\|A - B\|_F^2 \leq (1+\epsilon)\|A - A_k\|_F^2 + 2^{-\text{poly}(n)}$, in $f(k, 1/\epsilon) \cdot \text{poly}(n)$ time in the standard unit cost RAM model with words of size $O(\log n)$ bits. Thus, our algorithm is *fixed parameter tractable* in k and $1/\epsilon$, and in fact remains polynomial time for any values of k and $1/\epsilon$ for which $k^2/\epsilon = O(\log n)$. This is motivated by a number of low rank approximation applications in which k is typically small. The additive error of

$2^{-\operatorname{poly}(n)}$ is only needed in order to write down our solution B in the unit cost RAM model, since in general the entries of B may be irrational, even if the entries of A are specified by $\operatorname{poly}(n)$ bits. If instead we only want to output an approximation to the value $\|A - A_k\|_F^2$, then we can output a number Z for which $\operatorname{OPT} \leq Z \leq (1+\epsilon)\operatorname{OPT}$, that is, we do not incur additive error.

When A_k does not exist, there still exists a rank-k tensor $\widetilde{A}$ for which $\|A - \widetilde{A}\|_F^2 \leq \operatorname{OPT} + \gamma$, for arbitrary $\gamma > 0$. We require there exists such a $\widetilde{A}$ for which if $\widetilde{A} = \sum_{i=1}^{k} u_i \otimes v_i \otimes w_i$, then the norms $\|u_i\|_2$, $\|v_i\|_2$, and $\|w_i\|_2$ are bounded by $2^{\operatorname{poly}(n)}$.

The assumption in the previous two paragraphs that the factors of A_k and of $\widetilde{A}$ have norm bounded by $2^{\operatorname{poly}(n)}$ is necessary in certain cases, e.g., if $\operatorname{OPT} = 0$ and we are to write down the factors in $\operatorname{poly}(n)$ time. An abridged version of our theorem is as follows.

Theorem 21.1.2 (Combination of Theorem 21.4.1 and 21.4.2, rank-k)**.** *Given a 3rd order tensor $A \in \mathbb{R}^{n \times n \times n}$, for any $\delta > 0$, if $A_k = \sum_{i=1}^{k} u_i \otimes v_i \otimes w_i$ exists and each of $\|u_i\|_2, \|v_i\|_2$, and $\|w_i\|_2$ is bounded by $2^{O(n^\delta)}$, then there is a randomized algorithm running in $O(\operatorname{nnz}(A) + n\operatorname{poly}(k, 1/\epsilon) + 2^{O(k^2/\epsilon)}) \cdot n^\delta$ time in the unit cost* RAM *model with words of size $O(\log n)$ bits[3], which outputs a (factorization of a) rank-k tensor B for which $\|A - B\|_F^2 \leq (1+\epsilon)\|A - A_k\|_F^2 + 2^{-O(n^\delta)}$. Further, we can output a number Z for which $\operatorname{OPT} \leq Z \leq (1+\epsilon)\operatorname{OPT}$ in the same amount of time. When A_k does not exist, if there exists a rank-k tensor $\widetilde{A}$ for which $\|A - \widetilde{A}\|_F^2 \leq \operatorname{OPT} + 2^{-O(n^\delta)}$ and $\widetilde{A} = \sum_{i=1}^{k} u_i \otimes v_i \otimes w_i$ is such that the norms $\|u_i\|_2$, $\|v_i\|_2$, and $\|w_i\|_2$ are bounded by $2^{O(n^\delta)}$, then we can output a (factorization of a) rank-k tensor $\widetilde{A}$ for which $\|A - \widetilde{A}\|_F^2 \leq (1+\epsilon)\operatorname{OPT} + 2^{-O(n^\delta)}$.*

[3]The entries of A are assumed to fit in n^δ words.

Our techniques for proving Theorem 21.1.1 and Theorem 21.1.2 open up avenues for many other problems in linear algebra on tensors. We now define the problems and state our results for them.

There is a long line of research on matrix column subset selection and CUR decomposition [DMM08, BMD09, DR10, BDM11, FEGK13, BW14, WS15, ABF+16, SWZ17] under operator, Frobenius, and entry-wise ℓ_1 norm. It is natural to consider tensor column subset selection or tensor-CURT[4], however most previous works either give error bounds in terms of the tensor flattenings [DMM08], assume the original tensor has certain properties [OST08, FT15, TM17], consider the exact case which assumes the tensor has low rank [CC10], or only fit a high dimensional cross-shape to the tensor rather than to all of its entries [FMMN11]. Such works are not able to provide a $(1+\epsilon)$-approximation guarantee as in the matrix case without assumptions. We consider tensor column, row, and tube subset selection, with the goal being to find three matrices: a subset $C \in \mathbb{R}^{n\times c}$ of columns of A, a subset $R \in \mathbb{R}^{n\times r}$ of rows of A, and a subset $T \in \mathbb{R}^{n\times t}$ of tubes of A, such that there exists a tensor $U \in \mathbb{R}^{c\times r\times t}$ for which

$$\|U(C,R,T) - A\|_\xi \leq \alpha\|A_k - A\|_\xi + \gamma, \tag{21.2}$$

where $\gamma = 0$ if A_k exists and $\gamma = 2^{-\mathrm{poly}(n)}$ otherwise, $\alpha > 1$ is the approximation ratio, ξ is either Frobenius norm or Entry-wise ℓ_1 norm, and $U(C,R,T) = \sum_{i=1}^{c}\sum_{j=1}^{r}\sum_{l=1}^{t} U_{i,j,l}\cdot C_i \otimes R_j \otimes T_l$. In tensor CURT decomposition, we also want to output U.

We provide a (nearly) input sparsity time algorithm for this, together with an alternative input sparsity time algorithm which chooses slightly larger factors C, R, and T.

[4]T denotes the tube which is the column in 3rd dimension of tensor.

To do this, we combine Theorem 21.1.1 with the following theorem which, given a factorization of a rank-k tensor B, obtains C, U, R, and T in terms of it:

Theorem 21.1.3 (Combination of Theorem 21.4.39 and 21.4.40, $\|\|\|_F$-norm, CURT decomposition)**.** *Given a 3rd order tensor $A \in \mathbb{R}^{n\times n\times n}$, let $k \geq 1$, and let $U_B, V_B, W_B \in \mathbb{R}^{n\times k}$ be given. There is an algorithm running in $O(\mathrm{nnz}(A)\log n) + \widetilde{O}(n^2)\,\mathrm{poly}(k, 1/\epsilon)$ time (respectively, $O(\mathrm{nnz}(A)) + n\,\mathrm{poly}(k, 1/\epsilon)$ time) which outputs a subset $C \in \mathbb{R}^{n\times c}$ of columns of A, a subset $R \in \mathbb{R}^{n\times r}$ of rows of A, a subset $T \in \mathbb{R}^{n\times t}$ of tubes of A, together with a tensor $U \in \mathbb{R}^{c\times r\times t}$ with $\mathrm{rank}(U) = k$ such that $c = r = t = O(k/\epsilon)$ (respectively, $c = r = t = O(k\log k + k/\epsilon)$), and $\|U(C, R, T) - A\|_F^2 \leq (1+\epsilon)\|U_B \otimes V_B \otimes W_B - A\|_F^2$ holds with probability at least $9/10$.*

Combining Theorems 21.1.2 and 21.1.3 (with B being a $(1 + O(\epsilon))$-approximation to A) we achieve Equation (21.2) with $\alpha = (1 + \epsilon)$ and $\xi = F$ with the *optimal* number of columns, rows, tubes, and rank of U (we mention our matching lower bound later), though the running time has an $2^{O(k^2/\epsilon)}$ term in it. We note that instead combining Theorem 21.1.1 and Theorem 21.1.3 gives a bicriteria result for CURT without a $2^{O(k^2/\epsilon)}$ term in the running time, though it is suboptimal in the number of columns, rows, tubes, and rank of U.

As a side result worth stating, our analysis improves the best matrix CUR decomposition algorithm under Frobenius norm [BW14], providing the first optimal $\mathrm{nnz}(A)$-time algorithm:

Theorem 21.1.4 (Informal Version of Theorem 21.6.1, Matrix CUR decomposition)**.** *There is an algorithm, which given a matrix $A \in \mathbb{R}^{n\times d}$ and an integer $k \geq 1$, runs in $O(\mathrm{nnz}(A)) + (n + d)\,\mathrm{poly}(k, 1/\epsilon)$ time and outputs three matrices: $C \in \mathbb{R}^{n\times c}$ containing c columns of A,*

$R \in \mathbb{R}^{r \times d}$ *containing* r *rows of* A*, and* $U \in \mathbb{R}^{c \times r}$ *with* $\operatorname{rank}(U) = k$ *for which* $r = c = O(k/\epsilon)$ *and* $\|CUR - A\|_F^2 \leq (1+\epsilon) \min_{\operatorname{rank}-k \ A_k} \|A_k - A\|_F^2$*, holds with probability at least* $9/10$*.*

21.1.2 Our Techniques

Many of our proofs, in particular those for Theorem 21.1.1 and Theorem 21.1.2, are based on what we call an "iterative existential proof", which we then turn into an algorithm in two different ways depending if we are proving Theorem 21.1.1 or Theorem 21.1.2.

Henceforth, we assume A_k exists; otherwise replace A_k with a suitably good tensor $\widetilde{A}$ in what follows. Since $A_k = \sum_{i=1}^{k} U_i^* \otimes V_i^* \otimes W_i^*$[5], we can create three $n \times k$ matrices U^*, V^*, and W^* whose columns are the vectors U_i^*, V_i^*, and W_i^*, respectively. Now we consider the three different flattenings (or unfoldings) of A_k, which express A_k as an $n \times n^2$ matrix. Namely, by thinking of A_k as the sum of outer products, we can write the three flattenings of A_k as $U^* \cdot Z_1$, $V^* \cdot Z_2$, and $W^* \cdot Z_3$, where the rows of Z_1 are $\mathrm{vec}(V_i^* \otimes W_i^*)$ [6] (For simplicity, we write $Z_1 = (V^{*\top} \odot W^{*\top})$. [7]), the rows of Z_2 are $\mathrm{vec}(U_i^* \otimes W_i^*)$, and the rows of Z_3 are $\mathrm{vec}(U_i^* \otimes V_i^*)$, for $i \in [k] \stackrel{\text{def}}{=} \{1, 2, \ldots, k\}$. Letting the three corresponding flattenings of the input tensor A be A_1, A_2, and A_3, by the symmetry of the Frobenius norm, we have $\|A - B\|_F^2 = \|A_1 - U^* Z_1\|_F^2 = \|A_2 - V^* Z_2\|_F^2 = \|A_3 - W^* Z_3\|_F^2$.

Let us consider the hypothetical regression problem $\min_U \|A_1 - U Z_1\|_F^2$. Note that we do not know Z_1, but we will not need to. Let $r = O(k/\epsilon)$, and suppose S_1 is an $n^2 \times r$ matrix of i.i.d. normal random variables with mean 0 and variance $1/r$, denoted $N(0, 1/r)$. Then by standard results for regression (see, e.g., [Woo14b] for a survey), if $\widehat{U}$ is the minimizer to

[5] For simplicity, we define $U \otimes V \otimes W = \sum_{i=1}^{k} U_i \otimes V_i \otimes W_i$, where U_i is the i-th column of U.

[6] $\mathrm{vec}(V_i^* \otimes W_i^*)$ denotes a row vector that has length $n_1 n_2$ where V_i^* has length n_1 and W_i^* has length n_2.

[7] $(V^{*\top} \odot W^{*\top})$ denotes a $k \times n_1 n_2$ matrix where the i-th row is $\mathrm{vec}(V_i^* \otimes W_i^*)$, where length n_1 vector V_i^* is the i-th column of $n_1 \times k$ matrix V^*, and length n_2 vector W_i^* is the i-th column of $n_2 \times k$ matrix W^*, $\forall i \in [k]$.

the smaller regression problem $\widehat{U} = \text{argmin}_U \|UZ_1S_1 - A_1S_1\|_F^2$, then

$$\|A_1 - \widehat{U}Z_1\|_F^2 \leq (1+\epsilon)\text{min}_U\|A_1 - UZ_1\|_F^2. \tag{21.3}$$

Moreover,$\widehat{U} = A_1S_1(Z_1S_1)^\dagger$. Although we do not know know Z_1, this implies $\widehat{U}$ is in the column span of A_1S_1, which we do know, since we can flatten A to compute A_1 and then compute A_1S_1. Thus, this hypothetical regression argument gives us an existential statement - there exists a good rank-k matrix $\widehat{U}$ in the column span of A_1S_1. We could similarly define $\widehat{V} = A_2S_2(Z_2S_2)^\dagger$ and $\widehat{W} = A_3S_3(Z_3S_3)^\dagger$ as solutions to the analogous regression problems for the other two flattenings of A, which are in the column spans of A_2S_2 and A_3S_3, respectively. Given A_1S_1, A_2S_2, and A_3S_3, which we know, we could hope there is a good rank-k tensor in the span of the rank-1 tensors

$$\{(A_1S_1)_a \otimes (A_2S_2)_b \otimes (A_3S_3)_c\}_{a,b,c\in[r]}. \tag{21.4}$$

However, an immediate issue arises. First, note that our hypothetical regression problem guarantees that $\|A_1 - \widehat{U}Z_1\|_F^2 \leq (1+\epsilon)\|A - A_k\|_F^2$, and therefore since the rows of Z_1 are of the special form $\text{vec}(V_i^* \otimes W_i^*)$, we can perform a "retensorization" to create a rank-k tensor $B = \sum_i \widehat{U}_i \otimes V_i^* \otimes W_i^*$ from the matrix $\widehat{U}Z_1$ for which $\|A - B\|_F^2 \leq (1+\epsilon)\|A - A_k\|_F^2$. While we do not know $\widehat{U}$, since it is in the column span of A_1S_1, it implies that B is in the span of the rank-1 tensors $\{(A_1S_1)_a \otimes V_b^* \otimes W_c^*\}_{a\in[r],b,c\in[k]}$. Analogously, we have that there is a good rank-k tensor B in the span of the rank-1 tensors $\{U_a^* \otimes (A_2S_2)_b \otimes W_c^*\}_{a,c\in[k],b\in[r]}$, and a good rank-$k$ tensor B in the span of the rank-1 tensors $\{U_a^* \otimes V_b^* \otimes (A_3S_3)_c\}_{a,b\in[k],c\in[r]}$. However, we do not know U^* or V^*, and it is not clear there is a rank-k tensor B for which *simultaneously* its first factors are in the column span of A_1S_1, its second factors are in the

column span of A_2S_2, and its third factors are in the column span of A_3S_3, i.e., whether there is a good rank-k tensor B in the span of rank-1 tensors in (21.4).

We fix this by an iterative argument. Namely, we first compute A_1S_1, and write $\widehat{U} = A_1S_1(Z_1S_1)^\dagger$. We now redefine Z_2 *with respect to* $\widehat{U}$, so the rows of Z_2 are $\mathrm{vec}(\widehat{U}_i \otimes W_i^*)$ for $i \in [k]$, and consider the regression problem $\min_V \|A_2 - VZ_2\|_F^2$. While we do not know Z_2, if S_2 is an $n^2 \times r$ matrix of i.i.d. Gaussians, we again have the statement that $\widehat{V} = A_2S_2(Z_2S_2)^\dagger$ satisfies

$$\begin{aligned}
\|A_2 - \widehat{V}Z_2\|_F^2 &\leq (1+\epsilon)\min_V \|A_2 - VZ_2\|_F^2 \text{ by the regression guarantee with Gaussians} \\
&\leq (1+\epsilon)\|A_2 - V^*Z_2\|_F^2 \text{ since } V^* \text{ is no better than the minimizer } V \\
&= (1+\epsilon)\|A_1 - \widehat{U}Z_1\|_F^2 \text{ by retensorizing and flattening along a different dimension} \\
&\leq (1+\epsilon)^2\min_U \|A_1 - UZ_1\|_F^2 \text{ by (21.3)} \\
&= (1+\epsilon)^2\|A - A_k\|_F^2 \text{ by definition of } Z_1 \ .
\end{aligned}$$

Now we can retensorize $\widehat{V}Z_2$ to obtain a rank-k tensor B for which $\|A - B\|_F^2 = \|A_2 - \widehat{V}Z_2\|_F^2 \leq (1+\epsilon)^2\|A - A_k\|_F^2$. Note that since the columns of $\widehat{V}$ are in the span of A_2S_2, and the rows of Z_2 are $\mathrm{vec}(\widehat{U}_i \otimes W_i^*)$ for $i \in [k]$, where the columns of $\widehat{U}$ are in the span of A_1S_1, it follows that B is in the span of rank-1 tensors $\{(A_1S_1)_a \otimes (A_2S_2)_b \otimes \widehat{V}_c\}_{a,b\in[r],c\in[k]}$.

Suppose we now redefine Z_3 so that it is now an $r^2 \times n^2$ matrix with rows $\mathrm{vec}((A_1S_1)_a \otimes (A_2S_2)_b)$ for all pairs $a, b \in [r]$, and consider the regression problem $\min_W \|A_3 - WZ_3\|_F^2$. Now observe that since *we know* Z_3, and since we can form A_3 by flattening A, we can solve for $W \in \mathbb{R}^{n \times r^2}$ in polynomial time by solving a regression problem. Retensorizing WZ_3 to a tensor B, it follows that we have found a rank-$r^2 = O(k^2/\epsilon^2)$ tensor B for which

$\|A - B\|_F^2 \leq (1+\epsilon)^2 \|A - A_k\|_F^2 = (1 + O(\epsilon))\|A - A_k\|_F^2$, and the result follows by adjusting ϵ by a constant factor.

To obtain the $\text{nnz}(A) + n\,\text{poly}(k/\epsilon)$ running time guarantee of Theorem 21.1.1, while we can replace S_1 and S_2 with compositions of a sparse CountSketch matrix and a Gaussian matrix (see chapter 2 of [Woo14b] for a survey), enabling us to compute $A_1 S_1$ and $A_2 S_2$ in $\text{nnz}(A) + n\,\text{poly}(k/\epsilon)$ time, we still need to solve the regression problem $\min_W \|A_3 - W Z_3\|_F^2$ quickly, and note that we cannot even write down Z_3 without spending $r^2 n^2$ time. Here we use a different random matrix S_3 called TensorSketch, which was introduced in [Pag13, PP13], but for which we will need the stronger properties of a subspace embedding and approximate matrix product shown to hold for it in [ANW14]. Given the latter properties, we can instead solve the regression problem $\min_W \|A_3 S_3 - W Z_3 S_3\|_F^2$, and importantly $A_3 S_3$ and $Z_3 S_3$ can be computed in $\text{nnz}(A) + n\,\text{poly}(k/\epsilon)$ time. Finally, this small problem can be solved in $n\,\text{poly}(k/\epsilon)$ time.

If we want to output a rank-k solution as in Theorem 21.1.2, then we need to introduce indeterminates at several places in the preceding argument and run a generic polynomial optimization procedure which runs in time exponential in the number of indeterminates. Namely, we write $\widehat{U}$ as $A_1 S_1 X_1$, where X_1 is an $r \times k$ matrix of indeterminates, we write $\widehat{V}$ as $A_2 S_2 X_2$, where X_2 is an $r \times k$ matrix of indeterminates, and we write $\widehat{W}$ as $A_3 S_3 X_3$, where X_3 is an $r \times k$ matrix of indeterminates. When executing the above iterative argument, we let the rows of Z_1 be the vectors $\text{vec}(V_i^* \otimes W_i^*)$, the rows of Z_2 be the vectors $\text{vec}(\widehat{U}_i \otimes W_i^*)$, and the rows of Z_3 be the vectors $\text{vec}(\widehat{U}_i \otimes V_i)$. Then $\widehat{U}$ is a $(1+\epsilon)$-approximate minimizer to $\min_U \|A_1 - U Z_1\|_F$, while $\widehat{V}$ is a $(1+\epsilon)$-approximate minimizer to $\min_V \|A_2 - V Z_2\|_F$, while $\widehat{W}$ is a $(1+\epsilon)$-approximate minimizer to $\min_W \|A_3 - W Z_3\|_F$. Note that by assigning

$X_1 = (Z_1S_1)^\dagger$, $X_2 = (Z_2S_2)^\dagger$, and $X_3 = (Z_3S_3)^\dagger$, it follows that the rank-k tensor $B = \sum_{i=1}^k (A_1S_1X_1)_i \otimes (A_2S_2X_2)_i \otimes (A_3S_3X_3)_i$ satisfies $\|A-B\|_F^2 \leq (1+\epsilon)^3\|A-A_k\|_F^2$, as desired. Note that here the rows of Z_2 are a function of X_1, while the rows of Z_3 are a function of both X_1 and X_2. What is important for us though is that it suffices to minimize the degree-6 polynomial $\sum_{a,b,c\in[n]}(\sum_{i=1}^k (A_1S_1X_1)_{a,i} \cdot (A_2S_2X_2)_{b,i} \cdot (A_3S_3X_3)_{c,i} - A_{a,b,c})^2$, over the $3rk = O(k^2/\epsilon)$ indeterminates X_1, X_2, X_3, since we know there exists an assignment to X_1, X_2, and X_3 providing a $(1 + O(\epsilon))$-approximate solution, and any solution X_1, X_2, and X_3 found by minimizing the above polynomial will be no worse than that solution. This polynomial can be minimized up to additive $2^{-\operatorname{poly}(n)}$ additive error in $\operatorname{poly}(n)$ time [Ren92a, BPR96] assuming the entries of U^*, V^*, and W^* are bounded by $2^{\operatorname{poly}(n)}$, as assumed in Theorem 21.1.2. Similar arguments can be made for obtaining a relative error approximation to the OPT as well as handling the case when A_k does not exist.

To optimize the running time to $\operatorname{nnz}(A)$, we can choose CountSketch matrices T_1, T_2, T_3 of $t = \operatorname{poly}(k, 1/\epsilon) \times n$ dimensions and reapply the above iterative argument. Then it suffices to minimize this small size degree-6 polynomial $\sum_{a,b,c\in[t]}(\sum_{i=1}^k (T_1A_1S_1X_1)_{a,i} \cdot (T_2A_2S_2X_2)_{b,i} \cdot (T_3A_3S_3X_3)_{c,i} - (A(T_1, T_2, T_3))_{a,b,c})^2$, over the $3rk = O(k^2/\epsilon)$ indeterminates X_1, X_2, X_3. Outputting $A_1S_1X_1$, $A_2S_2X_2$, $A_3S_3X_3$ then provides a $(1 + \epsilon)$-approximate solution.

Our iterative existential argument provides a general framework for obtaining low rank approximation results for tensors for many other error measures as well.

21.1.3 Other Low Rank Approximation Algorithms Following Our Framework.

Column, row, tube subset selection, and CURT decomposition. In tensor column, row, tube subset selection, the goal is to find three matrices: a subset C of columns of A, a subset R of rows of A, and a subset T of tubes of A, such that there exists a small tensor U for which $\|U(C,R,T)-A\|_F^2 \leq (1+\epsilon)\,\mathrm{OPT}$. We first choose two Gaussian matrices S_1 and S_2 with $s_1 = s_2 = O(k/\epsilon)$ columns, and form a matrix $Z_3' \in \mathbb{R}^{(s_1 s_2)\times n^2}$ with (i,j)-th row equal to the vectorization of $(A_1S_1)_i \otimes (A_2S_2)_j$. Motivated by the regression problem $\min_W \|A_3 - WZ_3'\|_F$, we sample $d_3 = O(s_1 s_2/\epsilon)$ columns from A_3 and let D_3 denote this selection matrix. There are a few ways to do the sampling depending on the tradeoff between the number of columns and running time, which we describe below. Proceeding iteratively, we write down Z_2' by setting its (i,j)-th row to the vectorization of $(A_1S_1)_i \otimes (A_3D_3)_j$. We then sample $d_2 = O(s_1 d_3/\epsilon)$ columns from A_2 and let D_2 denote that selection matrix. Finally, we define Z_1' by setting its (i,j)-th row to be the vectorization of $(A_2D_2)_i \otimes (A_3D_3)_j$. We obtain $C = A_1D_1$, $R = A_2D_2$ and $T = A_3D_3$. For the sampling steps, we can use a generalized matrix column subset selection technique, which extends a column subset selection technique of [BW14] in the context of CUR decompositions to the case when C is not necessarily a subset of the input. This gives $O(\mathrm{nnz}(A)\log n) + \widetilde{O}(n^2)\,\mathrm{poly}(k,1/\epsilon)$ time. Alternatively, we can use a technique we develop called tensor leverage score sampling described below, yielding $O(\mathrm{nnz}(A)) + n\,\mathrm{poly}(k,1/\epsilon)$ time.

A body of work in the matrix case has focused on finding the best possible number of columns and rows of a CUR decomposition, and we can ask the same question for tensors. It turns out that if one is given the factorization $\sum_{i=1}^k (U_B)_i \otimes (V_B)_i \otimes (W_B)_i$ of a rank-k tensor $B \in \mathbb{R}^{n\times n\times n}$ with $U_B, V_B, W_B \in \mathbb{R}^{n\times k}$, then one can find a set C of $O(k/\epsilon)$ columns,

a set R of $O(k/\epsilon)$ rows, and a set T of $O(k/\epsilon)$ tubes of A, together with a rank-k tensor U for which $\|U(C,R,T)-A\|_F^2 \leq (1+\epsilon)\|A-B\|_F^2$. This is based on an iterative argument, where the initial sampling (which needs to be our generalized matrix column subset selection rather than tensor leverage score sampling to achieve optimal bounds) is done with respect to $V_B^\top \odot W_B^\top$, and then an iterative argument is carried out. Since we show a matching lower bound on the number of columns, rows, tubes and rank of U, these parameters are tight. The algorithm is efficient if one is given a rank-k tensor B which is a $(1+O(\epsilon))$-approximation to A; if not then one can use Theorem 21.1.2 and and this step will be exponential time in k. If one just wants $O(k\log k + k/\epsilon)$ columns, rows, and tubes, then one can achieve $O(\text{nnz}(A)) + n\,\text{poly}(k,1/\epsilon)$ time, if one is given B.

Column-row, row-tube, tube-column face subset selection, and CURT decomposition. In tensor column-row, row-tube, tube-column face subset selection, the goal is to find three tensors: a subset $C \in \mathbb{R}^{c\times n\times n}$ of row-tube faces of A, a subset $R \in \mathbb{R}^{n\times r\times n}$ of tube-column faces of A, and a subset $T \in \mathbb{R}^{n\times n\times t}$ of column-row faces of A, such that there exists a tensor $U \in \mathbb{R}^{tn\times cn\times rn}$ with small rank for which $\|U(T_1,C_2,R_3)-A\|_F^2 \leq (1+\epsilon)\,\text{OPT}$, where $T_1 \in \mathbb{R}^{n\times tn}$ denotes the matrix obtained by flattening the tensor T along the first dimension, $C_2 \in \mathbb{R}^{n\times cn}$ denotes the matrix obtained by flattening the tensor C along the second dimension, and $R_3 \in \mathbb{R}^{n\times rn}$ denotes the matrix obtained by flattening the tensor T along the third dimension.

We solve this problem by first choosing two Gaussian matrices S_1 and S_2 with $s_1 = s_2 = O(k/\epsilon)$ columns, and then forming matrix $U_3 \in \mathbb{R}^{n\times s_1 s_2}$ with (i,j)-th column equal to $(A_1S_1)_i$, as well as matrix $V_3 \in \mathbb{R}^{n\times s_1 s_2}$ with (i,j)-th column equal to $(A_2S_2)_j$. Inspired by

the regression problem $\min_{W \in \mathbb{R}^{n \times s_1 s_2}} \|V_3 \cdot (W^\top \odot U_3^\top) - A_2\|_F$, we sample $d_3 = O(s_1 s_2/\epsilon)$ rows from A_2 and let $D_3 \in \mathbb{R}^{n \times n}$ denote this selection matrix. In other words, D_3 selects d_3 tube-column faces from the original tensor A. Thus, we obtain a small regression problem: $\min_W \|D_3 V_3 \cdot (W^\top \odot U_3^\top) - D_3 A_2\|_F$. By retensorizing the objective function, we obtain the problem $\min_W \|U_3 \otimes (D_3 V_3) \otimes W - A(I, D_3, I)\|_F$. Flattening the objective function along the third dimension, we obtain $\min_W \|W \cdot (U_3^\top \odot (D_3 V_3)^\top) - (A(I, D_3, I))_3\|_F$ which has optimal solution $(A(I, D_3, I))_3 (U_3^\top \odot (D_3 V_3)^\top)^\dagger$. Let W' denote $A(I, D_3, I))_3$. In the next step, we fix $W_2 = W'(U_3^\top \odot (D_3 V_3)^\top)^\dagger$ and $U_2 = U_3$, and consider the objective function $\min_V \|U_2 \cdot (V^\top \odot W_2^\top) - A_1\|_F$. Applying a similar argument, we obtain $V' = (A(D_2, I, I))_2$ and $U' = (A(I, I, D_1)_1)$. Let C denote $A(D_2, I, I)$, R denote $A(I, D_3, I)$, and T denote $A(I, I, D_1)$. Overall, this algorithm selects $\mathrm{poly}(k, 1/\epsilon)$ faces from each dimension.

Similar to our column-based CURT decomposition, our face-based CURT decomposition has the property that if one is given the factorization $\sum_{i=1}^k (U_B)_i \otimes (V_B)_i \otimes (W_B)_i$ of a rank-k tensor $B \in \mathbb{R}^{n \times n \times n}$ with $U_B, V_B, W_B \in \mathbb{R}^{n \times k}$ which is a $(1 + O(\epsilon))$-approximation to A, then one can find a set C of $O(k/\epsilon)$ row-tube faces, a set R of $O(k/\epsilon)$ tube-column faces, and a set T of $O(k/\epsilon)$ column-row faces of A, together with a rank-k tensor U for which $\|U(T_1, C_2, R_3) - A\|_F^2 \leq (1 + \epsilon)\,\mathrm{OPT}$.

Tensor multiple regression and tensor leverage score sampling. In the above we need to consider standard problems for matrices in the context of tensors. Suppose we are given a matrix $A \in \mathbb{R}^{n_1 \times n_2 n_3}$ and a matrix $B = (V^\top \odot W^\top) \in \mathbb{R}^{k \times n_2 n_3}$ with rows $(V_i \otimes W_i)$ for an $n_2 \times k$ matrix V and $n_3 \times k$ matrix W. Using TensorSketch [Pag13, PP13, ANW14] one can solve multiple regression $\min_U \|UB - A\|_F$ without forming B in $O(n_2 + n_3)\,\mathrm{poly}(k, 1/\epsilon)$

time, rather than the naïve $O(n_2 n_3)\,\mathrm{poly}(k, 1/\epsilon)$ time. However, this does not immediately help us if we would like to sample columns of such a matrix B proportional to its leverage scores. Even if we apply TensorSketch to compute a $k \times k$ change of basis matrix R in $O(n_2 + n_3)\,\mathrm{poly}(k, \log(n_2 n_3))$ time, for which the leverage scores of B are (up to a constant factor) the squared column norms of $R^{-1}B$, there are still $n_2 n_3$ leverage scores and we cannot write them all down! Nevertheless, we show we can still sample by them by using that the matrix of interest is formed via a tensor product, which can be rewritten as a matrix multiplication which we never need to explicily materialize. In more detail, for the i-th row $e_i R^{-1}$ of R^{-1}, we create a matrix V'^i by scaling each of the columns of $V^\top$ entrywise by the entries of z. The squared norms of $e_i R^{-1} B$ are exactly the squared entries of $(V'^i)W^\top$. We cannot compute this matrix product, but we can first sample a column of it proportional to its squared norm and then sample an entry in that column proportional to its square. To sample a column, we compute $G(V'^i)W^\top$ for a Gaussian matrix G with $O(\log n_3)$ rows by computing $G \cdot V'^i$, then computing $(G \cdot V'^i) \cdot W^\top$, which is $O(n_2 + n_3)\,\mathrm{poly}(k, \log(n_2 n_3))$ total time. After sampling a column, we compute the column exactly and sample a squared entry. We do this for each $i \in [k]$, first sampling an i proportional to $\|GV'^i W^\top\|_F^2$, then running the above scheme on that i. The $\mathrm{poly}(\log n)$ factor in the running time can be replaced by $\mathrm{poly}(k)$ if one wants to avoid a $\mathrm{poly}(\log n)$ dependence in the running time.

Entry-wise ℓ_1 low-rank approximation. We consider the problem of entrywise ℓ_1-low rank approximation of an $n \times n \times n$ tensor A, namely, the problem of finding a rank-k tensor B for which $\|A - B\|_1 \leq \mathrm{poly}(k, \log n)\,\mathrm{OPT}$, where $\mathrm{OPT} = \inf_{\text{rank-}k\ B} \|A - B\|_1$, and where for a tensor A, $\|A\|_1 = \sum_{i,j,k} |A_{i,j,k}|$. Our iterative existential argument can be applied in

much the same way as for the Frobenius norm. We iteratively flatten A along each of its three dimensions, obtaining A_1, A_2, and A_3 as above, and iteratively build a good rank-k solution B of the form $(A_1S_1X_1) \otimes (A_2S_2X_2) \otimes (A_3S_3X_3)$, where now the S_i are matrices of i.i.d. Cauchy random variables or sparse matrices of Cauchy random variables and the X_i are $O(k \log k) \times k$ matrices of indeterminates. For a matrix C and a matrix S of i.i.d. Cauchy random variables with k columns, it is known [SWZ17] that the column span of CS contains a $\text{poly}(k \log n)$-approximate rank-k space with respect to the entrywise ℓ_1-norm for C. In the case of tensors, we must perform an iterative flattening and retensorizing argument to guarantee there exists a tensor B of the form above. Also, if we insist on outputting a rank-k solution as opposed to a bicriteria solution, $\|(A_1S_1X_1) \otimes (A_2S_2X_2) \otimes (A_3S_3X_3) - A\|_1$ is not a polynomial of the X_i, and if we introduce sign variables for the n^3 absolute values, the running time of the polynomial solver will be $2^{\#\text{ of variables}} = 2^{\Omega(n^3)}$. We perform additional dimensionality reduction by Lewis weight sampling [CP15] from the flattenings to reduce the problem size to $\text{poly}(k)$. This small problem still has $\widetilde{O}(k^3)$ sign variables, and to obtain a $2^{\widetilde{O}(k^2)}$ running time we relax the reduced problem to a Frobenius norm problem, mildly increasing the approximation factor by another $\text{poly}(k)$ factor.

Combining the iterative existential argument with techniques in [SWZ17], we also obtain an ℓ_1 CURT decomposition algorithm (which is similar to the Frobenius norm result in Theorem 21.1.3), which can find $\widetilde{O}(k)$ columns, $\widetilde{O}(k)$ rows, $\widetilde{O}(k)$ tubes, and a tensor U. Our algorithm starts from a given factorization of a rank-k tensor $B = U_B \otimes V_B \otimes W_B$ found above. We compute a sampling and rescaling diagonal matrix D_1 according to the Lewis weights of matrix $B_1 = (V_B^\top \odot W_B^\top)$, where D_1 has $\widetilde{O}(k)$ nonzero entries. Then we iteratively construct B_2, D_2, B_3 and D_3. Finally we have $C = A_1D_1$ (selecting $\widetilde{O}(k)$ columns from

Algorithm 21.1 Main Meta-Algorithm

procedure TENSORLOWRANKAPPROXBICRITERIA(A, n, k, ϵ) ▷ Theorem 21.1.1
    Choose sketching matrices S_2,S_3(Composition of Gaussian and CountSketch.)
    Choose sketching matrices T_2,T_3(CountSketch.)
    Compute $T_2A_2S_2$, $T_3A_3S_3$.
    Construct $\widehat{V}$ by setting (i,j)-th column to be $(A_2S_2)_i$.
    Construct $\widehat{W}$ by setting (i,j)-th column to be $(A_3S_3)_j$.
    Construct matrix B by setting (i,j)-th row of B is vectorization of $(T_2A_2S_2)_i \otimes (T_3A_3S_3)_j$.
    Solve $\min_U \|UB - (A(I,T_2,T_3))_1\|_F^2$.
    return $\widehat{U}$, $\widehat{V}$, and $\widehat{W}$.
end procedure
procedure TENSORLOWRANKAPPROX(A, n, k, ϵ) ▷ Theorem 21.1.2
    Choose sketching matrices S_1,S_2,S_3(Composition of Gaussian and CountSketch.)
    Choose sketching matrices T_1,T_2,T_3(CountSketch.)
    Compute $T_1A_1S_1$, $T_2A_2S_2$, $T_3A_3S_3$.
    Solve $\min_{X_1,X_2,X_3} \|(T_1A_1S_1X_1) \otimes (T_2A_2S_2X_2) \otimes (T_3A_3S_3X_3) - A(T_1,T_2,T_3)\|_F^2$.
    return $A_1S_1X_1$, $A_2S_2X_2$, and $A_3S_3X_3$.
end procedure

A), $R = A_2D_2$ (selecting $\widetilde{O}(k)$ rows from A), $T = A_3D_3$ (selecting $\widetilde{O}(k)$ tubes from A) and tensor $U = ((B_1D_1)^\dagger) \otimes ((B_2D_2)^\dagger) \otimes ((B_3D_3)^\dagger)$.

Optimal matrix CUR decomposition. We also improve the $\mathrm{nnz}(A)\log n + (n+d)\,\mathrm{poly}(\log n, k, 1/\epsilon)$ running time of [BW14] for CUR decomposition of $A \in \mathbb{R}^{n\times d}$ to $\mathrm{nnz}(A) + (n+d)\,\mathrm{poly}(k, 1/\epsilon)$, while selecting the optimal number of columns, rows, and a rank-k matrix U. Using [CW13, MM13, NN13a], we find a matrix $\widehat{U}$ with k orthonormal columns in $\mathrm{nnz}(A) + n\,\mathrm{poly}(k/\epsilon)$ time for which $\min_V \|\widehat{U}V - A\|_F^2 \le (1+\epsilon)\|A - A_k\|_F^2$. Let $s_1 = \widetilde{O}(k/\epsilon^2)$ and $S_1 \in \mathbb{R}^{s_1\times n}$ be a sampling/rescaling matrix by the leverage scores of $\widehat{U}$. By strengthening the affine embedding analysis of [CW13] to leverage score sampling (the analysis of

[CW13] gives a weaker analysis for affine embeddings using leverage scores which does not allow approximation in the sketch space to translate to approximation in the original space), with probability at least 0.99, for all X' which satisfy $\|S_1\widehat{U}X' - S_1A\|_F^2 \leq (1+\epsilon')\min_X \|S_1\widehat{U}X - S_1A\|_F^2$, we have $\|\widehat{U}X' - A\|_F^2 \leq (1+\epsilon)\min_X \|\widehat{U}X - A\|_F^2$, where $\epsilon' = 0.0001\epsilon$. Applying our generalized row subset selection procedure, we can find Y, R for which $\|S_1\widehat{U}YR - S_1A\|_F^2 \leq (1+\epsilon')\min_X \|S_1\widehat{U}X - S_1A\|_F^2$, where R contains $O(k/\epsilon') = O(k/\epsilon)$ rescaled rows of S_1A. A key point is that rescaled rows of S_1A are also rescaled rows of A. Then, $\|\widehat{U}YR - A\|_F^2 \leq (1+\epsilon)\min_X \|\widehat{U}X - A\|_F^2$. Finding Y, R can be done in $d\,\mathrm{poly}(s_1/\epsilon) = d\,\mathrm{poly}(k/\epsilon)$ time. Now set $\widehat{V} = YR$. We can choose S_2 to be a sampling/rescaling matrix, and then find C, Z for which $\|CZ\widehat{V}S_2 - AS_2\|_F^2 \leq (1+\epsilon')\min_X \|X\widehat{V}S_2 - AS_2\|_F^2$ in a similar way, where C contains $O(k/\epsilon)$ rescaled columns of AS_2, and thus also of A. We thus have $\|CZYR - A\|_F^2 \leq (1+O(\epsilon))\|A - A_k\|_F^2$.

21.1.4 Comparison to [BCV14]

[BCV14] shows for a third order $n_1 \times n_2 \times n_3$ tensor A how to find a rank-k tensor B for which $\|A - B\|_F^2 \leq 5\,\mathrm{OPT}$ in $\mathrm{poly}(n_1 n_2 n_3)\exp(\mathrm{poly}(k))$ time. They generalize this to q-th order tensors to find a rank-k tensor B for which $\|A - B\|_F^2 = O(q)\,\mathrm{OPT}$ in $\mathrm{poly}(n_1 n_2 \cdots n_q)\exp(\mathrm{poly}(qk))$ time.

In contrast, we obtain a rank-k tensor B for which $\|A - B\|_F^2 \leq (1+\epsilon)\,\mathrm{OPT}$ in $\mathrm{nnz}(A) + n \cdot \mathrm{poly}(k/\epsilon) + \exp((k^2/\epsilon)\,\mathrm{poly}(q))$ time for every order q. Thus, we obtain a $(1+\epsilon)$ instead of an $O(q)$ approximation. The $O(q)$ approximation in [BCV14] seems inherent since the authors apply triangle inequality q times, each time losing a constant factor. This seems necessary since their argument is based on the span of the top k principal components in the SVD in each flattening separately containing a good space to project onto for a given mode. In contrast, our iterative existential argument chooses the space to project onto in successive modes *adaptively* as a function of spaces chosen for previous modes, and thus we obtain a $(1+\epsilon)^{O(q)} = (1 + O(\epsilon q))$-approximation, which becomes a $(1+\epsilon)$-approximation after replacing ϵ with ϵ/q. Also, importantly, our algorithm runs in $\mathrm{nnz}(A) + n \cdot \mathrm{poly}(k/\epsilon) + \exp((k^2/\epsilon)\,\mathrm{poly}(q))$ time and there are multiple hurdles we overcome to achieve this, as described in Section 21.1.2 above.

21.1.5 A Roadmap

Roadmap Section 21.2 introduces notation and definitions. Section 21.3 includes several tools. Our Frobenius norm low rank approximation algorithms are in Section 21.4. Section 21.5 extends our results to general q-th order tensors.

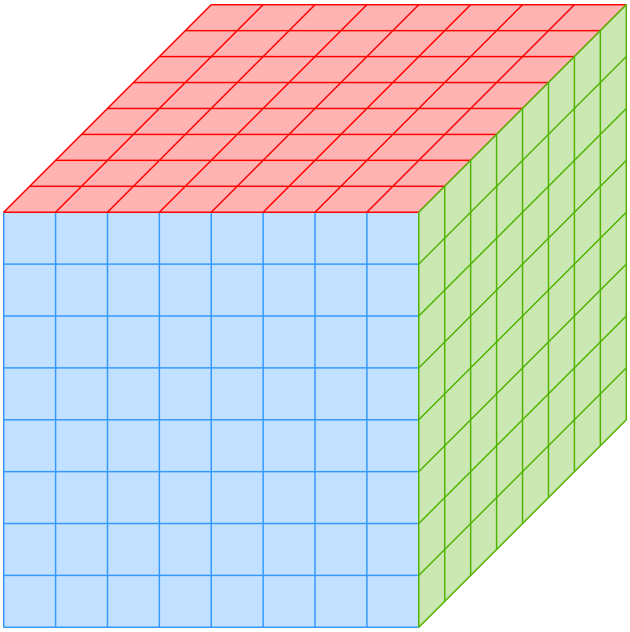


Figure 21.1: A 3rd order tensor with size $8 \times 8 \times 8$.

21.2 Notation

For an $n \in \mathbb{N}_+$, let $[n]$ denote the set $\{1, 2, \cdots, n\}$.

For any function f, we define $\widetilde{O}(f)$ to be $f \cdot \log^{O(1)}(f)$. In addition to $O(\cdot)$ notation, for two functions f, g, we use the shorthand $f \lesssim g$ (resp. $\gtrsim$) to indicate that $f \leq Cg$ (resp. $\geq$) for an absolute constant C. We use $f \eqsim g$ to mean $cf \leq g \leq Cf$ for constants c, C.

For a matrix A, we use $\|A\|_2$ to denote the spectral norm of A. For a tensor A, let $\|A\|$ and $\|A\|_2$ (which we sometimes use interchangeably) denote the spectral norm of tensor A,

$$\|A\| = \sup_{x,y,z \neq 0} \frac{|A(x, y, z)|}{\|x\| \cdot \|y\| \cdot \|z\|}.$$

Let $\|A\|_F$ denote the Frobenius norm of a matrix/tensor A, i.e., $\|A\|_F$ is the square root of

sum of squares of all the entries of A. For $1 \leq p < 2$, we use $\|A\|_p$ to denote the entry-wise ℓ_p-norm of a matrix/tensor A, i.e., $\|A\|_p$ is the p-th root of the sum of p-th powers of the absolute values of the entries of A. $\|A\|_1$ will be an important special case of $\|A\|_p$, which corresponds to the sum of absolute values of all of the entries.

Let $\text{nnz}(A)$ denote the number of nonzero entries of A. Let $\det(A)$ denote the determinant of a square matrix A. Let $A^\top$ denote the transpose of A. Let $A^\dagger$ denote the Moore-Penrose pseudoinverse of A. Let A^{-1} denote the inverse of a full rank square matrix.

For a 3rd order tensor $A \in \mathbb{R}^{n \times n \times n}$, its x-mode fibers are called column fibers ($x = 1$), row fibers ($x = 2$) and tube fibers ($x = 3$). For tensor A, we use $A_{*,j,l}$ to denote its (j,l)-th column, we use $A_{i,*,l}$ to denote its (i,l)-th row, and we use $A_{i,j,*}$ to denote its (i,j)-th tube.

A tensor A is symmetric if and only if for any i, j, k, $A_{i,j,k} = A_{i,k,j} = A_{j,i,k} = A_{j,k,i} = A_{k,i,j} = A_{k,j,i}$.

For a tensor $A \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, we use $\top$ to denote rotation (3 dimensional transpose) so that $A^\top \in \mathbb{R}^{n_3 \times n_1 \times n_2}$. For a tensor $A \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ and matrix $B \in \mathbb{R}^{n_3 \times k}$, we define the tensor-matrix dot product to be $A \cdot B \in \mathbb{R}^{n_1 \times n_2 \times k}$.

We use $\otimes$ to denote outer product, $\circ$ to denote entrywise product, and $\cdot$ to denote dot product. Given two column vectors $u, v \in \mathbb{R}^n$, let $u \otimes v \in \mathbb{R}^{n \times n}$ and $(u \otimes v)_{i,j} = u_i \cdot v_j$, $u^\top v = \sum_{i=1}^n u_i v_i \in \mathbb{R}$ and $(u \circ v)_i = u_i v_i$.

Definition 21.2.1 ($\otimes$ product for vectors)**.** Given q vectors $u_1 \in \mathbb{R}^{n_1}$, $u_2 \in \mathbb{R}^{n_2}$, $\cdots$, $u_q \in \mathbb{R}^{n_q}$, we use $u_1 \otimes u_2 \otimes \cdots \otimes u_q$ to denote an $n_1 \times n_2 \times \cdots \times n_q$ tensor such that, for each $(j_1, j_2, \cdots, j_q) \in [n_1] \times [n_2] \times \cdots \times [n_q]$,

$$(u_1 \otimes u_2 \otimes \cdots \otimes u_q)_{j_1, j_2, \cdots, j_q} = (u_1)_{j_1} (u_2)_{j_2} \cdots (u_q)_{j_q},$$

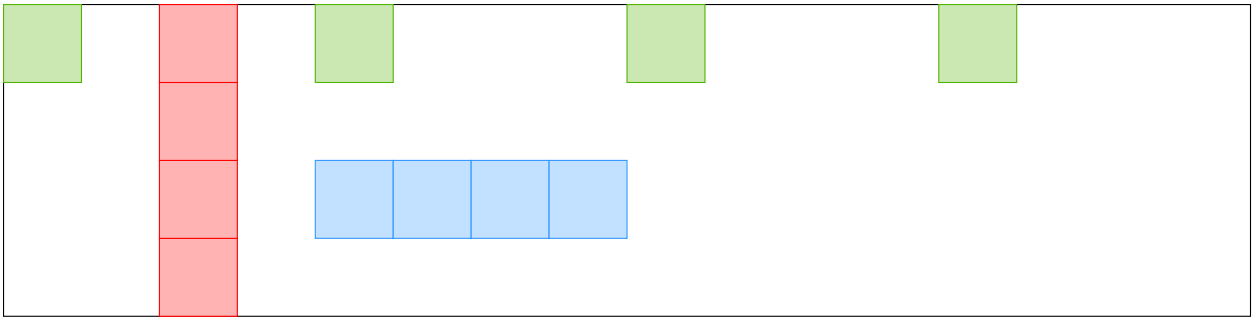


Figure 21.2: Flattening. We flatten a third order $4 \times 4 \times 4$ tensor along the 1st dimension to obtain a 4×16 matrix. The red blocks correspond to a column in the original third order tensor, the blue blocks correspond to a row in the original third order tensor, and the green blocks correspond to a tube in the original third order tensor.

where $(u_i)_{j_i}$ denotes the j_i-th entry of vector u_i.

Definition 21.2.2 (vec(), convert tensor into a vector)**.** Given a tensor $A \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_q}$, let $\text{vec}(A) \in \mathbb{R}^{1 \times \prod_{i=1}^{q} n_i}$ be a row vector, such that the *t-th* entry of $\text{vec}(A)$ is $A_{j_1, j_2, \cdots, j_q}$ where $t = (j_1 - 1) \prod_{i=2}^{q} n_i + (j_2 - 1) \prod_{i=3}^{q} n_i + \cdots + (j_{q-1} - 1) n_q + j_q$.

$$\text{For example if } u = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, v = \begin{bmatrix} 3 \\ 4 \\ 5 \end{bmatrix} \text{ then } \text{vec}(u \otimes v) = \begin{bmatrix} 3 & 4 & 5 & 6 & 8 & 10 \end{bmatrix}.$$

Definition 21.2.3 ($\otimes$ product for matrices)**.** Given q matrices $U_1 \in \mathbb{R}^{n_1 \times k}$, $U_2 \in \mathbb{R}^{n_2 \times k}$, $\cdots$, $U_q \in \mathbb{R}^{n_q \times k}$, we use $U_1 \otimes U_2 \otimes \cdots \otimes U_q$ to denote an $n_1 \times n_2 \times \cdots \times n_q$ tensor which can be written as,

$$U_1 \otimes U_2 \otimes \cdots \otimes U_q = \sum_{i=1}^{k} (U_1)_i \otimes (U_2)_i \otimes \cdots \otimes (U_q)_i \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_q},$$

where $(U_j)_i$ denotes the i-th column of matrix $U_j \in \mathbb{R}^{n_j \times k}$.

Definition 21.2.4 ($\odot$ product for matrices)**.** Given q matrices $U_1 \in \mathbb{R}^{k \times n_1}$, $U_2 \in \mathbb{R}^{k \times n_2}$, $\cdots$, $U_q \in \mathbb{R}^{k \times n_q}$, we use $U_1 \odot U_2 \odot \cdots \odot U_q$ to denote a $k \times \prod_{j=1}^{q} n_j$ matrix where the i-th

row of $U_1 \odot U_2 \odot \cdots \odot U_q$ is the vectorization of $(U_1)^i \otimes (U_2)^i \otimes \cdots \otimes (U_q)^i$, i.e.,

$$U_1 \odot U_2 \odot \cdots \odot U_q = \begin{bmatrix} \mathrm{vec}((U_1)^1 \otimes (U_2)^1 \otimes \cdots \otimes (U_q)^1) \\ \mathrm{vec}((U_1)^2 \otimes (U_2)^2 \otimes \cdots \otimes (U_q)^2) \\ \cdots \\ \mathrm{vec}((U_1)^k \otimes (U_2)^k \otimes \cdots \otimes (U_q)^k) \end{bmatrix} \in \mathbb{R}^{k \times \prod_{j=1}^q n_j}.$$

where $(U_j)^i \in \mathbb{R}^{n_j}$ denotes the i-th row of matrix $U_j \in \mathbb{R}^{k \times n_j}$.

Definition 21.2.5 (Flattening vs unflattening/retensorizing)**.** Suppose we are given three matrices $U \in \mathbb{R}^{n_1 \times k}$, $V \in \mathbb{R}_{n_2 \times k}$, $W \in \mathbb{R}^{n_3 \times k}$. Let tensor $A \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ denote $U \otimes V \otimes W$. Let $A_1 \in \mathbb{R}^{n_1 \times n_2 n_3}$ denote a matrix obtained by flattening tensor A along the 1st dimension. Then $A_1 = U \cdot B$, where $B = V^\top \odot W^\top \in \mathbb{R}^{k \times n_2 n_3}$ denotes the matrix for which the i-th row is $\mathrm{vec}(V_i \otimes W_i), \forall i \in [k]$. We let the "flattening" be the operation that obtains A_1 by A. Given $A_1 = U \cdot B$, we can obtain tensor A by unflattening/retensorizing A_1. We let "retensorization" be the operation that obtains A from A_1. Similarly, let $A_2 \in \mathbb{R}^{n_2 \times n_1 n_3}$ denote a matrix obtained by flattening tensor A along the 2nd dimension, so $A_2 = V \cdot C$, where $C = W^\top \odot U^\top \in \mathbb{R}^{k \times n_1 n_3}$ denotes the matrix for which the i-th row is $\mathrm{vec}(W_i \otimes U_i), \forall i \in [k]$. Let $A_3 \in \mathbb{R}^{n_3 \times n_1 n_2}$ denote a matrix obtained by flattening tensor A along the 3rd dimension. Then, $A_3 = W \cdot D$, where $D = U^\top \odot V^\top \in \mathbb{R}^{k \times n_1 n_2}$ denotes the matrix for which the i-th row is $\mathrm{vec}(U_i \otimes V_i), \forall i \in [k]$.

Definition 21.2.6 ($(\cdot, \cdot, \cdot)$ operator for tensors and matrices)**.** Given tensor $A \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ and three matrices $B_1 \in \mathbb{R}^{n_1 \times d_1}$, $B_2 \in \mathbb{R}^{n_2 \times d_2}$, $B_3 \in \mathbb{R}^{n_3 \times d_3}$, we define tensors $A(B_1, I, I) \in \mathbb{R}^{d_1 \times n_2 \times n_3}$, $A(I, B_2, I) \in \mathbb{R}^{n_1 \times d_2 \times n_3}$, $A(I, I, B_3) \in \mathbb{R}^{n_1 \times n_2 \times d_3}$, $A(B_1, B_2, I) \in \mathbb{R}^{d_1 \times d_2 \times n_3}$,

$A(B_1, B_2, B_3) \in \mathbb{R}^{d_1 \times d_2 \times d_3}$ as follows,

$$\begin{aligned}
A(B_1, I, I)_{i,j,l} &= \sum_{i'=1}^{n_1} A_{i',j,l}(B_1)_{i',i}, && \forall (i,j,l) \in [d_1] \times [n_2] \times [n_3] \\
A(I, B_2, I)_{i,j,l} &= \sum_{j'=1}^{n_2} A_{i,j',l}(B_2)_{j',j}, && \forall (i,j,l) \in [n_1] \times [d_2] \times [n_3] \\
A(I, I, B_3)_{i,j,l} &= \sum_{l'=1}^{n_3} A_{i,j,l'}(B_3)_{l',l}, && \forall (i,j,l) \in [n_1] \times [n_2] \times [d_3] \\
A(B_1, B_2, I)_{i,j,l} &= \sum_{i'=1}^{n_1} \sum_{j'=1}^{n_2} A_{i',j',l}(B_1)_{i',i}(B_2)_{j',j}, && \forall (i,j,l) \in [d_1] \times [d_2] \times [n_3] \\
A(B_1, B_2, B_3)_{i,j,l} &= \sum_{i'=1}^{n_1} \sum_{j'=1}^{n_2} \sum_{l'=1}^{n_3} A_{i',j',l'}(B_1)_{i',i}(B_2)_{j',j}(B_3)_{l',l}, && \forall (i,j,l) \in [d_1] \times [d_2] \times [d_3]
\end{aligned}$$

Note that $B_1^\top A = A(B_1, I, I)$, $AB_3 = A(I, I, B_3)$ and $B_1^\top A B_3 = A(B_1, I, B_3)$. In our paper, if $\forall i \in [3], B_i$ is either a rectangular matrix or a symmetric matrix, then we sometimes use $A(B_1, B_2, B_3)$ to denote $A(B_1^\top, B_2^\top, B_3^\top)$ for simplicity. Similar to the $(\cdot, \cdot, \cdot)$ operator on 3rd order tensors, we can define the $(\cdot, \cdot, \cdots, \cdot)$ operator on higher order tensors.

For the matrix case, $\min_{\text{rank}-k\ A'} \|A - A'\|_F^2$ always exists. However, this is not true for tensors [DSL08]. For convenience, we redefine the notation of OPT and min.

Definition 21.2.7. Given tensor $A \in \mathbb{R}^{n_1 \times n_2 \times n_3}, k > 0$, if $\min_{\text{rank}-k\ A'} \|A - A'\|_F^2$ does not exist, then we define $\text{OPT} = \inf_{\text{rank}-k\ A'} \|A - A'\|_F^2 + \gamma$ for sufficiently small $\gamma > 0$, which can be an arbitrarily small positive function of n. We let $\min_{\text{rank}-k\ A'} \|A - A'\|_F^2$ be the value of OPT, and we let $\arg\min_{\text{rank}-k\ A'} \|A - A'\|_F^2$ be a rank $-k$ tensor $A_k \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ which satisfies $\|A - A_k\|_F^2 = \text{OPT}$.

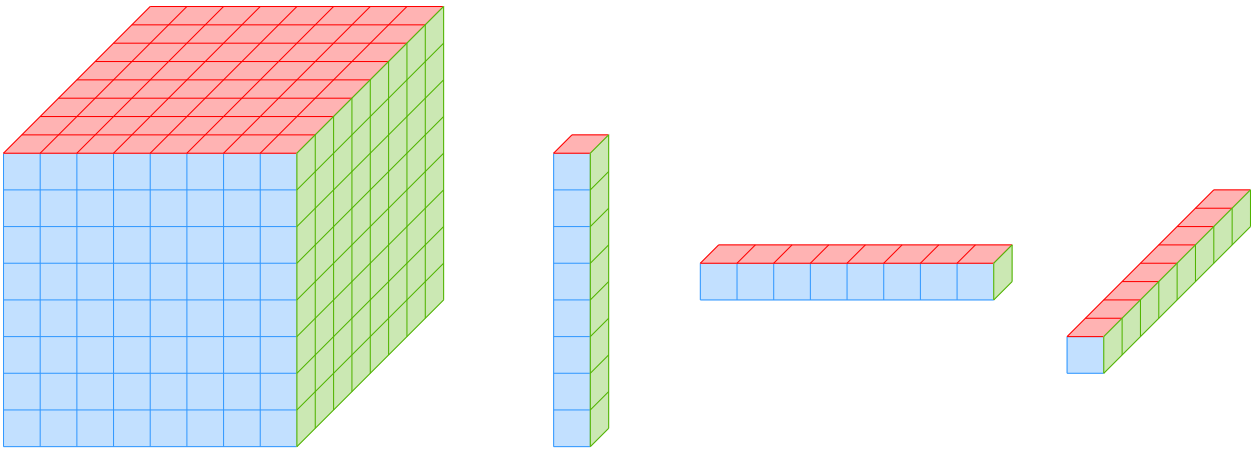


Figure 21.3: A 3rd order tensor contains n^2 columns, n^2 rows, and n^2 tubes.

21.3 Preliminaries

Section 21.3.1 provides the definitions for Subspace Embeddings and Approximate Matrix Product. We introduce the definition for Tensor-CURT decomposition in Section 21.3.2. Section 21.3.3 presents a tool which we call a "polynomial system verifier". Section 21.3.4 introduces a tool which is able to determine the minimum nonzero value of the absolute value of a polynomial evaluated on a set, provided the polynomial is never equal to 0 on that set. Section 21.3.5 shows how to relax an ℓ_p problem to an ℓ_2 problem. We provide definitions for CountSketch and Gaussian transforms in Section 21.3.6. We present Cauchy and p-stable transforms in Section 21.3.7. We introduce leverage scores and Lewis weights in Section 21.3.8 and Section 21.3.9. Finally, we explain an extension of CountSketch, which is called TensorSketch in Section 21.3.10.

21.3.1 Subspace Embeddings and Approximate Matrix Product

Definition 21.3.1 (Subspace Embedding)**.** A $(1 \pm \epsilon)$ ℓ_2-subspace embedding for the column space of an $n \times d$ matrix A is a matrix S for which for all $x \in \mathbb{R}^d$, $\|SAx\|_2^2 = (1 \pm \epsilon)\|Ax\|_2^2$.

Definition 21.3.2 (Approximate Matrix Product)**.** Let $0 < \epsilon < 1$ be a given approximation parameter. Given matrices A and B, where A and B each have n rows, the goal is to output a matrix C so that $\|A^\top B - C\|_F \leq \epsilon \|A\|_F \|B\|_F$. Typically C has the form $A^\top S^\top SB$, for a random matrix S with a small number of rows. See, e.g., Lemma 32 of [CW13] for a number of example matrices S with $O(\epsilon^{-2})$ rows for which this property holds.

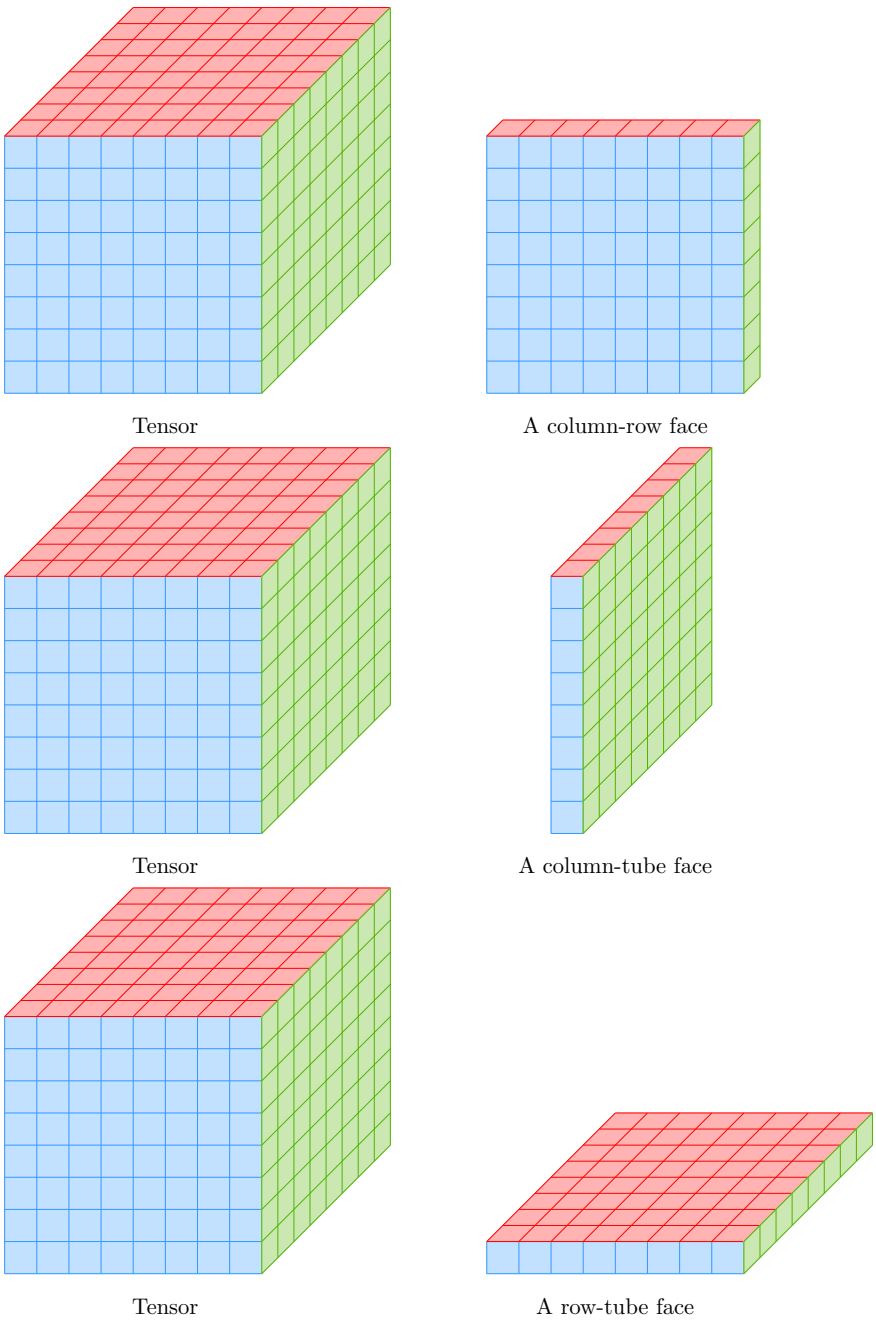


Figure 21.4: A third order tensor has three types of faces: the column-row faces, the column-tube faces, and the row-tube faces

21.3.2 Tensor CURT decomposition

We first review matrix CUR decompositions:

Definition 21.3.3 (Matrix CUR, exact)**.** Given a matrix $A \in \mathbb{R}^{n \times d}$, we choose $C \in \mathbb{R}^{n \times c}$ to be a subset of columns of A and $R \in \mathbb{R}^{r \times n}$ to be a subset of rows of A. If there exists a matrix $U \in \mathbb{R}^{c \times r}$ such that A can be written as,

$$CUR = A,$$

then we say C, U, R is matrix A's CUR decomposition.

Definition 21.3.4 (Matrix CUR, approximate)**.** Given a matrix $A \in \mathbb{R}^{n \times d}$, a parameter $k \geq 1$, an approximation ratio $\alpha > 1$, and a norm $\|\|\|_\xi$, we choose $C \in \mathbb{R}^{n \times c}$ to be a subset of columns of A and $R \in \mathbb{R}^{r \times n}$ to be a subset of rows of A. Then if there exists a matrix $U \in \mathbb{R}^{c \times r}$ such that,

$$\|CUR - A\|_\xi \leq \alpha \min_{\text{rank}-k \ A_k} \|A_k - A\|_\xi,$$

where $\|\|\|_\xi$ can be operator norm, Frobenius norm or Entry-wise ℓ_1 norm, we say that C, U, R is matrix A's approximate CUR decomposition, and sometimes just refer to this as a CUR decomposition.

Definition 21.3.5 ([Bou11])**.** Given matrix $A \in \mathbb{R}^{m \times n}$, integer k, and matrix $C \in \mathbb{R}^{m \times r}$ with $r > k$, we define the matrix $\Pi^\xi_{C,k}(A) \in \mathbb{R}^{m \times n}$ to be the best approximation to A (under the ξ-norm) within the column space of C of rank at most k; so, $\Pi^\xi_{C,k}(A) \in \mathbb{R}^{m \times n}$ minimizes the residual $\|A - \widehat{A}\|_\xi$, over all $\widehat{A} \in \mathbb{R}^{m \times n}$ in the column space of C of rank at most k.

We define the following notion of tensor-CURT decomposition.

Definition 21.3.6 (Tensor CURT, exact)**.** Given a tensor $A \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, we choose three sets of pair of coordinates $S_1 \subseteq [n_2] \times [n_3], S_2 \subseteq [n_1] \times [n_3], S_3 \subseteq [n_1] \times [n_2]$. We define $c = |S_1|$, $r = |S_2|$ and $t = |S_3|$. Let $C \in \mathbb{R}^{n_1 \times c}$ denote a subset of columns of A, $R \in \mathbb{R}^{n_2 \times r}$ denote a subset of rows of A, and $T \in \mathbb{R}^{n_3 \times t}$ denote a subset of tubes of A. If there exists a tensor $U \in \mathbb{R}^{c \times r \times t}$ such that A can be written as

$$(((U \cdot T^\top)^\top \cdot R^\top)^\top \cdot C^\top)^\top = A,$$

or equivalently,

$$U(C, R, T) = A,$$

or equivalently,

$$\forall (i,j,l) \in [n_1] \times [n_2] \times [n_3], A_{i,j,l} = \sum_{u_1=1}^{c} \sum_{u_2=1}^{r} \sum_{u_3=1}^{t} U_{u_1,u_2,u_3} C_{i,u_1} R_{j,u_2} T_{l,u_3},$$

then we say C, U, R, T is tensor A's CURT decomposition.

Definition 21.3.7 (Tensor CURT, approximate)**.** Given a tensor $A \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, for some $k \geq 1$, for some approximation $\alpha > 1$, for some norm $\|\|_\xi$, we choose three sets of pair of coordinates $S_1 \subseteq [n_2] \times [n_3], S_2 \subseteq [n_1] \times [n_3], S_3 \subseteq [n_1] \times [n_2]$. We define $c = |S_1|$, $r = |S_2|$ and $t = |S_3|$. Let $C \in \mathbb{R}^{n_1 \times c}$ denote a subset of columns of A, $R \in \mathbb{R}^{n_2 \times r}$ denote a subset of rows of A, and $T \in \mathbb{R}^{n_3 \times t}$ denote a subset of tubes of A. If there exists a tensor $U \in \mathbb{R}^{c \times r \times t}$ such that

$$\|U(C, R, T) - A\|_\xi \leq \alpha \min_{\text{rank}-k \ A_k} \|A_k - A\|_\xi,$$

where $\|\|_\xi$ is operator norm, Frobenius norm or Entry-wise ℓ_1 norm, then we refer to C, U, R, T as an approximate CUR decomposition of A, and sometimes just refer to this as a CURT decomposition of A.

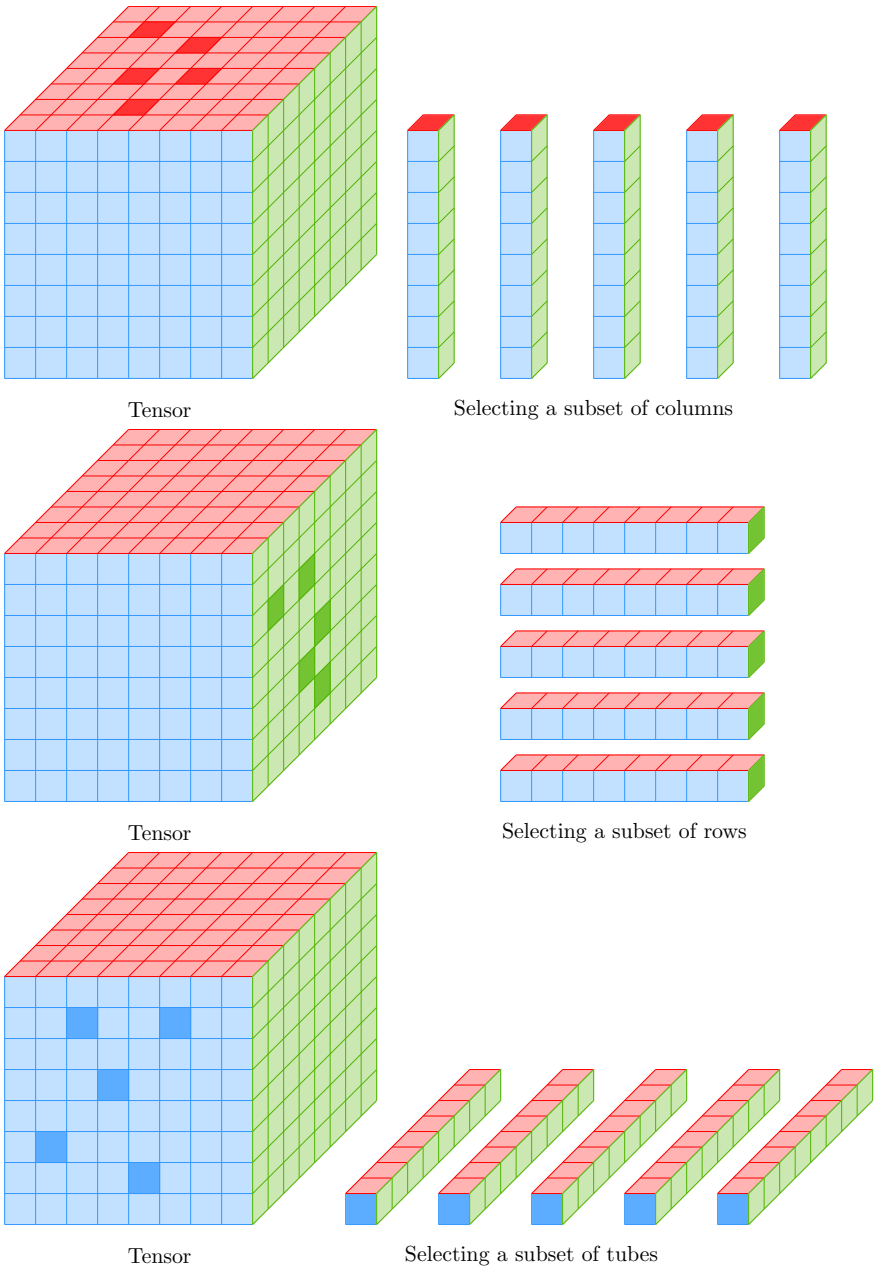


Figure 21.5: Column subset selection, row subset selection and tube subset selection.

Recently, [TM17] studied a very different face-based tensor-CUR decomposition, which selects faces from tensors rather than columns. To achieve their results, [TM17] need to make several incoherence assumptions on the original tensor. Their sample complexity depends on $\log n$, and they only sample two of the three dimensions. We will provide more general face-based tensor CURT decompositions.

Definition 21.3.8 (Tensor (face-based) CURT, exact)**.** Given a tensor $A \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, we choose three sets of coordinates $S_1 \subseteq [n_1], S_2 \subseteq [n_2], S_3 \subseteq [n_3]$. We define $c = |S_1|$, $r = |S_2|$ and $t = |S_3|$. Let $C \in \mathbb{R}^{c \times n_2 \times n_3}$ denote a subset of row-tube faces of A, $R \in \mathbb{R}^{n_1 \times r \times n_3}$ denote a subset of column-tube faces of A, and $T \in \mathbb{R}^{n_1 \times n_2 \times t}$ denote a subset of column-row faces of A. Let $C_2 \in \mathbb{R}^{n_2 \times cn_3}$ denote the matrix obtained by flattening the tensor C along the second dimension. Let $R_3 \in \mathbb{R}^{n_3 \times rn_1}$ denote the matrix obtained by flattening the tensor R along the third dimension. Let $T_1 \in \mathbb{R}^{n_1 \times tn_2}$ denote the matrix obtained by flattening the tensor T along the first dimension. If there exists a tensor $U \in \mathbb{R}^{tn_2 \times cn_3 \times rn_1}$ such that A can be written as

$$\sum_{i=1}^{tn_2} \sum_{j=1}^{cn_3} \sum_{l=1}^{rn_1} U_{i,j,l} (T_1)_l \otimes (C_2)_i \otimes (R_3)_j = A,$$

$$U(T_1, C_2, R_3) = A,$$

or equivalently,

$$\forall (i', j', l') \in [n_1] \times [n_2] \times [n_3], A_{i,j,l} = \sum_{i=1}^{tn_1} \sum_{j=1}^{cn_3} \sum_{l=1}^{rn_2} U_{i,j,l} (T_1)_{i',i} (C_2)_{j',j} (R_3)_{l',l},$$

then we say C, U, R, T is tensor A's (face-based) CURT decomposition.

Definition 21.3.9 (Tensor (face-based) CURT, approximate)**.** Given a tensor $A \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, for some $k \geq 1$, for some approximation $\alpha > 1$, for some norm $\|\|\|_\xi$,we choose three sets of coordinates $S_1 \subseteq [n_1], S_2 \subseteq [n_2], S_3 \subseteq [n_3]$. We define $c = |S_1|$, $r = |S_2|$ and $t = |S_3|$. Let $C \in \mathbb{R}^{c \times n_2 \times n_3}$ denote a subset of row-tube faces of A, $R \in \mathbb{R}^{n_1 \times r \times n_3}$ denote a subset of column-tube faces of A, and $T \in \mathbb{R}^{n_1 \times n_2 \times t}$ denote a subset of column-row faces of A. Let $C_2 \in \mathbb{R}^{n_2 \times cn_3}$ denote the matrix obtained by flattening the tensor C along the second dimension. Let $R_3 \in \mathbb{R}^{n_3 \times rn_1}$ denote the matrix obtained by flattening the tensor R along the third dimension. Let $T_1 \in \mathbb{R}^{n_1 \times tn_2}$ denote the matrix obtained by flattening the tensor T along the first dimension. If there exists a tensor $U \in \mathbb{R}^{tn_2 \times cn_3 \times rn_1}$ such that

$$\|U(T_1, C_2, R_3) - A\|_\xi \leq \alpha \min_{\text{rank}-k \ A_k} \|A_k - A\|_\xi,$$

where $\|\|\|_\xi$ is operator norm, Frobenius norm or Entry-wise ℓ_1 norm, then we refer to C, U, R, T as an approximate CUR decomposition of A, and sometimes just refer to this as a (face-based) CURT decomposition of A.

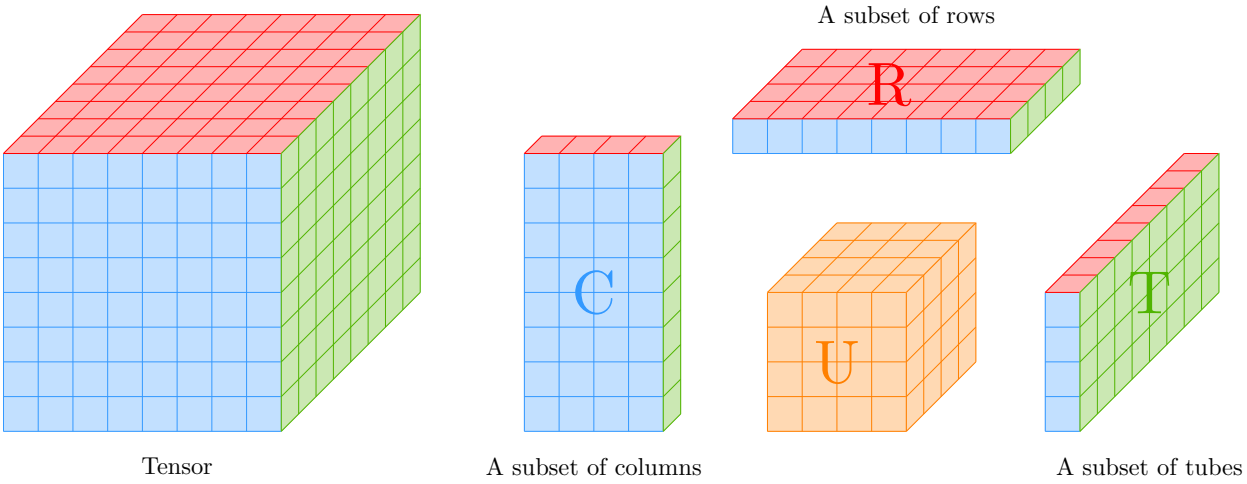


Figure 21.6: An example tensor CURT decomposition.

21.3.3 Polynomial system verifier

We use the polynomial system verifiers independently developed by Renegar [Ren92a, Ren92b] and Basu *et al.* [BPR96].

Theorem 21.3.1 (Decision Problem [Ren92a, Ren92b, BPR96])**.** *Given a real polynomial system $P(x_1, x_2, \cdots, x_v)$ having v variables and m polynomial constraints $f_i(x_1, x_2, \cdots, x_v)\Delta_i 0, \forall i \in [m]$, where Δ_i is any of the "standard relations": $\{>, \geq, =, \neq, \leq, <\}$, let d denote the maximum degree of all the polynomial constraints and let H denote the maximum bitsize of the coefficients of all the polynomial constraints. Then in*

$$(md)^{O(v)} \operatorname{poly}(H),$$

time one can determine if there exists a solution to the polynomial system P.

Recently, this technique has been used to solve a number of low-rank approximation and matrix factorization problems [AGKM12, Mad13, CW15a, BDL16, RSW16, SWZ17].

21.3.4 Lower bound on the cost of a polynomial system

An important result we use is the following lower bound on the minimum value attained by a polynomial restricted to a compact connected component of a basic closed semi-algebraic subset of $\mathbb{R}^v$.

Theorem 21.3.2 ([JPT13]). *Let* $T = \{x \in \mathbb{R}^v | f_1(x) \geq 0, \cdots, f_\ell(x) \geq 0, f_{\ell+1}(x) = 0, \cdots, f_m(x) = 0\}$ *be defined by polynomials* $f_1, \cdots, f_m \in \mathbb{Z}[x_1, \cdots, x_v]$ *with* $n \geq 2$*, degrees bounded by an even integer* d*, and coefficients of absolute value at most* H*, and let* C *be a compact connected (in the topological sense) component of* T*. Let* $g \in \mathbb{Z}[x_1, \cdots, x_v]$ *be a polynomial of degree at most* d *and coefficients of absolute value bounded by* H*. Then, the minimum value that* g *takes over* C *satisfies that if it is not zero, then its absolute value is greater than or equal to*

$$(2^{4-v/2}\widetilde{H}d^v)^{-v2^v d^v},$$

where $\widetilde{H} = \max\{H, 2v + 2m\}$.

While the above theorem involves notions from topology, we shall apply it in an elementary way. Namely, in our setting T will be bounded and so every connected component, which is by definition closed, will also be bounded and therefore compact. As the connected components partition T the theorem will just be applied to give a global minimum value of g on T provided that it is non-zero.

21.3.5 Frobenius norm and ℓ_2 relaxation

Theorem 21.3.3 (Generalized rank-constrained matrix approximations, Theorem 2 in [FT07])**.** *Given matrices $A \in \mathbb{R}^{n\times d}$, $B \in \mathbb{R}^{n\times p}$, and $C \in \mathbb{R}^{q\times d}$, let the SVD of B be $B = U_B \Sigma_B V_B^\top$ and the SVD of C be $C = U_C \Sigma_C V_C^\top$. Then,*

$$B^\dagger (U_B U_B^\top A V_C C_C^\top)_k C^\dagger = \underset{\text{rank}-k \;\; X\in\mathbb{R}^{p\times q}}{\arg\min} \|A - BXC\|_F,$$

where $(U_B U_B^\top A V_C V_C^\top)_k \in \mathbb{R}^{p\times q}$ is of rank at most k and denotes the best rank-k approximation to $U_B U_B^\top A V_C V_C^\top \in \mathbb{R}^{p\times d}$ in Frobenius norm.

Claim 21.3.4 (ℓ_2 relaxation of ℓ_p-regression)**.** *Let $p \in [1, 2)$. For any $A \in \mathbb{R}^{n\times d}$ and $b \in \mathbb{R}^n$, define $x^* = \underset{x\in\mathbb{R}^d}{\arg\min} \|Ax - b\|_p$ and $x' = \underset{x\in\mathbb{R}^d}{\arg\min} \|Ax - b\|_2$. Then,*

$$\|Ax^* - b\|_p \le \|Ax' - b\|_p \le n^{1/p-1/2} \cdot \|Ax^* - b\|_p.$$

Claim 21.3.5 ((Matrix) Frobenius norm relaxation of ℓ_p-low rank approximation)**.** *Let $p \in [1, 2)$ and for any matrix $A \in \mathbb{R}^{n\times d}$, define $A^* = \underset{\text{rank}-k \;\; B\in\mathbb{R}^{n\times d}}{\arg\min} \|B - A\|_p$ and $A' = \underset{\text{rank}-k \;\; B\in\mathbb{R}^{n\times d}}{\arg\min} \|B - A\|_F$. Then*

$$\|A^* - A\|_p \le \|A' - A\|_p \le (nd)^{1/p-1/2} \|A^* - A\|_p.$$

Claim 21.3.6 ((Tensor) Frobenius norm relaxation of ℓ_p-low rank approximation)**.** *Let $p \in [1, 2)$ and for any matrix $A \in \mathbb{R}^{n_1\times n_2\times n_3}$, define*

$$A^* = \underset{\text{rank}-k \;\; B\in\mathbb{R}^{n_1\times n_2\times n_3}}{\arg\min} \|B - A\|_p$$

and

$$A' = \underset{\text{rank}-k \;\; B\in\mathbb{R}^{n_1\times n_2\times n_3}}{\arg\min} \|B - A\|_F.$$

Then

$$\|A^* - A\|_p \leq \|A' - A\|_p \leq (n_1 n_2 n_3)^{1/p-1/2} \|A^* - A\|_p.$$

21.3.6 CountSketch and Gaussian transforms

Definition 21.3.10 (Sparse embedding matrix or CountSketch transform)**.** A CountSketch transform is defined to be $\Pi = \sigma \cdot \Phi D \in \mathbb{R}^{m \times n}$. Here, σ is a scalar, D is an $n \times n$ random diagonal matrix with each diagonal entry independently chosen to be $+1$ or -1 with equal probability, and $\Phi \in \{0,1\}^{m \times n}$ is an $m \times n$ binary matrix with $\Phi_{h(i),i} = 1$ and all remaining entries 0, where $h : [n] \to [m]$ is a random map such that for each $i \in [n]$, $h(i) = j$ with probability $1/m$ for each $j \in [m]$. For any matrix $A \in \mathbb{R}^{n \times d}$, ΠA can be computed in $O(\text{nnz}(A))$ time. For any tensor $A \in \mathbb{R}^{n \times d_1 \times d_2}$, ΠA can be computed in $O(\text{nnz}(A))$ time. Let Π_1, Π_2, Π_3 denote three CountSktech transforms. For any tensor $A \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, $A(\Pi_1, \Pi_2, \Pi_3)$ can be computed in $O(\text{nnz}(A))$ time.

If the above scalar σ is not specified in the context, we assume the scalar σ to be 1.

Definition 21.3.11 (Gaussian matrix or Gaussian transform)**.** Let $S = \sigma \cdot G \in \mathbb{R}^{m \times n}$ where σ is a scalar, and each entry of $G \in \mathbb{R}^{m \times n}$ is chosen independently from the standard Gaussian distribution. For any matrix $A \in \mathbb{R}^{n \times d}$, SA can be computed in $O(m \cdot \text{nnz}(A))$ time. For any tensor $A \in \mathbb{R}^{n \times d_1 \times d_2}$, SA can be computed in $O(m \cdot \text{nnz}(A))$ time.

If the above scalar σ is not specified in the context, we assume the scalar σ to be $1/\sqrt{m}$. In most places, we can combine CountSketch and Gaussian transforms to achieve the following:

Definition 21.3.12 (CountSketch + Gaussian transform)**.** Let $S' = S\Pi$, where $\Pi \in \mathbb{R}^{t \times n}$ is the CountSketch transform (defined in Definition 21.3.10) and $S \in \mathbb{R}^{m \times t}$ is the Gaussian transform (defined in Definition 21.3.11). For any matrix $A \in \mathbb{R}^{n \times d}$, $S'A$ can be computed in $O(\text{nnz}(A) + dtm^{\omega-2})$ time, where ω is the matrix multiplication exponent.

Lemma 21.3.7 (Affine Embedding - Theorem 39 in [CW13]). *Given matrices $A \in \mathbb{R}^{n\times r}, B \in \mathbb{R}^{n\times d}$, and $\mathrm{rank}(A) = k$, let $m = \mathrm{poly}(k/\epsilon)$, $S \in \mathbb{R}^{m\times n}$ be a sparse embedding matrix (Definition 21.3.10) with scalar $\sigma = 1$. Then with probability at least 0.999, $\forall X \in \mathbb{R}^{r\times d}$, we have*

$$(1-\epsilon)\cdot\|AX-B\|_F^2 \leq \|S(AX-B)\|_F^2 \leq (1+\epsilon)\|AX-B\|_F^2.$$

Lemma 21.3.8 (see, e.g., Lemma 10 in version 1 of [BWZ16][8]). *Let $m = \Omega(k/\epsilon)$, $S = \frac{1}{\sqrt{m}}\cdot G$, where $G \in \mathbb{R}^{m\times n}$ is a random matrix where each entry is an i.i.d Gaussian $N(0,1)$. Then with probability at least 0.998, S satisfies $(1\pm 1/8)$ Subspace Embedding (Definition 21.3.1) for any fixed matrix $C \in \mathbb{R}^{n\times k}$, and it also satisfies $O(\sqrt{\epsilon/k})$ Approximate Matrix Product (Definition 21.3.2) for any fixed matrix A and B which has the same number of rows.*

Lemma 21.3.9 (see, e.g., Lemma 11 in version 1 of [BWZ16]). *Let $m = \Omega(k^2 + k/\epsilon)$, $\Pi \in \mathbb{R}^{m\times n}$, where Π is a sparse embedding matrix (Definition 21.3.10) with scalar $\sigma = 1$, then with probability at least 0.998, S satisfies $(1\pm 1/8)$ Subspace Embedding (Definition 21.3.1) for any fixed matrix $C \in \mathbb{R}^{n\times k}$, and it also satisfies $O(\sqrt{\epsilon/k})$ Approximate Matrix Product (Definition 21.3.2) for any fixed matrix A and B which has the same number of rows.*

Lemma 21.3.10 (see, e.g., Lemma 12 in version 1 of [BWZ16]). *Let $m_2 = \Omega(k^2 + k/\epsilon)$, $\Pi \in \mathbb{R}^{m_2\times n}$, where Π is a sparse embedding matrix (Definition 21.3.10) with scalar $\sigma = 1$. Let $m_1 = \Omega(k/\epsilon)$, $S = \frac{1}{\sqrt{m_1}}\cdot G$, where $G \in \mathbb{R}^{m_1\times m_2}$ is a random matrix where each entry is an i.i.d Gaussian $N(0,1)$. Let $S' = S\Pi$. Then with probability at least 0.99, S' is a $(1\pm 1/3)$ Subspace Embedding (Definition 21.3.1) for any fixed matrix $C \in \mathbb{R}^{n\times k}$, and it also satisfies*

[8] https://arxiv.org/pdf/1504.06729v1.pdf

$O(\sqrt{\epsilon/k})$ *Approximate Matrix Product (Definition 21.3.2) for any fixed matrix A and B which have the same number of rows.*

Theorem 21.3.11 (Theorem 36 in [CW13])**.** *Given $A \in \mathbb{R}^{n\times k}, B \in \mathbb{R}^{n\times d}$, suppose $S \in \mathbb{R}^{m\times n}$ is such that S is a $(1 \pm \frac{1}{\sqrt{2}})$ Subspace Embedding for A, and satisfies $O(\sqrt{\epsilon/k})$ Approximate Matrix Product for matrices A and C where C with n rows, where C depends on A and B. If*

$$\widehat{X} = \arg\min_{X\in\mathbb{R}^{k\times d}} \|SAX - SB\|_F^2,$$

then

$$\|A\widehat{X} - B\|_F^2 \leq (1+\epsilon)\min_{X\in\mathbb{R}^{k\times d}} \|AX - B\|_F^2.$$

21.3.7 Cauchy and p-stable transforms

Definition 21.3.13 (Dense Cauchy transform)**.** Let $S = \sigma \cdot C \in \mathbb{R}^{m \times n}$ where σ is a scalar, and each entry of $C \in \mathbb{R}^{m \times n}$ is chosen independently from the standard Cauchy distribution. For any matrix $A \in \mathbb{R}^{n \times d}$, SA can be computed in $O(m \cdot \text{nnz}(A))$ time.

Definition 21.3.14 (Sparse Cauchy transform)**.** Let $\Pi = \sigma \cdot SC \in \mathbb{R}^{m \times n}$, where σ is a scalar, $S \in \mathbb{R}^{m \times n}$ has each column chosen independently and uniformly from the m standard basis vectors of $\mathbb{R}^m$, and $C \in \mathbb{R}^{n \times n}$ is a diagonal matrix with diagonals chosen independently from the standard Cauchy distribution. For any matrix $A \in \mathbb{R}^{n \times d}$, ΠA can be computed in $O(\text{nnz}(A))$ time. For any tensor $A \in \mathbb{R}^{n \times d_1 \times d_2}$, ΠA can be computed in $O(\text{nnz}(A))$ time. Let $\Pi_1 \in \mathbb{R}^{m_1 \times n_1}, \Pi_2 \in \mathbb{R}^{m_2 \times n_2}, \Pi_3 \in \mathbb{R}^{m_3 \times n_3}$ denote three sparse Cauchy transforms. For any tensor $A \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, $A(\Pi_1, \Pi_2, \Pi_3) \in \mathbb{R}^{m_1 \times m_2 \times m_3}$ can be computed in $O(\text{nnz}(A))$ time.

Definition 21.3.15 (Dense p-stable transform)**.** Let $p \in (1, 2)$. Let $S = \sigma \cdot C \in \mathbb{R}^{m \times n}$, where σ is a scalar, and each entry of $C \in \mathbb{R}^{m \times n}$ is chosen independently from the standard p-stable distribution. For any matrix $A \in \mathbb{R}^{n \times d}$, SA can be computed in $O(m \, \text{nnz}(A))$ time.

Definition 21.3.16 (Sparse p-stable transform)**.** Let $p \in (1, 2)$. Let $\Pi = \sigma \cdot SC \in \mathbb{R}^{m \times n}$, where σ is a scalar, $S \in \mathbb{R}^{m \times n}$ has each column chosen independently and uniformly from the m standard basis vectors of $\mathbb{R}^m$, and $C \in \mathbb{R}^{n \times n}$ is a diagonal matrix with diagonals chosen independently from the standard p-stable distribution. For any matrix $A \in \mathbb{R}^{n \times d}$, ΠA can be computed in $O(\text{nnz}(A))$ time. For any tensor $A \in \mathbb{R}^{n \times d_1 \times d_2}$, ΠA can be computed in $O(\text{nnz}(A))$ time. Let $\Pi_1 \in \mathbb{R}^{m_1 \times n_1}, \Pi_2 \in \mathbb{R}^{m_2 \times n_2}, \Pi_3 \in \mathbb{R}^{m_3 \times n_3}$ denote three sparse p-stable transforms. For any tensor $A \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, $A(\Pi_1, \Pi_2, \Pi_3) \in \mathbb{R}^{m_1 \times m_2 \times m_3}$ can be computed in $O(\text{nnz}(A))$ time.

21.3.8 Leverage scores

Definition 21.3.17 (Leverage scores)**.** Let $U \in \mathbb{R}^{n \times k}$ have orthonormal columns, and let $p_i = u_i^2/k$, where $u_i^2 = \|e_i^\top U\|_2^2$ is the i-th leverage score of U.

Definition 21.3.18 (Leverage score sampling)**.** Given $A \in \mathbb{R}^{n \times d}$ with rank k, let $U \in \mathbb{R}^{n \times k}$ be an orthonormal basis of the column space of A, and for each i let p_i be the squared row norm of the i-th row of U, i.e., the i-th leverage score. Let $k \cdot p_i$ denote the i-th leverage score of U scaled by k. Let $\beta > 0$ be a constant and $q = (q_1, \cdots, q_n)$ denote a distribution such that, for each $i \in [n]$, $q_i \geq \beta p_i$. Let s be a parameter. Construct an $n \times s$ sampling matrix B and an $s \times s$ rescaling matrix D as follows. Initially, $B = 0^{n \times s}$ and $D = 0^{s \times s}$. For each column j of B, D, independently, and with replacement, pick a row index $i \in [n]$ with probability q_i, and set $B_{i,j} = 1$ and $D_{j,j} = 1/\sqrt{q_i s}$. We denote this procedure LEVERAGE SCORE SAMPLING according to the matrix A.

21.3.9 Lewis weights

We follow the exposition of Lewis weights from [CP15].

Definition 21.3.19. For a matrix A, let a_i denote the i^{th} row of A, where $a_i (= (A^i)^\top)$ is a column vector. The statistical leverage score of a row a_i is

$$\tau_i(A) \stackrel{\text{def}}{=} a_i^\top (A^\top A)^{-1} a_i = \|(A^\top A)^{-1/2} a_i\|_2^2.$$

For a matrix A and norm p, the ℓ_p Lewis weights w are the unique weights such that for each row i we have

$$w_i = \tau_i(W^{1/2-1/p} A).$$

or equivalently,

$$a_i^\top (A^\top W^{1-2/p} A)^{-1} a_i = w_i^{2/p}.$$

Lemma 21.3.12 (Lemma 2.4 of [CP15] and Lemma 7 of [CLM+15])**.** *Given a matrix $A \in \mathbb{R}^{n \times d}$, $n \geq d$, for any constant $C > 0, 4 > p \geq 1$, there is an algorithm which can compute C-approximate ℓ_p Lewis weights for every row i of A in $O((nnz(A) + d^\omega \log d) \log n)$ time, where $\omega < 2.373$ is the matrix multiplication exponent[Str69, CW87, Wil12].*

Lemma 21.3.13 (Theorem 7.1 of [CP15])**.** *Given matrix $A \in \mathbb{R}^{n \times d}$ ($n \geq d$) with ℓ_p ($4 > p \geq 1$) Lewis weights w, for any set of sampling probabilities p_i, $\sum_i p_i = N$,*

$$p_i \geq f(d, p) w_i,$$

if $S \in \mathbb{R}^{N \times n}$ has each row chosen independently as the i^{th} standard basis vector, multiplied by $1/p_i^{1/p}$, with probability p_i/N. Then, overall with probability at least 0.999,

$$\forall x \in \mathbb{R}^d, \frac{1}{2}\|Ax\|_p^p \leq \|SAx\|_p^p \leq 2\|Ax\|_p^p.$$

Furthermore, if $p = 1$, $N = O(d \log d)$. *If* $1 < p < 2$, $N = O(d \log d \log \log d)$. *If* $2 \leq p < 4$, $N = O(d^{p/2} \log d)$.

Lemma 21.3.14. *Given matrix* $A \in \mathbb{R}^{n \times d}$ $(n \geq d)$, *there is an algorithm to compute a diagonal matrix* $D = SS_1$ *with* N *nonzero entries in* $O(n \operatorname{poly}(d))$ *time such that, with probability at least* 0.999, *for all* $x \in \mathbb{R}^d$

$$\frac{1}{10}\|DAx\|_p^p \leq \|Ax\|_p^p \leq 10\|DAx\|_p^p,$$

where S, S_1 *are two sampling/rescaling matrices. Furthermore, if* $p = 1$, *then* $N = O(d \log d)$. *If* $1 < p < 2$, *then* $N = O(d \log d \log \log d)$. *If* $2 \leq p < 4$, *then* $N = O(d^{p/2} \log d)$.

Given a matrix $A \in \mathbb{R}^{n \times d}$ $(n \geq d)$, by Lemma 21.3.13 and Lemma 21.3.12, we can compute a sampling/rescaling matrix S in $O((nnz(A) + d^\omega \log d) \log n)$ time with $\widetilde{O}(d)$ nonzero entries such that

$$\forall x \in \mathbb{R}^d, \frac{1}{2}\|Ax\|_p^p \leq \|SAx\|_p^p \leq 2\|Ax\|_p^p.$$

Sometimes, $\operatorname{poly}(d)$ is much smaller than $\log n$. In this case, we are also able to compute such a sampling/rescaling matrix S in $n \operatorname{poly}(d)$ time in an alternative way.

To do so, we run one of the input sparsity ℓ_p embedding algorithms (see e.g., [MM13]) to compute a well conditioned basis U of the column span of A in $n \operatorname{poly}(d/\epsilon)$ time. By sampling according to the well conditioned basis (see e.g. [Cla05, DDH+09, Woo14b]), we can compute a sampling/rescaling matrix S_1 such that $(1-\epsilon)\|Ax\|_p^p \leq \|S_1 Ax\|_p^p \leq (1+\epsilon)\|Ax\|_p^p$ where $\epsilon \in (0, 1)$ is an arbitrary constant. Notice that S_1 has $\operatorname{poly}(d/\epsilon)$ nonzero entries, and thus $S_1 A$ has size $\operatorname{poly}(d/\epsilon)$. Next, we apply Lewis weight sampling according to $S_1 A$, and

we obtain a sampling/rescaling matrix S for which

$$\forall x \in \mathbb{R}^d, (1 - \frac{1}{3})\|S_1 Ax\|_p^p \leq \|SS_1 Ax\|_p^p \leq (1 + \frac{1}{3})\|S_1 Ax\|_p^p.$$

This implies that

$$\forall x \in \mathbb{R}^d, \frac{1}{2}\|Ax\|_p^p \leq \|SS_1 Ax\|_p^p \leq 2\|Ax\|_p^p.$$

Note that SS_1 is still a sampling/rescaling matrix according to A, and the number of non-zero entries is $\widetilde{O}(d)$. The total running time is thus $n\,\mathrm{poly}(d/\epsilon)$, as desired.

21.3.10 TensorSketch

Let $\phi(v_1, v_2, \cdots, v_q)$ denote the function that maps q vectors($u_i \in \mathbb{R}^{n_i}$) to the $\prod_{i=1}^{q} n_i$-dimensional vector formed by $v_1 \otimes v_2 \otimes \cdots \otimes u_q$.

We first give the definition of TensorSketch. Similar definitions can be found in previous work [Pag13, PP13, ANW14, WTSA15].

Definition 21.3.20 (TensorSketch [Pag13])**.** Given q points $v_1, v_2, \cdots, v_q$ where for each $i \in [q], v_i \in \mathbb{R}^{n_i}$, let m be the target dimension. The TensorSketch transform is specified using q 3-wise independent hash functions, $h_1, \cdots, h_q$, where for each $i \in [q]$, $h_i : [n_i] \to [m]$, as well as q 4-wise independent sign functions $s_1, \cdots, s_q$, where for each $i \in [q]$, $s_i : [n_i] \to \{-1, +1\}$.

TensorSketch applied to $v_1, \cdots, v_q$ is then CountSketch applied to $\phi(v_1, \cdots, v_q)$ with hash function $H : [\prod_{i=1}^{q} n_i] \to [m]$ and sign functions $S : [\prod_{i=1}^{q} n_i] \to \{-1, +1\}$ defined as follows:

$$H(i_1, \cdots, i_q) = h_1(i_1) + h_2(s_2) + \cdots + h_q(i_q) \pmod{m},$$

and

$$S(i_1, \cdots, i_q) = s_1(i_1) \cdot s_2(i_2) \cdot \cdots \cdot s_q(i_q).$$

Using the Fast Fourier Transform, TensorSketch$(v_1, \cdots, v_q)$ can be computed in $O(\sum_{i=1}^{q}(\text{nnz}(v_i) + m \log m))$ time.

Note that Theorem 1 in [ANW14] only defines $\phi(v) = v \otimes v \otimes \cdots \otimes v$. Here we state a stronger version of Theorem 1 than in [ANW14], though the proofs are identical; a formal derivation can be found in [DW17].

Theorem 21.3.15 (Generalized version of Theorem 1 in [ANW14]). *Let S be the $(\prod_{i=1}^{q} n_i) \times m$ matrix such that* TensorSketch *$(v_1, v_2, \cdots, v_q)$ is $\phi(v_1, v_2, \cdots, v_q)S$ for a randomly selected* TensorSketch. *The matrix S satisfies the following two properties.*

Property I (Approximate Matrix Product). Let A and B be matrices with $\prod_{i=1}^{q} n_i$ rows. For $m \geq (2 + 3^q)/(\epsilon^2 \delta)$, we have

$$\Pr[\|A^\top S S^\top B - A^\top B\|_F^2 \leq \epsilon^2 \|A\|_F^2 \|B\|_F^2] \geq 1 - \delta.$$

Property II (Subspace Embedding). Consider a fixed k-dimensional subspace V. If $m \geq k^2(2+3^q)/(\epsilon^2\delta)$, then with probability at least $1-\delta$, $\|xS\|_2 = (1 \pm \epsilon)\|x\|_2$ simultaneously for all $x \in V$.

21.4 Frobenius Norm for Arbitrary Tensors

Section 21.4.1 presents a Frobenius norm tensor low-rank approximation algorithm with $(1+\epsilon)$-approximation ratio. Section 21.4.2 introduces a tool which is able to reduce the size of the objective function from n^3 to $\text{poly}(k, 1/\epsilon)$. Section 21.4.3 introduces a new problem called tensor multiple regression. Section 21.4.4 presents several bicriteria algorithms. Section 21.4.5 introduces a powerful tool which we call generalized matrix row subset selection. Section 21.4.6 presents an algorithm that is able to select a batch of columns, rows and tubes from a given tensor, and those samples are also able to form a low-rank solution. Section 21.4.7 presents several useful tools for tensor problems, and also two $(1+\epsilon)$-approximation CURT decomposition algorithms: one has the optimal sample complexity, and the other has the optimal running time. Section 21.4.9 shows how to solve the problem if the size of the objective function is small. Section 21.5 extends several techniques from 3rd order tensors to general q-th order tensors, for any $q \geq 3$. Finally, in Section 21.6 we also provide a new matrix CUR decomposition algorithm, which is faster than [BW14].

For simplicity of presentation, we assume A_k exists in theorems (e.g., Theorem 21.4.1) which concern outputting a rank-k solution, as well as the theorems (e.g., Theorem 21.4.7, Theorem 21.4.8, Theorem 21.4.13) which concern outputting a bicriteria solution (the output rank is larger than k). For each of the bicriteria theorems, we can obtain a more detailed version when A_k does not exist, like Theorem 21.1.1 in Section 21.1 (by instead considering a tensor sufficiently close to A_k in objective function value). Note that the theorems for column, row, tube subset selection Theorem 21.4.20 and Theorem 21.4.21 also belong to this first category. In the second category, for each of the rank-k theorems we can obtain a more detailed version handling all cases, even when A_k does not exist, like Theorem 21.1.2

in Section 21.1 (by instead considering a tensor sufficiently close to A_k in objective function value).

Several other tensor results or tools (e.g., Theorem 21.4.4, Lemma 21.4.3, Theorem 21.4.39, Theorem 21.4.40, Theorem 21.4.14, Theorem 21.5.1) that we build in this section do not belong to the above two categories. It means those results do not depend on whether A_k exists or not and whether OPT is zero or not.

21.4.1 $(1+\epsilon)$-approximate low-rank approximation

Algorithm 21.2 Frobenius Norm Low-rank Approximation

procedure FLowRankApprox(A, n, k, ϵ) ▷ Theorem 21.4.1
  $s_1 \leftarrow s_2 \leftarrow s_3 \leftarrow O(k/\epsilon)$.
  Choose sketching matrices $S_1 \in \mathbb{R}^{n^2 \times s_1}$, $S_2 \in \mathbb{R}^{n^2 \times s_2}$, $S_3 \in \mathbb{R}^{n^2 \times s_3}$. ▷ Definition 21.3.12
  Compute $A_i S_i, \forall i \in [3]$.
  $Y_1, Y_2, Y_3, C \leftarrow$FInputSparsityReduction$(A, A_1S_1, A_2S_2, A_3S_3, n, s_1, s_2, s_3, k, \epsilon)$. ▷ Algorithm 21.3
  Create variables for $X_i \in \mathbb{R}^{s_i \times k}, \forall i \in [3]$.
  Run polynomial system verifier for $\|(Y_1X_1) \otimes (Y_2X_2) \otimes (Y_3X_3) - C\|_F^2$.
  return $A_1S_1X_1$, $A_2S_2X_2$, and $A_3S_3X_3$.
end procedure

Theorem 21.4.1. *Given a 3rd order tensor $A \in \mathbb{R}^{n \times n \times n}$, for any $k \geq 1, \epsilon \in (0,1)$, there exists an algorithm which takes $O(\mathrm{nnz}(A)) + n\,\mathrm{poly}(k, 1/\epsilon) + 2^{O(k^2/\epsilon)}$ time and outputs three matrices $U \in \mathbb{R}^{n \times k}$, $V \in \mathbb{R}^{n \times k}$, $W \in \mathbb{R}^{n \times k}$ such that*

$$\left\| \sum_{i=1}^{k} U_i \otimes V_i \otimes W_i - A \right\|_F^2 \leq (1+\epsilon) \min_{\mathrm{rank}-k\ A_k} \|A_k - A\|_F^2$$

holds with probability $9/10$.

Proof. Given any tensor $A \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, we define three matrices $A_1 \in \mathbb{R}^{n_1 \times n_2 n_3}, A_2 \in \mathbb{R}^{n_2 \times n_3 n_1}, A_3 \in \mathbb{R}^{n_3 \times n_1 n_2}$ such that, for any $i \in [n_1], j \in [n_2], l \in [n_3]$,

$$A_{i,j,l} = (A_1)_{i,(j-1)\cdot n_3 + l} = (A_2)_{j,(l-1)\cdot n_1 + i} = (A_3)_{l,(i-1)\cdot n_2 + j}.$$

We define OPT as

$$\mathrm{OPT} = \min_{\mathrm{rank}-k\ A'} \|A' - A\|_F^2.$$

Suppose the optimal $A_k = U^* \otimes V^* \otimes W^*$. We fix $V^* \in \mathbb{R}^{n\times k}$ and $W^* \in \mathbb{R}^{n\times k}$. We use $V_1^*, V_2^*, \cdots, V_k^*$ to denote the columns of V^* and $W_1^*, W_2^*, \cdots, W_k^*$ to denote the columns of W^*.

We consider the following optimization problem,

$$\min_{U_1,\cdots,U_k\in\mathbb{R}^n} \left\| \sum_{i=1}^k U_i \otimes V_i^* \otimes W_i^* - A \right\|_F^2,$$

which is equivalent to

$$\min_{U_1,\cdots,U_k\in\mathbb{R}^n} \left\| \begin{bmatrix} U_1 & U_2 & \cdots & U_k \end{bmatrix} \begin{bmatrix} V_1^* \otimes W_1^* \\ V_2^* \otimes W_2^* \\ \cdots \\ V_k^* \otimes W_k^* \end{bmatrix} - A \right\|_F^2.$$

We use matrix Z_1 to denote $\begin{bmatrix} \mathrm{vec}(V_1^* \otimes W_1^*) \\ \mathrm{vec}(V_2^* \otimes W_2^*) \\ \cdots \\ \mathrm{vec}(V_k^* \otimes W_k^*) \end{bmatrix} \in \mathbb{R}^{k\times n^2}$ and matrix U to denote $\begin{bmatrix} U_1 & U_2 & \cdots & U_k \end{bmatrix}$. Then we can obtain the following equivalent objective function,

$$\min_{U\in\mathbb{R}^{n\times k}} \|UZ_1 - A_1\|_F^2.$$

Notice that $\min_{U\in\mathbb{R}^{n\times k}} \|UZ_1 - A_1\|_F^2 = \mathrm{OPT}$, since $A_k = U^* Z_1$.

Let $S_1^\top \in \mathbb{R}^{s_1\times n^2}$ be a sketching matrix defined in Definition 21.3.12, where $s_1 = O(k/\epsilon)$. We obtain the following optimization problem,

$$\min_{U\in\mathbb{R}^{n\times k}} \|UZ_1S_1 - A_1S_1\|_F^2.$$

Let $\widehat{U} \in \mathbb{R}^{n\times k}$ denote the optimal solution to the above optimization problem. Then $\widehat{U} = A_1S_1(Z_1S_1)^\dagger$. By Lemma 21.3.10 and Theorem 21.3.11, we have

$$\|\widehat{U}Z_1 - A_1\|_F^2 \leq (1+\epsilon) \min_{U \in \mathbb{R}^{n \times k}} \|UZ_1 - A_1\|_F^2 = (1+\epsilon)\,\mathrm{OPT},$$

which implies

$$\left\| \sum_{i=1}^{k} \widehat{U}_i \otimes V_i^* \otimes W_i^* - A \right\|_F^2 \leq (1+\epsilon)\,\mathrm{OPT}\,.$$

To write down $\widehat{U}_1, \cdots, \widehat{U}_k$, we use the given matrix A_1, and we create $s_1 \times k$ variables for matrix $(Z_1 S_1)^\dagger$.

As our second step, we fix $\widehat{U} \in \mathbb{R}^{n \times k}$ and $W^* \in \mathbb{R}^{n \times k}$, and we convert tensor A into matrix A_2. Let matrix Z_2 denote $\begin{bmatrix} \mathrm{vec}(\widehat{U}_1 \otimes W_1^*) \\ \mathrm{vec}(\widehat{U}_2 \otimes W_2^*) \\ \cdots \\ \mathrm{vec}(\widehat{U}_k \otimes W_k^*) \end{bmatrix}$. We consider the following objective function,

$$\min_{V \in \mathbb{R}^{n \times k}} \|VZ_2 - A_2\|_F^2,$$

for which the optimal cost is at most $(1+\epsilon)\,\mathrm{OPT}$.

Let $S_2^\top \in \mathbb{R}^{s_2 \times n^2}$ be a sketching matrix defined in Definition 21.3.12, where $s_2 = O(k/\epsilon)$. We sketch S_2 on the right of the objective function to obtain the new objective function,

$$\min_{V \in \mathbb{R}^{n \times k}} \|VZ_2S_2 - A_2S_2\|_F^2.$$

Let $\widehat{V} \in \mathbb{R}^{n \times k}$ denote the optimal solution of the above problem. Then $\widehat{V} = A_2S_2(Z_2S_2)^\dagger$. By Lemma 21.3.10 and Theorem 21.3.11, we have,

$$\|\widehat{V}Z_2 - A_2\|_F^2 \leq (1+\epsilon) \min_{V \in \mathbb{R}^{n \times k}} \|VZ_2 - A_2\|_F^2 \leq (1+\epsilon)^2\,\mathrm{OPT},$$

which implies

$$\left\|\sum_{i=1}^{k} \widehat{U}_i \otimes \widehat{V}_i \otimes W_i^* - A\right\|_F^2 \leq (1+\epsilon)^2 \operatorname{OPT}.$$

To write down $\widehat{V}_1, \cdots, \widehat{V}_k$, we need to use the given matrix $A_2 \in \mathbb{R}^{n^2 \times n}$, and we need to create $s_2 \times k$ variables for matrix $(Z_2 S_2)^\dagger$.

As our third step, we fix the matrices $\widehat{U} \in \mathbb{R}^{n \times k}$ and $\widehat{V} \in \mathbb{R}^{n \times k}$. We convert tensor $A \in \mathbb{R}^{n \times n \times n}$ into matrix $A_3 \in \mathbb{R}^{n^2 \times n}$. Let matrix Z_3 denote $\begin{bmatrix} \operatorname{vec}(\widehat{U}_1 \otimes \widehat{V}_1) \\ \operatorname{vec}(\widehat{U}_2 \otimes \widehat{V}_2) \\ \cdots \\ \operatorname{vec}(\widehat{U}_k \otimes \widehat{V}_k) \end{bmatrix}$. We consider the following objective function,

$$\min_{W \in \mathbb{R}^{n \times k}} \|W Z_3 - A_3\|_F^2,$$

which has optimal cost at most $(1+\epsilon)^2 \operatorname{OPT}$.

Let $S_3^\top \in \mathbb{R}^{s_3 \times n^2}$ be a sketching matrix defined in Definition 21.3.12, where $s_3 = O(k/\epsilon)$. We sketch S_3 on the right of the objective function to obtain a new objective function,

$$\min_{W \in \mathbb{R}^{n \times k}} \|W Z_3 S_3 - A_3 S_3\|_F^2.$$

Let $\widehat{W} \in \mathbb{R}^{n \times k}$ denote the optimal solution of the above problem. Then $\widehat{W} = A_3 S_3 (Z_3 S_3)^\dagger$. By Lemma 21.3.10 and Theorem 21.3.11, we have,

$$\|\widehat{W} Z_3 - A_3\|_F^2 \leq (1+\epsilon) \min_{W \in \mathbb{R}^{n \times k}} \|W Z_3 - A_3\|_F^2 \leq (1+\epsilon)^3 \operatorname{OPT}.$$

Thus, we have

$$\min_{X_1, X_2, X_3} \left\|\sum_{i=1}^{k} (A_1 S_1 X_1)_i \otimes (A_2 S_2 X_2)_i \otimes (A_3 S_3 X_3)_i - A\right\|_F^2 \leq (1+\epsilon)^3 \operatorname{OPT}.$$

Let $V_1 = A_1 S_1, V_2 = A_2 S_2, V_3 = A_3 S_3$, we then apply Lemma 21.4.3, and we obtain $\widehat{V}_1, \widehat{V}_2, \widehat{V}_3, C$. We then apply Theorem 21.4.44. Correctness follows by rescaling ϵ by a constant factor.

Running time. Due to Definition 21.3.12, the running time of line 4 is $O(\mathrm{nnz}(A)) + n\,\mathrm{poly}(k)$. The running time of line 5 is shown by Lemma 21.4.3, and the running time of line 7 is shown by Theorem 21.4.44. □

Theorem 21.4.2. *Suppose we are given a 3rd order $n \times n \times n$ tensor A such that each entry can be written using n^δ bits, where $\delta > 0$ is a given, value which can be arbitrarily small (e.g., we could have n^δ being $O(\log n)$). Define $\mathrm{OPT} = \inf_{\mathrm{rank}-k\ A_k} \|A_k - A\|_F^2$. For any $k \geq 1$, and for any $0 < \epsilon < 1$, define $n^{\delta'} = O(n^\delta 2^{O(k^2/\epsilon)})$. (I) If $\mathrm{OPT} > 0$, and there exists a rank-k $A_k = U^* \otimes V^* \otimes W^*$ tensor, with size $n \times n \times n$, such that $\|A_k - A\|_F^2 = \mathrm{OPT}$, and $\max(\|U^*\|_F, \|V^*\|_F, \|W^*\|_F) \leq 2^{O(n^{\delta'})}$, then there exists an algorithm that takes $(\mathrm{nnz}(A) + n\,\mathrm{poly}(k, 1/\epsilon) + 2^{O(k^2/\epsilon)})n^\delta$ time in the unit cost* RAM *model with word size $O(\log n)$ bits*[9] *and outputs three $n \times k$ matrices U, V, W such that*

$$\|U \otimes V \otimes W - A\|_F^2 \leq (1+\epsilon)\,\mathrm{OPT} \tag{21.5}$$

holds with probability $9/10$, and each entry of each of U, V, W fits in $n^{\delta'}$ bits.

(II) *If $\mathrm{OPT} > 0$, and A_k does not exist, and there exist three $n \times k$ matrices U', V', W' for which $\max(\|U'\|_F, \|V'\|_F, \|W'\|_F) \leq 2^{O(n^{\delta'})}$ and $\|U' \otimes V' \otimes W' - A\|_F^2 \leq (1 + \epsilon/2)\,\mathrm{OPT}$, then we can find U, V, W such that (21.5) holds.*

[9] The entries of A are assumed to fit in n^δ words.

(III) *If* $\mathrm{OPT} = 0$ *and* A_k *does exist, and there exists a solution* U^*, V^*, W^* *such that each entry can be written by* $n^{\delta'}$ *bits, then we can obtain* (21.5).

(IV) *If* $\mathrm{OPT} = 0$*, and there exist three* $n \times k$ *matrices* U, V, W *such that*

$$\max(\|U\|_F, \|V\|_F, \|W\|_F) \leq 2^{O(n^{\delta'})}$$

and

$$\|U \otimes V \otimes W - A\|_F^2 \leq (1+\epsilon)\,\mathrm{OPT} + 2^{-\Omega(n^{\delta'})} = 2^{-\Omega(n^{\delta'})}, \tag{21.6}$$

then we can output U, V, W *such that* (21.6) *holds.*

Further if A_k *exists, we can output a number* Z *for which* $\mathrm{OPT} \leq Z \leq (1+\epsilon)\,\mathrm{OPT}$*. For all the cases above, the algorithm runs in the same time as (I) and succeeds with probability at least* $9/10$*.*

Proof. This follows by the discussion in Section 21.1, Theorem 21.4.1 and Theorem 21.4.44 in Section 21.4.9.

Part (I) Suppose $\delta > 0$ and $A_k = U^* \otimes V^* \otimes W^*$ exists and each of $\|U^*\|_F$, $\|V^*\|_F$, and $\|W^*\|_F$ is bounded by $2^{O(n^{\delta'})}$. We assume the computation model is the unit cost RAM model with words of size $O(\log n)$ bits, and allow each number of the input tensor A to be written using n^{δ} bits. For the case when OPT is nonzero, using the proof of Theorem 21.4.1 and Theorems 21.4.44, 21.3.2, there exists a lower bound on the cost OPT, which is at least $2^{-O(n^{\delta})2^{O(k^2/\epsilon)}}$. We can round each entry of matrices U^*, V^*, W^* to be an integer expressed using $O(n^{\delta'})$ bits to obtain U', V', W'. Using the triangle inequality and our lower bound on OPT, it follows that U', V', W' provide a $(1+\epsilon)$-approximation.

Thus, applying Theorem 21.4.1 by fixing U', V', W' and using Theorem 21.4.44 at the end, we can output three matrices U, V, W, where each entry can be written using $n^{\delta'}$ bits, so that we satisfy $\|U \otimes V \otimes W - A\|_F^2 \leq (1+\epsilon)\,\mathrm{OPT}$.

For the running time, since each entry of the input is bounded by n^{δ} bits, due to Theorem 21.4.1, we need $(\mathrm{nnz}(A) + n\,\mathrm{poly}(k/\epsilon)) \cdot n^{\delta}$ time to reduce the size of the problem to $\mathrm{poly}(k/\epsilon)$ size (with each number represented using $O(n^{\delta})$ bits). According to Theorem 21.4.44, the running time of using a polynomial system verifier to get the solution is $2^{O(k^2/\epsilon)} n^{O(\delta')} = 2^{O(k^2/\epsilon)} n^{O(\delta)}$ time. Thus the total running time is $(\mathrm{nnz}(A) + n\,\mathrm{poly}(k/\epsilon))n^{\delta} + 2^{O(k^2/\epsilon)} \cdot n^{O(\delta)}$.

Part (II) is similar to Part (I). Part (III) is trivial to prove since there exists a solution which can be written down in the bit model, so we obtain a $(1+\epsilon)$-approximation. Part (IV) is also very similar to Part (II).

□

21.4.2 Input sparsity reduction

Algorithm 21.3 Reducing the Size of the Objective Function from poly(n) to poly(k)

procedure FInputSparsityReduction$(A, V_1, V_2, V_3, n, b_1, b_2, b_3, k, \epsilon)$ ▷ Lemma 21.4.3
  $c_1 \leftarrow c_2 \leftarrow c_3 \leftarrow \text{poly}(k, 1/\epsilon)$.
  Choose sparse embedding matrices $T_1 \in \mathbb{R}^{c_1 \times n}$, $T_2 \in \mathbb{R}^{c_2 \times n}$, $T_3 \in \mathbb{R}^{c_3 \times n}$. ▷ Definition 21.3.10
  $\widehat{V}_i \leftarrow T_i V_i \in \mathbb{R}^{c_i \times b_i}, \forall i \in [3]$.
  $C \leftarrow A(T_1, T_2, T_3) \in \mathbb{R}^{c_1 \times c_2 \times c_3}$.
  return $\widehat{V}_1$, $\widehat{V}_2$, $\widehat{V}_3$ and C.
end procedure

Lemma 21.4.3. *Let* $\text{poly}(k, 1/\epsilon) \geq b_1 b_2 b_3 \geq k$. *Given a tensor* $A \in \mathbb{R}^{n \times n \times n}$ *and three matrices* $V_1 \in \mathbb{R}^{n \times b_1}$, $V_2 \in \mathbb{R}^{n \times b_2}$, *and* $V_3 \in \mathbb{R}^{n \times b_3}$, *there exists an algorithm that takes* $O(\text{nnz}(A) + \text{nnz}(V_1) + \text{nnz}(V_2) + \text{nnz}(V_3)) = O(\text{nnz}(A) + n\,\text{poly}(k/\epsilon))$ *time and outputs a tensor* $C \in \mathbb{R}^{c_1 \times c_2 \times c_3}$ *and three matrices* $\widehat{V}_1 \in \mathbb{R}^{c_1 \times b_1}$, $\widehat{V}_2 \in \mathbb{R}^{c_2 \times b_2}$ *and* $\widehat{V}_3 \in \mathbb{R}^{c_3 \times b_3}$ *with* $c_1 = c_2 = c_3 = \text{poly}(k, 1/\epsilon)$, *such that with probability at least* 0.99, *for all* $\alpha > 0, X_1, X_1' \in \mathbb{R}^{b_1 \times k}, X_2, X_2' \in \mathbb{R}^{b_2 \times k}, X_3, X_3' \in \mathbb{R}^{b_3 \times k}$ *satisfy that,*

$$\left\| \sum_{i=1}^k (\widehat{V}_1 X_1')_i \otimes (\widehat{V}_2 X_2')_i \otimes (\widehat{V}_3 X_3')_i - C \right\|_F^2 \leq \alpha \left\| \sum_{i=1}^k (\widehat{V}_1 X_1)_i \otimes (\widehat{V}_2 X_2)_i \otimes (\widehat{V}_3 X_3)_i - C \right\|_F^2,$$

then,

$$\left\| \sum_{i=1}^k (V_1 X_1')_i \otimes (V_2 X_2')_i \otimes (V_3 X_3')_i - A \right\|_F^2 \leq (1+\epsilon)\alpha \left\| \sum_{i=1}^k (V_1 X_1)_i \otimes (V_2 X_2)_i \otimes (V_3 X_3)_i - A \right\|_F^2.$$

Proof. Let $X_1 \in \mathbb{R}^{b_1 \times k}, X_2 \in \mathbb{R}^{b_2 \times k}, X_3 \in \mathbb{R}^{b_3 \times k}$. First, we define $Z_1 = ((V_2 X_2)^\top \odot (V_3 X_3)^\top) \in \mathbb{R}^{k \times n^2}$. (Note that, for each $i \in [k]$, the i-th row of matrix Z_1 is $\text{vec}((V_2 X_2)_i \otimes (V_3 X_3)_i)$.) Then, by flattening we have

$$\left\| \sum_{i=1}^k (V_1 X_1)_i \otimes (V_2 X_2)_i \otimes (V_3 X_3)_i - A \right\|_F^2 = \|V_1 X_1 \cdot Z_1 - A_1\|_F^2.$$

We choose a sparse embedding matrix (Definition 21.3.10) $T_1 \in \mathbb{R}^{c_1 \times n}$ with $c_1 = \text{poly}(k, 1/\epsilon)$ rows. Since V_1 has $b_1 \leq \text{poly}(k/\epsilon)$ columns, according to Lemma 21.3.7 with probability 0.999, for all $X_1 \in \mathbb{R}^{b_1 \times k}, Z \in \mathbb{R}^{k \times n^2}$,

$$(1-\epsilon)\|V_1 X_1 Z - A_1\|_F^2 \leq \|T_1 V_1 X_1 Z - T_1 A_1\|_F^2 \leq (1+\epsilon)\|V_1 X_1 Z - A_1\|_F^2.$$

Therefore, we have

$$\|T_1 V_1 X_1 \cdot Z_1 - T_1 A_1\|_F^2 = (1 \pm \epsilon) \left\| \sum_{i=1}^k (V_1 X_1)_i \otimes (V_2 X_2)_i \otimes (V_3 X_3)_i - A \right\|_F^2.$$

Second, we unflatten matrix $T_1 A_1 \in \mathbb{R}^{c_1 \times n^2}$ to obtain a tensor $A' \in \mathbb{R}^{c_1 \times n \times n}$. Then we flatten A' along the second direction to obtain $A_2 \in \mathbb{R}^{n \times c_1 n}$. We define $Z_2 = (T_1 V_1 X_1)^\top \odot (V_3 X_3)^\top \in \mathbb{R}^{k \times c_1 n}$. Then, by flattening,

$$\begin{aligned} \|V_2 X_2 \cdot Z_2 - A_2\|_F^2 &= \|T_1 V_1 X_1 \cdot Z_1 - T_1 A_1\|_F^2 \\ &= (1 \pm \epsilon) \left\| \sum_{i=1}^k (V_1 X_1)_i \otimes (V_2 X_2)_i \otimes (V_3 X_3)_i - A \right\|_F^2. \end{aligned}$$

We choose a sparse embedding matrix (Definition 21.3.10) $T_2 \in \mathbb{R}^{c_2 \times n}$ with $c_2 = \text{poly}(k, 1/\epsilon)$ rows. Then according to Lemma 21.3.7 with probability 0.999, for all $X_2 \in \mathbb{R}^{b_2 \times k}$, $Z \in \mathbb{R}^{k \times c_1 n}$,

$$(1-\epsilon)\|V_2 X_2 Z - A_2\|_F^2 \leq \|T_2 V_2 X_2 Z - T_2 A_2\|_F^2 \leq (1+\epsilon)\|V_2 X_2 Z - A_2\|_F^2.$$

Therefore, we have

$$\begin{aligned} \|T_2 V_2 X_2 \cdot Z_2 - T_2 A_2\|_F^2 &= (1 \pm \epsilon)\|V_2 X_2 \cdot Z_2 - A_2\|_F^2 \\ &= (1 \pm \epsilon)^2 \left\| \sum_{i=1}^k (V_1 X_1)_i \otimes (V_2 X_2)_i \otimes (V_3 X_3)_i - A \right\|_F^2. \end{aligned}$$

Third, we unflatten matrix $T_2A_2 \in \mathbb{R}^{c_2 \times c_1 n}$ to obtain a tensor $A''(= A(T_1, T_2, I)) \in \mathbb{R}^{c_1 \times c_2 \times n}$. Then we flatten tensor A'' along the last direction (the third direction) to obtain matrix $A_3 \in \mathbb{R}^{n \times c_1 c_2}$. We define $Z_3 = (T_1V_1X_1)^\top \odot (T_2V_2X_2)^\top \in \mathbb{R}^{k \times c_1 c_2}$. Then, by flattening, we have

$$\begin{aligned}\|V_3X_3 \cdot Z_3 - A_3\|_F^2 = & \|T_2V_2X_2 \cdot Z_2 - T_2A_2\|_F^2 \\ = & (1 \pm \epsilon)^2 \left\| \sum_{i=1}^k (V_1X_1)_i \otimes (V_2X_2)_i \otimes (V_3X_3)_i - A \right\|_F^2.\end{aligned}$$

We choose a sparse embedding matrix (Definition 21.3.10) $T_3 \in \mathbb{R}^{c_3 \times n}$ with $c_3 = \mathrm{poly}(k, 1/\epsilon)$ rows. Then according to Lemma 21.3.7 with probability 0.999, for all $X_3 \in \mathbb{R}^{b_3 \times k}$, $Z \in \mathbb{R}^{k \times c_1 c_2}$,

$$(1-\epsilon)\|V_3X_3Z - A_3\|_F^2 \leq \|T_3V_3X_3Z - T_3A_3\|_F^2 \leq (1+\epsilon)\|V_3X_3Z - A_3\|_F^2.$$

Therefore, we have

$$\|T_3V_3X_3 \cdot Z_3 - T_3A_3\|_F^2 = (1 \pm \epsilon)^3 \left\| \sum_{i=1}^k (V_1X_1)_i \otimes (V_2X_2)_i \otimes (V_3X_3)_i - A \right\|_F^2.$$

Note that

$$\|T_3V_3X_3 \cdot Z_3 - T_3A_3\|_F^2 = \left\| \sum_{i=1}^k (T_1V_1X_1)_i \otimes (T_2V_2X_2)_i \otimes (T_3V_3X_3)_i - A(T_1, T_2, T_3) \right\|_F^2,$$

and thus, we have $\forall X_1 \in \mathbb{R}^{b_1 \times k}, X_2 \in \mathbb{R}^{b_2 \times k}, X_3 \in \mathbb{R}^{b_3 \times k}$

$$\begin{aligned}& \left\| \sum_{i=1}^k (T_1V_1X_1)_i \otimes (T_2V_2X_2)_i \otimes (T_3V_3X_3)_i - A(T_1, T_2, T_3) \right\|_F^2 \\ = & (1 \pm \epsilon)^3 \left\| \sum_{i=1}^k (V_1X_1)_i \otimes (V_2X_2)_i \otimes (V_3X_3)_i - A \right\|_F^2.\end{aligned}$$

Let $\widehat{V}_i$ denote $T_i V_i$, for each $i \in [3]$. Let $C \in \mathbb{R}^{c_1 \times c_2 \times c_3}$ denote $A(T_1, T_2, T_3)$. For $\alpha > 1$, if

$$\left\| \sum_{i=1}^k (\widehat{V}_1 X_1')_i \otimes (\widehat{V}_2 X_2')_i \otimes (\widehat{V}_3 X_3')_i - C \right\|_F^2 \leq \alpha \left\| \sum_{i=1}^k (\widehat{V}_1 X_1)_i \otimes (\widehat{V}_2 X_2)_i \otimes (\widehat{V}_3 X_3)_i - C \right\|_F^2,$$

then

$$\begin{aligned}
& \left\| \sum_{i=1}^k (V_1 X_1')_i \otimes (V_2 X_2')_i \otimes (V_3 X_3')_i - C \right\|_F^2 \\
\leq & \frac{1}{(1-\epsilon)^3} \left\| \sum_{i=1}^k (\widehat{V}_1 X_1')_i \otimes (\widehat{V}_2 X_2')_i \otimes (\widehat{V}_3 X_3')_i - C \right\|_F^2 \\
\leq & \frac{1}{(1-\epsilon)^3} \alpha \left\| \sum_{i=1}^k (\widehat{V}_1 X_1)_i \otimes (\widehat{V}_2 X_2)_i \otimes (\widehat{V}_3 X_3)_i - C \right\|_F^2 \\
\leq & \frac{(1+\epsilon)^3}{(1-\epsilon)^3} \alpha \left\| \sum_{i=1}^k (V_1 X_1)_i \otimes (V_2 X_2)_i \otimes (V_3 X_3)_i - C \right\|_F^2
\end{aligned}$$

By rescaling ϵ by a constant, we complete the proof of correctness.

Running time. According to Section 21.3.6, for each $i \in [3]$, $T_i V_i$ can be computed in $O(\mathrm{nnz}(V_i))$ time, and $A(T_1, T_2, T_3)$ can be computed in $O(\mathrm{nnz}(A))$ time.

By the analysis above, the proof is complete. □

21.4.3 Tensor multiple regression

Algorithm 21.4 Frobenius Norm Tensor Multiple Regression

procedure FTensorMultipleRegression(A, U, V, d, n, k) ▷ Theorem 21.4.4
  $s \leftarrow O(k^2 + k/\epsilon)$.
  Choose $S \in \mathbb{R}^{n^2 \times s}$ to be a TensorSketch. ▷ Definition 21.3.20
  Compute $A \cdot S$.
  Compute $B \cdot S$. ▷ $B = U^\top \odot V^\top$
  $W \leftarrow (AS)(BS)^\dagger$
  return W.
end procedure

Theorem 21.4.4. *Given matrices $A \in \mathbb{R}^{d \times n^2}$, $U, V \in \mathbb{R}^{n \times k}$, let $B \in \mathbb{R}^{k \times n^2}$ denote $U^\top \odot V^\top$. There exists an algorithm that takes $O(\mathrm{nnz}(A) + \mathrm{nnz}(U) + \mathrm{nnz}(V) + d\,\mathrm{poly}(k, 1/\epsilon))$ time and outputs a matrix $W' \in \mathbb{R}^{d \times k}$ such that,*

$$\|W'B - A\|_F^2 \leq (1+\epsilon) \min_{W \in \mathbb{R}^{d \times k}} \|WB - A\|_F^2.$$

Proof. We choose a TensorSketch (Definition 21.3.20) $S \in \mathbb{R}^{n^2 \times s}$ to reduce the problem to a smaller problem,

$$\min_{W \in \mathbb{R}^{d \times k}} \|WBS - AS\|_F^2.$$

Let W' denote the optimal solution to the above problem. Following a similar proof to that in Section 21.4.7.3, if S is a $(1 \pm 1/2)$-subspace embedding and satisfies $\sqrt{\epsilon/k}$-approximate matrix product, then W' provides a $(1+\epsilon)$-approximation to the original problem. By Theorem 21.3.15, we have $s = O(k^2 + k/\epsilon)$.

Running time. According to Definition 21.3.20, BS can be computed in $O(\text{nnz}(U) + \text{nnz}(V)) + \text{poly}(k/\epsilon)$ time. Notice that each row of S has exactly 1 nonzero entry, thus AS can be computed in $O(\text{nnz}(A))$ time. Since $BS \in \mathbb{R}^{k \times s}$ and $AS \in \mathbb{R}^{d \times s}$, $\min_{W \in \mathbb{R}^{d \times k}} \|WBS - AS\|_F^2$ can be solved in $d\,\text{poly}(sk) = d\,\text{poly}(k/\epsilon)$ time. □

21.4.4 Bicriteria algorithms

21.4.4.1 Solving a small regression problem

Lemma 21.4.5. *Given tensor $A \in \mathbb{R}^{n\times n\times n}$ and three matrices $U \in \mathbb{R}^{n\times s_1}, V \in \mathbb{R}^{n\times s_2}$ and $W \in \mathbb{R}^{n\times s_3}$, there exists an algorithm that takes $O(\mathrm{nnz}(A) + n\,\mathrm{poly}(s_1, s_2, s_3, 1/\epsilon))$ time and outputs $\alpha' \in \mathbb{R}^{s_1\times s_2\times s_3}$ such that*

$$\left\| \sum_{i=1}^{s_1}\sum_{j=1}^{s_2}\sum_{l=1}^{s_3} \alpha'_{i,j,l} \cdot U_i \otimes V_j \otimes W_l - A \right\|_F^2$$
$$\leq (1+\epsilon) \min_{\alpha\in\mathbb{R}^{s_1\times s_2\times s_3}} \left\| \sum_{i=1}^{s_1}\sum_{j=1}^{s_2}\sum_{l=1}^{s_3} \alpha_{i,j,l} \cdot U_i \otimes V_j \otimes W_l - A \right\|_F^2.$$

holds with probability at least .99.

Proof. We define $\widetilde{b} \in \mathbb{R}^{n^3}$ to be the vector where the $i+(j-1)n+(l-1)n^2$-th entry of $\widetilde{b}$ is $A_{i,j,l}$. We define $\widetilde{A} \in \mathbb{R}^{n^3\times s_1 s_2 s_3}$ to be the matrix where the $(i+(j-1)n+(l-1)n^2, i'+(j'-1)s_2+(l'-1)s_2s_3)$ entry is $U_{i',i} \cdot V_{j',j} \cdot W_{l',l}$. This problem is equivalent to a linear regression problem,

$$\min_{x\in\mathbb{R}^{s_1 s_2 s_3}} \|\widetilde{A}x - \widetilde{b}\|_2^2,$$

where $\widetilde{A} \in \mathbb{R}^{n^3\times s_1 s_2 s_3}, \widetilde{b} \in \mathbb{R}^{n^3}$. Thus, it can be solved fairly quickly using recent work [CW13, MM13, NN13a]. However, the running time of this naïvely is $\Omega(n^3)$, since we have to write down each entry of $\widetilde{A}$. In the next few paragraphs, we show how to improve the running time to $\mathrm{nnz}(A) + n\,\mathrm{poly}(s_1, s_2, s_3)$.

Since $\alpha \in \mathbb{R}^{s_1\times s_2\times s_3}$, α can be always written as $\alpha = X_1 \otimes X_2 \otimes X_3$, where $X_1 \in$

$\mathbb{R}^{s_1 \times s_1 s_2 s_3}, X_2 \in \mathbb{R}^{s_2 \times s_1 s_2 s_3}, X_3 \in \mathbb{R}^{s_3 \times s_1 s_2 s_3}$, we have

$$\begin{aligned}
&\min_{\alpha \in \mathbb{R}^{s_1 \times s_2 \times s_3}} \left\| \sum_{i=1}^{s_1} \sum_{j=1}^{s_2} \sum_{l=1}^{s_3} \alpha_{i,j,l} \cdot U_i \otimes V_j \otimes W_l - A \right\|_F^2 \\
= &\min_{\substack{X_1 \in \mathbb{R}^{s_1 \times s_1 s_2 s_3} \\ X_2 \in \mathbb{R}^{s_2 \times s_1 s_2 s_3} \\ X_3 \in \mathbb{R}^{s_3 \times s_1 s_2 s_3}}} \|(UX_1) \otimes (VX_2) \otimes (WX_3) - A\|_F^2 .
\end{aligned}$$

By Lemma 21.4.3, we can reduce the problem size $n \times n \times n$ to a smaller problem that has size $t_1 \times t_2 \times t_3$,

$$\min_{X_1, X_2, X_3} \left\| \sum_{i=1}^{s_1 s_2 s_3} (T_1 U X_1)_i \otimes (T_2 V X_2)_i \otimes (T_3 W X_3)_i - A(T_1, T_2, T_3) \right\|_F^2$$

where $T_1 \in \mathbb{R}^{t_1 \times n}$, $T_2 \in \mathbb{R}^{t_2 \times n}$, $T_3 \in \mathbb{R}^{t_3 \times n}, t_1 = t_2 = t_3 = \mathrm{poly}(s_1 s_2 s_3 / \epsilon)$. Notice that

$$\begin{aligned}
&\min_{X_1, X_2, X_3} \left\| \sum_{i=1}^{s_1 s_2 s_3} (T_1 U X_1)_i \otimes (T_2 V X_2)_i \otimes (T_3 W X_3)_i - A(T_1, T_2, T_3) \right\|_F^2 \\
= &\min_{\alpha \in \mathbb{R}^{s_1 \times s_2 \times s_3}} \left\| \sum_{i=1}^{s_1} \sum_{j=1}^{s_2} \sum_{l=1}^{s_3} \alpha_{i,j,l} \cdot (T_1 U)_i \otimes (T_2 V)_j \otimes (T_3 W)_l - A(T_1, T_2, T_3) \right\|_F^2 .
\end{aligned}$$

Let

$$\alpha' = \arg\min_{\alpha \in \mathbb{R}^{s_1 \times s_2 \times s_3}} \left\| \sum_{i=1}^{s_1} \sum_{j=1}^{s_2} \sum_{l=1}^{s_3} \alpha_{i,j,l} \cdot (T_1 U)_i \otimes (T_2 V)_j \otimes (T_3 W)_l - A(T_1, T_2, T_3) \right\|_F^2 ,$$

then we have

$$\begin{aligned}
&\left\| \sum_{i=1}^{s_1} \sum_{j=1}^{s_2} \sum_{l=1}^{s_3} \alpha'_{i,j,l} \cdot U_i \otimes V_j \otimes W_l - A \right\|_F^2 \\
\leq &(1+\epsilon) \min_{\alpha \in \mathbb{R}^{s_1 \times s_2 \times s_3}} \left\| \sum_{i=1}^{s_1} \sum_{j=1}^{s_2} \sum_{l=1}^{s_3} \alpha_{i,j,l} \cdot U_i \otimes V_j \otimes W_l - A \right\|_F^2 .
\end{aligned}$$

Again, according to Lemma 21.4.3, the total running time is then $O(\text{nnz}(A)+n\,\text{poly}(s_1,s_2,s_3,1/\epsilon))$.

□

Lemma 21.4.6. *Given tensor $A \in \mathbb{R}^{n\times n\times n}$, and two matrices $U \in \mathbb{R}^{n\times s}, V \in \mathbb{R}^{n\times s}$ with $\text{rank}(U) = r_1, \text{rank}(V) = r_2$, let $T_1 \in \mathbb{R}^{t_1\times n}, T_2 \in \mathbb{R}^{t_2\times n}$ be two sparse embedding matrices (Definition 21.3.10) with $t_1 = \text{poly}(r_1/\epsilon), t_2 = \text{poly}(r_2/\epsilon)$. Then with probability at least 0.99, $\forall X \in \mathbb{R}^{n\times s}$,*

$$(1-\epsilon)\|U\otimes V\otimes X - A\|_F^2 \leq \|T_1U\otimes T_2V\otimes X - A(T_1,T_2,I)\|_F^2 \leq (1+\epsilon)\|U\otimes V\otimes X - A\|_F^2.$$

Proof. Let $X \in \mathbb{R}^{n\times s}$. We define $Z_1 = (V^\top \odot X^\top) \in \mathbb{R}^{s\times n^2}$. We choose a sparse embedding matrix (Definition 21.3.10) $T_1 \in \mathbb{R}^{t_1\times n}$ with $t_1 = \text{poly}(r_1/\epsilon)$ rows. According to Lemma 21.3.7 with probability 0.999, for all $Z \in \mathbb{R}^{s\times n^2}$,

$$(1-\epsilon)\|UZ - A_1\|_F^2 \leq \|T_1UZ - T_1A_1\|_F^2 \leq (1+\epsilon)\|T_1UZ - A_1\|_F^2.$$

It means that

$$(1-\epsilon)\|UZ_1 - A_1\|_F^2 \leq \|T_1UZ_1 - T_1A_1\|_F^2 \leq (1+\epsilon)\|T_1UZ_1 - A_1\|_F^2.$$

Second, we unflatten matrix $T_1A_1 \in \mathbb{R}^{t_1\times n^2}$ to obtain a tensor $A' \in \mathbb{R}^{t_1\times n\times n}$. Then we flatten A' along the second direction to obtain $A'_2 \in \mathbb{R}^{n\times t_1 n}$. We define $Z_2 = ((T_1U)^\top \odot X^\top) \in \mathbb{R}^{s\times t_1 n}$. Then, by flattening,

$$\|V\cdot Z_2 - A'_2\|_F^2 = \|T_1U\cdot Z_1 - T_1A_1\|_F^2 = (1\pm\epsilon)\|U\otimes V\otimes X - A\|_F^2.$$

We choose a sparse embedding matrix (Definition 21.3.10) $T_2 \in \mathbb{R}^{t_2\times n}$ with $t_2 = \text{poly}(r_2/\epsilon)$ rows. Then according to Lemma 21.3.7 with probability 0.999, for all $Z \in \mathbb{R}^{s\times t_1 n}$,

$$(1-\epsilon)\|VZ - A'_2\|_F^2 \leq \|T_2VZ - T_2A'_2\|_F^2 \leq (1+\epsilon)\|VZ - A'_2\|_F^2.$$

Thus,

$$\|T_2V \cdot Z_2 - T_2A_2'\|_F^2 = (1 \pm \epsilon)^2\|U \otimes V \otimes X - A\|_F^2.$$

After rescaling ϵ by a constant, with probability at least 0.99, $\forall X \in \mathbb{R}^{n \times s}$,

$$(1-\epsilon)\|U \otimes V \otimes X - A\|_F^2 \leq \|T_1U \otimes T_2V \otimes X - A(T_1, T_2, I)\|_F^2 \leq (1+\epsilon)\|U \otimes V \otimes X - A\|_F^2.$$

□

21.4.4.2 Algorithm I

We start with a slightly unoptimized bicriteria low rank approximation algorithm.

Algorithm 21.5 Frobenius Norm Bicriteria Low Rank Approximation Algorithm, rank-$O(k^3/\epsilon^3)$

procedure FTensorLowRankBicriteriaCubicRank(A, n, k) ▷ Theorem 21.4.7
  $s_1 \leftarrow s_2 \leftarrow s_3 \leftarrow O(k/\epsilon)$.
  $t_1 \leftarrow t_2 \leftarrow t_3 \leftarrow \text{poly}(k/\epsilon)$.
  Choose $S_i \in \mathbb{R}^{n^2 \times s_i}$ to be a Sketching matrix, $\forall i \in [3]$. ▷ Definition 21.3.12
  Choose $T_i \in \mathbb{R}^{t_i \times n}$ to be a Sketching matrix, $\forall i \in [3]$. ▷ Definition 21.3.10
  Compute $U \leftarrow T_1 \cdot (A_1 \cdot S_1)$, $V \leftarrow T_2 \cdot (A_2 \cdot S_2)$, $W \leftarrow T_3 \cdot (A_3 \cdot S_3)$.
  Compute $C \leftarrow A(T_1, T_2, T_3)$.
  $X \leftarrow$FTensorRegression$(C, U, V, W, t_1, s_1, t_2, s_2, t_3, s_3)$. ▷ Linear regression
  return $X(A_1S_1, A_2S_2, A_3S_3)$.
end procedure

Theorem 21.4.7. *Given a 3rd order tensor $A \in \mathbb{R}^{n \times n \times n}$, for any $k \geq 1, \epsilon \in (0,1)$, let $r = O(k^3/\epsilon^3)$. There exists an algorithm that takes $O(\text{nnz}(A) + n\,\text{poly}(k, 1/\epsilon))$ time and outputs three matrices $U \in \mathbb{R}^{n \times r}, V \in \mathbb{R}^{n \times r}$, $W \in \mathbb{R}^{n \times r}$ such that*

$$\left\| \sum_{i=1}^{r} U_i \otimes V_i \otimes W_i - A \right\|_F^2 \leq (1+\epsilon) \min_{\text{rank}-k\ A_k} \|A_k - A\|_F^2$$

holds with probability 9/10.

Proof. At the end of Theorem 21.4.1, we need to run a polynomial system verifier. This is why we obtain exponential in k running time. Instead of running the polynomial system verifier, we can use Lemma 21.4.5. This reduces the running time to be polynomial in all parameters: $n, k, 1/\epsilon$. However, the output tensor has rank $(k/\epsilon)^3$ (Here we mean that we do not obtain a better decomposition than $(k/\epsilon)^3$ components). According to Section 21.3.6, for each i, $A_i S_i$ can be computed in $O(\mathrm{nnz}(A)) + n\,\mathrm{poly}(k/\epsilon)$ time. Then $T_i(A_i S_i)$ can be computed in $n\,\mathrm{poly}(k, 1/\epsilon)$ time and $A(T_1, T_2, T_3)$ also can be computed in $O(\mathrm{nnz}(A))$ time. The running time for the regression is $\mathrm{poly}(k/\epsilon)$. □

Now we present an optimized bicriteria algorithm.

Theorem 21.4.8. *Given a 3rd order tensor $A \in \mathbb{R}^{n\times n\times n}$, for any $k \geq 1, \epsilon \in (0,1)$, let $r = O(k^2/\epsilon^2)$. There exists an algorithm that takes $O(\mathrm{nnz}(A) + n\,\mathrm{poly}(k, 1/\epsilon))$ time and outputs three matrices $U \in \mathbb{R}^{n\times r}, V \in \mathbb{R}^{n\times r}$, $W \in \mathbb{R}^{n\times r}$ such that*

$$\left\| \sum_{i=1}^{r} U_i \otimes V_i \otimes W_i - A \right\|_F^2 \leq (1+\epsilon) \min_{\mathrm{rank}-k\ A_k} \|A_k - A\|_F^2$$

holds with probability 9/10.

Note that there are two different ways to implement algorithm FTensorLowRankBicriteriaQuadraticRank. We present the proofs for both of them here.

Approach I.

Proof. Let $\mathrm{OPT} = \min_{\mathrm{rank}-k\ A_k} \|A_k - A\|_F^2$. According to Theorem 21.4.1, we know that there exists a sketching matrix $S_3 \in \mathbb{R}^{n^2\times s_3}$ where $s_3 = O(k/\epsilon)$, such that

$$\min_{X_1\in\mathbb{R}^{s_1\times k}, X_2\in\mathbb{R}^{s_2\times k}, X_3\in\mathbb{R}^{s_3\times k}} \left\| \sum_{l=1}^{k} (A_1 S_1 X_1)_l \otimes (A_2 S_2 X_2)_l \otimes (A_3 S_3 X_3)_l - A \right\|_F^2 \leq (1+\epsilon)\,\mathrm{OPT}$$

Algorithm 21.6 Frobenius Norm Low Rank Approximation Algorithm, rank-$O(k^2/\epsilon^2)$

procedure FTensorLowRankBicriteriaQuadraticRank(A, n, k) ▷ Theorem 21.4.8
  $s_1 \leftarrow s_2 \leftarrow O(k/\epsilon)$.
  Choose $S_i \in \mathbb{R}^{n^2 \times s_i}$ to be a sketching matrix, $\forall i \in [3]$. ▷ Definition 21.3.12
  Compute $A_1 \cdot S_1$, $A_2 \cdot S_2$.
  Form $\widehat{U}$ by using $A_1 S_1$ according to Equation (21.9).
  Form $\widehat{V}$ by using $A_2 S_2$ according to Equation (21.10).
  $\widehat{W} \leftarrow$FTensorMultipleRegression$(A, \widehat{U}, \widehat{V}, n, n, s_1 s_2)$. ▷ Algorithm 21.4
  return $\widehat{U}, \widehat{V}, \widehat{W}$.
end procedure
procedure FTensorLowRankBicriteriaQuadraticRank(A, n, k) ▷ Theorem 21.4.8
  $s_1 \leftarrow s_2 \leftarrow O(k/\epsilon)$.
  $t_1 \leftarrow t_2 \leftarrow \mathrm{poly}(k/\epsilon)$.
  Choose $S_i \in \mathbb{R}^{n^2 \times s_i}$ to be a Sketching matrix, $\forall i \in [2]$. ▷ Definition 21.3.12
  Choose $T_i \in \mathbb{R}^{t_i \times n}$ to be a Sketching matrix, $\forall i \in [2]$. ▷ Definition 21.3.10
  Form $\widehat{U}$ by using $A_1 S_1$ according to Equation (21.9).
  Form $\widehat{V}$ by using $A_2 S_2$ according to Equation (21.10).
  Compute $C \leftarrow A(T_1, T_2, I)$. ▷ $C \in \mathbb{R}^{t_1 \times t_2 \times n}$
  Compute $B \leftarrow (T_1 \widehat{U})^\top \odot (T_2 \widehat{V})^\top$.
  $\widehat{W} \leftarrow \arg\min_{X \in \mathbb{R}^{n \times s_1 s_2}} \|XB - C_3\|_F^2$.
  return $\widehat{U}, \widehat{V}, \widehat{W}$.
end procedure

Now we fix an l and we have:

$$\begin{aligned}
&(A_1 S_1 X_1)_l \otimes (A_2 S_2 X_2)_l \otimes (A_3 S_3 X_3)_l \\
=&\left(\sum_{i=1}^{s_1} (A_1 S_1)_i (X_1)_{i,l}\right) \otimes \left(\sum_{j=1}^{s_2} (A_2 S_2)_j (X_2)_{j,l}\right) \otimes (A_3 S_3 X_3)_l \\
=&\sum_{i=1}^{s_1} \sum_{j=1}^{s_2} (A_1 S_1)_i \otimes (A_2 S_2)_j \otimes (A_3 S_3 X_3)_l (X_1)_{i,l} (X_2)_{j,l}
\end{aligned}$$

Thus, we have

$$\min_{X_1,X_2,X_3} \left\| \sum_{i=1}^{s_1} \sum_{j=1}^{s_2} (A_1S_1)_i \otimes (A_2S_2)_j \otimes \left(\sum_{l=1}^{k} (A_3S_3X_3)_l (X_1)_{i,l} (X_2)_{j,l} \right) - A \right\|_F^2 \leq (1+\epsilon)\,\mathrm{OPT}\,. \tag{21.7}$$

We use matrices $A_1S_1 \in \mathbb{R}^{n\times s_1}$ and $A_2S_2 \in \mathbb{R}^{n\times s_2}$ to construct a matrix $B \in \mathbb{R}^{s_1s_2\times n^2}$ in the following way: each row of B is the vector corresponding to the matrix generated by the $\otimes$ product between one column vector in A_1S_1 and the other column vector in A_2S_2, i.e.,

$$B^{i+(j-1)s_1} = \mathrm{vec}((A_1S_1)_i \otimes (A_2S_2)_j), \forall i \in [s_1], j \in [s_2], \tag{21.8}$$

where $(A_1S_1)_i$ denotes the i-th column of A_1S_1 and $(A_2S_2)_j$ denote the j-th column of A_2S_2.

We create matrix $\widehat{U} \in \mathbb{R}^{n\times s_1s_2}$ by copying matrix A_1S_1 s_2 times, i.e.,

$$\widehat{U} = \begin{bmatrix} A_1S_1 & A_1S_1 & \cdots & A_1S_1 \end{bmatrix}. \tag{21.9}$$

We create matrix $\widehat{V} \in \mathbb{R}^{n\times s_1s_2}$ by copying the i-th column of A_2S_2 a total of s_1 times, into columns $(i-1)s_1, \cdots, is_1$ of $\widehat{V}$, for each $i \in [s_2]$, i.e.,

$$\widehat{V} = \begin{bmatrix} (A_2S_2)_1 & \cdots & (A_2S_2)_1 & (A_2S_2)_2 & \cdots & (A_2S_2)_2 & \cdots & (A_2S_2)_{s_2} & \cdots & (A_2S_2)_{s_2} \end{bmatrix}. \tag{21.10}$$

Thus, we can use $\widehat{U}$ and $\widehat{V}$ to represent B,

$$B = (\widehat{U}^\top \odot \widehat{V}^\top) \in \mathbb{R}^{s_1s_2\times n^2}.$$

According to Equation (21.7), we have:

$$\min_{W\in\mathbb{R}^{n\times s_1s_2}} \|WB - A_3\|_F^2 \leq (1+\epsilon)\,\mathrm{OPT}\,.$$

Next, we want to find matrix $W \in \mathbb{R}^{n \times s_1 s_2}$ by solving the following optimization problem,

$$\min_{W \in \mathbb{R}^{n \times s_1 s_2}} \|WB - A_3\|_F^2.$$

Note that B has size $s_1 s_2 \times n^2$. Naïvely writing down B already requires $\Omega(n^2)$ time. In order to achieve nearly linear time in n, we cannot write down B. We choose $S_3 \in \mathbb{R}^{n_1 n_2 \times s_3}$ to be a TensorSketch (Definition 21.3.20). In order to solve multiple regression, we need to set $s_3 = O((s_1 s_2)^2 + (s_1 s_2)/\epsilon)$. Let $\widehat{W}$ denote the optimal solution to $\|WBS_3 - A_3 S_3\|_F^2$. Then $\widehat{W} = (A_3 S_3)(BS_3)^\dagger$. Since each row of S_3 has exactly 1 nonzero entry, $A_3 S_3$ can be computed in $O(\mathrm{nnz}(A))$ time. Since $B = (\widehat{U}^\top \odot \widehat{V}^\top)$, according to Definition 21.3.20, BS_3 can be computed in $n \,\mathrm{poly}(s_1 s_2/\epsilon) = n \,\mathrm{poly}(k/\epsilon)$ time. By Theorem 21.4.4, we have

$$\|\widehat{W}B - A_3\|_F^2 \leq (1+\epsilon) \min_{W \in \mathbb{R}^{n \times s_1 s_2}} \|WB - A_3\|_F^2.$$

Thus, we have

$$\|\widehat{U} \otimes \widehat{V} \otimes \widehat{W} - A\|_F^2 \leq (1+\epsilon)\,\mathrm{OPT}\,.$$

According to Definition 21.3.12, $A_1 S_1, A_2 S_2$ can be computed in $O(\mathrm{nnz}(A) + \mathrm{poly}(k/\epsilon))$ time. Te total running time is thus $O(\mathrm{nnz}(A) + \mathrm{poly}(k/\epsilon))$. □

Approach II.

Proof. Let $\mathrm{OPT} = \min_{\mathrm{rank}-k\ A_k} \|A_k - A\|_F^2$. Choose sketching matrices (Definition 21.3.12) $S_1 \in \mathbb{R}^{n^2 \times s_1}$, $S_2 \in \mathbb{R}^{n^2 \times s_2}$, $S_3 \in \mathbb{R}^{n^2 \times s_3}$, and sketching matrices (Definition 21.3.10) $T_1 \in \mathbb{R}^{t_1 \times n}$ and

$T_2 \in \mathbb{R}^{t_2 \times n}$ with $s_1 = s_2 = s_3 = O(k/\epsilon), t_1 = t_2 = \mathrm{poly}(k/\epsilon)$. We create matrix $\widehat{U} \in \mathbb{R}^{n \times s_1 s_2}$ by copying matrix $A_1 S_1$ s_2 times, i.e.,

$$\widehat{U} = \begin{bmatrix} A_1 S_1 & A_1 S_1 & \cdots & A_1 S_1 \end{bmatrix}.$$

We create matrix $\widehat{V} \in \mathbb{R}^{n \times s_1 s_2}$ by copying the i-th column of $A_2 S_2$ a total of s_1 times, into columns $(i-1)s_1, \cdots, is_1$ of $\widehat{V}$, for each $i \in [s_2]$, i.e.,

$$\widehat{V} = \begin{bmatrix} (A_2 S_2)_1 & \cdots & (A_2 S_2)_1 & (A_2 S_2)_2 & \cdots & (A_2 S_2)_2 & \cdots & (A_2 S_2)_{s_2} & \cdots & (A_2 S_2)_{s_2} \end{bmatrix}.$$

As we proved in Approach I, we have

$$\min_{X \in \mathbb{R}^{n \times s_1 s_2}} \|\widehat{U} \otimes \widehat{V} \otimes X - A\|_F^2 \leq (1+\epsilon)\,\mathrm{OPT}\,.$$

Let $B = ((T_1 \widehat{U})^\top \odot (T_2 \widehat{V})^\top) \in \mathbb{R}^{s_1 s_2 \times t_1 t_2}$, and flatten $A(T_1, T_2, I)$ along the third direction to obtain $C_3 \in \mathbb{R}^{n \times t_1 t_2}$. Let

$$\widehat{W} = \arg\min_{X \in \mathbb{R}^{n \times s_1 s_2}} \|T_1 \widehat{U} \otimes T_2 \widehat{V} \otimes X - A(T_1, T_2, I)\|_F^2 = \arg\min_{X \in \mathbb{R}^{n \times s_1 s_2}} \|XB - C_3\|_F^2.$$

Let

$$W^* = \arg\min_{X \in \mathbb{R}^{n \times s_1 s_2}} \|\widehat{U} \otimes \widehat{V} \otimes X - A\|_F^2.$$

According to Lemma 21.4.6,

$$\begin{aligned}
& \|\widehat{U} \otimes \widehat{V} \otimes \widehat{W} - A\|_F^2 \\
\leq & \frac{1}{1-\epsilon} \|T_1 \widehat{U} \otimes T_2 \widehat{V} \otimes \widehat{W} - A(T_1, T_2, I)\|_F^2 \\
\leq & \frac{1}{1-\epsilon} \|T_1 \widehat{U} \otimes T_2 \widehat{V} \otimes W^* - A(T_1, T_2, I)\|_F^2 \\
\leq & \frac{1+\epsilon}{1-\epsilon} \|\widehat{U} \otimes \widehat{V} \otimes W^* - A\|_F^2 \\
\leq & \frac{(1+\epsilon)^2}{1-\epsilon}\,\mathrm{OPT}\,.
\end{aligned}$$

According to Definition 21.3.12, A_1S_1, A_2S_2 can be computed in $O(\text{nnz}(A)+\text{poly}(k/\epsilon))$ time. The total running time is thus $O(\text{nnz}(A)+\text{poly}(k/\epsilon))$. Since T_1, T_2 are sparse embedding matrices, $T_1\widehat{U}, T_2\widehat{V}$ can be computed in $O(\text{nnz}(A)+\text{poly}(k/\epsilon))$ time. The total running time is in $O(\text{nnz}(A)+\text{poly}(k/\epsilon))$. □

Theorem 21.4.9. *Given a 3rd order tensor $A \in \mathbb{R}^{n\times n\times n}$, for any $k \geq 1$ and any $0 < \epsilon < 1$, if A_k exists then there is a randomized algorithm running in $\text{nnz}(A) + n \cdot \text{poly}(k/\epsilon)$ time which outputs a rank-$O(k^2/\epsilon^2)$ tensor B for which $\|A - B\|_F^2 \leq (1+\epsilon)\|A - A_k\|_F^2$. If A_k does not exist, then the algorithm outputs a rank-$O(k^2/\epsilon^2)$ tensor B for which $\|A - B\|_F^2 \leq (1+\epsilon)\,\text{OPT} + \gamma$, where γ is an arbitrarily small positive function of n. In both cases, the algorithm succeeds with probability at least $9/10$.*

Proof. If A_k exists, then the proof directly follows the proof of Theorem 21.4.1 and Theorem 21.4.8. If A_k does not exist, then for any $\gamma > 0$, there exist $U^* \in \mathbb{R}^{n\times k}, V^* \in \mathbb{R}^{n\times k}, W^* \in \mathbb{R}^{n\times k}$ such that

$$\|U^* \otimes V^* \otimes W^* - A\|_F^2 \leq \inf_{\text{rank}-k\ A'} \|A - A'\|_F^2 + \frac{1}{10}\gamma.$$

Then we just regard $U^* \otimes V^* \otimes W^*$ as the "best" rank k approximation to A, and follow the same argument as in the proof of Theorem 21.4.1 and the proof of Theorem 21.4.8. We can finally output a tensor $B \in \mathbb{R}^{n\times n\times n}$ with rank-$O(k^2/\epsilon^2)$ such that

$$\begin{aligned}
\|B - A\|_F^2 \leq\ & (1+\epsilon)\|U^* \otimes V^* \otimes W^* - A\|_F^2 \\
\leq\ & (1+\epsilon)\left(\inf_{\text{rank}-k\ A'} \|A - A'\|_F^2 + \frac{1}{10}\gamma\right) \\
\leq\ & (1+\epsilon)\inf_{\text{rank}-k\ A'} \|A - A'\|_F^2 + \gamma
\end{aligned}$$

where the first inequality follows by the proof of Theorem 21.4.1 and the proof of theorem 21.4.8. The second inequality follows by our choice of U^*, V^*, W^*. The third inequality follows since $1 + \epsilon < 2$ and $\gamma > 0$. □

21.4.4.3 poly(k)-approximation to multiple regression

Lemma 21.4.10 ((1.4) and (1.9) in [RV09]). *Let $s \geq k$. Let $U \in \mathbb{R}^{n \times k}$ denote a matrix that has orthonormal columns, and $S \in \mathbb{R}^{s \times n}$ denote an i.i.d. $N(0, 1/s)$ Gaussian matrix. Then SU is also an $s \times k$ i.i.d. Gaussian matrix with each entry draw from $N(0, 1/s)$, and furthermore, we have with arbitrarily large constant probability,*

$$\sigma_{\max}(SU) = O(1) \text{ and } \sigma_{\min}(SU) = \Omega(1/\sqrt{s}).$$

Proof. Note that $\sqrt{s} - \sqrt{k-1} = \frac{s-k-1}{\sqrt{s}+\sqrt{k-1}} = \Omega(1/\sqrt{s})$. □

Lemma 21.4.11. *Given matrices $A \in \mathbb{R}^{n \times k}$, $B \in \mathbb{R}^{n \times d}$, let $S \in \mathbb{R}^{s \times n}$ denote a standard Gaussian $N(0, 1)$ matrix with $s = k$. Let $X^* = \min_{X \in \mathbb{R}^{k \times d}} \|AX - B\|_F$. Let $X' = \min_{X \in \mathbb{R}^{k \times d}} \|SAX - SB\|_F$. Then, we have that*

$$\|AX' - B\|_F \leq O(\sqrt{k}) \|AX^* - B\|_F,$$

holds with probability at least 0.99.

Proof. Let $X^* \in \mathbb{R}^{k \times d}$ denote the optimal solution such that

$$\|AX^* - B\|_F = \min_{X \in \mathbb{R}^{k \times d}} \|AX - B\|_F.$$

Consider a standard Gaussian matrix $S \in \mathbb{R}^{k \times n}$ scaled by $1/\sqrt{k}$ with exactly k rows. Then for any $X \in \mathbb{R}^{k \times d}$, by the triangle inequality, we have

$$\|SAX - SB\|_F \leq \|SAX - SAX^*\|_F + \|SAX^* - SB\|_F,$$

and

$$\|SAX - SB\|_F \geq \|SAX - SAX^*\|_F - \|SAX^* - SB\|_F.$$

We first show how to bound $\|SAX - SAX^*\|_F$, and then show how to bound $\|SAX^* - SB\|_F$.

Note that Lemma 21.4.10 implies the following result,

Claim 21.4.12. *For any $X \in \mathbb{R}^{k \times d}$, with probability 0.999, we have*

$$\frac{1}{\sqrt{k}}\|AX - AX^*\|_F \lesssim \|SAX - SAX^*\|_F \lesssim \|AX - AX^*\|_F.$$

Proof. First, we can write $A = UR \in \mathbb{R}^{n \times k}$ where $U \in \mathbb{R}^{n \times k}$ has orthonormal columns and $R \in \mathbb{R}^{k \times k}$. It gives,

$$\|SAX - SAX^*\|_F = \|SU(RX - RX^*)\|_F.$$

Second, applying Lemma 21.4.10 to $SU \in \mathbb{R}^{s \times k}$ completes the proof. □

Using Markov's inequality, for any fixed matrix $AX^* - B$, choosing a Gaussian matrix S, we have that

$$\|SAX^* - SB\|_F^2 = O(\|AX^* - B\|_F^2)$$

holds with probability at least 0.999. This is equivalent to

$$\|SAX^* - SB\|_F = O(\|AX^* - B\|_F), \tag{21.11}$$

holding with probability at least 0.999.

Let $X' = \arg\min_{X \in \mathbb{R}^{k \times d}} \|SAX - SB\|_F$. Putting it all together, we have

$$\begin{aligned}
&\|AX' - B\|_F \\
\leq\ & \|AX' - AX^*\|_F + \|AX^* - B\|_F && \text{by triangle inequality} \\
\leq\ & O(\sqrt{k})\|SAX' - SAX^*\|_F + \|AX^* - B\|_F && \text{by Claim 21.4.12} \\
\leq\ & O(\sqrt{k})\|SAX' - SB\|_F + O(\sqrt{k})\|SAX^* - SB\|_F + \|AX^* - B\|_F && \text{by triangle inequality} \\
\leq\ & O(\sqrt{k})\|SAX^* - SB\|_F + O(\sqrt{k})\|SAX^* - SB\|_F + \|AX^* - B\|_F && \text{by definition of } X' \\
\leq\ & O(\sqrt{k})\|AX^* - B\|_F. && \text{by Equation (21.11)}
\end{aligned}$$

□

21.4.4.4 Algorithm II

Theorem 21.4.13. *Given a 3rd order tensor $A \in \mathbb{R}^{n \times n \times n}$, for any $k \geq 1$, let $r = k^2$. There exists an algorithm which takes $O(\mathrm{nnz}(A)k) + n\,\mathrm{poly}(k)$ time and outputs three matrices $U, V, W \in \mathbb{R}^{n \times r}$ such that,*

$$\left\| \sum_{i=1}^{r} U_i \otimes V_i \otimes W_i - A \right\|_F \leq \mathrm{poly}(k) \min_{\mathrm{rank}-k\ A'} \|A' - A\|_F$$

holds with probability 9/10.

Proof. Let $\mathrm{OPT} = \min_{\mathrm{rank}-k\ A'} \|A' - A\|_F$, we fix $V^* \in \mathbb{R}^{n \times k}, W^* \in \mathbb{R}^{n \times k}$ to be the optimal solution of the original problem. We use $Z_1 = (V^{*\top} \odot W^{*\top}) \in \mathbb{R}^{k \times n^2}$ to denote the matrix where the i-th row is the vectorization of $V_i^* \otimes W_i^*$. Let $A_1 \in \mathbb{R}^{n \times n^2}$ denote the matrix

obtained by flattening tensor $A \in \mathbb{R}^{n \times n \times n}$ along the first direction. Then, we have

$$\min_U \|UZ_1 - A_1\|_F \leq \mathrm{OPT}\,.$$

Choosing an $N(0, 1/k)$ Gaussian sketching matrix $S_1 \in \mathbb{R}^{n^2 \times s_1}$ with $s_1 = k$, we can obtain the smaller problem,

$$\min_{U \in \mathbb{R}^{n \times k}} \|UZ_1S_1 - A_1S_1\|_F.$$

Define $\widehat{U} = A_1S_1(Z_1S_1)^\dagger$. Define $\alpha = O(\sqrt{k})$. By Lemma 21.4.11, we have

$$\|\widehat{U}Z_1 - A_1\|_F \leq \alpha\,\mathrm{OPT}\,.$$

Second, we fix $\widehat{U}$ and W^*. Define Z_2, A_2 similarly as above. Choosing an $N(0, 1/k)$ Gaussian sketching matrix $S_2 \in \mathbb{R}^{n^2 \times s_2}$ with $s_2 = k$, we can obtain another smaller problem,

$$\min_{V \in \mathbb{R}^{n \times k}} \|VZ_2S_2 - A_2S_2\|_F.$$

Define $\widehat{V} = A_2S_2(Z_2S_2)^\dagger$. By Lemma 21.4.11 again, we have

$$\|\widehat{V}Z_2 - A_2\|_F \leq \alpha^2\,\mathrm{OPT}\,.$$

Thus, we now have

$$\min_{X_1, X_2, W} \|A_1S_1X_1 \otimes A_2S_2X_2 \otimes W - A\|_F \leq \alpha^2\,\mathrm{OPT}$$

We use a similar idea as in the proof of Theorem 21.4.8. We create matrix $\widetilde{U} \in \mathbb{R}^{n \times s_1s_2}$ by copying matrix A_1S_1 s_2 times, i.e.,

$$\widetilde{U} = \begin{bmatrix} A_1S_1 & A_1S_1 & \cdots & A_1S_1 \end{bmatrix}.$$

We create matrix $\widetilde{V} \in \mathbb{R}^{n \times s_1 s_2}$ by copying the i-th column of $A_2 S_2$ a total of s_1 times, into columns $(i-1)s_1, \cdots, is_1$ of $\widetilde{V}$, for each $i \in [s_2]$, i.e.,

$$\widetilde{V} = \begin{bmatrix} (A_2S_2)_1 & \cdots & (A_2S_2)_1 & (A_2S_2)_2 & \cdots & (A_2S_2)_2 & \cdots & (A_2S_2)_{s_2} & \cdots & (A_2S_2)_{s_2} \end{bmatrix}.$$

We have

$$\min_{X \in \mathbb{R}^{n \times s_1 s_2}} \|\widetilde{U} \otimes \widetilde{V} \otimes X - A\|_F \leq \alpha^2 \operatorname{OPT}.$$

Choose $T_i \in \mathbb{R}^{t_i \times n}$ to be a sparse embedding matrix (Definition 21.3.10) with $t_i = \operatorname{poly}(k/\epsilon)$, for each $i \in [2]$. By applying Lemma 21.4.6, we have, if W' satisfies,

$$\|T_1\widetilde{U} \otimes T_2\widetilde{V} \otimes W' - A(T_1, T_2, I)\|_F = \min_{X \in \mathbb{R}^{n \times s_1 s_2}} \|T_1\widetilde{U} \otimes T_2\widetilde{V} \otimes X - A(T_1, T_2, I)\|_F$$

then,

$$\|\widetilde{U} \otimes \widetilde{V} \otimes W' - A\|_F \leq (1+\epsilon) \min_{X \in \mathbb{R}^{n \times s_1 s_2}} \|\widetilde{U} \otimes \widetilde{V} \otimes X - A\|_F \leq (1+\epsilon)\alpha^2 \operatorname{OPT}.$$

Thus, we only need to solve

$$\min_{X \in \mathbb{R}^{n \times s_1 s_2}} \|T_1\widetilde{U} \otimes T_2\widetilde{V} \otimes X - A(T_1, T_2, I)\|_F.$$

which is similar to the proof of Theorem 21.4.8. Therefore, we complete the proof of correctness. For the running time, A_1S_1, A_2S_2 can be computed in $O(\operatorname{nnz}(A)k)$ time, $T_1\widetilde{U}, T_2\widetilde{V}$ can be computed in $n \operatorname{poly}(k)$ time. The final regression problem can be computed in $n \operatorname{poly}(k)$ running time. □

21.4.5 Generalized matrix row subset selection

Note that in this section, the notation $\Pi_{C,k}^{\xi}$ is given in Definition 21.3.5.

Algorithm 21.7 Generalized Matrix Row Subset Selection: Constructing R with $r = O(k + k/\epsilon)$ Rows and a rank-k $U \in \mathbb{R}^{k \times r}$

procedure GeneralizedMatrixRowSubsetSelection(A, C, n, m, k, ϵ) ▷ Theorem 21.4.14
  $Y, \Phi, \Delta \leftarrow$ ApproxSubspaceSVD(A, C, k). ▷ Claim 21.4.16 and Lemma 3.12 in [BW14]
  $B \leftarrow Y\Delta$.
  $Z_2, D \leftarrow \mathrm{QR}(B)$. ▷ $Z_2 \in \mathbb{R}^{m \times k}$, $Z_2^\top Z_2 = I_k$, $D \in \mathbb{R}^{k \times k}$
  $h_2 \leftarrow 8k \ln(20k)$.
  $\Omega_2, D_2 \leftarrow$ RandSampling($Z_2, h_2, 1$) ▷ Definition 3.6 in [BW14]
  $M_2 \leftarrow Z_2^\top \Omega_2 D_2 \in \mathbb{R}^{k \times h_2}$.
  $U_{M_2}, \Sigma_{M_2}, V_{M_2}^\top \leftarrow \mathrm{SVD}(M_2)$. ▷ $\mathrm{rank}(M_2) = k$ and $V_{M_2} \in \mathbb{R}^{h_2 \times k}$
  $r_1 \leftarrow 4k$.
  $S_2 \leftarrow$ BSSSamplingSparse($V_{M_2}, ((A^\top - A^\top Z_2 Z_2^\top)\Omega_2 D_2)^\top, r_1, 0.5$) ▷ Lemma 4.3 in [BW14]
  $R_1 \leftarrow (A^\top \Omega_2 D_2 S_2)^\top \in \mathbb{R}^{r_1 \times n}$ containing rescaled rows from A.
  $r_2 \leftarrow 4820k/\epsilon$.
  $R_2 \leftarrow$ AdaptiveRowsSparse(A, Z_2, R_1, r_2) ▷ Lemma 4.5 in [BW14]
  $R \leftarrow [R_1^\top, R_2^\top]^\top$. ▷ $R \in \mathbb{R}^{(r_1+r_2) \times n}$ containing $r = 4k + 4820k/\epsilon$ rescaled rows of A.
  Choose $W \in \mathbb{R}^{\xi \times m}$ to be a randomly chosen sparse subspace embedding with $\xi = \Omega(k^2\epsilon^{-2})$.
  $U \leftarrow \Phi^{-1}\Delta D^{-1}(WC\Phi^{-1}\Delta D^{-1})^\dagger WAR^\dagger = \Phi^{-1}\Delta\Delta^\top(WC)^\dagger WAR^\dagger$.
  return R, U.
end procedure

Theorem 21.4.14. *Given matrices $A \in \mathbb{R}^{n \times m}$ and $C \in \mathbb{R}^{n \times k}$, there exists an algorithm which takes $O(\mathrm{nnz}(A) \log n) + (m+n)\,\mathrm{poly}(k, 1/\epsilon)$ time and outputs a diagonal matrix $D \in \mathbb{R}^{n \times n}$ with $d = O(k/\epsilon)$ nonzeros (or equivalently a matrix R that contains $d = O(k/\epsilon)$ rescaled*

rows of A) and a matrix $U \in \mathbb{R}^{k\times d}$ such that

$$\|CUDA - A\|_F^2 \leq (1+\epsilon) \min_{X\in\mathbb{R}^{k\times m}} \|CX - A\|_F^2$$

holds with probability .99.

Proof. This follows by combining Lemma 21.4.17 and 21.4.18. Let U, R denote the output of procedure GeneralizedMatrixRowSubsetSelection,

$$\begin{aligned}\|A - CUR\|_F^2 &\leq (1+\epsilon)\|A - Z_2 Z_2^\top A R^\dagger R\|_F^2 \\ &\leq (1+\epsilon)(1+60\epsilon)\|A - \Pi_{C,k}^F(A)\|_F^2 \\ &\leq (1+130\epsilon)\|A - \Pi_{C,k}^F(A)\|_F^2.\end{aligned}$$

Because R is a subset of rows of A and R has size $O(k/\epsilon) \times m$, there must exist a diagonal matrix $D \in \mathbb{R}^{n\times n}$ with $O(k/\epsilon)$ nonzeros such that $R = DA$. This completes the proof. □

Corollary 21.4.15 (A slightly different version of Theorem 21.4.14, faster running time, and small input matrix)**.** *Given matrices $A \in \mathbb{R}^{n\times m}$ and $C \in \mathbb{R}^{n\times k}$, if $\min(m,n) = \mathrm{poly}(k, 1/\epsilon)$, then there exists an algorithm which takes $O(\mathrm{nnz}(A)) + (m+n)\,\mathrm{poly}(k, 1/\epsilon)$ time and outputs a diagonal matrix $D \in \mathbb{R}^{n\times n}$ with $d = O(k/\epsilon)$ nonzeros (or equivalently a matrix R that contains $d = O(k/\epsilon)$ rescaled rows of A) and a matrix $U \in \mathbb{R}^{k\times d}$ such that*

$$\|CUDA - A\|_F^2 \leq (1+\epsilon) \min_{X\in\mathbb{R}^{k\times m}} \|CX - A\|_F^2$$

holds with probability .99.

Proof. The $\log n$ factor comes from the adaptive sampling where we need to choose a Gaussian matrix with $O(\log n)$ rows and compute SA. If A has $\mathrm{poly}(k, 1/\epsilon)$ columns, it is sufficient

to choose S to be a CountSketch matrix with $\text{poly}(k, 1/\epsilon)$ rows. Then, we do not need a $\log n$ factor in the running time. If S has $\text{poly}(k, 1/\epsilon)$ rows, then we no longer need the matrix S. □

Claim 21.4.16. *Given matrices $A \in \mathbb{R}^{m\times n}$ and $C \in \mathbb{R}^{m\times c}$, let $Y \in \mathbb{R}^{m\times c}, \Phi \in \mathbb{R}^{c\times c}$ and $\Delta \in \mathbb{R}^{c\times k}$ denote the output of procedure* ApproxSubspaceSVD(A, C, k, ϵ). *Then with probability .99, we have,*

$$\|A - Y\Delta\Delta^\top Y^\top A\|_F^2 \leq (1 + 30\epsilon)\|A - \Pi_{C,k}^F(A)\|_F^2.$$

Proof. This follows by Lemma 3.12 in [BW14]. □

Lemma 21.4.17. *The matrices R and Z_2 in procedure* GeneralizedMatrixRowSubsetSelection *(Algorithm 21.7) satisfy with probability at least $0.17 - 2/n$,*

$$\|A - Z_2 Z_2^\top A R^\dagger R\|_F^2 \leq \|A - \Pi_{C,k}^F(A)\|_F^2 + 60\epsilon\|A - \Pi_{C,k}^F(A)\|_F^2.$$

Proof. We can show,

$$\begin{aligned}
&\|A - Z_2 Z_2^\top A\|_F^2 + \frac{30\epsilon}{4820}\|A - AR_1^\dagger R_1\|_F^2 \\
=\ &\|A - BB^\dagger A\|_F^2 + \frac{30\epsilon}{4820}\|A - AR_1^\dagger R_1\|_F^2 \\
\leq\ &\|A - BB^\dagger A\|_F^2 + 30\epsilon\|A - A_k\|_F^2 \\
\leq\ &\|A - Y\Delta\Delta^\top Y A\|_F^2 + 30\epsilon\|A - \Pi_{C,k}^F(A)\|_F^2 \\
\leq\ &(1 + 30\epsilon)\|A - \Pi_{C,k}^F(A)\|_F^2 + 30\epsilon\|A - \Pi_{C,k}^F(A)\|_F^2,
\end{aligned}$$

where the first step follows by the fact that $Z_2 Z_2^\top = Z_2 D D^{-1} Z_2^\top = (Z_2 D)(Z_2 D)^\dagger = BB^\dagger$, the second step follows by $\|A - AR_1^\dagger R_1\|_F^2 \leq 4820\|A - A_k\|_F^2$, the third step follows by $B = Y\Delta$ and $B^\dagger = (Y\Delta)^\dagger = \Delta^\dagger Y^\dagger = \Delta^\top Y^\top$, and the last step follows by Claim 21.4.16. □

Lemma 21.4.18. *The matrices C, U and R in procedure* GeneralizedMatrixRowSubsetSelection *(Algorithm 21.7) satisfy that*

$$\|A - CUR\|_F^2 \leq (1+\epsilon)\|A - Z_2 Z_2^\top A R^\dagger R\|_F^2$$

with probability at least .99.

Proof. Let U_R, Σ_R, V_R denote the SVD of R. Then $V_R V_R^\top = R^\dagger R$.

We define Y^* to be the optimal solution of

$$\min_{X \in \mathbb{R}^{k \times r}} \|WAV_R V_R^\top - WC\Phi^{-1}\Delta D^{-1} Y R\|_F^2.$$

We define $\widehat{X}^*$ to be $Y^* R \in \mathbb{R}^{k \times n}$, which is also equivalent to defining $\widehat{X}^*$ to be the optimal solution of

$$\min_{X \in \mathbb{R}^{k \times n}} \|WAV_R V_R^\top - WC\Phi^{-1}\Delta D^{-1} X\|_F^2.$$

Furthermore, it implies $\widehat{X}^* = (WC\Phi^{-1}\Delta D^{-1})^\dagger WAV_R V_R^\dagger$.

We also define X^* to be the optimal solution of

$$\min_{X \in \mathbb{R}^{k \times n}} \|AV_R V_R^\dagger - C\Phi^{-1}\Delta D^{-1} X\|_F^2,$$

which implies that,

$$X^* = (C\Phi^{-1}\Delta D^{-1})^\dagger AV_R V_R^\top = Z_2^\top AV_R V_R^\top.$$

Now, we start to prove an upper bound on $\|A - CUR\|_F^2$,

$$\begin{aligned}
\|A - CUR\|_F^2 &= \|A - C\Phi^{-1}\Delta D^{-1}Y^*R\|_F^2 && \text{by definition of } U \\
&= \|A - C\Phi^{-1}\Delta D^{-1}\widehat{X}^*\|_F^2 && \text{by } \widehat{X}^* = Y^*R \\
&= \|AV_RV_R^\top - C\Phi^{-1}\Delta D^{-1}\widehat{X}^* + A - AV_RV_R^\top\|_F^2 \\
&= \underbrace{\|AV_RV_R^\top - C\Phi^{-1}\Delta D^{-1}\widehat{X}^*\|_F^2}_{\alpha} + \underbrace{\|A - AV_RV_R^\top\|_F^2}_{\beta}, && (21.12)
\end{aligned}$$

where the last step follows by $\widehat{X}^* = MV_R^\top$, $A - AV_RV_R^\top = A(I - V_RV_R^\top)$ and the Pythagorean theorem. We show how to upper bound the term α,

$$\begin{aligned}
\alpha &\leq (1+\epsilon)\|AV_RV_R^\top - C\Phi^{-1}\Delta D^{-1}X^*\|_F^2 && \text{by Lemma 21.4.19} \\
&= \epsilon\|AV_RV_R^\top - C\Phi^{-1}\Delta D^{-1}X^*\|_F^2 + \|AV_RV_R^\top - C\Phi^{-1}\Delta D^{-1}X^*\|_F^2 \\
&= \epsilon\|AV_RV_R^\top - C\Phi^{-1}\Delta D^{-1}X^*\|_F^2 + \|AV_RV_R^\top - C\Phi^{-1}\Delta D^{-1}(Z_2^\top AR^\dagger R)\|_F^2. && (21.13)
\end{aligned}$$

By the Pythagorean theorem and the definition of Z_2 (which means $Z_2 = C\Phi^{-1}\Delta D^{-1}$), we have,

$$\begin{aligned}
&\|AV_RV_R^\top - C\Phi^{-1}\Delta D^{-1}(Z_2^\top AR^\dagger R)\|_F^2 + \beta \\
&= \|AV_RV_R^\top - C\Phi^{-1}\Delta D^{-1}(Z_2^\top AR^\dagger R)\|_F^2 + \|A - AV_RV_R^\top\|_F^2 \\
&= \|A - C\Phi^{-1}\Delta D^{-1}(Z_2^\top AR^\dagger R)\|_F^2 \\
&= \|A - Z_2Z_2^\top AR^\dagger R\|_F^2. && (21.14)
\end{aligned}$$

Combining Equations (21.12), (21.13) and (21.14) together, we obtain,

$$\|A - CUR\|_F^2 \leq \epsilon\|AV_RV_R^\top - C\Phi^{-1}\Delta D^{-1}X^*\|_F^2 + \|A - Z_2Z_2^\top AR^\dagger R\|_F^2.$$

We want to show $\|AV_RV_R^\top - C\Phi^{-1}\Delta D^{-1}X^*\|_F^2 \leq \|A - Z_2Z_2^\top AR^\dagger R\|_F^2$,

$$\begin{aligned}
&\|AV_RV_R^\top - C\Phi^{-1}\Delta D^{-1}X^*\|_F^2 \\
&= \|AV_RV_R^\top - C\Phi^{-1}\Delta D^{-1}Z_2^\top AV_RV_R^\top\|_F^2 && \text{by } X^* = Z_2^\top AV_RV_R^\top \\
&\leq \|A - C\Phi^{-1}\Delta D^{-1}Z_2^\top A\|_F^2 && \text{by properties of projections} \\
&\leq \|A - C\Phi^{-1}\Delta D^{-1}Z_2^\top AR^\dagger R\|_F^2 && \text{by properties of projections} \\
&= \|A - Z_2Z_2^\top AR^\dagger R\|_F^2. && \text{by } Z_2 = C\Phi^{-1}\Delta D^{-1}
\end{aligned}$$

This completes the proof. □

Lemma 21.4.19 ([CW13])**.** *Let $A \in \mathbb{R}^{n\times d}$ have rank ρ and $B \in \mathbb{R}^{n\times r}$. Let $W \in \mathbb{R}^{r\times n}$ be a randomly chosen sparse subspace embedding with $r = \Omega(\rho^2\epsilon^{-2})$. Let $\widehat{X}^* = \arg\min_{X\in\mathbb{R}^{d\times r}} \|WAX - WB\|_F^2$ and let $X^* = \arg\min_{X\in\mathbb{R}^{d\times r}} \|AX - B\|_F^2$. Then with probability at least .99,*

$$\|A\widetilde{X}^* - B\|_F^2 \leq (1+\epsilon)\|AX^* - B\|_F^2.$$

21.4.6 Column, row, and tube subset selection, $(1+\epsilon)$-approximation

Algorithm 21.8 Frobenius Norm Tensor Column, Row and Tube Subset Selection, Polynomial Time

procedure FCRTSelection(A, n, k, ϵ) ▷ Theorem 21.4.20
  $s_1 \leftarrow s_2 \leftarrow O(k/\epsilon)$.
  Choose a Gaussian matrix S_1 with s_1 columns. ▷ Definition 21.3.12
  Choose a Gaussian matrix S_2 with s_2 columns. ▷ Definition 21.3.12
  Form matrix Z_3' by setting the (i,j)-th row to be the vectorization of $(A_1S_1)_i \otimes (A_2S_2)_j$.
  $D_3 \leftarrow$GeneralizedMatrixRowSubsetSelection($A_3^\top$, $(Z_3')^\top$,n^2,n,s_1s_2,ϵ). ▷ Algorithm 21.7
  Let d_3 denote the number of nonzero entries in D_3. ▷ $d_3 = O(s_1s_2/\epsilon)$
  Form matrix Z_2' by setting the (i,j)-th row to be the vectorization of $(A_1S_1)_i \otimes (A_3S_3')_j$.
  $D_2 \leftarrow$GeneralizedMatrixRowSubsetSelection($A_2^\top$, $(Z_2')^\top$,n^2,n,s_1d_3,ϵ).
  Let d_2 denote the number of nonzero entries in D_2. ▷ $d_2 = O(s_1d_3/\epsilon)$
  Form matrix Z_1' by setting the (i,j)-th row to be the vectorization of $(A_2D_2)_i \otimes (A_3D_3)_j$.
  $D_1 \leftarrow$GeneralizedMatrixRowSubsetSelection($A_1^\top$, $(Z_1')^\top$,n^2,n,d_2d_3,ϵ).
  Let d_1 denote the number of nonzero entries in D_1. ▷ $d_1 = O(d_2d_3/\epsilon)$
  $C \leftarrow A_1D_1$, $R \leftarrow A_2D_2$ and $T \leftarrow A_3D_3$.
  return C, R and T.
end procedure

We provide two bicriteria CURT results in this Section. We first present a warm-up result. That result (Theorem 21.4.20) does not output tensor U and only guarantees that there is a rank-poly(k/ϵ) tensor U. Then we show the second result (Theorem 21.4.21), our second result is able to output tensor U. The U has rank poly(k/ϵ), but not k.

Theorem 21.4.20. *Given a 3rd order tensor $A \in \mathbb{R}^{n\times n\times n}$, for any $k \geq 1$, there exists an algorithm which takes $O(\mathrm{nnz}(A)) + n\,\mathrm{poly}(k, 1/\epsilon)$ time and outputs three matrices: $C \in \mathbb{R}^{n\times c}$, a subset of columns of A, $R \in \mathbb{R}^{n\times r}$ a subset of rows of A, and $T \in \mathbb{R}^{n\times t}$, a subset of tubes*

of A where $c = r = t = \text{poly}(k, 1/\epsilon)$, and there exists a tensor $U \in \mathbb{R}^{c \times r \times t}$ such that

$$\|(((U \cdot T^\top)^\top \cdot R^\top)^\top \cdot C^\top)^\top - A\|_F^2 \leq (1+\epsilon) \min_{\text{rank}-k \ A_k} \|A_k - A\|_F^2,$$

or equivalently,

$$\left\| \sum_{i=1}^{c} \sum_{j=1}^{r} \sum_{l=1}^{t} U_{i,j,l} \cdot C_i \otimes R_j \otimes T_l - A \right\|_F^2 \leq (1+\epsilon) \min_{\text{rank}-k \ A_k} \|A_k - A\|_F^2$$

holds with probability $9/10$.

Proof. We mainly analyze Algorithm 21.8 and it is easy to extend to Algorithm 21.9.

We fix $V^* \in \mathbb{R}^{n \times k}$ and $W^* \in \mathbb{R}^{n \times k}$. We define $Z_1 \in \mathbb{R}^{k \times n^2}$ where the i-th row of Z_1 is the vector $V_i \otimes W_i$. Choose sketching (Gaussian) matrix $S_1 \in \mathbb{R}^{n^2 \times s_1}$ (Definition 21.3.12), and let $\widehat{U} = A_1 S_1 (Z_1 S_1)^\dagger \in \mathbb{R}^{n \times k}$. Following a similar argument as in the previous theorem, we have

$$\|\widehat{U} Z_1 - A_1\|_F^2 \leq (1+\epsilon) \, \text{OPT} \, .$$

We fix $\widehat{U}$ and W^*. We define $Z_2 \in \mathbb{R}^{k \times n^2}$ where the i-th row of Z_2 is the vector $\widehat{U}_i \otimes W_i^*$. Choose sketching (Gaussian) matrix $S_2 \in \mathbb{R}^{n^2 \times s_2}$ (Definition 21.3.12), and let $\widehat{V} = A_2 S_2 (Z_2 S_2)^\dagger \in \mathbb{R}^{n \times k}$. Following a similar argument as in the previous theorem, we have

$$\|\widehat{V} Z_2 - A_2\|_F^2 \leq (1+\epsilon)^2 \, \text{OPT} \, .$$

We fix $\widehat{U}$ and $\widehat{V}$. Note that $\widehat{U} = A_1 S_1 (Z_1 S_1)^\dagger$ and $\widehat{V} = A_2 S_2 (Z_2 S_2)^\dagger$. We define $Z_3 \in \mathbb{R}^{k \times n^2}$ such that the i-th row of Z_3 is the vector $\widehat{U}_i \otimes \widehat{V}_i$. Let $z_3 = s_1 \cdot s_2$. We define

$Z_3' \in \mathbb{R}^{z_3 \times n^2}$ such that, $\forall i \in [s_1], \forall j \in [s_2]$, the $i + (j-1)s_1$-th row of Z_3' is the vector $(A_1 S_1)_i \otimes (A_2 S_2)_j$. We consider the following objective function,

$$\min_{W \in \mathbb{R}^{n \times k}, X \in \mathbb{R}^{k \times z_3}} \|WXZ_3' - A_3\|_F^2 \leq \min_{W \in \mathbb{R}^{n \times k}} \|WZ_3 - A_3\|_F^2 \leq (1+\epsilon)^2 \, \mathrm{OPT} \, .$$

Using Theorem 21.4.14, we can find a diagonal matrix $D_3 \in \mathbb{R}^{n^2 \times n^2}$ with $d_3 = O(z_3/\epsilon) = O(k^2/\epsilon^3)$ nonzero entries such that

$$\min_{X \in \mathbb{R}^{d_3 \times z_3}} \|A_3 D_3 X Z_3' - A_3\|_F^2 \leq (1+\epsilon)^3 \, \mathrm{OPT} \, .$$

In the following, we abuse notation and let $A_3 D_3 \in \mathbb{R}^{n \times d_3}$ by deleting zero columns. Let W' denote $A_3 D_3 \in \mathbb{R}^{n \times d_3}$. Then,

$$\min_{X \in \mathbb{R}^{d_3 \times z_3}} \|W' X Z_3' - A_3\|_F^2 \leq (1+\epsilon)^3 \, \mathrm{OPT} \, .$$

We fix $\widehat{U}$ and W'. Let $z_2 = s_1 \cdot d_3$. We define $Z_2' \in \mathbb{R}^{z_2 \times n^2}$ such that, $\forall i \in [s_1], \forall j \in [d_3]$, the $i + (j-1)s_1$-th row of Z_2' is the vector $(A_1 S_1)_i \otimes (A_3 D_3)_j$.

Using Theorem 21.4.14, we can find a diagonal matrix $D_2 \in \mathbb{R}^{n^2 \times n^2}$ with $d_2 = O(z_2/\epsilon) = O(s_1 d_3/\epsilon) = O(k^3/\epsilon^5)$ nonzero entries such that

$$\min_{X \in \mathbb{R}^{d_2 \times z_2}} \|A_2 D_2 X Z_2' - A_2\|_F^2 \leq (1+\epsilon)^4 \, \mathrm{OPT} \, .$$

Let V' denote $A_2 D_2$. Then,

$$\min_{X \in \mathbb{R}^{d_2 \times z_2}} \|V' X Z_2' - A_2\|_F^2 \leq (1+\epsilon)^4 \, \mathrm{OPT} \, .$$

We fix V' and W'. Let $z_1 = d_2 \cdot d_3$. We define $Z_1' \in \mathbb{R}^{z_1 \times n^2}$ such that, $\forall i \in [d_2], \forall j \in [d_3]$, the $i + (j-1)s_1$-th row of Z_1' is the vector $(A_2 D_2)_i \otimes (A_3 D_3)_j$.

Using Theorem 21.4.14, we can find a diagonal matrix $D_1 \in \mathbb{R}^{n^2 \times n^2}$ with $d_1 = O(z_1/\epsilon) = O(d_2 d_3/\epsilon) = O(k^5/\epsilon^9)$ nonzero entries such that

$$\min_{X \in \mathbb{R}^{d_1 \times z_1}} \|A_1 D_1 X Z_1' - A_1\|_F^2 \leq (1+\epsilon)^5 \operatorname{OPT}.$$

Let U' denote $A_1 D_1$. Then,

$$\min_{X \in \mathbb{R}^{d_1 \times z_1}} \|U' X Z_1' - A_1\|_F^2 \leq (1+\epsilon)^5 \operatorname{OPT}.$$

Putting U', V', W' all together, we complete the proof.

All the above analysis gives the running time $O(\operatorname{nnz}(A)) \log n + n^2 \operatorname{poly}(\log n, k, 1/\epsilon)$. To improve the running time, we need to use Algorithm 21.9, the similar analysis will go through, the running time will be improved to $O(\operatorname{nnz}(A) + n \operatorname{poly}(k, 1/\epsilon))$, but the sample complexity of c, r, k will be slightly worse (poly log factors). □

Theorem 21.4.21. *Given a 3rd order tensor $A \in \mathbb{R}^{n \times n \times n}$, for any $k \geq 1$, there exists an algorithm which takes $O(\operatorname{nnz}(A) + n \operatorname{poly}(k, 1/\epsilon))$ time and outputs three matrices: $C \in \mathbb{R}^{n \times c}$, a subset of columns of A, $R \in \mathbb{R}^{n \times r}$ a subset of rows of A, and $T \in \mathbb{R}^{n \times t}$, a subset of tubes of A, together with a tensor $U \in \mathbb{R}^{c \times r \times t}$ with $\operatorname{rank}(U) = k'$ where $c = r = t = \operatorname{poly}(k, 1/\epsilon)$ and $k' = \operatorname{poly}(k, 1/\epsilon)$ such that*

$$\|U(C, R, T) - A\|_F^2 \leq (1+\epsilon) \min_{\operatorname{rank}-k \ A_k} \|A_k - A\|_F^2,$$

or equivalently,

$$\left\| \sum_{i=1}^{c} \sum_{j=1}^{r} \sum_{l=1}^{t} U_{i,j,l} \cdot C_i \otimes R_j \otimes T_l - A \right\|_F^2 \leq (1+\epsilon) \min_{\operatorname{rank}-k \ A_k} \|A_k - A\|_F^2$$

holds with probability $9/10$.

Proof. The proof follows by combining Theorem 21.1.1 and Theorem 21.1.3 directly. □

Algorithm 21.9 Frobenius Norm Tensor Column, Row and Tube Subset Selection, Input Sparsity Time

procedure FCRTSELECTION(A, n, k, ϵ) ▷ Theorem 21.4.20
  $s_1 \leftarrow s_2 \leftarrow O(k/\epsilon)$.
  $\epsilon_0 \leftarrow 0.001$.
  Choose a Gaussian matrix S_1 with s_1 columns. ▷ Definition 21.3.12
  Choose a Gaussian matrix S_2 with s_2 columns. ▷ Definition 21.3.12
  Form matrix B_1 by setting (i,j)-th column to be $(A_1 S_1)_i$.
  Form matrix B_2 by setting (i,j)-th column to be $(A_2 S_2)_j$. ▷ $Z_3' = B_1^\top \odot B_2^\top$
  $d_3 \leftarrow O(s_1 s_2 \log(s_1 s_2) + (s_1 s_2/\epsilon))$.
  $D_3 \leftarrow$FASTTENSORLEVERAGESCOREGENERALORDER$(B_1^\top, B_2^\top, n, n, s_1 s_2, \epsilon_0, d_1)$. ▷ Algorithm 21.15
  Form matrix B_1 by setting (i,j)-th column to be $(A_1 S_1)_i$.
  Form matrix B_3 by setting (i,j)-th column to be $(A_3 D_3)_j$. ▷ $Z_2' = B_1^\top \odot B_3^\top$
  $d_2 \leftarrow O(s_1 d_3 \log(s_1 d_3) + (s_1 d_3/\epsilon))$.
  $D_2 \leftarrow$FASTTENSORLEVERAGESCOREGENERALORDER$(B_1^\top, B_3^\top, n, n, s_1 d_3, \epsilon_0, d_2)$.
  Form matrix B_2 by setting (i,j)-th column to be $(A_2 D_2)_i$.
  Form matrix B_3 by setting (i,j)-th column to be $(A_3 D_3)_j$. ▷ $Z_1' = B_2^\top \odot B_3^\top$
  $d_1 \leftarrow O(d_2 d_3 \log(d_2 d_3) + (d_2 d_3/\epsilon))$.
  $D_1 \leftarrow$FASTTENSORLEVERAGESCOREGENERALORDER$(B_2^\top, B_3^\top, n, n, d_2 d_3, \epsilon_0, d_1)$.
  $C \leftarrow A_1 D_1$, $R \leftarrow A_2 D_2$ and $T \leftarrow A_3 D_3$.
  return C, R and T.
end procedure

21.4.7 CURT decomposition, $(1+\epsilon)$-approximation

21.4.7.1 Properties of leverage score sampling and BSS sampling

Notice that, the BSS algorithm is a deterministic procedure developed in [BSS12] for selecting rows from a matrix $A \in \mathbb{R}^{n \times d}$ (with $\|A\|_2 \leq 1$ and $\|A\|_F^2 \leq k$) using a selection matrix S so that

$$\|A^\top S^\top S A - A^\top A\|_2 \leq \epsilon.$$

The algorithm runs in $\text{poly}(n, d, 1/\epsilon)$ time. Using the ideas from [BW14] and [CEM+15], we are able to reduce the number of nonzero entries from $O(\epsilon^{-2}k \log k)$ to $O(\epsilon^{-2}k)$, and also improve the running time to input sparsity.

Lemma 21.4.22 (Leverage score preserves subspace embedding - Theorem 2.11 in [Woo14b])**.** *Given a* rank-k *matrix* $A \in \mathbb{R}^{n \times d}$*, via leverage score sampling, we can obtain a diagonal matrix* D *with* m *nonzero entries such that, letting* $B = DA$*, if* $m = O(\epsilon^{-2}k \log k)$*, then, with probability at least* 0.999*, for all* $x \in \mathbb{R}^d$*,*

$$(1 - \epsilon)\|Ax\|_2 \leq \|Bx\|_2 \leq (1 + \epsilon)\|Ax\|_2$$

Lemma 21.4.23. *Given a* rank-k *matrix* $A \in \mathbb{R}^{n \times d}$*, there exists an algorithm that runs in* $O(\text{nnz}(A) + n \,\text{poly}(k, 1/\epsilon))$ *time and outputs a matrix* B *containing* $O(\epsilon^{-2}k \log k)$ *re-weighted rows of* A*, such that with probability at least* 0.999*, for all* $x \in \mathbb{R}^d$*,*

$$(1 - \epsilon)\|Ax\|_2 \leq \|Bx\|_2 \leq (1 + \epsilon)\|Ax\|_2$$

Proof. We choose a sparse embedding matrix (Definition 21.3.10) $\Pi \in \mathbb{R}^{d \times s}$ with $s = \text{poly}(k/\epsilon)$. With probability at least 0.999, $\Pi^\top$ is a subspace embedding of $A^\top$. Thus, $\text{rank}(A\Pi) = \text{rank}(A)$. Also, the leverage scores of $A\Pi$ are the same as those of A. Thus, we can compute the leverage scores of $A\Pi$. The running time of computing $A\Pi$ is $O(\text{nnz}(A))$. Thus the total running time is $O(\text{nnz}(A) + n \,\text{poly}(k, 1/\epsilon))$. □

Lemma 21.4.24. *Let* B *denote a matrix which contains* $O(\epsilon^{-2}k \log k)$ *rows of* $A \in \mathbb{R}^{n \times d}$*. Choosing* Π *to be a sparse subspace embedding matrix of size* $d \times O(\epsilon^{-6}(k \log k)^2)$*, with probability at least* 0.999*,*

$$\|B\Pi\Pi^\top B^\top - BB^\top\|_2 \leq \epsilon\|B\|_2^2.$$

Combining Lemma 21.4.23, 21.4.24 and the BSS algorithm, we obtain:

Lemma 21.4.25. *Given a* rank-k *matrix* $A \in \mathbb{R}^{n \times d}$*, there exists an algorithm that runs in* $O(\mathrm{nnz}(A) + n\,\mathrm{poly}(k, 1/\epsilon))$ *time and outputs a sampling and rescaling diagonal matrix* S *that selects* $O(\epsilon^{-2}k)$ *re-weighted rows of* A*, such that, with probability at least* 0.999*,*

$$\|A^\top S^\top SA - A^\top A\|_2 \leq \epsilon \|A\|_2^2.$$

or equivalently, for all $x \in \mathbb{R}^d$*,*

$$(1-\epsilon)\|Ax\|_2 \leq \|SAx\|_2 \leq (1+\epsilon)\|Ax\|_2.$$

Proof. Using Lemma 21.4.23, we can obtain B. Then we apply a sparse subspace embedding matrix Π on the right of B. At the end, we run the BSS algorithm on $B\Pi$ and we are able to output $O(\epsilon^{-2}k)$ re-weighted rows of $B\Pi$. Using these rows, we are able to determine $O(\epsilon^{-2}k)$ re-weighted rows of A. □

21.4.7.2 Row sampling for linear regression

Theorem 21.4.26 (Theorem 5 in [CNW15])**.** *We are given* $A \in \mathbb{R}^{n \times d}$ *with* $\|A\|_2^2 \leq 1$ *and* $\|A\|_F^2 \leq k$*, and an* $\epsilon \in (0, 1)$*. There exists a diagonal matrix* S *with* $O(k/\epsilon^2)$ *nonzero entries such that*

$$\|(SA)^\top SA - A^\top A\|_2 \leq \epsilon.$$

Corollary 21.4.27. *Given a rank-*k *matrix* $A \in \mathbb{R}^{n \times d}$*, vector* $b \in \mathbb{R}^n$*, and parameter* $\epsilon > 0$*, let* $U \in \mathbb{R}^{n \times (k+1)}$ *denote an orthonormal basis of* $[A, b]$*. Let* $S \in \mathbb{R}^{n \times n}$ *denote a sampling and rescaling diagonal matrix according to Leverage score sampling and sparse BSS sampling of*

U with m nonzero entries. If $m = O(k)$, then S is a $(1 \pm 1/2)$ subspace embedding for U; if $m = O(k/\epsilon)$, then S satisfies $\sqrt{\epsilon}$-operator norm approximate matrix product for U.

Proof. This follows by Lemma 21.4.22, Lemma 21.4.24 and Theorem 21.4.26. □

Lemma 21.4.28 ([NW14])**.** *Given $A \in \mathbb{R}^{n \times d}$ and $b \in \mathbb{R}^n$, let $S \in \mathbb{R}^{n \times n}$ denote a sampling and rescaling diagonal matrix. Let x^* denote $\arg\min_x \|Ax - b\|_2^2$ and x' denote $\arg\min_x \|SAx - Sb\|_2^2$. If S is a $(1 \pm 1/2)$ subspace embedding for the column span of A, and ϵ' $(=\sqrt{\epsilon})$-operator norm approximate matrix product for U adjoined with $b - Ax^*$, then, with probability at least .999,*

$$\|Ax' - b\|_2^2 \leq (1 + \epsilon)\|Ax^* - b\|_2^2.$$

Proof. We define $\text{OPT} = \min_x \|Ax - b\|_2$. We define $x' = \arg\min_x \|SAx - Sb\|_2^2$ and $x^* = \arg\min_x \|Ax - b\|_2^2$. Let $w = b - Ax^*$. Let U denote an orthonormal basis of A. We can write $Ax' - Ax^* = U\beta$. Then, we have,

$$\begin{aligned}
\|Ax' - b\|_2^2 &= \|Ax' - Ax^* + AA^\dagger b - b\|_2^2 && \text{by } x^* = A^\dagger b \\
&= \|U\beta + (UU^\top - I)b\|_2^2 \\
&= \|Ax^* - Ax'\|_2^2 + \|Ax^* - b\|_2^2 && \text{by Pythagorean Theorem} \\
&= \|U\beta\|_2^2 + \text{OPT}^2 \\
&= \|\beta\|_2^2 + \text{OPT}^2 \,.
\end{aligned}$$

If S is a $(1 \pm 1/2)$ subspace embedding for U, then we can show

$$
\begin{aligned}
&\|\beta\|_2 - \|U^\top S^\top SU\beta\|_2 \\
&\leq \|\beta - U^\top S^\top SU\beta\|_2 && \text{by triangle inequality} \\
&= \|(I - U^\top S^\top SU)\beta\|_2 \\
&\leq \|I - U^\top S^\top SU\|_2 \cdot \|\beta\|_2 \\
&\leq \frac{1}{2}\|\beta\|_2.
\end{aligned}
$$

Thus, we obtain

$$\|U^\top S^\top SU\beta\|_2 \geq \|\beta\|_2/2.$$

Next, we can show

$$
\begin{aligned}
\|U^\top S^\top SU\beta\|_2 &= \|U^\top S^\top S(Ax' - Ax^*)\|_2^2 \\
&= \|U^\top S^\top S(A(SA)^\dagger Sb - Ax^*)\|_2 && \text{by } x' = (SA)^\dagger Sb \\
&= \|U^\top S^\top S(b - Ax^*)\|_2 && \text{by } SA(SA)^\dagger = I \\
&= \|U^\top S^\top Sw\|_2. && \text{by } w = b - Ax^*
\end{aligned}
$$

We define $U' = \begin{bmatrix} U & w/\|w\|_2 \end{bmatrix}$. We define X and y to satisfy $U = U'X$ and $w = U'y$. Then,

we have

$$\begin{aligned}
&\|U^\top S^\top S w\|_2 \\
&= \|U^\top S^\top S w - U^\top w\|_2 && \text{by } U^\top w = 0 \\
&= \|X^\top U'^\top S^\top S U' y - X^\top U'^\top U' y\|_2 \\
&= \|X^\top (U'^\top S^\top S U' - I) y\|_2 \\
&\le \|X\|_2 \cdot \|U'^\top S^\top S U' - I\|_2 \cdot \|y\|_2 \\
&\le \epsilon' \|X\|_2 \|y\|_2 \\
&= \epsilon' \|U\|_2 \|w\|_2 \\
&= \epsilon' \operatorname{OPT}, && \text{by } \|U\|_2 = 1 \text{ and } \|w\|_2 = \operatorname{OPT}
\end{aligned}$$

where the fifth inequality follows since S satisfies ϵ'-operator norm approximate matrix product for the column span of U adjoined with w.

Putting it all together, we have

$$\begin{aligned}
\|Ax' - b\|_2^2 &= \|Ax^* - b\|_2^2 + \|Ax^* - Ax'\|_2^2 \\
&= \operatorname{OPT}^2 + \|\beta\|_2^2 \\
&\le \operatorname{OPT}^2 + 4\|U^\top S^\top S w\|_2^2 \\
&\le \operatorname{OPT}^2 + 4(\epsilon' \operatorname{OPT})^2 \\
&\le (1+\epsilon) \operatorname{OPT}^2 . && \text{by } \epsilon' = \frac{1}{2}\sqrt{\epsilon}.
\end{aligned}$$

Finally, note that S satisfies ϵ'-operator norm approximate matrix product for U adjoined with w if it is a $(1 \pm \epsilon')$-subspace embedding for U adjoined with w, which holds using BSS sampling by Theorem 5 of [CNW15] with $O(d/\epsilon)$ samples. □

21.4.7.3 Leverage scores for multiple regression

Lemma 21.4.29 (see, e.g., Lemma 32 in [CW13] among other places)**.** *Given matrix $A \in \mathbb{R}^{n \times d}$ with orthonormal columns, and parameter $\epsilon > 0$, if $S \in \mathbb{R}^{n \times n}$ is a sampling and rescaling diagonal matrix according to the leverage scores of A where the number of nonzero entries is $t = O(1/\epsilon^2)$, then, for any $B \in \mathbb{R}^{n \times m}$, we have*

$$\|A^\top S^\top S B - A^\top B\|_F^2 < \epsilon^2 \|A\|_F^2 \|B\|_F^2,$$

holds with probability at least 0.9999*.*

Corollary 21.4.30. *Given matrix $A \in \mathbb{R}^{n \times d}$ with orthonormal columns, and parameter $\epsilon > 0$, if $S \in \mathbb{R}^{n \times n}$ is a sampling and rescaling diagonal matrix according to the leverage scores of A with m nonzero entries, then if $m = O(d \log d)$, then S is a $(1 \pm 1/2)$ subspace embedding for A. If $m = O(d/\epsilon)$, then S satisfies $\sqrt{\epsilon/d}$-Frobenius norm approximate matrix product for A.*

Proof. This follows by Lemma 21.4.22 and Lemma 21.4.29. □

Lemma 21.4.31 ([NW14])**.** *Given $A \in \mathbb{R}^{n \times d}$ and $B \in \mathbb{R}^{n \times m}$, let $S \in \mathbb{R}^{n \times n}$ denote a sampling and rescaling matrix according to A. Let X^* denote $\arg\min_X \|AX - B\|_F^2$ and X' denote $\arg\min_X \|SAX - SB\|_F^2$. Let U denote an orthonormal basis for A. If S is a $(1 \pm 1/2)$ subspace embedding for U, and satisfies $\epsilon' (= \sqrt{\epsilon/d})$-Frobenius norm approximate matrix product for U, then, we have that*

$$\|AX' - B\|_F^2 \leq (1 + \epsilon)\|AX^* - B\|_F^2$$

holds with probability at least 0.999*.*

Proof. We define $\mathrm{OPT} = \min_X \|AX - B\|_F$. Let $A = U\Sigma V^\top$ denote the SVD of A. Since A has rank k, U and V have k columns. We can write $A(X' - X^*) = U\beta$. Then, we have

$$\begin{aligned}
\|AX' - B\|_F^2 &= \|AX' - AX^* + AA^\dagger B - B\|_F^2 && \text{by } X^* = A^\dagger B \\
&= \|U\beta + (UU^\top - I)B\|_F^2 \\
&= \|AX^* - AX'\|_F^2 + \|AX^* - B\|_F^2 && \text{by Pythagorean Theorem} \\
&= \|U\beta\|_F^2 + \mathrm{OPT}^2 \\
&= \|\beta\|_F^2 + \mathrm{OPT}^2 \,. && (21.15)
\end{aligned}$$

If S is a $(1 \pm 1/2)$ subspace embedding for U, then we can show,

$$\begin{aligned}
&\|\beta\|_F - \|U^\top S^\top SU\beta\|_F \\
\leq\ &\|\beta - U^\top S^\top SU\beta\|_F && \text{by triangle inequality} \\
=\ &\|(I - U^\top S^\top SU)\beta\|_F \\
\leq\ &\|(I - U^\top S^\top SU)\|_2 \cdot \|\beta\|_F && \text{by } \|AB\|_F \leq \|A\|_2 \|B\|_F \\
\leq\ &\frac{1}{2}\|\beta\|_F. && \text{by } \|(I - U^\top S^\top SU)\|_2 \leq 1/2
\end{aligned}$$

Thus, we obtain

$$\|U^\top S^\top SU\beta\|_F \geq \|\beta\|_F/2. \tag{21.16}$$

Next, we can show

$$\begin{aligned}
\|U^\top S^\top SU\beta\|_F &= \|U^\top S^\top S(AX' - AX^*)\|_F \\
&= \|U^\top S^\top S(A(SA)^\dagger Sb - AX^*)\|_F && \text{by } X' = (SA)^\dagger SB \\
&= \|U^\top S^\top S(B - AX^*)\|_F. && \text{by } SA(SA)^\dagger = I
\end{aligned}$$

Then, we can show

$$\begin{aligned}\|U^\top S^\top S(B - AX^*)\|_F &\leq \epsilon' \|U^\top\|_F \|B - AX^*\|_F \\ &= \epsilon' \sqrt{d}\, \mathrm{OPT}, \end{aligned} \tag{21.17}$$

where the first step follows from Lemma 21.4.29, the second step follows from $\|U\|_F = \sqrt{d}$ and $\|B - AX^*\|_F = \mathrm{OPT}$.

Putting it all together, we have

$$\begin{aligned}\|AX' - B\|_F^2 &= \|AX^* - B\|_F^2 + \|AX^* - AX'\|_F^2 \\ &= \mathrm{OPT}^2 + \|\beta\|_F^2 && \text{by Equation (21.15)} \\ &\leq \mathrm{OPT}^2 + 4\|U^\top S^\top S w\|_F^2 && \text{by Equation (21.16)} \\ &\leq \mathrm{OPT}^2 + 4(\epsilon' \sqrt{d}\, \mathrm{OPT})^2 && \text{by Equation (21.17)} \\ &\leq (1+\epsilon)\, \mathrm{OPT}^2 . && \text{by } \epsilon' = \frac{1}{2}\sqrt{\epsilon/d} \end{aligned}$$

□

21.4.7.4 Sampling columns according to leverage scores implicitly, improving polynomial running time to nearly linear running time

This section explains an algorithm that is able to sample from the leverage scores from the $\odot$ product of two matrices U, V without explicitly writing down $U \odot V$. To build this algorithm we combine TensorSketch, some ideas from [DMIMW12] and some ideas from [AKO11, MW10]. Finally, we are able to improve the running time of sampling columns according to leverage scores from $\Omega(n^2)$ to $\widetilde{O}(n)$. Given two matrices $U, V \in \mathbb{R}^{k \times n}$, we define $A \in \mathbb{R}^{k \times n_1 n_2}$ to be the matrix where the i-th row of A is the vectorization of $U^i \otimes V^i$, $\forall i \in [k]$.

Naïvely, in order to sample $O(\mathrm{poly}(k, 1/\epsilon))$ rows from $A^\top$ according to leverage scores, we need to write down n^2 leverage scores. This approach will take at least $\Omega(n^2)$ running time. In the rest of this section, we will explain how to do it in $O(n \cdot \mathrm{poly}(\log n, k, 1/\epsilon))$ time. In Section 21.5.1, we will explain how to extend this idea from 3rd order tensors to general q-th order tensors and remove the $\mathrm{poly}(\log n)$ factor from running time, i.e., obtain $O(n \cdot \mathrm{poly}(k, 1/\epsilon))$ time.

Lemma 21.4.32. *Given two matrices $U \in \mathbb{R}^{k \times n_1}$ and $V \in \mathbb{R}^{k \times n_2}$, there exists an algorithm that takes $O((n_1 + n_2) \cdot \mathrm{poly}(\log(n_1 n_2), k) \cdot R_{\mathrm{samples}})$ time and samples R_{samples} columns of $U \odot V \in \mathbb{R}^{k \times n_1 n_2}$ according to the leverage scores of $R^{-1}(U \odot V)$, where R is the R of a QR factorization.*

Proof. We choose $\Pi \in \mathbb{R}^{n_1 n_2 \times s_1}$ to be a TensorSketch. Then, according to Section 21.3.10, we can compute R^{-1} in $n \cdot \mathrm{poly}(\log n, k, 1/\epsilon)$ time, where R is the R in a QR-factorization. We want to sample columns from $U \odot V$ according to the square of the ℓ_2-norms of each column of $R^{-1}(U \odot V)$. However, explicitly writing down the matrix $R^{-1}(U \odot V)$ takes $k n_1 n_2$ time, and the number of columns is already $n_1 n_2$. The goal is to sample columns from $R^{-1}(U \odot V)$ without explicitly computing the square of the ℓ_2-norm of each column.

The first simple observation is that the following two sampling procedures are equivalent in terms of the column samples of a matrix that they take. (1) We sample a single entry from the matrix $R^{-1}(U \odot V)$ proportional to its squared value. (2) We sample a column from the matrix $R^{-1}(U \odot V)$ proportional to its squared ℓ_2-norm. Let the (i, j_1, j_2)-th entry denote the entry in the i-th row and the $(j_1 - 1)n_2 + j_2$-th column. We can show, for a

Algorithm 21.10 Fast Tensor Leverage Score Sampling

procedure FastTensorLeverageScore($U, V, n_1, n_2, k, \epsilon, R_{\text{samples}}$) ▷ Lemma 21.4.32
  $s_1 \leftarrow \text{poly}(k, 1/\epsilon)$.
  $g_1 \leftarrow g_2 \leftarrow g_3 \leftarrow O(\epsilon^{-2} \log(n_1 n_2))$.
  Choose $\Pi \in \mathbb{R}^{n_1 n_2 \times s_1}$ to be a TensorSketch. ▷ Definition 21.3.20
  Compute $R^{-1} \in \mathbb{R}^{k \times k}$ by using $(U \odot V)\Pi$. ▷ $U \in \mathbb{R}^{k \times n_1}$, $V \in \mathbb{R}^{k \times n_2}$
  Choose $G_1 \in \mathbb{R}^{g_1 \times k}$ to be a Gaussian sketching matrix.
  for $i = 1 \to g_1$ **do**
    $w \leftarrow (G^i R^{-1})^\top$ ▷ G^i denotes the i-th row of G
    for $j = 1 \to [n_1]$ **do** ▷ Form matrix $U'^i \in \mathbb{R}^{k \times n_1}$
      $U'^i_j \leftarrow w \circ U_j, \forall j \in [n_1]$. ▷ U_j denotes the j-th column of $U \in \mathbb{R}^{k \times n_1}$
    end for
  end for
  Choose $G_{2,i} \in \mathbb{R}^{g_2 \times n_1}$ to be a Gaussian sketching matrix.
  for $i = 1 \to g_1$ **do**
    $\alpha_i \leftarrow \|(G_{2,i} U'^{i\top}) V\|_F^2$.
    Choose $G_{3,i} \in \mathbb{R}^{g_3 \times n_1}$ to be a Gaussian sketching matrix.
    for $j_2 = 1 \to n_2$ **do**
      $\beta_{i,j} \leftarrow \|G_{3,i}(U'^{i\top}) V_{j_2}\|_2^2$.
    end for
  end for
  $\mathcal{S} \leftarrow \emptyset$.
  for $r = 1 \to R_{\text{samples}}$ **do**
    Sample i from $[g_1]$ with probability $\alpha_i / \sum_{i'=1}^{g_1} \alpha_{i'}$.
    Sample j_2 from $[n_2]$ with probability $\beta_{i,j_2} / \sum_{j_2'=1}^{n_2} \beta_{i,j_2'}$.
    for $j_1 = 1 \to n_1$ **do**
      $\gamma_{j_1} \leftarrow ((U'^{i\top})^{j_1} V_{j_2})^2$.
    end for
    Sample j_1 from $[n_1]$ with probability $\gamma_{j_1} / \sum_{j_1'=1}^{n_1} \gamma_{j_1'}$.
    $\mathcal{S} \leftarrow \mathcal{S} \cup (j_1, j_2)$.
  end for
  Convert $\mathcal{S}$ into a diagonal matrix D with at most R_{samples} nonzero entries.
  return D. ▷ Diagonal matrix $D \in \mathbb{R}^{n_1 n_2 \times n_1 n_2}$
end procedure

particular column $(j_1-1)n_2+j_2$,

$$\begin{aligned}
&\Pr[\text{sample an entry from the } (j_1-1)n_2+j_2 \text{ th column of a matrix}] \\
=& \sum_{i=1}^{k} \Pr[\text{sample the } (i,j_1,j_2)\text{-th entry of matrix}] \\
=& \sum_{i=1}^{k} \frac{|(R^{-1}(U\odot V))_{i,(j_1-1)n_2+j_2}|^2}{\|R^{-1}(U\odot V)\|_F^2} \\
=& \frac{\|(R^{-1}(U\odot V))_{(j_1-1)n_2+j_2}\|^2}{\|R^{-1}(U\odot V)\|_F^2} \\
=& \Pr[\text{sample the } (j_1-1)n_2+j_2 \text{ th column of matrix}]. \qquad (21.18)
\end{aligned}$$

Thus, it is sufficient to show how to sample a single entry from matrix $R^{-1}(U\odot V)$ proportional to its squared value without writing down all of the entries of a $k\times n_1n_2$ matrix.

We choose a Gaussian matrix $G_1\in\mathbb{R}^{g_1\times k}$ with $g_1=O(\epsilon^{-2}\log(n_1n_2))$. By Claim 21.4.33 we can reduce the length of each column vector of matrix $R^{-1}U\odot V$ from k to g_1 while preserving the squared ℓ_2-norm of all columns simultaneously. Thus, we obtain a new matrix $GR^{-1}(U\odot V)\in\mathbb{R}^{g_1\times n_1n_2}$, and sampling from this new matrix is equivalent to sampling from the original matrix $R^{-1}(U\odot V)$.

In the following paragraphs, we explain a sampling procedure (also described in Procedure FastTensorLeverageScore in Algorithm 21.10) which contains three sampling steps. The first step is sampling i from $[g_1]$, the second step is sampling j_2 from $[n_2]$, and the last step is sampling j_1 from $[n_1]$.

For each $j_1\in[n_1]$, let U_{j_1} denote the j_1-th column of U. For each $i\in[g_1]$, let G_1^i denote the i-th row of matrix $G_1\in\mathbb{R}^{g_1\times k}$, let $U'^i\in\mathbb{R}^{k\times n_1}$ denote a matrix where the j_1-th column is $(G^iR^{-1})^\top\circ U_{j_1}\in\mathbb{R}^k, \forall j\in[n_1]$. Then, using Claim 21.4.37, we have that $(G^iR^{-1})\cdot(U\odot V)\in$

$\mathbb{R}^{n_1 n_2}$ is a row vector where the entry in the $(j_1-1)n_2+j_2$-th coordinate is the entry in the j_1-th row and j_2-th column of matrix $(U'^{i\top}V) \in \mathbb{R}^{n_1 \times n_2}$. Further, the squared ℓ_2-norm of vector $(G^i R^{-1}) \cdot (U \odot V)$ is equal to the squared Frobenius norm of matrix $(U'^{i\top}V)$. Thus, sampling i proportional to the squared ℓ_2-norm of vector $(G^i R^{-1}) \cdot (U \odot V)$ is equivalent to sampling i proportional to the squared Frobenius norm of matrix $(U'^{i\top}V)$. Naïvely, computing the Frobenius norm of an $n_1 \times n_2$ matrix requires $O(n_1 n_2)$ time. However, we can choose a Gaussian matrix $G_{2,i} \in \mathbb{R}^{g_2 \times n_1}$ to sample according to the value $\|(G_{2,i}U'^{i\top})V\|_F^2$, which can be computed in $O((n_1+n_2)g_2 k)$ time. By claim 21.4.35, $\|(G_{2,i}U'^{i\top})V\|_F^2 \approx \|(U'^{i\top})V\|_F^2$ with high probability. So far, we have finished the first step of the sampling procedure.

For the second step of the sampling procedure, we need to sample j_2 from $[n_2]$. To do that, we need to compute the squared ℓ_2-norm of each column of $U'^{i\top}V \in \mathbb{R}^{n_1 \times n_2}$. This can be done by choosing another Gaussian matrix $G_{3,i} \in \mathbb{R}^{g_3 \times n_1}$. For all $j_2 \in [n_2]$, by Claim 21.4.36, we have $\|G_{3,i}U'^{i\top}V_{j_2}\|_2^2 \approx \|U'^{i\top}V_{j_2}\|_2^2$. Also, for $j_2 \in [n_2]$, $\|G_{3,i}U'^{i\top}V_{j_2}\|_2^2$ can be computed in nearly linear in n_1+n_2 time.

For the third step of the sampling procedure, we need to sample j_1 from $[n_1]$. Since we already have i and j_2 from the previous two steps, we can directly compute $|(U'^{i\top})^{j_1}V_{j_2}|^2$, for all j_1. This only takes $O(n_1 k)$ time.

Overall, the running time is $O((n_1+n_2)\cdot \mathrm{poly}(\log(n_1 n_2), k, 1/\epsilon))$. Because our estimates are accurate enough, our sampling probabilities are also good approximations to the leverage score sampling probabilities. Putting it all together, we complete the proof. □

Claim 21.4.33. *Given matrix $R^{-1}(U \odot V) \in \mathbb{R}^{k \times n_1 n_2}$, let $G_1 \in \mathbb{R}^{g_1 \times k}$ denote a Gaussian matrix with $g_1 = (\epsilon^{-2}\log(n_1 n_2))$. Then with probability at least $1 - 1/\mathrm{poly}(n_1 n_2)$, we have:*

for all $j \in [n_1 n_2]$,

$$(1-\epsilon)\|R^{-1}(U \odot V)_j\|_2^2 \leq \|G_1 R^{-1}(U \odot V)_j\|_2^2 \leq (1+\epsilon)\|R^{-1}(U \odot V)_j\|_2^2.$$

Proof. This follows by the Johnson-Lindenstrauss Lemma. □

Claim 21.4.34. *For a fixed* $i \in [g_1]$, *let* $G_{2,i} \in \mathbb{R}^{g_2 \times n_1}$ *denote a Gaussian matrix with* $g_2 = O(\epsilon^{-2}\log(n_1 n_2))$. *Then with probability at least* $1 - 1/\operatorname{poly}(n_1 n_2)$, *we have: for all* $j_2 \in [n_2]$,

$$(1-\epsilon)\|U'^{i\top} V_{j_2}\|_2^2 \leq \|(G_{2,i} U'^{i\top}) V_{j_2}\|_2 \leq (1+\epsilon)\|U'^{i\top} V_{j_2}\|_2^2.$$

By taking the union bound over all $i \in [g_1]$, we obtain a stronger claim,

Claim 21.4.35. *With probability at least* $1 - 1/\operatorname{poly}(n_1 n_2)$, *we have : for all* $i \in [g_1]$, *for all* $j_2 \in [n_2]$,

$$(1-\epsilon)\|U'^{i\top} V_{j_2}\|_2^2 \leq \|(G_{2,i} U'^{i\top}) V_{j_2}\|_2 \leq (1+\epsilon)\|U'^{i\top} V_{j_2}\|_2^2.$$

Similarly, if we choose $G_{3,i}$ to be a Gaussian matrix, we can obtain the same result as for $G_{2,i}$:

Claim 21.4.36. *With probability at least* $1 - 1/\operatorname{poly}(n_1 n_2)$, *we have : for all* $i \in [g_1]$, *for all* $j_2 \in [n_2]$,

$$(1-\epsilon)\|U'^{i\top} V_{j_2}\|_2^2 \leq \|(G_{3,i} U'^{i\top}) V_{j_2}\|_2 \leq (1+\epsilon)\|U'^{i\top} V_{j_2}\|_2^2.$$

Claim 21.4.37. *For any* $i \in [g_1]$, $j_1 \in [n_1]$, $j_2 \in [n_2]$, *let* G_1^i *denote the* i*-th row of matrix* $G_1 \in \mathbb{R}^{g_1 \times k}$. *Let* $(U \odot V)_{(j_1-1)n_2+j_2}$ *denote the* $(j_1-1)n_2 + j_2$*-th column of matrix* $\mathbb{R}^{k \times n_1 n_2}$.

Let $(U'^{i\top})^{j_1}$ *denote the* j_1*-th row of matrix* $(U'^{i\top}) \in \mathbb{R}^{n_1 \times k}$*. Let* V_{j_2} *denote the* j_2*-th column of matrix* $V \in \mathbb{R}^{k \times n_2}$*. Then, we have*

$$G_1^i R^{-1} (U \odot V)_{(j_1-1)n_2+j_2} = (U'^{i\top})^{j_1} V_{j_2}.$$

Proof. This follows by,

$$G_1^i R^{-1} (U \odot V)_{(j_1-1)n_2+j_2} = G_1^i R^{-1} (U_{j_1} \circ V_{j_2}) = (G_1^i R^{-1} \circ (U_{j_1})^\top) V_{j_2} = (U'^{i\top})^{j_1} V_{j_2}.$$

□

Lemma 21.4.38. *Given* $A \in \mathbb{R}^{n \times n^2}$*,* $V, W \in \mathbb{R}^{k \times n}$*, for any* $\epsilon > 0$*, there exists an algorithm that runs in* $O(n \cdot \mathrm{poly}(k, 1/\epsilon))$ *time and outputs a diagonal matrix* $D \in \mathbb{R}^{n^2 \times n^2}$ *with* $m = O(k \log k + k/\epsilon)$ *nonzero entries such that,*

$$\|\widehat{U}(V \odot W) - A\|_F^2 \leq (1+\epsilon) \min_{U \in \mathbb{R}^{n \times k}} \|U(V \odot W) - A\|_F^2,$$

holds with probability at least 0.999*, where* $\widehat{U}$ *denotes the optimal solution to* $\min_U \|U(V \odot W)D - AD\|_F^2$*.*

Proof. This follows by combining Theorem 21.5.1, Corollary 21.4.30, and Lemma 21.4.31. □

Remark 21.4.1. Replacing Theorem 21.5.1 (Algorithm 21.15) by Lemma 21.4.32 (Algorithm 21.10), we can obtain a slightly different version of Lemma 21.4.38 with $n\,\mathrm{poly}(\log n, k, 1/\epsilon)$ running time, where the dependence on k is better.

Algorithm 21.11 Frobenius Norm CURT Decomposition Algorithm, Input Sparsity Time and Nearly Optimal Number of Samples

procedure FCURTInputSparsity($A, U_B, V_B, W_B, n, k, \epsilon$) ▷ Theorem 21.4.39
  $d_1 \leftarrow d_2 \leftarrow d_3 \leftarrow O(k \log k + k/\epsilon)$.
  $\epsilon_0 \leftarrow 0.01$.
  Form $B_1 = V_B^\top \odot W_B^\top \in \mathbb{R}^{k \times n^2}$.
  $D_1 \leftarrow$FastTensorLeverageScoreGeneralOrder($V_B^\top, W_B^\top, n, n, k, \epsilon_0, d_1$). ▷ Algorithm 21.15
  Form $\widehat{U} = A_1 D_1 (B_1 D_1)^\dagger \in \mathbb{R}^{n \times k}$.
  Form $B_2 = \widehat{U}^\top \odot W_B^\top \in \mathbb{R}^{k \times n^2}$.
  $D_2 \leftarrow$FastTensorLeverageScoreGeneralOrder($\widehat{U}^\top, W_B^\top, n, n, k, \epsilon_0, d_2$).
  Form $\widehat{V} = A_2 D_2 (B_2 D_2)^\dagger \in \mathbb{R}^{n \times k}$.
  Form $B_3 = \widehat{U}^\top \odot \widehat{V}^\top \in \mathbb{R}^{k \times n^2}$.
  $D_3 \leftarrow$FastTensorLeverageScoreGeneralOrder($\widehat{U}^\top, \widehat{V}^\top, n, n, k, \epsilon_0, d_3$).
  $C \leftarrow A_1 D_1$, $R \leftarrow A_2 D_2$, $T \leftarrow A_3 D_3$.
  $U \leftarrow \sum_{i=1}^{k} ((B_1 D_1)^\dagger)_i \otimes ((B_2 D_2)^\dagger)_i \otimes ((B_3 D_3)^\dagger)_i$.
  return C, R, T and U.
end procedure

21.4.7.5 Input sparsity time algorithm

Theorem 21.4.39. *Given a 3rd order tensor $A \in \mathbb{R}^{n \times n \times n}$, let $k \geq 1$, and let $U_B, V_B, W_B \in \mathbb{R}^{n \times k}$ denote a rank-k, α-approximation to A. Then there exists an algorithm which takes $O(\text{nnz}(A) + n \,\text{poly}(k, 1/\epsilon))$ time and outputs three matrices $C \in \mathbb{R}^{n \times c}$ with columns from A, $R \in \mathbb{R}^{n \times r}$ with rows from A, $T \in \mathbb{R}^{n \times t}$ with tubes from A, and a tensor $U \in \mathbb{R}^{c \times r \times t}$ with $\text{rank}(U) = k$ such that $c = r = t = O(k \log k + k/\epsilon)$, and*

$$\left\| \sum_{i=1}^{c} \sum_{j=1}^{r} \sum_{l=1}^{t} U_{i,j,l} \cdot C_i \otimes R_j \otimes T_l - A \right\|_F^2 \leq (1+\epsilon)\alpha \min_{\text{rank}-k \ A'} \|A' - A\|_F^2$$

holds with probability $9/10$.

Proof. We define

$$\mathrm{OPT} := \min_{\mathrm{rank}-k\ A'} \|A' - A\|_F^2.$$

We already have three matrices $U_B \in \mathbb{R}^{n\times k}$, $V_B \in \mathbb{R}^{n\times k}$ and $W_B \in \mathbb{R}^{n\times k}$ and these three matrices provide a rank-k, α-approximation to A, i.e.,

$$\left\| \sum_{i=1}^{k} (U_B)_i \otimes (V_B)_i \otimes (W_B)_i - A \right\|_F^2 \leq \alpha\, \mathrm{OPT}\,. \tag{21.19}$$

Let $B_1 = V_B^\top \odot W_B^\top \in \mathbb{R}^{k\times n^2}$ denote the matrix where the i-th row is the vectorization of $(V_B)_i \otimes (W_B)_i$. Let $D_1 \in \mathbb{R}^{n^2\times n^2}$ be a sampling and rescaling matrix corresponding to sampling by the leverage scores of $B_1^\top$; there are d_1 nonzero entries on the diagonal of D_1. Let $A_i \in \mathbb{R}^{n\times n^2}$ denote the matrix obtained by flattening A along the i-th direction, for each $i \in [3]$.

Define $U^* \in \mathbb{R}^{n\times k}$ to be the optimal solution to $\min_{U\in\mathbb{R}^{n\times k}} \|UB_1 - A_1\|_F^2$, $\widehat{U} = A_1 D_1 (B_1 D_1)^\dagger \in \mathbb{R}^{n\times k}$, and $V_0 \in \mathbb{R}^{n\times k}$ to be the optimal solution to $\min_{V\in\mathbb{R}^{n\times k}} \|V \cdot (\widehat{U}^\top \odot W_B^\top) - A_2\|_F^2$. Due to Lemma 21.4.38, if $d_1 = O(k\log k + k/\epsilon)$ then with constant probability, we have

$$\|\widehat{U}B_1 - A_1\|_F^2 \leq \alpha_{D_1} \|U^* B_1 - A_1\|_F^2. \tag{21.20}$$

Recall that $(\widehat{U}^\top \odot W_B^\top) \in \mathbb{R}^{k\times n^2}$ denotes the matrix where the i-th row is the vector-

ization of $\widehat{U}_i \otimes (W_B)_i$, $\forall i \in [k]$. Now, we can show,

$$\begin{aligned}
&\|V_0 \cdot (\widehat{U}^\top \odot W_B^\top) - A_2\|_F^2 \\
\leq\ & \|\widehat{U} B_1 - A_1\|_F^2 && \text{by } V_0 = \arg\min_{V \in \mathbb{R}^{n \times k}} \|V \cdot (\widehat{U}^\top \odot W_B^\top) - A_2\|_F^2 \\
\leq\ & \alpha_{D_1} \|U^* B_1 - A_1\|_F^2 && \text{by Equation (21.20)} \\
\leq\ & \alpha_{D_1} \|U_B B_1 - A_1\|_F^2 && \text{by } U^* = \arg\min_{U \in \mathbb{R}^{n \times k}} \|U B_1 - A_1\|_F^2 \\
\leq\ & \alpha_{D_1} \alpha \, \mathrm{OPT} \,. && \text{by Equation (21.19)}
\end{aligned} \tag{21.21}$$

We define $B_2 = \widehat{U}^\top \odot W_B^\top$. Let $D_2 \in \mathbb{R}^{n^2 \times n^2}$ be a sampling and rescaling matrix corresponding to the leverage scores of $B_2^\top$. Suppose there are d_2 nonzero entries on the diagonal of D_2.

Define $V^* \in \mathbb{R}^{n \times k}$ to be the optimal solution to $\min_{V \in \mathbb{R}^{n \times k}} \|V B_2 - A_2\|_F^2$, $\widehat{V} = A_2 D_2 (B_2 D_2)^\dagger \in \mathbb{R}^{n \times k}$, $W_0 \in \mathbb{R}^{n \times k}$ to be the optimal solution to $\min_{W \in \mathbb{R}^{n \times k}} \|W \cdot (\widehat{U}^\top \odot \widehat{V}^\top) - A_3\|_F^2$, and V' to be the optimal solution to $\min_{V \in \mathbb{R}^{n \times k}} \|V B_2 D_2 - A_2 D_2\|_F^2$.

Due to Lemma 21.4.38, with constant probability, we have

$$\|\widehat{V} B_2 - A_2\|_F^2 \leq \alpha_{D_2} \|V^* B_2 - A_2\|_F^2. \tag{21.22}$$

Recall that $(\widehat{U}^\top \odot \widehat{V}^\top) \in \mathbb{R}^{k \times n^2}$ denotes the matrix where the i-th row is the vector-

ization of $\widehat{U}_i \otimes \widehat{V}_i$, $\forall i \in [k]$. Now, we can show,

$$\begin{aligned}
&\|W_0 \cdot (\widehat{U}^\top \odot \widehat{V}^\top) - A_3\|_F^2 \\
\leq\ & \|\widehat{V} B_2 - A_2\|_F^2 && \text{by } W_0 = \arg\min_{W \in \mathbb{R}^{n \times k}} \|W \cdot (\widehat{U}^\top \odot \widehat{V}^\top) - A_3\|_F^2 \\
\leq\ & \alpha_{D_2} \|V^* B_2 - A_2\|_F^2 && \text{by Equation (21.22)} \\
\leq\ & \alpha_{D_2} \|V_0 B_2 - A_2\|_F^2 && \text{by } V^* = \arg\min_{V \in \mathbb{R}^{n \times k}} \|V B_2 - A_2\|_F^2 \\
\leq\ & \alpha_{D_2} \alpha_{D_1} \alpha \, \mathrm{OPT} \,. && \text{by Equation (21.21)}
\end{aligned} \tag{21.23}$$

We define $B_3 = \widehat{U}^\top \odot \widehat{V}^\top$. Let $D_3 \in \mathbb{R}^{n^2 \times n^2}$ denote a sampling and rescaling matrix corresponding to sampling by the leverage scores of $B_3^\top$. Suppose there are d_3 nonzero entries on the diagonal of D_3.

Define $W^* \in \mathbb{R}^{n \times k}$ to be the optimal solution to $\min_{W \in \mathbb{R}^{n \times k}} \|W B_3 - A_3\|_F^2$, $\widehat{W} = A_3 D_3 (B_3 D_3)^\dagger \in \mathbb{R}^{n \times k}$, and W' to be the optimal solution to $\min_{W \in \mathbb{R}^{n \times k}} \|W B_3 D_3 - A_3 D_3\|_F^2$.

Due to Lemma 21.4.38 with constant probability, we have

$$\|\widehat{W} B_3 - A_3\|_F^2 \leq \alpha_{D_3} \|W^* B_3 - A_3\|_F^2. \tag{21.24}$$

Now we can show,

$$\begin{aligned}
\|\widehat{W} B_3 - A_3\|_F^2 &\leq \alpha_{D_3} \|W^* B_3 - A_3\|_F^2, && \text{by Equation (21.24)} \\
&\leq \alpha_{D_3} \|W_0 B_3 - A_3\|_F^2, && \text{by } W^* = \arg\min_{W \in \mathbb{R}^{n \times k}} \|W B_3 - A_3\|_F^2 \\
&\leq \alpha_{D_3} \alpha_{D_2} \alpha_{D_1} \alpha \, \mathrm{OPT} \,. && \text{by Equation (21.23)}
\end{aligned}$$

This implies,

$$\left\| \sum_{i=1}^k \widehat{U}_i \otimes \widehat{V}_i \otimes \widehat{W}_i - A \right\|_F^2 \leq O(1) \alpha \, \mathrm{OPT}^2 \,.$$

where $\widehat{U} = A_1 D_1 (B_1 D_1)^\dagger$, $\widehat{V} = A_2 D_2 (B_2 D_2)^\dagger$, $\widehat{W} = A_3 D_3 (B_3 D_3)^\dagger$.

By Lemma 21.4.38, we need to set $d_1 = d_2 = d_3 = O(k \log k + k/\epsilon)$. Note that $B_1 = (V_B^\top \odot W_B^\top)$. Thus D_1 can be found in $n \cdot \mathrm{poly}(k, 1/\epsilon)$ time. Because D_1 has a small number of nonzero entries on the diagonal, we can compute $B_1 D_1$ quickly without explicitly writing down B_1. Also $A_1 D_1$ can be computed in $\mathrm{nnz}(A)$ time. Using $(A_1 D_1)$ and $(B_1 D_1)$, we can compute $\widehat{U}$ in $n\,\mathrm{poly}(k, 1/\epsilon)$ time. In a similar way, we can compute B_2, D_2, B_3, and D_3. Since tensor U is constructed based on three $\mathrm{poly}(k, 1/\epsilon)$ size matrices, $(B_1 D_1)^\dagger$, $(B_2 D_2)^\dagger$, and $(B_3 D_3)^\dagger$, the overall running time is $O(\mathrm{nnz}(A) + n\,\mathrm{poly}(k, 1/\epsilon))$ □

21.4.7.6 Optimal sample complexity algorithm

Theorem 21.4.40. *Given a 3rd order tensor $A \in \mathbb{R}^{n \times n \times n}$, let $k \geq 1$, and let $U_B, V_B, W_B \in \mathbb{R}^{n \times k}$ denote a rank-k, α-approximation to A. Then there exists an algorithm which takes $O(\mathrm{nnz}(A) \log n + n^2\,\mathrm{poly}(\log n, k, 1/\epsilon))$ time and outputs three matrices: $C \in \mathbb{R}^{n \times c}$ with columns from A, $R \in \mathbb{R}^{n \times r}$ with rows from A, $T \in \mathbb{R}^{n \times t}$ with tubes from A, and a tensor $U \in \mathbb{R}^{c \times r \times t}$ with $\mathrm{rank}(U) = k$ such that $c = r = t = O(k/\epsilon)$, and*

$$\left\| \sum_{i=1}^{c} \sum_{j=1}^{r} \sum_{l=1}^{t} U_{i,j,l} \cdot C_i \otimes R_j \otimes T_l - A \right\|_F^2 \leq (1+\epsilon)\alpha \min_{\mathrm{rank}-k\ A'} \|A' - A\|_F^2$$

holds with probability 9/10.

Proof. The proof is almost the same as the proof of Theorem 21.4.39. The only difference is that instead of using Theorem 21.4.38, we use Theorem 21.4.14. □

Algorithm 21.12 Frobenius Norm CURT Decomposition Algorithm, Optimal Sample Complexity

procedure FCURTOptimalSamples(A, U_B, V_B, W_B, n, k) ▷ Theorem 21.4.40
  $d_1 \leftarrow d_2 \leftarrow d_3 \leftarrow O(k/\epsilon)$.
  Form $B_1 = V_B^\top \odot W_B^\top \in \mathbb{R}^{k \times n^2}$.
  $D_1 \leftarrow$GeneralizedMatrixRowSubsetSelection$(A_1^\top, B_1^\top, n^2, n, k, \epsilon)$. ▷ Algorithm 21.7
  Let d_1 denote the number of nonzero entries in D_1. ▷ $d_1 = O(k/\epsilon)$
  Form $\widehat{U} = A_1 D_1 (B_1 D_1)^\dagger \in \mathbb{R}^{n \times k}$.
  Form $B_2 = \widehat{U}^\top \odot W_B^\top \in \mathbb{R}^{k \times n^2}$.
  $D_2 \leftarrow$GeneralizedMatrixRowSubsetSelection$(A_2^\top, B_2^\top, n^2, n, k, \epsilon)$. ▷ Algorithm 21.7
  Let d_2 denote the number of nonzero entries in D_2. ▷ $d_2 = O(k/\epsilon)$
  Form $\widehat{V} = A_2 D_2 (B_2 D_2)^\dagger \in \mathbb{R}^{n \times k}$.
  Form $B_3 = \widehat{U}^\top \odot \widehat{V}^\top \in \mathbb{R}^{k \times n^2}$.
  $D_3 \leftarrow$GeneralizedMatrixRowSubsetSelection$(A_3^\top, B_3^\top, n^2, n, k, \epsilon)$. ▷ Algorithm 21.7
  d_3 denote the number of nonzero entries in D_3. ▷ $d_3 = O(k/\epsilon)$
  $C \leftarrow A_1 D_1$, $R \leftarrow A_2 D_2$, $T \leftarrow A_3 D_3$.
  $U \leftarrow \sum_{i=1}^{k} ((B_1 D_1)^\dagger)_i \otimes ((B_2 D_2)^\dagger)_i \otimes ((B_3 D_3)^\dagger)_i$.
  return C, R, T and U.
end procedure

21.4.8 Face-based selection and decomposition

Previously we provided column-based tensor CURT algorithms, which are algorithms that can select a subset of columns from each of the three dimensions. Here we provide two face-based tensor CURT decomposition algorithms. The first algorithm runs in polynomial time and is a bicriteria algorithm (the number of samples is poly(k/ϵ)). The second algorithm needs to start with a rank-k $(1 + O(\epsilon))$-approximate solution, which we then show how to combine with our previous algorithm. Both of our algorithms are able to select a subset of column-row faces, a subset of row-tube faces and a subset of column-tube faces. The second

algorithm is able to output U, but the first algorithm is not.

21.4.8.1 Column-row, column-tube, row-tube face subset selection

Algorithm 21.13 Frobenius Norm Tensor Column-row, Row-tube and Tube-column Face Subset Selection

procedure FFACECRTSELECTION(A, n, k, ϵ) ▷ Theorem 21.4.41
  $s_1 \leftarrow s_2 \leftarrow O(k/\epsilon)$.
  Choose a Gaussian matrix S_1 with s_1 columns. ▷ Definition 21.3.12
  Choose a Gaussian matrix S_2 with s_2 columns. ▷ Definition 21.3.12
  Form matrix V_3 by setting the (i,j)-th column to be $(A_2S_2)_j$.
  $D_3 \leftarrow$GENERALIZEDMATRIXROWSUBSETSELECTION(A_2,V_3,n,n^2,s_1s_2,ϵ). ▷ Algorithm 21.7
  Let d_3 denote the number of nonzero entries in D_3. ▷ $d_3 = O(s_1s_2/\epsilon)$
  Form matrix U_2 by setting the (i,j)-th column to be $(A_1S_1)_i$.
  $D_2 \leftarrow$GENERALIZEDMATRIXROWSUBSETSELECTION(A_1,U_2,n,n^2,s_1s_2,ϵ).
  Let d_2 denote the number of nonzero entries in D_2. ▷ $d_2 = O(s_1s_2/\epsilon)$
  Form matrix W_1 by setting the (i,j)-th column to be $(A(I, D_3, I)_3)_j$.
  $D_1 \leftarrow$GENERALIZEDMATRIXROWSUBSETSELECTION(A_3,W_1,n,n^2,s_1s_2,ϵ).
  Let d_1 denote the number of nonzero entries in D_1. ▷ $d_1 = O(s_1s_2/\epsilon)$
  $T \leftarrow A(I, I, D_1)$, $C \leftarrow A(D_2, I, I)$, and $R \leftarrow A(I, D_3, I)$.
  return C, R and T.
end procedure

Theorem 21.4.41. *Given a 3rd order tensor $A \in \mathbb{R}^{n\times n\times n}$, for any $k \geq 1$, there exists an algorithm which takes $O(\text{nnz}(A)) \log n + n^2 \,\text{poly}(\log n, k, 1/\epsilon)$ time and outputs three tensors : a subset $C \in \mathbb{R}^{c\times n\times n}$ of row-tube faces of A, a subset $R \in \mathbb{R}^{n\times r\times n}$ of column-tube faces of A, and a subset $T \in \mathbb{R}^{n\times n\times t}$ of column-row faces of A, where $c = r = t = \text{poly}(k, 1/\epsilon)$, and for which there exists a tensor $U \in \mathbb{R}^{tn\times cn\times rn}$ for which*

$$\|U(T_1, C_2, R_3) - A\|_F^2 \leq (1+\epsilon) \min_{\text{rank}-k\ A'} \|A' - A\|_F^2,$$

or equivalently,

$$\left\| \sum_{i=1}^{tn} \sum_{j=1}^{cn} \sum_{l=1}^{rn} U_{i,j,l} \cdot (T_1)_i \otimes (C_2)_j \otimes (R_3)_l - A \right\|_F^2 \leq (1+\epsilon) \min_{\text{rank}-k\ A'} \|A' - A\|_F^2.$$

Proof. We fix $V^* \in \mathbb{R}^{n \times k}$ and $W^* \in \mathbb{R}^{n \times k}$. We define $Z_1 \in \mathbb{R}^{k \times n^2}$ where the i-th row of Z_1 is the vector $V_i \otimes W_i$. Choose a sketching (Gaussian) matrix $S_1 \in \mathbb{R}^{n^2 \times s_1}$ (Definition 21.3.12), and let $\widehat{U} = A_1 S_1 (Z_1 S_1)^\dagger \in \mathbb{R}^{n \times k}$. Following a similar argument as in the previous theorem, we have

$$\|\widehat{U} Z_1 - A_1\|_F^2 \leq (1+\epsilon)\,\mathrm{OPT}\,.$$

We fix $\widehat{U}$ and W^*. We define $Z_2 \in \mathbb{R}^{k \times n^2}$ where the i-th row of Z_2 is the vector $\widehat{U}_i \otimes W_i^*$. Choose a sketching (Gaussian) matrix $S_2 \in \mathbb{R}^{n^2 \times s_2}$ (Definition 21.3.12), and let $\widehat{V} = A_2 S_2 (Z_2 S_2)^\dagger \in \mathbb{R}^{n \times k}$. Following a similar argument as in the previous theorem, we have

$$\|\widehat{V} Z_2 - A_2\|_F^2 \leq (1+\epsilon)^2\,\mathrm{OPT}\,.$$

We fix $\widehat{U}$ and $\widehat{V}$. Note that $\widehat{U} = A_1 S_1 (Z_1 S_1)^\dagger$ and $\widehat{V} = A_2 S_2 (Z_2 S_2)^\dagger$. We define $Z_3 \in \mathbb{R}^{k \times n^2}$ such that the i-th row of Z_3 is the vector $\widehat{U}_i \otimes \widehat{V}_i$. Let $z_3 = s_1 \cdot s_2$. We define $Z_3' \in \mathbb{R}^{z_3 \times n^2}$ such that, $\forall i \in [s_1], \forall j \in [s_2]$, the $i + (j-1)s_1$-th row of Z_3' is the vector $(A_1 S_1)_i \otimes (A_2 S_2)_j$.

We define $U_3 \in \mathbb{R}^{n \times z_3}$ to be the matrix where the $i + (j-1)s_1$-th column is $(A_1 S_1)_i$ and $V_3 \in \mathbb{R}^{n \times z_3}$ to be the matrix where the $i + (j-1)s_1$-th column is $(A_2 S_2)_j$. Then $Z_3' = (U_3^\top \odot V_3^\top)$.

We first have,

$$\min_{W \in \mathbb{R}^{n \times k}, X \in \mathbb{R}^{k \times z_3}} \|W X Z_3' - A_3\|_F^2 \leq \min_{W \in \mathbb{R}^{n \times k}} \|W Z_3 - A_3\|_F^2 \leq (1+\epsilon)^2\,\mathrm{OPT}\,.$$

Now consider the following objective function,

$$\min_{W \in \mathbb{R}^{n \times z_3}} \|V_3 \cdot (W^\top \odot U_3^\top) - A_2\|_F^2.$$

Let D_3 denote a sampling and rescaling diagonal matrix according to $V_1 \in \mathbb{R}^{n \times z_3}$, let d_3 denote the number of nonzero entries of D_3. Then we have

$$\begin{aligned} & \min_{W \in \mathbb{R}^{n \times z_3}} \|D_3 V_3 \cdot (W^\top \odot U_3^\top) - D_3 A_2\|_F^2 \\ = & \min_{W \in \mathbb{R}^{n \times z_3}} \|U_3 \otimes (D_3 V_3) \otimes W - A(I, D_3, I)\|_F^2 \\ = & \min_{W \in \mathbb{R}^{n \times z_3}} \|W \cdot (U_3^\top \odot (D_3 V_3)^\top) - (A(I, D_3, I))_3\|_F^2, \end{aligned}$$

where the first equality follows by retensorizing the objective function, and the second equality follows by flattening the tensor along the third dimension.

Let $\overline{Z}_3$ denote $(U_3^\top \odot (D_3 V_3)^\top) \in \mathbb{R}^{z_3 \times n d_3}$ and $W' = (A(I, D_3, I))_3 \in \mathbb{R}^{n \times n d_3}$. Using Theorem 21.4.14, we can find a diagonal matrix $D_3 \in \mathbb{R}^{n^2 \times n^2}$ with $d_3 = O(z_3/\epsilon) = O(k^2/\epsilon^3)$ nonzero entries such that

$$\|U_3 \otimes V_3 \otimes (W' Z_3^\dagger) - A\|_F^2 \leq (1+\epsilon)^3 \, \mathrm{OPT} \, .$$

We define $U_2 = U_3 \in \mathbb{R}^{n \times z_2}$ with $z_2 = z_3$. We define $W_2 = W' \overline{Z}_3^\dagger \in \mathbb{R}^{n \times z_2}$ with $z_2 = z_3$. We consider,

$$\min_{V \in \mathbb{R}^{n \times z_2}} \|U_2 \cdot (V^\top \odot W_2^\top) - A_1\|_F^2.$$

Let D_2 denote a sampling and rescaling matrix according to U_2, and let d_2 denote the number

of nonzero entries of D_2. Then, we have

$$\begin{aligned}
&\min_{V \in \mathbb{R}^{n \times z_2}} \|D_2 U_2 \cdot (V^\top \odot W_2^\top) - D_2 A_1\|_F^2 \\
= &\min_{V \in \mathbb{R}^{n \times z_2}} \|D_2 U_2 \otimes V \otimes W_2 - A(D_2, I, I)\|_F^2 \\
= &\min_{V \in \mathbb{R}^{n \times z_2}} \|V \cdot (W_2^\top \odot (D_2 U_2)^\top) - (A(D_2, I, I))_2\|_F^2,
\end{aligned}$$

where the first equality follows by retensorizing the objective function, and the second equality follows by flattening the tensor along the second dimension.

Let $\overline{Z}_2$ denote $(W_2^\top \odot (D_2 U_2)^\top) \in \mathbb{R}^{z_2 \times n d_2}$ and $V' = (A(D_2, I, I))_2 \in \mathbb{R}^{n \times n d_2}$. Using Theorem 21.4.14, we can find a diagonal matrix $D_2 \in \mathbb{R}^{n^2 \times n^2}$ with $d_2 = O(z_2/\epsilon)$ nonzero entries such that

$$\|U_2 \otimes (V' \overline{Z}_2^\dagger) \otimes W_2 - A\|_F^2 \leq (1+\epsilon)^4 \operatorname{OPT}.$$

We define $W_1 = W_2 \in \mathbb{R}^{n \times z_1}$ with $z_1 = z_2$, and define $V_1 = (V' \overline{Z}_2^\dagger) \in \mathbb{R}^{n \times z_1}$ with $z_1 = z_2$.

Let D_1 denote a sampling and rescaling matrix according to W_1, and let d_1 denote the number of nonzero entries of D_1. Then we have

$$\begin{aligned}
&\min_{U \in \mathbb{R}^{n \times z_1}} \|D_1 W_1 \cdot (U^\top \odot V_1^\top) - D_1 A_3\|_F^2 \\
= &\min_{U \in \mathbb{R}^{n \times z_1}} \|U \otimes V_1 \otimes (D_1 W_1) - A(I, I, D_1)\|_F^2 \\
= &\min_{U \in \mathbb{R}^{n \times z_1}} \|U \cdot (V_1^\top \odot (D_1 W_1)^\top) - A(I, I, D_1)_1\|_F^2
\end{aligned}$$

where the first equality follows by unflattening the objective function, and second equality follows by flattening the tensor along the first dimension.

Let $\overline{Z}_1$ denote $(V_1^\top \odot (D_1 W_1)^\top) \in \mathbb{R}^{z_1 \times nd_1}$, and $U' = A(I, I, D_1)_1 \in \mathbb{R}^{n \times nd_1}$. Using Theorem 21.4.14, we can find a diagonal matrix $D_1 \in \mathbb{R}^{n^2 \times n^2}$ with $d_1 = O(z_1/\epsilon)$ nonzero entries such that

$$\|(U'\overline{Z}_1^\dagger) \otimes (V_1) \otimes W_1 - A\|_F^2 \leq (1+\epsilon)^5 \,\mathrm{OPT},$$

which means,

$$\|(U'\overline{Z}_1^\dagger) \otimes (V'\overline{Z}_2^\dagger) \otimes (W'\overline{Z}_3^\dagger) - A\|_F^2 \leq (1+\epsilon)^5 \,\mathrm{OPT}\,.$$

Putting U', V', W' together completes the proof. □

Corollary 21.4.42. *Given a 3rd order tensor $A \in \mathbb{R}^{n \times n \times n}$, for any $k \geq 1$, there exists an algorithm which takes $O(\mathrm{nnz}(A)) + n^2\,\mathrm{poly}(k, 1/\epsilon)$ time and outputs three tensors : a subset $C \in \mathbb{R}^{c \times n \times n}$ of row-tube faces of A, a subset $R \in \mathbb{R}^{n \times r \times n}$ of column-tube faces of A, and a subset $T \in \mathbb{R}^{n \times n \times t}$ of column-row faces of A, where $c = r = t = \mathrm{poly}(k, 1/\epsilon)$, so that there exists a tensor $U \in \mathbb{R}^{tn \times cn \times rn}$ for which*

$$\|U(T_1, C_2, R_3) - A\|_F^2 \leq (1+\epsilon) \min_{\mathrm{rank}-k \;\; A'} \|A' - A\|_F^2,$$

or equivalently,

$$\left\| \sum_{i=1}^{tn} \sum_{j=1}^{cn} \sum_{l=1}^{rn} U_{i,j,l} \cdot (T_1)_i \otimes (C_2)_j \otimes (R_3)_l - A \right\|_F^2 \leq (1+\epsilon) \min_{\mathrm{rank}-k \;\; A'} \|A' - A\|_F^2$$

Proof. If we allow a $\mathrm{poly}(k/\epsilon)$ factor increase in running time and a $\mathrm{poly}(k/\epsilon)$ factor increase in the number of faces selected, then instead of using generalized row subset selection, which has running time depending on $\log n$, we can use the technique in Section 21.6 to avoid the $\log n$ factor. □

Algorithm 21.14 Frobenius Norm (Face-based) CURT Decomposition Algorithm, Optimal Sample Complexity

procedure FFACECURTDECOMPOSITION(A, U_B, V_B, W_B, n, k) ▷ Theorem 21.4.43
  $D_1 \leftarrow$GENERALIZEDMATRIXROWSUBSETSELECTION($A_3, W_B, n, n^2, k, \epsilon$). ▷ Algorithm 21.7, the number of nonzero entries is $d_1 = O(k/\epsilon)$
  Form $Z_1 = V_B^\top \odot (D_1 W_B)^\top$.
  Form $\widehat{U} = (A(I, I, D_1))_1 Z_1^\dagger \in \mathbb{R}^{n \times k}$.
  $D_2 \leftarrow$GENERALIZEDMATRIXROWSUBSETSELECTION($A_1, \widehat{U}, n, n^2, k, \epsilon$). ▷ The number of nonzero entries is $d_2 = O(k/\epsilon)$
  Form $Z_2 = (W_B^\top \odot (D_2 \widehat{U}))$.
  Form $\widehat{V} = (A(D_2, I, I))_2 Z_2^\dagger \in \mathbb{R}^{n \times k}$.
  $D_3 \leftarrow$GENERALIZEDMATRIXROWSUBSETSELECTION($A_2, \widehat{V}, n, n^2, k, \epsilon$). ▷ The number of nonzero entries is $d_3 = O(k/\epsilon)$
  Form $Z_3 = \widehat{U}^\top \odot (D_3 \widehat{V})^\top$.
  Form $\widehat{W} = (A(I, D_3, I))_3 (Z_3)^\dagger \in \mathbb{R}^{n \times k}$.
  $T \leftarrow A(I, I, D_1)$, $C \leftarrow A(D_2, I, I)$, $R \leftarrow A(I, D_3, I)$.
  $U \leftarrow \sum_{i=1}^k ((Z_1)^\dagger)_i \otimes ((Z_2)^\dagger)_i \otimes ((Z_3)^\dagger)_i$.
  return C, R, T and U.
end procedure

21.4.8.2 CURT decomposition

Theorem 21.4.43. *Given a 3rd order tensor $A \in \mathbb{R}^{n \times n \times n}$, let $k \geq 1$, and let $U_B, V_B, W_B \in \mathbb{R}^{n \times k}$ denote a rank-k, α-approximation to A. Then there exists an algorithm which takes $O(\mathrm{nnz}(A)) \log n + n^2 \, \mathrm{poly}(\log n, k, 1/\epsilon)$ time and outputs three tensors: $C \in \mathbb{R}^{c \times n \times n}$ with row-tube faces from A, $R \in \mathbb{R}^{n \times r \times n}$ with colum-tube faces from A, $T \in \mathbb{R}^{n \times n \times t}$ with column-row faces from A, and a (factorization of a) tensor $U \in \mathbb{R}^{tn \times cn \times rn}$ with $\mathrm{rank}(U) = k$ for which $c = r = t = O(k/\epsilon)$ and*

$$\|U(T_1, C_2, R_3) - A\|_F^2 \leq (1 + \epsilon)\alpha \min_{\mathrm{rank}-k \ A'} \|A' - A\|_F^2,$$

or equivalently,

$$\left\| \sum_{i=1}^{tn} \sum_{j=1}^{cn} \sum_{l=1}^{rn} U_{i,j,l} \cdot (T_1)_i \otimes (C_2)_j \otimes (R_3)_l - A \right\|_F^2 \leq (1+\epsilon)\alpha \min_{\text{rank}-k \ A'} \|A' - A\|_F^2$$

holds with probability $9/10$.

Proof. We already have three matrices $U_B \in \mathbb{R}^{n\times k}$, $V_B \in \mathbb{R}^{n\times k}$ and $W_B \in \mathbb{R}^{n\times k}$ and these three matrices provide a rank-k, α-approximation to A, i.e.,

$$\|U_B \otimes V_B \otimes W_B - A\|_F^2 \leq \alpha \underbrace{\min_{\text{rank}-k \ A'} \|A' - A\|_F^2}_{\text{OPT}}.$$

We can consider the following problem,

$$\min_{U \in \mathbb{R}^{n\times k}} \|W_B \cdot (U^\top \odot V_B^\top) - A_3\|_F^2.$$

Let D_1 denote a sampling and rescaling diagonal matrix according to W_B, and let d_1 denote the number of nonzero entries of D_1. Then we have

$$\begin{aligned}
&\min_{U \in \mathbb{R}^{n\times k}} \|(D_1 W_B) \cdot (U^\top \odot V_B^\top) - D_1 A_3\|_F^2 \\
=&\min_{U \in \mathbb{R}^{n\times k}} \|U \otimes V_B \otimes D_1 W_B - A(I, I, D_1)\|_F^2 \\
=&\min_{U \in \mathbb{R}^{n\times k}} \|U \cdot (V_B^\top \odot (D_1 W_B)^\top) - (A(I, I, D_1))_1\|_F^2,
\end{aligned}$$

where the first equality follows by retensorizing the objective function, and the second equality follows by flattening the tensor along the first dimension. Let Z_1 denote $V_B^\top \odot (D_1 W_B)^\top \in \mathbb{R}^{k\times nd_1}$, and define $\widehat{U} = (A(I, I, D_1))_1 Z_1^\dagger \in \mathbb{R}^{n\times k}$. Then we have

$$\|\widehat{U} \otimes V_B \otimes W_B - A\|_F^2 \leq (1+\epsilon)\alpha \,\text{OPT}\,.$$

In the second step, we fix $\widehat{U}$ and W_B, and consider the following objective function,

$$\min_{V \in \mathbb{R}^{n \times k}} \|\widehat{U} \cdot (V^\top \odot W_B) - A_1\|_F^2.$$

Let D_2 denote a sampling and rescaling matrix according to $\widehat{U}$, and let d_2 denote the number of nonzero entries of D_2. Then we have,

$$\begin{aligned} & \min_{V \in \mathbb{R}^{n \times k}} \|(D_2 \widehat{U}) \cdot (V^\top \odot W_B^\top) - D_2 A_1\|_F^2 \\ = & \min_{V \in \mathbb{R}^{n \times k}} \|(D_2 \widehat{U}) \otimes V \otimes W_B - A(D_2, I, I)\|_F^2 \\ = & \min_{V \in \mathbb{R}^{n \times k}} \|V \cdot (W_B^\top \odot (D_2 \widehat{U})^\top) - (A(D_2, I, I))_2\|_F^2, \end{aligned}$$

where the first equality follows by unflattening the objective function, and the second equality follows by flattening the tensor along the second dimension. Let Z_2 denote $(W_B^\top \odot (D_2 \widehat{U})^\top) \in \mathbb{R}^{k \times n d_2}$, and define $\widehat{V} = (A(D_2, I, I))_2 (Z_2)^\dagger \in \mathbb{R}^{n \times k}$. Then we have,

$$\|\widehat{U} \otimes \widehat{V} \otimes W_B - A\|_F^2 \leq (1+\epsilon)^2 \alpha \, \mathrm{OPT} \, .$$

In the third step, we fix $\widehat{U}$ and $\widehat{V}$, and consider the following objective function,

$$\min_{W \in \mathbb{R}^{n \times k}} \|\widehat{V} \cdot (W \odot \widehat{U}) - A_2\|_F^2.$$

Let D_3 denote a sampling and rescaling matrix according to $\widehat{V}$, and let d_3 denote the number of nonzero entries of D_3. Then we have,

$$\begin{aligned} & \min_{W \in \mathbb{R}^{n \times k}} \|(D_3 \widehat{V}) \cdot (W^\top \odot \widehat{U}^\top) - D_3 A_2\|_F^2 \\ = & \min_{W \in \mathbb{R}^{n \times k}} \|\widehat{U} \otimes (D_3 \widehat{V}) \otimes W - A(I, D_3, I)\|_F^2 \\ = & \min_{W \in \mathbb{R}^{n \times k}} \|W \cdot (\widehat{U}^\top \odot (D_3 \widehat{V})^\top) - (A(I, D_3, I))_3\|_F^2, \end{aligned}$$

where the first equality follows by retensorizing the objective function, and the second equality follows by flattening the tensor along the third dimension. Let Z_3 denote $(\widehat{U}^\top \odot (D_3 \widehat{V})^\top) \in \mathbb{R}^{k \times nd_3}$, and define $\widehat{W} = (A(I, D_3, I))_3 (Z_3)^\dagger$. Putting it all together, we have,

$$\|\widehat{U} \otimes \widehat{V} \otimes \widehat{W} - A\|_F^2 \leq (1+\epsilon)^3 \alpha \, \mathrm{OPT} \, .$$

This implies

$$\|(A(I, I, D_1))_1 Z_1^\dagger \otimes (A(D_2, I, I))_2 Z_2^\dagger \otimes (A(I, D_3, I))_3 Z_3^\dagger - A\|_F^2 \leq (1+\epsilon)^3 \alpha \, \mathrm{OPT} \, .$$

□

21.4.9 Solving small problems

Theorem 21.4.44. *Let* $\max_i\{t_i, d_i\} \leq n$. *Given a* $t_1 \times t_2 \times t_3$ *tensor* A *and three matrices: a* $t_1 \times d_1$ *matrix* T_1, *a* $t_2 \times d_2$ *matrix* T_2, *and a* $t_3 \times d_3$ *matrix* T_3, *if for any* $\delta > 0$ *there exists a solution to*

$$\min_{X_1,X_2,X_3} \left\| \sum_{i=1}^{k} (T_1X_1)_i \otimes (T_2X_2)_i \otimes (T_3X_3)_i - A \right\|_F^2 := \mathrm{OPT},$$

and each entry of X_i *can be expressed using* $O(n^\delta)$ *bits, then there exists an algorithm that takes* $n^{O(\delta)} \cdot 2^{O(d_1k+d_2k+d_3k)}$ *time and outputs three matrices:* $\widehat{X}_1$, $\widehat{X}_2$, *and* $\widehat{X}_3$ *such that* $\|(T_1\widehat{X}_1) \otimes (T_2\widehat{X}_2) \otimes (T_3\widehat{X}_3) - A\|_F^2 = \mathrm{OPT}$.

Proof. For each $i \in [3]$, we can create $t_i \times d_i$ variables to represent matrix X_i. Let x denote this list of variables. Let B denote tensor $\sum_{i=1}^{k} (T_1X_1)_i \otimes (T_2X_2)_i \otimes (T_3X_3)_i$ and let $B_{i,j,l}(x)$ denote an entry of tensor B (which can be thought of as a polynomial written in terms of x). Then we can write the following objective function,

$$\min_x \sum_{i=1}^{t_1} \sum_{j=1}^{t_2} \sum_{l=1}^{t_3} (B_{i,j,l}(x) - A_{i,j,l})^2.$$

We slightly modify the above objective function to obtain a new objective function,

$$\begin{aligned} \min_{x,\sigma} \ & \sum_{i=1}^{t_1} \sum_{j=1}^{t_2} \sum_{l=1}^{t_3} (B_{i,j,l}(x) - A_{i,j,l})^2, \\ \text{s.t. } & \|x\|_2^2 \leq 2^{O(n^\delta)}, \end{aligned}$$

where the last constraint is unharmful, because there exists a solution that can be written using $O(n^\delta)$ bits. Note that the number of inequality constraints in the above system is

$O(1)$, the degree is $O(1)$, and the number of variables is $v = (d_1k + d_2k + d_3k)$. Thus by Theorem 21.3.2, the minimum nonzero cost is at least

$$(2^{O(n^\delta)})^{-2^{O(v)}}.$$

It is clear that the upper bound on the cost is at most $2^{O(n^\delta)}$. Thus the number of binary search steps is at most $\log(2^{O(n^\delta)})2^{O(v)}$. In each step of the binary search, we need to choose a cost C between the lower bound and the upper bound, and write down the polynomial system,

$$\sum_{i=1}^{t_1}\sum_{j=1}^{t_2}\sum_{l=1}^{t_3}(B_{i,j,l}(x) - A_{i,j,l})^2 \leq C,$$
$$\|x\|_2^2 \leq 2^{O(n^\delta)}.$$

Using Theorem 21.3.1, we can determine if there exists a solution to the above polynomial system. Since the number of variables is v, and the degree is $O(1)$, the number of inequality constraints is $O(1)$. Thus, the running time is

$$\text{poly(bitsize)} \cdot (\#\,\text{constraints} \cdot \text{degree})^{\#\,\text{variables}} = n^{O(\delta)}2^{O(v)}.$$

□

21.5 Extension to general q-th order tensors

This section provides the details for our extensions from 3rd order tensors to general q-th order tensors. In most practical applications, the order q is a constant. Thus, to simplify the analysis, we use $O_q(\cdot)$ to hide dependencies on q.

21.5.1 Fast sampling of columns according to leverage scores, implicitly

This section explains an algorithm that is able to sample from the leverage scores from the $\odot$ product of q matrices $U_1, U_2, \cdots, U_q$ without explicitly writing down $U_1 \odot U_2 \odot \cdots U_q$. To build this algorithm we combine TensorSketch, some ideas from [DMIMW12], and some techniques from [AKO11, MW10]. Finally, we improve the running time for sampling columns according to the leverage scores from $\mathrm{poly}(n)$ to $\widetilde{O}(n)$. Given q matrices $U_1, U_2, \cdots, U_q$, with each such matrix U_i having size $k \times n_i$, we define $A \in \mathbb{R}^{k \times \prod_{i=1}^q n_i}$ to be the matrix where the i-th row of A is the vectorization of $U_1^i \otimes U_2^i \otimes \cdots \otimes U_q^i$, $\forall i \in [k]$. Naïvely, in order to sample $\mathrm{poly}(k, 1/\epsilon)$ rows from A according to the leverage scores, we need to write down $\prod_{i=1}^q n_i$ leverage scores. This approach will take at least $\prod_{i=1}^q n_i$ running time. In the remainder of this section, we will explain how to do it in $O_q(n \cdot \mathrm{poly}(k, 1/\epsilon))$ time for any constant p, and $\max_{i \in [q]} n_i \leq n$.

Theorem 21.5.1. *Given q matrices $U_1 \in \mathbb{R}^{k \times n_1}$, $U_2 \in \mathbb{R}^{k \times n_2}$, $\cdots$, $U_q \in \mathbb{R}^{k \times n_q}$, let $\max_i n_i \leq n$. There exists an algorithm that takes $O_q(n \cdot \mathrm{poly}(k, 1/\epsilon) \cdot R_{\mathrm{samples}})$ time and samples R_{samples} columns of $U_1 \odot U_2 \odot \cdots \odot U_q \in \mathbb{R}^{k \times \prod_{i=1}^q n_i}$ according to the leverage scores of $U_1 \odot U_2 \odot \cdots \odot U_q$.*

Proof. Let $\max_i n_i \leq n$. First, choosing Π_0 to be a TensorSketch, we can compute R^{-1} in $O_q(n \, \mathrm{poly}(k, 1/\epsilon))$ time, where R is the R in a QR-factorization. We want to sample columns from $U_1 \odot U_2 \odot \cdots \odot U_q$ according to the square of the ℓ_2-norm of each column of $R^{-1}(U_1 \odot U_2 \odot \cdots U_q)$. The issue is the number of columns of this matrix is already $\prod_{i=1}^q n_i$. The goal is to sample columns from $R^{-1}(U_1 \odot U_2 \odot \cdots U_q)$ without explicitly computing the square of the ℓ_2-norm of each column.

Similarly as in the proof of Lemma 21.4.32, we have the observation that the following

Algorithm 21.15 Fast Tensor Leverage Score Sampling, for General q-th Order

procedure FastTensorLeverageScoreGeneralOrder($\{U_i\}_{i\in[q]}, \{n_i\}_{i\in[q]}, k, \epsilon, R_{\text{samples}}$) ▷ Theorem 21.5.1
  $s_1 \leftarrow \text{poly}(k, 1/\epsilon)$.
  Choose $\Pi_0, \Pi_1 \in \mathbb{R}^{n_1 n_2 \cdots n_q \times s_1}$ to each be a TensorSketch. ▷ Definition 21.3.20
  Compute $R^{-1} \in \mathbb{R}^{k\times k}$ by using $(U_1 \odot U_2 \odot \cdots \odot U_q)\Pi_0$. ▷ $U_i \in \mathbb{R}^{k\times n_i}, \forall i \in [q]$
  $V_0 \leftarrow R^{-1}$, $n_0 \leftarrow k$.
  for $i = 1 \to [n_0]$ **do**
    $\alpha_i \leftarrow \|(V_0)^i((U_1 \odot U_2 \odot \cdots \odot U_q)\Pi_1)\|_2^2$.
  end for
  for $r = 1 \to R_{\text{samples}}$ **do**
    Sample $\widehat{j_0}$ from $[n_0]$ with probability $\alpha_i / \sum_{i'=1}^{n_0} \alpha_{i'}$.
    for $l = 1 \to q-1$ **do**
      $s_{l+1} \leftarrow O_q(\text{poly}(k, 1/\epsilon))$.
      Choose $\Pi_{l+1} \in \mathbb{R}^{n_{l+1}\cdots n_q \times s_{l+1}}$ to be a TensorSketch.
      for $j_l = 1 \to [n_l]$ **do** ▷ Form $V_l \in \mathbb{R}^{n_l \times k}$
        $(V_l)^{j_l} \leftarrow (V_{l-1})^{\widehat{j_{l-1}}} \circ (U_l)_{j_l}^\top$.
      end for
      for $i = 1 \to n_q$ **do**
        $\beta_i \leftarrow \|(V_l)^i((U_{l+1} \odot \cdots \odot U_q)\Pi_{l+1})\|_2^2$.
      end for
      Sample $\widehat{j_l}$ from $[n_l]$ with probability $\beta_i / \sum_{i'=1}^{n_l} \beta_{i'}$.
    end for
    for $i = 1 \to n_q$ **do**
      $\beta_i \leftarrow |(V_{q-1})^{\widehat{j_{q-1}}}(U_q)_i|^2$.
    end for
    Sample $\widehat{j_q}$ from $[n_q]$ with probability $\beta_i / \sum_{i'=1}^{n_q} \beta_{i'}$.
    $\mathcal{S} \leftarrow \mathcal{S} \cup (\widehat{j_1}, \cdots, \widehat{j_q})$.
  end for
  Convert $\mathcal{S}$ into a diagonal matrix D with at most R_{samples} nonzero entries.
  return D. ▷ Diagonal matrix $D \in \mathbb{R}^{n_1 n_2 \cdots n_q \times n_1 n_2 \cdots n_q}$
end procedure

two sampling procedures are equivalent in terms of sampling a column of a matrix: (1) We sample a single entry from matrix $R^{-1}(U_1 \odot U_2 \odot \cdots \odot U_q)$ proportional to its squared value,

(2) We sample a column from matrix $R^{-1}(U_1 \odot U_2 \odot \cdots \odot U_q)$ proportional to its squared ℓ_2-norm. Let the $(i, j_1, j_2, \cdots, j_q)$-th entry denote the entry in the i-th row and the j-th column, where

$$j = \sum_{l=1}^{q-1} (j_l - 1) \prod_{t=l+1}^{q} n_t + j_q.$$

Similarly to Equation (21.18), we can show, for a particular column j,

$$\Pr[\text{we sample an entry from the } j\text{-th column of matrix}] = \Pr[\text{we sample the } j\text{-th column of a matrix}].$$

Thus, it is sufficient to show how to sample a single entry from matrix $R^{-1}(U_1 \odot U_2 \odot \cdots \odot U_q)$ proportional to its squared value without writing down all the entries of the $k \times \prod_{i=1}^{q} n_i$ matrix.

Let V_0 denote R^{-1}. Let n_0 denote the number of rows of V_0.

In the next few paragraphs, we describe a sampling procedure (procedure FastTensorLeverageScoreGeneralOrder in Algorithm 21.15) which first samples $\widehat{j_0}$ from $[n_0]$, then samples $\widehat{j_1}$ from $[n_1]$, $\cdots$, and at the end samples $\widehat{j_q}$ from $[n_q]$.

In the first step, we want to sample $\widehat{j_0}$ from $[n_0]$ proportional to the squared ℓ_2-norm of that row. To do this efficiently, we choose $\Pi_1 \in \mathbb{R}^{\prod_{i=1}^{q} n_i \times s_1}$ to be a TensorSketch to sketch on the right of $V_0(U_1 \odot U_2 \odot \cdots \odot U_q)$. By Section 21.3.10, as long as $s_1 = O_q(\mathrm{poly}(k, 1/\epsilon))$, then Π_1 is a $(1 \pm \epsilon)$-subspace embedding matrix. Thus with probability $1 - 1/\Omega(q)$, for all $i \in [n_0]$,

$$\|(V_0)^i((U_1 \odot U_2 \odot \cdots \odot U_q)\Pi_1)\|_2^2 = (1 \pm \epsilon)\|(V_0)^i((U_1 \odot U_2 \odot \cdots \odot U_q))\|_2^2,$$

which means we can sample $\widehat{j_0}$ from $[n_0]$ in $O_q(n\,\mathrm{poly}(k, 1/\epsilon))$ time.

In the second step, we have already obtained $\widehat{j_0}$. Using that row of V_0 with U_1, we can form a new matrix $V_1 \in \mathbb{R}^{n_1 \times k}$ in the following sense,

$$(V_1)^i = (V_0)^{\widehat{j_0}} \circ (U_1)_i^\top, \forall i \in [n_1],$$

where $(V_1)^i$ denotes the i-th row of matrix V_1, $(V_0)^{\widehat{j_0}}$ denotes the $\widehat{j_0}$-th row of V_0 and $(U_1)_i$ is the i-th column of U_1. Another important observation is, the entry in the $(j_1, j_2, \cdots, j_q)$-th coordinate of vector $(V_0)^{\widehat{j_0}}(U_1 \odot U_2 \odot \cdots \odot U_q)$ is the same as the entry in the j_1-th row and $(j_2, \cdots, j_q)$-th column of matrix $V_1(U_2 \odot U_3 \odot \cdots \odot U_q)$. Thus, sampling j_1 is equivalent to sampling j_1 from the new matrix $V_1(U_2 \odot U_3 \odot \cdots \odot U_q)$ proportional to the squared ℓ_2-norm of that row. We still have the computational issue that the length of the row vector is very long. To deal with this, we can choose $\Pi_2 \in \mathbb{R}^{\prod_{i=2}^q n_i \times s_2}$ to be a TensorSketch to multiply on the right of $V_1(U_2 \odot U_3 \odot \cdots \odot U_q)$.

By Section 21.3.10, as long as $s_2 = O_q(\mathrm{poly}(k, 1/\epsilon))$, then Π_2 is a $(1 \pm \epsilon)$-subspace embedding matrix. Thus with probability $1 - 1/\Omega(q)$, for all $i \in [n_1]$,

$$\|(V_1)^i((U_2 \odot \cdots \odot U_q)\Pi_2)\|_2^2 = (1 \pm \epsilon)\|(V_1)^i((U_2 \odot \cdots \odot U_q))\|_2^2,$$

which means we can sample $\widehat{j_1}$ from $[n_1]$ in $O_q(n\,\mathrm{poly}(k, 1/\epsilon))$ time.

We repeat the above procedure until we obtain each of $\widehat{j_0}, \widehat{j_1}, \cdots, \widehat{j_q}$. Note that the last one, $\widehat{j_q}$, is easier, since the length of the vector is already small enough, and so we do not need to use TensorSketch for it.

By Section 21.3.10, the time for multiplying by TensorSketch is $O_q(n\,\mathrm{poly}(k, 1/\epsilon))$. Setting ϵ to be a small constant, and taking a union bound over $O(q)$ events completes the proof. □

Lemma 21.5.2. *Given $A \in \mathbb{R}^{n_0 \times \prod_{i=1}^{q} n_i}$, $U_1, U_2, \cdots, U_q \in \mathbb{R}^{k \times n}$, for any $\epsilon > 0$, there exists an algorithm that runs in $O(n \cdot \mathrm{poly}(k, 1/\epsilon))$ time and outputs a diagonal matrix $D \in \mathbb{R}^{\prod_{i=1}^{q} n_i \times \prod_{i=1}^{q} n_i}$ with $m = O(k \log k + k/\epsilon)$ nonzero entries such that,*

$$\|\widehat{U}(U_1 \odot U_2 \odot \cdots \odot U_q) - A\|_F^2 \leq (1+\epsilon) \min_{U \in \mathbb{R}^{n \times k}} \|U(U_1 \odot U_2 \odot \cdots \odot U_q) - A\|_F^2,$$

holds with probability at least 0.999*, where $\widehat{U}$ denotes the optimal solution of*

$$\min_{U \in \mathbb{R}^{n_0 \times k}} \|U(U_1 \odot U_2 \odot \cdots \odot U_q)D - AD\|_F^2.$$

Proof. This follows by combining Theorem 21.5.1, Corollary 21.4.30, and Lemma 21.4.31. □

21.5.2 General iterative existential proof

Algorithm 21.16 General q-th Order Iterative Existential Proof

procedure GeneralIterativeExistentialProof(A, n, k, q, ϵ) ▷ Section 21.5.2
  Fix $U_1^*, U_2^*, \cdots, U_q^* \in \mathbb{R}^{n \times k}$.
  for $i = 1 \to q$ **do**
    Choose sketching matrix $S_i \in \mathbb{R}^{n^{q-1} \times s_i}$ with $s_i = O_q(k/\epsilon)$.
    Define $Z_i \in \mathbb{R}^{k \times n^{q-1}}$ to be $\bigodot_{j<i} \widehat{U}_j^\top \odot \bigodot_{j'>i} U_{j'}^{*\top}$.
    Let A_i denote the matrix obtained by flattening tensor A along the i-th dimension.
    Define $\widehat{U}_i$ to be $A_i S_i (Z_i S_i)^\dagger$.
  end for
  return $\widehat{U}_1, \widehat{U}_2, \cdots, \widehat{U}_q$.
end procedure

Given a q-th order tensor $A \in \mathbb{R}^{n \times n \times \cdots \times n}$, we fix $U_1^*, U_2^*, \cdots, U_q^* \in \mathbb{R}^{n \times k}$ to be the best rank-k solution (if it does not exist, then we replace it by a good approximation, as discussed). We define $\mathrm{OPT} = \|U_1^* \otimes U_2^* \otimes \cdots \otimes U_q^* - A\|_F^2$. Our iterative proof works as follows. We first obtain the objective function,

$$\min_{U_1 \in \mathbb{R}^{n \times k}} \|U_1 \cdot Z_1 - A_1\|_F^2 \leq \mathrm{OPT},$$

where A_1 is a matrix obtained by flattening tensor A along the first dimension, $Z_1 = (U_2^{*\top} \odot U_3^{*\top} \odot \cdots \odot U_q^{*\top})$ denotes a $k \times n^{q-1}$ matrix. Choosing $S_1 \in \mathbb{R}^{n^{q-1} \times s_1}$ to be a Gaussian sketching matrix with $s_1 = O(k/\epsilon)$, we obtain a smaller problem,

$$\min_{U_1 \in \mathbb{R}^{n \times k}} \|U_1 \cdot Z_1 S_1 - A_1 S_1\|_F^2.$$

We define $\widehat{U}_1$ to be $A_1 S_1 (Z_1 S_1)^\dagger \in \mathbb{R}^{n \times k}$, which gives,

$$\|\widehat{U}_1 \cdot Z_1 - A_1\|_F^2 \leq (1+\epsilon)\,\mathrm{OPT}\,.$$

After retensorizing the above, we have,

$$\|\widehat{U}_1 \otimes U_2^* \otimes \cdots \otimes U_q^* - A\|_F^2 \leq (1+\epsilon)\,\mathrm{OPT}\,.$$

In the second round, we fix $\widehat{U}_1$, U_3^*, $\cdots$, $U_q^* \in \mathbb{R}^{n\times k}$, and choose $S_2 \in \mathbb{R}^{n^{q-1}\times s_2}$ to be a Gaussian sketching matrix with $s_2 = O(k/\epsilon)$. We define $Z_2 \in \mathbb{R}^{k\times n^{q-1}}$ to be $(\widehat{U}_1^\top \odot U_3^{*\top} \odot \cdots \odot U_q^{*\top})$. We define $\widehat{U}_2$ to be $A_2 S_2 (Z_2 S_2)^\dagger \in \mathbb{R}^{n\times k}$. Then, we have

$$\|\widehat{U}_1 \otimes \widehat{U}_2 \otimes U_3^* \otimes \cdots \otimes U_q^* - A\|_F^2 \leq (1+\epsilon)^2\,\mathrm{OPT}\,.$$

We repeat the above process, where in the i-th round we fix $\widehat{U}_1, \cdots, \widehat{U}_{i-1}$, U_{i+1}^*, $\cdots$, $U_q^* \in \mathbb{R}^{n\times k}$, and choose $S_i \in \mathbb{R}^{n^{q-1}\times s_i}$ to be a Gaussian sketching matrix with $s_i = O(k/\epsilon)$. We define $Z_i \in \mathbb{R}^{k\times n^{q-1}}$ to be $(\widehat{U}_1^\top \odot \cdots \odot \widehat{U}_{i-1}^\top \odot U_{i+1}^{*\top} \odot \cdots \odot U_q^{*\top})$. We define $\widehat{U}_i$ to be $A_i S_i (Z_i S_i)^\dagger \in \mathbb{R}^{n\times k}$. Then, we have

$$\|\widehat{U}_1 \otimes \cdots \otimes \widehat{U}_{i-1} \otimes \widehat{U}_i \otimes U_{i+1}^* \otimes \cdots \otimes U_q^* - A\|_F^2 \leq (1+\epsilon)^2\,\mathrm{OPT}\,.$$

At the end of the q-th round, we have

$$\|\widehat{U}_1 \otimes \cdots \otimes \widehat{U}_q - A\|_F^2 \leq (1+\epsilon)^q\,\mathrm{OPT}\,.$$

Replacing $\epsilon = \epsilon'/(2q)$, we obtain

$$\|\widehat{U}_1 \otimes \cdots \otimes \widehat{U}_q - A\|_F^2 \leq (1+\epsilon')\,\mathrm{OPT}\,.$$

where for all $i \in [q]$, $s_i = O(kq/\epsilon') = O_q(k/\epsilon')$.

21.5.3 General input sparsity reduction

This section shows how to extend the input sparsity reduction from third order tensors to general q-th order tensors. Given a tensor $A \in \mathbb{R}^{n \times n \times \cdots \times n}$ and q matrices, for each $i \in [q]$, matrix V_i has size $V_i \in \mathbb{R}^{n \times b_i}$, with $b_i \leq \mathrm{poly}(k, 1/\epsilon)$. We choose a batch of sparse embedding matrices $T_i \in \mathbb{R}^{t_i \times n}$. Define $\widehat{V}_i = T_i V_i$, and $C = A(T_1, T_2, \cdots, T_q)$. Thus we have with probability 99/100, for any $\alpha \geq 0$, for all $\{X_i, X_i' \in \mathbb{R}^{b_i \times k}\}_{i \in [q]}$, if

$$\|\widehat{V}_1 X_1' \otimes \widehat{V}_2 X_2' \otimes \cdots \otimes \widehat{V}_q X_q' - C\|_F^2 \leq \alpha \|\widehat{V}_1 X_1 \otimes \widehat{V}_2 X_2 \otimes \cdots \otimes \widehat{V}_q X_q - C\|_F^2,$$

then

$$\|V_1 X_1' \otimes V_2 X_2' \otimes \cdots \otimes V_q X_q' - A\|_F^2 \leq (1+\epsilon)\alpha \|V_1 X_1 \otimes V_2 X_2 \otimes \cdots \otimes V_q X_q - A\|_F^2,$$

where $t_i = O_q(\mathrm{poly}(b_i, 1/\epsilon))$.

Algorithm 21.17 General q-th Order Input Sparsity Reduction

procedure GeneralInputSparsityReduction($A, \{V_i\}_{i \in [q]}, n, k, q, \epsilon$) ▷ Section 21.5.3
  for $i = 1 \to q$ **do**
    Choose sketching matrix $T_i \in \mathbb{R}^{t_i \times n}$ with $t_i = \mathrm{poly}(k, q, 1/\epsilon)$.
    $\widehat{V}_i \leftarrow T_i V_i$.
  end for
  $C \leftarrow A(T_1, T_2, \cdots, T_q)$.
  return $\{\widehat{V}_i\}_{i \in [q]}, C$.
end procedure

21.5.4 Bicriteria algorithm

This section explains how to extend the bicriteria algorithm from third order tensors (Section 21.4.4) to general q-th order tensors. Given any q-th order tensor $A \in \mathbb{R}^{n \times n \times \cdots \times n}$, we can output a rank-$r$ tensor (or equivalently q matrices $U_1, U_2, \cdots, U_q \in \mathbb{R}^{n \times r}$) such that,

$$\|U_1 \otimes U_2 \otimes \cdots \otimes U_q - A\|_F^2 \leq (1+\epsilon)\,\mathrm{OPT},$$

where $r = O_q((k/\epsilon)^{q-1})$ and the algorithm takes $O_q(\mathrm{nnz}(A) + n \cdot \mathrm{poly}(k, 1/\epsilon))$.

Algorithm 21.18 General q-th Order Bicriteria Algorithm

procedure GeneralBicriteriaAlgorithm(A, n, k, q, ϵ) ▷ Section 21.5.4
  for $i = 2 \to q$ **do**
    Choose sketching matrix $S_i \in \mathbb{R}^{n^{q-1} \times s_i}$ with $s_i = O(kq/\epsilon)$.
    Choose sketching matrix $T_i \in \mathbb{R}^{t_i \times n}$ with $t_i = \mathrm{poly}(k, q, 1/\epsilon)$.
    Form matrix $\widehat{U}_i$ by setting $(j_2, j_3, \cdots, j_q)$-th column to be $(A_i S_i)_{j_i}$.
  end for
  Solve $\min_{U_1} \|U_1 B - (A(I, T_2, \cdots, T_q))_1\|_F^2$.
  return $\{\widehat{U}_i\}_{i \in [q]}$.
end procedure

21.5.5 CURT decomposition

This section extends the tensor CURT algorithm from 3rd order tensors (Section 21.4.7) to general q-th order tensors. Given a q-th order tensor $A \in \mathbb{R}^{n \times n \times \cdots \times n}$ and a batch of matrices $U_1, U_2, \cdots, U_q \in \mathbb{R}^{n \times k}$, we iteratively apply the proof in Theorem 21.4.39 (or Theorem 21.4.40) q times. Then for each $i \in [q]$, we are able to select d_i columns from the i-th dimension of tensor A (let C_i denote those columns) and also find a tensor $U \in \mathbb{R}^{d_1 \times d_2 \times \cdots \times d_q}$ such that,

$$\|U(C_1, C_2, \cdots, C_q) - A\|_F^2 \leq (1+\epsilon)\|U_1 \otimes U_2 \otimes \cdots \otimes U_q - A\|_F^2,$$

where either $d_i = O_q(k \log k + k/\epsilon)$ (similar to Theorem 21.4.39) or $d_i = O_q(k/\epsilon)$ (similar to Theorem 21.4.40).

Algorithm 21.19 General q-th Order CURT Decomposition

procedure GeneralCURTDecomposition($A, \{U_i\}_{i\in[q]}, n, k, q, \epsilon$) ▷ Section 21.5.5
  for $i = 1 \to q$ **do**
    Form $B_i = \bigodot_{j<i} \widehat{U}_j^\top \odot \bigodot_{j>i} U_j^\top \in \mathbb{R}^{k \times n^{q-1}}$.
    if fast = true **then** ▷ Optimal running time
      $\epsilon_0 \leftarrow 0.01$.
      $d_i \leftarrow O_q(k \log k + k/\epsilon)$.
      $D_i \leftarrow$ FastTensorLeverageScoreGeneralOrder $(\{\widehat{U}_j\}_{j<i}, \{U_j\}_{j>i}, n, k, \epsilon_0, d_i)$. ▷ Algorithm 21.15
    else ▷ Optimal sample complexity
      $\epsilon_0 \leftarrow O_q(\epsilon)$.
      $D_i \leftarrow$ GeneralizedMatrixRowSubsetSelection $(A_i^\top, B_i^\top, n^{q-1}, n, k, \epsilon_0)$. ▷ Algorithm 21.4.5, $d_i = O_q(k/\epsilon)$.
    end if
    $\widehat{U}_i \leftarrow A_i D_i (B_i D_i)^\dagger$.
    $C_i \leftarrow A_i D_i$.
  end for
  $U \leftarrow (B_1 D_1)^\dagger \otimes (B_2 D_2)^\dagger \otimes \cdots \otimes (B_q D_q)^\dagger$.
  return $\{C_i\}_{i\in[q]}, U$.
end procedure

21.6 Matrix CUR decomposition

There is a long line of research on matrix CUR decomposition under operator, Frobenius or recently, entry-wise ℓ_1 norm [DMM08, BMD09, DR10, BDM11, BW14, SWZ17]. We provide the first algorithm that runs in $\text{nnz}(A)$ time, which improves the previous best matrix CUR decomposition algorithm under Frobenius norm [BW14].

21.6.1 Algorithm

Algorithm 21.20 Optimal Matrix CUR Decomposition Algorithm

procedure OPTIMALMATRIXCUR(A, n, k, ϵ) ▷ Theorem 21.6.1
  $\epsilon' \leftarrow 0.1\epsilon$. $\epsilon'' \leftarrow 0.001\epsilon'$.
  $\widehat{U} \leftarrow$SPARSESVD(A, k, ϵ'). ▷ $\widehat{U} \in \mathbb{R}^{n \times k}$
  Choose $S_1 \in \mathbb{R}^{n \times n}$ to be a sampling and rescaling diagonal matrix according to the leverage scores of $\widehat{U}$ with $s_1 = O(\epsilon^{-2} k \log k)$ nonzero entries.
  $R, Y \leftarrow$GENERALIZEDMATRIXROWSUBSETSELECTION$(S_1 A, S_1 \widehat{U}, s_1, n, k, \epsilon'')$. ▷ Algorithm 21.7, $R \in \mathbb{R}^{r \times n}, Y \in \mathbb{R}^{k \times r}$ and $r = O(k/\epsilon)$
  $\widehat{V} \leftarrow YR \in \mathbb{R}^{k \times n}$.
  Choose $S_2^\top \in \mathbb{R}^{n \times n}$ to be a sampling and rescaling diagonal matrix according to the leverage scores of $\widehat{V}^\top \in \mathbb{R}^{n \times k}$ with $s_2 = O(\epsilon^{-2} k \log k)$ nonzero entries.
  $C^\top, Z^\top \leftarrow$ GENERALIZEDMATRIXROWSUBSETSELECTION $((AS_2)^\top, (\widehat{V} S_2)^\top, s_2, n, k, \epsilon'')$. ▷ Algorithm 21.7, $C \in \mathbb{R}^{n \times c}, Z \in \mathbb{R}^{c \times k}$, and $c = O(k/\epsilon)$
  $U \leftarrow ZY$. ▷ $U \in \mathbb{R}^{c \times r}$ and $\mathrm{rank}(U) = k$
  return C, U, R.
end procedure

Theorem 21.6.1. *Given matrix $A \in \mathbb{R}^{n \times n}$, for any $k \geq 1$ and $\epsilon \in (0, 1)$, there exists an algorithm that takes $O(\mathrm{nnz}(A) + n\,\mathrm{poly}(k, 1/\epsilon))$ time and outputs three matrices $C \in \mathbb{R}^{n \times c}$ with c columns from A, $R \in \mathbb{R}^{r \times n}$ with r rows from A, and $U \in \mathbb{R}^{c \times r}$ with $\mathrm{rank}(U) = k$ such that $r = c = O(k/\epsilon)$ and,*

$$\|CUR - A\|_F^2 \leq (1 + \epsilon) \min_{\mathrm{rank}-k\ A_k} \|A_k - A\|_F^2,$$

holds with probability at least $9/10$.

Proof. We define

$$\mathrm{OPT} = \min_{\mathrm{rank}-k\ A_k} \|A_k - A\|_F^2.$$

We first compute $\widehat{U} \in \mathbb{R}^{n \times k}$ by using the result of [CW13], so that $\widehat{U}$ satisfies:

$$\min_{X \in \mathbb{R}^{k \times n}} \|\widehat{U} X - A\|_F^2 \leq (1+\epsilon)\,\mathrm{OPT}\,. \tag{21.25}$$

This step can be done in $O(\mathrm{nnz}(A) + n\,\mathrm{poly}(k, 1/\epsilon))$ time.

We choose $S_1 \in \mathbb{R}^{n \times n}$ to be a sampling and rescaling diagonal matrix according to the leverage scores of $\widehat{U}$, where here $s_1 = O(\epsilon^{-2} k \log k)$ is the number of samples. This step also can be done in $O(n\,\mathrm{poly}(k, 1/\epsilon))$ time.

We run GeneralizedMatrixRowSubsetSelection(Algorithm 21.7) on matrices $S_1 A$ and $S_1 \widehat{U}$. Then we obtain two new matrices R and Y, where R contains $r = O(k/\epsilon)$ rows of $S_1 A$ and Y has size $k \times r$. According to Theorem 21.4.14 and Corollary 21.4.15, this step takes $n\,\mathrm{poly}(k, 1/\epsilon)$ time.

We construct $\widehat{V} = YR$, and choose $S_2^\top$ to be another sampling and rescaling diagonal matrix according to the leverage scores of $\widehat{V}^\top$ with $s_2 = O(\epsilon^{-2} k \log k)$ nonzero entries. As in the case of constructing S_1, this step can be done in $O(n\,\mathrm{poly}(k, 1/\epsilon))$ time.

We run GeneralizedMatrixRowSubsetSelection(Algorithm 21.7) on matrices $(AS_2)^\top$ and $(\widehat{V} S_2)^\top$. Then we can obtain two new matrices $C^\top$ and $Y^\top$, where $C^\top$ contains $c = O(k/\epsilon)$ rows of $(AS_2)^\top$ and $Z^\top$ has size $k \times c$. According to Theorem 21.4.14 and Corollary 21.4.15, this step takes $n\,\mathrm{poly}(k, 1/\epsilon)$ time.

Thus, overall the running time is $O(\mathrm{nnz}(A) + n\,\mathrm{poly}(k, 1/\epsilon))$.

Correctness. Let

$$X^* = \arg \min_{X \in \mathbb{R}^{n \times k}} \|X \widehat{V} - A\|_F^2.$$

According to Corollary 21.4.15,

$$\|CZ\widehat{V}S_2 - AS_2\|_F^2 \leq (1+\epsilon'') \min_{X\in\mathbb{R}^{n\times k}} \|X\widehat{V}S_2 - AS_2\|_F^2 \leq (1+\epsilon'')\|X^*\widehat{V}S_2 - AS_2\|_F^2.$$

According to Theorem 21.6.5, $\epsilon'' = 0.001\epsilon'$,

$$\|CZ\widehat{V} - A\|_F^2 \leq (1+\epsilon')\|X^*\widehat{V} - A\|_F^2. \tag{21.26}$$

Let

$$\widetilde{X} = \arg\min_{X\in\mathbb{R}^{k\times n}} \|\widehat{U}X - A\|_F^2.$$

According to Corollary 21.4.15,

$$\|S_1\widehat{U}YR - S_1A\|_F^2 \leq (1+\epsilon'') \min_{X\in\mathbb{R}^{k\times n}} \|S_1\widehat{U}X - S_1A\|_F^2 \leq (1+\epsilon'')\|S_1\widehat{U}\widetilde{X} - S_1A\|_F^2.$$

According to Theorem 21.6.5, since $\epsilon'' = 0.001\epsilon'$,

$$\|\widehat{U}YR - A\|_F^2 \leq (1+\epsilon')\|\widehat{U}\widetilde{X} - A\|_F^2. \tag{21.27}$$

Then, we can conclude

$$\begin{aligned}
\|CUR - A\|_F^2 &= \|CZYR - A\|_F^2 \\
&= \|CZ\widehat{V} - A\|_F^2 \\
&\leq (1+\epsilon') \min_{X\in\mathbb{R}^{n\times k}} \|X\widehat{V} - A\|_F^2 \\
&\leq (1+\epsilon')\|\widehat{U}\widehat{V} - A\|_F^2 \\
&\leq (1+\epsilon')^2 \min_{X\in\mathbb{R}^{k\times n}} \|\widehat{U}X - A\|_F^2 \\
&\leq (1+\epsilon')^3\,\mathrm{OPT} \\
&\leq (1+\epsilon)\,\mathrm{OPT}\,.
\end{aligned}$$

The first equality follows since $U = ZY$. The second equality follows since $YR = \widehat{V}$. The first inequality follows by Equation (21.26). The third inequality follows by Equation (21.27). The fourth inequality follows by Equation (21.25). The last inequality follows since $\epsilon' = 0.1\epsilon$.

Notice that C has $O(k/\epsilon)$ reweighted columns of AS_2, and AS_2 is a subset of reweighted columns of A, so C has $O(k/\epsilon)$ reweighted columns of A. Similarly, we can prove that R has $O(k/\epsilon)$ reweighted rows of A. Thus, CUR is a CUR decomposition of A.

□

21.6.2 Stronger property achieved by leverage scores

Claim 21.6.2. *Given matrix $A \in \mathbb{R}^{n\times m}$, for any distribution $p = (p_1, p_2, \cdots, p_n)$ define random variable X such that $X = \|A_i\|_2^2/p_i$ with probability p_i, where A_i is the i-th row of matrix A. Then take m independent samples $X^1, X^2, \cdots, X^m$, and let $Y = \frac{1}{m}\sum_{j=1}^m X^j$. We have*

$$\Pr[Y \leq 100\|A\|_F^2] \geq .99.$$

Proof. We can compute the expectation of X^j, for any $j \in [m]$,

$$\mathbb{E}[X^j] = \sum_{i=1}^n \frac{\|A_i\|_2^2}{p_i} \cdot p_i = \|A\|_F^2.$$

Then $\mathbb{E}[Y] = \frac{1}{m}\sum_{j=1}^m \mathbb{E}[X^j] = \|A\|_F^2$. Using Markov's inequality, we have

$$\Pr[Y \geq \|A\|_F^2] \leq .01.$$

□

Theorem 21.6.3 (The leverage score case of Theorem 39 in [CW13])**.** *Let $A \in \mathbb{R}^{n\times k}$, $B \in \mathbb{R}^{n\times d}$. Let $S \in \mathbb{R}^{n\times n}$ denote a sampling and rescaling diagonal matrix according to the leverage scores of A. If the event occurs that S satisfies $(\epsilon/\sqrt{k})$-Frobenius norm approximate matrix product for A, and also S is a $(1+\epsilon)$-subspace embedding for A, then let X^* be the optimal solution of $\min_X \|AX - B\|_F^2$, and $\widetilde{B} \equiv AX^* - B$. Then, for all $X \in \mathbb{R}^{k\times d}$,*

$$(1-2\epsilon)\|AX - B\|_F^2 \leq \|S(AX - B)\|_F^2 + \|\widetilde{B}\|_F^2 - \|S\widetilde{B}\|_F^2 \leq (1+2\epsilon)\|AX - B\|_F^2.$$

Furthermore, if S has $m = O(\epsilon^{-2}k\log(k))$ nonzero entries, the above event happens with probability at least 0.99.

Note that Theorem 39 in [CW13] is stated in a way that holds for general sketching matrices. However, we are only interested in the case when S is a sampling and rescaling diagonal matrix according to the leverage scores. For completeness, we provide the full proof of the leverage score case with certain parameters.

Proof. Suppose S is a sampling and rescaling diagonal matrix according to the leverage scores of A, and it has $m = O(\epsilon^{-2} k \log k)$ nonzero entries. Then, according to Lemma 21.4.22, S is a $(1+\epsilon)$-subspace embedding for A with probability at least 0.999, and according to Lemma 21.4.29, S satisfies $(\epsilon/\sqrt{k})$-Frobenius norm approximate matrix product for A with probability at least 0.999.

Let $U \in \mathbb{R}^{n \times k}$ denote an orthonormal basis of the column span of A. Then the leverage scores of U are the same as the leverage scores of A. Furthermore, for any $X \in \mathbb{R}^{k \times d}$, there is a matrix Y such that $AX = UY$, and vice versa, so we can now assume A has k orthonormal columns.

Then,

$$\begin{aligned}
&\|S(AX - B)\|_F^2 - \|S\widetilde{B}\|_F^2 \\
&= \|SA(X - X^*) + S(AX^* - B)\|_F^2 - \|S\widetilde{B}\|_F^2 \\
&= \|SA(X - X^*)\|_F^2 + \|S(AX^* - B)\|_F^2 + 2\operatorname{tr}\left((X - X^*)^\top A^\top S^\top S(AX^* - B)\right) - \|S\widetilde{B}\|_F^2 \\
&= \underbrace{\|SA(X - X^*)\|_F^2 + 2\operatorname{tr}\left((X - X^*)^\top A^\top S^\top S\widetilde{B}\right)}_{\alpha}. \qquad (21.28)
\end{aligned}$$

The second equality follows using $\|C+D\|_F^2 = \|C\|_F^2 + \|D\|_F^2 + 2\operatorname{tr}(C^\top D)$. The third equality

follows from $\widetilde{B} = AX^* - B$. Now, let us first upper bound the term α in Equation (21.28):

$$\begin{aligned}
&\|SA(X - X^*)\|_F^2 + 2\,\mathrm{tr}\left((X - X^*)^\top A^\top S^\top S\widetilde{B}\right) \\
&\leq (1+\epsilon)\|A(X - X^*)\|_F^2 + 2\|X - X^*\|_F\|A^\top S^\top S\widetilde{B}\|_F \\
&\leq (1+\epsilon)\|A(X - X^*)\|_F^2 + 2(\epsilon/\sqrt{k}) \cdot \|X - X^*\|_F\|A\|_F\|\widetilde{B}\|_F \\
&\leq (1+\epsilon)\|A(X - X^*)\|_F^2 + 2\epsilon\|A(X - X^*)\|_F\|\widetilde{B}\|_F.
\end{aligned}$$

The first inequality follows since S is a $(1+\epsilon)$ subspace embedding of A, and $\mathrm{tr}(C^\top D) \leq \|C\|_F\|D\|_F$. The second inequality follows since S satisfies $(\epsilon/\sqrt{k})$-Frobenius norm approximate matrix product for A. The last inequality follows using that $\|A\|_F \leq \sqrt{k}$ since A only has k orthonormal columns. Now, let us lower bound the term α in Equation (21.28):

$$\begin{aligned}
&\|SA(X - X^*)\|_F^2 + 2\,\mathrm{tr}\left((X - X^*)^\top A^\top S^\top S\widetilde{B}\right) \\
&\geq (1-\epsilon)\|A(X - X^*)\|_F^2 - 2\|X - X^*\|_F\|A^\top S^\top S\widetilde{B}\|_F \\
&\geq (1-\epsilon)\|A(X - X^*)\|_F^2 - 2(\epsilon/\sqrt{k}) \cdot \|X - X^*\|_F\|A\|_F\|\widetilde{B}\|_F \\
&\geq (1-\epsilon)\|A(X - X^*)\|_F^2 - 2\epsilon\|A(X - X^*)\|_F\|\widetilde{B}\|_F.
\end{aligned}$$

The first inequality follows since S is a $(1+\epsilon)$ subspace embedding of A, and $\mathrm{tr}(C^\top D) \geq -\|C\|_F\|D\|_F$. The second inequality follows since S satisfies $(\epsilon/\sqrt{k})$-Frobenius norm approximate matrix product for A. The last inequality follows using that $\|A\|_F \leq \sqrt{k}$ since A only has k orthonormal columns.

Therefore,

$$(1-\epsilon)\|A(X - X^*)\|_F^2 - 2\epsilon\|A(X - X^*)\|_F\|\widetilde{B}\|_F \leq \|S(AX - B)\|_F^2 - \|S\widetilde{B}\|_F^2, \qquad (21.29)$$

and

$$(1+\epsilon)\|A(X-X^*)\|_F^2 + 2\epsilon\|A(X-X^*)\|_F\|\widetilde{B}\|_F \geq \|S(AX-B)\|_F^2 - \|S\widetilde{B}\|_F^2. \tag{21.30}$$

Notice that $\widetilde{B} = AX^* - B = AA^\dagger B - B = (AA^\dagger - I)B$, so according to the Pythagorean theorem, we have

$$\|AX-B\|_F^2 = \|A(X-X^*)\|_F^2 + \|\widetilde{B}\|_F^2,$$

which means that

$$\|A(X-X^*)\|_F^2 = \|AX-B\|_F^2 - \|\widetilde{B}\|_F^2. \tag{21.31}$$

Using Equation (21.31), we can rewrite and lower bound the LHS of Equation (21.29),

$$\begin{aligned}
&(1-\epsilon)\|A(X-X^*)\|_F^2 - 2\epsilon\|A(X-X^*)\|_F\|\widetilde{B}\|_F \\
&= \|A(X-X^*)\|_F^2 - \epsilon\left(\|A(X-X^*)\|_F^2 + 2\|A(X-X^*)\|_F\|\widetilde{B}\|_F\right) \\
&= \|AX-B\|_F^2 - \|\widetilde{B}\|_F^2 - \epsilon\left(\|A(X-X^*)\|_F^2 + 2\|A(X-X^*)\|_F\|\widetilde{B}\|_F\right) \\
&\geq \|AX-B\|_F^2 - \|\widetilde{B}\|_F^2 - \epsilon\left(\|A(X-X^*)\|_F + \|\widetilde{B}\|_F\right)^2 \\
&\geq \|AX-B\|_F^2 - \|\widetilde{B}\|_F^2 - 2\epsilon\left(\|A(X-X^*)\|_F^2 + \|\widetilde{B}\|_F^2\right) \\
&= (1-2\epsilon)\|AX-B\|_F^2 - \|\widetilde{B}\|_F^2.
\end{aligned} \tag{21.32}$$

The second step follows by Equation (21.31). The first inequality follows using $a^2 + 2ab < (a+b)^2$. The second inequality follows using $(a+b)^2 \leq 2(a^2+b^2)$. The last equality follows using $\|A(X-X^*)\|_F^2 + \|\widetilde{B}\|_F^2 = \|AX-B\|_F^2$. Similarly, using Equation (21.31), we can rewrite and upper bound the LHS of Equation (21.30)

$$(1+\epsilon)\|A(X-X^*)\|_F^2 + 2\epsilon\|A(X-X^*)\|_F\|\widetilde{B}\|_F \leq (1+2\epsilon)\|AX-B\|_F^2 - \|\widetilde{B}\|_F^2. \tag{21.33}$$

Combining Equations (21.29),(21.32),(21.30),(21.33), we conclude that

$$(1-2\epsilon)\|AX-B\|_F^2 - \|\widetilde{B}\|_F^2 \leq \|S(AX-B)\|_F^2 - \|S\widetilde{B}\|_F^2 \leq (1+2\epsilon)\|AX-B\|_F^2 - \|\widetilde{B}\|_F^2.$$

□

Theorem 21.6.4. *Let $A \in \mathbb{R}^{n\times k}$, $B \in \mathbb{R}^{n\times d}$, and $\frac{1}{2} > \epsilon > 0$. Let X^* be the optimal solution to $\min_X \|AX-B\|_F^2$, and $\widetilde{B} \equiv AX^* - B$. Let $S \in \mathbb{R}^{n\times n}$ denote a sketching matrix which satisfies the following:*

1. $\|S\widetilde{B}\|_F^2 \leq 100 \cdot \|\widetilde{B}\|_F^2$,
2. *for all $X \in \mathbb{R}^{k\times d}$,*

$$(1-\epsilon)\|AX-B\|_F^2 \leq \|S(AX-B)\|_F^2 + \|\widetilde{B}\|_F^2 - \|S\widetilde{B}\|_F^2 \leq (1+\epsilon)\|AX-B\|_F^2.$$

Then, for all $X_1, X_2 \in \mathbb{R}^{k\times d}$ satisfying

$$\|SAX_1 - SB\|_F^2 \leq \left(1+\frac{\epsilon}{100}\right) \cdot \|SAX_2 - SB\|_F^2,$$

we have

$$\|AX_1 - B\|_F^2 \leq (1+5\epsilon) \cdot \|AX_2 - B\|_F^2.$$

Proof. Let A, B, S, ϵ be the same as in the statement of the theorem, and suppose S satisfies those two conditions. Let $X_1, X_2 \in \mathbb{R}^{k\times d}$ satisfy

$$\|SAX_1 - SB\|_F^2 \leq \left(1+\frac{\epsilon}{100}\right) \|SAX_2 - SB\|_F^2.$$

We have

$$\begin{aligned}
&\|AX_1 - B\|_F^2 \\
&\leq \frac{1}{1-\epsilon}\left(\|S(AX_1 - B)\|_F^2 + \|\widetilde{B}\|_F^2 - \|S\widetilde{B}\|_F^2\right) \\
&\leq \frac{1}{1-\epsilon}\left(\left(1+\frac{\epsilon}{100}\right)\cdot\|S(AX_2 - B)\|_F^2 + \|\widetilde{B}\|_F^2 - \|S\widetilde{B}\|_F^2\right) \\
&= \frac{1}{1-\epsilon}\left(\left(1+\frac{\epsilon}{100}\right)\cdot\left(\|S(AX_2 - B)\|_F^2 + \|\widetilde{B}\|_F^2 - \|S\widetilde{B}\|_F^2\right) - \frac{\epsilon}{100}\cdot\left(\|\widetilde{B}\|_F^2 - \|S\widetilde{B}\|_F^2\right)\right) \\
&\leq \frac{1}{1-\epsilon}\cdot\left(1+\frac{\epsilon}{100}\right)\cdot\|AX_2 - B\|_F^2 - \frac{1}{1-\epsilon}\cdot\frac{\epsilon}{100}\cdot\left(\|\widetilde{B}\|_F^2 - \|S\widetilde{B}\|_F^2\right) \\
&\leq (1+3\epsilon)\|AX_2 - B\|_F^2 + \frac{1}{1-\epsilon}\cdot\frac{\epsilon}{100}\|S\widetilde{B}\|_F^2 \\
&\leq (1+3\epsilon)\|AX_2 - B\|_F^2 + 2\epsilon\|\widetilde{B}\|_F^2 \\
&\leq (1+5\epsilon)\|AX_2 - B\|_F^2.
\end{aligned}$$

The first inequality follows since S satisfies the second condition. The second inequality follows by the relationship between X_1 and X_2. The third inequality follows since S satisfies the second condition. The fifth inequality follows using that $\epsilon < \frac{1}{2}$ and that S satisfies the first condition. The last inequality follows using that $\|\widetilde{B}\|_F^2 = \|AX^* - B\|_F^2 \leq \|AX_2 - B\|_F^2$. □

Theorem 21.6.5. *Let $A \in \mathbb{R}^{n\times k}$, $B \in \mathbb{R}^{n\times d}$, and $\frac{1}{2} > \epsilon > 0$. Let $S \in \mathbb{R}^{n\times n}$ denote a sampling and rescaling diagonal matrix according to the leverage scores of A. If S has at least $m = O(k\log(k)/\epsilon^2)$ nonzero entries, then with probability at least 0.98, for all $X_1, X_2 \in \mathbb{R}^{k\times d}$ satisfying*

$$\|SAX_1 - SB\|_F^2 \leq (1+\frac{\epsilon}{500})\cdot\|SAX_2 - SB\|_F^2,$$

we have

$$\|AX_1 - B\|_F^2 \leq (1+\epsilon)\cdot\|AX_2 - B\|_F^2.$$

Proof. The proof directly follows by Claim 21.6.2, Theorem 21.6.3 and Theorem 21.6.4. Because of Claim 21.6.2, S satisfies the first condition in the statement of Theorem 21.6.4 with probability at least 0.99. According to Theorem 21.6.3, S satisfies the second condition in the statement of Theorem 21.6.4 with probability at least 0.99. Thus, with probability 0.98, by Theorem 21.6.4, we complete the proof. □

21.7 Entry-wise ℓ_1 Norm for Arbitrary Tensors

In this section, we provide several different algorithms for tensor ℓ_1-low rank approximation. Section 21.7.1 provides some useful facts and definitions. Section 21.7.2 presents several existence results. Section 21.7.3 describes a tool that is able to reduce the size of the objective function from poly(n) to poly(k). Section 21.7.4 discusses the case when the problem size is small. Section 21.7.5 provides several bicriteria algorithms. Section 21.7.6 summarizes a batch of algorithms. Section 21.7.7 provides an algorithm for ℓ_1 norm CURT decomposition.

Notice that if the rank $-k$ solution does not exist, then every bicriteria algorithm in Section 21.7.5 can be stated in a form similar to Theorem 21.1.1, and every algorithm which can output a rank $-k$ solution in Section 21.7.6 can be stated in a form similar to Theorem 21.1.2. See Section 21.1 for more details.

21.7.1 Facts

We present a method that is able to reduce the entry-wise ℓ_1-norm objective function to the Frobenius norm objective function.

Fact 21.7.1. *Given a 3rd order tensor $C \in \mathbb{R}^{c_1 \times c_2 \times c_3}$, three matrices $V_1 \in \mathbb{R}^{c_1 \times b_1}$, $V_2 \in \mathbb{R}^{c_2 \times b_2}$, $V_3 \in \mathbb{R}^{c_3 \times b_3}$, for any $k \in [1, \min_i b_i]$, if $X_1' \in \mathbb{R}^{b_1 \times k}, X_2' \in \mathbb{R}^{b_2 \times k}, X_3' \in \mathbb{R}^{b_3 \times k}$ satisfies that,*

$$\|(V_1 X_1') \otimes (V_2 X_2') \otimes (V_3 X_3') - C\|_F \leq \alpha \min_{X_1, X_2, X_3} \|(V_1 X_1) \otimes (V_2 X_2) \otimes (V_3 X_3) - C\|_F,$$

then

$$\|(V_1 X_1') \otimes (V_2 X_2') \otimes (V_3 X_3') - C\|_1 \leq \alpha \sqrt{c_1 c_2 c_3} \min_{X_1, X_2, X_3} \|(V_1 X_1) \otimes (V_2 X_2) \otimes (V_3 X_3) - C\|_1.$$

We extend Lemma C.15 in [SWZ17] to tensors:

Fact 21.7.2. *Given tensor $A \in \mathbb{R}^{n \times n \times n}$, let $\mathrm{OPT} = \min_{\mathrm{rank}-k\ A_k} \|A - A_k\|_1$. For any $r \geq k$, if* rank-r *tensor $B \in \mathbb{R}^{n \times n \times n}$ is an f-approximation to A, i.e.,*

$$\|B - A\|_1 \leq f \cdot \mathrm{OPT},$$

and $U, V, W \in \mathbb{R}^{n \times k}$ is a g-approximation to B, i.e.,

$$\|U \otimes V \otimes W - B\|_1 \leq g \cdot \min_{\mathrm{rank}-k\ B_k} \|B_k - B\|_1,$$

then,

$$\|U \otimes V \otimes W - A\|_1 \lesssim gf \cdot \mathrm{OPT}\,.$$

Proof. We define $\widetilde{U}, \widetilde{V}, \widetilde{W} \in \mathbb{R}^{n\times k}$ to be three matrices, such that

$$\|\widetilde{U} \otimes \widetilde{V} \otimes \widetilde{W} - B\|_1 \leq g \min_{\mathrm{rank}-k\ B_k} \|B_k - B\|_1,$$

and also define,

$$\widehat{U}, \widehat{V}, \widehat{W} = \arg\min_{U,V,W\in\mathbb{R}^{n\times k}} \|U \otimes V \otimes W - B\|_1 \text{ and } U^*, V^*, W^* = \arg\min_{U,V,W\in\mathbb{R}^{n\times k}} \|U \otimes V \otimes W - A\|_1.$$

It is obvious that,

$$\|\widehat{U} \otimes \widehat{V} \otimes \widehat{W} - B\|_1 \leq \|U^* \otimes V^* \otimes W^* - B\|_1. \tag{21.34}$$

Then,

$$\begin{aligned}
&\|\widetilde{U} \otimes \widetilde{V} \otimes \widetilde{W} - A\|_1 \\
&\leq \|\widetilde{U} \otimes \widetilde{V} \otimes \widetilde{W} - B\|_1 + \|B - A\|_1 && \text{by the triangle inequality} \\
&\leq g\|\widehat{U} \otimes \widehat{V} \otimes \widehat{W} - B\|_1 + \|B - A\|_1 && \text{by definition} \\
&\leq g\|U^* \otimes V^* \otimes W^* - B\|_1 + \|B - A\|_1 && \text{by Equation (21.34)} \\
&\leq g\|U^* \otimes V^* \otimes W^* - A\|_1 + g\|B - A\|_1 + \|B - A\|_1 && \text{by the triangle inequality} \\
&= g\,\mathrm{OPT} + (g+1)\|B - A\|_1 && \text{by definition of OPT} \\
&\leq g\,\mathrm{OPT} + (g+1)f \cdot \mathrm{OPT} && \text{since } B \text{ is an } f\text{-approximation to } A \\
&\lesssim gf\,\mathrm{OPT}\,.
\end{aligned}$$

This completes the proof. □

Using the above fact, we are able to optimize our approximation ratio.

21.7.2 Existence results

Definition 21.7.1 (ℓ_1 multiple regression cost preserving sketch - Definition D.5 in [SWZ17])**.** Given matrices $U \in \mathbb{R}^{n\times r}, A \in \mathbb{R}^{n\times d}$, let $S \in \mathbb{R}^{m\times n}$. If $\forall \beta \geq 1, \widehat{V} \in \mathbb{R}^{r\times d}$ which satisfy

$$\|SU\widehat{V} - SA\|_1 \leq \beta \cdot \min_{V\in\mathbb{R}^{r\times d}} \|SUV - SA\|_1,$$

it holds that

$$\|U\widehat{V} - A\|_1 \leq \beta \cdot c \cdot \min_{V\in\mathbb{R}^{r\times d}} \|UV - A\|_1,$$

then S provides a c-ℓ_1-multiple-regression-cost-preserving-sketch for (U, A).

Theorem 21.7.3. *Given a 3rd order tensor $A \in \mathbb{R}^{n\times n\times n}$, for any $k \geq 1$, there exist three matrices $S_1 \in \mathbb{R}^{n^2\times s_1}$, $S_2 \in \mathbb{R}^{n^2\times s_2}$, $S_3 \in \mathbb{R}^{n^2\times s_3}$ such that*

$$\min_{X_1,X_2,X_3} \left\| \sum_{i=1}^{k} (A_1S_1X_1)_i \otimes (A_2S_2X_2)_i \otimes (A_3S_3X_3)_i - A \right\|_1 \leq \alpha \min_{\text{rank}-k\ A_k\in\mathbb{R}^{n\times n\times n}} \|A_k - A\|_1,$$

holds with probability 99/100*.*

(I). *Using a dense Cauchy transform,*

$s_1 = s_2 = s_3 = \widetilde{O}(k)$, $\alpha = \widetilde{O}(k^{1.5}) \log^3 n$.

(II). *Using a sparse Cauchy transform,*

$s_1 = s_2 = s_3 = \widetilde{O}(k^5)$, $\alpha = \widetilde{O}(k^{13.5}) \log^3 n$.

(III). *Guessing Lewis weights,*

$s_1 = s_2 = s_3 = \widetilde{O}(k)$, $\alpha = \widetilde{O}(k^{1.5})$.

Proof. We use OPT to denote

$$\text{OPT} := \min_{\text{rank}-k\ A_k\in\mathbb{R}^{n\times n\times n}} \|A_k - A\|_1.$$

Given a tensor $A \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, we define three matrices $A_1 \in \mathbb{R}^{n_1 \times n_2 n_3}$, $A_2 \in \mathbb{R}^{n_2 \times n_3 n_1}$, $A_3 \in \mathbb{R}^{n_3 \times n_1 n_2}$ such that, for any $i \in [n_1], j \in [n_2], l \in [n_3]$,

$$A_{i,j,l} = (A_1)_{i,(j-1)\cdot n_3 + l} = (A_2)_{j,(l-1)\cdot n_1 + i} = (A_3)_{l,(i-1)\cdot n_2 + j}.$$

We fix $V^* \in \mathbb{R}^{n \times k}$ and $W^* \in \mathbb{R}^{n \times k}$, and use $V_1^*, V_2^*, \cdots, V_k^*$ to denote the columns of V^* and $W_1^*, W_2^*, \cdots, W_k^*$ to denote the columns of W^*.

We consider the following optimization problem,

$$\min_{U_1, \cdots, U_k \in \mathbb{R}^n} \left\| \sum_{i=1}^k U_i \otimes V_i^* \otimes W_i^* - A \right\|_1,$$

which is equivalent to

$$\min_{U_1, \cdots, U_k \in \mathbb{R}^n} \left\| \begin{bmatrix} U_1 & U_2 & \cdots & U_k \end{bmatrix} \begin{bmatrix} V_1^* \otimes W_1^* \\ V_2^* \otimes W_2^* \\ \cdots \\ V_k^* \otimes W_k^* \end{bmatrix} - A \right\|_1.$$

We use matrix Z_1 to denote $V^{*\top} \odot W^{*\top} \in \mathbb{R}^{k \times n^2}$ and matrix U to denote $\begin{bmatrix} U_1 & U_2 & \cdots & U_k \end{bmatrix}$. Then we can obtain the following equivalent objective function,

$$\min_{U \in \mathbb{R}^{n \times k}} \|U Z_1 - A_1\|_1.$$

Choose an ℓ_1 multiple regression cost preserving sketch $S_1 \in \mathbb{R}^{n^2 \times s_1}$ for $(Z_1^\top, A_1^\top)$. We can obtain the optimization problem,

$$\min_{U \in \mathbb{R}^{n \times k}} \|U Z_1 S_1 - A_1 S_1\|_1 = \min_{U \in \mathbb{R}^{n \times k}} \sum_{i=1}^n \|U^i Z_1 S_1 - (A_1 S_1)^i\|_1,$$

where U^i denotes the i-th row of matrix $U \in \mathbb{R}^{n\times k}$ and $(A_1S_1)^i$ denotes the i-th row of matrix A_1S_1. Instead of solving it under the ℓ_1-norm, we consider the ℓ_2-norm relaxation,

$$\min_{U\in\mathbb{R}^{n\times k}} \|UZ_1S_1 - A_1S_1\|_F^2 = \min_{U\in\mathbb{R}^{n\times k}} \sum_{i=1}^{n} \|U^iZ_1S_1 - (A_1S_1)^i\|_2^2.$$

Let $\widehat{U} \in \mathbb{R}^{n\times k}$ denote the optimal solution of the above optimization problem. Then, $\widehat{U} = A_1S_1(Z_1S_1)^\dagger$. We plug $\widehat{U}$ into the objective function under the ℓ_1-norm. According to Claim 21.3.4, we have,

$$\|\widehat{U}Z_1S_1 - A_1S_1\|_1 = \sum_{i=1}^{n} \|\widehat{U}^iZ_1S_1 - (A_1S_1)^i\|_1 \leq \sqrt{s_1} \min_{U\in\mathbb{R}^{n\times k}} \|UZ_1S_1 - A_1S_1\|_1.$$

Since $S_1 \in \mathbb{R}^{n^2\times s_1}$ satisfies Definition 21.7.1, we have

$$\|\widehat{U}Z_1 - A_1\|_1 \leq \alpha \min_{U\in\mathbb{R}^{n\times k}} \|UZ_1 - A_1\|_1 = \alpha\,\mathrm{OPT},$$

where $\alpha = \sqrt{s_1}\beta$ and β (see Definition 21.7.1) is a parameter which depends on which kind of sketching matrix we actually choose. It implies

$$\|\widehat{U} \otimes V^* \otimes W^* - A\|_1 \leq \alpha\,\mathrm{OPT}\,.$$

As a second step, we fix $\widehat{U} \in \mathbb{R}^{n\times k}$ and $W^* \in \mathbb{R}^{n\times k}$, and convert tensor A into matrix A_2. Let matrix Z_2 denote $\widehat{U}^\top \odot W^{*\top}$. We consider the following objective function,

$$\min_{V\in\mathbb{R}^{n\times k}} \|VZ_2 - A_2\|_1,$$

and the optimal cost of it is at most $\alpha\,\mathrm{OPT}$.

Choose an ℓ_1 multiple regression cost preserving sketch $S_2 \in \mathbb{R}^{n^2\times s_2}$ for $(Z_2^\top, A_2^\top)$, and sketch on the right of the objective function to obtain this new objective function,

$$\min_{V\in\mathbb{R}^{n\times k}} \|VZ_2S_2 - A_2S_2\|_1 = \min_{U\in\mathbb{R}^{n\times k}} \sum_{i=1}^{n} \|V^iZ_2S_2 - (A_2S_2)^i\|_1,$$

where V^i denotes the i-th row of matrix V and $(A_2S_2)^i$ denotes the i-th row of matrix A_2S_2. Instead of solving this under the ℓ_1-norm, we consider the ℓ_2-norm relaxation,

$$\min_{U\in\mathbb{R}^{n\times k}}\|VZ_2S_2 - A_2S_2\|_F^2 = \min_{V\in\mathbb{R}^{n\times k}}\|V^i(Z_2S_2) - (A_2S_2)^i\|_2^2.$$

Let $\widehat{V} \in \mathbb{R}^{n\times k}$ denote the optimal solution of the above problem. Then $\widehat{V} = A_2S_2(Z_2S_2)^\dagger$. By properties of the sketching matrix $S_2 \in \mathbb{R}^{n^2\times s_2}$, we have,

$$\|\widehat{V}Z_2 - A_2\|_1 \leq \alpha \min_{V\in\mathbb{R}^{n\times k}}\|VZ_2 - A_2\|_1 \leq \alpha^2\,\mathrm{OPT},$$

which implies

$$\|\widehat{U}\otimes\widehat{V}\otimes W^* - A\|_1 \leq \alpha^2\,\mathrm{OPT}\,.$$

As a third step, we fix the matrices $\widehat{U} \in \mathbb{R}^{n\times k}$ and $\widehat{V} \in \mathbb{R}^{n\times k}$. We can convert tensor $A \in \mathbb{R}^{n\times n\times n}$ into matrix $A_3 \in \mathbb{R}^{n^2\times n}$. Let matrix Z_3 denote $\widehat{U}^\top \odot \widehat{V}^\top \in \mathbb{R}^{k\times n^2}$. We consider the following objective function,

$$\min_{W\in\mathbb{R}^{n\times k}}\|WZ_3 - A_3\|_1,$$

and the optimal cost of it is at most $\alpha^2\,\mathrm{OPT}$.

Choose an ℓ_1 multiple regression cost preserving sketch $S_3 \in \mathbb{R}^{n^2\times s_3}$ for $(Z_3^\top, A_3^\top)$ and sketch on the right of the objective function to obtain the new objective function,

$$\min_{W\in\mathbb{R}^{n\times k}}\|WZ_3S_3 - A_3S_3\|_1.$$

Let $\widehat{W} \in \mathbb{R}^{n\times k}$ denote the optimal solution of the above problem. Then $\widehat{W} = A_3S_3(Z_3S_3)^\dagger$. By properties of sketching matrix $S_3 \in \mathbb{R}^{n^2\times s_3}$, we have,

$$\|\widehat{W}Z_3 - A_3\|_1 \leq \alpha \min_{W\in\mathbb{R}^{n\times k}}\|WZ_3 - A_3\|_1 \leq \alpha^3\,\mathrm{OPT}\,.$$

Thus, we obtain,

$$\min_{X_1\in\mathbb{R}^{s_1\times k},X_2\in\mathbb{R}^{s_2\times k},X_3\in\mathbb{R}^{s_3\times k}}\left\|\sum_{i=1}^{k}(A_1S_1X_1)_i\otimes(A_2S_2X_2)_i\otimes(A_3S_3X_3)_i-A\right\|_1\leq\alpha^3\,\mathrm{OPT}\,.$$

Proof of (I) By Theorem C.1 in [SWZ17], we can use dense Cauchy transforms for S_1, S_2, S_3, and then $s_1 = s_2 = s_3 = O(k\log k)$ and $\alpha = O(\sqrt{k\log k}\log n)$.

Proof of (II) By Theorem C.1 in [SWZ17], we can use sparse Cauchy transforms for S_1, S_2, S_3, and then $s_1 = s_2 = s_3 = O(k^5\log^5 k)$ and $\alpha = O(k^{4.5}\log^{4.5} k\log n)$.

Proof of (III) By Theorem C.1 in [SWZ17], we can sample by Lewis weights. Then $S_1, S_2, S_3 \in \mathbb{R}^{n^2\times n^2}$ are diagonal matrices, and each of them has $O(k\log k)$ nonzero rows. This gives $\alpha = O(\sqrt{k\log k})$.

□

21.7.3 Polynomial in k size reduction

Definition 21.7.2 (Definition D.1 in [SWZ17])**.** Given a matrix $M \in \mathbb{R}^{n\times d}$, if matrix $S \in \mathbb{R}^{m\times n}$ satisfies

$$\|SM\|_1 \le \beta\|M\|_1,$$

then S has at most β dilation on M.

Definition 21.7.3 (Definition D.2 in [SWZ17])**.** Given a matrix $U \in \mathbb{R}^{n\times k}$, if matrix $S \in \mathbb{R}^{m\times n}$ satisfies

$$\forall x \in \mathbb{R}^k, \|SUx\|_1 \ge \frac{1}{\beta}\|Ux\|_1,$$

then S has at most β contraction on U.

Theorem 21.7.4. *Given a tensor $A \in \mathbb{R}^{n_1\times n_2\times n_3}$ and three matrices $V_1 \in \mathbb{R}^{n_1\times b_1}, V_2 \in \mathbb{R}^{n_2\times b_2}, V_3 \in \mathbb{R}^{n_3\times b_3}$, let $X_1^* \in \mathbb{R}^{b_1\times k}, X_2^* \in \mathbb{R}^{b_2\times k}, X_3^* \in \mathbb{R}^{b_3\times k}$ satisfies*

$$X_1^*, X_2^*, X_3^* = \arg\min_{X_1\in\mathbb{R}^{b_1\times k}, X_2\in\mathbb{R}^{b_2\times k}, X_3\in\mathbb{R}^{b_3\times k}} \|V_1X_1 \otimes V_2X_2 \otimes V_3X_3 - A\|_1.$$

Let $S \in \mathbb{R}^{m\times n}$ have at most $\beta_1 \ge 1$ dilation on $V_1X_1^* \cdot ((V_2X_2^*)^\top \odot (V_3X_3^*)^\top) - A_1$ and S have at most $\beta_2 \ge 1$ contraction on V_1. If $\widehat{X}_1 \in \mathbb{R}^{b_1\times k}, \widehat{X}_2 \in \mathbb{R}^{b_2\times k}, \widehat{X}_3 \in \mathbb{R}^{b_3\times k}$ satisfies

$$\|SV_1\widehat{X}_1 \otimes V_2\widehat{X}_2 \otimes V_3\widehat{X}_3 - SA\|_1 \le \beta \min_{X_1\in\mathbb{R}^{b_1\times k}, X_2\in\mathbb{R}^{b_2\times k}, X_3\in\mathbb{R}^{b_3\times k}} \|SV_1X_1 \otimes V_2X_2 \otimes V_3X_3 - SA\|_1,$$

where $\beta \ge 1$, then

$$\|V_1\widehat{X}_1 \otimes V_2\widehat{X}_2 \otimes V_3\widehat{X}_3 - A\|_1 \lesssim \beta_1\beta_2\beta \min_{X_1,X_2,X_3} \|V_1X_1 \otimes V_2X_2 \otimes V_3X_3 - A\|_1.$$

The proof idea is similar to [SWZ17].

Proof. Let $A, V_1, V_2, V_3, S, X_1^*, X_2^*, X_3^*, \beta_1, \beta_2$ be the same as stated in the theorem. Let $\widehat{X}_1 \in \mathbb{R}^{b_1 \times k}, \widehat{X}_2 \in \mathbb{R}^{b_2 \times k}, \widehat{X}_3 \in \mathbb{R}^{b_3 \times k}$ satisfy

$$\|SV_1\widehat{X}_1 \otimes V_2\widehat{X}_2 \otimes V_3\widehat{X}_3 - SA\|_1 \leq \beta \min_{X_1 \in \mathbb{R}^{b_1 \times k}, X_2 \in \mathbb{R}^{b_2 \times k}, X_3 \in \mathbb{R}^{b_3 \times k}} \|SV_1X_1 \otimes V_2X_2 \otimes V_3X_3 - SA\|_1.$$

We have,

$$\begin{aligned}
&\|SV_1\widehat{X}_1 \otimes V_2\widehat{X}_2 \otimes V_3\widehat{X}_3 - SA\|_1 \\
\geq\ &\|SV_1\widehat{X}_1 \otimes V_2\widehat{X}_2 \otimes V_3\widehat{X}_3 - SV_1X_1^* \otimes V_2X_2^* \otimes V_3X_3^*\|_1 - \|SV_1X_1^* \otimes V_2X_2^* \otimes V_3X_3^* - SA\|_1 \\
\geq\ &\frac{1}{\beta_2}\|V_1\widehat{X}_1 \otimes V_2\widehat{X}_2 \otimes V_3\widehat{X}_3 - V_1X_1^* \otimes V_2X_2^* \otimes V_3X_3^*\|_1 - \beta_1\|V_1X_1^* \otimes V_2X_2^* \otimes V_3X_3^* - A\|_1 \\
\geq\ &\frac{1}{\beta_2}\|V_1\widehat{X}_1 \otimes V_2\widehat{X}_2 \otimes V_3\widehat{X}_3 - A\|_1 - \frac{1}{\beta_2}\|V_1X_1^* \otimes V_2X_2^* \otimes V_3X_3^* - A\|_1 \\
&- \beta_1\|V_1X_1^* \otimes V_2X_2^* \otimes V_3X_3^* - A\|_1 \\
=\ &\frac{1}{\beta_2}\|V_1\widehat{X}_1 \otimes V_2\widehat{X}_2 \otimes V_3\widehat{X}_3 - A\|_1 - (\frac{1}{\beta_2} + \beta_1)\|V_1X_1^* \otimes V_2X_2^* \otimes V_3X_3^* - A\|_1. \qquad (21.35)
\end{aligned}$$

The first and the third inequality follow by the triangle inequalities. The second inequality follows using that

$$\begin{aligned}
&\|SV_1\widehat{X}_1 \otimes V_2\widehat{X}_2 \otimes V_3\widehat{X}_3 - SV_1X_1^* \otimes V_2X_2^* \otimes V_3X_3^*\|_1 \\
=\ &\left\|SV_1(\widehat{X}_1 - X_1^*) \cdot \left((V_2(\widehat{X}_2 - X_2^*))^\top \odot (V_3(\widehat{X}_3 - X_3^*))^\top\right)\right\|_1 \\
\geq\ &\frac{1}{\beta_2}\left\|V_1(\widehat{X}_1 - X_1^*) \cdot \left((V_2(\widehat{X}_2 - X_2^*))^\top \odot (V_3(\widehat{X}_3 - X_3^*))^\top\right)\right\|_1 \\
\geq\ &\frac{1}{\beta_2}\|V_1\widehat{X}_1 \otimes V_2\widehat{X}_2 \otimes V_3\widehat{X}_3 - V_1X_1^* \otimes V_2X_2^* \otimes V_3X_3^*\|_1,
\end{aligned}$$

and

$$\begin{aligned}
&\|SV_1X_1^* \otimes V_2X_2^* \otimes V_3X_3^* - SA\|_1 \\
&= \|S(V_1X_1^* \cdot ((V_2X_2^*)^\top \odot (V_3X_3^*)^\top) - A_1)\|_1 \\
&\leq \|V_1X_1^* \cdot ((V_2X_2^*)^\top \odot (V_3X_3^*)^\top) - A_1\|_1 \\
&= \beta_1\|V_1X_1^* \otimes V_2X_2^* \otimes V_3X_3^* - A\|_1. \qquad (21.36)
\end{aligned}$$

Then, we have

$$\begin{aligned}
&\|V_1\widehat{X}_1 \otimes V_2\widehat{X}_2 \otimes V_3\widehat{X}_3 - A\|_1 \\
&\leq \beta_2\|SV_1\widehat{X}_1 \otimes V_2\widehat{X}_2 \otimes V_3\widehat{X}_3 - SA\|_1 + (1+\beta_1\beta_2)\|V_1X_1^* \otimes V_2X_2^* \otimes V_3X_3^* - A\|_1 \\
&\leq \beta_2\beta\|SV_1X_1^* \otimes V_2X_2^* \otimes V_3X_3^* - SA\|_1 + (1+\beta_1\beta_2)\|V_1X_1^* \otimes V_2X_2^* \otimes V_3X_3^* - A\|_1 \\
&\leq \beta_1\beta_2\beta\|V_1X_1^* \otimes V_2X_2^* \otimes V_3X_3^* - A\|_1 + (1+\beta_1\beta_2)\|V_1X_1^* \otimes V_2X_2^* \otimes V_3X_3^* - A\|_1 \\
&\leq \beta(1+2\beta_1\beta_2)\|V_1X_1^* \otimes V_2X_2^* \otimes V_3X_3^* - A\|_1.
\end{aligned}$$

The first inequality follows by Equation (21.35). The second inequality follows by

$$\|SV_1\widehat{X}_1 \otimes V_2\widehat{X}_2 \otimes V_3\widehat{X}_3 - SA\|_1 \leq \beta \min_{X_1,X_2,X_3} \|SV_1X_1 \otimes V_2X_2 \otimes V_3X_3 - SA\|_1.$$

The third inequality follows by Equation (21.36). The final inequality follows using that $\beta \geq 1$. □

Lemma 21.7.5. *Let* $\min(b_1, b_2, b_3) \geq k$. *Given three matrices* $V_1 \in \mathbb{R}^{n\times b_1}$, $V_2 \in \mathbb{R}^{n\times b_2}$, *and* $V_3 \in \mathbb{R}^{n\times b_3}$, *there exists an algorithm that takes* $O(\mathrm{nnz}(A)) + n\,\mathrm{poly}(b_1, b_2, b_3)$ *time and outputs a tensor* $C \in \mathbb{R}^{c_1\times c_2\times c_3}$ *and three matrices* $\widehat{V}_1 \in \mathbb{R}^{c_1\times b_1}$, $\widehat{V}_2 \in \mathbb{R}^{c_2\times b_2}$ *and* $\widehat{V}_3 \in \mathbb{R}^{c_3\times b_3}$

Algorithm 21.21 Reducing the Size of the Objective Function to poly(k)

procedure L1PolyKSizeReduction($A, V_1, V_2, V_3, n, b_1, b_2, b_3, k$) ▷ Lemma 21.7.5
  for $i = 1 \to 3$ **do**
    $c_i \leftarrow \widetilde{O}(b_i)$.
    Choose sampling and rescaling matrices $T_i \in \mathbb{R}^{c_i \times n}$ according to the Lewis weights of V_i.
    $\widehat{V}_i \leftarrow T_i V_i \in \mathbb{R}^{c_i \times b_i}$.
  end for
  $C \leftarrow A(T_1, T_2, T_3) \in \mathbb{R}^{c_1 \times c_2 \times c_3}$.
  return $\widehat{V}_1$, $\widehat{V}_2$, $\widehat{V}_3$ and C.
end procedure

with $c_1 = c_2 = c_3 = \mathrm{poly}(b_1, b_2, b_3)$*, such that with probability* 0.99*, for any* $\alpha \geq 1$*, if* X_1', X_2', X_3' *satisfy that,*

$$\left\| \sum_{i=1}^{k} (\widehat{V}_1 X_1')_i \otimes (\widehat{V}_2 X_2')_i \otimes (\widehat{V}_3 X_3')_i - C \right\|_1 \leq \alpha \min_{X_1, X_2, X_3} \left\| \sum_{i=1}^{k} (\widehat{V}_1 X_1)_i \otimes (\widehat{V}_2 X_2)_i \otimes (\widehat{V}_3 X_3)_i - C \right\|_1,$$

then,

$$\left\| \sum_{i=1}^{k} (V_1 X_1')_i \otimes (V_2 X_2')_i \otimes (V_3 X_3')_i - A \right\|_1 \lesssim \alpha \min_{X_1, X_2, X_3} \left\| \sum_{i=1}^{k} (V_1 X_1)_i \otimes (V_2 X_2)_i \otimes (V_3 X_3)_i - A \right\|_1.$$

Proof. For simplicity, we define OPT to be

$$\min_{X_1, X_2, X_3} \left\| \sum_{i=1}^{k} (V_1 X_1)_i \otimes (V_2 X_2)_i \otimes (V_3 X_3)_i - A \right\|_1.$$

Let $T_1 \in \mathbb{R}^{c_1 \times n}$ sample according to the Lewis weights of $V_1 \in \mathbb{R}^{n \times b_1}$, where $c_1 = \widetilde{O}(b_1)$. Let $T_2 \in \mathbb{R}^{c_2 \times n}$ sample according to the Lewis weights of $V_2 \in \mathbb{R}^{n \times b_2}$, where $c_2 = \widetilde{O}(b_2)$. Let $T_3 \in \mathbb{R}^{c_3 \times n}$ sample according to the Lewis weights of $V_3 \in \mathbb{R}^{n \times b_3}$, where $c_3 = \widetilde{O}(b_3)$.

For any $\alpha \geq 1$, let $X_1' \in \mathbb{R}^{b_1 \times k}, X_2' \in \mathbb{R}^{b_2 \times k}, X_3' \in \mathbb{R}^{b_3 \times k}$ satisfy

$$\begin{aligned}&\|T_1 V_1 X_1' \otimes T_2 V_2 X_2' \otimes T_3 V_3 X_3' - A(T_1, T_2, T_3)\|_1 \\ \leq\ & \alpha \min_{X_1 \in \mathbb{R}^{b_1 \times k}, X_2 \in \mathbb{R}^{b_2 \times k}, X_3 \in \mathbb{R}^{b_3 \times k}} \|T_1 V_1 X_1 \otimes T_2 V_2 X_2 \otimes T_3 V_3 X_3 - A(T_1, T_2, T_3)\|_1.\end{aligned}$$

First, we regard T_1 as the sketching matrix for the remainder. Then by Lemma D.11 in [SWZ17] and Theorem 21.7.4, we have

$$\begin{aligned}&\|V_1 X_1' \otimes T_2 V_2 X_2' \otimes T_3 V_3 X_3' - A(I, T_2, T_3)\|_1 \\ \lesssim\ & \alpha \min_{X_1 \in \mathbb{R}^{b_1 \times k}, X_2 \in \mathbb{R}^{b_2 \times k}, X_3 \in \mathbb{R}^{b_3 \times k}} \|V_1 X_1 \otimes T_2 V_2 X_2 \otimes T_3 V_3 X_3 - A(I, T_2, T_3)\|_1.\end{aligned}$$

Second, we regard T_2 as a sketching matrix for $V_1 X_1 \otimes V_2 X_2 \otimes T_3 V_3 X_3 - A(I, I, T_3)$. Then by Lemma D.11 in [SWZ17] and Theorem 21.7.4, we have

$$\begin{aligned}&\|V_1 X_1' \otimes V_2 X_2' \otimes T_3 V_3 X_3' - A(I, I, T_3)\|_1 \\ \lesssim\ & \alpha \min_{X_1 \in \mathbb{R}^{b_1 \times k}, X_2 \in \mathbb{R}^{b_2 \times k}, X_3 \in \mathbb{R}^{b_3 \times k}} \|V_1 X_1 \otimes V_2 X_2 \otimes T_3 V_3 X_3 - A(I, I, T_3)\|_1.\end{aligned}$$

Third, we regard T_3 as a sketching matrix for $V_1 X_1 \otimes V_2 X_2 \otimes V_3 X_3 - A$. Then by Lemma D.11 in [SWZ17] and Theorem 21.7.4, we have

$$\left\| \sum_{i=1}^k (V_1 X_1')_i \otimes (V_2 X_2')_i \otimes (V_3 X_3')_i - A \right\|_1 \lesssim \alpha \min_{X_1, X_2, X_3} \left\| \sum_{i=1}^k (V_1 X_1)_i \otimes (V_2 X_2)_i \otimes (V_3 X_3)_i - A \right\|_1.$$

□

Lemma 21.7.6. *Given tensor $A \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, and two matrices $U \in \mathbb{R}^{n_1 \times s}, V \in \mathbb{R}^{n_2 \times s}$ with $\mathrm{rank}(U) = r$, let $T \in \mathbb{R}^{t \times n_1}$ be a sampling/rescaling matrix according to the Lewis weights of U with $t = \widetilde{O}(r)$. Then with probability at least 0.99, for all $X' \in \mathbb{R}^{n_3 \times s}, \alpha \geq 1$ which satisfy*

$$\|T_1 U \otimes V \otimes X' - T_1 A\|_1 \leq \alpha \cdot \min_{X \in \mathbb{R}^{n_3 \times s}} \|T_1 U \otimes V \otimes X - T_1 A\|_1,$$

it holds that

$$\|U \otimes V \otimes X' - A\|_1 \lesssim \alpha \cdot \min_{X \in \mathbb{R}^{n_3 \times s}} \|U \otimes V \otimes X - A\|_1.$$

The proof is similar to the proof of Lemma 21.7.5.

Proof. Let $X^* = \arg\min_{X \in \mathbb{R}^{n_3 \times s}} \|U \otimes V \otimes X - A\|_1$. Then according to Lemma D.11 in [SWZ17], T has at most constant dilation (Definition 21.7.2) on $U \cdot (V^\top \odot (X^*)^\top) - A_1$, and has at most constant contraction (Definition 21.7.3) on U. We first look at

$$\begin{aligned}
& \|TU \otimes V \otimes X' - TA\|_1 \\
= & \|TU \cdot (V^\top \odot (X')^\top) - TA_1\|_1 \\
\geq & \|TU \cdot ((V^\top \odot (X')^\top) - (V^\top \odot (X^*)^\top))\|_1 - \|TU \cdot (V^\top \odot (X^*)^\top) - TA_1\|_1 \\
\geq & \frac{1}{\beta_2} \|U \cdot ((V^\top \odot (X')^\top) - A_1\|_1 - (\frac{1}{\beta_2} + \beta_1) \|U \cdot (V^\top \odot (X^*)^\top) - A_1\|_1,
\end{aligned}$$

where $\beta_1 \geq 1, \beta_2 \geq 1$ are two constants. Then we have:

$$\begin{aligned}
& \|U \otimes V \otimes X' - A\|_1 \\
\leq & \beta_2 \|TU \cdot (V^\top \odot (X')^\top) - TA_1\|_1 + (1 + \beta_1 \beta_2) \|U \cdot (V^\top \odot (X^*)^\top) - A_1\|_1 \\
\leq & \alpha \beta_2 \|TU \cdot (V^\top \odot (X^*)^\top) - TA_1\|_1 + (1 + \beta_1 \beta_2) \|U \cdot (V^\top \odot (X^*)^\top) - A_1\|_1 \\
\leq & \alpha \beta_1 \beta_2 \|U \cdot (V^\top \odot (X^*)^\top) - A_1\|_1 + (1 + \beta_1 \beta_2) \|U \cdot (V^\top \odot (X^*)^\top) - A_1\|_1 \\
\lesssim & \alpha \|U \otimes V \otimes X^* - A\|_1.
\end{aligned}$$

□

Corollary 21.7.7. *Given tensor $A \in \mathbb{R}^{n \times n \times n}$, and two matrices $U \in \mathbb{R}^{n \times s}, V \in \mathbb{R}^{n \times s}$ with $\mathrm{rank}(U) = r_1, \mathrm{rank}(V) = r_2$, let $T_1 \in \mathbb{R}^{t_1 \times n}$ be a sampling/rescaling matrix according to*

the Lewis weights of U, and let $T_2 \in \mathbb{R}^{t_2 \times n}$ be a sampling/rescaling matrix according to the Lewis weights of V with $t_1 = \widetilde{O}(r_1), t_2 = \widetilde{O}(r_2)$. Then with probability at least 0.99, for all $X' \in \mathbb{R}^{n \times s}, \alpha \geq 1$ which satisfy

$$\|T_1 U \otimes T_2 V \otimes X' - A(T_1, T_2, I)\|_1 \leq \alpha \cdot \min_{X \in \mathbb{R}^{n \times s}} \|T_1 U \otimes T_2 V \otimes X - A(T_1, T_2, I)\|_1,$$

it holds that

$$\|U \otimes V \otimes X' - A\|_1 \lesssim \alpha \cdot \min_{X \in \mathbb{R}^{n \times s}} \|U \otimes V \otimes X - A\|_1.$$

Proof. We apply Lemma 21.7.6 twice: if

$$\|T_1 U \otimes T_2 V \otimes X' - A(T_1, T_2, I)\|_1 \leq \alpha \cdot \min_{X \in \mathbb{R}^{n \times s}} \|T_1 U \otimes T_2 V \otimes X - A(T_1, T_2, I)\|_1,$$

then

$$\|U \otimes T_2 V \otimes X' - A(I, T_2, I)\|_1 \lesssim \alpha \cdot \min_{X \in \mathbb{R}^{n \times s}} \|U \otimes T_2 V \otimes X - A(I, T_2, I)\|_1.$$

Then, we have

$$\|U \otimes V \otimes X' - A\|_1 \lesssim \alpha \cdot \min_{X \in \mathbb{R}^{n \times s}} \|U \otimes V \otimes X - A\|_1.$$

□

21.7.4 Solving small problems

Theorem 21.7.8. *Let* $\max_i\{t_i, d_i\} \leq n$. *Given a* $t_1 \times t_2 \times t_3$ *tensor* A *and three matrices: a* $t_1 \times d_1$ *matrix* T_1, *a* $t_2 \times d_2$ *matrix* T_2, *and a* $t_3 \times d_3$ *matrix* T_3, *if for* $\delta > 0$ *there exists a solution to*

$$\min_{X_1,X_2,X_3} \left\| \sum_{i=1}^{k} (T_1X_1)_i \otimes (T_2X_2)_i \otimes (T_3X_3)_i - A \right\|_1 := \mathrm{OPT},$$

such that each entry of X_i *can be expressed using* $O(n^\delta)$ *bits, then there exists an algorithm that takes* $n^{O(\delta)} \cdot 2^{O(d_1k+d_2k+d_3k)}$ *time and outputs three matrices:* $\widehat{X}_1$, $\widehat{X}_2$, *and* $\widehat{X}_3$ *such that* $\|(T_1\widehat{X}_1) \otimes (T_2\widehat{X}_2) \otimes (T_3\widehat{X}_3) - A\|_1 = \mathrm{OPT}$.

Proof. For each $i \in [3]$, we can create $t_i \times d_i$ variables to represent matrix X_i. Let x denote the list of these variables. Let B denote tensor $\sum_{i=1}^{k} (T_1X_1)_i \otimes (T_2X_2)_i \otimes (T_3X_3)_i$. Then we can write the following objective function,

$$\min_x \sum_{i=1}^{t_1} \sum_{j=1}^{t_2} \sum_{l=1}^{t_3} |B_{i,j,l}(x) - A_{i,j,l}|.$$

To remove the $|\cdot|$, we create $t_1t_2t_3$ extra variables $\sigma_{i,j,l}$. Then we obtain the objective function:

$$\begin{aligned} \min_{x,\sigma} \ & \sum_{i=1}^{t_1} \sum_{j=1}^{t_2} \sum_{l=1}^{t_3} \sigma_{i,j,l}(B_{i,j,l}(x) - A_{i,j,l}) \\ \text{s.t. } & \sigma_{i,j,l}^2 = 1, \\ & \sigma_{i,j,l}(B_{i,j,l}(x) - A_{i,j,l}) \geq 0, \\ & \|x\|_2^2 + \|\sigma\|_2^2 \leq 2^{O(n^\delta)} \end{aligned}$$

where the last constraint is unharmful, because there exists a solution that can be written using $O(n^\delta)$ bits. Note that the number of inequality constraints in the above system is

$O(t_1t_2t_3)$, the degree is $O(1)$, and the number of variables is $v = (t_1t_2t_3 + d_1k + d_2k + d_3k)$. Thus by Theorem 21.3.2, we know that the minimum nonzero cost is at least

$$(2^{O(n^\delta)})^{-2^{\widetilde{O}(v)}}.$$

It is immediate that the upper bound on cost is at most $2^{O(n^\delta)}$, and thus the number of binary search steps is at most $\log(2^{O(n^\delta)})2^{\widetilde{O}(v)}$. In each step of the binary search, we need to choose a cost C between the lower bound and the upper bound, and write down the polynomial system,

$$\begin{aligned}
&\sum_{i=1}^{t_1}\sum_{j=1}^{t_2}\sum_{l=1}^{t_3}\sigma_{i,j,l}(B_{i,j,l}(x) - A_{i,j,l}) \leq C,\\
&\sigma_{i,j,l}^2 = 1,\\
&\sigma_{i,j,l}(B_{i,j,l}(x) - A_{i,j,l}) \geq 0,\\
&\|x\|_2^2 + \|\sigma\|_2^2 \leq 2^{O(n^\delta)}.
\end{aligned}$$

Using Theorem 21.3.1, we can determine if there exists a solution to the above polynomial system. Since the number of variables is v, and the degree is $O(1)$, the number of inequality constraints is $t_1t_2t_2$. Thus, the running time is

$$\text{poly}(\text{bitsize}) \cdot (\#\,\text{constraints} \cdot \text{degree})^{\#\,\text{variables}} = n^{O(\delta)}2^{\widetilde{O}(v)}$$

□

21.7.5 Bicriteria algorithms

We present several bicriteria algorithms with different tradeoffs. We first present an algorithm that runs in nearly linear time and outputs a solution with rank $\widetilde{O}(k^3)$ in Theorem 21.7.9. Then we show an algorithm that runs in $\mathrm{nnz}(A)$ time but outputs a solution with rank $\mathrm{poly}(k)$ in Theorem 21.7.10. Then we explain an idea which is able to decrease the cubic rank to quadratic rank, and thus we can obtain Theorem 21.7.11 and Theorem 21.7.12.

21.7.5.1 Input sparsity time

Algorithm 21.22 ℓ_1-Low Rank Approximation, Bicriteria Algorithm, rank-$\widetilde{O}(k^3)$, Nearly Input Sparsity Time

procedure L1BicriteriaAlgorithm(A, n, k) ▷ Theorem 21.7.9
  $s_1 \leftarrow s_2 \leftarrow s_3 \leftarrow \widetilde{O}(k)$.
  For each $i \in [3]$, choose $S_i \in \mathbb{R}^{n^2 \times s_i}$ to be a dense Cauchy transform. ▷ Part (I) of Theorem 21.7.2
  Compute $A_1 \cdot S_1$, $A_2 \cdot S_2$, $A_3 \cdot S_3$.
  $Y_1, Y_2, Y_3, C \leftarrow$L1PolyKSizeReduction($A, A_1S_1, A_2S_2, A_3S_3, n, s_1, s_2, s_3, k$) ▷ Algorithm 21.21
  Form objective function

$$\min_{X \in \mathbb{R}^{s_1 \times s_2 \times s_3}} \left\| \sum_{i=1}^{s_1} \sum_{j=1}^{s_2} \sum_{l=1}^{s_3} X_{i,j,l} (Y_1)_i \otimes (Y_2)_j \otimes (Y_3)_l - C \right\|_1 .$$

  Run ℓ_1-regression solver to find X.
  return A_1S_1, A_2S_2, A_3S_3 and X.
end procedure

Theorem 21.7.9. *Given a 3rd order tensor $A \in \mathbb{R}^{n \times n \times n}$, for any $k \geq 1$, $\epsilon \in (0,1)$, let $r = \widetilde{O}(k^3)$. There exists an algorithm which takes $\mathrm{nnz}(A) \cdot \widetilde{O}(k) + O(n)\,\mathrm{poly}(k) + \mathrm{poly}(k)$*

time and outputs three matrices $U, V, W \in \mathbb{R}^{n\times r}$ *such that*

$$\left\|\sum_{i=1}^{r} U_i \otimes V_i \otimes W_i - A\right\|_1 \leq \widetilde{O}(k^{3/2}) \log^3 n \min_{\mathrm{rank}-k\ A_k} \|A_k - A\|_1$$

holds with probability 9/10.

Proof. We first choose three dense Cauchy transforms $S_i \in \mathbb{R}^{n^2\times s_i}$. According to Section 21.3.7, for each $i \in [3]$, $A_i S_i$ can be computed in $\mathrm{nnz}(A) \cdot \widetilde{O}(k)$ time. Then we apply Lemma 21.7.5 (Algorithm 21.21). We obtain three matrices Y_1, Y_2, Y_3 and a tensor C. Note that for each $i \in [3]$, Y_i can be computed in $n\,\mathrm{poly}(k)$ time. Because $C = A(T_1, T_2, T_3)$ and $T_1, T_2, T_3 \in \mathbb{R}^{n\times\widetilde{O}(k)}$ are three sampling and rescaling matrices, C can be computed in $\mathrm{nnz}(A) + \widetilde{O}(k^3)$ time. At the end, we just need to run an ℓ_1-regression solver to find the solution to the problem,

$$\min_{X\in\mathbb{R}^{s_1\times s_2\times s_3}} \left\|\sum_{i=1}^{s_1}\sum_{j=1}^{s_2}\sum_{l=1}^{s_3} X_{i,j,l}(Y_1)_i \otimes (Y_2)_j \otimes (Y_3)_j\right\|_1,$$

where $(Y_1)_i$ denotes the i-th column of matrix Y_1. Since the size of the above problem is only $\mathrm{poly}(k)$, this can be solved in $\mathrm{poly}(k)$ time. □

Theorem 21.7.10. *Given a* 3*rd order tensor* $A \in \mathbb{R}^{n\times n\times n}$*, for any* $k \geq 1$*,* $\epsilon \in (0, 1)$*, let* $r = \widetilde{O}(k^{15})$*. There exists an algorithm that takes* $\mathrm{nnz}(A) + O(n)\,\mathrm{poly}(k) + \mathrm{poly}(k)$ *time and outputs three matrices* $U, V, W \in \mathbb{R}^{n\times r}$ *such that*

$$\left\|\sum_{i=1}^{r} U_i \otimes V_i \otimes W_i - A\right\|_1 \leq \mathrm{poly}(k, \log n) \min_{\mathrm{rank}-k\ A_k} \|A_k - A\|_1$$

holds with probability 9/10.

Algorithm 21.23 ℓ_1-Low Rank Approximation, Bicriteria Algorithm, rank-poly(k), Input Sparsity Time

procedure L1BicriteriaAlgorithm(A, n, k) ▷ Theorem 21.7.10
  $s_1 \leftarrow s_2 \leftarrow s_3 \leftarrow \widetilde{O}(k^5)$.
  For each $i \in [3]$, choose $S_i \in \mathbb{R}^{n^2 \times s_i}$ to be a sparse Cauchy transform. ▷ Part (II) of Theorem 21.7.3
  Compute $A_1 \cdot S_1$, $A_2 \cdot S_2$, $A_3 \cdot S_3$.
  $Y_1, Y_2, Y_3, C \leftarrow$L1PolyKSizeReduction$(A, A_1S_1, A_2S_2, A_3, S_3, n, s_1, s_2, s_3, k)$ ▷ Algorithm 21.21
  Form objective function

$$\min_{X \in \mathbb{R}^{s_1 \times s_2 \times s_3}} \left\| \sum_{i=1}^{s_1} \sum_{j=1}^{s_2} \sum_{l=1}^{s_3} X_{i,j,l} (Y_1)_i \otimes (Y_2)_j \otimes (Y_3)_l - C \right\|_1 .$$

  Run ℓ_1-regression solver to find X.
  return A_1S_1, A_2S_2, A_3S_3 and X.
end procedure

Proof. We first choose three dense Cauchy transforms $S_i \in \mathbb{R}^{n^2 \times s_i}$. According to Section 21.3.7, for each $i \in [3]$, A_iS_i can be computed in $O(\mathrm{nnz}(A))$ time. Then we apply Lemma 21.7.5 (Algorithm 21.21), and can obtain three matrices Y_1, Y_2, Y_3 and a tensor C. Note that for each $i \in [3]$, Y_i can be computed in $O(n)\,\mathrm{poly}(k)$ time. Because $C = A(T_1, T_2, T_3)$ and $T_1, T_2, T_3 \in \mathbb{R}^{n \times \widetilde{O}(k)}$ are three sampling and rescaling matrices, C can be computed in $\mathrm{nnz}(A) + \widetilde{O}(k^3)$ time. At the end, we just need to run an ℓ_1-regression solver to find the solution to the problem,

$$\min_{X \in \mathbb{R}^{s_1 \times s_2 \times s_3}} \left\| \sum_{i=1}^{s_1} \sum_{j=1}^{s_2} \sum_{l=1}^{s_3} X_{i,j,l} (Y_1)_i \otimes (Y_2)_j \otimes (Y_3)_l - C \right\|_1 ,$$

where $(Y_1)_i$ denotes the i-th column of matrix Y_1. Since the size of the above problem is only $\mathrm{poly}(k)$, it can be solved in $\mathrm{poly}(k)$ time. □

21.7.5.2 Improving cubic rank to quadratic rank

Algorithm 21.24 ℓ_1-Low Rank Approximation, Bicriteria Algorithm, rank-$\widetilde{O}(k^2)$, Nearly Input Sparsity Time

procedure L1BicriteriaAlgorithm(A, n, k) ▷ Theorem 21.7.11
  $s_1 \leftarrow s_2 \leftarrow s_3 \leftarrow \widetilde{O}(k)$.
  For each $i \in [3]$, choose $S_i \in \mathbb{R}^{n^2 \times s_i}$ to be a dense Cauchy transform. ▷ Part (I) of Theorem 21.7.2
  Compute $A_1 \cdot S_1$, $A_2 \cdot S_2$.
  For each $i \in [2]$, choose T_i to be a sampling and rescaling diagonal matrix according to the Lewis weights of $A_i S_i$, with $t_i = \widetilde{O}(k)$ nonzero entries.
  $C \leftarrow A(T_1, T_2, I)$.
  $B^{i+(j-1)s_1} \leftarrow \mathrm{vec}((T_1 A_1 S_1)_i \otimes (T_2 A_2 S_2)_j), \forall i \in [s_1], j \in [s_2]$.
  Form objective function $\min_W \|WB - C_3\|_1$
  Run ℓ_1-regression solver to find $\widehat{W}$.
  Construct $\widehat{U}$ by using $A_1 S_1$ according to Equation (21.38).
  Construct $\widehat{V}$ by using $A_2 S_2$ according to Equation (21.39).
  return $\widehat{U}, \widehat{V}, \widehat{W}$.
end procedure

Theorem 21.7.11. *Given a 3rd order tensor $A \in \mathbb{R}^{n \times n \times n}$, for any $k \geq 1$, $\epsilon \in (0,1)$, let $r = \widetilde{O}(k^2)$. There exists an algorithm which takes $\mathrm{nnz}(A) \cdot \widetilde{O}(k) + O(n)\,\mathrm{poly}(k) + \mathrm{poly}(k)$ time and outputs three matrices $U, V, W \in \mathbb{R}^{n \times r}$ such that*

$$\left\| \sum_{i=1}^{r} U_i \otimes V_i \otimes W_i - A \right\|_1 \leq \widetilde{O}(k^{3/2}) \log^3 n \min_{\mathrm{rank}-k\ A_k} \|A_k - A\|_1$$

holds with probability 9/10.

Proof. Let $\mathrm{OPT} = \min_{A_k \in \mathbb{R}^{n \times n \times n}} \|A_k - A\|_1$. We first choose three dense Cauchy transforms $S_i \in \mathbb{R}^{n^2 \times s_i}$, $\forall i \in [3]$. According to Section 21.3.7, for each $i \in [3]$, $A_i S_i$ can be computed in $\mathrm{nnz}(A) \cdot \widetilde{O}(k)$ time. Then we choose T_i to be a sampling and rescaling diagonal matrix according to the Lewis weights of $A_i S_i$, $\forall i \in [2]$.

According to Theorem 21.7.3, we have

$$\min_{X_1\in\mathbb{R}^{s_1\times k},X_2\in\mathbb{R}^{s_2\times k},X_3\in\mathbb{R}^{s_3\times k}}\left\|\sum_{l=1}^{k}(A_1S_1X_1)_l\otimes(A_2S_2X_2)_l\otimes(A_3S_3X_3)_l-A\right\|_1\leq\widetilde{O}(k^{1.5})\log^3 n\,\mathrm{OPT}$$

Now we fix an l and we have:

$$\begin{aligned}&(A_1S_1X_1)_l\otimes(A_2S_2X_2)_l\otimes(A_3S_3X_3)_l\\=&\left(\sum_{i=1}^{s_1}(A_1S_1)_i(X_1)_{i,l}\right)\otimes\left(\sum_{j=1}^{s_2}(A_2S_2)_j(X_2)_{j,l}\right)\otimes(A_3S_3X_3)_l\\=&\sum_{i=1}^{s_1}\sum_{j=1}^{s_2}(A_1S_1)_i\otimes(A_2S_2)_j\otimes(A_3S_3X_3)_l(X_1)_{i,l}(X_2)_{j,l}\end{aligned}$$

Thus, we have

$$\min_{X_1,X_2,X_3}\left\|\sum_{i=1}^{s_1}\sum_{j=1}^{s_2}(A_1S_1)_i\otimes(A_2S_2)_j\otimes\left(\sum_{l=1}^{k}(A_3S_3X_3)_l(X_1)_{i,l}(X_2)_{j,l}\right)-A\right\|_1\leq\widetilde{O}(k^{1.5})\log^3 n\,\mathrm{OPT}\,. \tag{21.37}$$

We create matrix $\widehat{U}\in\mathbb{R}^{n\times s_1s_2}$ by copying matrix A_1S_1 s_2 times, i.e.,

$$\widehat{U}=\begin{bmatrix}A_1S_1 & A_1S_1 & \cdots & A_1S_1\end{bmatrix}. \tag{21.38}$$

We create matrix $\widehat{V}\in\mathbb{R}^{n\times s_1s_2}$ by copying the i-th column of A_2S_2 a total of s_1 times into the columns $(i-1)s_1,\cdots,is_1$ of $\widehat{V}$, for each $i\in[s_2]$, i.e.,

$$\widehat{V}=\begin{bmatrix}(A_2S_2)_1 & \cdots & (A_2S_2)_1 & (A_2S_2)_2 & \cdots & (A_2S_2)_2 & \cdots(A_2S_2)_{s_2} & \cdots & (A_2S_2)_{s_2}.\end{bmatrix} \tag{21.39}$$

According to Equation (21.37), we have:

$$\min_{W\in\mathbb{R}^{n\times s_1s_2}}\|\widehat{U}\otimes\widehat{V}\otimes W-A\|_1\leq\widetilde{O}(k^{1.5})\log^3 n\cdot\mathrm{OPT}\,.$$

Let

$$\widehat{W} = \arg\min_{W \in \mathbb{R}^{n \times s_1 s_2}} \|T_1 \widehat{U} \otimes T_2 \widehat{V} \otimes W - A(T_1, T_2, I)\|_1.$$

Due to Corollary 21.7.7, we have

$$\|\widehat{U} \otimes \widehat{V} \otimes \widehat{W} - A\|_1 \leq \widetilde{O}(k^{1.5}) \log^3 n \cdot \mathrm{OPT}\,.$$

Putting it all together, we have that $\widehat{U}, \widehat{V}, \widehat{W}$ gives a rank-$\widetilde{O}(k^2)$ bicriteria algorithm to the original problem. □

Algorithm 21.25 ℓ_1-Low Rank Approximation, Bicriteria Algorithm, rank-poly(k), Input Sparsity Time

procedure L1BicriteriaAlgorithm(A, n, k) ▷ Theorem 21.7.12
  $s_1 \leftarrow s_2 \leftarrow s_3 \leftarrow \widetilde{O}(k^5)$.
  For each $i \in [3]$, choose $S_i \in \mathbb{R}^{n^2 \times s_i}$ to be a sparse Cauchy transform. ▷ Part (II) of Theorem 21.7.2
  Compute $A_1 \cdot S_1$, $A_2 \cdot S_2$.
  For each $i \in [2]$, choose T_i to be a sampling and rescaling diagonal matrix according to the Lewis weights of $A_i S_i$, with $t_i = \widetilde{O}(k)$ nonzero entries.
  $C \leftarrow A(T_1, T_2, I)$.
  $B^{i+(j-1)s_1} \leftarrow \mathrm{vec}((T_1 A_1 S_1)_i \otimes (T_2 A_2 S_2)_j), \forall i \in [s_1], j \in [s_2]$.
  Form objective function $\min_W \|WB - C_3\|_1$.
  Run ℓ_1-regression solver to find $\widehat{W}$.
  Construct $\widehat{U}$ by using $A_1 S_1$ according to Equation (21.38).
  Construct $\widehat{V}$ by using $A_2 S_2$ according to Equation (21.39).
  return $\widehat{U}, \widehat{V}, \widehat{W}$.
end procedure

Theorem 21.7.12. *Given a 3rd order tensor $A \in \mathbb{R}^{n \times n \times n}$, for any $k \geq 1$, $\epsilon \in (0,1)$, let $r = \widetilde{O}(k^{10})$. There exists an algorithm which takes $\mathrm{nnz}(A) + O(n)\,\mathrm{poly}(k) + \mathrm{poly}(k)$ time*

and outputs three matrices $U, V, W \in \mathbb{R}^{n \times r}$ such that

$$\left\| \sum_{i=1}^{r} U_i \otimes V_i \otimes W_i - A \right\|_1 \leq \mathrm{poly}(k, \log n) \min_{\mathrm{rank}-k \ A_k} \|A_k - A\|_1$$

holds with probability $9/10$.

Proof. The proof is similar to the proof of Theorem 21.7.11. The only difference is that instead of choosing dense Cauchy matrices S_1, S_2, we choose sparse Cauchy matrices. □

Notice that if we firstly apply a sparse Cauchy transform, we can reduce the rank of the matrix to $\mathrm{poly}(k)$. Then we apply a dense Cauchy transform and can further reduce the dimension while only incurring another $\mathrm{poly}(k)$ factor in the approximation ratio. By combining a sparse Cauchy transform and a dense Cauchy transform, we can improve the running time from $\mathrm{nnz}(A) \cdot \widetilde{O}(k)$ to $\mathrm{nnz}(A)$.

Corollary 21.7.13. *Given a 3rd order tensor $A \in \mathbb{R}^{n \times n \times n}$, for any $k \geq 1$, $\epsilon \in (0, 1)$, let $r = \widetilde{O}(k^2)$. There exists an algorithm which takes $\mathrm{nnz}(A) + O(n)\,\mathrm{poly}(k) + \mathrm{poly}(k)$ time and outputs three matrices $U, V, W \in \mathbb{R}^{n \times r}$ such that*

$$\left\| \sum_{i=1}^{r} U_i \otimes V_i \otimes W_i - A \right\|_1 \leq \mathrm{poly}(k, \log n) \min_{\mathrm{rank}-k \ A_k} \|A_k - A\|_1$$

holds with probability $9/10$.

Algorithm 21.26 ℓ_1-Low Rank Approximation, Bicriteria Algorithm, rank-$\widetilde{O}(k^2)$, Input Sparsity Time

procedure L1BICRITERIAALGORITHM(A, n, k) ▷ Corollary 21.7.13
  $s_1 \leftarrow s_2 \leftarrow s_3 \leftarrow \widetilde{O}(k)$.
  For each $i \in [3]$, choose $S_i \in \mathbb{R}^{n^2 \times s_i}$ to be the composition of a sparse Cauchy transform and a dense Cauchy transform. ▷ Part (I,II) of Theorem 21.7.2
  Compute $A_1 \cdot S_1$, $A_2 \cdot S_2$.
  For each $i \in [2]$, choose T_i to be a sampling and rescaling diagonal matrix according to the Lewis weights of $A_i S_i$, with $t_i = \widetilde{O}(k)$ nonzero entries.
  $C \leftarrow A(T_1, T_2, I)$.
  $B^{i+(j-1)s_1} \leftarrow \mathrm{vec}((T_1 A_1 S_1)_i \otimes (T_2 A_2 S_2)_j), \forall i \in [s_1], j \in [s_2]$.
  Form objective function $\min_W \|WB - C_3\|_1$.
  Run ℓ_1-regression solver to find $\widehat{W}$.
  Construct $\widehat{U}$ by using $A_1 S_1$ according to Equation (21.38).
  Construct $\widehat{V}$ by using $A_2 S_2$ according to Equation (21.39).
  return $\widehat{U}, \widehat{V}, \widehat{W}$.
end procedure

21.7.6 Algorithms

In this section, we show two different algorithms by using different kind of sketches. One is shown in Theorem 21.7.14 which gives a fast running time. Another one is shown in Theorem 21.7.16 which gives the best approximation ratio.

21.7.6.1 Input sparsity time algorithm

Theorem 21.7.14. *Given a 3rd tensor $A \in \mathbb{R}^{n \times n \times n}$, for any $k \geq 1$, there exists an algorithm that takes $\mathrm{nnz}(A) \cdot \widetilde{O}(k) + O(n)\,\mathrm{poly}(k) + 2^{\widetilde{O}(k^2)}$ time and outputs three matrices $U, V, W \in \mathbb{R}^{n \times k}$ such that,*

$$\|U \otimes V \otimes W - A\|_1 \leq \mathrm{poly}(k, \log n) \min_{\mathrm{rank}-k\ A'} \|A' - A\|_1.$$

Algorithm 21.27 ℓ_1-Low Rank Approximation, Input sparsity Time Algorithm

procedure L1TensorLowRankApproxInputSparsity(A, n, k) ▷ Theorem 21.7.14
  $s_1 \leftarrow s_2 \leftarrow s_3 \leftarrow \widetilde{O}(k^5)$.
  Choose $S_i \in \mathbb{R}^{n^2 \times s_i}$ to be a dense Cauchy transform, $\forall i \in [3]$. ▷ Part (I) of Theorem 21.7.3
  Compute $A_1 \cdot S_1$, $A_2 \cdot S_2$, and $A_3 \cdot S_3$.
  $Y_1, Y_2, Y_3, C \leftarrow$L1PolyKSizeReduction$(A, A_1S_1, A_2S_2, A_3S_3, n, s_1, s_2, s_3, k)$. ▷ Algorithm 21.21
  Create variables $s_1 \times k + s_2 \times k + s_3 \times k$ variables for each entry of X_1, X_2, X_3.
  Form objective function $\|(Y_1X_1) \otimes (Y_2X_2) \otimes (Y_3X_3) - C\|_F^2$.
  Run polynomial system verifier.
  return $A_1S_1X_1, A_2S_2X_2, A_3S_3X_3$.
end procedure

holds with probability at least $9/10$.

Proof. First, we apply part (II) of Theorem 21.7.3. Then A_iS_i can be computed in $O(\mathrm{nnz}(A))$ time. Second, we use Lemma 21.7.5 to reduce the size of the objective function from $O(n^3)$ to $\mathrm{poly}(k)$ in $n\,\mathrm{poly}(k)$ time by only losing a constant factor in approximation ratio. Third, we use Claim 21.3.6 to relax the objective function from entry-wise ℓ_1-norm to Frobenius norm, and this step causes us to lose some other $\mathrm{poly}(k)$ factors in approximation ratio. As a last step, we use Theorem 21.4.44 to solve the Frobenius norm objective function. □

Notice again that if we first apply a sparse Cauchy transform, we can reduce the rank of the matrix to $\mathrm{poly}(k)$. Then as before we can apply a dense Cauchy transform to further reduce the dimension while only incurring another $\mathrm{poly}(k)$ factor in the approximation ratio. By combining a sparse Cauchy transform and a dense Cauchy transform, we can improve the running time from $\mathrm{nnz}(A) \cdot \widetilde{O}(k)$ to $\mathrm{nnz}(A)$, while losing some additional $\mathrm{poly}(k)$ factors in approximation ratio.

Corollary 21.7.15. *Given a 3rd tensor $A \in \mathbb{R}^{n \times n \times n}$, for any $k \geq 1$, there exists an algorithm that takes $\mathrm{nnz}(A) + O(n)\,\mathrm{poly}(k) + 2^{\widetilde{O}(k^2)}$ time and outputs three matrices $U, V, W \in \mathbb{R}^{n \times k}$ such that,*

$$\|U \otimes V \otimes W - A\|_1 \leq \mathrm{poly}(k, \log n) \min_{\mathrm{rank}-k\ A'} \|A' - A\|_1.$$

holds with probability at least 9/10.

21.7.6.2 $\widetilde{O}(k^{3/2})$-approximation algorithm

Algorithm 21.28 ℓ_1-Low Rank Approximation Algorithm, $\widetilde{O}(k^{3/2})$-approximation

procedure L1TensorLowRankApproxK(A, n, k) ▷ Theorem 21.7.16
  $s_1 \leftarrow s_2 \leftarrow s_3 \leftarrow \widetilde{O}(k)$.
  Guess diagonal matrices $S_i \in \mathbb{R}^{n^2 \times s_i}$ with s_i nonzero entries, $\forall i \in [3]$. ▷ Part (III) of Theorem 21.7.3
  Compute $A_1 \cdot S_1$, $A_2 \cdot S_2$, and $A_3 \cdot S_3$.
  $Y_1, Y_2, Y_3, C \leftarrow$L1PolyKSizeReduction($A, A_1S_1, A_2S_2, A_3S_3, n, s_1, s_2, s_3, k$). ▷ Algorithm 21.21
  Create $s_1 \times k + s_2 \times k + s_3 \times k$ variables for each entry of X_1, X_2, X_3.
  Form objective function $\|(Y_1X_1) \otimes (Y_2X_2) \otimes (Y_3X_3) - C\|_1$.
  Run polynomial system verifier.
  return U, V, W.
end procedure

Theorem 21.7.16. *Given a 3rd order tensor $A \in \mathbb{R}^{n \times n \times n}$, for any $k \geq 1$, there exists an algorithm that takes $n^{\widetilde{O}(k)} 2^{\widetilde{O}(k^3)}$ time and output three matrices $U, V, W \in \mathbb{R}^{n \times k}$ such that,*

$$\|U \otimes V \otimes W - A\|_1 \leq \widetilde{O}(k^{3/2}) \min_{\mathrm{rank}-k\ A'} \|A' - A\|_1.$$

holds with probability at least 9/10.

Proof. First, we apply part (III) of Theorem 21.7.3. Then, guessing S_i requires $n^{\widetilde{O}(k)}$ time. Second, we use Lemma 21.7.5 to reduce the size of the objective from $O(n^3)$ to poly(k) in polynomial time while only losing a constant factor in approximation ratio. Third, we use Theorem 21.7.8 to solve the entry-wise ℓ_1-norm objective function directly. □

21.7.7 CURT decomposition

Algorithm 21.29 ℓ_1-CURT Decomposition Algorithm

procedure L1CURT(A, U_B, V_B, W_B, n, k) ▷ Theorem 21.7.17
  Form $B_1 = V_B^\top \odot W_B^\top \in \mathbb{R}^{k \times n^2}$.
  Let $D_1^\top \in \mathbb{R}^{n^2 \times n^2}$ be the sampling and rescaling diagonal matrix corresponding to the Lewis weights of $B_1^\top$, and let D_1 have $d_1 = O(k \log k)$ nonzero entries.
  Form $\widehat{U} = A_1 D_1 (B_1 D_1)^\dagger \in \mathbb{R}^{n \times k}$.
  Form $B_2 = \widehat{U}^\top \odot W_B^\top \in \mathbb{R}^{k \times n^2}$.
  Let $D_2^\top \in \mathbb{R}^{n^2 \times n^2}$ be the sampling and rescaling diagonal matrix corresponding to the Lewis weights of $B_2^\top$, and let D_2 have $d_2 = O(k \log k)$ nonzero entries.
  Form $\widehat{V} = A_2 D_2 (B_2 D_2)^\dagger \in \mathbb{R}^{n \times k}$.
  Form $B_3 = \widehat{U}^\top \odot \widehat{V}^\top \in \mathbb{R}^{k \times n^2}$.
  Let $D_3^\top \in \mathbb{R}^{n^2 \times n^2}$ be the sampling and rescaling diagonal matrix corresponding to the Lewis weights of $B_3^\top$, and let D_3 have $d_3 = O(k \log k)$ nonzero entries.
  $C \leftarrow A_1 D_1$, $R \leftarrow A_2 D_2$, $T \leftarrow A_3 D_3$.
  $U \leftarrow \sum_{i=1}^k ((B_1 D_1)^\dagger)_i \otimes ((B_2 D_2)^\dagger)_i \otimes ((B_3 D_3)^\dagger)_i$.
  return C, R, T and U.
end procedure

Theorem 21.7.17. *Given a 3rd order tensor $A \in \mathbb{R}^{n \times n \times n}$, let $k \geq 1$, let $U_B, V_B, W_B \in \mathbb{R}^{n \times k}$ denote a rank-k, α-approximation to A. Then there exists an algorithm which takes $O(\mathrm{nnz}(A)) + O(n^2)\,\mathrm{poly}(k)$ time and outputs three matrices: $C \in \mathbb{R}^{n \times c}$ with columns from A, $R \in \mathbb{R}^{n \times r}$ with rows from A, $T \in \mathbb{R}^{n \times t}$ with tubes from A, and a tensor $U \in \mathbb{R}^{c \times r \times t}$ with $\mathrm{rank}(U) = k$ such that $c = r = t = O(k \log k)$, and*

$$\left\| \sum_{i=1}^{c} \sum_{j=1}^{r} \sum_{l=1}^{t} U_{i,j,l} \cdot C_i \otimes R_j \otimes T_l - A \right\|_1 \leq \widetilde{O}(k^{1.5}) \alpha \min_{\mathrm{rank}-k\ A'} \|A' - A\|_1$$

holds with probability $9/10$.

Proof. We define

$$\mathrm{OPT} := \min_{\mathrm{rank}-k\ A'} \|A' - A\|_1.$$

We already have three matrices $U_B \in \mathbb{R}^{n\times k}$, $V_B \in \mathbb{R}^{n\times k}$ and $W_B \in \mathbb{R}^{n\times k}$ and these three matrices provide a rank-k, α approximation to A, i.e.,

$$\left\| \sum_{i=1}^{k} (U_B)_i \otimes (V_B)_i \otimes (W_B)_i - A \right\|_1 \leq \alpha\, \mathrm{OPT} \tag{21.40}$$

Let $B_1 = V_B^\top \odot W_B^\top \in \mathbb{R}^{k\times n^2}$ denote the matrix where the i-th row is the vectorization of $(V_B)_i \otimes (W_B)_i$. By Section B.3, we can compute $D_1 \in \mathbb{R}^{n^2\times n^2}$ which is a sampling and rescaling matrix corresponding to the Lewis weights of $B_1^\top$ in $O(n^2\,\mathrm{poly}(k))$ time, and there are $d_1 = O(k\log k)$ nonzero entries on the diagonal of D_1. Let $A_i \in \mathbb{R}^{n\times n^2}$ denote the matrix obtained by flattening A along the i-th direction, for each $i \in [3]$.

Define $U^* \in \mathbb{R}^{n\times k}$ to be the optimal solution to $\min_{U\in\mathbb{R}^{n\times k}} \|UB_1 - A_1\|_1$, $\widehat{U} = A_1 D_1 (B_1 D_1)^\dagger \in \mathbb{R}^{n\times k}$, $V_0 \in \mathbb{R}^{n\times k}$ to be the optimal solution to $\min_{V\in\mathbb{R}^{n\times k}} \|V \cdot (\widehat{U}^\top \odot W_B^\top) - A_2\|_1$, and U' to be the optimal solution to $\min_{U\in\mathbb{R}^{n\times k}} \|UB_1D_1 - A_1D_1\|_1$.

By Claim 21.3.4, we have

$$\|\widehat{U}B_1D_1 - A_1D_1\|_1 \leq \sqrt{d_1}\|U'B_1D_1 - A_1D_1\|_1$$

Due to Lemma D.11 and Lemma D.8 (in [SWZ17]) with constant probability, we have

$$\|\widehat{U}B_1 - A_1\|_1 \leq \sqrt{d_1}\alpha_{D_1}\|U^*B_1 - A_1\|_1, \tag{21.41}$$

where $\alpha_{D_1} = O(1)$.

Recall that $(\widehat{U}^\top \odot W_B^\top) \in \mathbb{R}^{k \times n^2}$ denotes the matrix where the i-th row is the vectorization of $\widehat{U}_i \otimes (W_B)_i$, $\forall i \in [k]$. Now, we can show,

$$\begin{aligned}
\|V_0 \cdot (\widehat{U}^\top \odot W_B^\top) - A_2\|_1 \leq \|\widehat{U} B_1 - A_1\|_1 & \quad \text{by } V_0 = \arg\min_{V \in \mathbb{R}^{n \times k}} \|V \cdot (\widehat{U}^\top \odot W_B^\top) - A_2\|_1 \\
\lesssim \sqrt{d_1} \|U^* B_1 - A_1\|_1 & \quad \text{by Equation (21.41)} \\
\leq \sqrt{d_1} \|U_B B_1 - A_1\|_1 & \quad \text{by } U^* = \arg\min_{U \in \mathbb{R}^{n \times k}} \|U B_1 - A_1\|_1 \\
\leq O(\sqrt{d_1}) \alpha \, \mathrm{OPT} & \quad \text{by Equation (21.40)}
\end{aligned} \tag{21.42}$$

We define $B_2 = \widehat{U}^\top \odot W_B^\top$. We can compute $D_2 \in \mathbb{R}^{n^2 \times n^2}$ which is a sampling and rescaling matrix corresponding to the Lewis weights of $B_2^\top$ in $O(n^2 \, \mathrm{poly}(k))$ time, and there are $d_2 = O(k \log k)$ nonzero entries on the diagonal of D_2.

Define $V^* \in \mathbb{R}^{n \times k}$ to be the optimal solution of $\min_{V \in \mathbb{R}^{n \times k}} \|V B_2 - A_2\|_1$, $\widehat{V} = A_2 D_2 (B_2 D_2)^\dagger \in \mathbb{R}^{n \times k}$, $W_0 \in \mathbb{R}^{n \times k}$ to be the optimal solution of $\min_{W \in \mathbb{R}^{n \times k}} \|W \cdot (\widehat{U}^\top \odot \widehat{V}^\top) - A_3\|_1$, and V' to be the optimal solution of $\min_{V \in \mathbb{R}^{n \times k}} \|V B_2 D_2 - A_2 D_2\|_1$.

By Claim 21.3.4, we have

$$\|\widehat{V} B_2 D_2 - A_2 D_2\|_1 \leq \sqrt{d_2} \|V' B_2 D_2 - A_2 D_2\|_1.$$

Due to Lemma D.11 and Lemma D.8(in [SWZ17]) with constant probability, we have

$$\|\widehat{V} B_2 - A_2\|_1 \leq \sqrt{d_2} \alpha_{D_2} \|V^* B_2 - A_2\|_1, \tag{21.43}$$

where $\alpha_{D_2} = O(1)$.

Recall that $(\widehat{U}^\top \odot \widehat{V}^\top) \in \mathbb{R}^{k \times n^2}$ denotes the matrix for which the i-th row is the

vectorization of $\widehat{U}_i \otimes \widehat{V}_i$, $\forall i \in [k]$. Now, we can show,

$$\begin{aligned}
\|W_0 \cdot (\widehat{U}^\top \odot \widehat{V}^\top) - A_3\|_1 &\leq \|\widehat{V} B_2 - A_2\|_1 && \text{by } W_0 = \arg\min_{W \in \mathbb{R}^{n \times k}} \|W \cdot (\widehat{U}^\top \odot \widehat{V}^\top) - A_3\|_1 \\
&\lesssim \sqrt{d_2} \|V^* B_2 - A_2\|_1 && \text{by Equation (21.43)} \\
&\leq \sqrt{d_2} \|V_0 B_2 - A_2\|_1 && \text{by } V^* = \arg\min_{V \in \mathbb{R}^{n \times k}} \|V B_2 - A_2\|_1 \\
&\leq O(\sqrt{d_1 d_2}) \alpha \, \mathrm{OPT} && \text{by Equation (21.42)}
\end{aligned} \tag{21.44}$$

We define $B_3 = \widehat{U}^\top \odot \widehat{V}^\top$. We can compute $D_3 \in \mathbb{R}^{n^2 \times n^2}$ which is a sampling and rescaling matrix corresponding to the Lewis weights of $B_3^\top$ in $O(n^2 \, \mathrm{poly}(k))$ time, and there are $d_3 = O(k \log k)$ nonzero entries on the diagonal of D_3.

Define $W^* \in \mathbb{R}^{n \times k}$ to be the optimal solution to $\min_{W \in \mathbb{R}^{n \times k}} \|W B_3 - A_3\|_1$, $\widehat{W} = A_3 D_3 (B_3 D_3)^\dagger \in \mathbb{R}^{n \times k}$, and W' to be the optimal solution to $\min_{W \in \mathbb{R}^{n \times k}} \|W B_3 D_3 - A_3 D_3\|_1$.

By Claim 21.3.4, we have

$$\|\widehat{W} B_3 D_3 - A_3 D_3\|_1 \leq \sqrt{d_3} \|W' B_3 D_3 - A_3 D_3\|_1.$$

Due to Lemma D.11 and Lemma D.8(in [SWZ17]) with constant probability, we have

$$\|\widehat{W} B_3 - A_3\|_1 \leq \sqrt{d_3} \alpha_{D_3} \|W^* B_3 - A_3\|_1, \tag{21.45}$$

where $\alpha_{D_3} = O(1)$. Now we can show,

$$\begin{aligned}
\|\widehat{W} B_3 - A_3\|_1 &\lesssim \sqrt{d_3} \|W^* B_3 - A_3\|_1, && \text{by Equation (21.45)} \\
&\leq \sqrt{d_3} \|W_0 B_3 - A_3\|_1, && \text{by } W^* = \arg\min_{W \in \mathbb{R}^{n \times k}} \|W B_3 - A_3\|_1 \\
&\leq O(\sqrt{d_1 d_2 d_3}) \alpha \, \mathrm{OPT} && \text{by Equation (21.44)}
\end{aligned}$$

Thus, it implies,

$$\left\| \sum_{i=1}^{k} \widehat{U}_i \otimes \widehat{V}_i \otimes \widehat{W}_i - A \right\|_1 \leq \mathrm{poly}(k, \log n)\, \mathrm{OPT}\,.$$

where $\widehat{U} = A_1 D_1 (B_1 D_1)^\dagger$, $\widehat{V} = A_2 D_2 (B_2 D_2)^\dagger$, $\widehat{W} = A_3 D_3 (B_3 D_3)^\dagger$.

□

Algorithm 21.30 ℓ_1-CURT decomposition algorithm

1: **procedure** L1CURT$^+(A, n, k)$ ▷ Theorem 21.7.18
2: $U_B, V_B, W_B \leftarrow$L1LowRankApproximation(A, n, k). ▷ Corollary 21.7.15
3: $C, R, T, U \leftarrow$ L1CURT(A, U_B, V_B, W_B, n, k). ▷ Algorithm 21.29
4: **return** C, R, T and U.
5: **end procedure**

Theorem 21.7.18. *Given a 3rd order tensor $A \in \mathbb{R}^{n \times n \times n}$, for any $k \geq 1$, there exists an algorithm which takes $O(\mathrm{nnz}(A)) + O(n^2)\,\mathrm{poly}(k) + 2^{\widetilde{O}(k^2)}$ time and outputs three matrices $C \in \mathbb{R}^{n \times c}$ with columns from A, $R \in \mathbb{R}^{n \times r}$ with rows from A, $T \in \mathbb{R}^{n \times t}$ with tubes from A, and a tensor $U \in \mathbb{R}^{c \times r \times t}$ with $\mathrm{rank}(U) = k$ such that $c = r = t = O(k \log k)$, and*

$$\left\| \sum_{i=1}^{c} \sum_{j=1}^{r} \sum_{l=1}^{t} U_{i,j,l} \cdot C_i \otimes R_j \otimes T_l - A \right\|_1 \leq \mathrm{poly}(k, \log n) \min_{\mathrm{rank}-k\ A'} \|A' - A\|_1,$$

holds with probability $9/10$.

Proof. This follows by combining Corollary 21.7.15 and Theorem 21.7.17. □

21.8 Weighted Frobenius Norm for Arbitrary Tensors

This section presents several tensor algorithms for the weighted case. For notational purposes, instead of using U, V, W to denote the ground truth factorization, we use U_1, U_2, U_3 to denote the ground truth factorization. We use A to denote the input tensor, and W to denote the tensor of weights. Combining our new tensor techniques with existing weighted low rank approximation algorithms [RSW16] allows us to obtain several interesting new results. We provide some necessary definitions and facts in Section 21.8.1. Section 21.8.2 provides an algorithm when W has at most r distinct faces in each dimension. Section 21.8.3 studies relationships between r distinct faces and r distinct columns. Finally, we provides an algorithm with a similar running time but weaker assumption, where W has at most r distinct columns and r distinct rows in Section 21.8.4. The result in Theorem 21.8.2 is fairly similar to Theorem 21.8.5, except for the running time. We only put a very detailed discussion in the statement of Theorem 21.8.5. Note that Theorem 21.8.2 also has other versions which are similar to the Frobnius norm rank-k algorithms described in Section 21.1. For simplicity of presentation, we only present one clean and simple version (which assumes A_k exists and has factor norms which are not too large).

21.8.1 Definitions and Facts

For a matrix $A \in \mathbb{R}^{n\times m}$ and a weight matrix $W \in \mathbb{R}^{n\times m}$, we define $\|W \circ A\|_F$ as follows,

$$\|W \circ A\|_F = \left(\sum_{i=1}^{n} \sum_{j=1}^{m} W_{i,j}^2 A_{i,j}^2 \right)^{\frac{1}{2}}.$$

For a tensor $A \in \mathbb{R}^{n\times n\times n}$ and a weight tensor $W \in \mathbb{R}^{n\times n\times n}$, we define $\|W \circ A\|_F$ as follows,

$$\|W \circ A\|_F = \left(\sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{l=1}^{n} W_{i,j,l}^2 A_{i,j,l}^2 \right)^{\frac{1}{2}}.$$

For three matrices $A \in \mathbb{R}^{n\times m}$, $U \in \mathbb{R}^{n\times k}$, $V \in \mathbb{R}^{k\times m}$ and a weight matrix W, from one perspective, we have

$$\|(UV - A) \circ W\|_F^2 = \sum_{i=1}^{n} \|(U^i V - A^i) \circ W^i\|_2^2 = \sum_{i=1}^{n} \|(U^i V - A^i) D_{W^i}\|_2^2,$$

where W^i denote the i-th row of matrix W, and $D_{W^i} \in \mathbb{R}^{m\times m}$ denotes a diagonal matrix where the j-th entry on diagonal is the j-th entry of vector W^i. From another perspective, we have

$$\|(UV - A) \circ W\|_F^2 = \sum_{j=1}^{m} \|(UV_j - A_j) \circ W_j\|_2^2 = \sum_{j=1}^{m} \|(UV_j - A_j) D_{W_j}\|_2^2,$$

where W_j denotes the j-th column of matrix W, and $D_{W_j} \in \mathbb{R}^{n\times n}$ denotes a diagonal matrix where the i-th entry on the diagonal is the i-th entry of vector W_j.

One of the key tools we use in this section is,

Lemma 21.8.1 (Cramer's rule)**.** *Let R be an $n \times n$ invertible matrix. Then, for each $i \in [n], j \in [n]$,*

$$(R^{-1})_i^j = \det(R_{\neg j}^{\neg i}) / \det(R),$$

where $R^{\neg i}_{\neg j}$ is the matrix R with the i-th row and the j-th column removed.

21.8.2 r distinct faces in each dimension

Notice that in the matrix case, it is sufficient to assume that $\|A'\|_F$ is upper bounded [RSW16]. Once we have that $\|A'\|_F$ is bounded, without loss of generality, we can assume that U_1^* is an orthonormal basis[CW15a, RSW16]. If U_1^* is not an orthonormal basis, then let $U_1'R$ denote a QR factorization of U_1^*, and then write $U_2' = RU_2^*$. However, in the case of tensors we have to assume that each factor $\|U_i^*\|_F$ is upper bounded due to border rank issues (see, e.g., [DSL08]).

Algorithm 21.31 Weighted Tensor Low-rank Approximation Algorithm when the Weighted Tensor has r Distinct Faces in Each of the Three Dimensions.

procedure WEIGHTEDRDISTINCTFACESIN3DIMENSIONS(A, W, n, r, k, ϵ) ▷ Theorem 21.8.2
  for $j = 1 \to 3$ **do**
    $s_j \leftarrow O(k/\epsilon)$.
    Choose a sketching matrix $S_j \in \mathbb{R}^{n^2 \times s_j}$.
    for $i = 1 \to r$ **do**
      Create $k \times s_1$ variables for matrix $P_{i,j} \in \mathbb{R}^{k \times s_j}$.
    end for
    for $i = 1 \to n$ **do**
      Write down $(\widehat{U}_j)^i = A_i^j D_{W_1^j} S_j P_{j,i}^\top (P_{j,i} P_{j,i}^\top)^{-1}$.
    end for
  end for
  Form $\|W \circ (\widehat{U}_1 \otimes \widehat{U}_2 \otimes \widehat{U}_3 - A)\|_F^2$.
  Run polynomial system verifier.
  return U_1, U_2, U_3
end procedure

Theorem 21.8.2. *Given a 3rd order $n \times n \times n$ tensor A and an $n \times n \times n$ tensor W of weights with r distinct faces in each of the three dimensions for which each entry can be written using $O(n^\delta)$ bits, for $\delta > 0$, define* $\mathrm{OPT} = \inf_{\mathrm{rank}-k\ A_k} \|W \circ (A_k - A)\|_F^2$. *Let $k \geq 1$ be an integer and let $0 < \epsilon < 1$.*

If $\mathrm{OPT} > 0$*, and there exists a rank-*k $A_k = U_1^* \otimes U_2^* \otimes U_3^*$ *tensor (with size* $n \times n \times n$*) such that* $\|W \circ (A_k - A)\|_F^2 = \mathrm{OPT}$*, and* $\max_{i \in [3]} \|U_i^*\|_F \leq 2^{O(n^\delta)}$*, then there exists an algorithm that takes* $(\mathrm{nnz}(A) + \mathrm{nnz}(W) + n2^{\widetilde{O}(rk^2/\epsilon)})n^{O(\delta)}$ *time in the unit cost* RAM *model with words of size* $O(\log n)$ *bits*[10] *and outputs three* $n \times k$ *matrices* U_1, U_2, U_3 *such that*

$$\|W \circ (U_1 \otimes U_2 \otimes U_3 - A)\|_F^2 \leq (1 + \epsilon)\,\mathrm{OPT} \tag{21.46}$$

holds with probability $9/10$*.*

Proof. Note that W has r distinct columns, rows, and tubes. Hence, each of the matrices $W_1, W_2, W_3 \in \mathbb{R}^{n \times n^2}$ has at most r distinct columns, and at most r distinct rows. Let $U_1^*, U_2^*, U_3^* \in \mathbb{R}^{n \times k}$ denote the matrices satisfying $\|W \circ (U_1^* \otimes U_2^* \otimes U_3^* - A)\|_F^2 = \mathrm{OPT}$. We fix U_2^* and U_3^*, and consider a flattening of the tensor along the first dimension,

$$\min_{U_1 \in \mathbb{R}^{n \times k}} \|(U_1 Z_1 - A_1) \circ W_1\|_F^2 = \mathrm{OPT},$$

where matrix $Z_1 = U_2^{*\top} \odot U_3^{*\top}$ has size $k \times n^2$ and for each $i \in [k]$ the i-th row of Z_1 is $\mathrm{vec}((U_2^*)_i \otimes (U_3^*)_i)$. For each $i \in [n]$, let W_1^i denote the i-th row of $n \times n^2$ matrix W_1. For each $i \in [n]$, let $D_{W_1^i}$ denote the diagonal matrix of size $n^2 \times n^2$, where each diagonal entry is from the vector $W_1^i \in \mathbb{R}^{n^2}$. Without loss of generality, we can assume the first r rows of W_1 are distinct. We can rewrite the objective function along the first dimension as a sum of multiple regression problems. For any $n \times k$ matrix U_1,

$$\|(U_1 Z_1 - A_1) \circ W_1\|_F^2 = \sum_{i=1}^{n} \|U_1^i Z_1 D_{W_1^i} - A_1^i D_{W_1^i}\|_2^2. \tag{21.47}$$

[10] The entries of A and W are assumed to fit in n^δ words.

Based on the observation that W_1 has r distinct rows, we can group the n rows of W^1 into r groups. We use $g_{1,1}, g_{1,2}, \cdots, g_{1,r}$ to denote r sets of indices such that, for each $i \in g_{1,j}$, $W_1^i = W_1^j$. Thus we can rewrite Equation (21.47),

$$\begin{aligned}\|(U_1 Z_1 - A_1) \circ W_1\|_F^2 &= \sum_{i=1}^{n} \|U_1^i Z_1 D_{W_1^i} - A_1^i D_{W_1^i}\|_2^2 \\ &= \sum_{j=1}^{r} \sum_{i \in g_{1,j}} \|U_1^i Z_1 D_{W_1^i} - A_1^i D_{W_1^i}\|_2^2.\end{aligned}$$

We can sketch the objective function by choosing Gaussian matrices $S_1 \in \mathbb{R}^{n^2 \times s_1}$ with $s_1 = O(k/\epsilon)$.

$$\sum_{i=1}^{n} \|U_1^i Z_1 D_{W_1^i} S_1 - A_1^i D_{W_1^i} S_1\|_2^2.$$

Let $\widehat{U}_1$ denote the optimal solution of the sketch problem,

$$\widehat{U}_1 = \arg\min_{U_1 \in \mathbb{R}^{n \times k}} \sum_{i=1}^{n} \|U_1^i Z_1 D_{W_1^i} S_1 - A_1^i D_{W_1^i} S_1\|_2^2.$$

By properties of S_1([RSW16]), plugging $\widehat{U} \in \mathbb{R}^{n \times k}$ into the original problem, we obtain,

$$\sum_{i=1}^{n} \|\widehat{U}_1^i Z_1 D_{W_1^i} - A_1^i D_{W_1^i}\|_2^2 \leq (1+\epsilon)\,\mathrm{OPT}\,.$$

Note that $\widehat{U}_1 \in \mathbb{R}^{n \times k}$ also has the following form. For each $i \in [n]$,

$$\begin{aligned}\widehat{U}_1^i &= A_1^i D_{W_1^i} S_1 (Z_1 D_{W_1^i} S_1)^\dagger \\ &= A_1^i D_{W_1^i} S_1 (Z_1 D_{W_1^i} S_1)^\top ((Z_1 D_{W_1^i} S_1)(Z_1 D_{W_1^i} S_1)^\top)^{-1}.\end{aligned}$$

Note that W_1 has r distinct rows. Thus, we only have r distinct $D_{W_1^i}$. This implies that there are r distinct matrices $Z_1 D_{W_1^i} S_1 \in \mathbb{R}^{k \times s_1}$. Using the definition of $g_{1,j}$, for $j \in [r]$, for

each $i \in g_{1,j} \subset [n]$, we have

$$\begin{aligned}\widehat{U}_1^i &= A_1^i D_{W_1^i} S_1 (Z_1 D_{W_1^i} S_1)^\dagger \\ &= A_1^i D_{W_1^j} S_1 (Z_1 D_{W_1^j} S_1)^\dagger \qquad \text{by } W_1^i = W_1^j,\end{aligned}$$

which means we only need to write down r different $Z_1 D_{W_1^j} S_1$. For each $k \times s_1$ matrix $Z_1 D_{W_1^j} S_1$, we create $k \times s_1$ variables to represent it. Thus, we need to create rks_1 variables to represent r matrices,

$$\{Z_1 D_{W_1^1} S_1, Z_1 D_{W_1^2} S_1, \cdots, Z_1 D_{W_1^r} S_1\}.$$

For simplicity, let $P_{1,i} \in \mathbb{R}^{k \times s_1}$ denote $Z_1 D_{W_1^i} S_1$. Then we can rewrite $\widehat{U}^i \in \mathbb{R}^k$ as follows,

$$\widehat{U}_1^i = A_1^i D_{W_1^i} S_1 P_{1,i}^\top (P_{1,i} P_{1,i}^\top)^{-1}.$$

If $P_{1,i} P_{1,i}^\top \in \mathbb{R}^{k \times k}$ has rank k, then we can use Cramer's rule (Lemma 21.8.1) to write down the inverse of $P_{1,i} P_{1,i}^\top$. However, vector W_1^i could have many zero entries. Then the rank of $P_{1,i} P_{1,i}^\top$ can be smaller than k. There are two different ways to solve this issue.

One way is by using the argument from [RSW16], which allows us to assume that $P_{1,i} P_{1,i}^\top \in \mathbb{R}^{k \times k}$ has rank k.

The other way is straightforward: we can guess the rank. There are k possibilities. Let $t_i \leq k$ denote the rank of $P_{1,i}$. Then we need to figure out a maximal linearly independent subset of rows of $P_{1,i}$. There are $2^{O(k)}$ possibilities. Next, we need to figure out a maximal linearly independent subset of columns of $P_{1,i}$. We can also guess all the possibilities, which is at most $2^{O(k)}$. Because we have r different $P_{1,i}$, the total number of guesses we have is at most $2^{O(rk)}$. Thus, we can write down $(P_{1,i} P_{1,i}^\top)^{-1}$ according to Cramer's rule.

After $\widehat{U}_1$ is obtained, we will fix $\widehat{U}_1$ and U_3^* in the next round. We consider the flattening of the tensor along the second direction,

$$\min_{U_2 \in \mathbb{R}^{n \times k}} \|(U_2 Z_2 - A_2) \circ W_2\|_F^2,$$

where $n \times n^2$ matrix A_2 is obtained by flattening tensor A along the second dimension, $k \times n^2$ matrix Z_2 denotes $\widehat{U}_1^\top \odot U_3^{*\top}$, and $n \times n^2$ matrix W_2 is obtained by flattening tensor W along the second dimension. For each $i \in [n]$, let W_2^i denote the i-th row of $n \times n^2$ matrix W_2. For each $i \in [n]$, let $D_{W_1^i}$ denote the diagonal matrix which has size $n^2 \times n^2$ and for which each entry is from vector $W_2^i \in \mathbb{R}^{n^2}$. Without loss of generality, we can assume the first r rows of W_2 are distinct. We can rewrite the objective function along the second dimension as a sum of multiple regression problems. For any $n \times k$ matrix U_2,

$$\|(U_2 Z_2 - A_2) \circ W_2\|_F^2 = \sum_{i=1}^n \|U_2^i Z_2 D_{W_2^i} - A_2^i D_{W_2^i}\|_2^2. \tag{21.48}$$

Based on the observation that W_2 has r distinct rows, we can group the n rows of W^2 into r groups. We use $g_{2,1}, g_{2,2}, \cdots, g_{2,r}$ to denote r sets of indices such that, for each $i \in g_{2,j}$, $W_2^i = W_2^j$. Thus we obtain,

$$\begin{aligned} \|(U_2 Z_2 - A_2) \circ W_2\|_F^2 = & \sum_{i=1}^n \|U_2^i Z_2 D_{W_2^i} - A_2^i D_{W_2^i}\|_2^2 \\ = & \sum_{j=1}^r \sum_{i \in g_{2,j}} \|U_2^i Z_2 D_{W_2^i} - A_2^i D_{W_2^i}\|_2^2. \end{aligned}$$

We can sketch the objective function by choosing a Gaussian sketch $S_2 \in \mathbb{R}^{n^2 \times s_2}$ with $s_2 = O(k/\epsilon)$. Let $\widehat{U}_2$ denote the optimal solution to the sketch problem. Then $\widehat{U}_2$ has the form, for each $i \in [n]$,

$$\widehat{U}_2^i = A_2^i D_{W_2^i} S_2 (Z_2 D_{W_2^i} S_2)^\dagger.$$

Similarly as before, we only need to write down r different matrices $Z_2 D_{W_2^i} S_1$, and for each of them, create $k \times s_2$ variables. Let $P_{2,i} \in \mathbb{R}^{k \times s_2}$ denote $Z_2 D_{W_2^i} S_2$. By our guessing argument, we can obtain $\widehat{U}_2$.

In the last round, we fix $\widehat{U}_1$ and $\widehat{U}_2$. We then write down $\widehat{U}_3$. Overall, by creating $l = O(rk^2/\epsilon)$ variables, we have rational polynomials $\widehat{U}_1(x)$, $\widehat{U}_2(x)$, $\widehat{U}_3(x)$. Putting it all together, we can write this objective function,

$$\begin{aligned} \min_{x \in \mathbb{R}^l} & \|(\widehat{U}_1(x) \otimes \widehat{U}_2(x) \otimes \widehat{U}_3(x) - A) \circ W\|_F^2. \\ \text{s.t. } & h_{1,i}(x) \neq 0, \forall i \in [r]. \\ & h_{2,i}(x) \neq 0, \forall i \in [r]. \\ & h_{3,i}(x) \neq 0, \forall i \in [r]. \end{aligned}$$

where $h_{1,i}(x)$ denotes the denominator polynomial related to a full rank sub-block of $P_{1,i}(x)$. By a perturbation argument in Section 4 in [RSW16], we know that the $h_{1,i}(x)$ are nonzero. By a similar argument as in Section 5 in [RSW16], we can show a lower bound on the cost of the denominator polynomial $h_{1,i}(x)$. Thus we can create new bounded variables $x_{l+1}, \cdots, x_{3r+l}$ to rewrite the objective function,

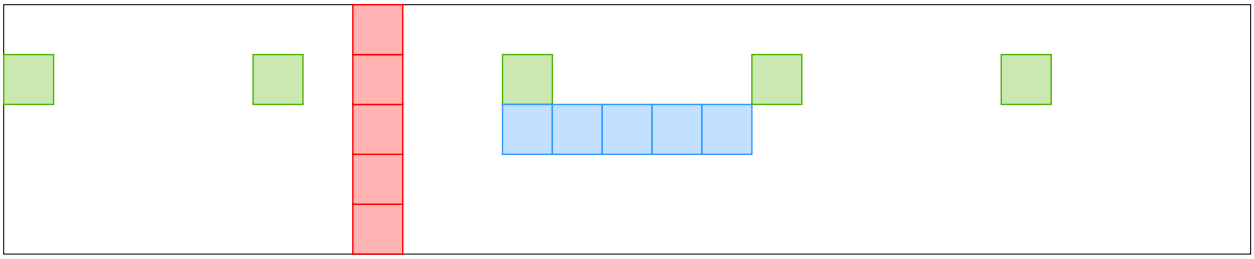
$$\begin{aligned}
\min_{x\in\mathbb{R}^{l+3r}} \; & q(x)/p(x). \\
\text{s.t. } & h_{1,i}(x)x_{l+i} = 0, \forall i \in [r]. \\
& h_{2,i}(x)x_{l+r+i} = 0, \forall i \in [r]. \\
& h_{3,i}(x)x_{l+2r+i} = 0, \forall i \in [r]. \\
& p(x) = \prod_{i=1}^{r} h_{1,i}^2(x)h_{2,i}^2(x)h_{3,i}^2(x)
\end{aligned}$$

Note that the degree of the above system is poly(kr) and all the equality constraints can be merged into one single constraint. Thus, the number of constraints is $O(1)$. The number of variables is $O(rk^2/\epsilon)$.

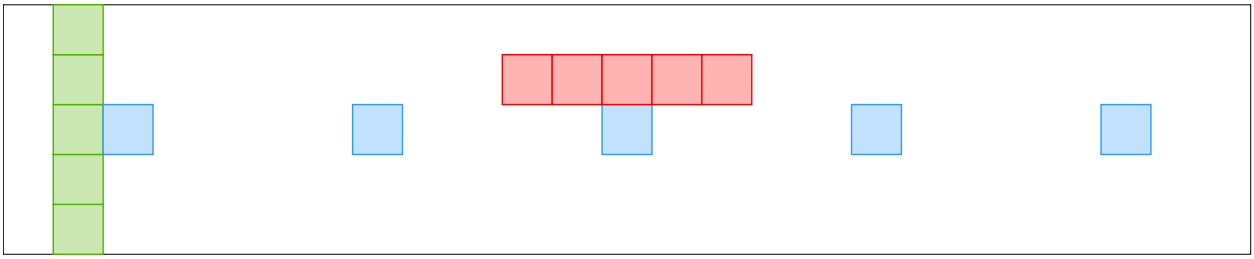
Using Theorem 21.3.2 and a similar argument from Section 5 of [RSW16], we have that the minimum nonzero cost is at least $2^{-n^\delta 2^{\widetilde{O}(rk^2/\epsilon)}}$. Combining the binary search explained in Section 21.4(similar techniques also can be found in Section 6 of [RSW16]) with the lower bound we obtained, we can find the solution for the original problem in time,

$$(\text{nnz}(A) + \text{nnz}(W) + n2^{\widetilde{O}(rk^2/\epsilon)})n^{O(\delta)}.$$

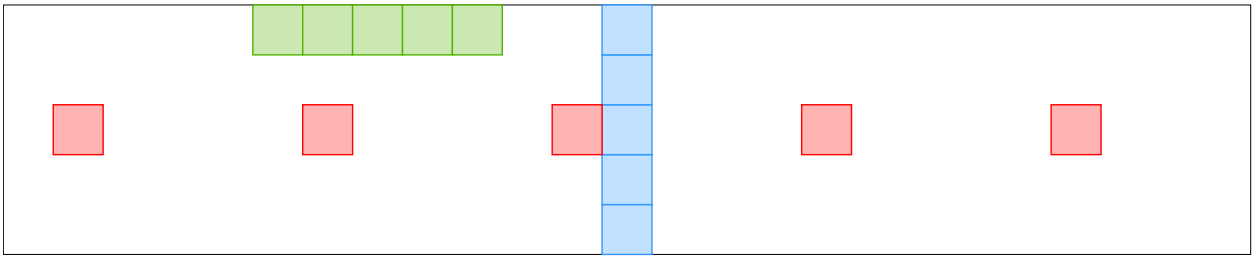
□



$(W_1)_{i,(j-1)n+l} = W_{i,j,l}$



$(W_2)_{j,(l-1)n+i} = W_{i,j,l}$



$(W_3)_{l,(i-1)n+j} = W_{i,j,l}$

Figure 21.7: Let W denote a tensor that has columns(red), rows(green) and tubes(blue). For each $i \in [3]$, let W_i denote the matrix obtained by flattening tensor W along the i-th dimension.

21.8.3 r distinct columns, rows and tubes

Lemma 21.8.3. *Let $W \in \mathbb{R}^{n \times n \times n}$ denote a tensor that has r distinct columns and r distinct rows, then W has*

(I) *r distinct column-tube faces.*

(II) *r distinct row-tube faces.*

Proof. Proof of Part (I). Without loss of generality, we consider the first (which is the bottom one) column-row face. Assume it has r distinct rows and r distinct columns. We can re-order all the column-tube faces to make sure that all the n columns in the bottom face have been split into r continuous disjoint groups C_i, e.g., $\{C_1, C_2, \cdots, C_r\} = [n]$. Next, we can re-order all the row-tube faces to make sure that all the n rows in the bottom face have been split into r continuous disjoint groups R_i, e.g., $\{R_1, R_2, \cdots, R_r\} = [n]$. Thus, the new bottom face can be regarded as $r \times r$ groups, and the number in each position of the same group is the same.

Suppose that the tensor has $r + 1$ distinct column-tube faces. By the pigeonhole principle there exist two different column-tube faces belonging to the same group C_i, for some $i \in [r]$. Note that these two column-tube faces are the same by looking at the bottom (column-row) face. Since they are distinct faces, there must exist one row vector v which is not in the bottom (column-row) face, and it has a different value in coordinates belong to group C_i. Note that, considering the bottom face, for each row vector, it has the same value over coordinates belonging to group C_i. But v has different values in coordinates belong to group C_i. Also, note that the bottom (column-row) face also has r distinct rows, and v is not one of them. This means there are at least $r + 1$ distinct rows, which contradicts that there are r distinct rows in total. Thus, there are at most r distinct column-tube faces.

Proof of Part (II). It is similar to Part (I). □

Corollary 21.8.4. *Let $W \in \mathbb{R}^{n \times n \times n}$ denote a tensor that has r distinct columns, r distinct rows, and r distinct rubes. Then W has r distinct column-tube faces, r distinct row-tube faces, and r distinct column-row faces.*

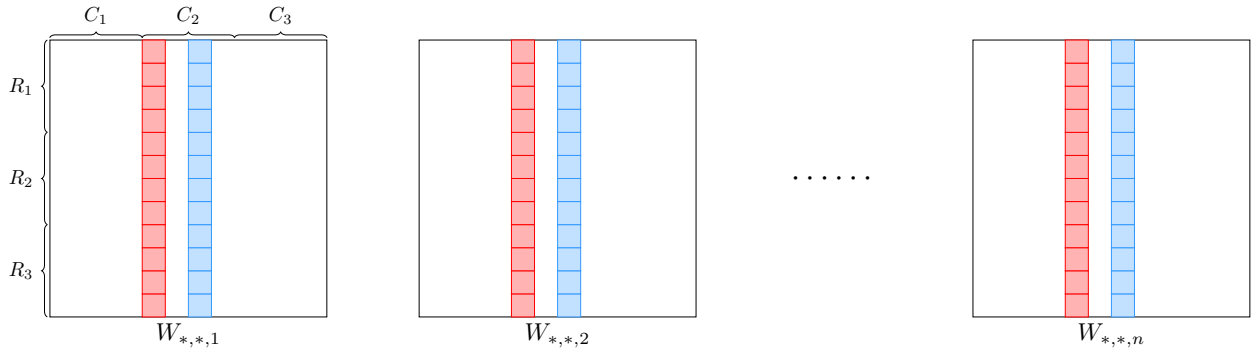


Figure 21.8: Each face $W_{*,*,i}$ is a column-row face. $W_{*,*,1}$ is the bottom column-row face. $r = 3$. The blue blocks represent column-tube faces, the red blocks represent column-tube faces.

Proof. This follows by applying Lemma 21.8.3 twice. □

Thus, we obtain the same result as in Theorem 21.8.2 by changing the assumption from r distinct faces in each dimension to r distinct columns, r distinct rows and r distinct tubes.

21.8.4 r distinct columns and rows

The main difference between Theorem 21.8.2 and Theorem 21.8.5 is the running time. The first one takes $2^{\widetilde{O}(rk^2/\epsilon)}$ time and the second one is slightly longer, $2^{\widetilde{O}(r^2k^2/\epsilon)}$. By Lemma 21.8.3, r distinct columns in two dimensions implies r distinct faces in two of the three kinds of faces. Thus, the following theorem also holds for r distinct columns in two dimensions.

Algorithm 21.32 Weighted Tensor Low-rank Approximation Algorithm when the Weighted Tensor has r Distinct Faces in Each of the Two Dimensions.

procedure WEIGHTEDRDISTINCTFACESIN2DIMENSIONS(A, W, n, r, k, ϵ) ▷ Theorem 21.8.5
  for $j = 1 \to 3$ **do**
    $s_j \leftarrow O(k/\epsilon)$.
    Choose a sketching matrix $S_j \in \mathbb{R}^{n^2 \times s_j}$.
    if $j \neq 3$ **then**
      for $i = 1 \to r$ **do**
        Create $k \times s_1$ variables for matrix $P_{i,j} \in \mathbb{R}^{k \times s_j}$.
      end for
    end if
    for $i = 1 \to n$ **do**
      Write down $(\widehat{U}_j)^i = A_i^j D_{W_1^j} S_j P_{j,i}^\top (P_{j,i} P_{j,i}^\top)^{-1}$.
    end for
  end for
  Form $\|W \circ (\widehat{U}_1 \otimes \widehat{U}_2 \otimes \widehat{U}_3 - A)\|_F^2$.
  Run polynomial system verifier.
  return U_1, U_2, U_3
end procedure

Theorem 21.8.5. *Given a 3rd order $n \times n \times n$ tensor A and an $n \times n \times n$ tensor W of weights with r distinct faces in two dimensions (out of three dimensions) such that each entry can be written using $O(n^\delta)$ bits for some $\delta > 0$, define* $\mathrm{OPT} = \inf_{\mathrm{rank}-k\ A_k} \|W \circ (A_k - A)\|_F^2$.

For any $k \geq 1$ and any $0 < \epsilon < 1$.

(I) *If* $\mathrm{OPT} > 0$, *and there exists a rank-k* $A_k = U_1^* \otimes U_2^* \otimes U_3^*$ *tensor (with size* $n \times n \times n$*) such that* $\|W \circ (A_k - A)\|_F^2 = \mathrm{OPT}$, *and* $\max_{i \in [3]} \|U_i^*\|_F \leq 2^{O(n^\delta)}$, *then there exists an algorithm that takes* $(\mathrm{nnz}(A) + \mathrm{nnz}(W) + n2^{\widetilde{O}(r^2k^2/\epsilon)})n^{O(\delta)}$ *time in the unit cost* RAM *model with words of size* $O(\log n)$ *bits*[11] *and outputs three* $n \times k$ *matrices* U_1, U_2, U_3 *such that*

$$\|W \circ (U_1 \otimes U_2 \otimes U_3 - A)\|_F^2 \leq (1 + \epsilon)\,\mathrm{OPT} \tag{21.49}$$

holds with probability $9/10$.

(II) *If* $\mathrm{OPT} > 0$, A_k *does not exist, and there exist three* $n \times k$ *matrices* U_1', U_2', U_3' *where each entry can be written using* $O(n^\delta)$ *bits and* $\|W \circ (U_1' \otimes U_2' \otimes U_3' - A)\|_F^2 \leq (1 + \epsilon/2)\,\mathrm{OPT}$, *then we can find* U, V, W *such that* (21.49) *holds.*

(III) *If* $\mathrm{OPT} = 0$, A_k *exists, and there exists a solution* U_1^*, U_2^*, U_3^* *such that each entry of the matrix can be written using* $O(n^\delta)$ *bits, then we can obtain* (21.49).

(IV) *If* $\mathrm{OPT} = 0$, *and there exist three* $n \times k$ *matrices* U_1, U_2, U_3 *such that* $\max_{i \in [3]} \|U_i^*\|_F \leq 2^{O(n^\delta)}$ *and*

$$\|W \circ (U_1 \otimes U_2 \otimes U_3 - A)\|_F^2 \leq (1 + \epsilon)\,\mathrm{OPT} + 2^{-\Omega(n^\delta)}, \tag{21.50}$$

then we can output U_1, U_2, U_3 *such that* (21.50) *holds.*

(V) *Further if* A_k *exists, we can output a number* Z *for which* $\mathrm{OPT} \leq Z \leq (1+\epsilon)\,\mathrm{OPT}$.

For all the cases, the algorithm succeeds with probability at least $9/10$.

[11] The entries of A and W are assumed to fit in n^δ words.

Proof. By Lemma 21.8.3, we have W has r distinct column-tube faces and r distinct row-tube faces. By Claim 21.8.6, we know that W has $R = 2^{O(r \log r)}$ distinct column-row faces.

We use the same approach as in proof of Theorem 21.8.2 (which is also similar to Section 8 of [RSW16]) to create variables, write down the polynomial systems and add not equal constraints. Instead of having $3r$ distinct denominators as in the proof of Theorem 21.8.2, we have $2r + R$.

We create $l = O(rk^2/\epsilon)$ variables for $\{Z_1 D_{W_1^1} S_1, Z_1 D_{W_1^2} S_1, \cdots, Z_1 D_{W_1^r} S_1\}$. Then we can write down $\widehat{U}_1$ with r distinct denominators $g_i(x)$. Each $g_i(x)$ is non-zero in an optimal solution using the perturbation argument in Section 4 in [RSW16]. We create new variables x_{2l+i} to remove the denominators $g_i(x)$, $\forall i \in [r]$. Then the entries of $\widehat{U}_1$ are polynomials as opposed to rational functions.

We create $l = O(rk^2/\epsilon)$ variables for $\{Z_2 D_{W_2^1} S_2, Z_2 D_{W_2^2} S_2, \cdots, Z_2 D_{W_2^r} S_2\}$. Then we can write down $\widehat{U}_2$ with r distinct denominators $g_{r+i}(x)$. Each $g_{r+i}(x)$ is non-zero in an optimal solution using the perturbation argument in Section 4 in [RSW16]. We create new variables x_{2l+r+i} to remove the denominators $g_{r+i}(x)$, $\forall i \in [r]$. Then the entries of $\widehat{U}_2$ are polynomials as opposed to rational functions.

Using $\widehat{U}_1$ and $\widehat{U}_2$ we can express $\widehat{U}_3$ with R distinct denominators $f_i(x)$, which are also non-zero by using the perturbation argument in Section 4 in [RSW16], and using that W_3 has at most this number of distinct rows. Finally we can write the following optimization

problem,

$$\begin{aligned} \min_{x\in\mathbb{R}^{2l+2r}} \quad & p(x)/q(x) \\ \text{s.t.} \quad & g_i(x)x_{2l+i} - 1 = 0, \forall i \in [r] \\ & g_{r+i}(x)x_{2l+r+i} - 1 = 0, \forall i \in [r] \\ & f_j^2(x) \neq 0, \forall j \in [R] \\ & q(x) = \prod_{j=1}^{R} f_j^2(x) \end{aligned}$$

We then determine if there exists a solution to the above semi-algebraic set in time

$$(\mathrm{poly}(k,r)R)^{O(rk^2/\epsilon)} = 2^{\widetilde{O}(r^2k^2/\epsilon)}.$$

Using similar techniques from Section 5 of [RSW16], we can show a lower bound on the cost similar to Section 8.3 of [RSW16], namely, the minimum nonzero cost is at least

$$2^{-n^\delta 2^{\widetilde{O}(r^2k^2/\epsilon)}}.$$

Combining the binary search explained in Section 21.4 (a similar techniques also can be found in Section 6 of [RSW16]) with the lower bound we obtained, we can find a solution for the original problem in time

$$(\mathrm{nnz}(A) + \mathrm{nnz}(W) + n2^{\widetilde{O}(r^2k^2/\epsilon)})n^{O(\delta)}.$$

□

Remark 21.8.1. Note that the running time for the Frobenius norm and for the ℓ_1 norm are of the form $\mathrm{poly}(n) + \exp(\mathrm{poly}(k/\epsilon))$ rather than $\mathrm{poly}(n) \cdot \exp(k/\epsilon)$. The reason is, we can use an input sparsity reduction to reduce the size of the objective function from $\mathrm{poly}(n)$ to $\mathrm{poly}(k)$.

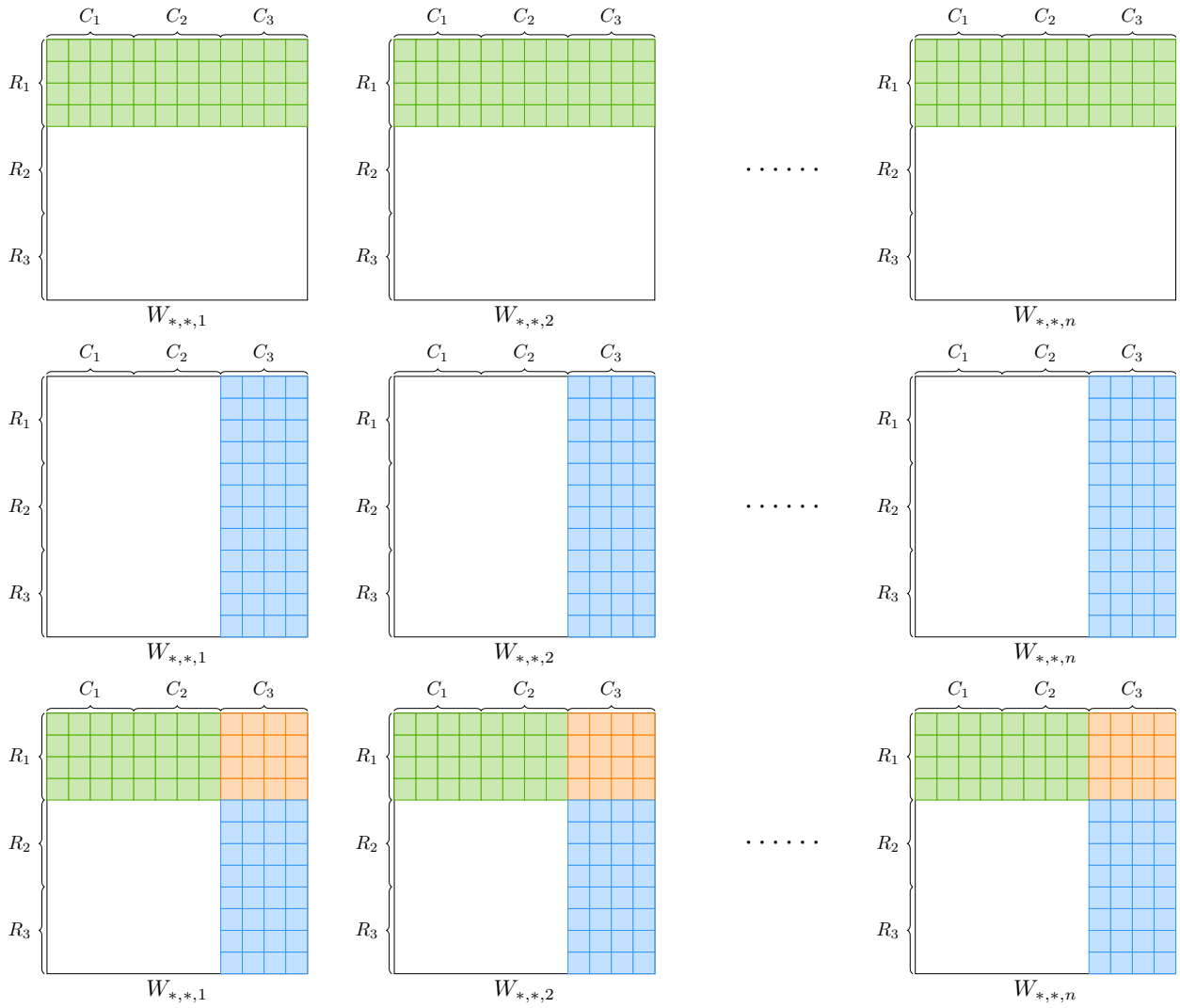


Figure 21.9: Each face $W_{*,*,i}$ is a column-row face. $W_{*,*,1}$ is the bottom column-row face. $r = 3$. The blue blocks represent $|C_3|$ column-tube faces. The green blocks represet $|R_3|$ row-tube faces. In each column-row face, the intersection between blue faces and green faces is a size $|R_3| \times |C_3|$ block, and all the entries in this block are the same.

Claim 21.8.6. *Let $W \in \mathbb{R}$ denote a third order tensor that has r distinct columns and r distinct rows. Then it has $2^{O(r \log r)}$ distinct column-row faces.*

Proof. By similar arguments as in the proof of Lemma 21.8.3, the bottom (column-row) face can be split into r groups $C_1, C_2, \cdots, C_r$ based on r columns, and split into r groups $R_1, R_2, \cdots, R_r$ based on rows. Thus, the bottom (column-row) face can be regarded as having $r \times r$ groups, and the number in each position of the same group is the same.

We can assume that all the r^2 blocks in the bottom column-row face have the same size. Otherwise, we can expand the tensor to the situation that all the r^2 blocks have the same size. Because this small tensor is a sub-tensor of the big tensor, if the big tensor has at most t distinct column-row faces, then the small tensor has at most t distinct column-row faces.

By Lemma 21.8.3, we know that the tensor W has at most r distinct column-tube faces and row-tube faces. Because it has r distinct column-tube faces, then all the faces belonging to coordinates in C_r are the same. Thus, all the columns belonging to C_r and in the second column-row face are the same. Similarly, we have that all the rows belonging to R_r and in the second column-row face are the same. Thus we have that all the entries in block $C_R \cup R_r$ and in the second column-row faces are the same. Further, we can conclude, for every column-row face, for every $C_i \cup R_j$ block, all the entries in the same block are the same.

The next observation is, if there exist $r^2 + 1$ different values in the tensor, then there exist either r distinct columns or r distinct rows. Indeed, otherwise since we have r distinct columns, each column has at most r distinct entries given our bound on the nunber of distinct rows. Thus, the r distinct columns could have at most r^2 distinct entries in total, a contradiction.

For each column-row face, there are at most r^2 blocks, and the value in each block can have at most r^2 possibilities. Thus, overall we have at most $(r^2)^{r^2} = 2^{O(r^2 \log r)}$ column-row faces.

By using different argument, we can improve the above bound. Note that we already show in each column-row face of a tensor, it has r^2 blocks, and all the values in each block have to be the same. Since we have r distinct rows, we can fix the those r distinct rows. If we copy row v into one row of R_i, then we have to copy row v into every row of R_i. This is because if R_i contains two distinct rows, then there must exist a block C_j for which the entries in block $R_i \cup C_j$ are not all the same. Thus, for each row group, all the rows in that group are the same.

Now, for each column-row face, consider the leftmost r blocks, $R_1 \cup C_1$, $R_2 \cup C_1$, $\cdots$, $R_r \cup C_1$. There are at most r possible values in each block, because we have r distinct rows in total. Overall the total number of possibilities for the leftmost r blocks is at most $(r)^r = 2^{O(r \log r)}$. Once the leftmost r blocks are determined, the remaining $r(r-1)$ are also determined. This completes the proof.

□

Also, notice that there is an example that has $2^{\Omega(r \log r)}$ distinct column-row faces. For the bottom column-row faces, there are $r \times r$ blocks for which all the blocks have the same size, the blocks on the diagonal have all 1s, and all the other blocks contain 0s everywhere. For the later column-row faces, we can arbitrarily permute this block diagonal matrix, and the total number of possibilities is $\Omega(r!) \geq 2^{\Omega(r \log r)}$.

21.9 Hardness

We first provide definitions and results for some fundamental problems in Section 21.9.1. Section 21.9.2 presents our hardness result for the symmetric tensor eigenvalue problem. Section 21.9.3 presents our hardness results for symmetric tensor singular value problems, computing tensor spectral norm, and rank-1 approximation. We improve Håstad's NP-hardness[Hås90] result for tensor rank in Section 21.9.4.

21.9.1 Definitions

We first provide the definitions for 3SAT , ETH , MAX-3SAT , MAX-E3SAT and then state some fundamental results related to those definitions.

Definition 21.9.1 (3SAT problem)**.** Given n variables and m clauses in a conjunctive normal form CNF formula with the size of each clause at most 3, the goal is to decide whether there exists an assignment to the n Boolean variables to make the CNF formula be satisfied.

Hypothesis 21.9.1 (Exponential Time Hypothesis (ETH) [IPZ98])**.** *There is a $\delta > 0$ such that the* 3SAT *problem defined in Definition 21.9.1 cannot be solved in $O(2^{\delta n})$ time.*

Definition 21.9.2 (MAX-3SAT)**.** Given n variables and m clauses, a conjunctive normal form CNF formula with the size of each clause at most 3, the goal is to find an assignment that satisfies the largest number of clauses.

We use MAX-E3SAT to denote the version of MAX-3SAT where each clause contains exactly 3 literals.

Theorem 21.9.2 ([Hås01])**.** *For every $\delta > 0$, it is* NP*-hard to distinguish a satisfiable instance of* MAX-E3SAT *from an instance where at most a $7/8 + \delta$ fraction of the clauses can be simultaneously satisfied.*

Theorem 21.9.3 ([Hås01, MR10])**.** *Assume* ETH *holds. For every $\delta > 0$, there is no $2^{o(n^{1-o(1)})}$ time algorithm to distinguish a satisfiable instance of* MAX-E3SAT *from an instance where at most a fraction $7/8 + \delta$ of the clauses can be simultaneously satisfied.*

We use MAX-E3SAT(B) to denote the restricted special case of MAX-3SAT where every variable occurs in at most B clauses. Håstad [Hås00] proved that the problem is

approximable to within a factor $7/8 + 1/(64B)$ in polynomial time, and that it is hard to approximate within a factor $7/8 + 1/(\log B)^{\Omega(1)}$. In 2001, Trevisan improved the hardness result,

Theorem 21.9.4 ([Tre01])**.** *Unless* **RP**=**NP**, *there is no polynomial time* $(7/8 + 5/\sqrt{B})$*-approximate algorithm for* MAX-E3SAT(B) .

Theorem 21.9.5 ([Hås01, Tre01, MR10])**.** *Unless* ETH *fails, there is no* $2^{o(n^{1-o(1)})}$ *time* $(7/8 + 5/\sqrt{B})$*-approximate algorithm for* MAX-E3SAT(B) .

Theorem 21.9.6 ([LMS11])**.** *Unless* ETH *fails, there is no* $2^{o(n)}$ *time algorithm for the Independent Set problem.*

Definition 21.9.3 (MAX-CUT decision problem)**.** Given a positive integer c^* and an unweighted graph $G = (V, E)$ where V is the set of vertices of G and E is the set of edges of G, the goal is to determine whether there is a cut of G that has at least c^* edges.

Note that Feige's original assumption[Fei02] states that there is no polynomial time algorithm for the problem in Assumption 21.9.7. We do not know of any better algorithm for the problem in Assumption 21.9.7 and have consulted several experts[12] about the assumption who do not know a counterexample to it.

Assumption 21.9.7 (Random Exponential Time Hypothesis)**.** *Let* $c > \ln 2$ *be a constant. Consider a random* 3SAT *formula on* n *variables in which each clause has* 3 *literals, and in which each of the* $8n^3$ *clauses is picked independently with probability* c/n^2*. Then any*

[12]Personal communication with Russell Impagliazzo and Ryan Williams.

algorithm which always outputs 1 *when the random formula is satisfiable, and outputs* 0 *with probability at least* $1/2$ *when the random formula is unsatisfiable, must run in* $2^{c'n}$ *time on some input, where* $c' > 0$ *is an absolute constant.*

The 4SAT-version of the above random-ETH assumption has been used in [GL04] and [RSW16] (Assumption 1.3).

21.9.2 Symmetric tensor eigenvalue

Definition 21.9.4 (Tensor Eigenvalue [HL13])**.** An eigenvector of a tensor $A \in \mathbb{R}^{n\times n\times n}$ is a nonzero vector $x \in \mathbb{R}^n$ such that

$$\sum_{i=1}^{n}\sum_{j=1}^{n} A_{i,j,k}x_ix_j = \lambda x_k, \forall k \in [n]$$

for some $\lambda \in \mathbb{R}$, which is called an eigenvalue of A.

Theorem 21.9.8 ([N+03])**.** *Let $G = (V, E)$ on v vertices have stability number (the size of a maximum independent set) $\alpha(G)$. Let $n = v + \frac{v(v-1)}{2}$ and $\mathbb{S}^{n-1} = \{(x, y) \in \mathbb{R}^v \times \mathbb{R}^{v(v-1)/2} : \|x\|_2^2 + \|y\|_2^2 = 1\}$. Then,*

$$\sqrt{1 - \frac{1}{\alpha(G)}} = 3\sqrt{3/2} \max_{(x,y)\in\mathbb{S}^{n-1}} \sum_{i<j,(i,j)\notin E} x_ix_jy_{i,j}.$$

For any graph $G(V, E)$, we can construct a symmetric tensor $A \in \mathbb{R}^{n\times n\times n}$. For any $1 \le i < j < k \le v$, let

$$A_{i,j,k} = \begin{cases} 1 & 1 \le i < j \le v, k = v + \phi(i, j), (i, j) \notin E, \\ 0 & \text{otherwise,} \end{cases}$$

where $\phi(i, j) = (i-1)v - i(i-1)/2 + j - i$ is a lexicographical enumeration of the $v(v-1)/2$ pairs $i < j$. For the other cases $i < k < j, \cdots, k < j < i$, we set

$$A_{i,j,k} = A_{i,k,j} = A_{j,i,k} = A_{j,k,i} = A_{k,i,j} = A_{k,j,i}.$$

If two or more indices are equal, we set $A_{i,j,k} = 0$. Thus tensor T has the following property,

$$A(z, z, z) = 6 \sum_{i<j,(i,j)\notin E} x_ix_jy_{i,j},$$

where $z = (x, y) \in \mathbb{R}^n$.

Thus, we have

$$\lambda = \max_{z \in \mathbb{S}^{n-1}} A(z,z,z) = \max_{(x,y) \in \mathbb{S}^{n-1}} 6 \sum_{i<j,(i,j) \notin E} x_i x_j y_{i,j}.$$

Furthermore, λ is the maximum eigenvalue of A.

Theorem 21.9.9. *Unless* ETH *fails, there is no* $2^{o(\sqrt{n})}$ *time to approximate the largest eigenvalue of an* n*-dimensional symmetric tensor within* $(1 \pm \Theta(1/n))$ *relative error.*

Proof. The additive error is at least

$$\sqrt{1-1/v} - \sqrt{1-1/(v-1)} = \frac{1/(v-1) - 1/v}{\sqrt{1-1/v} + \sqrt{1-1/(v-1)}} \gtrsim 1/(v-1) - 1/v \geq 1/v^2.$$

Thus, the relative error is $(1 \pm \Theta(1/v^2))$. By the definition of n, we know $n = \Theta(v^2)$. Assuming ETH , there is no $2^{o(v)}$ time algorithm to compute the clique number of $\overline{G}$. Because the clique number of $\overline{G}$ is $\alpha(G)$, there is no $2^{o(v)}$ time algorithm to compute $\alpha(G)$. Furthermore, there is no $2^{o(v)}$ time algorithm to approximate the maximum eigenvalue within $(1 \pm \Theta(1/v^2))$ relative error. Thus, we complete the proof. □

Corollary 21.9.10. *Unless* ETH *fails, there is no polynomial running time algorithm to approximate the largest eigenvalue of an* n*-dimensional tensor within* $(1 \pm \Theta(1/\log^{2+\gamma}(n)))$ *relative-error, where* $\gamma > 0$ *is an arbitrarily small constant.*

Proof. We can apply a padding argument here. According to Theorem 21.9.9, there is a d-dimensional tensor such that there is no $2^{o(\sqrt{d})}$ time algorithm that can give a $(1 + \Theta(1/d))$ relative error approximation. If we pad 0s everywhere to extend the size of the tensor

to $n = 2^{d^{(1-\gamma')/2}}$, where $\gamma' > 0$ is a sufficiently small constant, then $\text{poly}(n) = 2^{o(\sqrt{d})}$, so $d = \log^{2+O(\gamma')}(n)$. Thus, it means that there is no polynomial running time algorithm which can output a $(1 + 1/(\log^{2+\gamma}))$-relative approximation to the tensor which has size n.

□

21.9.3 Symmetric tensor singular value, spectral norm and rank-1 approximation

[HL13] defines two kinds of singular values of a tensor. In this paper, we only consider the following kind:

Definition 21.9.5 (ℓ_2 singular value in [HL13])**.** Given a 3rd order tensor $A \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, the number $\sigma \in \mathbb{R}$ is called a singular value and the nonzero $u \in \mathbb{R}^{n_1}$,$v \in \mathbb{R}^{n_2}$,$w \in \mathbb{R}^{n_3}$ are called singular vectors of A if

$$\begin{aligned}
\sum_{j=1}^{n_2}\sum_{k=1}^{n_3} A_{i,j,k} v_j w_k &= \sigma u_i, \forall i \in [n_1] \\
\sum_{i=1}^{n_1}\sum_{k=1}^{n_3} A_{i,j,k} u_i w_k &= \sigma v_j, \forall j \in [n_2] \\
\sum_{i=1}^{n_1}\sum_{j=1}^{n_2} A_{i,j,k} u_i v_j &= \sigma w_k, \forall k \in [n_3].
\end{aligned}$$

Definition 21.9.6 (Spectral norm [HL13])**.** The spectral norm of a tensor A is:

$$\|A\|_2 = \sup_{x,y,z \neq 0} \frac{|A(x,y,z)|}{\|x\|_2 \|y\|_2 \|z\|_2}$$

Notice that the spectral norm is the absolute value of either the maximum value of $\frac{A(x,y,z)}{\|x\|_2\|y\|_2\|z\|_2}$ or the minimum value of it. Thus, it is an ℓ_2-singular value of A. Furthermore, it is the maximum ℓ_2-singular value of A.

Theorem 21.9.11 ([Ban38])**.** *Let $A \in \mathbb{R}^{n \times n \times n}$ be a symmetric 3rd order tensor. Then,*

$$\|A\|_2 = \sup_{x,y,z \neq 0} \frac{A(x,y,z)}{\|x\|_2 \|y\|_2 \|z\|_2} = \sup_{x \neq 0} \frac{|A(x,x,x)|}{\|x\|_2^3}.$$

It means that if a tensor is symmetric, then its largest eigenvalue is the same as its largest singular value and its spectral norm. Then, by combining with Theorem 21.9.9, we have the following corollary:

Corollary 21.9.12. *Unless* ETH *fails,*

1. *There is no $2^{o(\sqrt{n})}$ time algorithm to approximate the largest singular value of an n-dimensional symmetric tensor within $(1 + \Theta(1/n))$ relative-error.*

2. *There is no $2^{o(\sqrt{n})}$ time algorithm to approximate the spectral norm of an n-dimensional symmetric tensor within $(1 + \Theta(1/n))$ relative-error.*

By Corollary 21.9.10, we have:

Corollary 21.9.13. *Unless* ETH *fails,*

1. *There is no polynomial time algorithm to approximate the largest singular value of an n-dimensional tensor within $(1 + \Theta(1/\log^{2+\gamma}(n)))$ relative-error, where $\gamma > 0$ is an arbitrarily small constant.*

2. *There is no polynomial time algorithm to approximate the spectral norm of an n-dimensional tensor within $(1 + \Theta(1/\log^{2+\gamma}(n)))$ relative-error, where $\gamma > 0$ is an arbitrarily small constant.*

Now, let us consider Frobenius norm rank-1 approximation.

Theorem 21.9.14 ([Ban38])**.** *Let $A \in \mathbb{R}^{n \times n \times n}$ be a symmetric 3rd order tensor. Then,*

$$\min_{\sigma \geq 0, \|u\|_2 = \|v\|_2 = \|w\|_2 = 1} \|A - \sigma u \otimes v \otimes w\|_F = \min_{\lambda \geq 0, \|v\|_2 = 1} \|A - \lambda v \otimes v \otimes v\|_F.$$

Furthermore, the optimal σ and λ may be chosen to be equal.

Notice that

$$\|A - \sigma u \otimes v \otimes w\|_F^2 = \|A\|_F^2 - 2\sigma A(u, v, w) + \sigma^2 \|u \otimes v \otimes w\|_F^2.$$

Then, if $\|u\|_2 = \|v\|_2 = \|w\|_2 = 1$, we have:

$$\|A - \sigma u \otimes v \otimes w\|_F^2 = \|A\|_F^2 - 2\sigma A(u, v, w) + \sigma^2.$$

When $A(u, v, w) = \sigma$, then the above is minimized.

Thus, we have:

$$\min_{\sigma \geq 0, \|u\|_2 = \|v\|_2 = \|w\|_2 = 1} \|A - \sigma u \otimes v \otimes w\|_F^2 + \|A\|_2^2 = \|A\|_F^2.$$

It is sufficient to prove the following theorem:

Theorem 21.9.15. *Given $A \in \mathbb{R}^{n \times n \times n}$, unless* ETH *fails, there is no $2^{o(\sqrt{n})}$ time algorithm to compute $u', v', w' \in \mathbb{R}^n$ such that*

$$\|A - u' \otimes v' \otimes w'\|_F^2 \leq (1 + \epsilon) \min_{u, v, w \in \mathbb{R}^n} \|A - u \otimes v \otimes w\|_F^2,$$

where $\epsilon = O(1/n^2)$.

Proof. Let $A \in \mathbb{R}^{n \times n \times n}$ be the same hard instance mentioned in Theorem 21.9.8. Notice that each entry of A is either 0 or 1. Thus, $\min_{u,v,w \in \mathbb{R}^n} \|A - u \otimes v \otimes w\|_F^2 \leq \|A\|_F^2$. Notice that Theorem 21.9.8 also implies that it is hard to distinguish the two cases $\|A\|_2 \leq 2\sqrt{2/3} \cdot \sqrt{1 - 1/c}$ or $\|A\|_2 \geq 2\sqrt{2/3} \cdot \sqrt{1 - 1/(c+1)}$ where c is an integer which is no greater than $\sqrt{n}$. So the difference between $(2\sqrt{2/3} \cdot \sqrt{1 - 1/c})^2$ and $(2\sqrt{2/3} \cdot \sqrt{1 - 1/(c+1)})^2$ is at

least $\Theta(1/n)$. Since $\|A\|_F^2$ is at most n (see construction of A in the proof of Lemma 21.9.8), $\Theta(1/n)$ is an $\epsilon = O(1/n^2)$ fraction of $\min_{u,v,w\in\mathbb{R}^n} \|A - u\otimes v\otimes w\|_F^2$. Because

$$\min_{u,v,w\in\mathbb{R}^n} \|A - u\otimes v\otimes w\|_F^2 + \|A\|_2^2 = \|A\|_F^2,$$

if we have a $2^{o(\sqrt{n})}$ time algorithm to compute $u', v', w' \in \mathbb{R}^n$ such that

$$\|A - u'\otimes v'\otimes w'\|_F^2 \leq (1+\epsilon) \min_{u,v,w\in\mathbb{R}^n} \|A - u\otimes v\otimes w\|_F^2$$

for $\epsilon = O(1/n^2)$, it will contradict the fact that we cannot distinguish whether $\|A\|_2 \leq 2\sqrt{2/3}\cdot\sqrt{1-1/c}$ or $\|A\|_2 \geq 2\sqrt{2/3}\cdot\sqrt{1-1/(c+1)}$. □

Corollary 21.9.16. *Given $A \in \mathbb{R}^{n\times n\times n}$, unless* ETH *fails, for any ϵ for which $\frac{1}{2} \geq \epsilon \geq c/n^2$ where c is any constant, there is no $2^{o(\epsilon^{-1/4})}$ time algorithm to compute $u', v', w' \in \mathbb{R}^n$ such that*

$$\|A - u'\otimes v'\otimes w'\|_F^2 \leq (1+\epsilon) \min_{u,v,w\in\mathbb{R}^n} \|A - u\otimes v\otimes w\|_F^2.$$

Proof. If $\epsilon = \Omega(1/n^2)$, it means that $n = \Omega(1/\sqrt{\epsilon})$. Then, we can construct a hard instance B with size $m\times m\times m$ where $m = \Theta(1/\sqrt{\epsilon})$, and we can put B into A, and let A have zero entries elsewhere. Since B is hard, i.e., there is no $2^{o(m^{-1/2})} = 2^{o(\epsilon^{-1/4})}$ running time to compute a rank-1 approximation to B, this means there is no $2^{o(\epsilon^{-1/4})}$ running time algorithm to find an approximate rank-1 approximation to A. □

Corollary 21.9.17. *Unless* ETH *fails, there is no polynomial time algorithm to approximate the best rank-1 approximation of an n-dimensional tensor within $(1+\Theta(1/\log^{2+\gamma}(n)))$ relative-error, where $\gamma > 0$ is an arbitrarily small constant.*

Proof. We can apply a padding argument here. According to Theorem 21.9.15, there is a d-dimensional tensor such that there is no $2^{o(\sqrt{d})}$ time algorithm which can give a $(1+\Theta(1/d^4))$ relative approximation. Then, if we pad with 0s everywhere to extend the size of the tensor to $n = 2^{d^{(1-\gamma')/2}}$ where $\gamma' > 0$ is a sufficiently small constant, then $\text{poly}(n) = 2^{o(\sqrt{d})}$, and $d^4 = \log^{2+O(\gamma')}(n)$. Thus, it means that there is no polynomial time algorithm which can output a $(1 + 1/(\log^{2+\gamma}))$-relative error approximation to the tensor which has size n. □

21.9.4 Tensor rank is hard to approximate

This section presents the hardness result for approximating tensor rank under ETH . According to our new result, we notice that not only deciding the tensor rank is a hard problem, but also approximating the tensor rank is a hard problem. This therefore strengthens Håstad's NP-Hadness [Hås90] for computing tensor rank.

21.9.4.1 Cover number

Before getting into the details of the reduction, we provide a definition of an important concept called the "cover number" and discuss the cover number for the MAX-E3SAT(B) problem.

Definition 21.9.7 (Cover number)**.** For any 3SAT instance S with n variables and m clauses, we are allowed to assign one of three values $\{0, 1, *\}$ to each variable. For each clause, if one of the literals outputs true, then the clause outputs true. For each clause, if the corresponding variable of one of the literals is assigned to $*$, then the clause outputs true. We say $y \in \{0,1\}^n$ is a string, and $z \in \{0, 1, *\}^n$ is a star string. For an instance S, if there exists a string $y \in \{0,1\}^n$ that causes all the clauses to be true, then we say that S is satisfiable, otherwise it is unsatisfiable. For an instance S, let Z_S denote the set of star strings which cause all of the clauses of S to be true. For each star string $z \in \{0, 1, *\}^n$, let star(z) denote the number of $*$s in the star-string z. We define the "cover number" of instance S to be

$$\text{cover-number}(S) = \min_{z \in Z_S} \text{star}(z).$$

Notice that for a satisfiable 3SAT instance S, the cover number p is 0. Also, for any

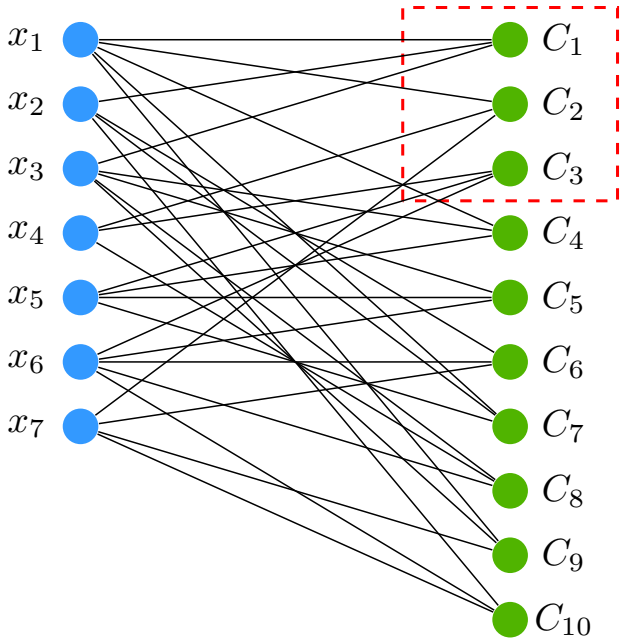


Figure 21.10: Cover number. For a 3SAT instance with n variables and m clauses, we can draw a bipartite graph which has n nodes on the left and m nodes on the right. Each node (blue) on the left corresponds to a variable x_i, each node (green) on the right corresponds to a clause C_j. If either x_i or $\overline{x}_i$ belongs to clause C_j, then we draw a line between these two nodes. Consider an input string $y \in \{0,1\}^7$. There exists some unsatisfied clauses with respect to this input string y. For for example, let C_1, C_2 and C_3 denote those unsatisfied clauses. We want to pick a smallest set of nodes on the left partition of the graph to guarantee that for each unsatisfied clause in the right partition, there exists a node on the left to cover it. The cover number is defined to be the smallest such number over all possible input strings.

unsatisfiable 3SAT instance S, the cover number p is at least 1. This is because for any input string, there exists at least one clause which cannot be satisfied. To fix that clause, we have to assign $*$ to a variable belonging to that clause. (Assigning $*$ to a variable can be regarded as assigning both 0 and 1 to a variable)

Lemma 21.9.18. *Let S denote a* MAX-E3SAT(B) *instance with n variables and m clauses and S suppose S is at most $7/8 + A$ satisfiable, where $A \in (0, 1/8)$. Then the cover number*

of S is at least $(1/8 - A)m/B$.

Proof. For any input string $y \in \{0,1\}^n$, there exists at least $(1/8 - A)m$ clauses which are not satisfied. Since each variable appears in at most B clauses, we need to assign $*$ to at least $(1/8 - A)m/B$ variables. Thus, the cover number of S is at least $(1/8 - A)m/B$. □

We say $x_1, x_2, \cdots, x_n$ are variables and $x_1, \overline{x}_1, x_2, \overline{x}_2, \cdots, x_n, \overline{x}_n$ are literals.

Definition 21.9.8. For a list of clauses C and a set of variables P, if for each clause, there exists at least one literal such that the corresponding variable of that literal belongs to P, then we say P covers L.

21.9.4.2 Properties of 3SAT instances

Fact 21.9.19. *For any* 3SAT *instance S with n variables and $m = \Theta(n)$ clauses, let $c > 0$ denote a constant. If S is $(1-c)m$ satisfiable, then let $y \in \{0,1\}^n$ denote a string for which S has the smallest number of unsatisfiable clauses. Let T denote the set of unsatisfiable clauses and let b denote the number of variables in T. Then $\Omega((cm)^{1/3}) \leq b \leq O(cm)$.*

Proof. Note that in S, there is no duplicate clause. Let T denote the set of unsatisfiable clauses by assigning string y to S. First, we can show that any two literals $x_i, \overline{x}_i$ cannot belong to T at the same time. If x_i and $\overline{x}_i$ belong to the same clause, then that clause must be an "always" satisfiable clause. If x_i and $\overline{x}_i$ belong to different clauses, then one of the clauses must be satisfiable. This contradicts the fact that that clause belongs to T. Thus, we can assume that literals $x_1, x_2, \cdots, x_b$ belong to T.

There are two extreme cases: one is that each clause only contains three literals and each literal appears in exactly one clause in T. Then $b = 3cm$. The other case is that each clause contains 3 literals, and each literal appears in as many clauses as possible. Then $\binom{b}{3} = cm$, which gives $b = \Theta((cm)^{1/3})$. □

Lemma 21.9.20. *For a random* 3SAT *instance, with probability* $1 - 2^{-\Omega(\log n \log\log n)}$ *there is no literal appearing in at least* $\log n$ *clauses.*

Proof. By the property of random 3SAT , for any literal x and any clause C, the probability that x appears in C is $\frac{3}{2n}$, i.e., $\Pr[x \in C] = \frac{3}{2n} = \Theta(1/n)$. Let p denote this probability. For any literal x, the probability of x appearing in at least $\log n$ clauses (out of m clauses) is

$$
\begin{aligned}
& \Pr[\ x \text{ appearing in} \geq \log n \text{ clauses }] \\
= & \sum_{i=\log n}^{m} \binom{m}{i} p^i (1-p)^{m-i} \\
= & \sum_{i=\log n}^{m/2} \binom{m}{i} p^i (1-p)^{m-i} + \sum_{i=m/2}^{m} \binom{m}{i} p^i (1-p)^{m-i} \\
\leq & \sum_{i=\log n}^{m/2} (em/i)^i p^i + \sum_{i=m/2}^{m} \binom{m}{i} p^i \qquad \text{by } (1-p) \leq 1, \binom{m}{i} \leq (em/i)^i \\
\leq & (\Theta(1/\log n))^{\log n} + 2 \cdot (2e)^{m/2} \cdot \Theta(1/n)^{m/2} \\
\leq & 2^{-\Omega(\log n \cdot \log\log n)}.
\end{aligned}
$$

Taking a union bound over all the literals, we complete the proof,

$$\Pr[\ \nexists\ x \text{ appearing in} \geq \log n \text{ clauses }] \geq 1 - 2^{-\Omega(\log n \log\log n)}.$$

□

Lemma 21.9.21. *For a sufficiently large constant $c' > 0$ and a constant $c > 0$, for any random* 3SAT *instance which has n variables and $m = c'n$ clauses, suppose it is $(1-c)m$ satisfiable. Then with probability $1 - 2^{-\Omega(\log n \log\log n)}$, for all input strings y, among the unsatisfied clauses, each literal appears in $O(\log n)$ places.*

Proof. This follows by Lemma 21.9.20. □

Next, we show how to reduce the $O(\log n)$ to $O(1)$.

Lemma 21.9.22. *For a sufficiently large constant c, for any random* 3SAT *instance that has n variables and $m = cn$ clauses, for any constant $B \geq 1, b \in (0,1)$, with probability at least $1 - \frac{9m}{Bbn}$, there exist at least $(1-b)m$ clauses such that each variable (in these $(1-b)m$ clauses) only appears in at most B clauses (out of these $(1-b)m$ clauses).*

Proof. For each $i \in [m]$, we use z_i to denote the indicator variable such that it is 1, if for each variable in the ith clause, it appears in at most a clauses. Let $B \in [1, \infty)$ denote a sufficiently large constant, which we will decide upon later.

For each variable x, the probability of it appearing in the i-th clause is $\frac{3}{n}$. Then we have

$$\mathbb{E}[\ \#\ \text{clauses that contain } x] = \sum_{i=1}^{m} \mathbb{E}[i\text{-th clause contains } x] = \frac{3m}{n}$$

By Markov's inequality,

$$\Pr[\ \#\ \text{clauses that contain } x \geq a] \leq \mathbb{E}[\ \#\ \text{clauses that contain } x]/B = \frac{3m}{Bn}$$

By a union bound, we can compute $\mathbb{E}[z_i]$,

$$\begin{aligned}\mathbb{E}[z_i] = &\Pr[z_i = 1] \\ &\geq 1 - 3\Pr[\text{ one variable in } i\text{-th clause appearing} \geq B \text{ clauses }] \\ &\geq 1 - \frac{9m}{Bn}.\end{aligned}$$

Furthermore, we have

$$\mathbb{E}[z] = \mathbb{E}[\sum_{i=1}^{m} z_i] = \sum_{i=1}^{m} \mathbb{E}[z_i] \geq (1 - \frac{9m}{Bn})m.$$

Note that $z \leq m$. Thus $\mathbb{E}[z] \leq m$. Let $b \in (0, 1)$ denote a sufficiently small constant. We can show

$$\begin{aligned}\Pr[m - z \geq bm] &\leq \frac{\mathbb{E}[m - z]}{bm} \\ &= \frac{m - \mathbb{E}[z]}{bm} \\ &\leq \frac{m - (1 - \frac{9m}{Bn})m}{bm} \\ &= \frac{9m}{Bbn}.\end{aligned}$$

This implies that with probability at least $1 - \frac{9m}{Bbn}$, we have $m - z \leq bm$. Notice that in random-ETH , $m = cn$ for a constant c. Thus, by choosing a sufficiently large constant B (which is a function of c, b), we can obtain arbitrarily large constant success probability. □

21.9.4.3 Reduction

We reduce 3SAT to tensor rank by following the same construction in [Hås90]. To obtain a stronger hardness result, we use the property that each variable only appears in at most B (some constant) clauses and that the cover number of an unsatisfiable 3SAT instance

is large. Note that both MAX-E3SAT(B) instances and random-ETH instances have that property. Also each MAX-E3SAT(B) is also a 3SAT instance. Thus if the reduction holds for 3SAT , it also holds for MAX-E3SAT(B) , and similarly for random-ETH .

Recall the definition of 3SAT : 3SAT is the problem of given a Boolean formula of n variables in CNF form with at most 3 variables in each of the m clauses, is it possible to find a satisfying assignment to the formula? We say $x_1, x_2, \cdots, x_n$ are variables and $x_1, \overline{x}_1, x_2, \overline{x}_2, \cdots, x_n, \overline{x}_n$ are literals. We transform this to the problem of computing the rank of a tensor of size $n_1 \times n_2 \times n_3$ where $n_1 = 2 + n + 2m$, $n_2 = 3n$ and $n_3 = 3n + m$. T has the following n_3 column-row faces, where each of the faces is an $m_1 \times n_2$ matrix,

- n variable matrices $V_i \in \mathbb{R}^{n_1 \times n_2}$. It has a 1 in positions $(1, 2i-1)$ and $(2, 2i)$ while all other elements are 0.
- n help matrices $S_i \in \mathbb{R}^{n_1 \times n_2}$. It has a 1 position in $(1, 2n+i)$ and is 0 otherwise.
- n help matrices $M_i \in \mathbb{R}^{n_1 \times n_2}$. It has a 1 in positions $(1, 2i-1), (2+i, 2i)$ and $(2+i, 2n+i)$ and is 0 otherwise.
- m clause matrices $C_l \in \mathbb{R}^{n_1 \times n_2}$. Suppose the clause c_l contains the literals $u_{l,1}, u_{l,2}$ and $u_{l,3}$. For each $j \in [3]$, $u_{l,j} \in \{x_1, x_2, \cdots, x_n, \overline{x}_1, \overline{x}_2, \cdots, \overline{x}_n\}$. Note that $x_i, \overline{x}_i$ are the literals of the 3SAT formula. We can also think of $x_i, \overline{x}_i$ as length $3n$ vectors. Let x_i denote the vector that has a 1 in position $2i-1$, i.e., $x_i = e_{2i-1}$. Let $\overline{x}_i$ denote the vector that has a 1 in positions $2i-1$ and $2i$, $\overline{x}_i = e_{2i-1} + e_{2i}$.
  - Row 1 is the vector $u_{l,1} \in \mathbb{R}^{3n}$,
  - Row $2 + n + 2l - 1$ is the vector $u_{l,1} - u_{l,2} \in \mathbb{R}^{3n}$,

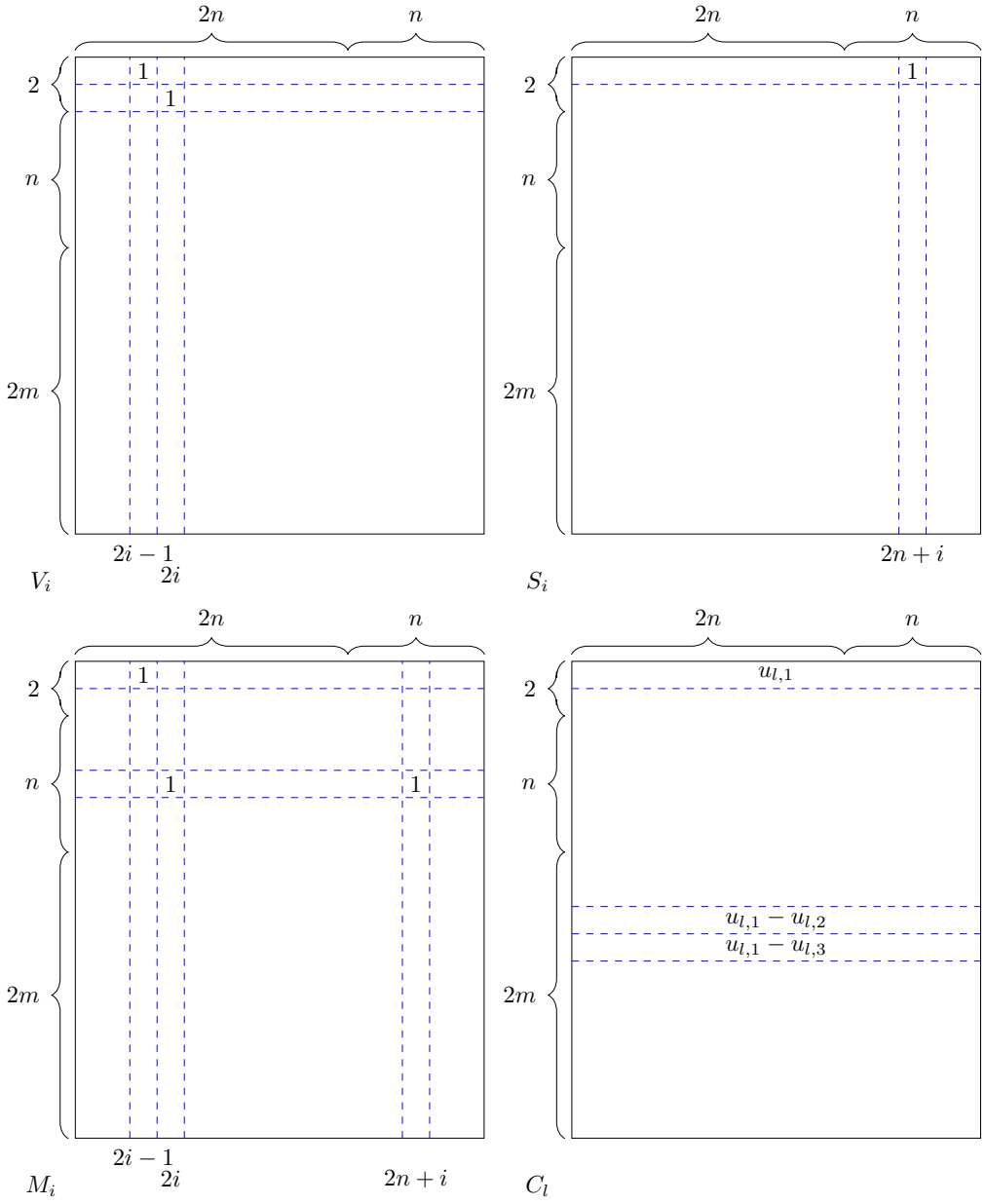


Figure 21.11: There are $3n + m$ column-row faces, $V_i, \forall i \in [n]$, $S_i, \forall i \in [n]$, $M_i, \forall i \in [n]$, $C_l, \forall l \in [m]$. In face C_l, each $u_{l,j}$ is either x_i or $\overline{x}_i$ where $x_i = e_{2i-1}$ and $\overline{x}_i = e_{2i-1} + e_{2i}$.

– Row $2 + n + 2l$ is the vector $u_{l,1} - u_{l,3} \in \mathbb{R}^{3n}$.

First, we can obtain Lemma 21.9.23 which follows by Lemma 2 in [Hås90]. For completeness, we provide a proof.

Lemma 21.9.23. *If the formula is satisfiable, then the constructed tensor has rank at most* $4n + 2m$.

Proof. We will construct $4n+2m$ rank-1 matrices $V_i^{(1)}, V_i^{(2)}, S_i^{(1)}, M_i^{(1)}, C_l^{(1)}$ and $C_l^{(2)}$. Then the goal is to show that for each matrix in the set

$$\{V_1, V_2, \cdots, V_n, S_1, S_2, \cdots, S_n, M_1, M_2, \cdots, M_n, C_1, C_2, \cdots, C_m\},$$

it can be written as a linear combination of these constructed matrices.

- Matrices $V_i^{(1)}$ and $V_i^{(2)}$. $V_i^{(1)}$ has the first row equal to x_i iff $\alpha_i = 1$ and otherwise $\overline{x}_i$. All the other rows are 0. We set $V_i^{(2)} = V_i - V_i^{(1)}$.
- Matrices $S_i^{(1)}$. $S_i^{(1)} = S_i$.
- Matrices $M_i^{(1)}$.

$$M_i^{(1)} = \begin{cases} M_i - V_i^{(1)} & \text{if } \alpha_i = 1 \\ M_i - V_i^{(1)} - S_i & \text{if } \alpha_i = 0 \end{cases}$$

- Matrices $C_l^{(1)}$ and $C_l^{(2)}$. Let $x_i = \alpha_i$ be the assignment that makes the clause c_l true. Then $C_l - V_i^{(1)}$ has rank 2, since either it has just two nonzero rows (in the case where x_i is the first variable in the clause) or it has three nonzero rows of which two are equal. In both cases we just need two additional rank 1 matrices.

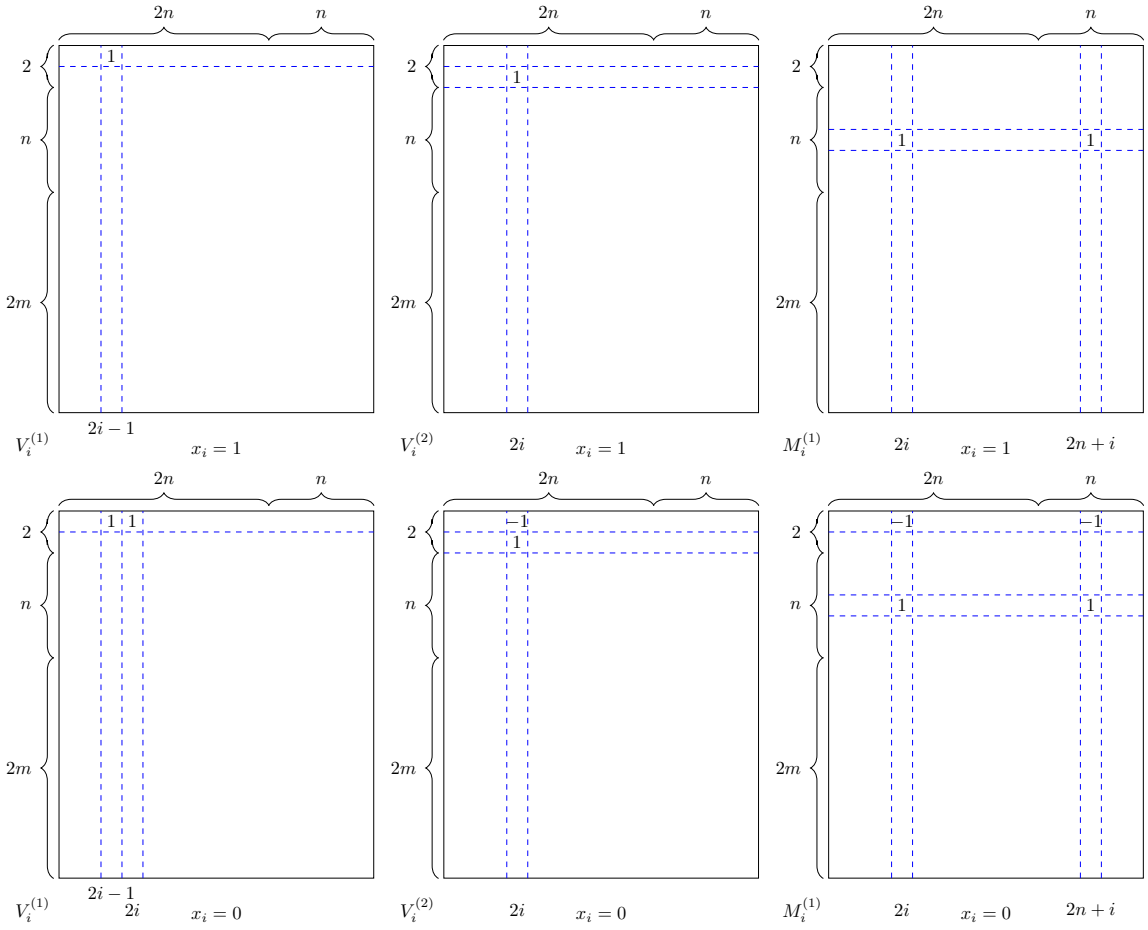


Figure 21.12: Two possibilities for $V_i^{(1)}, \forall i \in [n]$, $V^{(2)}, \forall i \in [n]$, $M_i^{(1)}, \forall i \in [n]$.

□

Once the 3SAT instance S is unsatisfiable, then its cover number is at least 1. For each unsatisfiable 3SAT instance S with cover number p, we can show that the constructed tensor has rank at most $4n + 2m + O(p)$ and also has rank at least $4n + 2m + \Omega(p)$. We first prove an upper bound,

Lemma 21.9.24. *For a* 3SAT *instance* S*, let* $y \in \{0, 1\}$ *denote a string such that* $S(y)$ *has*

a set L that contains unsatisfiable clauses. Let p denote the smallest number of variables that cover all clauses in L. Then the constructed tensor T has rank at most $4n + 2m + p$.

Proof. Let y denote a length-n Boolean string $(\alpha_1, \alpha_2, \cdots, \alpha_n)$. Based on the assignment y, all the clauses of S can be split into two sets: L contains all the unsatisfied clauses and $\overline{L}$ contains all the satisfied clauses. We use set P to denote a set of variables that covers all the clauses in set L. Let $p = |P|$. We will construct $4n + 2m + p$ rank-1 matrices $V_i^{(1)}, V_i^{(2)}$, $S_i^{(1)}$, $M_i^{(1)}$, $\forall i \in [n]$, $C_l^{(1)}$, $C_l^{(2)}$, $\forall l \in [m]$, and $V_j^{(3)}$, $\forall j \in P$. Then the goal is to show that the V_i, S_i, M_i and C_l can be written as linear combinations of these constructed matrices.

- Matrices $V_i^{(1)}$ and $V_i^{(2)}$. $V_i^{(1)}$ has first row equal to x_i iff $\alpha_i = 1$ and otherwise $\overline{x}_i$. All the other rows are 0. We set $V_i^{(2)} = V_i - V_i^{(1)}$.

- Matrices $V_j^{(3)}$. For each $j \in P$, $V_j^{(3)}$ has the first row equal to x_i iff $\alpha_i = 0$ and otherwise $\overline{x}_i$.

- Matrices $S_i^{(1)}$. $S_i^{(1)} = S_i$.

- Matrices $M_i^{(1)}$.

$$M_i^{(1)} = \begin{cases} M_i - V_i^{(1)} & \text{if } \alpha_i = 1 \\ M_i - V_i^{(1)} - S_i & \text{if } \alpha_i = 0 \end{cases}$$

- Matrices $C_l^{(1)}$ and $C_l^{(2)}$.

  - For each $l \notin L$, clause c_l is satisfied according to assignment y. Let $x_i = \alpha_i$ be the assignment that makes the clause c_l true. Then $C_l - V_i^{(1)}$ has rank 2, since either it has just two nonzero rows (in the case where x_i is the first variables in

the clause) or it has three nonzero rows of which two are equal. In both cases we just need two additional rank 1 matrices.

– For each $l \in L$. It means clause c_l is unsatisfied according to assignment y. Let $x_{j_1} = \alpha_{j_1}, x_{j_2} = \alpha_{j_2}, x_{j_3} = \alpha_{j_3}$ be an assignment that makes the clause c_l false. In other words, one of j_1, j_2, j_3 must be P according to the definition that P covers L. Then matrix $C_l - V_{j_1}^{(3)}$ has rank 2, since either it has just two nonzero rows (in the case where x_{j_1} is the first variables in the clause) or it has three nonzero rows of which two are equal. In both cases we just need two additional rank 1 matrices.

We finish the proof by taking the P that has the smallest size. □

Further, we have:

Corollary 21.9.25. *For a* 3SAT *instance S, let p denote the cover number of S, then the constructed tensor T has rank at most $4n + 2m + p$.*

Proof. This follows by applying Lemma 21.9.24 to all the input strings and the definition of cover number (Definition 21.9.7). □

We can split the tensor $T \in \mathbb{R}^{(2+n+3m)\times 3n\times(3n+m)}$ into two sub-tensors, one is $T_1 \in \mathbb{R}^{2\times 3n\times(3n+m)}$ (that contains the first two row-tube faces of T and linear combination of the remaining $2m$ row-tube faces of T), and the other is $T_2 \in \mathbb{R}^{(n+2m)\times 3n\times(3n+m)}$ (that contains the next $n + 2m$ row-tube faces of T). We first analyze the rank of T_1 and then analyze the rank of T_2.

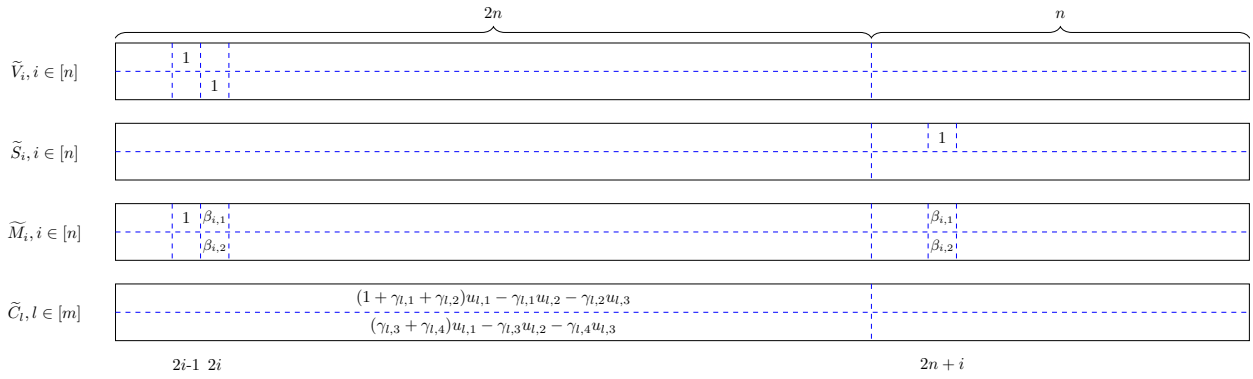


Figure 21.13: $\widetilde{V}_i$,$\widetilde{S}_i$,$\widetilde{M}_i$,$\widetilde{C}_l$.

Claim 21.9.26. *The rank of T_2 is $n + 2m$.*

Proof. According to Figure 21.11, the nonzero rows are distributed in $n + m$ fully separated sub-tensors. It is obvious that the rank of each one of those n sub-tensors is 1, and the rank of each of those m sub-tensors is 2. Thus, overall, the rank T_2 is $n + 2m$. □

To make sure $\text{rank}(T) = \text{rank}(T_1) + \text{rank}(T_2)$, the $T_1 \in \mathbb{R}^{2\times 3n\times(3n+m)}$ can be described as the following $3n + m$ column-row faces, and each of the faces is a $2 \times 3n$ matrix.

- Matrices $\widetilde{V}_i, \forall i \in [n]$. The two rows are from the first two rows of V_i in Figure 21.11, i.e., the first row is e_{2i-1} and the second row is e_{2i}.
- Matrices $\widetilde{S}_i, \forall i \in [n]$. The two rows are from the first two rows of S_i in Figure 21.11, i.e., the first row is e_{2n+i} and the second row is zero everywhere else.
- Matrices $\widetilde{M}_i, \forall i \in [n]$. The first row is $e_{2i-1} + \beta_{i,1}(e_{2i} + e_{2n+i})$, while the second row is $\beta_{i,2}(e_{2i} + e_{2n+i})$.

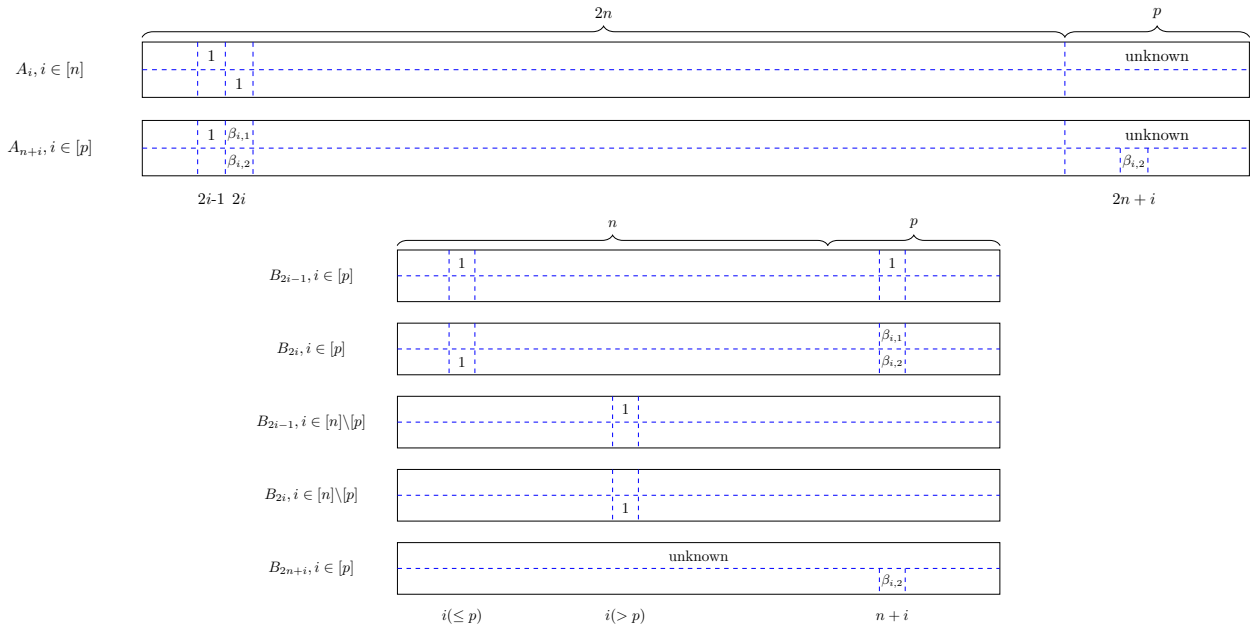


Figure 21.14: There are $n+p$ matrices $A_i \in \mathbb{R}^{2\times(2n+p)}, \forall i \in [n+p]$ and $2n+p$ matrices $B_i \in \mathbb{R}^{2\times(n+p)}, \forall i \in [2n+p]$. Tensor A and tensor B represet the same tensor, and for each $i \in [n+p], j \in [2], l \in [2n+p]$, $(A_i)_{j,l} = (B_l)_{j,i}$.

- Matrices $\widetilde{C}_l, \forall i \in [m]$. The first row is $(1+\gamma_{l,1}+\gamma_{l,2})u_{l,1} - \gamma_{l,1}u_{l,2} - \gamma_{l,2}u_{l,3}$ and the second is $(\gamma_{l,3}+\gamma_{l,4})u_{l,1} - \gamma_{l,3}u_{l,2} - \gamma_{l,4}u_{l,3}$,

where for each $i \in [3n]$, we use vector e_i to denote a length $3n$ vector such that it only has a 1 in position i and 0 otherwise. β, γ are variables. The goal is to show a lower bound for,

$$\operatorname*{rank}_{\beta,\gamma}(T_1).$$

Lemma 21.9.27. *Let P denote the set $\{i \mid$ the second row of matrix $\widetilde{M_i}$ is nonzero, $\forall i \in [n]\}$. Then the rank of T_1 is at least $3n + |P|$.*

Proof. We define $p = |P|$. Without loss of generality, we assume that for each $i \in [p]$, the second row of matrix $\widetilde{M_i}$ is nonzero.

Notice that matrices $\widetilde{V}_i$, $\widetilde{S}_i$, $\widetilde{M}_i$ have size $2 \times 3n$, but we only focus on the first $2n+p$ columns. Thus, we have $n+p$ column-row faces (from the 3rd dimension) $A_j \in \mathbb{R}^{2\times(2n+p)}$,

- A_j, $1 \le j \le n$, A_j is the first $2n+p$ columns of $\widetilde{V}_j - \sum_{i=1}^{n} \alpha_{i,j}\widetilde{S}_i \in \mathbb{R}^{2\times 3n}$, where $\alpha_{i,j}$ are some coefficients.
- A_{n+j}, $1 \le j \le p$, A_j is the first $2n+p$ columns of $\widetilde{M}_j - \sum_{i=1}^{n} \alpha_{i,n+j}\widetilde{S}_i \in \mathbb{R}^{2\times 3n}$, where $\alpha_{i,j}$ are some coefficients.

Consider the first $2n+p$ column-tube faces (from 2nd dimension), B_j, $\forall j \in [2n+p]$, of T_1. Notice that these matrices have size $2 \times (n+p)$.

- B_{2i-1}, $1 \le i \le p$, it has a 1 in positions $(1,i)$ and $(1,n+i)$.
- B_{2i}, $1 \le i \le p$, it has $\beta_{i,1}$ in position $(1,n+i)$, 1 in position $(2,i)$ and $\beta_{i,2}$ in position $(2,n+i)$.
- B_{2i-1}, $p+1 \le i \le n$, it has 1 in position $(1,i)$.
- B_{2i}, $p+1 \le i \le n$, it has 1 in position $(2,i)$.
- B_{2n+i}, $1 \le i \le p$, the first row is unknown, the second row has $\beta_{i,2}$ in position in $(2,n+i)$.

It is obvious that the first $2n$ matrices are linearly independent, thus the rank is at least $2n$. We choose the first $2n$ matrices as our basis. For B_{2n+1}, we try to write it as a linear combination of the first $2n$ matrices $\{B_i\}_{i\in[2n]}$. Consider the second row of B_{2n+1}. The first n positions are all 0. The matrices B_{2i} all have disjoint support for the second row of the

first n columns. Thus, the matrices B_{2i} should not be used. Consider the second row of $B_{2i-1}, \forall i \in [n]$. None of them has a nonzero value in position $n+1$. Thus B_{2n+1} cannot be written as a linear combination of of the first $2n$ matrices. Thus, we can show for any $i \in [p]$, B_{2n+i} cannot be written as a linear combination of matrices $\{B_i\}_{i\in[2n]}$. Consider the p matrices $\{B_{2n+i}\}_{i\in[p]}$. Each of them has a different nonzero position in the second row. Thus these matrices are all linearly independent. Putting it all together, we know that the rank of matrices $\{B_i\}_{i\in[2n+p]}$ is at least $2n+p$. □

Next, we consider another special case when $\beta_{i,2} = 0$, for all $i \in [n]$. If we subtract $\beta_{i,1}$ times $\widetilde{S}_i$ from $\widetilde{M}_i$ and leave the other column-row faces (from the 3rd dimension) as they are, and we make all column-tube faces(from the 2nd dimension) for $j > 2n$ identically 0, then all other choices do not change the first $2n$ column-tube faces (from the 2nd dimension) and make some other column-tube faces (from the 2nd dimension) nonzero. Such a choice could clearly only increase the rank of T. Thus, we obtain,

$$\operatorname{rank}(T) = 2n + 2m + \min \operatorname{rank}(T_3),$$

where T_3 is a tensor of size $2 \times 2n \times (2n+m)$ given by the following column-row faces (from 3rd dimension) $A_i, \forall i \in [2n+m]$ and each matrix has size $2 \times 2n$ (shown in Figure 21.15).

- A_i, $i \in [n]$, the first $2n$ columns of $\widetilde{V}_i$.
- A_{n+i}, $i \in [n]$, the first $2n$ columns of $\widetilde{M}_i$. The first row is $e_{2i-1} + \beta_{i,1} e_{2i}$, and the second row is 0.
- A_{2n+l}, $l \in [m]$, the first $2n$ columns of $\widetilde{C}_l$. The first row is $(1+\gamma_{l,1}+\gamma_{l,2})u_{l,1} - \gamma_{l,1}u_{l,2} - \gamma_{l,2}u_{l,3}$, and the second row is $(\gamma_{l,3}+\gamma_{l,4})u_{l,1} - \gamma_{l,3}u_{l,2} - \gamma_{l,4}u_{l,3}$.

We can show

Lemma 21.9.28. *Let p denote the cover number of the* 3SAT *instance. T_3 has rank at least $2n + \Omega(p)$.*

Proof. First, we can show that all matrices $A_{n+i} - A_i$ and A_{n+i} (for all $i \in [n]$) are in the expansion of tensor T_3. Thus, the rank of T_3 is at least $2n$.

We need the following claim:

Claim 21.9.29. *For any $l \in [m]$, if A_{2n+l} can be written as a linear combination of $\{A_{n+i} - A_i\}_{i\in[n]}$ and $\{A_{n+i}\}_{i\in[n]}$, then the second row of A_{2n+l} is 0, and the first row of one of the A_{n+i} is u_i where u_i is one of the literals appearing in clause c_l.*

Proof. We prove this for the second row first. For each $l \in [m]$, we consider the possibility of using all matrices $A_{n+i} - A_i$ and A_{n+i} to express matrix A_{2n+l}. If the second row of A_{2n+l} is nonzero, then it must have a nonzero entry in an odd position. But there is no nonzero in an odd position of the second row of any of matrices $A_{n+i} - A_i$ and A_{n+i}.

For the first row. It is obvious that the first row of A_{2n+l} must have at least one nonzero position, for any $\gamma_{l,1}, \gamma_{l,2}$. Let u_j be a literal belonging to the variable x_i which appears in the first row of A_{2n+l} with a nonzero coefficient. Since only A_{n+i} of all the other $A_{n+s}, \forall s \in [n]$ matrices has nonzero elements in either of the positions $(1, 2i-1)$ or $(1, 2i)$, then A_{n+i} must be used to cancel these elements. Thus, the first row of A_{n+i} must be a multiple of u_j and since the element in position $(1, 2i-1)$ of A_{n+i} is 1, this multiple must be 1.

□

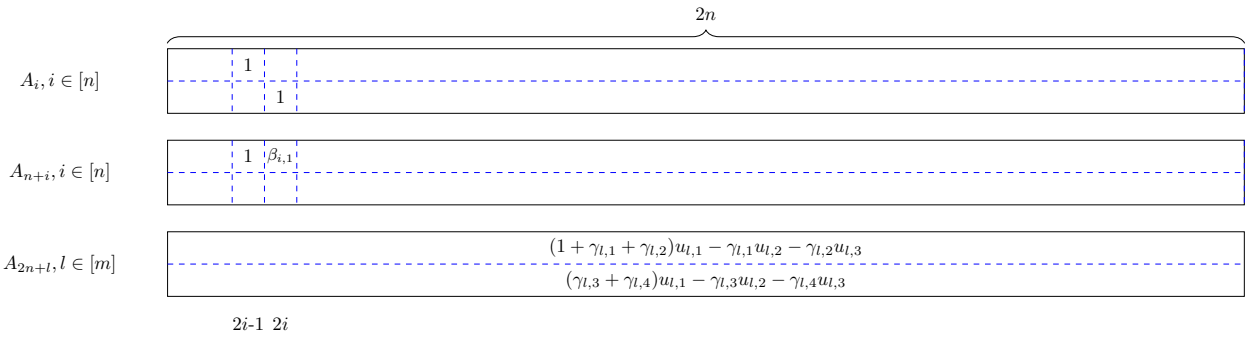


Figure 21.15: For any $i \in [n]$, $\beta_{i,1} \in \mathbb{R}$, for any $l \in [m]$, $\gamma_{l,1}, \gamma_{l,2} \in \mathbb{R}$, for any $l \in [m]$, if the first literal of clause l is x_j, then row vector $u_{l,1} = e_{2i-1} \in \mathbb{R}^{2n}$; if the first literal of clause l is $\overline{x}_j$, then row vector $u_{l,1} = e_{2i-1} + e_{2i} \in \mathbb{R}^{2n}$.

Note that matrices $A_i, \forall i \in [n]$ have the property that, for any matrix in $\{A_{n+1}, \cdots, A_{2n+m}\}$, it cannot be written as the linear combination of matrices $A_i, \forall i \in [n]$. Let $\widetilde{A} \in \mathbb{R}^{(n+m)\times 2n}$ denote a matrix that consists of the first rows of $\{A_{n+1}, \cdots, A_{2n+m}\}$. According to the property of matrices $A_i, \forall i \in [n]$, and that the rank of a tensor is always greater than or equal to the rank of any sub-tensor, we know that

$$\operatorname{rank}(T_3) \geq n + \min \operatorname{rank}(\widetilde{A}).$$

Claim 21.9.30. *For a* 3SAT *instance S, for any input string $y \in \{0,1\}^n$, set $\beta_{*,1}$ to be the entry-wise flipping of y,* (I) *if the clause l is satisfied, then the $(n+l)$-th row of $\widetilde{A} \in \mathbb{R}^{(n+m)\times 2n}$ can be written as a linear combination of the first n rows of $\widetilde{A}$.* (II) *if the clause l is unsatisfied, then the $(n+l)$-th row of $\widetilde{A}$ cannot be written as a linear combination of the first n rows of $\widetilde{A}$.*

Proof. Part (I), consider a clause l which is satisfied with input string y. Then there must exist a variable x_i belonging to clause l (either literal x_i or literal $\overline{x}_i$) and one of the following

holds: if x_i belongs to clause l, then $\alpha_i = 1$; if $\overline{x}_i$ belongs to clause l, then $\alpha_i = 0$. Suppose clause l contains literal x_i. The other case can be proved in a similar way. We consider the $(n+l)$-th row. One of the following assignments $(0,0), (-1,0), (0,-1)$ to $\gamma_{l,1}, \gamma_{l,1}$ is going to set the $(n+l)$-th row of $\widetilde{A}$ to be vector e_{2i-1}. We consider the i-th row of $\widetilde{A}$. Since we set $\alpha_i = 1$, then we set $\beta_{i,1} = 0$, it follows that the i-th row of A becomes e_{2i-1}. Therefore, the $(n+l)$-th row of $\widetilde{A}$ can be written as a linear combination of $\widetilde{A}$.

Part (II), consider a clause l which is unsatisfied with input string y. Suppose that clause contains three literals $x_{i_1}, x_{i_2}, x_{i_3}$ (the other seven possibilities can be proved in a similar way). Then for input string y, we have $\alpha_{i_1} = 0$, $\alpha_{i_2} = 0$ and $\alpha_{i_3} = 0$, otherwise this clause l is satisfied. Consider i_1-th row of $\widetilde{A}$. It becomes $e_{2i_1-1} + e_{2i_1}$. Similarly for the i_2-th row and i_3-th row. Consider the $(n+l)$-th row. We can observe that all of positions $2i_1, 2i_2, 2i_3$ must be 0. Any linear combination formed by the i_1, i_2, i_3-th row of $\widetilde{A}$ must have one nonzero in one of positions $2i_1, 2i_2, 2i_3$. However, if we consider the $(n+l)$-th row of $\widetilde{A}$, one of the positions $2i_1, 2i_2, 2i_3$ must be 0. Also, the remaining $n-3$ of the first n rows of $\widetilde{A}$ also have 0 in positions $2i_1, 2i_2, 2i_3$. Thus, we can show that the $(n+l)$-th row of $\widetilde{A}$ cannot be written as a linear combination of the first n rows. Similarly, for the other seven cases. □

Note that in order to make sure as many as possible rows in $n+1, \cdots, n+m$ can be written as linear combinations of the first n rows of $\widetilde{A}$, the $\beta_{i,1}$ should be set to either 0 or 1. Also each possibility of input string y is corresponding to a choice of $\beta_{i,1}$. According to the above Claim 21.9.30, let l_0 denote the smallest number of unsatisfied clauses over the choices of all the 2^n input strings. Then over all choices of β, γ, there must exist at least l_0 rows of

$\widetilde{A}_{n+1}, \cdots \widetilde{A}_{n+m}$, such that each of those rows cannot be written as the linear combination of the first n rows.

Claim 21.9.31. *Let $\widetilde{A} \in \mathbb{R}^{(n+m)\times 2n}$ denote a matrix that consists of the first rows of $A_{n+i}, \forall i \in [n]$ and $A_{n+l}, \forall l \in [m]$. Let p denote the cover number of* 3SAT *instance. Then* $\min \operatorname{rank}(\widetilde{A}) \geq n + \Omega(p)$.

Proof. For any choices of $\{\beta_{i,1}\}_{i\in[n]}$, there must exist a set of rows out of the next m rows such that, each of those rows cannot be written as a linear combination of the first n rows. Let L denote the set of those rows. Let t denote the maximum size set of disjoint rows from L. Since those t rows in L all have disjoint support, they are always linearly independent. Thus the rank is at least $n + t$.

Note that each row corresponds to a unique clause and each clause corresponds to a unique row. We can just pick an arbitrary clause l in L, then remove the clauses that are using the same literal as clause l from L. Because each variable occurs in at most B clauses, we only need to remove at most $3B$ clauses from L. We repeat the procedure until there is no clause L. The corresponding rows of all the clauses we picked have disjoint supports, thus we can show a lower bound for t,

$$t \geq |L|/(3B) \geq l_0/(3B) \geq p/(9B) \gtrsim p,$$

where the second step follows by $|L| \geq l_0$, the third step follows $3l_0 \geq p$, and the last step follows by B is some constant. □

Thus, putting it all together, we complete the proof.

□

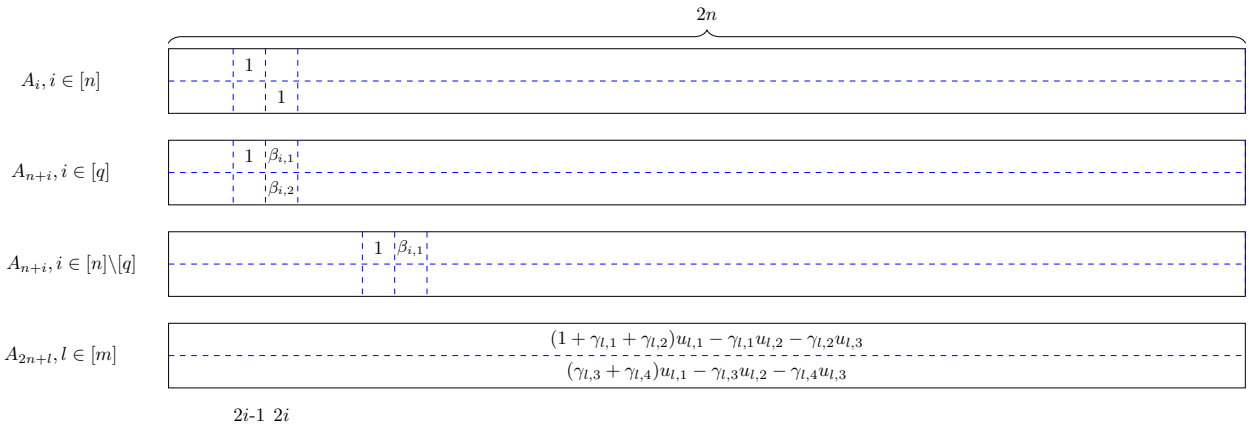


Figure 21.16: For any $i \in [n]$, $\beta_{i,1} \in \mathbb{R}$. For any $i \in [q]$, $\beta_{i,2} \in \mathbb{R}$. For any $l \in [m]$, $\gamma_{l,1}, \gamma_{l,2} \in \mathbb{R}$. For any $l \in [m]$, if the first literal of clause l is x_j, then row vector $u_{l,1} = e_{2i-1} \in \mathbb{R}^{2n}$; if the first literal of clause l is $\overline{x}_j$, then row vector $u_{l,1} = e_{2i-1} + e_{2i} \in \mathbb{R}^{2n}$.

Now, we consider a general case when there are q different $i \in [n]$ satisfying that $\beta_{i,2} \neq 0$. Similar to tensor T_3, we can obtain T_4 such that,

$$\operatorname{rank}(T) = 2n + 2m + \min \operatorname{rank}(T_4)$$

where T_4 is a tensor of size $2 \times 2n \times (2n+m)$ given by the following column-row faces (from 3rd dimension) A_i, $\forall i \in [2n+m]$ and each matrix has size $2 \times 2n$ (shown in Figure 21.16).

- A_i, $i \in [n]$, the first $2n$ columns of $\widetilde{V}_i$.
- A_{n+i}, $i \in [q]$, the first $2n$ columns of $\widetilde{M}_i$. The first row is $e_{2i-1} + \beta_{i,1} e_{2i}$, and the second row is $\beta_{i,2} e_{2i}$.
- A_{n+i}, $i \in \{q+1, \cdots, n\}$, the first $2n$ columns of $\widetilde{M}_i$. The first row is $e_{2i-1} + \beta_{i,1} e_{2i}$, and the second row is 0.

- A_{2n+l}, $l \in [m]$, the first $2n$ columns of $\widetilde{C}_l$. The first row is $(1+\gamma_{l,1}+\gamma_{l,2})u_{l,1} - \gamma_{l,1}u_{l,2} - \gamma_{l,2}u_{l,3}$, and the second row is $(\gamma_{l,3}+\gamma_{l,4})u_{l,1} - \gamma_{l,3}u_{l,2} - \gamma_{l,4}u_{l,3}$.

Note that modifying q entries(from Figure 21.15 to Figure 21.16) of a tensor can only decrease the rank by q, thus we obtain

Lemma 21.9.32. *Let q denote the number of i such that $\beta_{i,2} \neq 0$, and let p denote the cover number of the* 3SAT *instance. Then T_4 has rank at least $2n + \Omega(p) - q$.*

Combining the two perspectives we have

Lemma 21.9.33. *Let p denote the cover number of an unsatisfiable* 3SAT *instance. Then the tensor has rank at least $4n + 2m + \Omega(p)$.*

Proof. Let q denote the q in Figure 21.16. From one perspective, we know that the tensor has rank at least $4n + 2m + \Omega(p) - q$. From another perspective, we know that the tensor has rank at least $4n + 2m + q$. Combining them together, we obtain the rank is at least $4n + 2m + \Omega(p)/2$, which is still $4n + 2m + \Omega(p)$. □

Theorem 21.9.34. *Unless* ETH *fails, there is a $\delta > 0$ and an absolute constant $c_0 > 1$ such that the following holds. For the problem of deciding if the rank of a q-th order tensor, $q \geq 3$, with each dimension n, is at most k or at least $c_0 k$, there is no $2^{\delta k^{1-o(1)}}$ time algorithm.*

Proof. The reduction can be split into three parts.[13] The first part reduces the MAX-3SAT problem to the MAX-E3SAT problem by [MR10]. For each MAX-3SAT instance with

[13]The first two parts are accomplished by personal communication with Dana Moshkovitz and Govind Ramnarayan.

size n, the corresponding MAX-E3SAT instance has size $n^{1+o(1)}$. The second part is by reducing the MAX-E3SAT problem to MAX-E3SAT(B) by [Tre01]. For each MAX-E3SAT instance with size n, the corresponding MAX-E3SAT(B) instance has size $\Theta(n)$ when B is a constant. The third part is by reducing the MAX-E3SAT(B) problem to the tensor problem. Combining Theorem 21.9.5, Lemma 21.9.18 with this reduction, we complete the proof. □

Theorem 21.9.35. *Unless random-*ETH *fails, there is an absolute constant $c_0 > 1$ for which any deterministic algorithm for deciding if the rank of a q-th order tensor is at most k or at least $c_0 k$, requires $2^{\Omega(k)}$ time.*

Proof. This follows by combining the reduction with random-ETH and Lemma 21.9.22. □

Note that, if **BPP** = **P** then it also holds for randomized algorithms which succeed with probability 2/3.

Indeed, we know that any deterministic algorithm requires $2^{\Omega(n)}$ running time on tensors that have size $n \times n \times n$. Let $g(n)$ denote a fixed function of n, and $g(n) = o(n)$. We change the original tensor from size $n \times n \times n$ to $2^{g(n)} \times 2^{g(n)} \times 2^{g(n)}$ by adding zero entries. Then the number of entries in the new tensor is $2^{3g(n)}$ and the deterministic algorithm still requires $2^{\Omega(n)}$ running time on this new tensor. Assume there is a randomized algorithm that runs in $2^{cg(n)}$ time, for some constant $c > 3$. Then considering the size of this new tensor, the deterministic algorithm is a super-polynomial time algorithm, but the randomized algorithm is a polynomial time algorithm. Thus, by assuming **BPP** = **P**, we can rule out randomized algorithms, which means Theorem 21.9.35 also holds for randomized algorithms which succeed with probability 2/3.

We provide some some motivation for the $\mathbf{BPP} = \mathbf{P}$ assumption: this is a standard conjecture in complexity theory, as it is implied by the existence of strong pseudorandom generators or if any problem in deterministic exponential time has exponential size circuits [IW97].

21.10 Extension to Other Tensor Ranks

The tensor rank studied in the previous sections is also called the CP rank or canonical rank. The tensor rank can be thought of as a direct extension of the matrix rank. We would like to point out that there are other definitions of tensor rank, e.g., the tucker rank and train rank. In this section we explain how to extend our proofs to other notions of tensor rank. Section 21.10.1 provides the extension to tucker rank, and Section 21.10.2 provides the extension to train rank.

21.10.1 Tensor Tucker rank

Tensor Tucker rank has been studied in a number of works [KC07, PC08, MH09, ZW13, YC14]. We provide the formal definition here:

21.10.1.1 Definitions

Definition 21.10.1 (Tucker rank)**.** Given a third order tensor $A \in \mathbb{R}^{n \times n \times n}$, we say A has tucker rank k if k is the smallest integer such that there exist three matrices $U, V, W \in \mathbb{R}^{n \times k}$ and a (small) tensor $C \in \mathbb{R}^{k \times k \times k}$ satisfying

$$A_{i,j,l} = \sum_{i'=1}^{k} \sum_{j'=1}^{k} \sum_{l'=1}^{k} C_{i',j',l'} U_{i,i'} V_{j,j'} W_{l,l'}, \forall i, j, l \in [n] \times [n] \times [n],$$

or equivalently,

$$A = C(U, V, W).$$

21.10.1.2 Algorithm

Theorem 21.10.1. *Given a third order tensor $A \in \mathbb{R}^{n \times n \times n}$, for any $k \geq 1$ and $\epsilon \in (0, 1)$, there exists an algorithm which takes $O(\mathrm{nnz}(A)) + n \,\mathrm{poly}(k, 1/\epsilon) + 2^{O(k^2/\epsilon + k^3)}$ time and outputs three matrices $U, V, W \in \mathbb{R}^{n \times k}$, and a tensor $C \in \mathbb{R}^{k \times k \times k}$ for which*

$$\|C(U, V, W) - A\|_F^2 \leq (1 + \epsilon) \min_{\text{tucker rank} - k \ A_k} \|A_k - A\|_F^2$$

holds with probability $9/10$.

Proof. We define OPT to be

$$\mathrm{OPT} = \min_{\text{tucker rank} - k \ A'} \|A' - A\|_F^2.$$

Algorithm 21.33 Frobenius Norm Low (Tucker) Rank Approximation

procedure FLOWTUCKERRANKAPPROX(A, n, k, ϵ) ▷ Theorem 21.10.1
  $s_1 \leftarrow s_2 \leftarrow s_3 \leftarrow O(k/\epsilon)$.
  $t_1 \leftarrow t_2 \leftarrow t_3 \leftarrow \text{poly}(k, 1/\epsilon)$.
  Choose sketching matrices $S_1 \in \mathbb{R}^{n^2 \times s_1}$, $S_2 \in \mathbb{R}^{n^2 \times s_2}$, $S_3 \in \mathbb{R}^{n^2 \times s_3}$. ▷ Definition 21.3.12
  Choose sketching matrices $T_1 \in \mathbb{R}^{t_1 \times n}$, $T_2 \in \mathbb{R}^{t_2 \times n}$, $T_3 \in \mathbb{R}^{t_3 \times n}$.
  Compute $A_i S_i, \forall i \in [3]$.
  Compute $T_i A_i S_i$, $\forall i \in [3]$.
  Compute $B \leftarrow A(T_1, T_2, T_3)$.
  Create variables for $X_i \in \mathbb{R}^{s_i \times k}, \forall i \in [3]$.
  Create variables for $C \in \mathbb{R}^{k \times k \times k}$.
  Run a polynomial system verifier for $\|C((Y_1 X_1), (Y_2 X_2), (Y_3 X_3)) - B\|_F^2$.
  return C,$A_1 S_1 X_1$, $A_2 S_2 X_2$, and $A_3 S_3 X_3$.
end procedure

Suppose the optimal $A_k = C^*(U^*, V^*, W^*)$. We fix $C^* \in \mathbb{R}^{k \times k \times k}$, $V^* \in \mathbb{R}^{n \times k}$ and $W^* \in \mathbb{R}^{n \times k}$. We use $V_1^*, V_2^*, \cdots, V_k^*$ to denote the columns of V^* and $W_1^*, W_2^*, \cdots, W_k^*$ to denote the columns of W^*.

We consider the following optimization problem,

$$\min_{U_1, \cdots, U_k \in \mathbb{R}^n} \|C^*(U, V^*, W^*) - A\|_F^2,$$

which is equivalent to

$$\min_{U_1, \cdots, U_k \in \mathbb{R}^n} \|U \cdot C^*(I, V^*, W^*) - A\|_F^2,$$

because $C^*(U, V^*, W^*) = U \cdot C^*(I, V^*, W^*)$ according to Definition 21.2.6.

Recall that $C^*(I, V^*, W^*)$ denotes a $k \times n \times n$ tensor. Let $(C^*(I, V^*, W^*))_1$ denote the matrix obtained by flattening $C^*(I, V^*, W^*)$ along the first dimension. We use matrix Z_1

to denote $(C^*(I, V^*, W^*))_1 \in \mathbb{R}^{k \times n^2}$. Then we can obtain the following equivalent objective function,

$$\min_{U \in \mathbb{R}^{n \times k}} \|UZ_1 - A_1\|_F^2.$$

Notice that $\min_{U \in \mathbb{R}^{n \times k}} \|UZ_1 - A_1\|_F^2 = \mathrm{OPT}$, since $A_k = U^* Z_1$.

Let $S_1^\top \in \mathbb{R}^{s_1 \times n^2}$ be the sketching matrix defined in Definition 21.3.12, where $s_1 = O(k/\epsilon)$. We obtain the following optimization problem,

$$\min_{U \in \mathbb{R}^{n \times k}} \|UZ_1 S_1 - A_1 S_1\|_F^2.$$

Let $\widehat{U} \in \mathbb{R}^{n \times k}$ denote the optimal solution to the above optimization problem. Then $\widehat{U} = A_1 S_1 (Z_1 S_1)^\dagger$. By Lemma 21.3.10 and Theorem 21.3.11, we have

$$\|\widehat{U} Z_1 - A_1\|_F^2 \leq (1+\epsilon) \min_{U \in \mathbb{R}^{n \times k}} \|UZ_1 - A_1\|_F^2 = (1+\epsilon)\,\mathrm{OPT},$$

which implies

$$\left\| C^*(\widehat{U}, V^*, W^*) - A \right\|_F^2 \leq (1+\epsilon)\,\mathrm{OPT}\,.$$

To write down $\widehat{U}_1, \cdots, \widehat{U}_k$, we use the given matrix A_1, and we create $s_1 \times k$ variables for matrix $(Z_1 S_1)^\dagger$.

As our second step, we fix $\widehat{U} \in \mathbb{R}^{n \times k}$ and $W^* \in \mathbb{R}^{n \times k}$, and we convert tensor A into matrix A_2. Let matrix Z_2 denote $(C^*(\widehat{U}, I, W^*))_2 \in \mathbb{R}^{k \times n^2}$. We consider the following objective function,

$$\min_{V \in \mathbb{R}^{n \times k}} \|VZ_2 - A_2\|_F^2,$$

for which the optimal cost is at most $(1+\epsilon)\,\mathrm{OPT}$.

Let $S_2^\top \in \mathbb{R}^{s_2 \times n^2}$ be a sketching matrix defined in Definition 21.3.12, where $s_2 = O(k/\epsilon)$. We sketch S_2 on the right of the objective function to obtain a new objective function,

$$\min_{V \in \mathbb{R}^{n \times k}} \|V Z_2 S_2 - A_2 S_2\|_F^2.$$

Let $\widehat{V} \in \mathbb{R}^{n \times k}$ denote the optimal solution to the above problem. Then $\widehat{V} = A_2 S_2 (Z_2 S_2)^\dagger$. By Lemma 21.3.10 and Theorem 21.3.11, we have,

$$\|\widehat{V} Z_2 - A_2\|_F^2 \leq (1+\epsilon) \min_{V \in \mathbb{R}^{n \times k}} \|V Z_2 - A_2\|_F^2 \leq (1+\epsilon)^2\,\mathrm{OPT},$$

which implies

$$\left\| C^*(\widehat{U}, \widehat{V}, W^*) - A \right\|_F^2 \leq (1+\epsilon)^2\,\mathrm{OPT}\,.$$

To write down $\widehat{V}_1, \cdots, \widehat{V}_k$, we need to use the given matrix $A_2 \in \mathbb{R}^{n^2 \times n}$, and we need to create $s_2 \times k$ variables for matrix $(Z_2 S_2)^\dagger$.

As our third step, we fix the matrices $\widehat{U} \in \mathbb{R}^{n \times k}$ and $\widehat{V} \in \mathbb{R}^{n \times k}$. We convert tensor $A \in \mathbb{R}^{n \times n \times n}$ into matrix $A_3 \in \mathbb{R}^{n^2 \times n}$. Let matrix Z_3 denote $(C^*(\widehat{U}, \widehat{V}, I))_3 \in \mathbb{R}^{k \times n^2}$. We consider the following objective function,

$$\min_{W \in \mathbb{R}^{n \times k}} \|W Z_3 - A_3\|_F^2,$$

which has optimal cost at most $(1+\epsilon)^2\,\mathrm{OPT}$.

Let $S_3^\top \in \mathbb{R}^{s_3 \times n^2}$ be a sketching matrix defined in Definition 21.3.12, where $s_3 = O(k/\epsilon)$. We sketch S_3 on the right of the objective function to obtain a new objective

function,

$$\min_{W \in \mathbb{R}^{n \times k}} \|W Z_3 S_3 - A_3 S_3\|_F^2.$$

Let $\widehat{W} \in \mathbb{R}^{n \times k}$ denote the optimal solution of the above problem. Then $\widehat{W} = A_3 S_3 (Z_3 S_3)^\dagger$. By Lemma 21.3.10 and Theorem 21.3.11, we have,

$$\|\widehat{W} Z_3 - A_3\|_F^2 \leq (1+\epsilon) \min_{W \in \mathbb{R}^{n \times k}} \|W Z_3 - A_3\|_F^2 \leq (1+\epsilon)^3 \operatorname{OPT}.$$

Thus, we have

$$\min_{X_1, X_2, X_3} \|C^*((A_1 S_1 X_1), (A_2 S_2 X_2), (A_3 S_3 X_3)) - A\|_F^2 \leq (1+\epsilon)^3 \operatorname{OPT}.$$

Let $V_1 = A_1 S_1, V_2 = A_2 S_2$, and $V_3 = A_3 S_3$. We then apply Lemma 21.4.3, and we obtain $\widehat{V}_1, \widehat{V}_2, \widehat{V}_3, B$. We then apply Theorem 21.4.44. Correctness follows by rescaling ϵ by a constant factor.

Running time. Due to Definition 21.3.12, the running time of line 7 (Algorithm 21.33) is $O(\operatorname{nnz}(A)) + n \operatorname{poly}(k, 1/\epsilon)$. Due to Lemma 21.4.3, line 7 and 8 can be executed in $\operatorname{nnz}(A) + n \operatorname{poly}(k, 1/\epsilon)$ time. The running time of line 11 is given by Theorem 21.4.44. (For simplicity, we ignore the bit complexity in the running time.) □

21.10.2 Tensor Train rank

21.10.2.1 Definitions

The tensor train rank has been studied in several works [Ose11, OTZ11, ZWZ16, PTBD16]. We provide the formal definition here.

Definition 21.10.2 (Tensor Train rank)**.** Given a third order tensor $A \in \mathbb{R}^{n\times n\times n}$, we say A has train rank k if k is the smallest integer such that there exist three tensors $U \in \mathbb{R}^{1\times n\times k}$, $V \in \mathbb{R}^{k\times n\times k}$, $W \in \mathbb{R}^{k\times n\times 1}$ satisfying:

$$A_{i,j,l} = \sum_{i_1=1}^{1}\sum_{i_2=1}^{k}\sum_{i_3=1}^{k}\sum_{i_4=1}^{1} U_{i_1,i,i_2}V_{i_2,j,i_3}W_{i_3,l,i_4}, \forall i,j,l \in [n]\times[n]\times[n],$$

or equivalently,

$$A_{i,j,l} = \sum_{i_2=1}^{k}\sum_{i_3=1}^{k}(U_2)_{i,i_2}(V_2)_{j,i_2+k(i_3-1)}(W_2)_{l,i_3},$$

where $V_2 \in \mathbb{R}^{n\times k^2}$ denotes the matrix obtained by flattening the tensor U along the second dimension, and $(V_2)_{i,i_1+k(i_2-1)}$ denotes the entry in the i-th row and $i_1+k(i_2-1)$-th column of V_2. We similarly define $U_2, W_2 \in \mathbb{R}^{n\times k}$.

21.10.2.2 Algorithm

Theorem 21.10.2. *Given a third order tensor $A \in \mathbb{R}^{n\times n\times n}$, for any $k \geq 1$, $\epsilon \in (0,1)$, there exists an algorithm which takes $O(\mathrm{nnz}(A)) + n\,\mathrm{poly}(k, 1/\epsilon) + 2^{O(k^4/\epsilon)}$ time and outputs three tensors $U \in \mathbb{R}^{1\times n\times k}$, $V \in \mathbb{R}^{k\times n\times k}$, $W \in \mathbb{R}^{k\times n\times 1}$ such that*

$$\left\|\sum_{i=1}^{k}\sum_{j=1}^{k}(U_2)_i \otimes (V_2)_{i+k(j-1)} \otimes (W_2)_j - A\right\|_F^2 \leq (1+\epsilon)\min_{\text{train rank}-k\ A_k}\|A_k - A\|_F^2$$

holds with probability $9/10$.

Algorithm 21.34 Frobenius Norm Low (Train) rank Approximation

procedure FLOWTRAINRANKAPPROX(A, n, k, ϵ) ▷ Theorem 21.10.2
  $s_1 \leftarrow s_3 \leftarrow O(k/\epsilon)$.
  $s_2 \leftarrow O(k^2/\epsilon)$.
  $t_1 \leftarrow t_2 \leftarrow t_3 \leftarrow \text{poly}(k, 1/\epsilon)$.
  Choose sketching matrices $S_1 \in \mathbb{R}^{n^2 \times s_1}$, $S_2 \in \mathbb{R}^{n^2 \times s_2}$, $S_3 \in \mathbb{R}^{n^2 \times s_3}$. ▷ Definition 21.3.12
  Choose sketching matrices $T_1 \in \mathbb{R}^{t_1 \times n}$, $T_2 \in \mathbb{R}^{t_2 \times n}$, $T_3 \in \mathbb{R}^{t_3 \times n}$.
  Compute $A_i S_i, \forall i \in [3]$.
  Compute $T_i A_i S_i$, $\forall i \in [3]$.
  Compute $B \leftarrow A(T_1, T_2, T_3)$.
  Create variables for $X_1 \in \mathbb{R}^{s_1 \times k}$.
  Create variables for $X_3 \in \mathbb{R}^{s_3 \times k}$.
  Create variables for $X_2 \in \mathbb{R}^{s_2 \times k^2}$.
  Create variables for $C \in \mathbb{R}^{k \times k \times k}$.
  Run polynomial system verifier for $\| \sum_{i_2=1}^{k} \sum_{i_3=1}^{k} (Y_1 X_1)_{i_2} (Y_2 X_2)_{i_2 + k(i_3 - 1)} (Y_3 X_3)_{i_3} - B \|_F^2$.
  return $A_1 S_1 X_1$, $A_2 S_2 X_2$, and $A_3 S_3 X_3$.
end procedure

Proof. We define OPT as

$$\mathrm{OPT} = \min_{\text{train}} \min_{\text{rank}-k \ A'} \|A' - A\|_F^2.$$

Suppose the optimal

$$A_k = \sum_{i=1}^{k} \sum_{j=1}^{k} U_i^* \otimes V_{i+k(j-1)}^* \otimes W_j^*.$$

We fix $V^* \in \mathbb{R}^{n \times k^2}$ and $W^* \in \mathbb{R}^{n \times k}$. We use $V_1^*, V_2^*, \cdots, V_{k^2}^*$ to denote the columns of V^*, and $W_1^*, W_2^*, \cdots, W_k^*$ to denote the columns of W^*.

We consider the following optimization problem,

$$\min_{U \in \mathbb{R}^{n \times k}} \left\| \sum_{i=1}^{k} \sum_{j=1}^{k} U_i \otimes V_{i+k(j-1)}^* \otimes W_j^* - A \right\|_F^2,$$

which is equivalent to

$$\min_{U \in \mathbb{R}^{n \times k}} \left\| U \cdot \begin{bmatrix} \sum_{j=1}^{k} V_{1+k(j-1)}^* \otimes W_j^* \\ \sum_{j=1}^{k} V_{2+k(j-1)}^* \otimes W_j^* \\ \cdots \\ \sum_{j=1}^{k} V_{k+k(j-1)}^* \otimes W_j^* \end{bmatrix} - A \right\|_F^2.$$

Let $A_1 \in \mathbb{R}^{n \times n^2}$ denote the matrix obtained by flattening the tensor A along the first dimension. We use matrix $Z_1 \in \mathbb{R}^{k \times n^2}$ to denote

$$\begin{bmatrix} \sum_{j=1}^{k} \mathrm{vec}(V_{1+k(j-1)}^* \otimes W_j^*) \\ \sum_{j=1}^{k} \mathrm{vec}(V_{2+k(j-1)}^* \otimes W_j^*) \\ \cdots \\ \sum_{j=1}^{k} \mathrm{vec}(V_{k+k(j-1)}^* \otimes W_j^*) \end{bmatrix}.$$

Then we can obtain the following equivalent objective function,

$$\min_{U \in \mathbb{R}^{n \times k}} \|U Z_1 - A_1\|_F^2.$$

Notice that $\min_{U \in \mathbb{R}^{n \times k}} \|U Z_1 - A_1\|_F^2 = \mathrm{OPT}$, since $A_k = U^* Z_1$.

Let $S_1^\top \in \mathbb{R}^{s_1 \times n^2}$ be a sketching matrix defined in Definition 21.3.12, where $s_1 = O(k/\epsilon)$. We obtain the following optimization problem,

$$\min_{U \in \mathbb{R}^{n \times k}} \|U Z_1 S_1 - A_1 S_1\|_F^2.$$

Let $\widehat{U} \in \mathbb{R}^{n \times k}$ denote the optimal solution to the above optimization problem. Then $\widehat{U} = A_1 S_1 (Z_1 S_1)^\dagger$. By Lemma 21.3.10 and Theorem 21.3.11, we have

$$\|\widehat{U} Z_1 - A_1\|_F^2 \leq (1+\epsilon) \min_{U \in \mathbb{R}^{n \times k}} \|U Z_1 - A_1\|_F^2 = (1+\epsilon)\,\mathrm{OPT},$$

which implies

$$\left\| \sum_{i=1}^k \sum_{j=1}^k \widehat{U}_i \otimes V_{i+k(j-1)}^* \otimes W_j^* - A \right\|_F^2 \leq (1+\epsilon)\,\mathrm{OPT}\,.$$

To write down $\widehat{U}_1, \cdots, \widehat{U}_k$, we use the given matrix A_1, and we create $s_1 \times k$ variables for matrix $(Z_1 S_1)^\dagger$.

As our second step, we fix $\widehat{U} \in \mathbb{R}^{n \times k}$ and $W^* \in \mathbb{R}^{n \times k}$, and we convert the tensor A into matrix A_2. Let matrix $Z_2 \in \mathbb{R}^{k^2 \times n^2}$ denote the matrix where the (i,j)-th row is the vectorization of $\widehat{U}_i \otimes W_j^*$. We consider the following objective function,

$$\min_{V \in \mathbb{R}^{n \times k}} \|V Z_2 - A_2\|_F^2,$$

for which the optimal cost is at most $(1+\epsilon)\,\mathrm{OPT}$.

Let $S_2^\top \in \mathbb{R}^{s_2 \times n^2}$ be a sketching matrix defined in Definition 21.3.12, where $s_2 = O(k^2/\epsilon)$. We sketch S_2 on the right of the objective function to obtain the new objective function,

$$\min_{V \in \mathbb{R}^{n \times k}} \|V Z_2 S_2 - A_2 S_2\|_F^2.$$

Let $\widehat{V} \in \mathbb{R}^{n \times k}$ denote the optimal solution of the above problem. Then $\widehat{V} = A_2 S_2 (Z_2 S_2)^\dagger$. By Lemma 21.3.10 and Theorem 21.3.11, we have,

$$\|\widehat{V} Z_2 - A_2\|_F^2 \leq (1+\epsilon) \min_{V \in \mathbb{R}^{n \times k}} \|V Z_2 - A_2\|_F^2 \leq (1+\epsilon)^2\,\mathrm{OPT},$$

which implies

$$\left\| \sum_{i=1}^k \sum_{j=1}^k \widehat{U}_i \otimes \widehat{V}_{i+k(j-1)} \otimes W^* - A \right\|_F^2 \leq (1+\epsilon)^2\,\mathrm{OPT}\,.$$

To write down $\widehat{V}_1, \cdots, \widehat{V}_k$, we need to use the given matrix $A_2 \in \mathbb{R}^{n^2 \times n}$, and we need to create $s_2 \times k$ variables for matrix $(Z_2 S_2)^\dagger$.

As our third step, we fix the matrices $\widehat{U} \in \mathbb{R}^{n \times k}$ and $\widehat{V} \in \mathbb{R}^{n \times k}$. We convert tensor $A \in \mathbb{R}^{n \times n \times n}$ into matrix $A_3 \in \mathbb{R}^{n^2 \times n}$. Let matrix $Z_3 \in \mathbb{R}^{k \times n^2}$ denote

$$\begin{bmatrix} \sum_{i=1}^k \mathrm{vec}(\widehat{U}_i \otimes \widehat{V}_{i+k\cdot 0}) \\ \sum_{i=1}^k \mathrm{vec}(\widehat{U}_i \otimes \widehat{V}_{i+k\cdot 1}) \\ \cdots \\ \sum_{i=1}^k \mathrm{vec}(\widehat{U}_i \otimes \widehat{V}_{i+k\cdot(k-1)}) \end{bmatrix}.$$

We consider the following objective function,

$$\min_{W \in \mathbb{R}^{n \times k}} \|W Z_3 - A_3\|_F^2,$$

which has optimal cost at most $(1+\epsilon)^2$ OPT.

Let $S_3^\top \in \mathbb{R}^{s_3 \times n^2}$ be a sketching matrix defined in Definition 21.3.12, where $s_3 = O(k/\epsilon)$. We sketch S_3 on the right of the objective function to obtain a new objective function,

$$\min_{W \in \mathbb{R}^{n \times k}} \|W Z_3 S_3 - A_3 S_3\|_F^2.$$

Let $\widehat{W} \in \mathbb{R}^{n \times k}$ denote the optimal solution of the above problem. Then $\widehat{W} = A_3 S_3 (Z_3 S_3)^\dagger$. By Lemma 21.3.10 and Theorem 21.3.11, we have,

$$\|\widehat{W} Z_3 - A_3\|_F^2 \leq (1+\epsilon) \min_{W \in \mathbb{R}^{n \times k}} \|W Z_3 - A_3\|_F^2 \leq (1+\epsilon)^3 \, \mathrm{OPT} \, .$$

Thus, we have

$$\min_{X_1, X_2, X_3} \left\| \sum_{i=1}^k \sum_{j=1}^k (A_1 S_1 X_1)_i \otimes (A_2 S_2 X_2)_{i+k(j-1)} \otimes (A_3 S_3 X_3)_j - A \right\|_F^2 \leq (1+\epsilon)^3 \, \mathrm{OPT} \, .$$

Let $V_1 = A_1 S_1, V_2 = A_2 S_2$, and $V_3 = A_3 S_3$. We then apply Lemma 21.4.3, and we obtain $\widehat{V}_1, \widehat{V}_2, \widehat{V}_3, B$. We then apply Theorem 21.4.44. Correctness follows by rescaling ϵ by a constant factor.

Running time. Due to Definition 21.3.12, the running time of line 7 (Algorithm 21.34) is $O(\mathrm{nnz}(A)) + n \, \mathrm{poly}(k, 1/\epsilon)$. Due to Lemma 21.4.3, lines 8 and 9 can be executed in $\mathrm{nnz}(A) + n \, \mathrm{poly}(k, 1/\epsilon)$ time. The running time of $2^{O(k^4/\epsilon)}$ comes from running Theorem 21.4.44 (For simplicity, we ignore the bit complexity in the running time.) □

21.11 Entry-wise ℓ_p Norm for Arbitrary Tensors, $1 < p < 2$

There is a long line of research dealing with ℓ_p norm-related problems [DDH+09, MM13, CDMI+13, CP15, BCKY16, YCRM16, BBC+17].

In this section, we provide several different algorithms for tensor ℓ_p-low rank approximation. Section 21.11.1 formally states the ℓ_p version of Theorem C.1 in [SWZ17]. Section 21.11.2 presents several existence results. Section 21.11.3 describes a tool that is able to reduce the size of the objective function from poly(n) to poly(k). Section 21.11.4 discusses the case when the problem size is small. Section 21.11.5 provides several bicriteria algorithms. Section 21.11.6 summarizes a batch of algorithms. Section 21.11.7 provides an algorithm for ℓ_p norm CURT decomposition.

Notice that if the rank-k solution does not exist, then every bicriteria algorithm in Section 21.11.5 can be stated in the form as Theorem 21.1.1, and every algorithm which can output a rank-k solution in Section 21.11.6 can be stated in the form as Theorem 21.1.2. See Section 21.1 for more details.

21.11.1 Existence results for matrix case

Theorem 21.11.1 ([SWZ17]). *Let $1 \leq p < 2$. Given $V \in \mathbb{R}^{k\times n}, A \in \mathbb{R}^{d\times n}$. Let $S \in \mathbb{R}^{n\times s}$ be a proper random sketching matrix. Let*

$$\widehat{U} = \arg\min_{U\in\mathbb{R}^{d\times k}} \|UVS - AS\|_F^2,$$

i.e.,

$$\widehat{U} = AS(VS)^\dagger.$$

Then with probability at least 0.999,

$$\|\widehat{U}V - A\|_p^p \leq \alpha \cdot \min_{U\in\mathbb{R}^{d\times k}} \|UV - A\|_p^p.$$

(I). *S denotes a dense p-stable transform,*
$s = \widetilde{O}(k)$, $\alpha = \widetilde{O}(k^{1-p/2})\log d$.

(II). *S denotes a sparse p-stable transform,*
$s = \widetilde{O}(k^5)$, $\alpha = \widetilde{O}(k^{5-5p/2+2/p})\log d$.

(III). *$S^\top$ denotes a sampling/rescaling matrix according to the ℓ_p Lewis weights of $V^\top$,*
$s = \widetilde{O}(k)$, $\alpha = \widetilde{O}(k^{1-p/2})$.

We give the proof for completeness.

Proof. Let $S \in \mathbb{R}^{n\times s}$ be a sketching matrix which satisfies the property $(*)$: $\forall c \geq 1, \widetilde{U} \in \mathbb{R}^{d\times k}$ which satisfy

$$\|\widetilde{U}VS - AS\|_p^p \leq c \cdot \min_{U\in\mathbb{R}^{d\times k}} \|UVS - AS\|_p^p,$$

we have

$$\|\widetilde{U}V - A\|_p^p \leq c\beta_S \cdot \min_{U \in \mathbb{R}^{d \times k}} \|UV - A\|_p^p,$$

where $\beta_S \geq 1$ only depends on the sketching matrix S. Let

$$\forall i \in [d], (\widehat{U}^i)^\top = \arg\min_{x \in \mathbb{R}^k} \|x^\top VS - A^i S\|_2^2,$$

i.e.,

$$\widehat{U} = AS(VS)^\dagger.$$

Let

$$\widetilde{U} = \arg\min_{U \in \mathbb{R}^{d \times k}} \|UVS - AS\|_p^p.$$

Then, we have:

$$\begin{aligned}
& \|\widehat{U}VS - AS\|_p^p \\
= & \sum_{i=1}^{d} \|\widehat{U}^i VS - A^i S\|_p^p \\
\leq & \sum_{i=1}^{d} (s^{1/p-1/2} \|\widehat{U}^i VS - A^i S\|_2)^p \\
\leq & \sum_{i=1}^{d} (s^{1/p-1/2} \|\widetilde{U}^i VS - A^i S\|_2)^p \\
\leq & \sum_{i=1}^{d} (s^{1/p-1/2} \|\widetilde{U}^i VS - A^i S\|_p)^p \\
\leq & s^{1-p/2} \|\widetilde{U}VS - AS\|_p^p.
\end{aligned}$$

The first inequality follows using $\forall x \in \mathbb{R}^s, \|x\|_p \leq s^{1/p-1/2}\|x\|_2$ since $p < 2$. The third inequality follows using $\forall x \in \mathbb{R}^s, \|x\|_2 \leq \|x\|_p$ since $p < 2$. Thus, according to the property $(*)$ of S,

$$\|\widehat{U}V - A\|_p^p \leq s^{1-p/2}\beta_S \min_{U \in \mathbb{R}^{d\times k}} \|UV - A\|_p^p.$$

Due to Lemma E.8 and Lemma E.11 of [SWZ17], we have:

for (I), $s = \widetilde{O}(k), \beta_S = O(\log d), \alpha = s^{1-p/2}\beta_S = \widetilde{O}(k^{1-p/2})\log d$,

for (II), $s = \widetilde{O}(k^5), \beta_S = \widetilde{O}(k^{2/p}\log d), \alpha = s^{1-p/2}\beta_S = \widetilde{O}(k^{5-5p/2+2/p})\log d$,

for (III), $s = \widetilde{O}(k), \beta_S = O(1), \alpha = s^{1-p/2}\beta_S = \widetilde{O}(k^{1-p/2})$. □

21.11.2 Existence results

Theorem 21.11.2. *Given a 3rd order tensor $A \in \mathbb{R}^{n\times n\times n}$, for any $k \geq 1$, there exist three matrices $S_1 \in \mathbb{R}^{n^2\times s_1}$, $S_2 \in \mathbb{R}^{n^2\times s_2}$, $S_3 \in \mathbb{R}^{n^2\times s_3}$ such that*

$$\min_{X_1,X_2,X_3} \left\| \sum_{i=1}^{k} (A_1S_1X_1)_i \otimes (A_2S_2X_2)_i \otimes (A_3S_3X_3)_i - A \right\|_p^p \leq \alpha \min_{\mathrm{rank}-k \ A_k\in\mathbb{R}^{n\times n\times n}} \|A_k - A\|_p^p,$$

holds with probability $99/100$.

(I). *Using a dense p-stable transform,*
$s_1 = s_2 = s_3 = \widetilde{O}(k)$, $\alpha = \widetilde{O}(k^{3-1.5p}) \log^3 n$.

(II). *Using a sparse p-stable transform,*
$s_1 = s_2 = s_3 = \widetilde{O}(k^5)$, $\alpha = \widetilde{O}(k^{15-7.5p+6/p}) \log^3 n$.

(III). *Guessing Lewis weights,*
$s_1 = s_2 = s_3 = \widetilde{O}(k)$, $\alpha = \widetilde{O}(k^{3-1.5p})$.

Proof. We use OPT to denote

$$\mathrm{OPT} := \min_{\mathrm{rank}-k \ A_k\in\mathbb{R}^{n\times n\times n}} \|A_k - A\|_p^p.$$

Given a tensor $A \in \mathbb{R}^{n_1\times n_2\times n_3}$, we define three matrices $A_1 \in \mathbb{R}^{n_1\times n_2n_3}$, $A_2 \in \mathbb{R}^{n_2\times n_3n_1}$, $A_3 \in \mathbb{R}^{n_3\times n_1n_2}$ such that, for any $i \in [n_1], j \in [n_2], l \in [n_3]$

$$A_{i,j,l} = (A_1)_{i,(j-1)\cdot n_3+l} = (A_2)_{j,(l-1)\cdot n_1+i} = (A_3)_{l,(i-1)\cdot n_2+j}.$$

We fix $V^* \in \mathbb{R}^{n\times k}$ and $W^* \in \mathbb{R}^{n\times k}$, and use $V_1^*, V_2^*, \cdots, V_k^*$ to denote the columns of V^* and $W_1^*, W_2^*, \cdots, W_k^*$ to denote the columns of W^*.

We consider the following optimization problem,

$$\min_{U_1,\cdots,U_k\in\mathbb{R}^n}\left\|\sum_{i=1}^k U_i\otimes V_i^*\otimes W_i^* - A\right\|_p^p,$$

which is equivalent to

$$\min_{U_1,\cdots,U_k\in\mathbb{R}^n}\left\|\begin{bmatrix}U_1 & U_2 & \cdots & U_k\end{bmatrix}\begin{bmatrix}V_1^*\otimes W_1^*\\ V_2^*\otimes W_2^*\\ \cdots\\ V_k^*\otimes W_k^*\end{bmatrix} - A\right\|_p^p.$$

We use matrix Z_1 to denote $V^{*\top}\odot W^{*\top}\in\mathbb{R}^{k\times n^2}$ and matrix U to denote $\begin{bmatrix}U_1 & U_2 & \cdots & U_k\end{bmatrix}$. Then we can obtain the following equivalent objective function,

$$\min_{U\in\mathbb{R}^{n\times k}}\|UZ_1 - A_1\|_p^p.$$

Choose a sketching matrix (a dense p-stable, a sparse p-stable or an ℓ_p Lewis weight sampling/rescaling matrix to Z_1) $S_1\in\mathbb{R}^{n^2\times s_1}$. We can obtain the optimization problem,

$$\min_{U\in\mathbb{R}^{n\times k}}\|UZ_1S_1 - A_1S_1\|_p^p = \min_{U\in\mathbb{R}^{n\times k}}\sum_{i=1}^n\|U^iZ_1S_1 - (A_1S_1)^i\|_p^p,$$

where U^i denotes the i-th row of matrix $U\in\mathbb{R}^{n\times k}$ and $(A_1S_1)^i$ denotes the i-th row of matrix A_1S_1. Instead of solving it under the ℓ_p-norm, we consider the ℓ_2-norm relaxation,

$$\min_{U\in\mathbb{R}^{n\times k}}\|UZ_1S_1 - A_1S_1\|_F^2 = \min_{U\in\mathbb{R}^{n\times k}}\sum_{i=1}^n\|U^iZ_1S_1 - (A_1S_1)^i\|_2^2.$$

Let $\widehat{U}\in\mathbb{R}^{n\times k}$ denote the optimal solution of the above optimization problem. Then, $\widehat{U} = A_1S_1(Z_1S_1)^\dagger$. We plug $\widehat{U}$ into the objective function under the ℓ_p-norm. By choosing s_1 and by the properties of sketching matrices (a dense p-stable, a sparse p-stable or an ℓ_p Lewis weight sampling/rescaling matrix to Z_1) $S_1\in\mathbb{R}^{n^2\times s_1}$, we have

$$\|\widehat{U}Z_1 - A_1\|_p^p \le \alpha\min_{U\in\mathbb{R}^{n\times k}}\|UZ_1 - A_1\|_p^p = \alpha\,\mathrm{OPT}\,.$$

This implies

$$\|\widehat{U} \otimes V^* \otimes W^* - A\|_p^p \leq \alpha \, \mathrm{OPT} \, .$$

As a second step, we fix $\widehat{U} \in \mathbb{R}^{n \times k}$ and $W^* \in \mathbb{R}^{n \times k}$, and convert tensor A into matrix A_2. Let matrix Z_2 denote $\widehat{U}^\top \odot W^{*\top}$. We consider the following objective function,

$$\min_{V \in \mathbb{R}^{n \times k}} \|V Z_2 - A_2\|_p^p,$$

and the optimal cost of it is at most $\alpha \, \mathrm{OPT}$.

We choose a sketching matrix (a dense p-stable, a sparse p-stable or an ℓ_p Lewis weight sampling/rescaling matrix to Z_2) $S_2 \in \mathbb{R}^{n^2 \times s_2}$ and sketch on the right of the objective function to obtain the new objective function,

$$\min_{V \in \mathbb{R}^{n \times k}} \|V Z_2 S_2 - A_2 S_2\|_p^p = \min_{V \in \mathbb{R}^{n \times k}} \sum_{i=1}^n \|V^i Z_2 S_2 - (A_2 S_2)^i\|_p^p,$$

where V^i denotes the i-th row of matrix V and $(A_2 S_2)^i$ denotes the i-th row of matrix $A_2 S_2$. Instead of solving this under the ℓ_p-norm, we consider the ℓ_2-norm relaxation,

$$\min_{V \in \mathbb{R}^{n \times k}} \|V Z_2 S_2 - A_2 S_2\|_F^2 = \min_{V \in \mathbb{R}^{n \times k}} \sum_{i=1}^n \|V^i (Z_2 S_2) - (A_2 S_2)^i\|_2^2.$$

Let $\widehat{V} \in \mathbb{R}^{n \times k}$ denote the optimal solution of the above problem. Then $\widehat{V} = A_2 S_2 (Z_2 S_2)^\dagger$. By properties of sketching matrix $S_2 \in \mathbb{R}^{n^2 \times s_2}$, we have,

$$\|\widehat{V} Z_2 - A_2\|_p^p \leq \alpha \min_{V \in \mathbb{R}^{n \times k}} \|V Z_2 - A_2\|_p^p \leq \alpha^2 \, \mathrm{OPT},$$

which implies

$$\|\widehat{U} \otimes \widehat{V} \otimes W^* - A\|_p^p \leq \alpha^2 \, \mathrm{OPT},$$

As a third step, we fix the matrices $\widehat{U} \in \mathbb{R}^{n \times k}$ and $\widehat{V} \in \mathbb{R}^{n \times k}$. We can convert tensor $A \in \mathbb{R}^{n \times n \times n}$ into matrix $A_3 \in \mathbb{R}^{n^2 \times n}$. Let matrix Z_3 denote $\widehat{U}^\top \odot \widehat{V}^\top \in \mathbb{R}^{k \times n^2}$. We consider the following objective function,

$$\min_{W \in \mathbb{R}^{n \times k}} \|W Z_3 - A_3\|_p^p,$$

and the optimal cost of it is at most $\alpha^2 \, \mathrm{OPT}$.

We choose sketching matrix (a dense p-stable, a sparse p-stable or an ℓ_p Lewis weight sampling/rescaling matrix to Z_3) $S_3 \in \mathbb{R}^{n^2 \times s_3}$ and sketch on the right of the objective function to obtain the new objective function,

$$\min_{W \in \mathbb{R}^{n \times k}} \|W Z_3 S_3 - A_3 S_3\|_p^p.$$

Instead of solving this under the ℓ_p-norm, we consider the ℓ_2-norm relaxation,

$$\min_{W \in \mathbb{R}^{n \times k}} \|W Z_3 S_3 - A_3 S_3\|_F^2 = \min_{W \in \mathbb{R}^{n \times k}} \sum_{i=1}^n \|W^i (Z_3 S_3) - (A_3 S_3)^i\|_2^2.$$

Let $\widehat{W} \in \mathbb{R}^{n \times k}$ denote the optimal solution of the above problem. Then $\widehat{W} = A_3 S_3 (Z_3 S_3)^\dagger$. By properties of sketching matrix $S_3 \in \mathbb{R}^{n^2 \times s_3}$, we have,

$$\|\widehat{W} Z_3 - A_3\|_p^p \leq \alpha \min_{W \in \mathbb{R}^{n \times k}} \|W Z_3 - A_3\|_p^p \leq \alpha^3 \, \mathrm{OPT} \, .$$

Thus, we obtain,

$$\min_{X_1 \in \mathbb{R}^{s_1 \times k}, X_2 \in \mathbb{R}^{s_2 \times k}, X_3 \in \mathbb{R}^{s_3 \times k}} \left\| \sum_{i=1}^k (A_1 S_1 X_1)_i \otimes (A_2 S_2 X_2)_i \otimes (A_3 S_3 X_3)_i - A \right\|_p^p \leq \alpha^3 \, \mathrm{OPT} \, .$$

According to Theorem 21.11.1, we let $s = s_1 = s_2 = s_3$ and take the corresponding α. We can directly get the results for (I), (II) and (III). $\square$

21.11.3 Polynomial in k size reduction

Definition 21.11.1 (Definition E.1 in [SWZ17])**.** Given a matrix $M \in \mathbb{R}^{n\times d}$, if matrix $S \in \mathbb{R}^{m\times n}$ satisfies

$$\|SM\|_p^p \leq \beta \|M\|_p^p,$$

then S has at most β dilation on M in the ℓ_p case.

Definition 21.11.2 (Definition E.2 in [SWZ17])**.** Given a matrix $U \in \mathbb{R}^{n\times k}$, if matrix $S \in \mathbb{R}^{m\times n}$ satisfies

$$\forall x \in \mathbb{R}^k, \|SUx\|_p^p \geq \frac{1}{\beta}\|Ux\|_p^p,$$

then S has at most β contraction on U in the ℓ_p case.

Theorem 21.11.3. *Given a tensor $A \in \mathbb{R}^{n_1\times n_2\times n_3}$ and three matrices $V_1 \in \mathbb{R}^{n_1\times b_1}, V_2 \in \mathbb{R}^{n_2\times b_2}, V_3 \in \mathbb{R}^{n_3\times b_3}$, let $X_1^* \in \mathbb{R}^{b_1\times k}, X_2^* \in \mathbb{R}^{b_2\times k}, X_3^* \in \mathbb{R}^{b_3\times k}$ satisfy*

$$X_1^*, X_2^*, X_3^* = \mathop{\arg\min}_{X_1\in\mathbb{R}^{b_1\times k}, X_2\in\mathbb{R}^{b_2\times k}, X_3\in\mathbb{R}^{b_3\times k}} \|V_1X_1 \otimes V_2X_2 \otimes V_3X_3 - A\|_p^p.$$

Let $S \in \mathbb{R}^{m\times n}$ have at most $\beta_1 \geq 1$ dilation on $V_1X_1^* \cdot ((V_2X_2^*)^\top \odot (V_3X_3^*)^\top) - A_1$ and S have at most $\beta_2 \geq 1$ contraction on V_1 in the ℓ_p case. If $\widehat{X}_1 \in \mathbb{R}^{b_1\times k}, \widehat{X}_2 \in \mathbb{R}^{b_2\times k}, \widehat{X}_3 \in \mathbb{R}^{b_3\times k}$ satisfy

$$\|SV_1\widehat{X}_1 \otimes V_2\widehat{X}_2 \otimes V_3\widehat{X}_3 - SA\|_p^p \leq \beta \min_{X_1\in\mathbb{R}^{b_1\times k}, X_2\in\mathbb{R}^{b_2\times k}, X_3\in\mathbb{R}^{b_3\times k}} \|SV_1X_1 \otimes V_2X_2 \otimes V_3X_3 - SA\|_p^p,$$

where $\beta \geq 1$, then

$$\|V_1\widehat{X}_1 \otimes V_2\widehat{X}_2 \otimes V_3\widehat{X}_3 - A\|_p^p \lesssim \beta_1\beta_2\beta \min_{X_1,X_2,X_3} \|V_1X_1 \otimes V_2X_2 \otimes V_3X_3 - A\|_p^p.$$

The proof is essentially the same as the proof of Theorem 21.7.4:

Proof. Let $A, V_1, V_2, V_3, S, X_1^*, X_2^*, X_3^*, \beta_1, \beta_2$ be as stated in the theorem. Let $\widehat{X}_1 \in \mathbb{R}^{b_1 \times k}, \widehat{X}_2 \in \mathbb{R}^{b_2 \times k}, \widehat{X}_3 \in \mathbb{R}^{b_3 \times k}$ satisfy

$$\|SV_1\widehat{X}_1 \otimes V_2\widehat{X}_2 \otimes V_3\widehat{X}_3 - SA\|_p^p \leq \beta \min_{X_1 \in \mathbb{R}^{b_1 \times k}, X_2 \in \mathbb{R}^{b_2 \times k}, X_3 \in \mathbb{R}^{b_3 \times k}} \|SV_1X_1 \otimes V_2X_2 \otimes V_3X_3 - SA\|_p^p.$$

Similar to the proof of Theorem 21.7.4, we have,

$$\begin{aligned}
&\|SV_1\widehat{X}_1 \otimes V_2\widehat{X}_2 \otimes V_3\widehat{X}_3 - SA\|_p^p \\
&= 2^{2-2p}\frac{1}{\beta_2}\|V_1\widehat{X}_1 \otimes V_2\widehat{X}_2 \otimes V_3\widehat{X}_3 - A\|_p^p - (2^{1-p}\frac{1}{\beta_2} + \beta_1)\|V_1X_1^* \otimes V_2X_2^* \otimes V_3X_3^* - A\|_p^p
\end{aligned}$$

The only difference from the proof of Theorem 21.7.4 is that instead of using triangle inequality, we actually use $\|x+y\|_p^p \leq 2^{p-1}\|x\|_p^p + \|y\|_p^p$. Then, we have

$$\begin{aligned}
&\|V_1\widehat{X}_1 \otimes V_2\widehat{X}_2 \otimes V_3\widehat{X}_3 - A\|_p^p \\
&\leq 2^{2p-2}\beta_2\|SV_1\widehat{X}_1 \otimes V_2\widehat{X}_2 \otimes V_3\widehat{X}_3 - SA\|_p^p + (2^{p-1} + 2^{2p-2}\beta_1\beta_2)\|V_1X_1^* \otimes V_2X_2^* \otimes V_3X_3^* - A\|_p^p \\
&\leq 2^{2p-2}\beta_2\beta\|SV_1X_1^* \otimes V_2X_2^* \otimes V_3X_3^* - SA\|_p^p + (2^{p-1} + 2^{2p-2}\beta_1\beta_2)\|V_1X_1^* \otimes V_2X_2^* \otimes V_3X_3^* - A\|_p^p \\
&\leq 2^{2p-2}\beta_1\beta_2\beta\|V_1X_1^* \otimes V_2X_2^* \otimes V_3X_3^* - A\|_p^p + (2^{p-1} + 2^{2p-2}\beta_1\beta_2)\|V_1X_1^* \otimes V_2X_2^* \otimes V_3X_3^* - A\|_p^p \\
&\leq 2^{p-1}\beta(1 + 2\beta_1\beta_2)\|V_1X_1^* \otimes V_2X_2^* \otimes V_3X_3^* - A\|_p^p.
\end{aligned}$$

□

Lemma 21.11.4. *Let* $\min(b_1, b_2, b_3) \geq k$. *Given three matrices* $V_1 \in \mathbb{R}^{n \times b_1}$, $V_2 \in \mathbb{R}^{n \times b_2}$, *and* $V_3 \in \mathbb{R}^{n \times b_3}$, *there exists an algorithm which takes* $O(\mathrm{nnz}(A)) + n\,\mathrm{poly}(b_1, b_2, b_3)$ *time and outputs a tensor* $C \in \mathbb{R}^{c_1 \times c_2 \times c_3}$ *and three matrices* $\widehat{V}_1 \in \mathbb{R}^{c_1 \times b_1}$, $\widehat{V}_2 \in \mathbb{R}^{c_2 \times b_2}$ *and* $\widehat{V}_3 \in \mathbb{R}^{c_3 \times b_3}$

with $c_1 = c_2 = c_3 = \mathrm{poly}(b_1, b_2, b_3)$*, such that with probability* 0.99*, for any* $\alpha \geq 1$*, if* X_1', X_2', X_3' *satisfy that,*

$$\left\| \sum_{i=1}^{k} (\widehat{V}_1 X_1')_i \otimes (\widehat{V}_2 X_2')_i \otimes (\widehat{V}_3 X_3')_i - C \right\|_p^p \leq \alpha \min_{X_1, X_2, X_3} \left\| \sum_{i=1}^{k} (\widehat{V}_1 X_1)_i \otimes (\widehat{V}_2 X_2)_i \otimes (\widehat{V}_3 X_3)_i - C \right\|_p^p,$$

then,

$$\left\| \sum_{i=1}^{k} (V_1 X_1')_i \otimes (V_2 X_2')_i \otimes (V_3 X_3')_i - A \right\|_p^p \lesssim \alpha \min_{X_1, X_2, X_3} \left\| \sum_{i=1}^{k} (V_1 X_1)_i \otimes (V_2 X_2)_i \otimes (V_3 X_3)_i - A \right\|_p^p.$$

Proof. For simplicity, we define OPT to be

$$\min_{X_1, X_2, X_3} \left\| \sum_{i=1}^{k} (V_1 X_1)_i \otimes (V_2 X_2)_i \otimes (V_3 X_3)_i - A \right\|_p^p.$$

Let $T_1 \in \mathbb{R}^{c_1 \times n}$ correspond to sampling according to the ℓ_p Lewis weights of $V_1 \in \mathbb{R}^{n \times b_1}$, where $c_1 = \widetilde{b_1}$. Let $T_2 \in \mathbb{R}^{c_2 \times n}$ be sampling according to the ℓ_p Lewis weights of $V_2 \in \mathbb{R}^{n \times b_2}$, where $c_2 = \widetilde{b_2}$. Let $T_3 \in \mathbb{R}^{c_3 \times n}$ be sampling according to the ℓ_p Lewis weights of $V_3 \in \mathbb{R}^{n \times b_3}$, where $c_3 = \widetilde{b_3}$.

For any $\alpha \geq 1$, let $X_1' \in \mathbb{R}^{b_1 \times k}, X_2' \in \mathbb{R}^{b_2 \times k}, X_3' \in \mathbb{R}^{b_3 \times k}$ satisfy

$$\begin{aligned} &\|T_1 V_1 X_1' \otimes T_2 V_2 X_2' \otimes T_3 V_3 X_3' - A(T_1, T_2, T_3)\|_p^p \\ \leq\ & \alpha \min_{X_1 \in \mathbb{R}^{b_1 \times k}, X_2 \in \mathbb{R}^{b_2 \times k}, X_3 \in \mathbb{R}^{b_3 \times k}} \|T_1 V_1 X_1 \otimes T_2 V_2 X_2 \otimes T_3 V_3 X_3 - A(T_1, T_2, T_3)\|_p^p. \end{aligned}$$

First, we regard T_1 as the sketching matrix for the remainder. Then by Lemma D.11 in [SWZ17] and Theorem 21.7.4, we have

$$\begin{aligned} &\|V_1 X_1' \otimes T_2 V_2 X_2' \otimes T_3 V_3 X_3' - A(I, T_2, T_3)\|_p^p \\ \lesssim\ & \alpha \min_{X_1 \in \mathbb{R}^{b_1 \times k}, X_2 \in \mathbb{R}^{b_2 \times k}, X_3 \in \mathbb{R}^{b_3 \times k}} \|V_1 X_1 \otimes T_2 V_2 X_2 \otimes T_3 V_3 X_3 - A(I, T_2, T_3)\|_p^p. \end{aligned}$$

Second, we regard T_2 as the sketching matrix for $V_1X_1 \otimes V_2X_2 \otimes T_3V_3X_3 - A(I, I, T_3)$. Then by Lemma D.11 in [SWZ17] and Theorem 21.7.4, we have

$$\begin{aligned}
&\|V_1X_1' \otimes V_2X_2' \otimes T_3V_3X_3' - A(I, I, T_3)\|_p^p \\
\lesssim\ &\alpha \min_{X_1 \in \mathbb{R}^{b_1 \times k}, X_2 \in \mathbb{R}^{b_2 \times k}, X_3 \in \mathbb{R}^{b_3 \times k}} \|V_1X_1 \otimes V_2X_2 \otimes T_3V_3X_3 - A(I, I, T_3)\|_p^p.
\end{aligned}$$

Third, we regard T_3 as the sketching matrix for $V_1X_1 \otimes V_2X_2 \otimes V_3X_3 - A$. Then by Lemma D.11 in [SWZ17] and Theorem 21.7.4, we have

$$\left\|\sum_{i=1}^k (V_1X_1')_i \otimes (V_2X_2')_i \otimes (V_3X_3')_i - A\right\|_p^p \lesssim \alpha \min_{X_1, X_2, X_3} \left\|\sum_{i=1}^k (V_1X_1)_i \otimes (V_2X_2)_i \otimes (V_3X_3)_i - A\right\|_p^p.$$

□

21.11.4 Solving small problems

Combining Section B.5 in [SWZ17] and the proof of Theorem 21.7.4, for any $p = a/b$ with a, b are integers, we can obtain the ℓ_p version of Theorem 21.7.4.

21.11.5 Bicriteria algorithm

We present several bicriteria algorithms with different tradeoffs. We first present an algorithm that runs in nearly linear time and outputs a solution with rank $\widetilde{O}(k^3)$ in Theorem 21.11.5. Then we show an algorithm that runs in $\mathrm{nnz}(A)$ time but outputs a solution with rank $\mathrm{poly}(k)$ in Theorem 21.11.6. Then we explain an idea which is able to decrease the cubic rank to quadratic, and thus we can obtain Theorem 21.11.7.

Theorem 21.11.5. *Given a 3rd order tensor $A \in \mathbb{R}^{n\times n\times n}$, for any $k \geq 1$, let $r = \widetilde{O}(k^3)$. There exists an algorithm which takes $\mathrm{nnz}(A) \cdot \widetilde{O}(k) + n\,\mathrm{poly}(k) + \mathrm{poly}(k)$ time and outputs three matrices $U, V, W \in \mathbb{R}^{n\times r}$ such that*

$$\left\| \sum_{i=1}^{r} U_i \otimes V_i \otimes W_i - A \right\|_p^p \leq \widetilde{O}(k^{3-p/2}) \log^3 n \min_{\mathrm{rank}-k\ A_k} \|A_k - A\|_p^p$$

holds with probability $9/10$.

Proof. We first choose three dense Cauchy transforms $S_i \in \mathbb{R}^{n^2\times s_i}$. According to Section 21.3.7, for each $i \in [3]$, $A_i S_i$ can be computed in $\mathrm{nnz}(A) \cdot \widetilde{O}(k)$ time. Then we apply Lemma 21.11.4. We obtain three matrices $Y_1 = T_1 A_1 S_1, Y_2 = T_2 A_2 S_2, Y_3 = T_3 A_3 S_3$ and a tensor $C = A(T_1, T_2, T_3)$. Note that for each $i \in [3]$, Y_i can be computed in $n\,\mathrm{poly}(k)$ time. Because $C = A(T_1, T_2, T_3)$ and $T_1, T_2, T_3 \in \mathbb{R}^{n\times \widetilde{O}(k)}$ are three sampling and rescaling matrices, C can be computed in $\mathrm{nnz}(A) + \widetilde{O}(k^3)$ time. At the end, we just need to run an ℓ_p-regression solver to find the solution for the problem:

$$\min_{X\in\mathbb{R}^{s_1\times s_2\times s_3}} \left\| \sum_{i=1}^{s_1}\sum_{j=1}^{s_2}\sum_{l=1}^{s_3} X_{i,j,l} (Y_1)_i \otimes (Y_2)_j \otimes (Y_3)_j \right\|_p^p,$$

where $(Y_1)_i$ denotes the i-th column of matrix Y_1. Since the size of the above problem is only $\mathrm{poly}(k)$, this can be solved in $\mathrm{poly}(k)$ time. □

Theorem 21.11.6. *Given a 3rd order tensor $A \in \mathbb{R}^{n \times n \times n}$, for any $k \geq 1$, let $r = \widetilde{O}(k^{15})$. There exists an algorithm that takes $\mathrm{nnz}(A) + n\,\mathrm{poly}(k) + \mathrm{poly}(k)$ time and outputs three matrices $U, V, W \in \mathbb{R}^{n \times r}$ such that*

$$\left\| \sum_{i=1}^{r} U_i \otimes V_i \otimes W_i - A \right\|_p^p \leq \mathrm{poly}(k, \log n) \min_{\mathrm{rank}-k \ A_k} \|A_k - A\|_p^p$$

holds with probability 9/10.

Proof. We first choose three sparse p-stable transforms $S_i \in \mathbb{R}^{n^2 \times s_i}$. According to Section 21.3.7, for each $i \in [3]$, $A_i S_i$ can be computed in $O(\mathrm{nnz}(A))$ time. Then we apply Lemma 21.11.4, and can obtain three matrices $Y_1 = T_1 A_1 S_1, Y_2 = T_2 A_2 S_2, Y_3 = T_3 A_3 S_3$ and a tensor $C = A(T_1, T_2, T_3)$. Note that for each $i \in [3]$, Y_i can be computed in $n\,\mathrm{poly}(k)$ time. Because $C = A(T_1, T_2, T_3)$ and $T_1, T_2, T_3 \in \mathbb{R}^{n \times \widetilde{O}(k)}$ are three sampling and rescaling matrices, C can be computed in $\mathrm{nnz}(A) + \widetilde{O}(k^3)$ time. At the end, we just need to run an ℓ_p-regression solver to find the solution to the problem,

$$\min_{X \in \mathbb{R}^{s_1 \times s_2 \times s_3}} \left\| \sum_{i=1}^{s_1} \sum_{j=1}^{s_2} \sum_{l=1}^{s_3} X_{i,j,l} (Y_1)_i \otimes (Y_2)_j \otimes (Y_3)_l - C \right\|_p^p,$$

where $(Y_1)_i$ denotes the i-th column of matrix Y_1. Since the size of the above problem is only $\mathrm{poly}(k)$, it can be solved in $\mathrm{poly}(k)$ time. □

Theorem 21.11.7. *Given a 3rd order tensor $A \in \mathbb{R}^{n \times n \times n}$, for any $k \geq 1$, $\epsilon \in (0, 1)$, let $r = \widetilde{O}(k^2)$. There exists an algorithm which takes $\mathrm{nnz}(A) \cdot \widetilde{O}(k) + n\,\mathrm{poly}(k) + \mathrm{poly}(k)$ time and outputs three matrices $U, V, W \in \mathbb{R}^{n \times r}$ such that*

$$\left\| \sum_{i=1}^{r} U_i \otimes V_i \otimes W_i - A \right\|_p^p \leq \widetilde{O}(k^{3-1.5p}) \log^3 n \min_{\mathrm{rank}-k \ A_k} \|A_k - A\|_p^p$$

holds with probability 9/10.

Proof. The proof is similar to Theorem 21.7.11. □

Algorithm 21.35 ℓ_p-Low Rank Approximation, Bicriteria Algorithm, rank-$\widetilde{O}(k^2)$, Input Sparsity Time

procedure LpBicriteriaAlgorithm(A, n, k) ▷ Corollary 21.11.8
  $s_1 \leftarrow s_2 \leftarrow s_3 \leftarrow \widetilde{O}(k)$.
  For each $i \in [3]$, choose $S_i \in \mathbb{R}^{n^2 \times s_i}$ to be the composition of a sparse p-stable transform and a dense p-stable transform. ▷ Part (I,II) of Theorem 21.11.2
  Compute $A_1 \cdot S_1$, $A_2 \cdot S_2$.
  For each $i \in [2]$, choose T_i to be a sampling and rescaling diagonal matrix according to the Lewis weights of $A_i S_i$, with $t_i = \widetilde{O}(k)$ nonzero entries.
  $C \leftarrow A(T_1, T_2, I)$.
  $B^{i+(j-1)s_1} \leftarrow \mathrm{vec}((T_1 A_1 S_1)_i \otimes (T_2 A_2 S_2)_j), \forall i \in [s_1], j \in [s_2]$.
  Form objective function $\min_W \|WB - C_3\|_1$.
  Run ℓ_p-regression solver to find $\widehat{W}$.
  Construct $\widehat{U}$ by copying $(A_1 S_1)_i$ to the (i,j)-th column of $\widehat{U}$.
  Construct $\widehat{V}$ by copying $(A_2 S_2)_j$ to the (i,j)-th column of $\widehat{V}$.
  return $\widehat{U}, \widehat{V}, \widehat{W}$.
end procedure

As for ℓ_1, notice that if we first apply a sparse Cauchy transform, we can reduce the rank of the matrix to poly(k). Theyn we can apply a dense Cauchy transform and further reduce the dimension, while only incurring another poly(k) factor in the approximation ratio. By combining sparse p-stable and dense p-stable transforms, we can improve the running time from $\mathrm{nnz}(A) \cdot \widetilde{O}(k)$ to be $\mathrm{nnz}(A)$ by losing some additional poly(k) factors in the approximation ratio.

Corollary 21.11.8. *Given a 3rd order tensor $A \in \mathbb{R}^{n \times n \times n}$, for any $k \geq 1$, $\epsilon \in (0,1)$, let $r = \widetilde{O}(k^2)$. There exists an algorithm which takes $\mathrm{nnz}(A) + n\,\mathrm{poly}(k) + \mathrm{poly}(k)$ time and*

outputs three matrices $U, V, W \in \mathbb{R}^{n \times r}$ *such that*

$$\left\| \sum_{i=1}^{r} U_i \otimes V_i \otimes W_i - A \right\|_p^p \leq \mathrm{poly}(k, \log n) \min_{\mathrm{rank}-k \ A_k} \|A_k - A\|_p^p$$

holds with probability $9/10$.

21.11.6 Algorithms

In this section, we show two different algorithms by using different kind of sketches. One is shown in Theorem 21.11.9 which gives a fast running time. Another one is shown in Theorem 21.11.10 which gives the best approximation ratio.

Theorem 21.11.9. *Given a 3rd tensor $A \in \mathbb{R}^{n\times n\times n}$, for any $k \geq 1$, there exists an algorithm which takes $O(\mathrm{nnz}(A)) + n\,\mathrm{poly}(k) + 2^{\widetilde{O}(k^2)}$ time and outputs three matrices $U, V, W \in \mathbb{R}^{n\times k}$ such that,*

$$\|U \otimes V \otimes W - A\|_p^p \leq \mathrm{poly}(k, \log n) \min_{\mathrm{rank}-k\ A'} \|A' - A\|_p^p.$$

holds with probability at least $9/10$.

Proof. First, we apply part (II) of Theorem 21.11.2. Then $A_i S_i$ can be computed in $O(\mathrm{nnz}(A))$ time. Second, we use Lemma 21.11.4 to reduce the size of the objective function from $O(n^3)$ to $\mathrm{poly}(k)$ in $n\,\mathrm{poly}(k)$ time by only losing a constant factor in approximation ratio. Third, we use Claim 21.3.6 to relax the objective function from entry-wise ℓ_p-norm to Frobenius norm, and this step causes us to lose some other $\mathrm{poly}(k)$ factors in approximation ratio. As a last step, we use Theorem 21.4.44 to solve the Frobenius norm objective function. □

Theorem 21.11.10. *Given a 3rd order tensor $A \in \mathbb{R}^{n\times n\times n}$, for any $k \geq 1$, there exists an algorithm that takes $n^{\widetilde{O}(k)} 2^{\widetilde{O}(k^3)}$ time and output three matrices $U, V, W \in \mathbb{R}^{n\times k}$ such that,*

$$\|U \otimes V \otimes W - A\|_p^p \leq \widetilde{O}(k^{3-1.5p}) \min_{\mathrm{rank}-k\ A'} \|A' - A\|_p^p.$$

holds with probability at least $9/10$.

Proof. First, we apply part (III) of Theorem 21.11.2. Then, guessing S_i requires $n^{\widetilde{O}(k)}$ time. Second, we use Lemma 21.11.4 to reduce the size of the objective from $O(n^3)$ to poly(k) in polynomial time while only losing a constant factor in approximation ratio. Third, we solve the small optimization problem. □

21.11.7 CURT decomposition

Theorem 21.11.11. *Given a 3rd order tensor $A \in \mathbb{R}^{n\times n\times n}$, let $k \geq 1$, and let $U_B, V_B, W_B \in \mathbb{R}^{n\times k}$ denote a rank-k, α-approximation to A. Then there exists an algorithm which takes $O(\mathrm{nnz}(A)) + O(n^2)\,\mathrm{poly}(k)$ time and outputs three matrices $C \in \mathbb{R}^{n\times c}$ with columns from A, $R \in \mathbb{R}^{n\times r}$ with rows from A, $T \in \mathbb{R}^{n\times t}$ with tubes from A, and a tensor $U \in \mathbb{R}^{c\times r\times t}$ with $\mathrm{rank}(U) = k$ such that $c = r = t = O(k \log k \log\log k)$, and*

$$\left\| \sum_{i=1}^{c}\sum_{j=1}^{r}\sum_{l=1}^{t} U_{i,j,l} \cdot C_i \otimes R_j \otimes T_l - A \right\|_p^p \leq \widetilde{O}(k^{3-1.5p})\alpha \min_{\mathrm{rank}-k\ A'} \|A' - A\|_p^p$$

holds with probability $9/10$.

Proof. We define

$$\mathrm{OPT} := \min_{\mathrm{rank}-k\ A'} \|A' - A\|_p^p.$$

We already have three matrices $U_B \in \mathbb{R}^{n\times k}$, $V_B \in \mathbb{R}^{n\times k}$ and $W_B \in \mathbb{R}^{n\times k}$ and these three matrices provide a rank-k, α approximation to A, i.e.,

$$\left\| \sum_{i=1}^{k} (U_B)_i \otimes (V_B)_i \otimes (W_B)_i - A \right\|_p^p \leq \alpha\,\mathrm{OPT}\,. \tag{21.51}$$

Let $B_1 = V_B^\top \odot W_B^\top \in \mathbb{R}^{k\times n^2}$ denote the matrix where the i-th row is the vectorization of $(V_B)_i \otimes (W_B)_i$. By Section B.3 in [SWZ17], we can compute $D_1 \in \mathbb{R}^{n^2\times n^2}$ which is a sampling and rescaling matrix corresponding to the Lewis weights of $B_1^\top$ in $O(n^2\,\mathrm{poly}(k))$ time, and there are $d_1 = O(k \log k \log\log k)$ nonzero entries on the diagonal of D_1. Let $A_i \in \mathbb{R}^{n\times n^2}$ denote the matrix obtained by flattening A along the i-th direction, for each $i \in [3]$.

Define $U^* \in \mathbb{R}^{n\times k}$ to be the optimal solution to $\min_{U\in\mathbb{R}^{n\times k}} \|UB_1 - A_1\|_p^p$, $\widehat{U} = A_1 D_1 (B_1 D_1)^\dagger \in \mathbb{R}^{n\times k}$, $V_0 \in \mathbb{R}^{n\times k}$ to be the optimal solution to $\min_{V\in\mathbb{R}^{n\times k}} \|V \cdot (\widehat{U}^\top \odot W_B^\top) - A_2\|_p^p$, and U' to be the optimal solution to $\min_{U\in\mathbb{R}^{n\times k}} \|UB_1D_1 - A_1D_1\|_p^p$.

By Claim 21.3.4, we have

$$\|\widehat{U}B_1D_1 - A_1D_1\|_p^p \leq d_1^{1-p/2}\|U'B_1D_1 - A_1D_1\|_p^p.$$

Due to Lemma E.11 and Lemma E.8 in [SWZ17], with constant probability, we have

$$\|\widehat{U}B_1 - A_1\|_p^p \leq d_1^{1-p/2}\alpha_{D_1}\|U^*B_1 - A_1\|_p^p, \tag{21.52}$$

where $\alpha_{D_1} = O(1)$.

Recall that $(\widehat{U}^\top \odot W_B^\top) \in \mathbb{R}^{k\times n^2}$ denotes the matrix where the i-th row is the vectorization of $\widehat{U}_i \otimes (W_B)_i$, $\forall i \in [k]$. Now, we can show,

$$\begin{aligned}
\|V_0 \cdot (\widehat{U}^\top \odot W_B^\top) - A_2\|_p^p &\leq \|\widehat{U}B_1 - A_1\|_p^p && \text{by } V_0 = \arg\min_{V\in\mathbb{R}^{n\times k}} \|V \cdot (\widehat{U}^\top \odot W_B^\top) - A_2\|_p^p \\
&\lesssim d_1^{1-p/2}\|U^*B_1 - A_1\|_p^p && \text{by Equation (21.52)} \\
&\leq d_1^{1-p/2}\|U_BB_1 - A_1\|_p^p && \text{by } U^* = \arg\min_{U\in\mathbb{R}^{n\times k}} \|UB_1 - A_1\|_p^p \\
&\leq O(d_1^{1-p/2})\alpha\,\mathrm{OPT}\,. && \text{by Equation (21.51)}
\end{aligned} \tag{21.53}$$

We define $B_2 = \widehat{U}^\top \odot W_B^\top$. We can compute $D_2 \in \mathbb{R}^{n^2\times n^2}$ which is a sampling and rescaling matrix corresponding to the ℓ_p Lewis weights of $B_2^\top$ in $O(n^2\,\mathrm{poly}(k))$ time, and there are $d_2 = O(k\log k\log\log k)$ nonzero entries on the diagonal of D_2.

Define $V^* \in \mathbb{R}^{n\times k}$ to be the optimal solution of $\min_{V\in\mathbb{R}^{n\times k}} \|VB_2 - A_2\|_p^p$, $\widehat{V} = A_2D_2(B_2D_2)^\dagger \in \mathbb{R}^{n\times k}$, $W_0 \in \mathbb{R}^{n\times k}$ to be the optimal solution of $\min_{W\in\mathbb{R}^{n\times k}} \|W\cdot(\widehat{U}^\top \odot \widehat{V}^\top) - A_3\|_p^p$, and V' to be the optimal solution of $\min_{V\in\mathbb{R}^{n\times k}} \|VB_2D_2 - A_2D_2\|_p^p$.

By Claim 21.3.4, we have

$$\|\widehat{V}B_2D_2 - A_2D_2\|_p^p \leq d_2^{1-p/2}\|V'B_2D_2 - A_2D_2\|_p^p.$$

Due to Lemma E.11 and Lemma E.8 in [SWZ17], with constant probability, we have

$$\|\widehat{V}B_2 - A_2\|_p^p \leq d_2^{1-p/2}\alpha_{D_2}\|V^*B_2 - A_2\|_p^p, \tag{21.54}$$

where $\alpha_{D_2} = O(1)$.

Recall that $(\widehat{U}^\top \odot \widehat{V}^\top) \in \mathbb{R}^{k\times n^2}$ denotes the matrix for which the i-th row is the vectorization of $\widehat{U}_i \otimes \widehat{V}_i$, $\forall i \in [k]$. Now, we can show,

$$\begin{aligned}
&\|W_0 \cdot (\widehat{U}^\top \odot \widehat{V}^\top) - A_3\|_p^p \\
\leq\ &\|\widehat{V}B_2 - A_2\|_p^p && \text{by } W_0 = \arg\min_{W\in\mathbb{R}^{n\times k}}\|W \cdot (\widehat{U}^\top \odot \widehat{V}^\top) - A_3\|_p^p \\
\lesssim\ &d_2^{1-p/2}\|V^*B_2 - A_2\|_p^p && \text{by Equation (21.54)} \\
\leq\ &d_2^{1-p/2}\|V_0B_2 - A_2\|_p^p && \text{by } V^* = \arg\min_{V\in\mathbb{R}^{n\times k}}\|VB_2 - A_2\|_p^p \\
\leq\ &O((d_1d_2)^{1-p/2})\alpha\,\mathrm{OPT}\,. && \text{by Equation (21.53)}
\end{aligned} \tag{21.55}$$

We define $B_3 = \widehat{U}^\top \odot \widehat{V}^\top$. We can compute $D_3 \in \mathbb{R}^{n^2\times n^2}$ which is a sampling and rescaling matrix corresponding to the ℓ_p Lewis weights of $B_3^\top$ in $O(n^2\,\mathrm{poly}(k))$ time, and there are $d_3 = O(k\log k\log\log k)$ nonzero entries on the diagonal of D_3.

Define $W^* \in \mathbb{R}^{n\times k}$ to be the optimal solution to $\min_{W\in\mathbb{R}^{n\times k}}\|WB_3 - A_3\|_p^p$, $\widehat{W} = A_3D_3(B_3D_3)^\dagger \in \mathbb{R}^{n\times k}$, and W' to be the optimal solution to $\min_{W\in\mathbb{R}^{n\times k}}\|WB_3D_3 - A_3D_3\|_p^p$.

By Claim 21.3.4, we have

$$\|\widehat{W}B_3D_3 - A_3D_3\|_p^p \leq d_3^{1-p/2}\|W'B_3D_3 - A_3D_3\|_p^p.$$

Due to Lemma E.11 and Lemma E.8 in [SWZ17], with constant probability, we have

$$\|\widehat{W}B_3 - A_3\|_p^p \leq d_3^{1-p/2}\alpha_{D_3}\|W^*B_3 - A_3\|_p^p, \tag{21.56}$$

where $\alpha_{D_3} = O(1)$. Now we can show,

$$\begin{aligned}
\|\widehat{W}B_3 - A_3\|_p^p &\lesssim d_3^{1-p/2}\|W^*B_3 - A_3\|_p^p, && \text{by Equation (21.56)} \\
&\leq d_3^{1-p/2}\|W_0B_3 - A_3\|_p^p, && \text{by } W^* = \arg\min_{W\in\mathbb{R}^{n\times k}}\|WB_3 - A_3\|_p^p \\
&\leq O((d_1d_2d_3)^{1-p/2})\alpha\,\mathrm{OPT}\,. && \text{by Equation (21.55)}
\end{aligned}$$

Thus, it implies,

$$\left\|\sum_{i=1}^{k}\widehat{U}_i\otimes\widehat{V}_i\otimes\widehat{W}_i - A\right\|_p^p \leq \mathrm{poly}(k,\log n)\,\mathrm{OPT}\,.$$

where $\widehat{U} = A_1D_1(B_1D_1)^\dagger$, $\widehat{V} = A_2D_2(B_2D_2)^\dagger$, $\widehat{W} = A_3D_3(B_3D_3)^\dagger$.

□

21.12 Robust Subspace Approximation (Asymmetric Norms for Arbitrary Tensors)

Recently, [CW15b] and [CW15a] study the linear regression problem and low-rank approximation problem under M-Estimator loss functions. In this section, we extend the matrix version of the low rank approximation problem to tensors, i.e., in particular focusing on tensor low-rank approximation under M-Estimator norms. Note that M-Estimators are very different from Frobenius norm and Entry-wise ℓ_1 norm, which are symmetric norms. Namely, flattening the tensor objective function along any of the dimensions does not change the cost if the norm is Frobenius or Entry-wise ℓ_1-norm. However, for M-Estimator norms, we cannot flatten the tensor along all three dimensions. This property makes the tensor low-rank approximation problem under M-Estimator norms more difficult. This section can be split into two independent parts. Section 21.12.2 studies the ℓ_1-ℓ_2-ℓ_2 norm setting, and Section 21.12.3 studies the ℓ_1-ℓ_1-ℓ_2 norm setting.

21.12.1 Preliminaries

Definition 21.12.1 (Nice functions for M-Estimators, $\mathcal{M}_2$, $\mathcal{L}_p$, [CW15a])**.** We say an M-Estimator is **nice** if $M(x) = M(-x)$, $M(0) = 0$, M is non-decreasing in $|x|$, there is a constant $C_M > 0$ and a constant $p \geq 1$ so that for all $a, b \in \mathbb{R}_{>0}$ with $a \geq b$, we have

$$C_m \frac{|a|}{|b|} \leq \frac{M(a)}{M(b)} \leq (\frac{a}{b})^p,$$

and also that $M(x)^{\frac{1}{p}}$ is subadditive, that is, $M(x+y)^{\frac{1}{p}} \leq M(x)^{\frac{1}{p}} + M(y)^{\frac{1}{p}}$.

Let $\mathcal{M}_2$ denote the set of such nice M-estimators, for $p = 2$. Let $\mathcal{L}_p$ denote M-Estimators with $M(x) = |x|^p$ and $p \in [1, 2)$.

21.12.2 ℓ_1-Frobenius (a.k.a ℓ_1-ℓ_2-ℓ_2) norm

Section 21.12.2.1 presents basic definitions and facts for the ℓ_1-ℓ_2-ℓ_2 norm setting. Section 21.12.2.2 introduces some useful tools. Section 21.12.2.3 presents the "no dilation" and "no contraction" bounds, which are the key ideas for reducing the problem to a "generalized" Frobenius norm low rank approximation problem. Finally, we provide our algorithms in Section 21.12.2.6.

21.12.2.1 Definitions

We first give the definition for the v-norm of a tensor, and then give the definition of the v-norm for a matrix and a weighted version of the v-norm for a matrix.

Definition 21.12.2 (Tensor v-norm)**.** For an $n \times n \times n$ tensor A, we define the v-norm of A, denoted $\|A\|_v$, to be

$$\left(\sum_{i=1}^{n} M(\|A_{i,*,*}\|_F) \right)^{1/p},$$

where $A_{i,*,*}$ is the i-th face of A (along the 1st direction), and p is a parameter associated with the function $M()$, which defines a nice M-Estimator.

Definition 21.12.3 (Matrix v-norm)**.** For an $n \times d$ matrix A, we define the v-norm of A, denoted $\|A\|_v$, to be

$$\sum_{i=1}^{n} M(\|A_{i,*}\|_2)^{1/p},$$

where $A_{i,*}$ is the i-th row of A, and p is a parameter associated with the function $M()$, which defines a nice M-Estimator.

Definition 21.12.4. Given matrix $A \in \mathbb{R}^{n\times d}$, let $A_{i,*}$ denote the i-th row of A. Let $T_S \subset [n]$ denote the indices i such that e_i is chosen for S. Using a probability vector q and a sampling and rescaling matrix $S \in \mathbb{R}^{n\times n}$ from q, we will estimate $\|A\|_v$ using S and a re-weighted version, $\|S\cdot\|_{v,w'}$ of $\|\cdot\|_v$, with

$$\|SA\|_{v,w'} = \left(\sum_{i\in T_S} w_i' M(\|A_{i,*}\|_2)\right)^{1/p},$$

where $w_i' = w_i/q_i$. Since w' is generally understood, we will usually just write $\|SA\|_v$. We will also need an "entrywise row-weighted" version :

$$|||SA||| = \left(\sum_{i\in T_S} \frac{w_i}{q_i}\|A_{i,*}\|_M^p\right)^{1/p} = \left(\sum_{i\in T_S, j\in[d]} \frac{w_i}{q_i} M(A_{i,j})\right)^{1/p},$$

where $A_{i,j}$ denotes the entry in the i-th row and j-th column of A.

Fact 21.12.1. *For $p = 1$, for any two matrices A and B, we have $\|A+B\|_v \le \|A\|_v + \|B\|_v$. For any two tensors A and B, we have $\|A+B\|_v \le \|A\|_v + \|B\|_v$.*

21.12.2.2 Sampling and rescaling sketches

Note that Lemmas 42 and 44 in [CW15a] are stronger than stated. In particular, we do not need to assume X is a square matrix. For any $m \ge z$, if $X \in \mathbb{R}^{d\times m}$, then we have the same result.

Lemma 21.12.2 (Lemma 42 in [CW15a])**.** *Let $\rho > 0$ and integer $z > 0$. For sampling matrix S, suppose for a given $y \in \mathbb{R}^d$ with failure probability δ it holds that $\|SAy\|_M = (1\pm 1/10)\|Ay\|_M$. There is $K_1 = O(z^2/C_M)$ so that with failure probability $\delta(K_N/C_M)^{(1+p)d}$, for a constant K_N, any rank-z matrix $X \in \mathbb{R}^{d\times m}$ has the property that if $\|AX\|_v \ge K_1\rho$, then $\|SAX\|_v \ge \rho$, and that if $\|AX\|_v \le \rho/K_1$, then $\|SAX\|_v \le \rho$.*

Lemma 21.12.3 (Lemma 44 in [CW15a]). *Let $\delta, \rho > 0$ and integer $z > 0$. Given matrix $A \in \mathbb{R}^{n\times d}$, there exists a sampling and rescaling matrix $S \in \mathbb{R}^{n\times n}$ with $r = O(\gamma(A, M, w)\epsilon^{-2}dz^2 \log(z/\epsilon)\log(1/\delta))$ nonzero entries such that, with probability at least $1-\delta$, for any* rank-z *matrix $X \in \mathbb{R}^{d\times m}$, we have either*

$$\|SAX\|_v \geq \rho,$$

or

$$(1-\epsilon)\|AX\|_v - \epsilon\rho \leq \|SAX\|_v \leq (1+\epsilon)\|AX\|_v + \epsilon\rho.$$

Lemma 21.12.4 (Lemma 43 in [CW15a]). *For $r > 0$, let $\widehat{r} = r/\gamma(A, M, w)$, and let $q \in \mathbb{R}^n$ have*

$$q_i = \min\{1, \widehat{r}\gamma_i(A, M, w)\}.$$

Let S be a sampling and rescaling matrix generated using q, with weights as usual $w_i' = w_i/q_i$. Let $W \in \mathbb{R}^{d\times z}$, and $\delta > 0$. There is an absolute constant C so that for $\widehat{r} \geq Cz\log(1/\delta)/\epsilon^2$, with probability at least $1-\delta$, we have

$$(1-\epsilon)\|AW\|_{v,w} \leq \|SAW\|_{v,w'} \leq (1+\epsilon)\|AW\|_{v,w}.$$

21.12.2.3 No dilation and no contraction

Lemma 21.12.5. *Given matrices $A \in \mathbb{R}^{n\times m}$, $U \in \mathbb{R}^{n\times d}$, let $V^* = \underset{\text{rank}-k\ V\in\mathbb{R}^{d\times m}}{\arg\min} \|UV - A\|_v$. If $S \in \mathbb{R}^{s\times n}$ has at most c_1-dilation on $UV^* - A$, i.e.,*

$$\|S(UV^* - A)\|_v \leq c_1\|UV^* - A\|_v,$$

and it has at most c_2-contraction on U, i.e.,

$$\forall x \in \mathbb{R}^d, \|SUx\|_v \geq \frac{1}{c_2}\|Ux\|_v,$$

then S has at most $(c_2, c_1 + \frac{1}{c_2})$-contraction on (U, A), i.e.,

$$\forall \text{ rank}-k\ V \in \mathbb{R}^{d\times m}, \|SUV - SA\|_v \geq \frac{1}{c_2}\|UV - A\|_v - (c_1 + \frac{1}{c_2})\|UV^* - A\|_v.$$

Proof. Let $A \in \mathbb{R}^{n\times m}$, $U \in \mathbb{R}^{n\times d}$ and $S \in \mathbb{R}^{s\times n}$ be the same as that described in the lemma. Let $(V - V^*)_j$ denote the j-th column of $V - V^*$. Then $\forall$ rank $-k$ $V \in \mathbb{R}^{d\times m}$,

$$\begin{aligned}
\|SUV - SA\|_v &\geq \|SUV - SUV^*\|_v - \|SUV^* - SA\|_v \\
&\geq \|SUV - SUV^*\|_v - c_1\|UV^* - A\|_v \\
&= \|SU(V - V^*)\|_v - c_1\|UV^* - A\|_v \\
&= \sum_{j=1}^{m} \|SU(V - V^*)_j\|_v - c_1\|UV^* - A\|_v \\
&\geq \sum_{j=1}^{m} \frac{1}{c_2}\|U(V - V^*)_j\|_v - c_1\|UV^* - A\|_v \\
&= \frac{1}{c_2}\|UV - UV^*\|_v - c_1\|UV^* - A\|_v \\
&\geq \frac{1}{c_2}\|UV - A\|_v - \frac{1}{c_2}\|UV^* - A\|_v - c_1\|UV^* - A\|_v \\
&= \frac{1}{c_2}\|UV - A\|_v - \left((\frac{1}{c_2} + c_2)\|UV^* - A\|_v\right),
\end{aligned}$$

where the first inequality follows by the triangle inequality, the second inequality follows since S has at most c_1 dilation on $UV^* - A$, the third inequality follows since S has at most c_2 contraction on U, and the fourth inequality follows by the triangle inequality. □

Claim 21.12.6. *Given matrix $A \in \mathbb{R}^{n\times m}$, for any distribution $p = (p_1, p_2, \cdots, p_n)$ define random variable X such that $X = \|A_i\|_2/p_i$ with probability p_i where A_i is the i-th row of matrix A. Then take m independent samples $X^1, X^2, \cdots, X^m$, and let $Y = \frac{1}{m}\sum_{j=1}^m X^j$. We have*

$$\Pr[Y \leq 1000\|A\|_v] \geq .999.$$

Proof. We can compute the expectation of X^j, for any $j \in [m]$,

$$\mathbb{E}[X^j] = \sum_{i=1}^n \frac{\|A_i\|_2}{p_i} \cdot p_i = \|A\|_v.$$

Then $\mathbb{E}[Y] = \frac{1}{m}\sum_{j=1}^m \mathbb{E}[X^j] = \|A\|_v$. Using Markov's inequality, we have

$$\Pr[Y \geq \|A\|_v] \leq .001.$$

□

Lemma 21.12.7. *For any fixed $U^* \in \mathbb{R}^{n\times d}$ and* rank-k *$V^* \in \mathbb{R}^{d\times m}$ with $d = \mathrm{poly}(k)$, there exists an algorithm that takes $\mathrm{poly}(n, d)$ time to compute a sampling and rescaling diagonal matrix $S \in \mathbb{R}^{n\times n}$ with $s = \mathrm{poly}(k)$ nonzero entries such that, with probability at least .999, we have: for all* rank-k *$V \in \mathbb{R}^{d\times m}$,*

$$\|U^*V^* - U^*V\|_v \lesssim \|SU^*V^* - SU^*V\|_v \lesssim \|U^*V^* - U^*V\|_v.$$

Lemma 21.12.8 (No dilation)**.** *Given matrices $A \in \mathbb{R}^{n\times m}$, $U^* \in \mathbb{R}^{n\times d}$ with $d = \mathrm{poly}(k)$, define $V^* \in \mathbb{R}^{d\times m}$ to be the optimal solution $\min_{\mathrm{rank}-k\ V\in\mathbb{R}^{d\times m}} \|U^*V - A\|_v$. Choose a sampling and rescaling diagonal matrix $S \in \mathbb{R}^{n\times n}$ with $s = \mathrm{poly}(k)$ according to Lemma 21.12.4. Then with probability at least .99, we have: for all* rank-k *$V \in \mathbb{R}^{d\times m}$,*

$$\|SU^*V - SA\|_v \lesssim \|U^*V^* - U^*V\|_v + O(1)\|U^*V^* - A\|_v \lesssim \|U^*V - A\|_v.$$

Proof. Using Claim 21.12.6 and Lemma 21.12.7, we have with probability at least .99, for all rank-k $V \in \mathbb{R}^{d\times m}$,

$$\begin{aligned}
&\|SU^*V - SA\|_v \\
\leq\ &\|SU^*V - SU^*V^*\|_v + \|SU^*V^* - SA\|_v && \text{by triangle inequality} \\
\lesssim\ &\|SU^*V - SU^*V^*\|_v + O(1)\|U^*V^* - A\|_v && \text{by Claim 21.12.6} \\
\lesssim\ &\|U^*V - U^*V^*\|_v + O(1)\|U^*V^* - A\|_v && \text{by Lemma 21.12.7} \\
\lesssim\ &\|U^*V - A\|_v + \|U^*V^* - A\|_v + O(1)\|U^*V^* - A\|_v && \text{by triangle inequality} \\
\lesssim\ &\|U^*V - A\|_v.
\end{aligned}$$

□

Lemma 21.12.9 (No contraction)**.** *Given matrices $A \in \mathbb{R}^{n\times m}$, $U^* \in \mathbb{R}^{n\times d}$ with $d = \mathrm{poly}(k)$, define $V^* \in \mathbb{R}^{d\times m}$ to be the optimal solution $\min_{\mathrm{rank}-k\ V\in\mathbb{R}^{d\times m}} \|U^*V - A\|_v$. Choose a sampling and rescaling diagonal matrix $S \in \mathbb{R}^{n\times n}$ with $s = \mathrm{poly}(k)$ according to Lemma 21.12.4. Then with probability at least .99, we have: for all* rank-k $V \in \mathbb{R}^{d\times m}$,

$$\|U^*V - A\|_v \lesssim \|SU^*V - SA\|_v + O(1)\|U^*V^* - A\|_v.$$

Proof. This follows by Lemma 21.12.5, Claim 21.12.6 and Lemma 21.12.8. □

21.12.2.4 Oblivious sketches, MSketch

In this section, we recall a concept called M-sketches for M-estimators which is defined in [CW15b]. M-sketch is an oblivious sketch for matrices.

Theorem 21.12.10 (Theorem 3.1 in [CW15b]). *Let* OPT *denote* $\min_{x\in\mathbb{R}^d}\|Ax-b\|_G$. *There is an algorithm that in* $O(\mathrm{nnz}(A))+\mathrm{poly}(d\log n)$ *time, with constant probability finds* x' *such that* $\|Ax'-b\|_G \leq O(1)\,\mathrm{OPT}$.

Definition 21.12.5 (M-Estimator sketches or MSketch [CW15b]). Given parameters $N, n, m, b > 1$, define $h_{\max} = \lfloor \log_b(n/m) \rfloor$, $\beta = (b - b^{-h_{\max}})/(b-1)$ and $s = Nh_{\max}$. For each $p \in [n]$, σ_p, g_p, h_p are generated (independently) in the following way,

$$\begin{aligned} &\sigma_p \leftarrow \pm 1, && \text{chosen with equal probability,} \\ &g_p \in [N], && \text{chosen with equal probability,} \\ &h_p \leftarrow t, && \text{chosen with probability } 1/(\beta b^t) \text{ for } t \in \{0, 1, \cdots h_{\max}\}. \end{aligned}$$

For each $p \in [n]$, we define $j_p = g_p + Nh_p$. Let $w \in \mathbb{R}^s$ denote the scaling vector such that, for each $j \in [s]$,

$$w_j = \begin{cases} \beta b^{h_p}, & \text{if there exists } p \in [n] \text{ s.t.} j = j_p, \\ 0 & \text{otherwise.} \end{cases}$$

Let $\overline{S} \in \mathbb{R}^{Nh_{\max} \times n}$ be such that, for each $j \in [s]$,for each $p \in [n]$,

$$\overline{S}_{j,p} = \begin{cases} \sigma_p, & \text{if } j = g_p + N \cdot h_p, \\ 0, & \text{otherwise.} \end{cases}$$

Let D_w denote the diagonal matrix where the i-th entry on the diagonal is the i-th entry of w. Let $S = D_w\overline{S}$. We say $(\overline{S}, w)$ or S is an MSketch.

Definition 21.12.6 (Tensor $\|\|\|_{v,w}$-norm). For a tensor $A \in \mathbb{R}^{d\times n_1\times n_2}$ and a vector $w \in$, we define

$$\|A\|_{v,w} = \sum_{i=1}^{d} w_i \|A_{i,*,*}\|_F.$$

Let $(\overline{S}, w)$ denote an MSketch, and let $S = D_w \overline{S}$. If v corresponds to a scale-invariant M-Estimator, then for any three matrices U, V, W, we have the following,

$$\|(\overline{S}U) \otimes V \otimes W\|_{v,w} = \|(D_w \overline{S}U) \otimes V \otimes W\|_v = \|(SU) \otimes V \otimes W\|_v.$$

Fact 21.12.11. *For a tensor $A \in \mathbb{R}^{n \times n \times n}$, let $S \in \mathbb{R}^{s \times n}$ denote an* MSketch *(defined in 21.12.5) with $s = \mathrm{poly}(k, \log n)$. Then SA can be computed in $O(\mathrm{nnz}(A))$ time.*

Lemma 21.12.12. *For any fixed $U^* \in \mathbb{R}^{n \times d}$ and* rank-k *$V^* \in \mathbb{R}^{d \times m}$ with $d = \mathrm{poly}(k)$, let $S \in \mathbb{R}^{s \times n}$ denote an* MSketch *(defined in Definition 21.12.5) with $s = \mathrm{poly}(k, \log n)$ rows. Then with probability at least .999, we have: for all* rank-k *$V \in \mathbb{R}^{d \times m}$,*

$$\|U^*V^* - U^*V\|_v \lesssim \|SU^*V^* - SU^*V\|_v \lesssim \|U^*V^* - U^*V\|_v.$$

Lemma 21.12.13 (No dilation, Theorem 3.4 in [CW15b])**.** *Given matrices $A \in \mathbb{R}^{n \times m}$, $U^* \in \mathbb{R}^{n \times d}$ with $d = \mathrm{poly}(k)$, define $V^* \in \mathbb{R}^{d \times m}$ to be the optimal solution to $\min_{\mathrm{rank}-k \ V \in \mathbb{R}^{d \times m}} \|U^*V - A\|_v$. Choose an* MSketch *$S \in \mathbb{R}^{s \times n}$ with $s = \mathrm{poly}(k, \log n)$ according to Definition 21.12.5. Then with probability at least .99, we have: for all* rank-k *$V \in \mathbb{R}^{d \times m}$,*

$$\|SU^*V - SA\|_v \lesssim \|U^*V^* - U^*V\|_v + O(1)\|U^*V^* - A\|_v \lesssim \|U^*V - A\|_v.$$

Lemma 21.12.14 (No contraction)**.** *Given matrices $A \in \mathbb{R}^{n \times m}$, $U^* \in \mathbb{R}^{n \times d}$ with $d = \mathrm{poly}(k)$, define $V^* \in \mathbb{R}^{d \times m}$ to be the optimal solution to $\min_{\mathrm{rank}-k \ V \in \mathbb{R}^{d \times m}} \|U^*V - A\|_v$. Choose an* MSketch *$S \in \mathbb{R}^{s \times n}$ with $s = \mathrm{poly}(k, \log n)$ according to Definition 21.12.5. Then with probability at least .99, we have: for all* rank-k *$V \in \mathbb{R}^{d \times m}$,*

$$\|U^*V - A\|_v \lesssim \|SU^*V - SA\|_v + O(1)\|U^*V^* - A\|_v.$$

21.12.2.5 Running time analysis

Lemma 21.12.15. *Given a tensor $A \in \mathbb{R}^{n\times d\times d}$, let $S \in \mathbb{R}^{s\times n}$ denote an* MSketch *with s rows. Let SA denote a tensor that has size $s \times d \times d$. For each $i \in \{2, 3\}$, let $(SA)_i \in \mathbb{R}^{d\times ds}$ denote a matrix obtained by flattening tensor SA along the i-th dimension. For each $i \in \{2, 3\}$, let $S_i \in \mathbb{R}^{ds\times s_i}$ denote a CountSketch transform with s_i columns. For each $i \in \{2, 3\}$, let $T_i \in \mathbb{R}^{t_i\times d}$ denote a CountSketch transform with t_i rows. Then*

(I) *For each $i \in \{2, 3\}$, $(SA)_i S_i$ can be computed in $O(\mathrm{nnz}(A))$ time.*

(II) *For each $i \in \{2, 3\}$, $T_i(SA)_i S_i$ can be computed in $O(\mathrm{nnz}(A))$ time.*

Proof. Proof of Part (I). First note that $(SA)_2 S_2$ has size $n\times S_2$. Thus for each $i \in [d], j \in [s_2]$, we have,

$$\begin{aligned}
((SA)_2 S_2)_{i,j} &= \sum_{x'=1}^{ds} ((SA)_2)_{i,x'} (S_2)_{x',j} && \text{by } (SA)_2 \in \mathbb{R}^{d\times ds}, S_2 \in \mathbb{R}^{ds\times s_2} \\
&= \sum_{y=1}^{d}\sum_{z=1}^{s} ((SA)_2)_{i,(y-1)s+z} (S_2)_{(y-1)s+z,j} \\
&= \sum_{y=1}^{d}\sum_{z=1}^{s} (SA)_{z,i,y} (S_2)_{(y-1)s+z,j} && \text{by unflattening} \\
&= \sum_{y=1}^{d}\sum_{z=1}^{s} \left(\sum_{x=1}^{n} S_{z,x} A_{x,i,y}\right) (S_2)_{(y-1)s+z,j} \\
&= \sum_{y=1}^{d}\sum_{z=1}^{s}\sum_{x=1}^{n} S_{z,x} \cdot A_{x,i,y} \cdot (S_2)_{(y-1)s+z,j}.
\end{aligned}$$

For each nonzero entry $A_{x,i,y}$, there is only one z such that $S_{z,x}$ is nonzero. Thus there is only one j such that $(S_2)_{(y-1)s+z,j}$ is nonzero. It means that $A_{x,i,y}$ can only affect one entry of $((SA)_2 S_2)_{i,j}$. Thus, $(SA)_2 S_2$ can be computed in $O(\mathrm{nnz}(A))$ time. Similarly, we can compute $(SA)_3 S_3$ in $O(\mathrm{nnz}(A))$ time.

Proof of Part (II). Note that $T_2(SA)_2S_2$ has size $t_2 \times s_2$. Thus for each $i \in [t_2], j \in [s_2]$, we have,

$$\begin{aligned}
(T_2(SA)_2S_2)_{i,j} &= \sum_{x=1}^{d}\sum_{y'=1}^{ds}(T_2)_{i,x}((SA)_2)_{x,y'}(S_2)_{y',j} && \text{by } (SA)_2 \in \mathbb{R}^{d\times ds} \\
&= \sum_{x=1}^{d}\sum_{y=1}^{d}\sum_{z=1}^{s}(T_2)_{i,x}((SA)_2)_{x,(y-1)s+z}(S_2)_{(y-1)s+z,j} \\
&= \sum_{x=1}^{d}\sum_{y=1}^{d}\sum_{z=1}^{s}(T_2)_{i,x}(SA)_{z,x,y}(S_2)_{(y-1)s+z,j} && \text{by unflattening} \\
&= \sum_{x=1}^{d}\sum_{y=1}^{d}\sum_{z=1}^{s}(T_2)_{i,x}\left(\sum_{w=1}^{n}S_{z,w}A_{w,x,y}\right)(S_2)_{(y-1)s+z,j} \\
&= \sum_{x=1}^{d}\sum_{y=1}^{d}\sum_{z=1}^{s}\sum_{w=1}^{n}(T_2)_{i,x}\cdot S_{z,w}\cdot A_{w,x,y}\cdot(S_2)_{(y-1)s+z,j}.
\end{aligned}$$

For each nonzero entry $A_{w,x,y}$, there is only one z such that $S_{z,w}$ is nonzero. There is only one i such that $(T_2)_{i,x}$ is nonzero. Since there is only one z to make $S_{z,w}$ nonzero, there is only one j, such that $(S_2)_{(y-1)s+z,j}$ is nonzero. Thus, $T_2(SA)_2S_2$ can be computed in $O(\text{nnz}(A))$ time. Similarly, we can compute $T_3(SA)_3S_3$ in $O(\text{nnz}(A))$ time. □

21.12.2.6 Algorithms

We first give a "warm-up" algorithm in Theorem 21.12.16 by using a sampling and rescaling matrix. Then we improve the running time to be polynomial in all the parameters by using an oblivious sketch, and thus we obtain Theorem 21.12.17.

Theorem 21.12.16. *Given a 3rd order tensor $A \in \mathbb{R}^{n\times n\times n}$, for any $k \geq 1$, let $r = O(k^2)$. There exists an algorithm which takes $n^{\text{poly}(k)}$ time and outputs three matrices $U, V, W \in \mathbb{R}^{n\times r}$*

Algorithm 21.36 ℓ_1-Frobenius(ℓ_1-ℓ_2-ℓ_2) Low-rank Approximation Algorithm, poly(k) approximation

procedure L122TensorLowRankApprox(A, n, k) ▷ Theorem 21.12.16
  $\epsilon \leftarrow 0.1$.
  $s \leftarrow \text{poly}(k, 1/\epsilon)$.
  Guess a sampling and rescaling matrix $S \in \mathbb{R}^{s \times n}$.
  $s_2 \leftarrow s_3 \leftarrow O(k/\epsilon)$.
  $r \leftarrow s_2 s_3$.
  Choose sketching matrices $S_2 \in \mathbb{R}^{ns \times s_2}$, $S_3 \in \mathbb{R}^{ns \times s_3}$.
  Compute $(SA)_2 S_2, (SA)_3 S_3$.
  Form $\widetilde{V} \in \mathbb{R}^{n \times r}$ by repeating $(SA)_2 S_2$ s_3 times according to Equation (21.64).
  Form $\widetilde{W} \in \mathbb{R}^{n \times r}$ by repeating $(SA)_3 S_3$ s_2 times according to Equation (21.65).
  Form objective function $\min_{U \in \mathbb{R}^{n \times r}} \|U \cdot (\widetilde{V}^\top \odot \widetilde{W}^\top) - A_1\|_F$.
  Use a linear regression solver to find a solution $\widetilde{U}$.
  Take the best solution found over all guesses.
  return $\widetilde{U}$, $\widetilde{V}$, $\widetilde{W}$.
end procedure

such that

$$\|U \otimes V \otimes W - A\|_v \leq \text{poly}(k) \min_{\text{rank}-k\ A'} \|A' - A\|_v,$$

holds with probability at least $9/10$.

Proof. We define OPT as follows,

$$\text{OPT} = \min_{U,V,W \in \mathbb{R}^{n \times k}} \|U \otimes V \otimes W - A\|_v = \min_{U,V,W \in \mathbb{R}^{n \times k}} \left\| \sum_{i=1}^k U_i \otimes V_i \otimes W_i - A \right\|_v.$$

Let $A_1 \in \mathbb{R}^{n \times n^2}$ denote the matrix obtained by flattening tensor A along the 1st dimension. Let $U^* \in \mathbb{R}^{n \times k}$ denote the optimal solution. We fix $U^* \in \mathbb{R}^{n \times k}$, and consider this objective function,

$$\min_{V,W \in \mathbb{R}^{n \times k}} \|U^* \otimes V \otimes W - A\|_v \equiv \min_{V,W \in \mathbb{R}^{n \times k}} \left\|U^* \cdot (V^\top \odot W^\top) - A_1\right\|_v, \tag{21.57}$$

which has cost at most OPT, and where $V^\top \odot W^\top \in \mathbb{R}^{k\times n^2}$ denotes the matrix for which the i-th row is a vectorization of $V_i \otimes W_i, \forall i \in [k]$. (Note that $V_i \in \mathbb{R}^n$ is the i-th column of matrix $V \in \mathbb{R}^{n\times k}$). Choose a sampling and rescaling diagonal matrix $S \in \mathbb{R}^{n\times n}$ according to U^*, which has $s = \text{poly}(k)$ non-zero entries. Using S to sketch on the left of the objective function when U^* is fixed (Equation (21.57)), we obtain a smaller problem,

$$\min_{V,W\in\mathbb{R}^{n\times k}} \|(SU^*)\otimes V\otimes W - SA\|_v \equiv \min_{V,W\in\mathbb{R}^{n\times k}} \left\|SU^*\cdot(V^\top \odot W^\top) - SA_1\right\|_v . \tag{21.58}$$

Let V', W' denote the optimal solution to the above problem, i.e.,

$$V', W' = \arg\min_{V,W\in\mathbb{R}^{n\times k}} \|(SU^*)\otimes V\otimes W - SA\|_v .$$

Then using properties (no dilation Lemma 21.12.8 and no contraction Lemma 21.12.9) of S, we have

$$\|U^*\otimes V'\otimes W' - A\|_v \le \alpha\,\text{OPT}\,.$$

where α is an approximation ratio determined by S.

By definition of $\|\cdot\|_v$ and $\|\cdot\|_2 \le \|\cdot\|_1 \le \sqrt{\dim}\|\cdot\|_2$, we can rewrite Equation (21.58) in the following way,

$$\begin{aligned}
&\|(SU^*)\otimes V\otimes W - SA\|_v \\
&= \sum_{i=1}^{s}\left(\sum_{j=1}^{n}\sum_{l=1}^{n}\left(((SU^*)\otimes V\otimes W)_{i,j,l} - (SA)_{i,j,l}\right)^2\right)^{\frac{1}{2}} \\
&\le \sqrt{s}\left(\sum_{i=1}^{s}\sum_{j=1}^{n}\sum_{l=1}^{n}\left(((SU^*)\otimes V\otimes W)_{i,j,l} - (SA)_{i,j,l}\right)^2\right)^{\frac{1}{2}} \\
&= \sqrt{s}\,\|(SU^*)\otimes V\otimes W - SA\|_F\,.
\end{aligned} \tag{21.59}$$

Given the above properties of S and Equation (21.59), for any $\beta \geq 1$, let V'', W'' denote a β-approximate solution of $\min_{V,W \in \mathbb{R}^{n \times k}} \|(SU^*) \otimes V \otimes W - SA\|_F$, i.e.,

$$\|(SU^*) \otimes V'' \otimes W'' - SA\|_F \leq \beta \cdot \min_{V,W \in \mathbb{R}^{n \times k}} \|(SU^*) \otimes V \otimes W - SA\|_F . \tag{21.60}$$

Then,

$$\|U^* \otimes V'' \otimes W'' - A\|_v \leq \sqrt{s}\alpha\beta \cdot \mathrm{OPT} . \tag{21.61}$$

In the next few paragraphs we will focus on solving Equation (21.60). We start by fixing $W^* \in \mathbb{R}^{n \times k}$ to be the optimal solution of

$$\min_{V,W \in \mathbb{R}^{n \times k}} \|(SU^*) \otimes V \otimes W - SA\|_F .$$

We use $(SA)_2 \in \mathbb{R}^{n \times ns}$ to denote the matrix obtained by flattening the tensor $SA \in \mathbb{R}^{s \times n \times n}$ along the second direction. We use $Z_2 = (SU^*)^\top \odot (W^*)^\top \in \mathbb{R}^{k \times ns}$ to denote the matrix where the i-th row is the vectorization of $(SU^*)_i \otimes W_i^*$. We can consider the following objective function,

$$\min_{V \in \mathbb{R}^{n \times k}} \|VZ_2 - (SA)_2\|_F .$$

Choosing a sketching matrix $S_2 \in \mathbb{R}^{ns \times s_2}$ with $s_2 = O(k/\epsilon)$ gives a smaller problem,

$$\min_{V \in \mathbb{R}^{n \times k}} \|VZ_2S_2 - (SA)_2S_2\|_F .$$

Letting $\widehat{V} = (SA)_2 S_2 (Z_2 S_2)^\dagger \in \mathbb{R}^{n\times k}$, then

$$\begin{aligned}
\|\widehat{V} Z_2 - (SA)_2\|_F &\leq (1+\epsilon) \min_{V\in\mathbb{R}^{n\times k}} \|V Z_2 - (SA)_2\|_F \\
&= (1+\epsilon) \min_{V\in\mathbb{R}^{n\times k}} \|V((SU^*)^\top \odot (W^*)^\top) - (SA)_2\|_F \\
&= (1+\epsilon) \min_{V\in\mathbb{R}^{n\times k}} \|(SU^*)\otimes V \otimes W^* - SA\|_F \qquad \text{by unflattening} \\
&= (1+\epsilon) \min_{V,W\in\mathbb{R}^{n\times k}} \|(SU^*)\otimes V \otimes W - SA\|_F. \qquad \text{by definition of } W^*
\end{aligned} \tag{21.62}$$

We define $D_2 \in \mathbb{R}^{n^2\times n^2}$ to be a diagonal matrix obrained by copying the $n\times n$ identity matrix s times on n diagonal blocks of D_2. Then it has ns nonzero entries. Thus, D_2 also can be thought of as a matrix that has size $n^2 \times ns$.

We can think of $(SA)_2 S_2 \in \mathbb{R}^{n\times s_2}$ as follows,

$$\begin{aligned}
(SA)_2 S_2 &= (A(S,I,I))_2 S_2 \\
&= \underbrace{A_2}_{n\times n^2} \cdot \underbrace{D_2}_{n^2\times n^2} \cdot \underbrace{S_2}_{ns\times s_2} \text{ by } D_2 \text{ can be thought of as having size } n^2\times ns \\
&= A_2 \cdot \begin{bmatrix} c_{2,1} I_n & & & \\ & c_{2,2} I_n & & \\ & & \ddots & \\ & & & c_{2,n} I_n \end{bmatrix} \cdot S_2
\end{aligned}$$

where I_n is an $n\times n$ identity matrix, $c_{2,i} \geq 0$ for each $i \in [n]$, and the number of nonzero $c_{2,i}$ is s.

For the last step, we fix SU^* and $\widehat{V}$. We use $(SA)_3 \in \mathbb{R}^{n\times ns}$ to denote the matrix obtained by flattening the tensor $SA \in \mathbb{R}^{s\times n\times n}$ along the third direction. We use $Z_3 = (SU^*)^\top \odot \widehat{V}^\top \in \mathbb{R}^{k\times ns}$ to denote the matrix where the i-th row is the vectorization of

$(SU^*)_i \otimes \widehat{V}_i$. We can consider the following objective function,

$$\min_{W \in \mathbb{R}^{n \times k}} \|WZ_3 - (SA)_3\|_F.$$

Choosing a sketching matrix $S_3 \in \mathbb{R}^{ns \times s_3}$ with $s_3 = O(k/\epsilon)$ gives a smaller problem,

$$\min_{W \in \mathbb{R}^{n \times k}} \|WZ_3S_3 - (SA)_3S_3\|_F.$$

Let $\widehat{W} = (SA)_3S_3(Z_3S_3)^\dagger \in \mathbb{R}^{n \times k}$. Then

$$\begin{aligned}
\|\widehat{W}Z_3 - (SA)_3\|_F &\leq (1+\epsilon) \min_{W \in \mathbb{R}^{n \times k}} \|WZ_3 - (SA)_3\|_F && \text{by property of } S_3 \\
&= (1+\epsilon) \min_{W \in \mathbb{R}^{n \times k}} \|W((SU^*)^\top \odot \widehat{V}^\top) - (SA)_3\|_F && \text{by definition } Z_3 \\
&= (1+\epsilon) \min_{W \in \mathbb{R}^{n \times k}} \|(SU^*) \otimes \widehat{V} \otimes W - SA\|_F && \text{by unflattening} \\
&\leq (1+\epsilon)^2 \|(SU^*) \otimes V \otimes W - SA\|_F . && \text{by Equation (21.62)}
\end{aligned}$$

We define $D_3 \in \mathbb{R}^{n^2 \times n^2}$ to be a diagonal matrix formed by copying the $n \times n$ identity matrix s times on n diagonal blocks of D_3. Then it has ns nonzero entries. Thus, D_3 also can be thought of as a matrix that has size $n^2 \times ns$ and D_3 is uniquely determined by S.

Similarly as to the 2nd dimension, for the 3rd dimension, we can think of $(SA)_3S_3$ as follows,

$$\begin{aligned}
(SA)_3S_3 &= (A(S, I, I))_3S_3 \\
&= \underbrace{A_3}_{n \times n^2} \cdot \underbrace{D_3}_{n^2 \times n^2} \cdot \underbrace{S_3}_{ns \times s_3} && \text{by } D_3 \text{ can be thought of as having size } n^2 \times ns \\
&= A_3 \cdot \begin{bmatrix} c_{3,1}I_n & & & \\ & c_{3,2}I_n & & \\ & & \ddots & \\ & & & c_{3,n}I_n \end{bmatrix} \cdot S_3
\end{aligned}$$

where I_n is an $n \times n$ identity matrix, $c_{3,i} \geq 0$ for each $i \in [n]$ and the number of nonzero $c_{3,i}$ is s.

Overall, we have proved that,

$$\min_{X_2, X_3} \|(SU^*) \otimes (A_2 D_2 S_2 X_2) \otimes (A_3 D_3 S_3 X_3) - SA\|_F \leq (1+\epsilon)^2 \|(SU^*) \otimes V \otimes W - SA\|_F , \tag{21.63}$$

where diagonal matrix $D_2 \in \mathbb{R}^{n^2 \times n^2}$ (with ns nonzero entries) and $D_3 \in \mathbb{R}^{n^2 \times n^2}$ (with ns nonzero entries) are uniquely determined by diagonal matrix $S \in \mathbb{R}^{n \times n}$ (s nonzero entries). Let X_2' and X_3' denote the optimal solution to the above problem (Equation (21.63)). Let $V'' = (A_2 D_2 S_2 X_2') \in \mathbb{R}^{n \times k}$ and $W'' = (A_3 D_3 S_3 X_3') \in \mathbb{R}^{n \times k}$. Then we have

$$\|U^* \otimes V'' \otimes W'' - A\|_v \leq \sqrt{s}\alpha\beta \, \mathrm{OPT} .$$

We construct matrix $\widetilde{V} \in \mathbb{R}^{n \times s_2 s_3}$ by copying matrix $(SA)_2 S_2 \in \mathbb{R}^{n \times s_2}$ s_3 times,

$$\widetilde{V} = \begin{bmatrix} (SA)_2 S_2 & (SA)_2 S_2 & \cdots & (SA)_2 S_2 . \end{bmatrix} \tag{21.64}$$

We construct matrix $\widetilde{W} \in \mathbb{R}^{n \times s_2 s_3}$ by copying the i-th column of matrix $(SA)_3 S_3 \in \mathbb{R}^{n \times s_3}$ into $(i-1)s_2 + 1, \cdots, is_2$ columns of $\widetilde{W}$,

$$\widetilde{W} = \begin{bmatrix} ((SA)_3 S_3)_1 \cdots ((SA)_3 S_3)_1 & ((SA)_3 S_3)_2 \cdots ((SA)_3 S_3)_2 & \cdots & ((SA)_3 S_3)_{s_3} \cdots ((SA)_3 S_3)_{s_3} . \end{bmatrix} \tag{21.65}$$

Although we don't know S, we can guess all of the possibilities. For each possibility,

we can find a solution $\widetilde{U} \in \mathbb{R}^{n \times s_2 s_3}$ to the following problem,

$$\begin{aligned}
& \min_{U \in \mathbb{R}^{n \times s_2 s_3}} \left\| \sum_{i=1}^{s_2} \sum_{j=1}^{s_3} U_{(i-1)s_3+j} \otimes ((SA)_2 S_2)_i \otimes ((SA)_3 S_3)_j - A \right\|_v \\
= & \min_{U \in \mathbb{R}^{n \times s_2 s_3}} \left\| \sum_{i=1}^{s_2} \sum_{j=1}^{s_3} U_{(i-1)s_3+j} \cdot \mathrm{vec}(((SA)_2 S_2)_i \otimes ((SA)_3 S_3)_j) - A_1 \right\|_v \\
= & \min_{U \in \mathbb{R}^{n \times s_2 s_3}} \left\| \sum_{i=1}^{s_2} \sum_{j=1}^{s_3} U_{(i-1)s_3+j} \cdot (\widetilde{V}^\top \odot \widetilde{W}^\top)^{(i-1)s_3+j} - A_1 \right\|_v \\
= & \min_{U \in \mathbb{R}^{n \times s_2 s_3}} \left\| U \cdot (\widetilde{V}^\top \odot \widetilde{W}^\top) - A_1 \right\|_v \\
= & \min_{U \in \mathbb{R}^{n \times s_2 s_3}} \| UZ - A_1 \|_v \\
= & \min_{U \in \mathbb{R}^{n \times s_2 s_3}} \sum_{i=1}^{s_2 s_3} \| U^i Z - A_1^i \|_2,
\end{aligned}$$

where the first step follows by flattening the tensor along the 1st dimension, $U_{(i-1)s_3+j}$ denotes the $(i-1)s_3 + j$-th column of $U \in \mathbb{R}^{n \times s_2 s_3}$, $A_1 \in \mathbb{R}^{n \times n^2}$ denotes the matrix obtained by flattening tensor A along the 1st dimension, the second step follows since $\widetilde{V}^\top \odot \widetilde{W}^\top \in \mathbb{R}^{s_2 s_3 \in n^2}$ is defined to be the matrix where the $(i-1)s_3 + j$-th row is vectorization of $((SA)_2 S_2)_i \otimes ((SA)_3 S_3)_j$, the fourth step follows by defining Z to be $\widetilde{V}^\top \odot \widetilde{W}^\top$, and the last step follows by definition of $\| \cdot \|_v$ norm. Thus, we obtain a multiple regression problem and it can be solved directly by using [CW13, NN13a].

Finally, we take the best $\widetilde{U}, \widetilde{V}, \widetilde{W}$ over all the guesses. The entire running time is dominated by the number of guesses, which is $n^{\mathrm{poly}(k)}$. This completes the proof. □

Theorem 21.12.17. *Given a 3rd order tensor $A \in \mathbb{R}^{n \times n \times n}$, for any $k \geq 1$, let $r = O(k^2)$. There exists an algorithm which takes $O(\mathrm{nnz}(A)) + n\,\mathrm{poly}(k, \log n)$ time and outputs three*

Algorithm 21.37 ℓ_1-Frobenius(ℓ_1-ℓ_2-ℓ_2) Low-rank Approximation Algorithm, poly$(k, \log n)$ approximation

procedure L122TensorLowRankApprox(A, n, k) ▷ Theorem 21.12.17
  $\epsilon \leftarrow 0.1$.
  $s \leftarrow \text{poly}(k, \log n)$.
  Choose $S \in \mathbb{R}^{s \times n}$ to be an MSketch. ▷ Definition 21.12.5
  $s_2 \leftarrow s_3 \leftarrow O(k/\epsilon)$.
  $t_2 \leftarrow t_3 \leftarrow \text{poly}(k/\epsilon)$.
  $r \leftarrow s_2 s_3$.
  Choose sketching matrices $S_2 \in \mathbb{R}^{ns \times s_2}$, $S_3 \in \mathbb{R}^{ns \times s_3}$.
  Choose sketching matrices $T_2 \in \mathbb{R}^{t_2 \times n}$, $T_3 \in \mathbb{R}^{t_3 \times n}$.
  Compute $(SA)_2 S_2, (SA)_3 S_3$.
  Compute $T_2(SA)_2 S_2, T_3(SA)_3 S_3$.
  Form $\widetilde{V} \in \mathbb{R}^{n \times r}$ by repeating $(SA)_2 S_2$ s_3 times according to Equation (21.74).
  Form $\widetilde{W} \in \mathbb{R}^{n \times r}$ by repeating $(SA)_3 S_3$ s_2 times according to Equation (21.75).
  Form $\overline{V} \in \mathbb{R}^{t_2 \times r}$ by repeating $T_2(SA)_2 S_2$ s_3 times according to Equation (21.72).
  Form $\overline{W} \in \mathbb{R}^{t_3 \times r}$ by repeating $T_3(SA)_3 S_3$ s_2 times according to Equation (21.73).
  $C \leftarrow A(I, T_2, T_3)$.
  Form objective function $\min_{U \in \mathbb{R}^{n \times r}} \|U \cdot (\overline{V}^\top \odot \overline{W}^\top) - C_1\|_F$.
  Use linear regression solver to find a solution $\widetilde{U}$.
  return $\widetilde{U}$, $\widetilde{V}$, $\widetilde{W}$.
end procedure

matrices $U, V, W \in \mathbb{R}^{n \times r}$ *such that*

$$\|U \otimes V \otimes W - A\|_v \leq \text{poly}(k, \log n) \min_{\text{rank}-k\ A'} \|A' - A\|_v$$

holds with probability at least $9/10$.

Proof. We define OPT as follows,

$$\text{OPT} = \min_{U,V,W \in \mathbb{R}^{n \times k}} \|U \otimes V \otimes W - A\|_v = \min_{U,V,W \in \mathbb{R}^{n \times k}} \left\| \sum_{i=1}^{k} U_i \otimes V_i \otimes W_i - A \right\|_v .$$

Let $A_1 \in \mathbb{R}^{n \times n^2}$ denote the matrix obtained by flattening tensor A along the 1st dimension. Let $U^* \in \mathbb{R}^{n \times k}$ denote the optimal solution. We fix $U^* \in \mathbb{R}^{n \times k}$, and consider the objective function,

$$\min_{V,W \in \mathbb{R}^{n \times k}} \|U^* \otimes V \otimes W - A\|_v \equiv \min_{V,W \in \mathbb{R}^{n \times k}} \left\|U^* \cdot (V^\top \odot W^\top) - A_1\right\|_v , \tag{21.66}$$

which has cost at most OPT, and where $V^\top \odot W^\top \in \mathbb{R}^{k \times n^2}$ denotes the matrix for which the i-th row is a vectorization of $V_i \otimes W_i, \forall i \in [k]$. (Note that $V_i \in \mathbb{R}^n$ is the i-th column of matrix $V \in \mathbb{R}^{n \times k}$). Choose an (oblivious) MSketch $S \in \mathbb{R}^{s \times n}$ with $s = \mathrm{poly}(k, \log n)$ according to Definition 21.12.5. Using MSketch S, w to sketch on the left of the objective function when U^* is fixed (Equation (21.66)), we obtain a smaller problem,

$$\min_{V,W \in \mathbb{R}^{n \times k}} \|(SU^*) \otimes V \otimes W - SA\|_v \equiv \min_{V,W \in \mathbb{R}^{n \times k}} \left\|SU^* \cdot (V^\top \odot W^\top) - SA_1\right\|_v . \tag{21.67}$$

Let V', W' denote the optimal solution to the above problem, i.e.,

$$V', W' = \arg\min_{V,W \in \mathbb{R}^{n \times k}} \|(SU^*) \otimes V \otimes W - SA\|_v .$$

Then using properties (no dilation Lemma 21.12.13 and no contraction Lemma 21.12.14) of S, we have

$$\|U^* \otimes V' \otimes W' - A\|_v \leq \alpha \, \mathrm{OPT} .$$

where α is an approximation ratio determined by S.

By definition of $\|\cdot\|_v$ and $\|\cdot\|_2 \leq \|\cdot\|_1 \leq \sqrt{\dim}\|\cdot\|_2$, we can rewrite Equation (21.67)

in the following way,

$$\begin{aligned}
&\|(SU^*) \otimes V \otimes W - SA\|_v \\
&= \sum_{i=1}^{s} \left(\sum_{j=1}^{n} \sum_{l=1}^{n} \left(((SU^*) \otimes V \otimes W)_{i,j,l} - (SA)_{i,j,l} \right)^2 \right)^{\frac{1}{2}} \\
&\leq \sqrt{s} \left(\sum_{i=1}^{s} \sum_{j=1}^{n} \sum_{l=1}^{n} \left(((SU^*) \otimes V \otimes W)_{i,j,l} - (SA)_{i,j,l} \right)^2 \right)^{\frac{1}{2}} \\
&= \sqrt{s} \, \|(SU^*) \otimes V \otimes W - SA\|_F
\end{aligned} \tag{21.68}$$

Using the properties of S and Equation (21.68), for any $\beta \geq 1$, let V'', W'' denote a β-approximation solution of $\min_{V,W \in \mathbb{R}^{n \times k}} \|(SU^*) \otimes V \otimes W - SA\|_F$, i.e.,

$$\|(SU^*) \otimes V'' \otimes W'' - SA\|_F \leq \beta \cdot \min_{V,W \in \mathbb{R}^{n \times k}} \|(SU^*) \otimes V \otimes W - SA\|_F . \tag{21.69}$$

Then,

$$\|U^* \otimes V'' \otimes W'' - A\|_v \leq \sqrt{s}\alpha\beta \cdot \mathrm{OPT} . \tag{21.70}$$

Let $\widehat{A}$ denote SA. Choose $S_i \in \mathbb{R}^{ns \times s_i}$ to be Gaussian matrix with $s_i = O(k/\epsilon)$, $\forall i \{2, 3\}$. By a similar proof as in Theorem 21.12.16, we have if X_2', X_3' is a β-approximate solution to

$$\min_{X_2, X_3} \|(SU^*) \otimes (\widehat{A}_2 S_2 X_2) \otimes (\widehat{A}_3 S_3 X_3) - SA\|_F,$$

then,

$$\|U^* \otimes (\widehat{A}_2 S_2 X_2) \otimes (\widehat{A}_3 S_3 X_3) - A\|_v \leq \sqrt{s}\alpha\beta.$$

To reduce the size of the objective function from poly(n) to poly(k/ϵ), we use perform an "input sparsity reduction" (in Lemma 21.4.3). Note that, we do not need to use this idea to optimize the running time in Theorem 21.12.16. The running time of Theorem 21.12.16 is dominated by guessing sampling and rescaling matrices. (That running time is $\gg \text{nnz}(A)$.) Choose $T_i \in \mathbb{R}^{t_i \times n}$ to be a sparse subspace embedding matrix (CountSketch transform) with $t_i = \text{poly}(k, 1/\epsilon)$, $\forall i \in \{2, 3\}$. Applying the proof of Lemma 21.4.3 here, we obtain, if X_2', X_3' is a β-approximate solution to

$$\min_{X_2, X_3} \|(SU^*) \otimes (T_2(SA)_2 S_2 X_2) \otimes (T_3(SA)_3 S_3 X_3) - SA\|_F,$$

then,

$$\|U^* \otimes ((SA)_2 S_2 X_2) \otimes ((SA)_3 S_3 X_3) - A\|_v \leq \sqrt{s}\alpha\beta. \tag{21.71}$$

Similar to the bicriteria results in Section 21.4.4, Equation (21.71) indicates that we can construct a bicriteria solution by using two matrices $(SA)_2 S_2$ and $(SA)_3 S_3$. The next question is how to obtain the final results $\widehat{U}, \widehat{V}, \widehat{W}$. We first show how to obtain $\widehat{U}$. Then we show to construct $\widehat{V}$ and $\widehat{W}$.

To obtain $\widehat{U}$, we need to solve a regression problem related to two matrices $\overline{V}, \widehat{W}$ and a tensor $A(I, T_2, T_3)$. We construct matrix $\overline{V} \in \mathbb{R}^{t_2 \times s_2 s_3}$ by copying matrix $T_2(SA)_2 S_2 \in \mathbb{R}^{t_2 \times s_2}$ s_3 times,

$$\overline{V} = \begin{bmatrix} T_2(SA)_2 S_2 & T_2(SA)_2 S_2 & \cdots & T_2(SA)_2 S_2 \end{bmatrix}. \tag{21.72}$$

We construct matrix $\overline{W} \in \mathbb{R}^{t_3 \times s_2 s_3}$ by copying the i-th column of matrix $T_3(SA)_3 S_3 \in \mathbb{R}^{t_3 \times s_3}$ into $(i-1)s_2 + 1, \cdots, is_2$ columns of $\overline{W}$,

$$\overline{W} = \begin{bmatrix} F_1 \cdots F_1 & F_2 \cdots F_2 & \cdots & F_{s_3} \cdots F_{s_3} \end{bmatrix}, \tag{21.73}$$

where $F = T_3(SA)_3 S_3$.

Thus, to obtain $\widetilde{U} \in \mathbb{R}^{s_2 s_3}$, we just need to use a linear regression solver to solve a smaller problem,

$$\min_{U \in \mathbb{R}^{s_2 s_3}} \|U \cdot (\overline{V}^\top \odot \overline{W}^\top) - A(I, T_2, T_3)\|_F,$$

which can be solved in $O(\mathrm{nnz}(A)) + n\,\mathrm{poly}(k, \log n)$ time. We will show how to obtain $\widetilde{V}$ and $\widetilde{W}$.

We construct matrix $\widetilde{V} \in \mathbb{R}^{n \times s_2 s_3}$ by copying matrix $(SA)_2 S_2 \in \mathbb{R}^{n \times s_2}$ s_3 times,

$$\widetilde{V} = \begin{bmatrix} (SA)_2 S_2 & (SA)_2 S_2 & \cdots & (SA)_2 S_2 . \end{bmatrix} \tag{21.74}$$

We construct matrix $\widetilde{W} \in \mathbb{R}^{n \times s_2 s_3}$ by copying the i-th column of matrix $(SA)_3 S_3 \in \mathbb{R}^{n \times s_3}$ into $(i-1)s_2 + 1, \cdots, i s_2$ columns of $\widetilde{W}$,

$$\widetilde{W} = \begin{bmatrix} F_1 \cdots F_1 & F_2 \cdots F_2 & \cdots & F_{s_3} \cdots F_{s_3} \end{bmatrix}, \tag{21.75}$$

where $F = (SA)_3 S_3$. □

21.12.3 ℓ_1-ℓ_1-ℓ_2 norm

Section 21.12.3.1 presents some definitions and useful facts for the tensor ℓ_1-ℓ_1-ℓ_2 norm. We provide some tools in Section 21.12.3.2. Section 21.12.3.3 presents a key idea which shows we are able to reduce the original problem to a new problem under entry-wise ℓ_1 norm. Section 21.12.3.4 presents several existence results. Finally, Section 21.12.3.6 introduces several algorithms with different tradeoffs.

21.12.3.1 Definitions

Definition 21.12.7. (Tensor u-norm) For an $n \times n \times n$ tensor A, we define the u-norm of A, denoted $\|A\|_u$, to be

$$\left(\sum_{i=1}^{n}\sum_{j=1}^{n} M(\|A_{i,j,*}\|_2)\right)^{1/p},$$

where $A_{i,j,*}$ is the (i,j)-th tube of A, and p is a parameter associated with the function $M()$, which defines a nice M-Estimator.

Definition 21.12.8. (Matrix u-norm) For an $n \times n$ matrix A, we define u-norm of A, denoted $\|A\|_u$, to be

$$\left(\sum_{i=1}^{n} M(\|A_{i,*}\|_2)\right)^{1/p},$$

where $A_{i,*}$ is the i-th row of A, and p is a parameter associated with the function $M()$, which defines a nice M-Estimator.

Fact 21.12.18. *For $p = 1$, for any two matrices A and B, we have $\|A+B\|_u \leq \|A\|_u + \|B\|_u$. For any two tensors A and B, we have $\|A + B\|_u \leq \|A\|_u + \|B\|_u$.*

21.12.3.2 Projection via Gaussians

Definition 21.12.9. Let $p \geq 1$. Let $\ell_p^{\mathcal{S}^{n-1}}$ be an infinite dimensional ℓ_p metric which consists of a coordinate for each vector r in the unit sphere $\mathcal{S}^{n-1}$. Define function $f : \mathcal{S}^{n-1} \to \mathbb{R}$. The ℓ_1-norm of any such f is defined as follows:

$$\|f\|_1 = \left(\int_{r \in \mathcal{S}^{n-1}} |f(r)|^p \mathrm{d}r \right)^{1/p}.$$

Claim 21.12.19. *Let $f_v(r) = \langle v, r \rangle$. There exists a universal constant α_p such that*

$$\|f_v\|_p = \alpha_p \|v\|_2.$$

Proof. We have,

$$\begin{aligned} \|f_v\|_p = & \left(\int_{r \in \mathcal{S}^{n-1}} |\langle v, r \rangle|^p \mathrm{d}r \right)^{1/p} \\ = & \left(\int_{\theta \in \mathcal{S}^{n-1}} \|v\|_2^p \cdot |\cos\theta|^p \mathrm{d}\theta \right)^{1/p} \\ = & \|v\|_2 \left(\int_{\theta \in \mathcal{S}^{n-1}} |\cos\theta|^p \mathrm{d}\theta \right)^{1/p} \\ = & \alpha_p \|v\|_2. \end{aligned}$$

This completes the proof. □

Lemma 21.12.20. *Let $G \in \mathbb{R}^{k \times n}$ denote i.i.d. random Gaussian matrices with rescaling. Then for any $v \in \mathbb{R}^n$, we have*

$$\Pr[(1-\epsilon)\|v\|_2 \leq \|Gv\|_1 \leq (1+\epsilon)\|v\|_2] \geq 1 - 2^{-\Omega(k\epsilon^2)}.$$

Proof. For each $i \in [k]$, we define $X_i = \langle v, g_i \rangle$, where $g_i \in \mathbb{R}^n$ is the i-th row of G. Then $X_i = \sum_{j=1}^n v_j g_{i,j}$ and $\mathbb{E}[|X_i|] = \alpha_p \|v\|_2$. Define $Y = \sum_{i=1}^k |X_i|$. We have $\mathbb{E}[Y] = k\alpha_1 \|v\|_2 = k\alpha_1$.

We can show

$$\begin{aligned}
\Pr[Y \geq (1+\epsilon)\alpha_1 k] =& \Pr[e^{sY} \geq e^{s(1+\epsilon)\alpha_1 k}] && \text{for all } s > 0 \\
\leq& \mathbb{E}[e^{sY}]/e^{s(1+\epsilon)\alpha_1 k} && \text{by Markov's inequality} \\
=& e^{-s(1+\epsilon)\alpha_1 k} \cdot \mathbb{E}[\prod_{i=1}^{k} e^{s|X_i|}] && \text{by } Y = \sum_{i=1}^{k} |X_i| \\
=& e^{-s(1+\epsilon)\alpha_1 k} \cdot (\mathbb{E}[e^{s|X_1|}])^k
\end{aligned}$$

It remains to bound $\mathbb{E}[e^{s|X_1|}]$. Since $X_1 \sim \mathcal{N}(0,1)$, we have that X_1 has density function $e^{-t^2/2}$. Thus, we have,

$$\begin{aligned}
\mathbb{E}[e^{s|X_1|}] &= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} e^{s|t|} \cdot e^{-t^2/2} \mathrm{d}t \\
&= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} e^{s^2/2} \cdot e^{-(|t|-s)^2/2} \mathrm{d}t \\
&= e^{s^2/2}(\mathrm{erf}(s/\sqrt{2}) + 1) \\
&\leq e^{s^2/2}((1-\exp(-2s^2/\pi))^{1/2} + 1) && \text{by } 1-\exp(-4x^2/\pi) \geq \mathrm{erf}(x)^2 \\
&\leq e^{s^2/2}(\sqrt{2/\pi}s + 1). && \text{by } 1 - e^{-x} \leq x
\end{aligned}$$

Thus, we have

$$\begin{aligned}
\Pr[Y \geq (1+\epsilon)\alpha_1 k] &\leq e^{-s(1+\epsilon)k} e^{ks^2/2} (1 + s\sqrt{2/\pi})^k \\
&= e^{-s(1+\epsilon)\alpha_1 k} e^{ks^2/2} e^{k \cdot \log(1+s\sqrt{2/\pi})} \\
&\leq e^{-s(1+\epsilon)\alpha_1 k + ks^2/2 + k \cdot s\sqrt{2/\pi}} \\
&\leq e^{-\Omega(k\epsilon^2)}. && \text{by } \alpha_1 \geq \sqrt{2/\pi} \text{ and setting } s = \epsilon
\end{aligned}$$

□

Lemma 21.12.21. *For any $\epsilon \in (0,1)$, let $k = O(n/\epsilon^2)$. Let $G \in \mathbb{R}^{k \times n}$ denote i.i.d. random Gaussian matrices with rescaling. Then for any $v \in \mathbb{R}^n$, with probability at least $1 - 2^{-\Omega(n/\epsilon^2)}$, we have : for all $v \in \mathbb{R}^n$,*

$$(1-\epsilon)\|v\|_2 \leq \|Gv\|_1 \leq (1+\epsilon)\|v\|_2.$$

Proof. Let $\mathcal{S}$ denote $\{y \in \mathbb{R}^n \mid \|y\|_2 = 1\}$. We construct a γ-net so that for all $y \in \mathcal{S}$, there exists a vector $w \in \mathcal{N}$ for which $\|y - w\|_2 \leq \gamma$. We set $\gamma = 1/2$.

For any unit vector y, we can write

$$y = y^0 + y^1 + y^2 + \cdots,$$

where $\|y^i\|_2 \leq 1/2^i$ and y^i is a scalar multiple of a vector in $\mathcal{N}$. Thus, we have

$$\begin{aligned}
\|Gy\|_1 &= \|G(y^0 + y^1 + y^2 + \cdots)\|_1 \\
&\leq \sum_{i=0}^{\infty} \|Gy^i\|_1 && \text{by triangle inequality} \\
&\leq \sum_{i=0}^{\infty} (1+\epsilon)\|y^i\|_2 \\
&\leq \sum_{i=0}^{\infty} (1+\epsilon)\frac{1}{2^i} \\
&\leq 1 + \Theta(\epsilon).
\end{aligned}$$

Similarly, we can lower bound $\|Gy\|_1$ by $1 - \Theta(\epsilon)$. By Lemma 2.2 in [Woo14b], we know that for any $\gamma \in (0,1)$, there exists a γ-net $\mathcal{N}$ of $\mathcal{S}$ for which $|\mathcal{N}| \leq (1 + 4/\gamma)^n$. □

21.12.3.3 Reduction, projection to high dimension

Lemma 21.12.22. *Given a 3rd order tensor $A \in \mathbb{R}^{n \times n \times n}$, let $S \in \mathbb{R}^{n \times s}$ denote a Gaussian matrix with $s = O(n/\epsilon^2)$ columns. With probability at least $1 - 2^{-\Omega(n/\epsilon^2)}$, for any $U, V, W \in$*

$\mathbb{R}^{n\times k}$, *we have*

$$(1-\epsilon)\left\|U\otimes V\otimes W-A\right\|_u \le \left\|(U\otimes V\otimes W)S-AS\right\|_1 \le (1+\epsilon)\left\|U\otimes V\otimes W-A\right\|_u.$$

Proof. By definition of the $\otimes$ product between matrices and $\cdot$ product between a tensor and a matrix, we have $(U\otimes V\otimes W)S = U\otimes V\otimes(SW)\in\mathbb{R}^{n\times n\times s}$. We use $A_{i,j,*}\in\mathbb{R}^n$ to denote the (i,j)-th tube (the column in the 3rd dimension) of tensor A. We first prove the upper bound,

$$\begin{aligned}\left\|(U\otimes V\otimes W)S-AS\right\|_1 &= \sum_{i=1}^n\sum_{j=1}^n \left\|\left((U\otimes V\otimes W)_{i,j,*}-A_{i,j,*}\right)S\right\|_1\\ &\le \sum_{i=1}^n\sum_{j=1}^n (1+\epsilon)\left\|(U\otimes V\otimes W)_{i,j,*}-A_{i,j,*}\right\|_2\\ &= (1+\epsilon)\left\|U\otimes V\otimes W-A\right\|_u,\end{aligned}$$

where the first step follows by definition of tensor $\|\cdot\|_u$ norm, the second step follows by Lemma 21.12.21, and the last step follows by tensor entry-wise ℓ_1 norm. Similarly, we can prove the lower bound,

$$\begin{aligned}\left\|(U\otimes V\otimes W)S-AS\right\|_1 &\ge \sum_{i=1}^n\sum_{j=1}^n (1-\epsilon)\left\|(U\otimes V\otimes W)_{i,j,*}-A_{i,j,*}\right\|_2\\ &= (1-\epsilon)\left\|U\otimes V\otimes W-A\right\|_u.\end{aligned}$$

This completes the proof. □

Corollary 21.12.23. *For any $\alpha\ge 1$, if U',V',W' satisfy*

$$\left\|(U'\otimes V'\otimes W'-A)S\right\|_1 \le \gamma \min_{\mathrm{rank}-k\ A_k}\left\|(A_k-A)S\right\|_1,$$

then

$$\left\|U'\otimes V'\otimes W'-A\right\|_u \le \gamma\frac{1+\epsilon}{1-\epsilon}\min_{\mathrm{rank}-k\ A_k}\left\|A_k-A\right\|_u.$$

Proof. Let $\widehat{U}, \widehat{V}, \widehat{W}$ denote the optimal solution to $\min_{\text{rank}-k\ A_k} \|(A_k - A)S\|_1$. Let U^*, V^*, W^* denote the optimal solution to $\min_{\text{rank}-k\ A_k} \|A_k - A\|_u$. Then,

$$\begin{aligned}
\|U' \otimes V' \otimes W' - A\|_u &\leq \frac{1}{1-\epsilon} \|(U' \otimes V' \otimes W' - A)S\|_1 \\
&\leq \gamma \frac{1}{1-\epsilon} \|(\widehat{U} \otimes \widehat{V} \otimes \widehat{W} - A)S\|_1 \\
&\leq \gamma \frac{1}{1-\epsilon} \|(U^* \otimes V^* \otimes W^* - A)S\|_1 \\
&\leq \gamma \frac{1+\epsilon}{1-\epsilon} \|U^* \otimes V^* \otimes W^* - A\|_u,
\end{aligned}$$

which completes the proof. □

21.12.3.4 Existence results

Theorem 21.12.24 (Existence results)**.** *Given a 3rd order tensor $A \in \mathbb{R}^{n \times n \times n}$ and a matrix $S \in \mathbb{R}^{n \times \overline{n}}$, let* OPT *denote* $\min_{\text{rank}-k\ A_k \in \mathbb{R}^{n \times n \times n}} \|(A_k - A)S\|_1$*, let $\widehat{A} = AS \in \mathbb{R}^{n \times n \times \overline{n}}$. For any $k \geq 1$, there exist three matrices $S_1 \in \mathbb{R}^{n\overline{n} \times s_1}$, $S_2 \in \mathbb{R}^{n\overline{n} \times s_2}$, $S_3 \in \mathbb{R}^{n^2 \times s_3}$ such that*

$$\min_{X_1 \in \mathbb{R}^{s_1 \times k}, X_2 \in \mathbb{R}^{s_2 \times k}, X_3 \in \mathbb{R}^{s_3 \times k}} \left\| (\widehat{A}_1 S_1 X_1) \otimes (\widehat{A}_2 S_2 X_2) \otimes (\widehat{A}_3 S_3 X_3) - \widehat{A} \right\|_1 \leq \alpha\, \text{OPT},$$

or equivalently,

$$\min_{X_1 \in \mathbb{R}^{s_1 \times k}, X_2 \in \mathbb{R}^{s_2 \times k}, X_3 \in \mathbb{R}^{s_3 \times k}} \left\| \left((\widehat{A}_1 S_1 X_1) \otimes (\widehat{A}_2 S_2 X_2) \otimes (A_3 S_3 X_3) - A \right) S \right\|_1 \leq \alpha\, \text{OPT},$$

holds with probability 99/100*.*

(I). *Using a dense Cauchy transform,*
$s_1 = s_2 = s_3 = \widetilde{O}(k)$, $\alpha = \widetilde{O}(k^{1.5}) \log^3 n$.

(II). *Using a sparse Cauchy transform,*
$s_1 = s_2 = s_3 = \widetilde{O}(k^5)$, $\alpha = \widetilde{O}(k^{13.5}) \log^3 n$.

(III). *Guessing Lewis weights,*

$s_1 = s_2 = s_3 = \widetilde{O}(k)$, $\alpha = \widetilde{O}(k^{1.5})$.

Proof. We use OPT to denote the optimal cost,

$$\mathrm{OPT} := \min_{\mathrm{rank}-k\ A_k \in \mathbb{R}^{n\times n\times n}} \|(A_k - A)S\|_1.$$

We fix $V^* \in \mathbb{R}^{n\times k}$ and $W^* \in \mathbb{R}^{n\times k}$ to be the optimal solution to

$$\min_{U,V,W} \|(U \otimes V \otimes W - A)S\|_1.$$

We define $Z_1 \in \mathbb{R}^{k\times n\overline{n}}$ to be the matrix where the i-th row is the vectorization of $V_i^* \otimes (SW_i^*)$. We define tensor

$$\widehat{A} = AS \in \mathbb{R}^{n\times n\times \overline{n}}.$$

Then we also have $\widehat{A} = A(I, I, S)$ according to the definition of the $\cdot$ product between a tensor and a matrix.

Let $\widehat{A}_1 \in \mathbb{R}^{n\times n\overline{n}}$ denote the matrix obtained by flattening tensor $\widehat{A}$ along the first direction. We can consider the following optimization problem,

$$\min_{U\in\mathbb{R}^{n\times k}} \left\| UZ_1 - \widehat{A}_1 \right\|_1.$$

Choosing S_1 to be one of the following sketching matrices:

(I) a dense Cauchy transform,

(II) a sparse Cauchy transform,

(III) a sampling and rescaling diagonal matrix according to Lewis weights.

Let α_{S_1} denote the approximation ratio produced by the sketching matrix S_1. We use $S_1 \in \mathbb{R}^{n\overline{n}\times s_1}$ to sketch on right of the above problem, and obtain the problem:

$$\min_{U\in\mathbb{R}^{n\times k}} \|UZ_1S_1 - \widehat{A}_1S_1\|_1 = \min_{U\in\mathbb{R}^{n\times k}} \sum_{i=1}^{n} \|U^iZ_1S_1 - (\widehat{A}_1S_1)^i\|_1,$$

where U^i denotes the i-th row of matrix $U \in \mathbb{R}^{n\times k}$ and $(\widehat{A}_1S_1)^i$ denotes the i-th row of matrix $\widehat{A}_1S_1$. Instead of solving it under ℓ_1-norm, we consider the ℓ_2-norm relaxation,

$$\min_{U\in\mathbb{R}^{n\times k}} \|UZ_1S_1 - \widehat{A}_1S_1\|_F^2 = \min_{U\in\mathbb{R}^{n\times k}} \sum_{i=1}^{n} \|U^iZ_1S_1 - (\widehat{A}_1S_1)^i\|_2^2.$$

Let $\widehat{U} \in \mathbb{R}^{n\times k}$ denote the optimal solution of the above optimization problem, so that $\widehat{U} = \widehat{A}_1S_1(Z_1S_1)^\dagger$. We plug $\widehat{U}$ into the objective function under the ℓ_1-norm. By the property of sketching matrix $S_1 \in \mathbb{R}^{n\overline{n}\times s_1}$, we have,

$$\|\widehat{U}Z_1 - \widehat{A}_1\|_1 \leq \alpha_{S_1} \min_{U\in\mathbb{R}^{n\times k}} \|UZ_1 - \widehat{A}_1\|_1 = \alpha_{S_1}\,\mathrm{OPT},$$

which implies that,

$$\|\widehat{U}\otimes V^*\otimes(SW^*) - \widehat{A}\|_1 = \|(\widehat{U}\otimes V^*\otimes W^*)S - \widehat{A}\|_1 \leq \alpha_{S_1}\,\mathrm{OPT}\,.$$

In the second step, we fix $\widehat{U} \in \mathbb{R}^{n\times k}$ and $W^* \in \mathbb{R}^{n\times k}$. Let $\widehat{A}_2 \in \mathbb{R}^{n\times n\overline{n}}$ denote the matrix obtained by flattening tensor $\widehat{A} \in \mathbb{R}^{n\times n\times\overline{n}}$ along the second direction. We choose a sketching matrix $S_2 \in \mathbb{R}^{n\overline{n}\times s_2}$. Let $Z_2 = \widehat{U}^\top \odot (SW^*)^\top \in \mathbb{R}^{k\times n\overline{n}}$ denote the matrix where the i-th row is the vectorization of $\widehat{U}_i \otimes (SW_i^*)$. Define $\widehat{V} = \widehat{A}_2S_2(Z_2S_2)^\dagger$. By the properties of sketching matrix S_2, we have

$$\|\widehat{V}Z_2 - \widehat{A}_2\|_1 \leq \alpha_{S_2}\alpha_{S_1}\,\mathrm{OPT},$$

In the third step, we fix $\widehat{U} \in \mathbb{R}^{n\times k}$ and $\widehat{V} \in \mathbb{R}^{n\times k}$. Let $\widehat{A}_3 \in \mathbb{R}^{\overline{n}\times n^2}$ denote the matrix obtained by flattening tensor $\widehat{A} \in \mathbb{R}^{n\times n\times \overline{n}}$ along the third direction. We choose a sketching matrix $S_3 \in \mathbb{R}^{n^2\times s_3}$. Let $Z_3 \in \mathbb{R}^{k\times n^2}$ denote the matrix where the i-th row is the vectorization of $\widehat{U}_i \otimes \widehat{V}_i$. Define $W' = \widehat{A}_3 S_3 (Z_3 S_3)^\dagger \in \mathbb{R}^{\overline{n}\times k}$ and $\widehat{W} = A_3 S_3 (Z_3 S_3)^\dagger \in \mathbb{R}^{n\times k}$. Then we have,

$$\begin{aligned} W' &= \widehat{A}_3 S_3 (Z_3 S_3)^\dagger \\ &= (A(I,I,S))_3 S_3 (Z_3 S_3)^\dagger \\ &= (S^\top A_3) S_3 (Z_3 S_3)^\dagger \\ &= S^\top \widehat{W} \end{aligned}$$

By properties of sketching matrix S_3, we have

$$\|W' Z_3 - \widehat{A}_3\|_1 \leq \alpha_{S_3}\alpha_{S_2}\alpha_{S_1}\,\mathrm{OPT}\,.$$

Replacing W' by $S^\top \widehat{W}$, we obtain,

$$\|W' Z_3 - \widehat{A}_3\|_1 = \|S^\top \widehat{W} Z_3 - \widehat{A}_3\|_1 = \|S^\top \widehat{W} Z_3 - S^\top A_3\|_1 = \|(\widehat{U}\otimes\widehat{V}\otimes\widehat{W} - A)S\|_1.$$

Thus, we have

$$\min_{X_1\in\mathbb{R}^{s_1\times k}, X_2\in\mathbb{R}^{s_2\times k}, X_3\in\mathbb{R}^{s_3\times k}} \left\|(\widehat{A}_1 S_1 X_1)\otimes(\widehat{A}_2 S_2 X_2)\otimes(\widehat{A}_3 S_3 X_3) - \widehat{A}\right\|_1 \leq \alpha_{S_1}\alpha_{S_2}\alpha_{S_3}\,\mathrm{OPT}\,.$$

□

21.12.3.5 Running time analysis

Fact 21.12.25. *Given tensor $A \in \mathbb{R}^{n\times n\times n}$ and a matrix $B \in \mathbb{R}^{n\times d}$ with $d = O(n)$, let AB denote an $n \times n \times d$ size tensor, For each $i \in [3]$, let $(AB)_i$ denote a matrix obtained by*

flattening tensor AB along the i-th dimension, then

$$(AB)_1 \in \mathbb{R}^{n \times nd}, (AB)_2 \in \mathbb{R}^{n \times nd}, (AB)_3 \in \mathbb{R}^{d \times n^2}.$$

For each $i \in [3]$, let $S_i \in \mathbb{R}^{nd \times s_i}$ denote a sparse Cauchy transform, $T_i \in \mathbb{R}^{t_i \times n}$. Then we have,

(I) *If T_1 denotes a sparse Cauchy transform or a sampling and rescaling matrix according to the Lewis weights, $T_1(AB)_1 S_1$ can be computed in $O(\mathrm{nnz}(A)d)$ time. Otherwise, it can be computed in $O(\mathrm{nnz}(A)d + ns_1t_1)$.*

(II) *If T_2 denotes a sparse Cauchy transform or a sampling and rescaling matrix according to the Lewis weights, $T_2(AB)_2 S_2$ can be computed in $O(\mathrm{nnz}(A)d)$ time. Otherwise, it can be computed in $O(\mathrm{nnz}(A)d + ns_2t_2)$.*

(III) *If T_3 denotes a sparse Cauchy transform or a sampling and rescaling matrix according to the Lewis weights, $T_3(AB)_3 S_3$ can be computed in $O(\mathrm{nnz}(A)d)$ time. Otherwise, it can be computed in $O(\mathrm{nnz}(A)d + ds_3t_3)$.*

Proof. Part (I). Note that $T_1(AB)_1S_1 \in \mathbb{R}^{t_1\times s_1}$ and $(AB)_1 \in \mathbb{R}^{n\times nd}$, for each $i \in [t_1], j \in [s_1]$,

$$\begin{aligned}
(T_1(AB)_1S_1)_{i,j} = & \sum_{x=1}^{n}\sum_{y'=1}^{nd}(T_1)_{i,x}((AB)_1)_{x,y'}(S_1)_{y',j} \\
= & \sum_{x=1}^{n}\sum_{y=1}^{n}\sum_{z=1}^{d}(T_1)_{i,x}((AB)_1)_{x,(y-1)d+z}(S_1)_{(y-1)d+z,j} \\
= & \sum_{x=1}^{n}\sum_{y=1}^{n}\sum_{z=1}^{d}(T_1)_{i,x}(AB)_{x,y,z}(S_1)_{(y-1)d+z,j} \\
= & \sum_{x=1}^{n}\sum_{y=1}^{n}\sum_{z=1}^{d}(T_1)_{i,x}\sum_{w=1}^{n}(A_{x,y,w}B_{w,z})(S_1)_{(y-1)d+z,j} \\
= & \sum_{x=1}^{n}\sum_{y=1}^{n}(T_1)_{i,x}\sum_{w=1}^{n}A_{x,y,w}\sum_{z=1}^{d}B_{w,z}(S_1)_{(y-1)d+z,j}.
\end{aligned}$$

We look at a non-zero entry $A_{x,y,w}$ and the entry $B_{w,z}$. If T_1 denotes a sparse Cauchy transform or a sampling and rescaling matrix according to the Lewis weights, then there is at most one pair (i,j) such that $(T_1)_{i,x}A_{x,y,w}B_{w,z}(S_1)_{(y-1)d+z,j}$ is non-zero. Therefore, computing $T_1(AB)_1S_1$ only needs $\mathrm{nnz}(A)d$ time. If T_1 is not in the above case, since S_1 is sparse, we can compute $(AB)_1S_1$ in $\mathrm{nnz}(A)d$ time by a similar argument. Then, we can compute $T_1(AB)_1S_1$ in nt_1s_1 time.

Part (II). It is as the same as Part (I).

Part (III). Note that $T_3(AB)_3S_3 \in \mathbb{R}^{t_3\times s_3}$ and $(AB)_3 \in \mathbb{R}^{d\times n^2}$. For each $i \in [t_3], j \in$

$[s_3]$,

$$\begin{aligned}
(T_3(AB)_3S_3)_{i,j} = & \sum_{x=1}^{d}\sum_{y'=1}^{n^2}(T_3)_{i,x}((AB)_3)_{x,y'}(S_3)_{y',j} \\
= & \sum_{x=1}^{d}\sum_{y=1}^{n}\sum_{z=1}^{n}(T_3)_{i,x}((AB)_3)_{x,(y-1)n+z}(S_3)_{(y-1)n+z,j} \\
= & \sum_{x=1}^{d}\sum_{y=1}^{n}\sum_{z=1}^{n}(T_3)_{i,x}(AB)_{y,z,x}(S_3)_{(y-1)n+z,j} \\
= & \sum_{x=1}^{d}\sum_{y=1}^{n}\sum_{z=1}^{n}(T_3)_{i,x}\sum_{w=1}^{n}A_{y,z,w}B_{w,x}(S_3)_{(y-1)n+z,j}
\end{aligned}$$

Similar to Part (I), if T_1 denotes a sparse Cauchy transform or a sampling and rescaling matrix according to the Lewis weights, computing $T_3(AB)_3S_3$ only needs $\mathrm{nnz}(A)d$ time. Otherwise, it needs $dt_3s_3 + \mathrm{nnz}(A)d$ running time.

□

21.12.3.6 Algorithms

Theorem 21.12.26. *Given a 3rd order tensor $A \in \mathbb{R}^{n\times n\times n}$, for any $k \geq 1$, there exists an algorithm which takes $O(\mathrm{nnz}(A)n) + \widetilde{O}(n)\,\mathrm{poly}(k) + n2^{\widetilde{O}(k^2)}$ time and outputs three matrices $U, V, W \in \mathbb{R}^{n\times k}$ such that,*

$$\|U \otimes V \otimes W - A\|_u \leq \mathrm{poly}(k, \log n) \min_{\mathrm{rank}-k\ A'} \|A' - A\|_u,$$

holds with probability at least $9/10$.

Proof. We first choose a Gaussian matrix $S \in \mathbb{R}^{n\times\overline{n}}$ with $\overline{n} = O(n)$. By applying Corollary 21.12.23, we can reduce the original problem to a "generalized" ℓ_1 low rank approximation

Algorithm 21.38 ℓ_1-ℓ_1-ℓ_2-Low Rank Approximation algorithm, input sparsity time

procedure L112TensorLowRankApproxInputSparsity(A, n, k) ▷ Theorem 21.12.26
  $\overline{n} \leftarrow O(n)$.
  $s_1 \leftarrow s_2 \leftarrow s_3 \leftarrow \widetilde{O}(k^5)$.
  Choose $S \in \mathbb{R}^{n \times \overline{n}}$ to be a Gaussian matrix.
  Choose $S_1 \in \mathbb{R}^{n\overline{n} \times s_1}$ to be a sparse Cauchy transform. ▷ Part (II) of Theorem 21.12.24
  Choose $S_2 \in \mathbb{R}^{n\overline{n} \times s_2}$ to be a sparse Cauchy transform.
  Choose $S_3 \in \mathbb{R}^{n^2 \times s_3}$ to be a sparse Cauchy transform.
  Form $\widehat{A} = AS$.
  Compute $\widehat{A}_1 S_1$, $\widehat{A}_2 S_2$, and $\widehat{A}_3 S_3$
  $Y_1, Y_2, Y_3, C \leftarrow$L1PolyKSizeReduction($\widehat{A}, \widehat{A}_1 S_1, \widehat{A}_2 S_2, \widehat{A}_3 S_3, n, n, \overline{n}, s_1, s_2, s_3, k$) ▷ Algorithm 21.21
  Create $s_1 k + s_2 k + s_3 k$ variables for each entry of X_1, X_2, X_3.
  Form objective function $\|(Y_1 X_1) \otimes (Y_2 X_2) \otimes (Y_3 X_3) - C\|_F^2$.
  Run polynomial system verifier.
  return $A_1 S_1 X_1, A_2 S_2 X_2, A_3 S_3 X_3$
end procedure

problem. Next, we use the existence results (Theorem 21.12.24) and polynomial in k size reduction (Lemma 21.7.5). At the end, we relax the ℓ_1-norm objective function to a Frobenius norm objective function (Fact 21.7.1). □

Theorem 21.12.27. *Given a 3rd order tensor $A \in \mathbb{R}^{n \times n \times n}$, for any $k \geq 1$, there exists an algorithm which takes $n^{\widetilde{O}(k)} 2^{\widetilde{O}(k^3)}$ time and outputs three matrices $U, V, W \in \mathbb{R}^{n \times k}$ such that,*

$$\|U \otimes V \otimes W - A\|_u \leq O(k^{3/2}) \min_{\text{rank}-k\ A'} \|A' - A\|_u,$$

holds with probability at least 9/10.

Proof. We first choose a Gaussian matrix $S \in \mathbb{R}^{n \times \overline{n}}$ with $\overline{n} = O(n)$. By applying Corollary 21.12.23, we can reduce the original problem to a "generalized" ℓ_1 low rank approximation

Algorithm 21.39 ℓ_1-ℓ_1-ℓ_2-Low Rank Approximation Algorithm, $\widetilde{O}(k^{2/3})$

procedure L112TensorLowRankApproxK(A, n, k) ▷ Theorem 21.12.27
  $\overline{n} \leftarrow O(n)$.
  $s_1 \leftarrow s_2 \leftarrow s_3 \leftarrow \widetilde{O}(k)$.
  Choose $S \in \mathbb{R}^{n \times \overline{n}}$ to be a Gaussian matrix.
  Guess a diagonal matrix $S_1 \in \mathbb{R}^{n\overline{n} \times s_1}$ with s_1 nonzero entries. ▷ Part (III) of Theorem 21.12.24
  Guess a diagonal matrix $S_2 \in \mathbb{R}^{n\overline{n} \times s_2}$ with s_2 nonzero entries.
  Guess a diagonal matrix $S_3 \in \mathbb{R}^{n^2 \times s_3}$ with s_3 nonzero entries.
  Form $\widehat{A} = AS$.
  Compute $\widehat{A}_1 S_1$, $\widehat{A}_2 S_2$, and $\widehat{A}_3 S_3$
  $Y_1, Y_2, Y_3, C \leftarrow$L1PolyKSizeReduction($\widehat{A}, \widehat{A}_1 S_1, \widehat{A}_2 S_2, \widehat{A}_3 S_3, n, n, \overline{n}, s_1, s_2, s_3, k$) ▷ Algorithm 21.21
  Create $s_1 k + s_2 k + s_3 k$ variables for each entry of X_1, X_2, X_3.
  Form objective function $\|(Y_1 X_1) \otimes (Y_2 X_2) \otimes (Y_3 X_3) - C\|_1$.
  Run polynomial system verifier.
  return $A_1 S_1 X_1, A_2 S_2 X_2, A_3 S_3 X_3$
end procedure

problem. Next, we use the existence results (Theorem 21.12.24) and polynomial in k size reduction (Lemma 21.7.5). At the end, we solve an entry-wise ℓ_1 norm objective function directly. □

Theorem 21.12.28. *Given a 3rd order tensor $A \in \mathbb{R}^{n \times n \times n}$, for any $k \geq 1$, let $r = \widetilde{O}(k^2)$. There is an algorithm which takes $O(\mathrm{nnz}(A)n) + \widetilde{O}(n)\,\mathrm{poly}(k)$ time and outputs three matrices $U, V, W \in \mathbb{R}^{n \times r}$ such that*

$$\|U \otimes V \otimes W - A\|_u \leq \mathrm{poly}(\log n, k) \min_{\mathrm{rank}-k\ A_k} \|A_k - A\|_u,$$

holds with probability at least 9/10.

Algorithm 21.40 ℓ_1-ℓ_1-ℓ_2-Low Rank Approximation Algorithm, Bicriteria Algorithm

procedure L112TensorLowRankApproxBicteriteria(A, n, k) ▷ Theorem 21.12.28
  $\overline{n} \leftarrow O(n)$.
  $s_2 \leftarrow s_3 \leftarrow \widetilde{O}(k^5)$.
  $t_2 \leftarrow t_3 \leftarrow \widetilde{O}(k)$.
  $r \leftarrow s_2 s_3$.
  Choose $S \in \mathbb{R}^{n \times \overline{n}}$ to be a Gaussian matrix.
  Form $\widehat{A} = AS \in \mathbb{R}^{n \times n \times \overline{n}}$.
  Choose a sketching matrix $S_2 \in \mathbb{R}^{n\overline{n} \times s_2}$ with s_2 nonzero entries (Sparse Cauchy transform), for each $i \in \{2, 3\}$. ▷ Part (II) of Theorem 21.12.24
  Choose a sampling and rescaling diagonal matrix D_i according to the Lewis weights of $\widehat{A}_i S_i$ with t_i nonzero entries, for each $i \in \{2, 3\}$.
  Form $\widehat{V} \in \mathbb{R}^{n \times r}$ by setting the (i, j)-th column to be $(\widehat{A}_2 S_2)_i$.
  Form $\widehat{W} \in \mathbb{R}^{n \times r}$ by setting the (i, j)-th column to be $(A_3 S_3)_j$.
  Form matrix $B \in \mathbb{R}^{r \times t_2 t_3}$ by setting the (i, j)-th column to be the vectorization of $(T_2 \widehat{A}_2 S_2)_i \otimes (T_3 \widehat{A}_3 S_3)_j$.
  Solve $\min_U \| U \cdot B - (\widehat{A}(I, T_2, T_3))_1 \|_1$.
  return $\widehat{U}, \widehat{V}, \widehat{W}$
end procedure

Proof. We first choose a Gaussian matrix $S \in \mathbb{R}^{n \times \overline{n}}$ with $\overline{n} = O(n)$. By applying Corollary 21.12.23, we can reduce the original problem to a "generalized" ℓ_1 low rank approximation problem. Next, we use the existence results (Theorem 21.12.24) and polynomial in k size reduction (Lemma 21.7.5). At the end, we solve an entry-wise ℓ_1 norm objective function directly. □

21.13 Streaming Setting

One of the computation models which is closely related to the distributed model of computation is the streaming model. There is a growing line of work in the streaming model. Some problems are very fundamental in the streaming model such like Heavy Hitters [LNNT16, BCI+16, BCIW16], and streaming numerical linear algebra problems [CW09]. Streaming low rank matrix approximation has been extensively studied by previous work like [CW09, KL11, GP14, Lib13, KLM+14a, BWZ16, SWZ17]. In this section, we show that there is a streaming algorithm which can compute a low rank tensor approximation.

In the following, we introduce the turnstile streaming model and the turnstile streaming tensor Frobenius norm low rank approximation problem. The following gives a formal definition of the computation model we study.

Definition 21.13.1 (Turnstile model)**.** Initially, tensor $A \in \mathbb{R}^{n\times n\times n}$ is an all zero tensor. In the turnstile streaming model, there is a stream of update operations, and the i^{th} update operation is in the form $(x_i, y_i, z_i, \delta_i)$ where $x_i, y_i, z_i \in [n]$, and $\delta_i \in \mathbb{R}$ has $O(\log n)$ bits. Each $(x_i, y_i, z_i, \delta_i)$ means that A_{x_i,y_i,z_i} should be incremented by δ_i. And each entry of A has at most $O(\log n)$ bits at the end of the stream. An algorithm in this computation model is only allowed one pass over the stream. At the end of the stream, the algorithm stores a summary of A. The space complexity of the algorithm is the total number of words required to compute and store this summary while scanning the stream. Here, each word has at most $O(\log(n))$ bits.

The following is the formal definition of the problem.

Definition 21.13.2 (Turnstile model Frobenius norm rank-k tensor approximation)**.** Given tensor $A \in \mathbb{R}^{n\times n\times n}$, $k \in \mathbb{N}_+$ and an error parameter $1 > \epsilon > 0$, the goal is to design an algorithm in the streaming model of Definition 21.13.1 such that

1. Upon termination, the algorithm outputs three matrices $U^*, V^*, W^* \in \mathbb{R}^{n\times k}$.
2. U^*, V^*, W^* satisft that
$$\left\|\sum_{i=1}^{k} U_i^* \otimes V_i^* \otimes W_i^* - A\right\|_F^2 \leq (1+\epsilon) \min_{\text{rank}-k\ A'} \|A' - A\|_F^2.$$
3. The space complexity of the algorithm is as small as possible.

Theorem 21.13.1. *Suppose tensor $A \in \mathbb{R}^{n\times n\times n}$ is given in the turnstile streaming model (see Definition 21.13.1), there is an streaming algorithm (in Algorithm 21.41) which solves the problem in Definition 21.13.2 with constant success probability. In addition, the space complexity of the algorithm is* $\mathrm{poly}(k/\epsilon) + O(nk/\epsilon)$ *words.*

Proof. **Correctness.** Similar to the distributed protocol, the correctness of this streaming algorithm is also implied by Algorithm 21.2 and Algorithm 21.3 (Theorem 21.4.1.) Notice that at the end of the stream $V_1 = A_1S_1 \in \mathbb{R}^{n\times s_1}, V_2 = A_2S_2 \in \mathbb{R}^{n\times s_2}, V_3 = A_3S_3 \in \mathbb{R}^{n\times s_3}, C = A(T_1, T_2, T_3) \in \mathbb{R}^{t_1\times t_2\times t_3}$. It also means that

$$Y_1 = T_1A_1S_1, Y_2 = T_2A_2S_2, Y_3 = T_3A_3S_3.$$

According to line 26 of procedure TurnstileStreaming,

$$X_1^*, X_2^*, X_3^* = \arg\min_{X_1\in\mathbb{R}^{s_1\times k}, X_2\in\mathbb{R}^{s_2\times k}, X_3\in\mathbb{R}^{s_3\times k}} \left\|\sum_{j=1}^{k} (Y_1X_1)_j \otimes (Y_2X_2)_j \otimes (Y_3X_3)_j - C\right\|_F$$

According to Lemma 21.4.3, we have

$$\begin{aligned}
&\left\|\sum_{j=1}^{k}(Y_1X_1)_j\otimes(Y_2X_2)_j\otimes(Y_3X_3)_j-C\right\|_F^2\\
=&\left\|\sum_{j=1}^{k}(T_1A_1S_1X_1^*)_j\otimes(T_2A_2S_2X_2^*)_j\otimes(T_3A_3S_3X_3^*)_j-A(T_1,T_2,T_3)\right\|_F^2\\
\leq&(1+O(\epsilon))\min_{X_1,X_2,X_3}\left\|\sum_{j=1}^{k}(A_1S_1X_1)_j\otimes(A_2S_2X_2)_j\otimes(A_3Y_3X_3)_j-A\right\|_F^2\\
\leq&(1+O(\epsilon))\min_{U,V,W}\left\|\sum_{i=1}^{k}U_i\otimes V_i\otimes W_i-A\right\|_F^2,
\end{aligned}$$

where the last inequality follows by the proof of Theorem 21.4.1. By scaling a constant of ϵ, we complete the proof of correctness.

Space complexity. Since S_1, S_2, S_3 are w_1-wise independent, and T_1, T_2, T_3 are w_2-wise independent, the space needed to construct these sketching matrices in line 3 and line 5 of procedure TurnstileStreaming is $O(w_1 + w_2)$ words, where $w_1 = O(k), w_2 = O(1)$ (see [KVW14, CW13, Woo14b, KN14]). The cost to maintain V_1, V_2, V_3 is $O(nk/\epsilon)$ words, and the cost to maintain C is $\text{poly}(k/\epsilon)$ words.

Notice that, since each entry of A has at most $O(\log(sn))$ bits, each entry of Y_1, Y_2, Y_3, C has at most $O(\log(sn))$ bits. Due to Theorem 21.15.2, each entry of X_1^*, X_2^*, X_3^* has at most $O(\log(sn))$ bits, and the sizes of X_1^*, X_2^*, X_3^* are $\text{poly}(k/\epsilon)$ words. Thus the space cost in line 26 is $\text{poly}(k/\epsilon)$ words.

The total space cost is $\text{poly}(k/\epsilon) + O(nk/\epsilon)$ words. □

Remark 21.13.1. In the Algorithm 21.41, for each update operation, we need $O(k/\epsilon)$ time to maintain matrices V_1, V_2, V_3, and we need $\text{poly}(k/\epsilon)$ time to maintain tensor C. Thus the

Algorithm 21.41 Turnstile Frobenius Norm Low Rank Approximation Algorithm

procedure TURNSTILESTREAMING(k,$\mathcal{S}$)
  $s_1 \leftarrow s_2 \leftarrow s_3 \leftarrow O(k/\epsilon)$.
  Construct sketching matrices $S_i \in \mathbb{R}^{n^2 \times s_i}, \forall i \in [3]$ where entries of S_1, S_2, S_3 are w_1-wise independent random $N(0, 1/s_i)$ Gaussian variables.
  $t_1 \leftarrow t_2 \leftarrow t_3 \leftarrow \text{poly}(k/\epsilon)$.
  Construct sparse embedding matrices $T_i \in \mathbb{R}^{t_i \times n}, \forall i \in [3]$ where entries are w_2-wise independent.
  Initialize matrices:
  $V_i \leftarrow \{0\}^{n \times s_i}, \forall i \in [3]$.
  $C \leftarrow \{0\}^{t_1 \times t_2 \times t_3}$
  for $i \in [l]$ **do**
    Receive update operation $(x_i, y_i, z_i, \delta_i)$ from the data stream $\mathcal{S}$.
    for $r = 1 \rightarrow s_1$ **do**
      $(V_1)_{x_i,r} \leftarrow (V_1)_{x_i,r} + \delta_i \cdot (S_1)_{(y_i-1)n+z_i,r}$.
    end for
    for $r = 1 \rightarrow s_2$ **do**
      $(V_2)_{y_i,r} \leftarrow (V_2)_{y_i,r} + \delta_i \cdot (S_2)_{(z_i-1)n+x_i,r}$.
    end for
    for $r = 1 \rightarrow s_3$ **do**
      $(V_3)_{z_i,r} \leftarrow (V_3)_{z_i,r} + \delta_i \cdot (S_3)_{(x_i-1)n+y_i,r}$.
    end for
    for $r = 1 \rightarrow t_1, p = 1 \rightarrow t_2, q = 1 \rightarrow t_3$ **do**
      $C_{r,p,q} \leftarrow C_{r,p,q} + \delta_i \cdot (T_1)_{r,x_i}(T_2)_{p,y_i}(T_3)_{q,z_i}$.
    end for
  end for
  Compute $Y_1 \leftarrow T_1 V_1, Y_2 \leftarrow T_2 V_2, Y_3 \leftarrow T_3 V_3$.
  Compute $X_i^* \in \mathbb{R}^{s_i \times k}, \forall i \in [3]$ by solving
  $\min_{X_1,X_2,X_3} \|(Y_1 X_1) \otimes (Y_2 X_2) \otimes (Y_3 X_3) - C\|_F$
  Compute $U^* \leftarrow V_1 X_1^*, V^* \leftarrow V_2 X_2^*,\ W^* \leftarrow V_3 X_3^*$.
  return U^*, V^*, W^*
end procedure

update time is poly(k/ϵ). At the end of the stream, the time to compute

$$X_1^*, X_2^*, X_3^* = \arg\min_{X_1,X_2,X_3 \in \mathbb{R}^{O(k/\epsilon) \times k}} \left\| \sum_{j=1}^{k} (Y_1 X_1)_j \otimes (Y_2 X_2)_j \otimes (Y_3 X_3)_j - C \right\|_F ,$$

is exponential in poly(k/ϵ) running time since it should use a polynomial system solver. Instead of computing the rank-k solution, we can solve the following:

$$\alpha^* = \arg\min_{\alpha \in \mathbb{R}^{s_1 \times s_2 \times s_3}} \left\| \sum_{i=1}^{s_1} \sum_{j=1}^{s_2} \sum_{l=1}^{s_3} \alpha_{i,j,l} \cdot (Y_1)_i \otimes (Y_2)_j \otimes (Y_3)_l - C \right\|_F$$

which will then give

$$\sum_{i=1}^{s_1} \sum_{j=1}^{s_2} \sum_{l=1}^{s_3} \alpha^*_{i,j,l} \cdot (Y_1)_i \otimes (Y_2)_j \otimes (Y_3)_l$$

to be a rank-$O(k^3/\epsilon^3)$ bicriteria solution.

Further, similar to Theorem 21.4.8, we can solve

$$\min_{U \in \mathbb{R}^{n \times s_2 s_3}} \left\| \sum_{i=1}^{s_1} \sum_{j=1}^{s_2} U_{i+s_1(j-1)} \otimes (Y_2)_i \otimes (Y_3)_j - C \right\|_F$$

where $C = \sum_i A_i(I, T_2, T_3)$. Thus, we can obtain a rank-$O(k^2/\epsilon^2)$ in polynomial time.

Remark 21.13.2. If we choose $S_1, S_2, S_3, T_1, T_2, T_3$ to be random Cauchy matrices, then we are able to apply the entry-wise ℓ_1 norm low rank tensor approximation algorithm (see Theorem 21.7.14) in turnstile model.

21.14 Distributed Setting

Input data to large-scale machine learning and data mining tasks may be distributed across different machines. The communication cost becomes the major bottleneck of distributed protocols, and so there is a growing body of work on low rank matrix approximations in the distributed model [TD99, QOSG02, BCL05, BRB08, MBZ10, FEGK13, PMvdG+13, KVW14, BKLW14, BLS+16a, BWZ16, WZ16, SWZ17] and also many other machine learning problems such as clustering, boosting, and column subset selection [BBLM14, BLG+15, ABW17]. Thus, it is natural to ask whether our algorithm can be applied in the distributed setting. This section will discuss the distributed Frobenius norm low rank tensor approximation protocol in the so-called arbitrary-partition model (see, e.g. [KVW14, BWZ16]).

In the following, we extend the definition of the arbitrary-partition model [KVW14] to fit our tensor setting.

Definition 21.14.1 (Arbitrary-partition model [KVW14])**.** There are s machines, and the i^{th} machine holds a tensor $A_i \in \mathbb{R}^{n\times n\times n}$ as its local data tensor. The global data tensor is implicit and is denoted as $A = \sum_{i=1}^{s} A_i$. Then, we say that A is arbitrarily partitioned into s matrices distributed in the s machines. In addition, there is also a coordinator. In this model, the communication is only allowed between the machines and the coordinator. The total communication cost is the total number of words delivered between machines and the coordinator. Each word has $O(\log(sn))$ bits.

Now, let us introduce the distributed Frobenius norm low rank tensor approximation problem in the arbitrary partition model:

Definition 21.14.2 (Arbitrary-partition model Frobenius norm rank-k tensor approximation)**.** Tensor $A \in \mathbb{R}^{n\times n\times n}$ is arbitrarily partitioned into s matrices $A_1, A_2, \cdots, A_s$ distributed in s machines respectively, and $\forall i \in [s]$, each entry of A_i is at most $O(\log(sn))$ bits. Given tensor A, $k \in \mathbb{N}_+$ and an error parameter $0 < \epsilon < 1$, the goal is to find a distributed protocol in the model of Definition 21.15.1 such that

1. Upon termination, the protocol leaves three matrices $U^*, V^*, W^* \in \mathbb{R}^{n\times k}$ on the coordinator.

2. U^*, V^*, W^* satisfies that
$$\left\| \sum_{i=1}^{k} U_i^* \otimes V_i^* \otimes W_i^* - A \right\|_F^2 \leq (1+\epsilon) \min_{\text{rank}-k\ A'} \|A' - A\|_F^2.$$

3. The communication cost is as small as possible.

Theorem 21.14.1. *Suppose tensor $A \in \mathbb{R}^{n\times n\times n}$ is distributed in the arbitrary partition model (See Definition 21.15.1). There is a protocol(in Algorithm 21.43) which solves the problem in Definition 21.15.2 with constant success probability. In addition, the communication complexity of the protocol is $s(\mathrm{poly}(k/\epsilon) + O(kn))$ words.*

Proof. **Correctness.** The correctness is implied by Algorithm 21.2 and Algorithm 21.3 (Theorem 21.4.1.) Notice that $A_1 = \sum_{i=1}^{s} A_{i,1}, A_2 = \sum_{i=1}^{s} A_{i,2}, A_3 = \sum_{i=1}^{s} A_{i,3}$, which means that

$$Y_1 = T_1 A_1 S_1, Y_2 = T_2 A_2 S_2, Y_3 = T_3 A_3 S_3,$$

and

$$C = A(T_1, T_2, T_3).$$

According to line 23,

$$X_1^*, X_2^*, X_3^* = \arg\min_{X_1,X_2,X_3} \left\| \sum_{j=1}^{k} (Y_1X_1)_j \otimes (Y_2X_2)_j \otimes (Y_3X_3)_j - C \right\|_F .$$

According to Lemma 21.4.3, we have

$$\begin{aligned}
&\left\| \sum_{j=1}^{k} (T_1A_1S_1X_1^*)_j \otimes (T_2A_2S_2X_2^*)_j \otimes (T_3A_3S_3X_3^*)_j - A(T_1,T_2,T_3) \right\|_F^2 \\
\leq &(1+O(\epsilon)) \min_{X_1,X_2,X_3} \left\| \sum_{j=1}^{k} (A_1S_1X_1)_j \otimes (A_2S_2X_2)_j \otimes (A_3Y_3X_3)_j - A \right\|_F^2 \\
\leq &(1+O(\epsilon)) \min_{U,V,W} \left\| \sum_{i=1}^{k} U_i \otimes V_i \otimes W_i - A \right\|_F^2 ,
\end{aligned}$$

where the last inequality follows by the proof of Theorem 21.4.1. By scaling a constant of ϵ, we complete the proof of correctness.

Communication complexity. Since S_1, S_2, S_3 are w_1-wise independent, and T_1, T_2, T_3 are w_2-wise independent, the communication cost of sending random seeds in line 5 is $O(s(w_1+w_2))$ words, where $w_1 = O(k), w_2 = O(1)$ (see [KVW14, CW13, Woo14b, KN14]). The communication cost in line 18 is $s \cdot \mathrm{poly}(k/\epsilon)$ words due to $T_1A_{i,1}S_1, T_2A_{i,2}S_2, T_3A_{i,3}S_3 \in \mathbb{R}^{\mathrm{poly}(k/\epsilon)\times O(k/\epsilon)}$ and $C_i = A_i(T_1,T_2,T_3) \in \mathbb{R}^{\mathrm{poly}(k/\epsilon)\times\mathrm{poly}(k/\epsilon)\times\mathrm{poly}(k/\epsilon)}$.

Notice that, since $\forall i \in [s]$ each entry of A_i has at most $O(\log(sn))$ bits, each entry of Y_1, Y_2, Y_3, C has at most $O(\log(sn))$ bits. Due to Theorem 21.15.2, each entry of X_1^*, X_2^*, X_3^* has at most $O(\log(sn))$ bits, and the sizes of X_1^*, X_2^*, X_3^* are $\mathrm{poly}(k/\epsilon)$ words. Thus the communication cost in line 24 is $s \cdot \mathrm{poly}(k/\epsilon)$ words.

Finally, since $\forall i \in [s], U_i^*, V_i^*, W_i^* \in \mathbb{R}^{n\times k}$, the communication here is at most $O(skn)$ words. The total communication cost is $s(\mathrm{poly}(k/\epsilon) + O(kn))$ words. □

Remark 21.14.1. If we slightly change the goal in Definition 21.15.2 to the following: the coordinator does not need to output U^*, V^*, W^*, but each machine i holds U_i^*, V_i^*, W_i^* such that $U^* = \sum_{i=1}^s U_i^*, V^* = \sum_{i=1}^s V_i^*, W^* = \sum_{i=1}^s W_i^*$, then the protocol shown in Algorithm 21.43 does not have to do the line 28. Thus the total communication cost is at most $s \cdot \mathrm{poly}(k/\epsilon)$ words in this setting.

Remark 21.14.2. Algorithm 21.43 needs exponential in $\mathrm{poly}(k/\epsilon)$ running time since it solves a polynomial solver in line 23. Instead of solving line 23, we can solve the following optimization problem:

$$\alpha^* = \arg\min_{\alpha \in \mathbb{R}^{s_1 \times s_2 \times s_3}} \left\| \sum_{i=1}^{s_1} \sum_{j=1}^{s_2} \sum_{l=1}^{s_3} \alpha_{i,j,l} \cdot (Y_1)_i \otimes (Y_2)_j \otimes (Y_3)_l - C \right\|_F .$$

Since it is actually a regression problem, it only takes polynomial running time to get α^*. And according to Lemma 21.4.5,

$$\sum_{i=1}^{s_1} \sum_{j=1}^{s_2} \sum_{l=1}^{s_3} \alpha_{i,j,l}^* \cdot (Y_1)_i \otimes (Y_2)_j \otimes (Y_3)_l$$

gives a rank-$O(k^3/\epsilon^3)$ bicriteria solution.

Further, similar to Theorem 21.4.8, we can solve

$$\min_{U \in \mathbb{R}^{n \times s_2 s_3}} \left\| \sum_{i=1}^{s_1} \sum_{j=1}^{s_2} U_{i+s_1(j-1)} \otimes (Y_2)_i \otimes (Y_3)_j - C \right\|_F ,$$

where $C = \sum_i A_i(I, T_2, T_3)$. Thus, we can obtain a rank-$O(k^2/\epsilon^2)$ in polynomial time.

Remark 21.14.3. If we select sketching matrices $S_1, S_2, S_3, T_1, T_2, T_3$ to be random Cauchy matrices, then we are able to compute distributed entry-wise ℓ_1 norm rank-k tensor approximation (see Theorem 21.7.14). The communication cost is still $s(\mathrm{poly}(k/\epsilon) + O(kn))$ words. If we only require a bicriteria solution, then it only needs polynomial running time.

Using similar techniques as in the proof of Theorem 21.4.44, we can obtain:

Theorem 21.14.2. *Let* $\max_i\{t_i, d_i\} \leq n$. *Given a* $t_1 \times t_2 \times t_3$ *tensor* A *and three matrices: a* $t_1 \times d_1$ *matrix* T_1, *a* $t_2 \times d_2$ *matrix* T_2, *and a* $t_3 \times d_3$ *matrix* T_3. *For any* $\delta > 0$, *if there exists a solution to*

$$\min_{X_1,X_2,X_3} \left\| \sum_{i=1}^{k} (T_1X_1)_i \otimes (T_2X_2)_i \otimes (T_3X_3)_i - A \right\|_F^2 := \mathrm{OPT},$$

and each entry of X_i *can be expressed using* $O(\log n)$ *bits, then there exists an algorithm that takes* $\mathrm{poly}(\log n) \cdot 2^{O(d_1k+d_2k+d_3k)}$ *time and outputs three matrices:* $\widehat{X}_1$, $\widehat{X}_2$, *and* $\widehat{X}_3$ *such that* $\|(T_1\widehat{X}_1) \otimes (T_2\widehat{X}_2) \otimes (T_3\widehat{X}_3) - A\|_F^2 = \mathrm{OPT}$.

21.15 Distributed Setting

Input data to large-scale machine learning and data mining tasks may be distributed across different machines. The communication cost becomes the major bottleneck of distributed protocols, and so there is a growing body of work on low rank matrix approximations in the distributed model [TD99, QOSG02, BCL05, BRB08, MBZ10, FEGK13, PMvdG+13, KVW14, BKLW14, BLS+16a, BWZ16, WZ16, SWZ17] and also many other machine learning problems such as clustering, boosting, and column subset selection [BBLM14, BLG+15, ABW17]. Thus, it is natural to ask whether our algorithm can be applied in the distributed setting. This section will discuss the distributed Frobenius norm low rank tensor approximation protocol in the so-called arbitrary-partition model (see, e.g. [KVW14, BWZ16]).

In the following, we extend the definition of the arbitrary-partition model [KVW14] to fit our tensor setting.

Definition 21.15.1 (Arbitrary-partition model [KVW14])**.** There are s machines, and the i^{th} machine holds a tensor $A_i \in \mathbb{R}^{n\times n\times n}$ as its local data tensor. The global data tensor is implicit and is denoted as $A = \sum_{i=1}^{s} A_i$. Then, we say that A is arbitrarily partitioned into s matrices distributed in the s machines. In addition, there is also a coordinator. In this model, the communication is only allowed between the machines and the coordinator. The total communication cost is the total number of words delivered between machines and the coordinator. Each word has $O(\log(sn))$ bits.

Now, let us introduce the distributed Frobenius norm low rank tensor approximation problem in the arbitrary partition model:

Definition 21.15.2 (Arbitrary-partition model Frobenius norm rank-k tensor approximation)**.** Tensor $A \in \mathbb{R}^{n\times n\times n}$ is arbitrarily partitioned into s matrices $A_1, A_2, \cdots, A_s$ distributed in s machines respectively, and $\forall i \in [s]$, each entry of A_i is at most $O(\log(sn))$ bits. Given tensor A, $k \in \mathbb{N}_+$ and an error parameter $0 < \epsilon < 1$, the goal is to find a distributed protocol in the model of Definition 21.15.1 such that

1. Upon termination, the protocol leaves three matrices $U^*, V^*, W^* \in \mathbb{R}^{n\times k}$ on the coordinator.

2. U^*, V^*, W^* satisfies that

$$\left\| \sum_{i=1}^{k} U_i^* \otimes V_i^* \otimes W_i^* - A \right\|_F^2 \leq (1+\epsilon) \min_{\text{rank}-k\ A'} \|A' - A\|_F^2.$$

3. The communication cost is as small as possible.

Theorem 21.15.1. *Suppose tensor $A \in \mathbb{R}^{n\times n\times n}$ is distributed in the arbitrary partition model (See Definition 21.15.1). There is a protocol(in Algorithm 21.43) which solves the problem in Definition 21.15.2 with constant success probability. In addition, the communication complexity of the protocol is $s(\mathrm{poly}(k/\epsilon) + O(kn))$ words.*

Proof. **Correctness.** The correctness is implied by Algorithm 21.2 and Algorithm 21.3 (Theorem 21.4.1.) Notice that $A_1 = \sum_{i=1}^{s} A_{i,1}, A_2 = \sum_{i=1}^{s} A_{i,2}, A_3 = \sum_{i=1}^{s} A_{i,3}$, which means that

$$Y_1 = T_1 A_1 S_1, Y_2 = T_2 A_2 S_2, Y_3 = T_3 A_3 S_3,$$

and

$$C = A(T_1, T_2, T_3).$$

According to line 23,

$$X_1^*, X_2^*, X_3^* = \arg\min_{X_1,X_2,X_3} \left\| \sum_{j=1}^{k} (Y_1 X_1)_j \otimes (Y_2 X_2)_j \otimes (Y_3 X_3)_j - C \right\|_F.$$

According to Lemma 21.4.3, we have

$$\begin{aligned}
&\left\| \sum_{j=1}^{k} (T_1 A_1 S_1 X_1^*)_j \otimes (T_2 A_2 S_2 X_2^*)_j \otimes (T_3 A_3 S_3 X_3^*)_j - A(T_1, T_2, T_3) \right\|_F^2 \\
\leq & (1 + O(\epsilon)) \min_{X_1,X_2,X_3} \left\| \sum_{j=1}^{k} (A_1 S_1 X_1)_j \otimes (A_2 S_2 X_2)_j \otimes (A_3 Y_3 X_3)_j - A \right\|_F^2 \\
\leq & (1 + O(\epsilon)) \min_{U,V,W} \left\| \sum_{i=1}^{k} U_i \otimes V_i \otimes W_i - A \right\|_F^2,
\end{aligned}$$

where the last inequality follows by the proof of Theorem 21.4.1. By scaling a constant of ϵ, we complete the proof of correctness.

Communication complexity. Since S_1, S_2, S_3 are w_1-wise independent, and T_1, T_2, T_3 are w_2-wise independent, the communication cost of sending random seeds in line 5 is $O(s(w_1 + w_2))$ words, where $w_1 = O(k), w_2 = O(1)$ (see [KVW14, CW13, Woo14b, KN14]). The communication cost in line 18 is $s \cdot \text{poly}(k/\epsilon)$ words due to $T_1 A_{i,1} S_1, T_2 A_{i,2} S_2, T_3 A_{i,3} S_3 \in \mathbb{R}^{\text{poly}(k/\epsilon) \times O(k/\epsilon)}$ and $C_i = A_i(T_1, T_2, T_3) \in \mathbb{R}^{\text{poly}(k/\epsilon) \times \text{poly}(k/\epsilon) \times \text{poly}(k/\epsilon)}$.

Notice that, since $\forall i \in [s]$ each entry of A_i has at most $O(\log(sn))$ bits, each entry of Y_1, Y_2, Y_3, C has at most $O(\log(sn))$ bits. Due to Theorem 21.15.2, each entry of X_1^*, X_2^*, X_3^* has at most $O(\log(sn))$ bits, and the sizes of X_1^*, X_2^*, X_3^* are $\text{poly}(k/\epsilon)$ words. Thus the communication cost in line 24 is $s \cdot \text{poly}(k/\epsilon)$ words.

Finally, since $\forall i \in [s], U_i^*, V_i^*, W_i^* \in \mathbb{R}^{n \times k}$, the communication here is at most $O(skn)$ words. The total communication cost is $s(\text{poly}(k/\epsilon) + O(kn))$ words. □

Remark 21.15.1. If we slightly change the goal in Definition 21.15.2 to the following: the coordinator does not need to output U^*, V^*, W^*, but each machine i holds U_i^*, V_i^*, W_i^* such that $U^* = \sum_{i=1}^s U_i^*, V^* = \sum_{i=1}^s V_i^*, W^* = \sum_{i=1}^s W_i^*$, then the protocol shown in Algorithm 21.43 does not have to do the line 28. Thus the total communication cost is at most $s \cdot \text{poly}(k/\epsilon)$ words in this setting.

Remark 21.15.2. Algorithm 21.43 needs exponential in $\text{poly}(k/\epsilon)$ running time since it solves a polynomial solver in line 23. Instead of solving line 23, we can solve the following opti-

mization problem:

$$\alpha^* = \arg\min_{\alpha \in \mathbb{R}^{s_1 \times s_2 \times s_3}} \left\| \sum_{i=1}^{s_1} \sum_{j=1}^{s_2} \sum_{l=1}^{s_3} \alpha_{i,j,l} \cdot (Y_1)_i \otimes (Y_2)_j \otimes (Y_3)_l - C \right\|_F .$$

Since it is actually a regression problem, it only takes polynomial running time to get α^*. And according to Lemma 21.4.5,

$$\sum_{i=1}^{s_1} \sum_{j=1}^{s_2} \sum_{l=1}^{s_3} \alpha^*_{i,j,l} \cdot (Y_1)_i \otimes (Y_2)_j \otimes (Y_3)_l$$

gives a rank-$O(k^3/\epsilon^3)$ bicriteria solution.

Further, similar to Theorem 21.4.8, we can solve

$$\min_{U \in \mathbb{R}^{n \times s_2 s_3}} \left\| \sum_{i=1}^{s_1} \sum_{j=1}^{s_2} U_{i+s_1(j-1)} \otimes (Y_2)_i \otimes (Y_3)_j - C \right\|_F ,$$

where $C = \sum_i A_i(I, T_2, T_3)$. Thus, we can obtain a rank-$O(k^2/\epsilon^2)$ in polynomial time.

Remark 21.15.3. If we select sketching matrices $S_1, S_2, S_3, T_1, T_2, T_3$ to be random Cauchy matrices, then we are able to compute distributed entry-wise ℓ_1 norm rank-k tensor approximation (see Theorem 21.7.14). The communication cost is still $s(\mathrm{poly}(k/\epsilon) + O(kn))$ words. If we only require a bicriteria solution, then it only needs polynomial running time.

Using similar techniques as in the proof of Theorem 21.4.44, we can obtain:

Theorem 21.15.2. *Let $\max_i\{t_i, d_i\} \leq n$. Given a $t_1 \times t_2 \times t_3$ tensor A and three matrices: a $t_1 \times d_1$ matrix T_1, a $t_2 \times d_2$ matrix T_2, and a $t_3 \times d_3$ matrix T_3. For any $\delta > 0$, if there exists a solution to*

$$\min_{X_1, X_2, X_3} \left\| \sum_{i=1}^{k} (T_1 X_1)_i \otimes (T_2 X_2)_i \otimes (T_3 X_3)_i - A \right\|_F^2 := \mathrm{OPT},$$

and each entry of X_i can be expressed using $O(\log n)$ bits, then there exists an algorithm that takes $\mathrm{poly}(\log n) \cdot 2^{O(d_1k+d_2k+d_3k)}$ time and outputs three matrices: $\widehat{X}_1$, $\widehat{X}_2$, and $\widehat{X}_3$ such that $\|(T_1\widehat{X}_1) \otimes (T_2\widehat{X}_2) \otimes (T_3\widehat{X}_3) - A\|_F^2 = \mathrm{OPT}$.

Algorithm 21.42 Distributed Frobenius Norm Low Rank Approximation Protocol

procedure DISTRIBUTEDFNORMLOWRANKAPPROXPROTOCOL(A,ϵ,k,s)

$A \in \mathbb{R}^{n \times n \times n}$ was arbitrarily partitioned into s matrices $A_1, \cdots, A_s \in \mathbb{R}^{n \times n \times n}$ on s machines.

Coordinator	**Machines** i
Chooses a random seed.	
Sends it to all machines.	
$- - - - - - - - - - >$	
	$s_i \leftarrow O(k/\epsilon)$, $\forall i \in [3]$.
	Agree on $S_i \in \mathbb{R}^{n^2 \times s_i}$, $\forall i \in [3]$
	which are w_1-wise independent random
	$N(0, 1/s_i)$ Gaussian matrices.
	$t_i \leftarrow \text{poly}(k/\epsilon)$, $\forall i \in [3]$.
	Agree on $T_i \in \mathbb{R}^{t_i \times n}$, $\forall i \in [3]$
	which are w_2-wise independent random
	sparse embedding matrices.
	Compute $Y_{i,1} \leftarrow T_1 A_{i,1} S_1$,
	$Y_{i,2} \leftarrow T_2 A_{i,2} S_2, Y_{i,3} \leftarrow T_3 A_{i,3} S_3$.
	Send $Y_{i,1}, Y_{i,2}, Y_{i,3}$ to the coordinator.
	Send $C_i \leftarrow A_i(T_1, T_2, T_3)$ to the coordinator.
$< - - - - - - - - - -$	
Compute $Y_1 \leftarrow \sum_{i=1}^{s} Y_{i,1}, Y_2 \leftarrow \sum_{i=1}^{s} Y_{i,2}$,	
$Y_3 \leftarrow \sum_{i=1}^{s} Y_{i,3}$, $C \leftarrow \sum_{i=1}^{s} C_i$.	
Compute X_1^*, X_2^*, X_3^* by solving	
$\min_{X_1, X_2, X_3} \|(Y_1 X_1) \otimes (Y_2 X_2) \otimes (Y_3 X_3) - C\|_F$	
Send X_1^*, X_2^*, X_3^* to machines.	
$- - - - - - - - - - >$	
	Compute $U_i^* \leftarrow A_{i,1} S_1 X_1^*$,
	$V_i^* \leftarrow A_{i,2} S_2 X_2^*$, $W_i^* \leftarrow A_{i,3} S_3 X_3^*$.
	Send U_i^*, V_i^*, W_i^* to the coordinator.
$< - - - - - - - - - -$	
Compute $U^* \leftarrow \sum_{i=1}^{s} U_i^*$.	
Compute $V^* \leftarrow \sum_{i=1}^{s} V_i^*$.	
Compute $W^* \leftarrow \sum_{i=1}^{s} W_i^*$.	
return U^*, V^*, W^*.	

end procedure

Algorithm 21.43 Distributed Frobenius Norm Low Rank Approximation Protocol

procedure DISTRIBUTEDFNORMLOWRANKAPPROXPROTOCOL(A,ϵ,k,s)
$A \in \mathbb{R}^{n \times n \times n}$ was arbitrarily partitioned into s matrices $A_1, \cdots, A_s \in \mathbb{R}^{n \times n \times n}$ on s machines.

Coordinator	**Machines** i
Chooses a random seed.	
Sends it to all machines.	
$- - - - - - - - - - >$	
	$s_i \leftarrow O(k/\epsilon)$, $\forall i \in [3]$.
	Agree on $S_i \in \mathbb{R}^{n^2 \times s_i}$, $\forall i \in [3]$
	which are w_1-wise independent random
	$N(0, 1/s_i)$ Gaussian matrices.
	$t_i \leftarrow \text{poly}(k/\epsilon)$, $\forall i \in [3]$.
	Agree on $T_i \in \mathbb{R}^{t_i \times n}$, $\forall i \in [3]$
	which are w_2-wise independent random
	sparse embedding matrices.
	Compute $Y_{i,1} \leftarrow T_1 A_{i,1} S_1$,
	$Y_{i,2} \leftarrow T_2 A_{i,2} S_2, Y_{i,3} \leftarrow T_3 A_{i,3} S_3$.
	Send $Y_{i,1}, Y_{i,2}, Y_{i,3}$ to the coordinator.
	Send $C_i \leftarrow A_i(T_1, T_2, T_3)$ to the coordinator.
$< - - - - - - - - - -$	
Compute $Y_1 \leftarrow \sum_{i=1}^{s} Y_{i,1}, Y_2 \leftarrow \sum_{i=1}^{s} Y_{i,2}$,	
$Y_3 \leftarrow \sum_{i=1}^{s} Y_{i,3}$, $C \leftarrow \sum_{i=1}^{s} C_i$.	
Compute X_1^*, X_2^*, X_3^* by solving	
$\min_{X_1, X_2, X_3} \|(Y_1 X_1) \otimes (Y_2 X_2) \otimes (Y_3 X_3) - C\|_F$	
Send X_1^*, X_2^*, X_3^* to machines.	
$- - - - - - - - - - >$	
	Compute $U_i^* \leftarrow A_{i,1} S_1 X_1^*$,
	$V_i^* \leftarrow A_{i,2} S_2 X_2^*$, $W_i^* \leftarrow A_{i,3} S_3 X_3^*$.
	Send U_i^*, V_i^*, W_i^* to the coordinator.
$< - - - - - - - - - -$	
Compute $U^* \leftarrow \sum_{i=1}^{s} U_i^*$.	
Compute $V^* \leftarrow \sum_{i=1}^{s} V_i^*$.	
Compute $W^* \leftarrow \sum_{i=1}^{s} W_i^*$.	
return U^*, V^*, W^*.	

end procedure

Chapter 22

Orthogonal Tensor Decomposition

A recent work [Wang, Tung, Smola, and Anandkumar, NIPS 2015] gives the fastest known algorithms for orthogonal tensor decomposition with provable guarantees. Their algorithm is based on computing sketches of the input tensor, which requires reading the entire input. We show in a number of cases one can achieve the same theoretical guarantees in sublinear time, i.e., even without reading most of the input tensor. Instead of using sketches to estimate inner products in tensor decomposition algorithms, we use importance sampling. To achieve sublinear time, we need to know the norms of tensor slices, and we show how to do this in a number of important cases. For symmetric tensors $A = \sum_{i=1}^{k} \lambda_i u_i^{\otimes p}$ with $\lambda_i > 0$ for all i, we estimate such norms in sublinear time whenever p is even. For the important case of $p = 3$ and small values of k, we can also estimate such norms. For asymmetric tensors sublinear time is not possible in general, but we show if the tensor slice norms are just slightly below $\|A\|_F$ then sublinear time is again possible. One of the main strengths of our work is empirical - in a number of cases our algorithm is orders of magnitude faster than existing methods with the same accuracy.

22.1 Introduction

Tensors are a powerful tool for dealing with multi-modal and multi-relational data. In recommendation systems, often using more than two attributes can lead to better recommendations. This could occur, for example, in Groupon where one could look at users, activities, and time (season, time of day, weekday/weekend, etc.), as three attributes to base predictions on (see [Moi14b] for a discussion). Similar to low rank matrix approximation, we seek a tensor decomposition to succinctly store the tensor and to apply it quickly. A popular decomposition method is the canonical polyadic decomposition, i.e., the CANDECOMP/PARAFAC (CP) decomposition, where the tensor is decomposed into a sum of rank-1 components [Har70]. We refer the reader to [WTSA15], where applications of CP including data mining, computational neuroscience, and statistical learning for latent variable models are mentioned.

A natural question, given the emergence of large data sets, is whether such decompositions can be performed quickly. There are a number of works on this topic [Tso10, PTC13, CV14, KPHF12, HNH+13, BS15, WLSH14]. Most related to ours are several recent works of Wang et al. [WTSA15] and Tung et al. [TWZS15], in which it is shown how to significantly speed up this decomposition for orthogonal tensor decomposition using the randomized technique of linear sketching [PP13]. In this work we also focus on orthogonal tensor decomposition. The idea in [WTSA15] is to create a succinct sketch of the input tensor, from which one can then perform implicit tensor decomposition by approximating inner products in existing decomposition methods.

Existing methods, like the power method, involve computing the inner product of a vector, which is now a rank-1 matrix, with another vector, which is now a slice of a

tensor. Such inner products can be approximated much faster by instead computing the inner product of the sketched vectors, which have significantly lower dimension. One can also replace the sketching with sampling to approximate inner products; we discuss some sampling schemes [Tso10, BS15] below and compare them to our work.

22.1.1 Our Contributions

We show in a number of important cases, one can achieve the same theoretical guarantees in the work of Wang et al. [WTSA15] (which was applied later by Tung et al. [TWZS15]), in *sublinear time*, that is, without reading most of the input tensor. While previous work needs to walk through the input at least once to create a sketch, we show one can instead perform *importance sampling* of the tensor based on the current iterate, together with reading a few entries of the tensor which help us learn the norms of tensor slices. We use a version of ℓ_2-sampling for our importance sampling. One source of speedup in our work and in Wang et al. [WTSA15] comes from approximating inner products in iterations in the robust tensor power method (see below). To estimate $\langle u, v\rangle$ for n-dimensional vectors u and v, their work computes sketches $S(u)$ and $S(v)$ and approximates $\langle u, v\rangle \approx \langle S(u), S(v)\rangle$. Instead, if one has u, one can sample coordinates i proportional to u_i^2, which is known as ℓ_2-sampling [MW10, CHW12]. One estimates $\langle u, v\rangle$ as $\frac{v_i\|u\|_2^2}{u_i}$, which is unbiased and has variance $O(\|u\|_2^2\|v\|_2^2)$. These guarantees are similar to those using sketching, though the constants are significantly smaller (see below), and unlike sketching, one does not need to read the entire tensor to perform such sampling.

Symmetric Tensors. As in [WTSA15], we focus on orthogonal tensor decomposition of symmetric tensors, though we explain the extension to the asymmetric case below. Symmetric tensors arise in engineering applications, for example, to represent the symmetric tensor field of stress, strain, and anisotropic conductivity. Another example is diffusion MRI in which one uses symmetric tensors to describe diffusion in the brain or other parts of the body. In spectral methods symmetric tensors are exactly those that come up in Latent

Dirichlet Allocation problems. Although one can symmetrize a tensor using simple matrix operations (see, e.g., [AGH+14]), we cannot do this in sublinear time.

In orthogonal tensor decompostion of a symmetric matrix, there is an underlying $n \times n \cdots n$ tensor $A^* = \sum_{i=1}^{k} \lambda_i v_i^{\otimes p}$, and the input tensor is $A = A^* + E$, where $\|E\|_2 \leq \epsilon$. We have $\lambda_1 > \lambda_2 > \cdots > \lambda_k > 0$ and that $\{v_i\}_{i=1}^{k}$ is a set of orthonormal vectors. The goal is to reconstruct approximations $\widehat{v}_i$ to the vectors v_i, and approximations $\widehat{\lambda}_i$ to the λ_i. Our results naturally generalize to tensors with different lengths in different dimensions. For simplicity, we first focus on order $p = 3$.

In the robust tensor power method [AGH+14], one generates a random initial vector u, and performs T update steps $\widehat{u} = A(I, u, u)/\|A(I, u, u)\|_2$, where

$$A(I, u, u) = \Big[\sum_{j=1}^{n}\sum_{\ell=1}^{n} A_{1,j,\ell}u_j u_\ell, \sum_{j=1}^{n}\sum_{\ell=1}^{n} A_{2,j,\ell}u_j u_\ell, \cdots, \sum_{j=1}^{n}\sum_{\ell=1}^{n} A_{n,j,\ell}u_j u_\ell\Big].$$

The matrices $A_{1,*,*}, \ldots, A_{n,*,*}$ are referred to as the slices. The vector $\widehat{u}$ typically converges to the top eigenvector in a small number of iterations, and one often chooses a small number L of random initial vectors to boost confidence. Successive eigenvectors can be found by deflation. The algorithm and analysis immediately extend to higher order tensors.

We use ℓ_2-sampling to estimate $A(I, u, u)$. To achieve the same guarantees as in [WTSA15], for typical settings of parameters (constant k and several eigenvalue assumptions) naïvely one needs to take $O(n^2)$ ℓ_2-samples from u *for each slice* in each iteration, resulting in $\Omega(n^3)$ time and destroying our sublinearity. We observe that if we additionally knew the squared norms $\|A_{1,*,*}\|_F^2, \ldots, \|A_{n,*,*}\|_F^2$, then we could take $O(n^2)$ ℓ_2-samples *in total*, where we take $\frac{\|A_{i,*,*}\|_F^2}{\|A\|_F^2} \cdot O(n^2)$ ℓ_2-samples from the i-th slice in expectation. Perhaps in some

applications such norms are known or cheap to compute in a single pass, but without further assumptions, how can one obtain such norms in sublinear time?

If A is a symmetric tensor, then $A_{j,j,j} = \sum_{i=1}^{k} \lambda_i v_{i,j}^3 + E_{j,j,j}$. Note that if there were no noise, then we could read off approximations to the slice norms, since $\|A_{j,*,*}\|_F^2 = \sum_{i=1}^{k} \lambda_i^2 v_{i,j}^2$, and so $A_{j,j,j}^{2/3}$ is an approximation to $\|A_{j,*,*}\|_F^2$ up to factors depending on k and the eigenvalues. However, there is indeed noise. To obtain non-trivial guarantees, the robust tensor power method assumes $\|E\|_2 = O(1/n)$, where

$$\|E\|_2 = \sup_{\|u\|_2=\|v\|_2=\|w\|_2=1} E(u,v,w) = \sup_{\|u\|_2=\|v\|_2=\|w\|_2=1} \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} E_{i,j,k} u_i v_j w_k,$$

which in particular implies $|E_{j,j,j}| = O(1/n)$. This assumption comes from the $\Theta(1/\sqrt{n})$-correlation of the random initial vector to v_1. This noise bound does not trivialize the problem; indeed, $E_{j,j,j}$ can be chosen adversarially subject to $|E_{j,j,j}| = O(1/n)$, and if the v_i were random unit vectors and the λ_i and k were constant, then $\sum_{i=1}^{k} \lambda_i v_{i,j}^3 = O(1/n^{3/2})$, which is small enough to be completely masked by the noise $E_{j,j,j}$. Nevertheless, there is a lot of information about the slice norms. Indeed, suppose $k = 1$, $\lambda_1 = \Theta(1)$, and $\|A\|_F = 1$. Then $A_{j,j,j} = \Theta(v_{1,j}^3) + E_{j,j,j}$, and one can show $\|A_{j,*,*}\|_F^2 = \lambda_1^2 v_{1,j}^2 \pm O(1/n)$. Again using that $|E_{j,j,j}| = O(1/n)$, this implies $\|A_{j,*,*}\|_F^2 = \omega(n^{-2/3})$ if and only if $A_{j,j,j} = \omega(1/n)$, and therefore one would notice this by reading $A_{j,j,j}$. There can only be $o(n^{2/3})$ slices j for which $\|A_{j,*,*}\|_F^2 = \omega(n^{-2/3})$, since $\|A\|_F^2 = 1$. Therefore, for each of them we can afford to take $O(n^2)$ ℓ_2-samples and still have an $O(n^{2+2/3}) = o(n^3)$ sublinear running time. The remaining slices all have $\|A_{j,*,*}\|_F^2 = O(n^{-2/3})$, and therefore if we also take $O(n^{1/3})$ ℓ_2-samples from *every slice*, we will also estimate the contribution to $A(I,u,u)$ from these slices well. This is also a sublinear $O(n^{2+1/3})$ number of samples.

While the previous paragraph illustrates the idea for $k = 1$, for $k = 2$ we need to read more than the $A_{j,j,j}$ entries to decide how many ℓ_2-samples to take from a slice. The analysis is more complicated because of *sign cancellations*. Even for $k = 2$ we could have $A_{j,j,j} = \lambda_1 v_{1,j}^3 + \lambda_2 v_{2,j}^3 + E_{j,j,j}$, and if $v_{1,j} = -v_{2,j}$ then we may not detect that $\|A_{j,*,*}\|_F^2$ is large. We fix this by also reading the entries $A_{i,j,j}, A_{j,i,j}$, and $A_{j,j,i}$ for every i and j. This is still only $O(n^2)$ entries and so we are still sublinear time. Without additional assumptions, we only give a formal analysis of this for $k \in \{1, 2\}$.

More importantly, if instead of third-order symmetric tensors we consider p-th order symmetric tensors for even p, we do not have such sign cancellations. In this case we do not have any restrictions on k for estimating slice norms. One does need to show after deflation, the slice norms can still be estimated; this holds because the eigenvectors and eigenvalues are estimated sufficiently well.

We also give several per-iteration optimizations of our algorithm, based on careful implementations of generating a sorted list of random numbers and random permutations. We find empirically (see below) that we are much faster per iteration than previous sketching algorithms, in addition to not having to read the entire input tensor in a preprocessing step.

Asymmetric Tensors: For asymmetric tensors, e.g., 3rd-order tensors of the form $\sum_{i=1}^k \lambda_i u_i \otimes v_i \otimes w_i$, it is impossible to achieve sublinear time in general, since it is hard to distinguish $A = e_i \otimes e_j \otimes e_k$ for random $i, j, k \in \{1, 2, \ldots, n\}$ from $A = 0^{\otimes 3}$. We make a necessary and sufficient assumption that all the entries of the u_i are less than $n^{-\gamma}$ for an arbitrarily small constant $\gamma > 0$. In this case, all slice norms are $o(n^{-\gamma})$ and by taking $O(n^{2-\gamma})$ samples from each slice we achieve sublinear time. We can also apply such an assumption to symmetric tensors.

Empirical Results. One of the main strengths of our work is our empirical results. In each iteration we approximate $A(I, u, u)$ a total of B times independently and take the median to increase our confidence. In the notation of [WTSA15], B corresponds to the number of independent sketches used. While the median works empirically, there are some theoretical issues with it discussed in Remark 22.3.1. Also let b be the total number of ℓ_2-samples we take per iteration, which corresponds to the sketch size in the notation of [WTSA15]. We found that empirically we can set B and b to be much smaller than that in [WTSA15] and achieve the same error guarantees. One explanation for this is that the variance bound we obtain via importance sampling is a factor of $4^3 = 64$ smaller than in [WTSA15], and for p-th order tensors, a factor of 4^p smaller.

To give an idea of how much smaller we can set b and B, to achieve roughly the same squared residual norm error on the synthetic data sets of dimension 1200 for finding a good rank-1 approximation, the algorithm of [WTSA15] would need to set parameters $b = 2^{16}$ and $B = 50$, whereas we can set $b = 10 \times 1200$ and $B = 5$. Our running time is 2.595 seconds and we have no preprocessing time, whereas the algorithm of [WTSA15] has a running time of 116.3 seconds and 55.34 seconds of preprocessing time. We refer the reader to Table 22.1 in Section 22.4. In total we are over 50 times faster.

We also demonstrate our algorithm in a real-world application using real datasets, even when the datasets are sparse. Namely, we consider a spectral algorithm for Latent Dirichlet Allocation [AGH+14, AFH+12] which uses tensor decomposition as its core computational step. We show a significant speedup can be achieved on tensors occurring in applications such as LDA, and we refer the reader to Table 22.2 in Section 22.4. For example, on the wiki [WTSA15] dataset with a tensor dimension of 200, we run more than 5

times faster than the sketching-based method.

Previous Sampling Algorithms. Previous sampling-based schemes of [Tso10, BS15] do not achieve our guarantees, because [Tso10] uses uniform sampling, which does not work for tensors with spiky elements, while the non-uniform sampling in [BS15] requires touching all of the entries in the tensor and making two passes over it.

22.2 Notation.

Let $[n]$ denote $\{1, 2, \ldots, n\}$. Let $\otimes$ denote the outer product, and $u^{\otimes 3} = u \otimes u \otimes u$. Let $A \in \mathbb{R}^{n^p}$, where p is the order of tensor A and n is the dimension of tensor A. Let $\langle A, B \rangle$ denote the entry-wise inner product between two tensors $A, B \in \mathbb{R}^{n^p}$, e.g., $\langle A, B \rangle = \sum_{i_1=1}^{n} \sum_{i_2=1}^{n} \cdots \sum_{i_p=1}^{n} A_{i_1,i_2,\cdots,i_p} \cdot B_{i_1,i_2,\cdots,i_p}$. For a tensor $A \in \mathbb{R}^{n^p}$,

$$\|A\|_F = \left(\sum_{i_1=1}^{n} \sum_{i_2=1}^{n} \cdots \sum_{i_p=1}^{n} A_{i_1,\cdots,i_p}^2 \right)^{\frac{1}{2}}.$$

For random variable X let $\mathbb{E}[X]$ denote its expectation of X and $\mathbb{V}[X]$ its variance (if these quantities exist).

For two functions f, g, we use the shorthand $f \lesssim g$ (resp. $\gtrsim$) to indicate that $f \leq Cg$ (resp. $\geq$) for some absolute constant C.

22.3 Main Results

We explain the details of our main results in this section. First, we state the importance sampling lemmas for our tensor application. Second, we explain how to quickly produce a list of random tuples according to a certain distribution needed by our algorithm. Third, we combine the first and the second parts to get a fast way of approximating tensor contractions, which are used as subroutines in each iteration of the robust tensor power method. We then provide our main theoretical results, and how to estimate the slice norms needed by our main algorithm.

Importance sampling lemmas. Approximating an inner product is a simple application of importance sampling. Tensor contraction $A(u, v, w)$ can be regarded as the inner product between two n^3-dimensional vectors, and thus importance sampling can be applied. Lemma 22.3.1 suggests that we can take a few samples according to their importance, e.g., we can sample $A_{i,j,k}u_iv_jw_k$ with probability $|u_iv_jw_k|^2/\|u\|_2^2\|v\|_2^2\|w\|_2^2$. As long as the number of samples is large enough, it will approximate the true tensor contraction $\sum_i \sum_j \sum_k A_{i,j,k}u_iv_jw_k$ with small variance after a final rescaling.

Lemma 22.3.1. Suppose random variable $X = A_{i_1,\cdots,i_p} \prod_{j=1}^p u_{j,i_j} / \prod_{j=1}^p q_{j,i_j}$ with probability $\prod_{j=1}^p q_{j,i_j}$ where $q_{j,i_j} = |u_{j,i_j}|^2/\|u_j\|_2^2, \forall j \in [n], \forall i_j \in [n]$, and we take L i.i.d. samples X, denote $X_1, X_2, \cdots, X_L$. Let $Y = \frac{1}{L}\sum_{\ell=1}^L X_\ell$. Then 1. $\mathbb{E}[Y] = \langle A, u_1 \otimes u_2 \otimes \cdots \otimes u_p \rangle$, and 2. $\mathbb{V}[Y] \leq \frac{1}{L}\|A\|_F^2 \cdot \|u_1 \otimes u_2 \otimes \cdots \otimes u_p\|_F^2$.

Similarly, we also have importance sampling for each slice $A_{i,*,*}$, i.e., "face" of A.

Lemma 22.3.2. For all $i_1 \in [n]$, suppose random variable $X^{i_1} = A_{i_1,i_2,\cdots,i_p} \prod_{j=2}^p u_{j,i_j} / \prod_{j=2}^p q_{j,i_j}$ with probability $\prod_{j=2}^p q_{j,i_j}$ where $q_{j,i_j} = |u_{j,i_j}|^2/\|u_j\|_2^2$, $\forall j \in \{2, 3, \cdots, n\}$, $\forall i_j \in [n]$ and we

take L_{i_1} i.i.d. samples of X^{i_1}, say $X_1^{i_1}, X_2^{i_1}, \cdots, X_L^{i_1}$. Let $Y^{i_1} = \frac{1}{L_{i_1}} \sum_{\ell=1}^{L} X_\ell^i$. Then 1. $\mathbb{E}[Y^i] = \langle A_{i_1,*,\cdots,*}, u_2 \otimes \cdots \otimes u_p \rangle$, and 2. $\mathbb{V}[Y^{i_1}] \leq \frac{1}{L_{i_1}} \|A_{i,*,\cdots,*}\|_F^2 \|u_2 \otimes \cdots \otimes u_p\|_F^2$.

Generating importance samples in linear time. We need an efficient way to sample indices of a vector based on their importance. We view this problem as follows: imagine $[0, 1]$ is divided into z "bins" with different lengths corresponding to the probability of selecting each bin, where z is the number of indices in a probability vector. We generate m random numbers uniformly from $[0, 1]$ and see which bin each random number belongs to. If a random number is in bin i, we sample the i-th index of a vector.

There are known algorithms [BP12, Wal77] to solve this problem in $O(z+m)$ time. We give an alternative algorithm GenRandTuples in Appendix 22.5. Our algorithm combines Bentley and Saxe's algorithm [BS80b] for efficiently generating m sorted random numbers in $O(m)$ time, and Knuth's shuffling algorithm [Knu69] for generating a random permutation of $[m]$ in $O(m)$ time. We use the notation CumProb(v, w) and CumProb(u, v, w) for the algorithm creating the distributions on $\mathbb{R}^{n^2}$ and $\mathbb{R}^{n^3}$ of Lemma 22.3.2 and Lemma 22.3.1, respectively. We note that naïvely applying previous algorithms would require $z = O(n^2)$ and $z = O(n^3)$ time to form these two distributions, but we can take $O(m)$ samples from them implicitly in $O(n + m)$ time.

Fast approximate tensor contractions. We propose a fast way to approximately compute tensor contractions $A(I, v, w)$ and $A(u, v, w)$ with a sublinear number of samples of A, as shown in Alogrithm 22.3 and Algorithm 22.4. Naïvely computing tensor contractions using all of the entries of A gives an exact answer but could take n^3 time. Also, to keep

our algorithm sublinear time, we never explicitly compute the deflated tensor; rather we represent it implicitly and sample from it.

The following theorem gives the error bounds of ApproxTIvw and ApproxTuvw (in Algorithm 22.3 and 22.4). Let $\widehat{b}_i$ be the number samples we take from slice $i \in [n]$ in ApproxTIvw, and let $\widehat{b}$ denote the total number of samples in our algorithm.

Theorem 22.3.3. For any $p \geq 3$, given tensor $A \in \mathbb{R}^{n^p}$, for any unit vector $u, v \in \mathbb{R}^n$, there is an algorithm that takes $\widehat{b}_i$ samples from i-th slice and it is able to output a value $\overline{A}(u, \cdots, u) \in \mathbb{R}$ and a vector $\overline{A}(I, u, \cdots, u) \in \mathbb{R}^n$ such that for any $b > 0$ if $\widehat{b}_i \geq b\|A_{i,*,\cdots,*}\|_F^2/\|A\|_F^2$ then the following bounds hold:

$$\mathbb{E}[\|\overline{E}(I, u, \cdots, u)\|_2^2] \leq n\|A\|_F^2/b, \text{ and } \mathbb{E}[|\overline{E}(u, \cdots, u)|^2] \leq \|A\|_F^2/b. \tag{22.1}$$

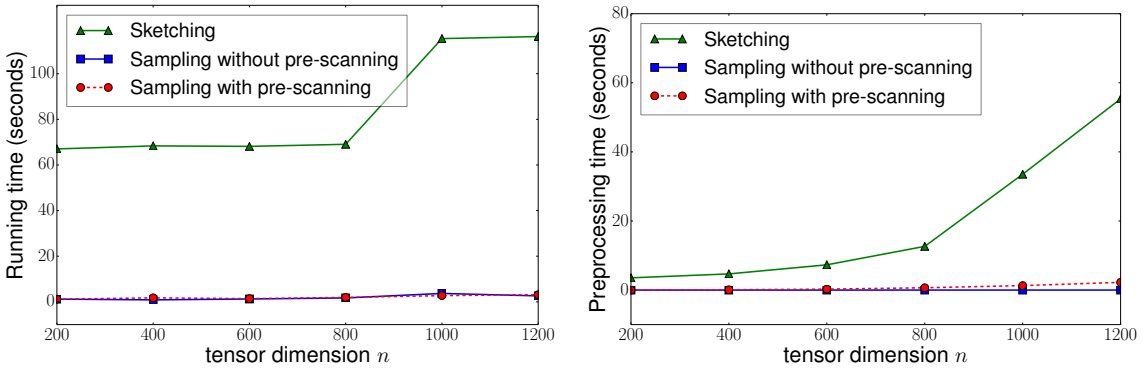
and

$$\mathbb{E}[|v^\top \overline{E}(I, u, \cdots, u)|^2] \leq \|A\|_F^2/b. \tag{22.2}$$

where $\overline{E} := \overline{A} - A$.

Eq. (22.2) can be obtained by observing that each random variable $[\overline{E}(I, u, \cdots, u)]_i$ is independent and so $\mathbb{E}[|v^\top \overline{E}(I, u, \cdots, u)|^2] = \sum_{i=1}^n v_i^2 \frac{\|A_{i,*,\cdots,*}\|_F^2}{\widehat{b}_i} \leq (\sum_{i=1}^n v_i^2)\frac{\|A\|_F^2}{b} = \frac{\|A\|_F^2}{b}$.

Remark 22.3.1. In [WTSA15], the coordinate-wise median of B estimates to the $A(I, v, w)$ is used to boost the success probability. There appears to be a gap [Wan16] in their argument as it is unclear how to achieve (22.2) after taking a coordinate-wise median, which is (7) in Theorem 1 of [WTSA15]. To fix this, we instead pay a factor proportional to the number of iterations in Algorithm 22.5 in the sample complexity $\widehat{b}$. Since we have expectation bounds on the quantities in Theorem 22.3.3, we can apply a Markov bound and a union bound across



(a) (b)

Figure 22.1: Sketching v.s. importance sampling. Running time with growing dimension

all iterations. This suffices for our main theorem concerning sublinear time below. One can obtain high probability bounds by running Algorithm 22.5 multiple times independently, and taking coordinate-wise medians of the output eigenvectors. Empirically, our algorithm works even if we take the median in each iteration, which is done in line 10 in Algorithm 22.3.

Replacing Theorem 1 in [WTSA15] by our Theorem 22.3.3, the rest of the analysis in [WTSA15] is unchanged. Our Algorithm 22.5 is the same as the sketching-based robust tensor power method in [WTSA15], except for lines 9, 11, 15, and 17, where the sketching-based approximate tensor contraction is replaced by our importance sampling procedures ApproxTuvw and ApproxTIvw. Rather than use Theorem 2 of Wang et al. [WTSA15], the main theorem concerning the correctness of the robust tensor decomposition algorithm, we use a recent improvement of it by Wang and Anandkumar in Theorems 4.1 and 4.2 of [WA16], which states general guarantees for any algorithm satisfying per iteration noise guarantees. These theorems also remove many of the earlier eigenvalue assumptions in Theorem 2 of [WTSA15].

We combine techniques from [WTSA15], [WA16] and [AGH+14] and extend the robust tensor power method analysis to any order $p \geq 3$.

Theorem 22.3.4. (Main, Abitrary order robust tensor power method) For any $p \geq 3$, $k \geq 1$, for any tensor $A = A^* + E \in \mathbb{R}^{n^p}$, where $A^* = \sum_{i=1}^k \lambda_i v_i^{\otimes p}$ with $\lambda_i > 0$ and orthonormal basis vectors $\{v_1, \ldots, v_k\} \subseteq \mathbb{R}^n$, $n \geq k$. Let λ_1, λ_k be the largest and smallest values in $\{\lambda_i\}_{i=1}^k$ and $\{\widehat{\lambda}_i, \widehat{v}_i\}_{i=1}^k$ be outputs of the robust tensor power method. For any sufficiently large constant $c_0 \geq 100$, there exists a sufficiently small constant $c > 0$, for any $\epsilon \in (0, c\lambda_k/(c_0 p^2 k n^{(p-2)/2})$ if E satisfies that

$$\|E\|_2 \leq \epsilon/(c_0\sqrt{n}) \tag{22.3}$$

and $T = \Omega(\log(\lambda_1 n/\epsilon)), L = \Omega(k\log(k))$, then with probability at least $9/10$, there exists a permutation $\pi : [k] \to [k]$, such that $\forall i \in [k]$,

$$|\lambda_i - \widehat{\lambda}_{\pi(i)}| \leq \epsilon, \quad \|v_i - \widehat{v}_{\pi(i)}\|_2 \leq \epsilon/\lambda_i. \tag{22.4}$$

Combining the previous theorem with our importance sampling analysis, we obtain:

Theorem 22.3.5 (Main, Sublinear time importance sampling robust tensor power method)**.** *Assume the notation of Theorem 22.3.4. For $p = 3$, there exists some sufficiently small constant $\gamma > 0$ such that $k \in [1, n^\gamma]$ and $\lambda_k \geq 1/n^\gamma$, then there exists some sufficiently small constant $\alpha \in (0, 1 - 10\gamma)$, if E satisfies (22.3), there exists an algorithm takes $O(n^{3-\alpha})$ time and uses $O(nk)$ space such that with probability $9/10$ the output $\widehat{\lambda}_i, \widehat{v}_i$ satisfies (22.4).*

Proof. For each $j \in [k]$, suppose we take $\widehat{b}^{(j)} = \sum_{i=1}^n \widehat{b}_i^{(j)}$ samples during the power iterations

for recovering $\widehat{\lambda}_j$ and $\widehat{v}_j$, the number of samples for slice i is

$$\widehat{b}_i^{(j)} \gtrsim b\|[A - \sum_{l=1}^{j-1} \widehat{\lambda}_l \widehat{v}_l^{\otimes 3}]_{i,*,*}\|_F^2 / \|A - \sum_{l=1}^{j-1} \widehat{\lambda}_l \widehat{v}_l^{\otimes 3}\|_F^2$$

where $b \gtrsim m \cdot n\|A\|_F^2/\epsilon^2$ and $m \geq k^2 LT$. Then the output guarantees of Theorem 22.3.4 hold for Algorithm 22.5 with constant probability. Our total time is $O(m\widehat{b})$ and the space is $O(nk)$, where $\widehat{b} = \max_{j\in[k]} \widehat{b}^{(j)}$. □

In Theorem 22.3.3, if we require $\widehat{b}_i = b\|A_{i,*,\cdots,*}\|_F^2/\|A\|_F^2$, we need to scan the entire tensor to compute $\|A_{i,*,\cdots,*}\|_F^2$, making our algorithm not sublinear. With the following mild assumption in Theorem 22.3.6, our algorithm is sublinear when sampling uniformly $(\widehat{b}_i = b/n)$ without computing $\|A_{i,*,\cdots,*}\|_F^2$:

Theorem 22.3.6 (Bounded slice norm). There exists a sufficiently small constant $\gamma > 0$ and a constant $\beta \in (0, 1]$, there exists a constant $\alpha > 0$(that depends on γ, β) such that for any order-p tensor $A = A^* + E \in \mathbb{R}^{n^p}$ with $\text{rank}(A^*) \leq n^\gamma$, $p \leq n^\gamma$, $\lambda_k \geq 1/n^\gamma$, $\|A_{i,*,\cdots,*}\|_F^2 \leq \frac{1}{n^\beta}\|A\|_F^2$ for all $i \in [n]$, and E satisfies (22.3), Algorithm 22.5 takes $O(n^{p-\alpha})$ time.

The condition $\alpha \in (0, 1]$ is a practical one. When $\alpha = 1$, all tensor slices have equal Frobenius norm. The case $\alpha = 0$ only occurs when $\|A_{i,*,\cdots,*}\|_F = \|A\|_F$; i.e., all except one slice is zero. This theorem can also be applied to asymmetric tensors, since the analysis in [WTSA15] can be extended to them.

For certain cases, we can remove the bounded slice norm assumption. The idea is to take a sublinear number of samples from the tensor to obtain upper bounds on all slice

norms. As outlined in Section 22.1, when p is even, because we do not have sign cancellations we can show:

Theorem 22.3.7 (Even order, informal of Theorem 22.9.14)**.** *There is a constant $\alpha > 0$ and a sufficiently small constant $\gamma > 0$, such that, for any even order-p tensor $A = A^* + E \in \mathbb{R}^{n^p}$ with $\text{rank}(A^*) \leq n^\gamma$, $p \leq n^\gamma$ and $\lambda_k \geq 1/n^\gamma$ and E satisfying (22.3), Algorithm 22.5 runs in $O(n^{p-\alpha})$ time.*

As outlined in Section 22.1, for $p = 3$ and small k we can take sign considerations into account:

Theorem 22.3.8 (Low rank)**.** *There is a constant $\alpha > 0$ such that for any symmetric tensor $A = A^* + E \in \mathbb{R}^{n^3}$ with E satisfying (22.3) and $\text{rank}(A^*) \leq 2$ and $\lambda_k \geq 1/n^\gamma$, Algorithm 22.5 runs in $O(n^{3-\alpha})$ time.*

22.4 Experiments

22.4.1 Experiment Setup and Datasets

Our implementation shares the same code base [1] as the sketching-based robust tensor power method proposed in [WTSA15]. We ran our experiments on an i7-5820k CPU with 64 GB of memory in single-threaded mode. We ran two versions of our algorithm: the version *with pre-scanning* scans the full tensor to accurately measure per-slice Frobenius norms and make samples for each slice in proportion to its Frobenius norm in ApproxTIvw; the version *without pre-scanning* assumes that the Frobenius norm of each slice is bounded by $\frac{1}{n^\alpha}\|A\|_F^2, \alpha \in (0, 1]$ and uses b/n samples per slice, where b is the total number of samples our algorithm makes, analogous to the sketch length b in [WTSA15].

Synthetic datasets. We first generated an orthonormal basis $\{v_i\}_{i=1}^k$ and then computed the synthetic tensor as $A^* = \sum_{i=1}^k \lambda_i v_i^{\otimes 3}$, with $\lambda_1 \geq \cdots \geq \lambda_k$. Then we normalized A^* such that $\|A^*\|_F = 1$, and added a symmetric Gaussian noise tensor E where $E_{ijl} \sim \mathcal{N}(0, \frac{\sigma}{n^{1.5}})$ for $i \leq j \leq l$. Then σ controls the noise-to-signal ratio and we kept it as 0.01 in all our synthetic tensors. For the eigenvalues λ_i, we generated three different decays: inverse decay $\lambda_i = \frac{1}{i}$, inverse square decay $\lambda_i = \frac{1}{i^2}$, and linear decay $\lambda_i = 1 - \frac{i-1}{k}$. We also set $k = 100$ when generating tensors, since higher rank eigenvalues were almost indistinguishable from the added noise. To show the scalability of our algorithm, we generated tensors with different dimensions: $n = 200, 400, 600, 800, 1000, 1200$.

[1] http://yining-wang.com/fftlda-code.zip

Real-life datasets. Latent Dirichlet Allocation [BNJ01] (LDA) is a powerful generative statistical model for topic modeling. A spectral method has been proposed to solve LDA models [AGH+14, AFH+12] and the most critical step in spectral LDA is to decompose a symmetric $K \times K \times K$ tensor with orthogonal eigenvectors, where K is the number of modeled topics. We followed the steps in [AGH+14, TWZS15] and built a $K \times K \times K$ tensor A_{LDA} for each dataset, and then ran our algorithms directly on A_{LDA} to see how it works on those tensors in real applications. In our experiments we keep $K = 200$. We used the two same datasets as the previous work [WTSA15]: Wiki and Enron, as well as four additional real-life datasets. We refer the reader to our GitHub repository [2] for our code and full results.

[2] https://github.com/huanzhang12/sampling_tensor_decomp/

22.4.2 Results

We considered running time and the squared residual norm to evaluate the performance of our algorithms. Given a tensor $A \in \mathbb{R}^{n^3}$, let $\|A - \sum_{i=1}^{k} \lambda_i u_i \otimes v_i \otimes w_i\|_F^2$ denote the squared residual norm where $\{(\lambda_1, u_1, v_1, w_1), \cdots, (\lambda_k, u_k, v_k, w_k)\}$ are the eigenvalue/eigenvectors obtained by the robust power method. To reduce the experiment time we looked only for the first eigenvalue and eigenvector, but our algorithm is capable of finding any number of eigenvalues/eigenvectors. We list the pre-scanning time as preprocessing time in tables. It only depends on the tensor dimension n and unlike the sketching based method, it does not depend on b. Pre-scanning time is very short, because it only requires one pass of sequential access to the tensor which is very efficient on hardware.

Sublinear time verification. Our theoretical result suggests the total number of samples $b_{\text{no-prescan}}$ for our algorithm without pre-scanning is $n^{1-\alpha}(\alpha \in (0, 1])$ times larger than b_{prescan} for our algorithm with pre-scanning. But in experiments we observe that when $b_{\text{no-prescan}} = b_{\text{prescan}}$ both algorithms achieve very similar accuracy, indicating that in practice $\alpha \approx 1$.

Synthetic datasets. We ran our algorithm on a large number of synthetic tensors with different dimensions and different eigengaps. Table 22.1 shows results for a tensor with 1200 dimensions with 100 non-zero eigenvalues decaying as $\lambda_i = \frac{1}{i^2}$. To reach roughly the same residual norm, the running time of our algorithm is over 50 times faster than that of the sketching-based robust tensor power method, thanks to the fact that we usually need a relatively small B and b to get a good residual, and the hidden constant factor in running time of sampling is much smaller than that of sketching.

Our algorithm scales well on large tensors due to its sub-linear nature. In Figure 22.1(a), for the sketching-based method we kept $b = 2^{16}$, $B = 30$ for $n \leq 800$ and $B = 50$ for $n > 800$ (larger n requires more sketches to observe a reasonable recovery). For our algorithm, we chose b and B such that for each n, our residual norm is on-par or better than the sketching-based method. Our algorithm needs much less time than the sketching-based one over all dimensions. Another advantage of our algorithm is that there are zero or very minimal preprocessing steps. In Figure 22.1(b), we can see how the preprocessing time grows to prepare sketches when the dimension increases. For applications where only the first few eigenvectors are needed, the preprocessing time could be a large overhead.

Real-life datasets. Due to the small tensor dimension (200), our algorithm shows less speedup than the sketching-based method. But it is still $2 \sim 6$ times faster in each of the six real-life datasets, achieving the same squared residual norm. Table 22.2 reports results for one of the datasets in many different settings of (b, B). Like in synthetic datasets, we also empirically observe that the constant b in importance sampling is much smaller than the b used in sketching to get the same error guarantee.

Sketching based robust power method: $n = 1200$, $\lambda_i = \frac{1}{i^2}$										
	Squared residual norm				**Running time (s)**				**Preprocessing time (s)**	
b \ B	10	30	50		10	30	50		10	30	50
2^{10}	1.010	1.014	0.5437		0.6114	2.423	4.374		5.361	15.85	26.08
2^{12}	1.020	0.2271	0.1549		1.344	4.563	8.022		5.978	17.23	28.31
2^{14}	0.1513	0.1097	0.1003		4.928	15.51	27.87		8.788	24.72	40.4
2^{16}	0.1065	0.09242	**0.08936**		22.28	69.7	**116.3**		13.76	34.74	**55.34**
Importance sampling based robust power method (without prescanning): $n = 1200$, $\lambda_i = \frac{1}{i^2}$										
	Squared residual norm				**Running time (s)**				**Preprocessing time (s)**	
b \ B	10	30	50		10	30	50		10	30	50
$5n$	**0.08684**	0.08637	0.08639		**2.595**	8.3	15.46		**0.0**	0.0	0.0
$10n$	0.08784	0.08671	0.08627		4.42	13.68	25.84		0.0	0.0	0.0
$20n$	0.08704	0.08700	0.08618		8.02	24.51	46.37		0.0	0.0	0.0
$30n$	0.08697	0.08645	0.08625		11.63	35.35	66.71		0.0	0.0	0.0
$40n$	0.08653	0.08664	0.08611		15.19	46.12	87.24		0.0	0.0	0.0
Importance sampling based robust power method (with prescanning): $n = 1200$, $\lambda_i = \frac{1}{i^2}$										
	Squared residual norm				**Running time (s)**				**Preprocessing time (s)**	
b \ B	10	30	50		10	30	50		10	30	50
$5n$	**0.08657**	0.08684	0.08636		**3.1**	10.47	18		**2.234**	2.236	2.234
$10n$	0.08741	0.08677	0.08668		5.427	17.43	30.26		2.232	2.233	2.233
$20n$	0.08648	0.08624	0.08634		9.843	31.42	54.49		2.226	2.226	2.226
$30n$	0.08635	0.08634	0.08615		14.33	45.4	63.85		2.226	2.224	2.227
$40n$	0.08622	0.08652	0.08619		18.68	59.32	82.83		2.225	2.225	2.225

Table 22.1: Synthetic tensor decomposition using the robust tensor power method. We use an order-3 normalized dense tensor with dimension $n = 1200$ with $\sigma = 0.01$ noise added. We run sketching-based and sampling-based methods to find the first eigenvalue and eigenvector by setting $L = 50$, $T = 30$ and varying B and b.

22.5 Generate importance sampling indices

In the section we present the full algorithm to generate importance sampling indices.

We view this problem as follows: imagine $[0, 1]$ is divided into n "bins" with different lengths corresponding to the probability of selecting each bin. We generate m random numbers uniformly from $[0, 1]$ and see which bin each random number belongs to. If a random number is in bin i, we sample the i-th index of a vector. There are two straightforward ways to generate such indices, but neither of them gives us linear running time in m and n. The first method is: (1) generate a list of random numbers (2) for each random number, choose

Sketching based robust power method: dataset **wiki**, $\|\mathbf{T}\|_F^2 = 2.135\text{e+}07$							
	Squared residual norm			**Running time (s)**		**Preprocessing time (s)**	
b \ B	10	30		10	30	10	30
2^{10}	2.091e+07	1.951e+07		0.2346	0.8749	0.1727	0.2535
2^{11}	1.971e+07	1.938e+07		0.4354	1.439	0.2408	0.3167
2^{12}	1.947e+07	1.930e+07		1.035	2.912	0.4226	0.4275
2^{13}	**1.931e+07**	1.927e+07		**2.04**	5.94	**0.5783**	0.6493
2^{14}	1.928e+07	1.926e+07		4.577	13.93	1.045	1.121
Importance sampling based robust power method (without prescanning): dataset **wiki**, $\|\mathbf{T}\|_F^2 = 2.135\text{e+}07$							
	Squared residual norm			**Running time (s)**		**Preprocessing time (s)**	
b \ B	10	30		10	30	10	30
$5n$	**1.931e+07**	1.928e+07		**0.3698**	1.146	**0.0**	0.0
$10n$	1.931e+07	1.929e+07		0.5623	1.623	0.0	0.0
$20n$	1.935e+07	1.926e+07		0.9767	2.729	0.0	0.0
$30n$	1.929e+07	1.926e+07		1.286	3.699	0.0	0.0
$40n$	1.928e+07	1.925e+07		1.692	4.552	0.0	0.0
Importance sampling based robust power method (with prescanning): dataset **wiki**, $\|\mathbf{T}\|_F^2 = 2.135\text{e+}07$							
	Squared residual norm			**Running time (s)**		**Preprocessing time (s)**	
b \ B	10	30		10	30	10	30
$5n$	**1.931e+07**	1.930e+07		**0.4376**	1.168	**0.01038**	0.01103
$10n$	1.928e+07	1.930e+07		0.6357	1.8	0.0104	0.01044
$20n$	1.931e+07	1.927e+07		1.083	2.962	0.01102	0.01042
$30n$	1.929e+07	1.925e+07		1.457	4.049	0.01102	0.01043
$40n$	1.929e+07	1.925e+07		1.905	5.246	0.01105	0.01105

Table 22.2: Tensor decomposition in LDA on the wiki dataset. The tensor is generated by spectral LDA with dimension $200 \times 200 \times 200$. It is symmetric but not normalized. We fix $L = 50$, $T = 30$ and vary B and b.

the correct bin for it. Step (1) takes $O(m)$ time, but step (2) takes $O(mn)$ time in the worst case. The second method is: (1) generate a list of random numbers (2) sort these random numbers (3) sequentially for each random number, choose the correct bin for it. Step (1) takes $O(m)$ time, step (2) takes $O(m \log m)$ time, and step (3) takes $O(m + n)$ time as it merges two sorted lists.

In Algorithm 22.1, we instead present an $O(m+n)$ time algorithm which we call GenRandTuples for generating such indices by combining existing algorithms. We use Bentley and Saxe's algorithm GenSortedRandN(m) [BS80b] to efficiently generate m sorted random numbers(a) in $O(m)$ time. Then RandNToBins outputs a list of m indices(e) such

that for each random number in a, there is an index in $\widetilde{q}$. The running time of this step is $O(m+n)$. We use Knuth's shuffling algorithm GenRandPerm(m) [Knu69] to generate a random permutation of $[m]$(c) in $O(m)$ time. Finally, GenRandTuples generates m tuples with two (resp. three) entries given n probabilities in array $\widetilde{q}$ for computing $A(I, v, w)$ (resp. $A(u, v, w)$).

Claim 22.5.1. *Function* GenSortedRandN(m) *uses Bentley and Saxe's algorithm [BS80b] and takes* $O(m)$ *time to generate a list of random numbers between* 0 *and* 1 *in increasing order.*

Proof. For the correctness of this claim, we refer the reader to [BS80b]. □

Claim 22.5.2. *Function* RandNToBins(A, m, Q, n) *takes* $O(m+n)$ *time.*

Proof. The input of RandNToBins is two arrays, A and Q. The array A is a list of sorted random numbers in $[0, 1]$. The array Q can be thought as a list of boundary values for bins, e.g., if some entry x satisfies $Q[j] < x \leq Q[j+1]$, then we should assign x to bin j. This procedure is equivalent to merging two sorted lists. Thus, it takes linear time in the summation of two lengths. □

Claim 22.5.3. *Function* GenRandPerm(m) *uses Knuth's shuffle algorithm [Knu69] to generate a random permutation of* $[m]$ *in* $O(m)$ *time.*

Proof. For the correctness of this claim, we refer the reader to [Knu69] and [Wik16]. □

Algorithm 22.1 Generate importance sampling indices

function GenRandTuples($m, \widetilde{q}$)
$a[1...m] \leftarrow$ GenSortedRandN(m)
$e[1...m] \leftarrow$ RandNToBins($a, m, \widetilde{q}, n$)
$c[1...m] \leftarrow$ GenRandPerm(m)
$\mathcal{L} \leftarrow \emptyset$
for $i = 1 \rightarrow m$ **do**
    $\mathcal{L} \leftarrow \mathcal{L} \cup e[c[i]]$
end for
return $\mathcal{L}$
end function
function RandNToBins(A, m, Q, n)
$i \leftarrow 1, j \leftarrow 1$
while $i \leq m$ **do**
    if $A[i] < Q[j]$ **then**
        $C[i] \leftarrow j$, $i \leftarrow i + 1$
    else
        $j \leftarrow j + 1$
    end if
end while
return $C[1...m]$
end function
function GenSortedRandN(m) [BS80b]
$s \leftarrow 0$
for $i = 1 \rightarrow L$ **do**
    $s \leftarrow s - \log(\text{uniform}(0, 1))$
    $A[i] \leftarrow s$
end for
$s \leftarrow s - \log(\text{uniform}(0, 1))$
for $i = 1 \rightarrow L$ **do**
    $A[i] \leftarrow A[i]/s$
end for
return $A[1...m]$
end function
function GenRandPerm(m) [Knu69]
for $i = 1 \rightarrow m$ **do**
    $C[i] \leftarrow i$
end for
for $i = 1 \rightarrow m$ **do**
    $t \leftarrow \text{uniform}(i)$, $C[i] \leftarrow C[t]$, $C[t] \leftarrow i$
end for
return $C[1...m]$
end function

22.6 Importance sampling lemmas

In this section we state some useful facts and give proofs for some lemmas that are not proved in the main text.

We shall make use of the following facts which follow from the definitions of a tensor. The following facts hold immediately by the definition of a tensor's ℓ_2-norm and Frobenius norm.

Fact 22.6.1. *For any $p \geq 3$, for any tensor $A \in \mathbb{R}^{n^p}$ and p unit vectors $u_1, u_2, \cdots, u_p \in \mathbb{R}^n$,*

1. $\|u_1 \otimes u_2 \otimes \cdots \otimes u_p\|_F^2 = \prod_{j=1}^p \|u_j\|_2^2$.
2. $\langle A, u_1 \otimes \cdots \otimes u_p \rangle = \sum_{i_1=1}^n \cdots \sum_{i_p=1}^n A_{i_1,\cdots,i_p} \prod_{j=1}^p u_{j,i_j}$.
3. $\langle A_{i,*,\cdots,*}, u_2 \otimes \cdots \otimes u_p \rangle = \sum_{i_2=1}^n \cdots \sum_{i_p=1}^n A_{i,i_2,\cdots,i_p} \prod_{j=2}^p u_{j,i_j}$.

We provide the proofs for Lemma 22.3.1 and 22.3.2.

Lemma 22.3.1. Suppose random variable $X = A_{i_1,\cdots,i_p} \prod_{j=1}^p u_{j,i_j} / \prod_{j=1}^p q_{j,i_j}$ with probability $\prod_{j=1}^p q_{j,i_j}$ where $q_{j,i_j} = |u_{j,i_j}|^2 / \|u_j\|_2^2, \forall j \in [n], \forall i_j \in [n]$, and we take L i.i.d. samples X, denote $X_1, X_2, \cdots, X_L$. Let $Y = \frac{1}{L} \sum_{\ell=1}^L X_\ell$. Then 1. $\mathbb{E}[Y] = \langle A, u_1 \otimes u_2 \otimes \cdots \otimes u_p \rangle$, and 2. $\mathbb{V}[Y] \leq \frac{1}{L} \|A\|_F^2 \cdot \|u_1 \otimes u_2 \otimes \cdots \otimes u_p\|_F^2$.

Proof. Consider the expectation of Y,

$$\mathbb{E}[Y] = \mathbb{E}[\frac{1}{L} \sum_{\ell=1}^L X_\ell] = \mathbb{E}[X_\ell] = \sum_{i_1=1}^n \cdots \sum_{i_p=1}^n \left(A_{i_1,\cdots,i_p} \prod_{j=1}^p \frac{1}{q_{j,i_j}} u_{j,i_j} \right) \cdot \left(\prod_{j=1}^p q_{j,i_j} \right),$$

Furthermore, we can obtain,

$$\mathbb{E}[Y] = \sum_{i_1=1}^{n} \cdots \sum_{i_p=1}^{n} A_{i_1,\cdots,i_p} \prod_{j=1}^{p} u_{j,i_j} = \langle A, u_1 \otimes \cdots \otimes u_p \rangle.$$

It remains to upper bound the variance of Y,

$$\mathbb{V}[Y] = \mathbb{E}[Y^2] - (\mathbb{E}[Y])^2 = \mathbb{E}[\frac{1}{L^2}(\sum_{\ell=1}^{L} X_\ell)^2] - (\mathbb{E}[\frac{1}{L}\sum_{\ell=1}^{L} X_\ell])^2 = \frac{1}{L^2}(\mathbb{E}[(\sum_{\ell=1}^{L} X_\ell)^2] - L^2(\mathbb{E}[X_\ell])^2).$$

Furthermore, we can obtain,

$$\begin{aligned}
\mathbb{V}[Y] &= \frac{1}{L^2}\left(L\mathbb{E}[X_\ell^2] + L(L-1)\mathbb{E}[X_\ell]^2 - L^2\mathbb{E}[X_\ell]^2\right) \\
&\leq \frac{1}{L}\mathbb{E}[X_\ell^2] \\
&= \frac{1}{L}\left(\sum_{i_1=1}^{n} \cdots \sum_{i_p=1}^{n} A_{i_1,\cdots,i_p}^2 \left(\prod_{j=1}^{n} \frac{1}{q_{j,i_j}} u_{j,i_j}\right)^2 \left(\prod_{j=1}^{p} q_{j,i_j}\right)\right) \\
&= \frac{1}{L}\sum_{i_1=1}^{n} \cdots \sum_{i_p=1}^{n} A_{i_1,\cdots,i_p}^2 \prod_{j=1}^{p} \|u_j\|_2^2 \\
&\leq \frac{1}{L}\|A\|_F^2 \cdot \|u_1 \otimes \cdots \otimes u_p\|_F^2.
\end{aligned}$$

Thus, we complete the proof. □

Lemma 22.3.2. For all $i_1 \in [n]$, suppose random variable $X^{i_1} = A_{i_1,i_2,\cdots,i_p} \prod_{j=2}^{p} u_{j,i_j} / \prod_{j=2}^{p} q_{j,i_j}$ with probability $\prod_{j=2}^{p} q_{j,i_j}$ where $q_{j,i_j} = |u_{j,i_j}|^2/\|u_j\|_2^2$, $\forall j \in \{2,3,\cdots,n\}$, $\forall i_j \in [n]$ and we take L_{i_1} i.i.d. samples of X^{i_1}, say $X_1^{i_1}, X_2^{i_1}, \cdots, X_L^{i_1}$. Let $Y^{i_1} = \frac{1}{L_{i_1}} \sum_{\ell=1}^{L} X_\ell^i$. Then 1. $\mathbb{E}[Y^i] = \langle A_{i_1,*,\cdots,*}, u_2 \otimes \cdots \otimes u_p \rangle$, and 2. $\mathbb{V}[Y^{i_1}] \leq \frac{1}{L_{i_1}} \|A_{i,*,\cdots,*}\|_F^2 \|u_2 \otimes \cdots \otimes u_p\|_F^2$.

Proof. Consider the expectation of Y^{i_1},

$$\mathbb{E}[Y^{i_1}] = \mathbb{E}[\frac{1}{L}\sum_{\ell=1}^{L} X_\ell^{i_1}] = \mathbb{E}[X^{i_1}] = \sum_{i_2=1}^{n} \cdots \sum_{i_p=1}^{n} A_{i,j,k} \left(\prod_{j=2}^{p} \frac{1}{q_{j,i_j}} u_{j,i_j}\right) \left(\prod_{j=2}^{p} q_{j,i_j}\right),$$

Furthermore, we can obtain,

$$\mathbb{E}[Y^{i_1}] = \sum_{i_2=1}^{n} \cdots \sum_{i_p=1}^{n} A_{i_1,i_2,\cdots,i_p} \prod_{j=2}^{p} u_{j,i_j} = \langle A_{i_1,*,\cdots,*}, u_2 \otimes \cdots \otimes u_p \rangle.$$

Consider the variance of Y^{i_1},

$$\mathbb{V}[Y^{i_1}] = \mathbb{E}[(Y^{i_1})^2] - (\mathbb{E}[Y^{i_1}])^2 = \frac{1}{L^2}(L\mathbb{E}[(X^{i_1})^2] + L(L-1)(\mathbb{E}[X^{i_1}])^2) - (\mathbb{E}[X^{i_1}])^2.$$

Furthermore, we can obtain,

$$\begin{aligned}
\mathbb{V}[Y^{i_1}] &= \frac{1}{L}(\mathbb{E}[(X^{i_1})^2] - (\mathbb{E}[X^{i_1}])^2) \\
&\leq \frac{1}{L}\mathbb{E}[(X^{i_1})^2] \\
&= \frac{1}{L} \sum_{i_2=1}^{n} \cdots \sum_{i_p=1}^{n} A_{i_1,i_2,\cdots,i_p}^2 \left(\prod_{j=2}^{p} \frac{1}{q_{j,i_j}} u_{j,i_j} \right)^2 \left(\prod_{j=2}^{p} q_{j,i_j} \right) \\
&= \frac{1}{L} \sum_{i_2=1}^{n} \cdots \sum_{i_p=1}^{n} A_{i_1,i_2,\cdots,i_p}^2 \prod_{j=2}^{n} \|u_j\|_2^2 \\
&= \frac{1}{L} \|A_{i_1,*,\cdots,*}\|_F^2 \|u_2 \otimes \cdots \otimes u_p\|_F^2.
\end{aligned}$$

Thus, we complete the proof. □

22.7 Approximate Tensor Contractions

We need a slightly different version of Theorem 1 in [WTSA15]. Here we denote $\widehat{b}_i$ as the samples we take for tensor slice i.

Theorem 22.3.3. For any $p \geq 3$, given tensor $A \in \mathbb{R}^{n^p}$, for any unit vector $u, v \in \mathbb{R}^n$, there is an algorithm that takes $\widehat{b}_i$ samples from i-th slice and it is able to output a value $\overline{A}(u, \cdots, u) \in \mathbb{R}$ and a vector $\overline{A}(I, u, \cdots, u) \in \mathbb{R}^n$ such that for any $b > 0$ if $\widehat{b}_i \geq b\|A_{i,*,\cdots,*}\|_F^2/\|A\|_F^2$ then the following bounds hold:

$$\mathbb{E}[\|\overline{E}(I, u, \cdots, u)\|_2^2] \leq n\|A\|_F^2/b, \text{ and } \mathbb{E}[|\overline{E}(u, \cdots, u)|^2] \leq \|A\|_F^2/b. \tag{22.1}$$

and

$$\mathbb{E}[|v^\top \overline{E}(I, u, \cdots, u)|^2] \leq \|A\|_F^2/b. \tag{22.2}$$

where $\overline{E} := \overline{A} - A$.

Proof. **Part 1. of Equation 22.1**

By combining Lemma (22.3.2) and $\widehat{b}_i \geq b\|A_{i,*,\cdots,*}\|_F^2/\|A\|_F^2$, we know that

$$\mathbb{E}\left[|\overline{E}(I, u, \cdots, u)_i|^2\right] = \frac{\|A_{i,*,*}\|^2}{b\|A_{i,*,*}\|_F^2/\|A\|_F^2} = \|A\|_F^2/b.$$

Part 2 of Equation (22.1) It follows by Lemma 22.3.1.

Equation (22.2) It follows by

$$\begin{aligned}\mathbb{E}[|v^\top \overline{E}|^2] &= \mathbb{E}\left[\sum_{i=1}^{n} v_i^2 \overline{E}(I, u, \cdots, u)_i^2\right] \\ &\leq \sum_{i=1}^{n} v_i^2 \|A_{i,*,\cdots,*}\|_F^2 / \widehat{b}_i \\ &\leq \sum_{i=1}^{n} v_i^2 \|A\|_F^2 / b = \|A\|_F^2 / b.\end{aligned}$$

Thus, we complete the proof. □

Without loss of generality, we can assume that $\|A\|_F^2 = 1$. By choosing sufficiently large b, we can guarantee the error bounds of sketch noise $\overline{E}$ to be small.

Corollary 22.7.1. *For any $m \geq 3$, if $b \geq mnc_0^2/\epsilon^2$, for any unit vector $u, v \in \mathbb{R}^n$, then with probability at least $1 - 3/m$ such that,*

1. $\|\overline{E}(I, u, \cdots, u)\|_2 \leq \epsilon/c_0$.
2. $|\overline{E}(u, \cdots, u)| \leq \epsilon/(c_0\sqrt{n})$.
3. $|v^\top \overline{E}(I, u, \cdots, u)| \leq \epsilon/(c_0\sqrt{n})$.

Proof. **Part 1.** Using Markov's inequality $\Pr[x \geq a] \leq \mathbb{E}[x]/a$, we have

$$\Pr[\|\overline{E}(I, u, \cdots, u)\|_2^2 \geq (\epsilon/c_0)^2] \leq \frac{\mathbb{E}[\|\overline{E}(I, u, \cdots, u)\|_2^2]}{(\epsilon/c_0)^2} \leq \frac{n\|A\|_F^2/b}{(\epsilon/c_0)^2} \leq \frac{1}{m},$$

where the last step follows by choosing $b \geq mnc_0^2/\epsilon^2$.

Part 2. Similarly, by Markov's inequality, we have

$$\Pr[|\overline{E}(u, u, \cdots, u)|^2 \geq (\epsilon/(c_0\sqrt{n}))^2] \leq \frac{\mathbb{E}[\|\overline{E}(u, u, \cdots, u)\|_2^2]}{(\epsilon/(c_0\sqrt{n}))^2} \leq \frac{\|A\|_F^2/b}{(\epsilon/(c_0\sqrt{n}))^2} \leq \frac{1}{m},$$

where the last step also follows by choosing $b \geq mnc_0^2/\epsilon^2$.

Part 3. This part also can be proved by Markov's inequality and it requires $b \geq mnc_0^2/\epsilon^2$. □

If we compare our Theorem 22.3.3 to the bounds obtained by sketching in [WTSA15], we see that our expectation bounds for $\overline{E}(I, u, \cdots, u)$ and $\overline{E}(u, \cdots, u)$ are the same. As long as the conditions for Theorem 22.3.3 holds, all analysis in [WTSA15, WA16] will follow. To guarantee the noise introduced by sampling satisfies the bounds in Theorem 22.3.4, we require $b \geq n\|A\|_F^2/\epsilon^2 = O(n^2k)$.

However, as a condition of Theorem 22.3.3, we require that $\widehat{b_i} \gtrsim b\|A_{i,*,*}\|_F^2/\|A\|_F^2$. Because $\|A_{i,*,*}\|_F^2$ can be as large as $\|A\|_F^2$, if we made no assumptions, then $\widehat{b_i} \gtrsim b$; a summation over all n slices is $nb = O(n^3k)$, making our algorithm not sublinear time.

We will discuss how to deal with the term $\|A_{i,*,*}\|_F^2$ and make our algorithm run in sublinear time in the following paragraphs.

22.7.1 Case 1: sampling with known per-slice Frobenius norm

If we do not uniformly take $\widehat{b}_i \gtrsim b$ samples per slice, but sample based the Frobenius norm of each slice, in other words, the number of samples for slice i is exactly

$$\widehat{b}_i = b\frac{\|A_{i,*,*}\|_F^2}{\|A\|_F^2}$$

Then we will need only $\sum_{i=1}^n \widehat{b}_i = b = O(n^2)$ samples. However, to compute $\|A_{i,*,*}\|_F^2$, we have to *pre-scan* the tensor in $O(n^3)$ time.

Sometimes during the data collection process we can store the tensor's per-slice Frobenius norms as meta data and use them for free. But if we don't know the per-slice Frobenius norms, we have to compute them. This is not ideal because we need to scan the entire tensor before we start power iterations to compute per-slice Frobenius norms. Although in practice computing per-slice Frobenius norms only needs sequential access and can be quite efficient in memory or on storage device, it makes our algorithm depend on the full tensor and run in linear time. However, after making a weak assumption on the per-slice Frobenius norm which can be easily satisfied in practice, we can still obtain sublinear run time, as shown in Case 2.

22.7.2 Case 2: sampling without known per-slice Frobenius norm

Theorem 22.3.6 (Bounded slice norm). There exists a sufficiently small constant $\gamma > 0$ and a constant $\beta \in (0, 1]$, there exists a constant $\alpha > 0$(that depends on γ, β) such that for any order-p tensor $A = A^* + E \in \mathbb{R}^{n^p}$ with $\text{rank}(A^*) \leq n^\gamma$, $p \leq n^\gamma$, $\lambda_k \geq 1/n^\gamma$, $\|A_{i,*,\cdots,*}\|_F^2 \leq \frac{1}{n^\beta}\|A\|_F^2$ for all $i \in [n]$, and E satisfies (22.3), Algorithm 22.5 takes $O(n^{p-\alpha})$ time.

Proof. Because we assume $\|A_{i,*,*}\|_F^2 \leq \frac{1}{n^\alpha}\|A\|_F^2$, we can set $\widehat{b}_i = \frac{b}{n^\alpha} \gtrsim b\|A_{i,*,*}\|_F^2/\|A\|_F^2$ for all i such that the conditions for Theorem 22.3.3 hold. Thus, the number of samples we need is $\sum_{i=1}^n \widehat{b}_i = bn^{1-\alpha} = O(n^{3-\alpha})$.

□

We should note that the condition $0 < \alpha \leq 1$ is a very reasonable assumption. When $\alpha = 1$, all tensor slices have the equal Frobenius norm. The case where $\alpha = 0$ only occurs when $\|A_{i,*,*}\|_F = \|A\|_F$; in other words, all except one slices are zero. Except for this extreme case, our algorithm only needs to take $o(n^3)$ samples from the tensor.

Also, note that this proof does not make any assumption for the symmetry of the tensor. Thus, theorem 22.3.6 also applies to asymmetric tensors, as long as the analysis in [WTSA15] is extended to asymmetric case.

22.7.3 Case 3: improve the bounds in case 2

We are still interested in improve the $O(n^{3-\alpha})$ running time bound of our algorithm. We want to go one step further and remove the assumption on per-slice Frobenius norm. The basic idea is to take a sublinear number of samples from the tensor to estimate per-slice Frobenius norm. With some properties of the tensor (symmetry, low-rank, orthogonal, etc), achieving a sub-linear runtime bound is possible, as we will discuss below.

It is worth noting that in our experiments we have seen that sampling with or without known Frobenius norm (with or without pre-scanning) does not make a big difference for most datasets. Because sampling with known per-slice Frobenius norm is the best case ($\sum_{i=1}^{n} \widehat{b}_i = O(n^2)$), the algorithm described below with an improved bound on $\sum_{i=1}^{n} \widehat{b}_i = \omega(n^2)$ is unlikely to outperform. Thus, we did not conduct experiments on this part, and focus on theoretical discussions here.

22.8 Estimating Slice Norms in Sublinear Time

We present the following facts before introducing our lemmas and theorems:

22.8.1 Definitions and Facts

Fact 22.8.1. *For any symmetric matrix $A \in \mathbb{R}^{n \times n}$, $\|A\|_2 = \max_{\|u\|_2=1} |u^\top A u|$.*

Proof. Because A is symmetric, we choose S such that $S^\top S = A$,

$$\|A\|_2 = \|S\|_2^2 = \max_{\|u\|_2=1} \|Su\|_2^2 = \max_{\|u\|_2=1} |u^\top S^\top S u| = \max_{\|u\|_2=1} |u^\top A u|.$$

□

The following technical lemmas give necessary bounds on norms that we will use in proofs.

Lemma 22.8.2. *For any $E \in \mathbb{R}^{n^3}$, if $\|E\|_2 := \max_{\|u\|_2=\|v\|_2=\|w\|_2=1} |E(u,v,w)| \leq 1/n$, then for all $i \in [n]$,*

1. $\max_{\|u\|_2=1} \|E(I,u,u)\|_2 \leq \|E\|_2$,
2. $\|E_{i,*,*}\|_2 \leq 1/n$,
3. $\|E_{i,*,*}\|_F \leq 1/\sqrt{n}$,
4. $\forall j \in [n], |E_{i,j,j}| \leq 1/n$.

Proof. **Part 1.**

$$\begin{aligned}
\|E\|_2^2 = & \max_{\|u\|_2=\|v\|_2=\|w\|_2=1} |E(u,v,w)|^2 \\
= & \max_{\|u\|_2=\|v\|_2=\|w\|_2=1} \left(\sum_{i=1}^n u_i \cdot \sum_{j=1}^n \sum_{k=1}^n E_{i,j,k} v_j w_k \right)^2 \\
\geq & \max_{\|v\|_2=\|w\|_2=1} \sum_{i=1}^n u_i^2 \cdot \sum_{i=1}^n \left(\sum_{j=1}^n \sum_{k=1}^n E_{i,j,k} v_j w_k \right)^2 \\
= & \max_{\|v\|_2=\|w\|_2=1} \sum_{i=1}^n \left(\sum_{j=1}^n \sum_{k=1}^n E_{i,j,k} v_j w_k \right)^2 \\
\geq & \max_{\|v\|_2} \sum_{i=1}^n \left(\sum_{j=1}^n \sum_{k=1}^n E_{i,j,k} v_j v_k \right)^2 \\
= & \max_{\|v\|_2} \|E(I,v,v)\|_2^2
\end{aligned}$$

where the first inequality follows by setting $u_i = c \cdot \sum_{j=1}^n \sum_{k=1}^n E_{i,j,k} v_j v_k$, $\forall i \in [n]$ and $\sum_{i=1}^n u_i^2 = 1$, c is a normalizing constant.

Part 2. By definition of $\|\cdot\|_2$ for a tensor $E \in \mathbb{R}^{n^3}$, we have

$$\max_{\|u\|_2=1} \|E(I,u,u)\|_2 \leq \max_{\|u\|_2=\|v\|_2=\|w\|_2=1} |E(u,v,w)|$$

By definition of $E(I,u,u) \in \mathbb{R}^n$, we have

$$E(I,u,u)_i = \sum_{j=1}^n \sum_{k=1}^n E_{i,j,k} u_j u_k = u^\top E_{i,*,*} u$$

Thus, we can upper bound the $\|\cdot\|_2$ for matrix $E_{i,*,*} \in \mathbb{R}^{n^2}$ in the following sense,

$$\begin{aligned}
1/n &\geq \|E\|_2 \\
&\geq \max_{\|u\|_2=1} \|E(I,u,u)\|_2 \\
&= \max_{\|u\|_2=1} (\sum_{i=1}^{n} (u^\top E_{i,*,*} u)^2)^{1/2} \\
&\geq \max_{\|u\|_2=1} |u^\top E_{i,*,*} u| && \text{for any } i \in [n] \\
&= \|E_{i,*,*}\|_2, && \text{by Fact 22.8.1}
\end{aligned}$$

which completes the proof of Part 2.

Part 3. Since $\|E_{i,*,*}\|_2 \leq 1/n$ and the rank of $n \times n$ matrix is at most n, we have

$$\|E_{i,*,*}\|_F^2 = \mathbf{Tr}(E_{i,*,*}^\top E_{i,*,*}) = \sum_{j=1}^{n} \lambda_j \leq n \cdot \|E_{i,*,*}\|_2^2 \leq n \cdot (1/n)^2 \leq 1/n,$$

where λ_j are the eigenvalues of $E_{i,*,*}^\top E_{i,*,*}$ and the largest one is $\|E_{i,*,*}\|_2$, by the definition of $\|\cdot\|_2$ norm. The inequality implies $\|E_{i,*,*}\|_F \leq 1/\sqrt{n}$ and finishes the proof of Part 3.

Part 4. For each slice $i \in [n]$, by $\|E_{i,*,*}\|_2 \leq 1/n$ and definition of $\|\cdot\|_2$ of a matrix, we have $\max_{\|u\|_2=1} |u^\top E_{i,*,*} u| \leq 1/n$, choosing $u = e_j$ where e_j is the vector that jth position is 1 and all the other positions are 0. Thus, we know for any $j \in [n]$, $|E_{i,j,j}| \leq 1/n$. Thus, we complete the proof of Part 4.

□

Fact 22.8.3. *Given $p \geq 3$ and two vectors $u, v \in \mathbb{R}^n$ with $\|u\|_2 = \|v\|_2 = 1$, then*

1. $\|u^{\otimes p} - v^{\otimes p}\|_F \leq \sqrt{p}\|u - v\|_2$.
2. $|u_i - \widehat{u}_i| \leq \|u - v\|_2, \forall i \in [n]$.

3. $\|u_i \cdot u^{\otimes p-1} - \widehat{u}_i \cdot \widehat{u}^{\otimes p-1}\|_F \leq \sqrt{2p}\|u-v\|_2^2, \forall i \in [n]$.

Proof. We have $\|u\|_2 = \|v\|_2 = 1$, and $\|u-v\|_2^2 = \sum_{i=1}^n u_i^2 + v_i^2 - 2u_i v_i = \|u\|_2^2 + \|v\|_2^2 - 2\langle u, v\rangle$, putting it all together, we can get $\langle u, v\rangle = 1 - \frac{1}{2}\|u-v\|_2^2$. For any $p \geq 3$, we consider $\|u^{\otimes p} - v^{\otimes p}\|_F^2$,

$$\begin{aligned}
\|u^{\otimes p} - v^{\otimes p}\|_F^2 = & \sum_{i_1=1}^n \sum_{i_2=1}^n \cdots \sum_{i_p=1}^n (u_{i_1} u_{i_2} \cdots u_{i_p} - v_{i_1} v_{i_2} \cdots v_{i_p})^2 \\
= & \sum_{i_1=1}^n \sum_{i_2=1}^n \cdots \sum_{i_p=1}^n u_{i_1}^2 u_{i_2}^2 \cdots u_{i_p}^2 - 2u_{i_1} u_{i_2} \cdots u_{i_p} v_{i_1} v_{i_2} \cdots v_{i_p} + v_{i_1}^2 v_{i_2}^2 \cdots v_{i_p}^2 \\
= & (\|u\|_2^2)^p - 2(\langle u, v\rangle)^p + (\|v\|_2^2)^p \\
= & 2 - 2(\langle u, v\rangle)^p \\
= & 2 - 2\left(1 - \frac{1}{2}\|u-v\|_2^2\right)^p \\
\leq & p\|u-v\|_2^2
\end{aligned}$$

We can upper bound $\|u_i \cdot u^{\otimes p-1} - \widehat{u}_i \cdot \widehat{u}^{\otimes p-1}\|_F$,

$$\begin{aligned}
& \|u_i \cdot u^{\otimes p-1} - \widehat{u}_i \cdot \widehat{u}^{\otimes p-1}\|_F \\
\leq & \|u_i \cdot u^{\otimes p-1} - u_i \cdot \widehat{u}^{\otimes p-1}\|_F + \|u_i \cdot \widehat{u}^{\otimes p-1} - \widehat{u}_i \cdot \widehat{u}^{\otimes p-1}\|_F \\
= & |u_i| \cdot \|u^{\otimes p-1} - \widehat{u}^{\otimes p-1}\|_F + |u_i - \widehat{u}_i| \cdot \|\widehat{u}^{\otimes p-1}\|_F \\
\leq & 1 \cdot \sqrt{p-1}\|u-v\|_2 + \|u-v\|_2 \cdot 1 \\
\leq & \sqrt{2p}\|u-v\|_2.
\end{aligned}$$

□

Definition 22.8.1. For any $p \geq 3$, for any tensor $A \in \mathbb{R}^{n^p}$, we say vector $\tau \in \mathbb{R}^n$ is a "good"

estimate of A if for any $i \in [n]$,

$$\|A_{i,*,\cdots,*}\|_F^2 \leq \tau_i. \tag{22.5}$$

We say vector $\tau \in \mathbb{R}^n$ is an "efficient" estimate of A, if there exists some constant $\alpha > 0$ such that,

$$\sum_{i=1}^{n} \min(n^{p-1}\tau_i/\|A\|_F^2, n^{p-1}) = O(n^{p-\alpha}). \tag{22.6}$$

Remark 22.8.1. For $p = 3$, Equation (22.6) implies $\sum_{i=1}^{n} \widehat{b}_i = o(n^3)$ as long as we choose $\widehat{b}_i = \min(n^2\tau_i/\|A\|_F^2, n^2)$. Note that in the most part of this section, we focus on order $p = 3$ tensor.

22.8.2 Order $p = 3$, rank $k = 1$ tensor

We start by considering the simplest case, where the tensor A^* is assumed to be rank-1, $A^* = u^{\otimes 3}$. We can observe the true tensor plus some noise, i.e., $A = A^* + E$. As long as the noise E is sufficiently small, we can get a good bound on the tensor's per-slice Frobenius norm by just looking at its diagonals! The intuition is that Frobenious norm of slice i ($\|A_{i,*,*}\|_F^2$) is in proportional to u_i^2, and the diagonals of the tensor ($|A_{i,i,i}|$) is actually $|u_i|^3$, giving us an accurate estimate. We can choose per-slice samples $\widehat{b}_i$ accordingly. For those slices that potentially have $\|A_{i,*,*}\|_F \gtrsim 1/n^{1/3}$, to estimate the norm of that slice i, we *oversample* them by choosing all the n^2 samples of that $n \times n$ matrix $A_{i,*,*}$. Since we can show there are only $o(n)$ slices that have large estimates for $\|A_{i,*,*}\|_F$, then the total number of samples is $o(n^3)$, which is sublinear in the size of tensor A.

Lemma 22.8.4 (rank-1 tensor)**.** *There exists some constant $\alpha > 0$, for any tensor $A = E + u^{\otimes 3} \in \mathbb{R}^{n^3}$ with $\|u\|_2 = 1$,$\|E\|_2 \leq 1/n$. There exists an algorithm that takes $O(n^{3-\alpha})$ samples/time is able to output a vector $\tau \in \mathbb{R}^n$ which is a good and efficient estimate of A, i.e., τ, A satisfies (22.5) and (22.6).*

Proof. For each slice $i \in [n]$, we have $\|E_{i,*,*}\|_F \leq 1/\sqrt{n}$ (by part 3 of Lemma 22.8.2) and $\|A_{i,*,*}\|_F = \|E_{i,*,*} + u_i \cdot uu^\top\|_F$, which implies

$$|u_i| - 1/\sqrt{n} \leq \|A_{i,*,*}\|_F \leq |u_i| + 1/\sqrt{n}.$$

Also, observing the entries from main diagonal of tensor which is this form, $|A_{i,i,i}| = |E_{i,i,i} + u_i^3|$, using $|E_{i,i,i}| \leq 1/n$ (by part 4 of Lemma 22.8.2), we have

$$|u_i|^3 - 1/n \leq |A_{i,i,i}| \leq |u_i|^3 + 1/n.$$

If there is no noise, the diagonal term is exactly u_i^3, and we can get accurate $\|A_{i,*,*}\|_F = u_i^2$. However, consider the potential noise term, we have to discuss the following two cases:

Part 1. $|A_{i,i,i}| \geq 2/n$. In this case, $|u_i|^3 \geq 1/n$ and $\|A_{i,*,*}\|_F \geq |u_i| - 1/\sqrt{n} \geq 1/n^{1/3} - 1/\sqrt{n} \geq 1/(2n^{1/3})$. For such slice i, we need to take $\widehat{b}_i = O(n^2)$ examples to guarantee that the bounds on $|[\epsilon_{2,A}(u)]_i|$ and $\langle \omega, \epsilon_{2,A}(u) \rangle^2$ are enough.

Let S denote a set of indices such that, for each $i \in S$, $|A_{i,i,i}| \geq 2/n$. By triangle inequality, it means $|u_i|^3 \geq 1/n$ and then $|u_i| \geq 1/n^{1/3}$. We can show that $|S|$ is at most $n^{2/3}$, because u is a unit vector and there are at most $n^{2/3}$ coordinates i could have $|u_i| \geq 1/n^{1/3}$. Overall, the sample complexity for all these slices is $n^{3-1/3}$.

Part 2. $|A_{i,i,i}| < 2/n$. In this case, $\|A_{i,*,*}\|_F \leq |u_i| + 1/\sqrt{n} \leq 3/n^{1/3} + 1/\sqrt{n} \leq 4/n^{1/3}$. Thus, for such slice, we choose $\widehat{b}_i = O(b\|A_{i,*,*}\|_F^2/\|A\|_F^2) = O(b/n^{2/3})$ samples is sufficient for the error bounds. The number of such slice can be $O(n)$, thus the total number of samples for this part is $O(n^{3-2/3})$.

We set $\tau_i = 1$ for part 1 and $\tau_i = O(1/n^{2/3})$ for part 2.

□

Now we consider a further case with tensor rank $k = 2$. Because there are two distinct eigenvectors, the way of estimating $\|A_{i,*,*}\|_F$ becomes trickier. Besides the diagonal terms of the tensor, we also need to look at the diagonal terms of each tensor slice, $A_{i,j,j}$. Given parameters b, $\alpha \in (0,1)$ and tensor $A \in \mathbb{R}^{n^3}$, we use the sampling procedure in Algorithm 22.2.

22.8.3 Order $p = 3$, rank $k = 2$ tensor

Lemma 22.8.5 (rank-2 tensor)**.** *There exists some constant $\alpha > 0$, for any tensor $A = E + u^{\otimes 3} + v^{\otimes 3} \in \mathbb{R}^{n^3}$ with $\|u\|_2 = \|v\|_2 = 1$, $u \cdot v = 0$, $\|E\|_2 \leq 1/n$. There exists an algorithm that takes $O(n^{3-\alpha})$ samples/time is able to output a vector $\tau \in \mathbb{R}^n$ which is a good and efficient estimate of A with probability $1 - 1/\operatorname{poly}(n)$, i.e., τ, A satisfies (22.5) and (22.6).*

Proof. We assume that $A = E + u \otimes u \otimes u + v \otimes v \otimes v$. Note that we ignore the eigenvalues λ_1 and λ_2 because it will only be constants before $u \otimes u \otimes u$ and $v \otimes v \otimes v$ and our proof is unaffected.

Because vector $u \in \mathbb{R}^n$ is orthogonal to vector $v \in \mathbb{R}^n$, then

$$u^\top \cdot v = \sum_{i=1}^{n} u_i \cdot v_i = 0 \tag{22.7}$$

Consider the absolute value on main diagonal of tensor,

$$|A_{i,i,i}| \leq |E_{i,i,i}| + |u_i^3 + v_i^3| \leq 1/n + |u_i^3 + v_i^3|$$

where the first step follows by triangle inequality and the second step follows by Part 3 of Lemma 22.8.2 where $|E_{i,i,i}| \leq 1/n$. Now we start to prove the 4 parts of our procedure.

Part 1. Following the same reasoning in Lemma 22.8.4, for $|A_{i,i,i}| \geq 3/n$, either one of u_i^3 and v_i^3 is large or both u_i^3 and v_i^3 is large and have the same sign, both of these two situations imply that $\|A_{i,*,*}\|_F^2$ is large. Then we need to set $\tau_i = 1$ and take $\widehat{b}_i = O(n^2)$ samples for these slice. Also, the number of such slice is $O(n^{2/3})$ by using the similar argument in part 1 of Lemma 22.8.4. But unlike the rank-1 case, even if $|A_{i,i,i}| < 3/n$, we cannot conclude that $\|A_{i,*,*}\|_F^2$ is small (i.e., $\|A_{i,*,*}\|_F^2 \lesssim 1/n^{2/3}$), because u_i^3 and v_i^3 can have opposite sign and then cancel each other.

Part 2. For each $i \in [n]$, if $|A_{i,i,i}| < 3/n$, we take $O(b/n^\alpha)$ samples for slice i. By doing this, we can guarantee that the error bounds given by Equation (22.2) are sufficient for slice i as long as $\|A_{i,*,*}\|_F^2 \leq 5/n^\alpha$.

Now we need to handle the slice i where $\|A_{i,*,*}\|_F^2 > 5/n^\alpha$ but $|A_{i,i,i}| < 3/n$ in the next part.

Part 3. Let us consider slice $i \in [n]$ such that

$$\|A_{i,*,*}\|_F^2 = \|E_{i,*,*} + u_i \cdot uu^\top + v_i \cdot vv^\top\|_F^2 > 5/n^\alpha, \text{ and } |A_{i,i,i}| = |E_{i,i,i} + u_i^3 + v_i^3| < 3/n. \tag{22.8}$$

Using Part 3 and 4 of Lemma 22.8.2, we have

$$\|E_{i,*,*}\|_F^2 \leq 1/n, \text{ and } |E_{i,i,i}| \leq 1/n. \tag{22.9}$$

Combining Equation (22.8) and (22.9) with triangle inequality implies,

$$u_i^2 + v_i^2 > 4/n^\alpha, \text{ and } |u_i^3 + v_i^3| < 4/n.$$

Using $u_i^2 + v_i^2 > 4/n^\alpha$ we can obtain the following result,

$$\begin{aligned}
&\implies \text{either} \quad u_i^2 > 2/n^\alpha \quad \text{or } v_i^2 > 2/n^\alpha \\
&\implies \text{either} \quad |u_i| > 2/n^{\alpha/2} \quad \text{or } |v_i| > 2/n^{\alpha/2} \\
&\implies \text{either} \quad |u_i|^3 > 2/n^{3\alpha/2} \quad \text{or } |v_i|^3 > 2/n^{3\alpha/2}.
\end{aligned}$$

Note that $|u_i^3 + v_i^3| < 4/n$, $3\alpha/2 < 1$ and $\text{sign}(u_i) \neq \text{sign}(v_i)$, without loss of generality, assume $|u_i|^3 > 2/n^{3\alpha/2}$, then we have $|v_i|^3 > |u_i|^3 - |u_i^3 + v_i^3| > 2/n^{3\alpha/2} - 4/n > 1/n^{3\alpha/2}$. Therefore, we can obtain that

$$\text{both } |u_i|^3 > 1/n^{3\alpha/2} \text{ and } |v_i|^3 > 1/(n^{3\alpha/2}).$$

Thus, $|u_i v_i| > 1/n^\alpha$. Recall that the i is the index we choose at the beginning, and u, v are unit vectors, i.e.,$\|u\|_2 = \|v\|_2 = 1$.

Define set $S \subseteq [n]$, such that for any $j \in S$, $\text{sign}(u_j) = \text{sign}(v_j)$. Define $\overline{S} = [n] \backslash S$. By Equation (22.7),

$$0 = \sum_{j \in S} u_j v_j + \sum_{j' \in \overline{S}} u'_j v'_j$$

It is obvious that the i satisfying Equation (22.8) belongs to $\overline{S}$, then we can lower bound the contribution from set $\overline{S}$,

$$\left| \sum_{j' \in \overline{S}} u_{j'} v_{j'} \right| \geq |u_i v_i| + \left| \sum_{j' \in \overline{S} \backslash \{i\}} u_{j'} v_{j'} \right| > 1/n^\alpha$$

where the first step follows by triangle inequality and the second step follows by $|u_i v_i| > 1/n^\alpha$. Using triangle inequality again, we can also lower bound the contribution from set S, $|\sum_{j \in S} v_j u_j| > 1/n^\alpha$.

For any $j \in S$, $|u_j^3 + v_j^3| < 4/n$ and $\text{sign}(u_j) = \text{sign}(v_j)$. Then, $|u_j^3| < 4/n$ and $|v_j^3| < 4/n$, which implies $|u_j| < 2/n^{1/3}, |v_j| < 2/n^{1/3}$. Furthermore, we can obtain $|u_j \cdot v_j| < 4/n^{2/3}$, so we can lower bound the number of entries in set S,

$$|S| \geq \left| \sum_{j \in S} u_j v_j \right| / \left(\max_{j \in S} |u_j v_j| \right) > (1/n^\alpha)/(4/n^{2/3}) = n^{2/3-\alpha}/4.$$

If we take $\Theta(n^{1/3+\alpha} \log n)$ samples out of $[n]$ slices uniformly at random, then with probability $1 - 1/\operatorname{poly}(n)$, there exists at least one index j from set S. Since for all $j' \in S$, $\text{sign}(u_{j'}) = \text{sign}(v_{j'})$. Then u_j and v_j have the same sign. But set S is not good enough, because there is no lower bound for $|u_j v_j|$, for all $j \in S$. To remove this issue, we can remove the small entries from set S to obtain another set F.

Let $F \subseteq S$ denote the set of coordinates such that for any $j \in F$, $|u_j v_j| \geq 1/(2n^{1+\alpha})$. We can upper bound the contribution from set $S \backslash F$,

$$\left| \sum_{j \in S \backslash F} u_j v_j \right| = \sum_{j \in S \backslash F} |u_j v_j| \leq |S \backslash F| \cdot 1/(2n^{1+\alpha}) \leq n \cdot 1/(2n^{1+\alpha}) = 1/(2n^{\alpha}).$$

Thus using triangle inequality, we can lower bound the contribution from set F,

$$\left| \sum_{j \in F} u_j v_j \right| \geq \left| \sum_{j \in S} u_j v_j \right| - \left| \sum_{j \in S \backslash F} u_j v_j \right| \geq 1/n^{\alpha} - 1/(2n^{\alpha}) = 1/(2n^{\alpha}),$$

which implies,

$$|F| \geq \left| \sum_{j \in F} u_j v_j \right| / \left(\max_{j \in F} |u_j v_j| \right) > (1/2n^{\alpha})/(4/n^{2/3}) = n^{2/3-\alpha}/8.$$

Notice that $|F| \gtrsim |S|$, thus after taking $\Theta(n^{1/3+\alpha} \log n)$ samples over $[n]$, we still can hope to see one j from F. Now the idea is, we use index j to find out the large entry on diagonal of slice $A_{j,*,*}$. By doing this, we can recover some index l such that the norm of l-th slice is large, but $|A_{l,l,l}|$ looks small.

For each $j \in F$, let us look at the diagonal entry of slice $A_{j,*,*}$, consider entry $A_{j,l,l}$,

$$|A_{j,l,l}| = |E_{j,l,l} + u_j u_l^2 + v_j v_l^2|.$$

Because both $u_l^2 > 2/n^{\alpha}$ and $v_l^2 > 2/n^{\alpha}$, and either $|u_j| \geq 1/(2n^{1/2+\alpha/2})$ or $|v_j| \geq 1/(2n^{1/2+\alpha/2})$, then we have

$$|A_{j,l,l}| \geq |u_j u_l^2 + v_j v_l^2| - |E_{j,l,l}| \geq 1/n^{1/2+3\alpha/2} - 1/n \geq 1/(2n^{1/2+3\alpha/2}).$$

We need to show that $\|A_{j,*,*}\|_F^2$ is small. Since u_j and v_j have the same sign and $|A_{j,j,j}| \leq 3/n$, then $|u_j|^3 \leq 4/n$ which implies $|u_j|^2 \leq 4/n^{2/3}$. Thus $|u_j^2 + v_j^2| \leq 8/n^{2/3}$. Furthermore, we

have

$$\|A_{j,*,*}\|_F^2 \leq 2(\|E_{j,*,*}\|_F^2 + |u_j^2 + v_j^2|) \leq 2(1/n + 8/n^{2/3}) \leq 18/n^{2/3}.$$

Thus,

$$\{\#l \text{ such that } |A_{j,l,l}|^2 \geq 1/(4n^{1+3\alpha})\} \leq \frac{18/n^{2/3}}{1/(4n^{1+3\alpha})} = 72n^{1/3+3\alpha},$$

which means number of $|A_{j,l,l}|^2$ is at least $\Theta(1/n^{1+3\alpha})$ is at most $O(n^{1/3+3\alpha})$.

Part 4. We consider the i such that

$$\text{sign}(u_i) = \text{sign}(v_i) \text{ and } |A_{i,i,i}| = |E_{i,i,i} + u_i^3 + v_i^3| \geq 3/n.$$

Using $|E_{i,i,i}| \leq 1/n$, then we have either $|u_i| \geq 1/n^{1/3}$ or $|v_i| \geq 1/n^{1/3}$. Let us consider the jth diagonal entry of slice i (which is matrix $A_{i,*,*}$),

$$|A_{i,j,j}| = |E_{i,j,j} + u_i u_j^2 + v_i v_j^2|.$$

We want to recover the j such that

$$\|A_{j,*,*}\|_F^2 \geq 5/n^{\alpha}.$$

Then either $u_j^2 \geq 1/n^{\alpha}$ or $v_j^2 \geq 1/n^{\alpha}$. Then

$$|A_{i,j,j}| \geq |u_i u_j^2 + v_i v_j^2| - |E_{i,j,j}| \geq 1/n^{1/3+\alpha} - 1/n \geq 1/(2n^{1/3+\alpha}).$$

Thus, $|A_{i,j,j}|^2 \geq 1/(4n^{2/3+2\alpha})$. Because $\|A\|_F^2 = 1$, the total number of such entries over entire tensor is at most $O(n^{2/3+2\alpha})$.

Sample complexity. Overall the sample complexity over four parts is dominated by $n^{1-\alpha}$ and $n^{2/3+4\alpha}$. To minimize the sample complexity, we can set $1-\alpha = 2/3+4\alpha$ which implies $\alpha = 1/15$. Thus, the sample is $O(n^{3-1/15})$, which is sublinear in tensor size. □

For the rank-2 case, we may also want to recover the second eigenvalue and eigenvector. Because the recovered first eigenvalue $\widehat{\lambda}_1$ and eigenvector $\widehat{v}_1$ contain noise, we need to show that under this noise, the oversampling procedure still works.

Lemma 22.8.6 (Deflation for rank-2 tensor)**.** *Given tensor $A = E + (u^{\otimes 3} - \widehat{u}^{\otimes 3}) + v^{\otimes 3} \in \mathbb{R}^{n^3}$ where $\|u\|_2 = \|\widehat{u}\|_2 = \|v\|_2 = 1$, $\|u - \widehat{u}\|_2 \leq 1/\sqrt{n}$, $u \cdot v = 0$ and $\|E\| \leq 1/n$. There exists an algorithm that takes $O(n^{3-\alpha})$ samples/time is able to output a vector $\tau \in \mathbb{R}^n$ which is a good and efficient estimate of A with probability $1 - 1/\operatorname{poly}(n)$, i.e., τ, A satisfies (22.5) and (22.6).*

Proof. We use u_i to denote the i-th coordinate of vector $u \in \mathbb{R}^n$. Because $\|u - \widehat{u}\|_2 \leq 1/\sqrt{n}$, then for all $i \in [n]$, $|u_i - \widehat{u}_i| \leq 1/\sqrt{n}$. The proof sketch follows the proof of Lemma 22.8.4, we need to use the information from $|A_{i,i,i}|$ and $|A_{i,j,j}|$ to figure out if i-th slice has large Frobenius norm, for any $i, j \in [n]$, we have,

$$
\begin{aligned}
|A_{i,i,i}| &= |E_{i,i,i} + u_i^3 - \widehat{u}_i^3 + v_i^3|, && \text{where } |E_{i,i,i}| \leq 1/n, |u_i^3 - \widehat{u}_i^3| \leq 4/\sqrt{n}; \\
|A_{i,j,j}| &= |E_{i,j,j} + u_i u_j^2 - \widehat{u}_i \widehat{u}_j^2 + v_i v_j^2|, && \text{where } |E_{i,j,j}| \leq 1/n, |u_i u_j^2 - \widehat{u}_i \widehat{u}_j^2| \leq 2/\sqrt{n}.
\end{aligned}
$$

Consider the Frobenious norm of i-th slice, it can be written as

$$
\|A_{i,*,*}\|_F = \|E_{i,*,*} + (u_i \cdot uu^\top - \widehat{u}_i \cdot \widehat{u}\widehat{u}^\top) + v_i \cdot vv^\top\|_F,
$$

where $\|E_{i,*,*}\|_F \leq 1/\sqrt{n}$(by part 3 of Lemma 22.8.2) and $\|u_i \cdot uu^\top - \widehat{u}_i \cdot \widehat{u}\widehat{u}^\top\|_F \leq 3/\sqrt{n}$(by part 4 of Fact 22.8.3).

For any $\beta \in (0, 1)$, we use R to denote a set of indices such that $\{i \mid |u_i - \widehat{u}_i| >$

$1/n^{1/2+\beta/2}, i \in [n]\}$. Then we know that $|R| \leq n^{\beta}$, otherwise

$$\|u - \widehat{u}\|_2^2 = \sum_{i \in [n]} |u_i - \widehat{u}_i|^2 \geq \sum_{i \in R} |u_i - \widehat{u}_i|^2 > 1/n,$$

which is a contradiction to $\|u - \widehat{u}\|_2 \leq 1/\sqrt{n}$.

Part 1. Follow the similar reason in Lemma 22.8.4 and 22.8.5, for $|A_{i,i,i}| \geq 4/n$, then at least one of $u_i^3 - \widehat{u}_i^3$ and v_i^3 has large absolute value or both of them are large and have the same sign. Both of these two situations imply that $\|A_{i,*,*}\|_F^2$ is large. Then we need to set $\tau_i = 1$ and take $\widehat{b}_i = O(n^2)$ samples for each of these slices. Also, the number of such slice could have

$$\max(|u_i|, |v_i|, |\widehat{u}_i|) > 1/n^{1/3}$$

is $O(n^{2/3})$ by using the similar argument in part 1 of Lemma 22.8.4. Even if $|A_{i,i,i}| < 4/n$, we cannot conclude that $\|A_{i,*,*}\|_F^2$ is small, because $u_i^3 - \widehat{u}_i^3$ and v_i^3 can have opposite sign and then cancel each other.

Part 2. For each $i \in [n]$, if $|A_{i,i,i}| < 4/n$, we take $O(b/n^{\alpha})$ samples for slice i. By doing this, we can guarantee that the error bounds are sufficient for slice i as long as $\|A_{i,*,*}\|_F^2 \leq 5/n^{\alpha}$.

Now we need to handle the slice where $\|A_{i,*,*}\|_F^2 > 5/n^{\alpha}$ but $\|A_{i,i,i}| < 4/n$ in the next part.

Part 3.

Let us consider slice $i \in [n]$ such that

$$\|A_{i,*,*}\|_F^2 = \|E_{i,*,*} + u_i \cdot uu^{\top} - \widehat{u}_i \cdot \widehat{u}\widehat{u}^{\top} + v_i \cdot vv^{\top}\|_F^2 > 5/n^{\alpha}, \tag{22.10}$$

$$|A_{i,i,i}| = |E_{i,i,i} + u_i^3 - \widehat{u}_i^3 + v_i^3| < 3/n. \tag{22.11}$$

Using Part 3 and 4 of Lemma 22.8.2, we have

$$\|E_{i,*,*}\|_F^2 \leq 1/n, \text{ and } |E_{i,i,i}| \leq 1/n. \tag{22.12}$$

Combining Equation (22.10), (22.11), (22.12) and Fact 22.8.3 with triangle inequality implies,

$$\begin{aligned} 2v_i^2 &\geq 5/n^\alpha - 2\|E_{i,*,*}\|_F^2 - 2\|u_i \cdot uu^\top - \widehat{u}_i \cdot \widehat{u}\widehat{u}^\top\|_F^2 \\ &\geq 5/n^\alpha - 2/n - 2 \cdot 9/n \\ &\geq 4/n^\alpha \end{aligned}$$

and

$$|u_i^3 - \widehat{u}_i^3 + v_i^3| < 6/n.$$

Combining $\alpha \in (0, 2/3)$ with the above two equations, we can conclude that $\text{sign}(u_i - \widehat{u}_i) \neq \text{sign}(v_i)$, $|v_i|^3 > 1/n^{3\alpha/2}$ and $|u_i^3 - \widehat{u}_i^3| \geq 1/n^{3\alpha/2}$. As long as we choose $\alpha \in (0, 1/3)$, then it is impossible that

$$1/n^{3\alpha/2} \leq |u_i^3 - \widehat{u}_i^3| \leq 4/\sqrt{n}.$$

Thus, we know part 3 will never happen.

□

22.8.4 Even order tensor

In this section, we prove the even order case.

Lemma 22.8.7. *For any $E \in \mathbb{R}^{n^4}$ and any $\beta > 0$, if $\|E\|_2 := \max_{\|u\|_2=\|v\|_2=\|w\|_2=\|x\|_2} |E(u,v,w,x)| \leq 1/n^{1+\beta}$, then for all $i \in [n]$,*

1. $\max_{\|v\|_2=\|w\|_2=\|x\|_2=1} \|E(I,v,w,x)\|_2 \leq \|E\|_2$,
2. $\|E_{i,*,*,*}\|_2 \leq 1/n^{1+\beta}$,
3. $\|E_{i,*,*,*}\|_F \leq 1/\sqrt{n}$,
4. $\forall j \in [n], |E_{i,j,j,j}| \leq 1/n^{1+\beta}$.

Proof. **Part 1.**

$$\begin{aligned}
\|E\|_2^2 = & \max_{\|u\|_2=\|v\|_2=\|w\|_2=\|x\|_2=1} |E(u,v,w,x)|^2 \\
= & \max_{\|u\|_2=\|v\|_2=\|w\|_2=\|x\|_2=1} \left(\sum_{i=1}^n u_i \cdot \sum_{j=1}^n \sum_{k=1}^n E_{i,j,k,t} v_j w_k x_t \right)^2 \\
\geq & \max_{\|v\|_2=\|w\|_2=\|x\|_2=1} \sum_{i=1}^n u_i^2 \cdot \sum_{i=1}^n \left(\sum_{j=1}^n \sum_{k=1}^n E_{i,j,k,t} v_j w_k x_t \right)^2 \\
= & \max_{\|v\|_2=\|w\|_2=\|x\|_2=1} \sum_{i=1}^n \left(\sum_{j=1}^n \sum_{k=1}^n E_{i,j,k,t} v_j w_k x_t \right)^2 \\
= & \max_{\|v\|_2=\|w\|_2=\|x\|_2=1} \|E(I,v,w,x)\|_2^2,
\end{aligned}$$

where the first inequality follows by setting $u_i = c \cdot \sum_{j=1}^n \sum_{k=1}^n \sum_{t=1}^n E_{i,j,k,t} v_j v_k v_t$, $\forall i \in [n]$ and $\sum_{i=1}^n u_i^2 = 1$.

Part 2. By definition of $\|\cdot\|_2$ for a tensor $E \in \mathbb{R}^{n^4}$, we have

$$\max_{\|u\|_2=1} \|E(I,v,w,x)\|_2 \leq \max_{\|u\|_2=\|v\|_2=\|w\|_2=\|x\|_2=1} |E(u,v,w,x)|$$

By definition of $E(I,v,w,x) \in \mathbb{R}^n$, we have

$$E(I,v,w,x)_i = \sum_{j=1}^n \sum_{k=1}^n \sum_{t=1}^n E_{i,j,k,t} v_j w_k x_t.$$

Thus, we can upper bound the $\|\cdot\|_2$ for matrix $E_{i,*,*,*} \in \mathbb{R}^{n^3}$ in the following sense,

$$\begin{aligned}
1/n^{1+\beta} &\geq \|E\|_2 \\
&\geq \max_{\|v\|_2=\|w\|_2=\|x\|_2=1} \|E(I,v,w,x)\|_2 && \text{by Part 1.} \\
&= \max_{\|v\|_2=\|w\|_2=\|x\|_2=1} \left(\sum_{i=1}^n \left(\sum_{j=1}^n \sum_{k=1}^n \sum_{t=1}^n E_{i,j,k,t} v_j w_k x_t \right)^2 \right)^{1/2} \\
&\geq \max_{\|v\|_2=\|w\|_2=\|x\|_2=1} \left| \sum_{j=1}^n \sum_{k=1}^n \sum_{t=1}^n E_{i,j,k,t} v_j w_k x_t \right| \\
&= \|E_{i,*,*,*}\|_2, && \text{by Fact 22.8.1}
\end{aligned}$$

which completes the proof of Part 2.

Part 3. By the proof of Lemma 22.8.2, $\forall i,j \in [n]$ we have

$$\|E_{i,j,*,*}\|_2 \leq \|E_{i,*,*,*}\|_2 \leq 1/n^{1+\beta}.$$

Since $\|E_{i,j,*,*}\|_2 \leq 1/n^{1+\beta}$ and the rank of $n \times n$ matrix is at most n, thus,

$$\|E_{i,*,*,*}\|_F^2 = \sum_{j=1}^n \|E_{i,j,*,*}\|_F^2 \leq \sum_{j=1}^n n \cdot \|E_{i,j,*,*}\|_2^2 \leq n \cdot n \cdot (1/n^{1+\beta})^2 \leq 1/n^{2\beta},$$

which implies $\|E_{i,*,*,*}\|_F \leq 1/n^\beta$ and finishes the proof of Part 3.

Part 4. For each slice $i \in [n]$, by $\|E_{i,*,*,*}\|_2 \leq 1/n^{1+\beta}$ and definition of $\|\cdot\|_2$ of a matrix, we have

$$\max_{\|v\|_2=\|w\|_2=\|x\|_2=1} \left| \sum_{j=1}^{n} \sum_{k=1}^{n} \sum_{t=1}^{n} E_{i,j,k,t} v_j w_k x_t \right| \leq 1/n,$$

choosing $v = w = x = e_j$ where e_j is the vector that jth position is 1 and all the other positions are 0. Thus, we know for any $j \in [n]$, $|E_{i,j,j,j}| \leq 1/n^{1+\beta}$. Thus, we complete the proof of Part 4.

□

Lemma 22.8.8 (Order $p = 4$)**.** *For sufficiently small constant $\gamma > 0$, for arbitrarily small constant $\beta > 0$, for any $k \leq n^\gamma$, $t < k$, for any given tensor $A = E + \sum_{i=1}^{k} u_i^{\otimes 4} - \sum_{i=1}^{t} \widehat{u}_i^{\otimes 4} \in \mathbb{R}^{n^4}$ with $\|E\|_2 \leq 1/n^{1+\beta}$, $u_i \cdot u_j = 0, \forall i, j \in [k]$, $\|\widehat{u}_i - u_i\|_2 \leq 1/\sqrt{n}$, $\forall i \in [t]$. There exists an algorithm that takes $O(n^{4-2\beta})$ samples/time is able to output a vector $\tau \in \mathbb{R}^n$ which is a good and efficient estimate of A, i.e., τ, A satisfies (22.5) and (22.6).*

Proof. Let $A = E + \sum_{i=1}^{k} u_i^{\otimes 4} - \sum_{i=1}^{t} \widehat{u}_i^{\otimes 4}$. For any $j \in [n]$, we use $u_{i,j}$ to denote the j-th coordinate of vector $u_i \in \mathbb{R}^n$. We can observe the main diagonal of A,

$$|A_{j,j,j,j}| = \left| E_{j,j,j,j} + \sum_{i=1}^{t} (u_{i,j}^4 - \widehat{u}_{i,j}^4) + \sum_{i=t+1}^{k} u_{i,j}^4 \right|.$$

Since $\forall i \in [t]$, $\|u_i - \widehat{u}_i\|_2 \leq 1/\sqrt{n}$, then $\forall j \in [n]$, $|u_{i,j} - \widehat{u}_{i,j}| \leq 1/\sqrt{n}$. Furthermore, we have

$$|u_{i,j}^4 - \widehat{u}_{i,j}^4| = |(u_{i,j} - \widehat{u}_{i,j})(u_{i,j} + \widehat{u}_{i,j})(u_{i,j}^2 + \widehat{u}_{i,j}^2)| \leq 4/\sqrt{n}.$$

By part 4 of Lemma 22.8.7, we have $|E_{j,j,j,j}| \leq 1/n$. Because there is no cancellation in $|A_{j,j,j,j}|$ and $1/\sqrt{n} \gg 1/n$, we can use the same threshold $1/n$ as Lemma 22.8.4 again.

Part 1, $|A_{j,j,j,j}| \geq 1/n$. In this case, for $i \in \{t+1, t+2, \cdots, k\}$ one of $|u_{i,j}|$ is large and $\|A_{j,*,*,*}\|_F^2$ is also large. Let S denote a set of indices such that, for each $j \in S$, $|A_{j,j,j,j}| \geq 1/n$. We can show an upper bound for $|S|$. For each j, there must exists one $i > t$ such that $|u_{i,j}| \geq 1/(kn^{1/4}) = 1/n^{1/4-\gamma}$. There are at most k different i, for each i we know u_i is a unit vector and then there are at most $n^{1/2-2\gamma}$ coordinates j of vector u_i could have $|u_{i,j}| \geq 1/n^{1/4-\gamma}$. Thus $|S| \leq kn^{1/2-2\gamma} \leq n^{1/2-\gamma}$. For each $i \in S$, we take $O(n^3)$ samples from slice j. Then the total number of samples is $O(n^{3+1/2-\gamma})$.

Part 2, $|A_{j,j,j,j}| < 1/n$. In this case, we know there is no $i > t$ such that $|u_{i,j}| \geq 2/n^{1/4}$. We can upper bound $\|A_{j,*,*,*}\|_F$,

$$
\begin{aligned}
&\|A_{j,*,*,*}\|_F \\
&\leq \|E_{j,*,*,*}\|_F + \left\| \sum_{i=1}^{t} u_{i,j} \cdot u_i^{\otimes 3} - \widehat{u}_{i,j} \cdot \widehat{u}_i^{\otimes 3} \right\|_F + \left\| \sum_{i=t+1}^{k} u_{i,j} \cdot u_i^{\otimes 3} \right\|_F \\
&\leq \|E_{j,*,*,*}\|_F + \left\| \sum_{i=1}^{t} u_{i,j} \cdot u_i^{\otimes 3} - u_{i,j} \cdot \widehat{u}_i^{\otimes 3} \right\|_F + \left\| \sum_{i=1}^{t} u_{i,j} \cdot \widehat{u}_i^{\otimes 3} - \widehat{u}_{i,j} \cdot \widehat{u}_i^{\otimes 3} \right\|_F + \left\| \sum_{i=t+1}^{k} u_{i,j} \cdot u_i^{\otimes 3} \right\|_F \\
&\leq 1/n^{\beta} + k \cdot 3/\sqrt{n} + k \cdot 1/\sqrt{n} + k \cdot 2/n^{1/4} \\
&\leq 3/n^{\min(\beta, 1/4-\gamma)}
\end{aligned}
$$

where the bound of the second term in the third step follows by Fact 22.8.3.

Thus, as long as $\beta < 1/4-\gamma$, for each such slice i, we choose $\widehat{b}_i = O(b\|A_{j,*,*,*}\|_F^2/\|A\|_F^2) = O(b/n^{2\beta})$ samples. Notice that $b = O(n^3)$ in order 4 tensor. The number of such i can be $O(n)$, thus the total sample complexity is $n^{4-2\beta}$. □

We can extend Lemma 22.8.8 to general case where order p is even,

Theorem 22.8.9 (Even order)**.** *For sufficiently small constant $\gamma > 0$, there exists some constant $\alpha > 0$(depends on γ) for any $k \in [1, n^\gamma]$, $p \in [3, n^\gamma]$ $r < k$, for any given tensor $A = E + \sum_{i=1}^{k} \lambda_i u_i^{\otimes p} - \sum_{i=1}^{t} \widehat{\lambda}_i \widehat{u}_i^{\otimes p} \in \mathbb{R}^{n^p}$.*

For any sufficiently large constant $c_0 \geq 100$, there exists a sufficiently small constant $c > 0$, for any $\epsilon \in (0, c\lambda_k/(c_0 p^2 k n^{(p-2)/2})$ if E satisfies that $\|E\|_2 \leq \epsilon/(c_0\sqrt{n})$ and $\forall i \in [t]$.

$$|\widehat{\lambda}_i - \lambda_i| \leq \epsilon, \text{ and } \|\widehat{u}_i - u_i\|_2 \leq \epsilon/\lambda_i$$

There exists an algorithm that takes $O(n^{p-\alpha})$ samples/time is able to output a vector $\tau \in \mathbb{R}^n$ which is a good and efficient estimate of A, i.e., τ, A satisfies (22.5) and (22.6).

22.9 Robust Tensor Power Method Analysis for General Order $p \geq 3$

In Section 22.9.1, we extend several useful facts from 3rd order and 4th order to arbitrary p-th order tensor when $p \geq 3$. In Section 22.9.2, we extend analysis of the original robust tensor power method [AGH+14] from 3rd to arbitrary order $p \geq 3$, some of our proofs also borrow the idea from a very recent work [WA16].

22.9.1 Useful facts

Fact 22.9.1. *For any three unit vectors $u, v, w \in \mathbb{R}^n$, for any $0 \leq x \leq 1$, if $\langle u, w \rangle \geq 1 - x$ and $\langle v, w \rangle = 0$, then $\langle u, v \rangle \leq \sqrt{2x - x^2}$.*

Proof.

$$\begin{aligned}
\langle u, v \rangle =& \cos \theta(u, v) \\
\leq & |\cos \theta(u, w) \cos \theta(v, w)| + |\sin \theta(u, w) \sin \theta(v, w)| \\
\leq & 0 + \sqrt{1 - (1 - x)^2} \\
= & \sqrt{2x - x^2}.
\end{aligned}$$

□

Fact 22.9.2. *For any $E \in \mathbb{R}^{n^p}$, $u, v \in \mathbb{R}^n$, then $|v^\top E(I, u, \cdots, u)| = |E(v, u, \cdots, u)|$.*

Proof. It follows

$$\begin{aligned}|v^\top E(I,u,\cdots,u)| &= \left|\sum_{i_1=1}^n v_{i_1}\cdot\left(\sum_{i_2=1}^n\cdots\sum_{i_p=1}^n E_{i_1,i_2,\cdots,i_p}u_{i_2}\cdots u_{i_p}\right)\right| \\ &= \left|\sum_{i_1=1}^n\sum_{i_2=1}^n\cdots\sum_{i_p=1}^n E_{i_1,i_2,\cdots,i_p}v_{i_1}u_{i_2}\cdots u_{i_p}\right| \\ &= |E(v,u,\cdots,u)|.\end{aligned}$$

□

Fact 22.9.3. *For any $p \geq 3$, given orthogonal tensor $A^* = \sum_{i=1}^k \lambda_j v_j^{\otimes p} \in \mathbb{R}^{n^p}$ and vector $u \in \mathbb{R}^n$, for any $j \in [k]$, we have $|v_j^\top A^*(I,u,\cdots,u)| = \lambda_j |v_j^\top u|^{p-1}$.*

Proof. For any $j \in [k]$, we have

$$\begin{aligned}|v_j^\top A^*(I,u,\cdots,u)| &= \left|\sum_{i=1}^n v_{j,i}\sum_{\ell=1}^k \lambda_\ell v_{\ell,i}\sum_{i_2=1}^n\cdots\sum_{i_p=1}^n v_{\ell,i_2}\cdots v_{\ell,i_p}u_{i_2}\cdots u_{i_p}\right| \\ &= \left|\lambda_j\sum_{i_2=1}^n\cdots\sum_{i_p=1}^n v_{j,i_2}\cdots v_{j,i_p}u_{i_2}\cdots u_{i_p}\right| \\ &= |\lambda_j||v_j^\top u|^{p-1}.\end{aligned}$$

□

Fact 22.9.4. *For any tensor $E \in \mathbb{R}^{n^p}$, for any unit vectors $u, v \in \mathbb{R}^n$,*

1. $|E(u,v,\cdots,v)| \leq \|E\|_2$.
2. $\|E(I,v,\ldots,v)\|_2 \leq \sqrt{n}\|E\|_2$.

Proof. The part 1 trivially follows by definition of $\|E\|_2$. For part 2, we define a unit vector w to be $(1/\sqrt{n}, \cdots, 1/\sqrt{n})$,

$$\begin{aligned}
& \|E(I, v, \ldots, v)\|_2^2 \\
= & \sum_{i_1=1}^n \left(\sum_{i_2=1}^n \cdots \sum_{i_p=1}^n E_{i_1,i_2,\cdots,i_p} v_{i_2} \cdots v_{i_p} \right)^2 \\
= & n \sum_{i_1=1}^n \left(\sum_{i_2=1}^n \cdots \sum_{i_p=1}^n E_{i_1,i_2,\cdots,i_p} w_{i_1} v_{i_2} \cdots v_{i_p} \right)^2 \\
\leq & n \|E\|_2^2,
\end{aligned}$$

which implies $\|E(I, v, \cdots, v)\|_2 \leq \sqrt{n}\|E\|_2$. □

Fact 22.9.5. *For any $p \geq 3$, for any unit vectors $x, y, u, v \in \mathbb{R}^n$, we have*

1. $\|[x \otimes v^{\otimes(p-1)}](I, u, \cdots, u) - [y \otimes v^{\otimes(p-1)}](I, u, \cdots, u)\|_2 = |\langle u, v\rangle|^{p-1} \cdot \|x - y\|_2$.
2. $\forall j \in \{0, 1, \cdots, p-2\}$,

$$\begin{aligned}
& \|[v^{\otimes(1+j)} \otimes x \otimes v^{\otimes(p-2-j)}](I, u, \cdots, u) - [v^{\otimes(1+j)} \otimes y \otimes v^{\otimes(p-2-j)}](I, u, \cdots, u)\|_2 \\
\leq & |\langle u, v\rangle|^{p-2} \cdot \|x - y\|_2.
\end{aligned}$$

Proof. **Part 1.** We consider the i-th coordinate of vector $[x \otimes v^{\otimes(p-1)}](I, u, \cdots, u) \in \mathbb{R}^n$, it can be written as

$$x_i \sum_{i_2=1}^n \cdots \sum_{i_p=1}^n v_{i_2} \cdots v_{i_p} u_{i_2} \cdots u_{i_p} = x_i \sum_{i_2=1}^n v_{i_2} u_{i_2} \cdots \sum_{i_p=1}^n v_{i_p} u_{i_p} = x_i \langle v, u\rangle^{p-1}.$$

Thus,

$$\text{LHS}^2 = \sum_{i=1}^{n} (x_i \langle v, u \rangle^{p-1} - y_i \langle v, u \rangle^{p-1})^2 = \|x - y\|_2^2 \cdot |\langle v, u \rangle|^{2(p-1)}.$$

Part 2. We consider the i-th coordinate of vector $[v^{\otimes(1+j)} \otimes x \otimes v^{\otimes(p-2-j)}](I, u, \cdots, u) \in \mathbb{R}^n$, it can be written as $v_i \langle x, u \rangle \cdot \langle v, u \rangle^{p-2}$. Thus,

$$\text{LHS}^2 = \sum_{i=1}^{n} (v_i \langle x, u \rangle \langle v, u \rangle^{p-2} - v_i \langle y, u \rangle \langle v, u \rangle^{p-2})^2 = \langle x - y, u \rangle^2 \langle v, u \rangle^{2(p-2)} \leq \|x - y\|_2^2 \langle v, u \rangle^{2(p-2)}.$$

□

Fact 22.9.6. *For any two unit vectors u, v, if $\theta(u, v) \in (0, \pi/2)$, then $\|u - v\|_2 \leq \tan \theta(u, v)$.*

Proof. Because $\theta(u, v) \in (0, \pi/2)$, we have $\cos \theta(u, v) \in (0, 1)$. We use x to denote $\langle u, v \rangle$. We want to show $\|u - v\|_2^2 \leq \tan^2 \theta(u, v)$, which is equivalent to

$$\begin{aligned}
& 2 - 2x \leq \sin^2 \theta(u, v) / \cos^2 \theta(u, v) \\
\iff & 2 - 2x \leq 1 - x^2 / x^2 \\
\iff & 0 \leq (1 - x)(1 + x - 2x^2) \\
\iff & 0 \leq (1 - x)^2 (1 + 2x)
\end{aligned}$$

which always holds. Thus we complete the proof. □

Fact 22.9.7. *Given an orthonormal basis, $v_1, v_2, \cdots, v_n$. Let $V = (v_2, \cdots, v_n) \in \mathbb{R}^{n \times (n-1)}$ and $A^* = \sum_{i=1}^{k} \lambda_i v_i^{\otimes p}$. Then for any vector $u \in \mathbb{R}^n$, we have $\|V^\top A^*(I, u, \cdots, u)\|_2^2 = \sum_{j=2}^{k} \lambda_j^2 |v_j^\top u|^{2(p-1)}$.*

Proof.

$$
\begin{aligned}
& \|V^\top A^*(I, u, \cdots, u)\|_2^2 \\
= & \sum_{j=2}^{n} |v_j^\top A^*(I, u, \cdots, u)|^2 \\
= & \sum_{j=2}^{n} \left(\sum_{i=1}^{n} v_{j,i} A^*(I, u, \cdots, u)_i \right)^2 \\
= & \sum_{j=2}^{n} \left(\sum_{i=1}^{n} v_{j,i} \sum_{i_2=1}^{n} \cdots \sum_{i_p=1}^{n} A^*_{i,i_2,\cdots,i_p} u_{i_2} \cdots u_{i_p} \right)^2 \\
= & \sum_{j=2}^{n} \left(\sum_{i=1}^{n} v_{j,i} \sum_{i_2=1}^{n} \cdots \sum_{i_p=1}^{n} (\sum_{\ell=1}^{n} \lambda_\ell v_{\ell,i} v_{\ell,i_2} \cdots v_{\ell,i_p}) u_{i_2} \cdots u_{i_p} \right)^2 \qquad \text{by definition of } A^* \\
= & \sum_{j=2}^{n} \left(\sum_{\ell=1}^{k} \lambda_\ell \sum_{i=1}^{n} v_{j,i} v_{\ell,i} \sum_{i_2=1}^{n} \cdots \sum_{i_p=1}^{n} (v_{\ell,i_2} \cdots v_{\ell,i_p}) u_{i_2} \cdots u_{i_p} \right)^2 \\
= & \sum_{j=2}^{k} \left(\lambda_j \sum_{i_2=1}^{n} \cdots \sum_{i_p=1}^{n} (v_{j,i_2} \cdots v_{j,i_p}) u_{i_2} \cdots u_{i_p} \right)^2 \\
= & \sum_{j=2}^{k} \lambda_j^2 |v_j^\top u|^{2(p-1)}
\end{aligned}
$$

□

22.9.2 Convergence guarantee and deflation

Lemma 22.9.8 (Lemma C.1 and Corollary C.1 in [WA16]). *For any $t \in [k]$ and $\eta \in (0, 1/2)$. Let $\mathcal{U}$ denote a set of random Gaussian vectors in $\mathbb{R}^n$. If $|\mathcal{U}| = \Omega(k \log(1/\eta))$, then with probability at least $1 - \eta$ there exists a vector $u \in \mathcal{U}$ such that*

$$\max_{j \in [k] \setminus \{t\}} |v_j^\top u| \le \frac{1}{4} |v_t^\top u| \text{ and } |v_t^\top u| \ge 1/\sqrt{n}.$$

We generalize Lemma C.2 in [WA16] from $p = 3$ to arbitrary $p \ge 3$,

Lemma 22.9.9. *For sufficiently small constant $c > 0$, for any constant $c_0 \ge 1$, for any $p \ge 3$, for any tensor $A = A^* + \widetilde{E} \in \mathbb{R}^{n^p}$ with $A^* = \sum_{i=1}^k \lambda_i v_i^{\otimes p}$, let*

$$u_{t+1} = A(I, u_t, \cdots, u_t) / \|A(I, u_t, \cdots, u_t)\|_2.$$

For any $\epsilon \le c\lambda_1/(c_0 p^2 k n^{(p-2)/2})$, and unit vector $u_t \in \mathbb{R}^n$, if

$$\|\widetilde{E}(I, u_t, \cdots, u_t)\|_2 \le \begin{cases} 4p\epsilon & \text{if } |v_1^\top u_t| \le 1 - 1/(c_0^2 p^2 k^2) \\ 6\epsilon/c_0 & \text{otherwise} \end{cases}$$

and

$$|\widetilde{E}(v, u_t, \cdots, u_t)| \le \begin{cases} 4\epsilon/\sqrt{n} & \text{if } |v_1^\top u_t| \le 1 - 1/(c_0^2 p^2 k^2) \\ 6\epsilon/(c_0\sqrt{n}) & \text{otherwise} \end{cases}$$

then for any $t \in [T]$,

1. $$\tan\theta(v_1, u_{t+1}) \le \begin{cases} 0.8 \tan\theta(v_1, u_t) & \text{if } |v_1^\top u_t| \le 1 - 1/(c_0^2 p^2 k^2) \\ 0.8 \tan\theta(v_1, u_t) + 18\epsilon/(c_0\lambda_1) & \text{otherwise} \end{cases}$$

2. $\max_{j \in [k] \setminus \{1\}} \lambda_j |v_j^\top u_t|^{p-2} \le (1/4)\lambda_1 |v_1^\top u_t|^{p-2}$.

3. For any $j \in \{2, \cdots, k\}$,

$$|v_j^\top u_{t+1}|/|v_1^\top u_{t+1}| \leq \begin{cases} 0.8|v_j^\top u_t|/|v_1^\top u_t| & \text{if } |v_1^\top u_t| \leq 1 - 1/(c_0^2 p^2 k^2) \\ 0.8|v_j^\top u_t|/|v_1^\top u_t| + 18\epsilon/(c_0 \lambda_1 \sqrt{n}) & \text{otherwise} \end{cases}$$

Proof. **Part 1.** Let $V \in \mathbb{R}^{n \times (n-1)}$ denote an orthonormal basis of the complement of v_1, i.e., V is $(v_2, \cdots, v_k, \cdots, v_n)$ and let $\widetilde{E}_{u_t} = \widetilde{E}(I, u_t, \cdots, u_t) \in \mathbb{R}^n$. We can give an upper bound for $\tan\theta(v_1, u_{t+1})$,

$$\begin{aligned}
& \tan\theta(v_1, u_{t+1}) \\
= \; & \tan\theta(v_1, A(I, u_t, \cdots, u_t)/\|A(I, u_t, \cdots, u_t)\|_2) && \text{by definition of } u_{t+1} \\
= \; & \tan\theta(v_1, A(I, u_t, \cdots, u_t)) && \text{by definition of angle} \\
= \; & \tan\theta(v_1, A^*(I, u_t, \cdots, u_t) + \widetilde{E}(I, u_t, \cdots, u_t)) && \text{by } A = A^* + \widetilde{E} \in \mathbb{R}^{n^p} \\
= \; & \tan\theta(v_1, A^*(I, u_t, \cdots, u_t) + \widetilde{E}_{u_t}) && \text{by } \widetilde{E}_{u_t} := \widetilde{E}(I, u_t, \cdots, u_t) \in \mathbb{R}^n \\
= \; & \frac{\|V^\top [A^*(I, u_t, \cdots, u_t) + \widetilde{E}_{u_t}]\|_2}{|v_1^\top [A^*(I, u_t, \cdots, u_t) + \widetilde{E}_{u_t}]|} && \text{by definition of } \tan\theta \\
\leq \; & \frac{\|V^\top A^*(I, u_t, \cdots, u_t)\|_2 + \|V^\top \widetilde{E}_{u_t}\|_2}{|v_1^\top A^*(I, u_t, \cdots, u_t)| - |v_1^\top \widetilde{E}_{u_t}|}. && \text{by triangle inequality}
\end{aligned}$$

Using Fact 22.9.7, we have

$$\begin{aligned}
\|V^\top A^*(I, u_t, \cdots, u_t)\|_2^2 = & \sum_{j=2}^k \lambda_j^2 |v_j^\top u_t|^{2(p-1)} && (22.13) \\
\leq & \left(\max_{j \in [k] \setminus \{1\}} |\lambda_j|^2 |v_j^\top u_t|^{2(p-2)} \right) \cdot \left(\sum_{j=2}^k |v_j^\top u_t|^2 \right) && (22.14)
\end{aligned}$$

Putting it all together, we have

$$
\begin{aligned}
& \tan\theta(v_1, u_{t+1}) \\
\leq\ & \frac{\|V^\top A^*(I, u_t, \cdots, u_t)\|_2 + \|V^\top \widetilde{E}_{u_t}\|_2}{|v_1^\top A^*(I, u_t, \cdots, u_t)| - |v_1^\top \widetilde{E}_{u_t}|} \\
\leq\ & \tan\theta(v_1, u_t) \cdot \frac{(\|V^\top A^*(I, u_t, \cdots, u_t)\|_2 + \|V^\top \widetilde{E}_{u_t}\|_2)/\|V^\top u_t\|_2}{|v_1^\top A^*(I, u_t, \cdots, u_t)|/|v_1^\top u_t| - |v_1^\top \widetilde{E}_{u_t}|/|v_1^\top u_t|} \\
\leq\ & \tan\theta(v_1, u_t) \cdot \frac{\max_{j\in[k]\setminus\{1\}} \lambda_j |v_j^\top u_t|^{p-2} + \|V^\top \widetilde{E}_{u_t}\|_2/\|V^\top u_t\|_2}{|v_1^\top A^*(I, u_t, \cdots, u_t)|/|v_1^\top u_t| - |v_1^\top \widetilde{E}_{u_t}|/|v_1^\top u_t|} \\
\leq\ & \tan\theta(v_1, u_t) \cdot \frac{\max_{j\in[k]\setminus\{1\}} \lambda_j |v_j^\top u_t|^{p-2} + \|V^\top \widetilde{E}_{u_t}\|_2/\|V^\top u_t\|_2}{\lambda_1 |v_1^\top u_t|^{p-2} - |v_1^\top \widetilde{E}_{u_t}|/|v_1^\top u_t|} \\
\leq\ & \tan\theta(v_1, u_t) \cdot \frac{(1/4)\lambda_1 |v_1^\top u_t|^{p-2} + \|V^\top \widetilde{E}_{u_t}\|_2/\|V^\top u_t\|_2}{\lambda_1 |v_1^\top u_t|^{p-2} - |v_1^\top \widetilde{E}_{u_t}|/|v_1^\top u_t|} \\
\leq\ & \tan\theta(v_1, u_t) \cdot (1/4) \cdot \underbrace{\frac{1}{1 - |v_1^\top \widetilde{E}_{u_t}|/(\lambda_1 |v_1^\top u_t|^{p-1})}}_{A} \\
& + \underbrace{\frac{1}{1 - |v_1^\top \widetilde{E}_{u_t}|/(\lambda_1 |v_1^\top u_t|^{p-1})}}_{A} \cdot \underbrace{\frac{\|V^\top \widetilde{E}_{u_t}\|_2}{\lambda_1 |v_1^\top u_t|^{p-1}}}_{B} \\
=\ & \tan\theta(v_1, u_t) \cdot (1/4) \cdot A + A \cdot B
\end{aligned}
$$

where the second step follows from $\tan\theta(v_1, u_t) = \frac{\|V^\top u_t\|_2}{|v_1^\top u_t|}$, the third step follows from Equation (22.13), the forth step follows from Fact 22.9.3,

the fifth step follows from Part 2 of Lemma 22.9.9.

We show

Claim 22.9.10. *For any $t \in [T]$, $|v_1^\top u_0| \leq |v_1^\top u_t|$.*

Proof. By induction hypothesis, we assume there exists some sufficiently small constant $c > 0$

such that $\tan\theta(v_1, u_t) \le 0.8\tan\theta(v_1, u_{t-1}) + c$. Thus, we can show

$$\begin{aligned}
\tan\theta(v_1, u_t) &\le 0.8(0.8\tan\theta(v_1, u_{t-1}) + c) + c \\
&\le 0.8^t \tan\theta(v_1, u_0) + c\sum_{j=0}^{t-1} 0.8^j \\
&\le 0.8^t \tan\theta(v_1, u_0) + 5c \\
&\le \tan\theta(v_1, u_0) && \text{by } \tan\theta(v_1, u_0) = \omega(1)
\end{aligned}$$

which implies $\theta(v_1, u_t) \le \theta(v_1, u_0)$ and hence $|v_1^\top u_t| = \cos\theta(v_1, u_t) \ge \cos\theta(v_1, u_0) = |v_1^\top u_0|$.

□

Then we can upper bound A and B

Claim 22.9.11. $A \le 1.1$.

Proof. Note that $|\widetilde{E}(v_j, u_t, \cdots, u_t)| = |v_j^\top \widetilde{E}(I, u_t, \cdots, u_t)| = |v_j^\top \widetilde{E}_{u_t}|$. Because

$$\begin{aligned}
|\widetilde{E}(v_j, u_t, \cdots, u_t)| &\le 4\epsilon/\sqrt{n} \\
&\le 4c\lambda_1/n^{(p-1)/2} && \text{by } \epsilon \le c\lambda_1/n^{(p-2)/2} \\
&\le 4c\lambda_1|v_1^\top u_0|^{p-1} && \text{by } |v_1^\top u_0| \ge 1/\sqrt{n}.
\end{aligned}$$

Thus, if we choose sufficiently small $c < 1/40$, using $|v_1^\top \widetilde{E}_{u_t}| \le \lambda_1|v_1^\top u_0|^{p-1}/10$ and $|v_1^\top u_0| \le |v_1^\top u_t|$, we can show

$$|v_1^\top \widetilde{E}_{u_t}| \le \lambda_1|v_1^\top u_0|^{p-1}/10 \le \lambda_1|v_1^\top u_t|^{p-1}/10$$

Thus, $A \le \frac{1}{1-1/11} = 1.1$ □

It remains to bound B. We give a proof by considering two cases separately, one is $|v_1^\top u_t| \le 1 - \frac{1}{c_0^2 p^2 k^2}$ and the other is $|v_1^\top u_t| > 1 - \frac{1}{c_0^2 p^2 k^2}$. If $|v_1^\top u_t| \le (1 - \frac{1}{c_0^2 p^2 k^2})$, we have

$$\begin{aligned}
B = & \frac{\|V^\top \widetilde{E}_{u_t}\|_2}{\lambda_1 |v_1^\top u_t|^{p-1}} \\
= & \frac{\sqrt{1-|v_1^\top u_t|^2}}{|v_1^\top u_t|} \cdot \frac{\|V^\top \widetilde{E}_{u_t}\|_2}{\lambda_1 |v_1^\top u_t|^{p-2}\sqrt{1-|v_1^\top u_t|^2}} \\
= & \tan\theta(v_1, u_t) \cdot \frac{\|V^\top \widetilde{E}_{u_t}\|_2}{\lambda_1 |v_1^\top u_t|^{p-2}\sqrt{1-|v_1^\top u_t|^2}} \\
\le & \tan\theta(v_1, u_t) \cdot \frac{\|\widetilde{E}_{u_t}\|_2}{\lambda_1 |v_1^\top u_t|^{p-2}\sqrt{1-|v_1^\top u_t|^2}} && \text{by } \|V^\top \widetilde{E}_{u_t}\|_2 \le \|\widetilde{E}_{u_t}\|_2 \\
\le & \tan\theta(v_1, u_t) \cdot \frac{c_0 pk \|\widetilde{E}_{u_t}\|_2}{\lambda_1 |v_1^\top u_t|^{p-2}} && \text{by } 1/\sqrt{1-|v_1^\top u_t|^2} \le c_0 pk
\end{aligned}$$

We need to bound $\frac{c_0 pk \|\widetilde{E}_{u_t}\|_2}{\lambda_1 |v_1^\top u_t|^{p-2}}$. On one side, we can show

$$\lambda_1 |v_1^\top u_t|^{p-2} \ge \lambda_1 / (n^{(p-2)/2}).$$

On the other side, by Part 1 of Lemma 22.9.12 and assumption on $\overline{E}$ and E, we have $\|\widetilde{E}_{u_t}\|_2$

$$c_0 pk \|\widetilde{E}_{u_t}\|_2 \le c_0 pk \cdot 4p\epsilon.$$

Thus, as long as we choose sufficiently small ϵ such that $\epsilon \le \lambda_1 / (n^{(p-2)/2} \cdot 40 \cdot c_0 p^2 k)$, then $B \le 0.1 \tan\theta(v_1, u_t)$.

If $|v_1^\top u_t| > 1 - \frac{1}{c_0^2 k^2 p^2}$, we have

$$\begin{aligned}
B &= \frac{\|V^\top \widetilde{E}_{u_t}\|_2}{\lambda_1 |v_1^\top u_t|^{p-1}} \\
&\leq \frac{\|V^\top \widetilde{E}_{u_t}\|_2}{\lambda_1 (1 - \frac{1}{c_0^2 p^2 k^2})^{p-1}} && \text{by } |v_1^\top u_t| > 1 - \frac{1}{c_0^2 p^2 k^2} \\
&\leq 3\|V^\top \widetilde{E}_{u_t}\|_2 / \lambda_1 && \text{by } 1/(1 - \frac{1}{c_0^2 p^2 k^2})^{p-1} \leq 3, \forall p \geq 3, k \geq 1, c_0 \geq 1 \\
&\leq 3\|\widetilde{E}_{u_t}\|_2 / \lambda_1
\end{aligned}$$

By Part 1 of Corollary 22.9.13, we have $\|\widehat{E}_{u_t}\|_2 \leq 4\epsilon/c_0$. By assumption on E and $\overline{E}$, we have $\|E_{u_t}\|_2 \leq \epsilon/c_0$ and $\|\overline{E}_{u_t}\|_2 \leq \epsilon/c_0$. Thus, we complete the proof $B \leq 18\epsilon/(c_0\lambda_1)$.

Part 2. For any $j \in [k]\backslash\{1\}$, we have lower bound $\frac{|v_1^\top u_{t+1}|}{|v_j^\top u_{t+1}|}$,

$$\begin{aligned}
&\frac{|v_1^\top u_{t+1}|}{|v_j^\top u_{t+1}|} \\
=\ & \frac{|v_1^\top [A^*(I, u_t, \cdots, u_t) + \widetilde{E}_{u_t}]|}{|v_j^\top [A^*(I, u_t, \cdots, u_t) + \widetilde{E}_{u_t}]|} && \text{by definition of } u_{t+1} \\
\geq\ & \frac{|v_1^\top A^*(I, u_t, \cdots, u_t)| - |v_1^\top \widetilde{E}_{u_t}|}{|v_j^\top A^*(I, u_t, \cdots, u_t)| + |v_j^\top \widetilde{E}_{u_t}|} && \text{by triangle inequality} \\
\geq\ & \frac{\lambda_1 |v_1^\top u_t|^{p-1} - |v_1^\top \widetilde{E}_{u_t}|}{\lambda_j |v_j^\top u_t|^{p-1} + |v_j^\top \widetilde{E}_{u_t}|} && \text{by Fact 22.9.3} \\
\geq\ & \frac{\lambda_1 |v_1^\top u_t|^{p-1} - \frac{1}{10}\lambda_1 |v_1^\top u_t|^{p-1}}{\lambda_j |v_j^\top u_t|^{p-1} + \frac{1}{10}\lambda_1 |v_1^\top u_t|^{p-1}} && \text{by } |v_j^\top \widetilde{E}_{u_t}| \leq \frac{1}{10}\lambda_1 |v_1^\top u_t|^{p-1} \\
\geq\ & \frac{\lambda_1 |v_1^\top u_t|^{p-1} - \frac{1}{10}\lambda_1 |v_1^\top u_t|^{p-1}}{\frac{1}{4}\lambda_1 |v_1^\top u_t|^{p-2} |v_j^\top u_t| + \frac{1}{10}\lambda_1 |v_1^\top u_t|^{p-1}} && \text{by Part 1. } \lambda_j |v_j^\top u_t|^{p-2} \leq \frac{1}{4}\lambda_1 |v_1^\top u_t|^{p-2} \\
=\ & \frac{\frac{9}{10}|v_1^\top u_t|}{\frac{1}{4}|v_j^\top u_t| + \frac{1}{10}|v_1^\top u_t|} && (22.15)
\end{aligned}$$

If $|v_j^\top u_t| < |v_1^\top u_t|$, then

$$\frac{\lambda_1 |v_1^\top u_{t+1}|^{p-2}}{\lambda_j |v_j^\top u_{t+1}|^{p-2}} \geq \frac{\lambda_1}{\lambda_j} 2^{p-2},$$

where the last step follows by $p \geq 4$. Note that for $p = 3$ we apply the better analysis like the proof of Lemma C.2 in [WA16]. Thus, it works for any $p \geq 3$.

If $|v_j^\top u_t| \geq |v_1^\top u_t|$, then

$$\frac{\lambda_1 |v_1^\top u_{t+1}|^{p-2}}{\lambda_j |v_j^\top u_{t+1}|^{p-2}} \geq \frac{\lambda_1}{\lambda_j} \frac{|v_1^\top u_t|^{p-2}}{|v_j^\top u_t|^{p-2}} 2^{p-2} \geq 4 \cdot 2^{p-2}.$$

Part 3. Similarly as Equation (22.15) and Part 1, we can also upper bound $\frac{|v_j^\top u_{t+1}|}{|v_1^\top u_{t+1}|}$,

$$\begin{aligned}
\frac{|v_j^\top u_{t+1}|}{|v_1^\top u_{t+1}|} &\leq \frac{\lambda_j |v_j^\top u_t|^{p-1} + |v_j^\top \widetilde{E}_{u_t}|}{\lambda_1 |v_1^\top u_t|^{p-1} - |v_1^\top \widetilde{E}_{u_t}|} \\
&= \frac{|v_j^\top u_t|}{|v_1^\top u_t|} \cdot \frac{\lambda_j |v_j^\top u_t|^{p-2} + |v_j^\top \widetilde{E}_{u_t}|/|v_j^\top u_t|}{\lambda_1 |v_1^\top u_t|^{p-2} - |v_1^\top \widetilde{E}_{u_t}|/|v_1^\top u_t|} \\
&= \frac{|v_j^\top u_t|}{|v_1^\top u_t|} \cdot \frac{\lambda_j |v_j^\top u_t|^{p-2}}{\lambda_1 |v_1^\top u_t|^{p-2} - |v_1^\top \widetilde{E}_{u_t}|/|v_1^\top u_t|} + \frac{|v_j^\top \epsilon_t|}{\lambda_1 |v_1^\top u_t|^{p-1} - |v_1^\top \widetilde{E}_{u_t}|} \\
&\leq \frac{|v_j^\top u_t|}{|v_1^\top u_t|} \cdot \frac{1}{4} \cdot \underbrace{\frac{1}{1 - |v_1^\top \widetilde{E}_{u_t}|/(\lambda_1 |v_1^\top u_t|^{p-1})}}_{A} + \underbrace{\frac{1}{1 - |v_1^\top \widetilde{E}_{u_t}|/(\lambda_1 |v_1^\top u_t|^{p-1})}}_{A} \cdot \underbrace{\frac{|v_j^\top \widetilde{E}_{u_t}|}{\lambda_1 |v_1^\top u_t|^{p-1}}}_{B}
\end{aligned}$$

Similarly as before, we can show $A \leq 1.1$ and if $|v_1^\top u_t| \geq 1 - \frac{1}{c_0^2 p^2 k^2}$. It remains to bound B. We give a proof by considering two cases separately, one is $|v_1^\top u_t| \leq 1 - \frac{1}{c_0^2 p^2 k^2}$ and the other

is $|v_1^\top u_t| > 1 - \frac{1}{c_0^2 p^2 k^2}$. If $|v_1^\top u_t| \le (1 - \frac{1}{c_0^2 p^2 k^2})$, we have

$$\begin{aligned}
B &= \frac{|v_j^\top \widetilde{E}_{u_t}|}{\lambda_1 |v_1^\top u_t|^{p-1}} \\
&= \frac{\sqrt{1 - |v_1^\top u_t|^2}}{|v_1^\top u_t|} \cdot \frac{|v_j^\top \widetilde{E}_{u_t}|}{\lambda_1 |v_1^\top u_t|^{p-2} \sqrt{1 - |v_1^\top u_t|^2}} \\
&= \tan\theta(v_1, u_t) \cdot \frac{|v_j^\top \widetilde{E}_{u_t}|}{\lambda_1 |v_1^\top u_t|^{p-2} \sqrt{1 - |v_1^\top u_t|^2}} \\
&\le \tan\theta(v_1, u_t) \cdot \frac{c_0 pk |v_j^\top \widetilde{E}_{u_t}|}{\lambda_1 |v_1^\top u_t|^{p-2}} && \text{by } 1/\sqrt{1 - |v_1^\top u_t|^2} \le c_0 pk
\end{aligned}$$

We need to bound $\frac{c_0 pk \|\widetilde{E}_{u_t}\|_2}{\lambda_1 |v_1^\top u_t|^{p-2}}$. On one side, we can show

$$\lambda_1 |v_1^\top u_t|^{p-2} \ge \lambda_1 / (n^{(p-2)/2}).$$

On the other side, by Part 2 of Lemma 22.9.12 and assumption on $\overline{E}$ and E, we have $|v_j^\top \widetilde{E}_{u_t}|$

$$c_0 pk |v_j^\top \widetilde{E}_{u_t}| \le c_0 pk \cdot 4\epsilon / \sqrt{n}.$$

Thus, as long as we choose sufficiently small ϵ such that $\epsilon \le \lambda_1 \sqrt{n} / (n^{(p-2)/2} \cdot 40 \cdot c_0 pk)$, then $B \le 0.1 \tan\theta(v_1, u_t)$.

If $|v_1^\top u_t| > 1 - \frac{1}{c_0^2 k^2 p^2}$, we have

$$\begin{aligned}
B &= \frac{|v_j^\top \widetilde{E}_{u_t}|}{\lambda_1 |v_1^\top u_t|^{p-1}} \\
&\le \frac{|v_j^\top \widetilde{E}_{u_t}|}{\lambda_1 (1 - \frac{1}{c_0^2 p^2 k^2})^{p-1}} && \text{by } |v_1^\top u_t| > 1 - \frac{1}{c_0^2 p^2 k^2} \\
&\le 3 |v_j^\top \widetilde{E}_{u_t}| / \lambda_1 && \text{by } 1/(1 - \frac{1}{c_0^2 p^2 k^2})^{p-1} \le 3, \forall p \ge 3, k \ge 1, c_0 \ge 1 \\
&\le 3 |v_j^\top \widetilde{E}_{u_t}| / \lambda_1
\end{aligned}$$

By Part 1 of Corollary 22.9.13, we have $|v_j^\top \widehat{E}_{u_t}| \le 4\epsilon/(c_0\sqrt{n})$. By assumption on E and $\overline{E}$, we have $|v_j^\top E_{u_t}| \le \epsilon/(c_0\sqrt{n})$ and $|v_j^\top \overline{E}_{u_t}| \le \epsilon/(c_0\sqrt{n})$. Thus, we complete the proof $B \le 18\epsilon/(c_0\lambda_1\sqrt{n})$. □

Definition 22.9.1. For any $\epsilon > 0$, we say $\{\widehat{\lambda}_i, \widehat{v}_i\}_{i=1}^k$ is ϵ-close to $\{\lambda_i, v_i\}_{i=1}^k$ if for all $i \in [k]$,

1. $|\widehat{\lambda}_i - \lambda_i| \le \epsilon$.
2. $\|\widehat{v}_i - v_i\|_2 \le \tan\theta(\widehat{v}_i, v_i) \le \min(\sqrt{2}, \epsilon/(\lambda_i))$.
3. $|\widehat{v}_i^\top v_j| \le \epsilon/(\sqrt{n}\lambda_i)$, $\forall j \in [k]\backslash[i]$.

Lemma 22.9.12. *Let $\widehat{E}_i = \lambda_i v_i^{\otimes p} - \widehat{\lambda}_i \widehat{v}_i^{\otimes p} \in \mathbb{R}^{n^p}, \forall i \in [k]$. For any $\epsilon > 0$, if $\{\widehat{\lambda}_i, \widehat{v}_i\}_{i=1}^k$ is ϵ-close to $\{\lambda_i, v_i\}_{i=1}^k$, then for all $r \in [k]$, for any unit vector $u \in \mathbb{R}^n$,*

1. $\left\|\sum_{i=1}^r \widehat{E}_i(I, u, \cdots, u)\right\|_2 \le 2p\epsilon\kappa^{1/2} + 2\phi\epsilon$.
2. $\forall j \in [k]\backslash[i], \left|\sum_{i=1}^r \widehat{E}_j(v_j, u, \cdots, u)\right| \le (2\kappa\epsilon + \phi\epsilon)/\sqrt{n}$.

where $\kappa = \sum_{i=1}^r |u^\top v_i|^2$ and $\phi = 2k(\epsilon/\lambda_k)^{p-1}$.

Proof. **Part 1.** For each $i \in [r]$, the error $\widehat{E}_i$ is

$$\widehat{E}_i(I, u, \cdots, u) = \lambda_i (u^\top v_i)^{p-1} v_i - \widehat{\lambda}_i (u^\top \widehat{v}_i)^{p-1} \widehat{v}_i$$

lives in $\mathrm{span}\{v_i, \widehat{v}_i\}$; this space is the same as $\mathrm{span}\{v_i, \widehat{v}_i^\perp\}$, where

$$\widehat{v}_i^\perp = \widehat{v}_i - (v_i^\top \widehat{v}_i) v_i$$

is the projection of $\widehat{v}_i$ onto the subspace orthogonal to v_i. Since $\|\widehat{v}_i - v\|_2^2 = 2(1 - v_i^\top \widehat{v}_i)$, it follows that

$$c_i = v_i^\top \widehat{v}_i = 1 - \|\widehat{v}_i - v_i\|_2^2/2 \geq 0$$

(the inequality follows from the assumption $\|\widehat{v}_i - v_i\|_2 \leq \sqrt{2}$, which in turn implies $0 \leq c_i \leq 1$). By the Pythagorean theorem and the above inequality for c_i,

$$\|\widehat{v}_i^\perp\|_2^2 = 1 - c_i^2 \leq \|\widehat{v}_i - v_i\|_2^2.$$

We can obtain this bound, for any $p \geq 3$,

$$|1 - c_i^p| = |1 - (1 - \|\widehat{v}_i - v_i\|_2^2/2)^p| \leq \frac{p}{2}\|\widehat{v}_i - v_i\|_2^2$$

We first define $a_i, b_i \in \mathbb{R}^n$ such that $a_i = u^\top v_i$ and $b_i = u^\top(\widehat{v}_i^\perp/\|\widehat{v}_i^\perp\|_2)$. We now rewrite $\widehat{E}_i(I, u, \cdots, u)$ in terms of the coordinate system defined by v_i and $\widehat{v}_i^\perp$,

$$\begin{aligned}
&\widehat{E}_i(I, u, \cdots, u) \\
=& \lambda_i (u^\top v_i)^{p-1} v_i - \widehat{\lambda}_i (u^\top \widehat{v}_i)^{p-1} \widehat{v}_i \\
=& \lambda_i a_i^{p-1} v_i - \widehat{\lambda}_i (a_i c_i + \|\widehat{v}_i^\perp b_i\|_2)^{p-1} (c_i v_i + \widehat{v}_i^\perp) \\
=& \underbrace{(\lambda_i a_i^{p-1} v_i - \widehat{\lambda}_i (a_i c_i + \|\widehat{v}_i^\perp\|_2 b_i)^{p-1} c_i)}_{A_i} \cdot v_i - \underbrace{\widehat{\lambda}_i (a_i c_i + \|\widehat{v}_i^\perp\|_2 b_i)^{p-1} \|\widehat{v}_i^\perp\|_2}_{B_i} \cdot \widehat{v}_i^\perp / \|\widehat{v}_i^\perp\|_2 \\
=& A_i \cdot v_i - B_i \cdot (\widehat{v}_i^\perp / \|\widehat{v}_i^\perp\|_2)
\end{aligned}$$

The overall error can be written as

$$\begin{aligned}
&\left\|\sum_{i=1}^{r}\widehat{E}_i(I,u,\cdots,u)\right\|_2^2 \\
&= \left\|\sum_{i=1}^{r}A_iv_i-\sum_{i=1}^{t}B_i(\widehat{v}_i^{\perp}/\|\widehat{v}_i^{\perp}\|_2)\right\|_2^2 \\
&\le 2\left\|\sum_{i=1}^{r}A_iv_i\right\|_2^2+2\left\|\sum_{i=1}^{r}B_i(\widehat{v}_i^{\perp}/\|\widehat{v}_i^{\perp}\|_2)\right\|_2^2 \\
&\le 2\sum_{i=1}^{r}A_i^2+2\left(\sum_{i=1}^{r}|B_i|\right)^2
\end{aligned}$$

We first try to bound A_i by using $|\lambda_i-\widehat{\lambda}_i|\le\epsilon$, $\|\widehat{v}_i-v_i\|_2\le\epsilon/\lambda_i$, $\|\widehat{v}_i^{\perp}\|_2^2\le\|\widehat{v}_i-v_i\|_2^2$, $0\le c_i\le 1$ and $|b_i|\le 1$,

$$\begin{aligned}
&|A_i| \\
&= |\lambda_i a_i^{p-1}-\widehat{\lambda}_i(a_ic_i+\|\widehat{v}_i^{\perp}\|_2b_i)^{p-1}c_i| \\
&\le |\lambda_i a_i^{p-1}-\widehat{\lambda}_ic_i^pa_i^{p-1}|+\sum_{j=1}^{p-1}\widehat{\lambda}_ic_i\binom{(p-1)}{j}|a_ic_i|^{(p-1)-j}\|\widehat{v}_i^{\perp}\|_2^j \\
&\le |\lambda_i a_i^{p-1}-\widehat{\lambda}_ia_i^{p-1}|+|(1-c_i^p)\widehat{\lambda}_ia_i^{p-1}|+\sum_{j=1}^{p-1}\widehat{\lambda}_ic_i\binom{(p-1)}{j}|a_ic_i|^{(p-1)-j}(\epsilon/\lambda_i)^j \\
&\le \epsilon|a_i|^{p-1}+\frac{p}{2}(\epsilon/\lambda_i)^2\widehat{\lambda}_i|a_i|^{p-1}+\sum_{j=1}^{p-1}\widehat{\lambda}_i\binom{(p-1)}{j}|a_i|^{(p-1)-j}(\epsilon/\lambda_i)^j
\end{aligned}$$

where the second term can be bounded as

$$\begin{aligned}
\frac{p}{2}(\epsilon/\lambda_i)^2\widehat{\lambda}_i|a_i|^{p-1} &\le \frac{p}{2}(\epsilon/\lambda_i)^2(|\widehat{\lambda}_i-\lambda_i|+|\lambda_i|)|a_i|^{p-1} \\
&\le \frac{p}{2}(\epsilon/\lambda_i)^2\epsilon|a_i|^{p-1}+\frac{p}{2}(\epsilon/\lambda_i)\cdot\epsilon|a_i|^{p-1} \\
&\le \frac{1}{100k}\epsilon|a_i|^{p-1} \qquad \text{by } \epsilon\ll\lambda_i/pk
\end{aligned}$$

The third term can be divided into two parts $j = 1, \cdots, (p-1)/2$ and $j = (p-1)/2, \cdots, (p-1)$. For the first part,

$$\begin{aligned}
&\sum_{j=1}^{(p-1)/2} \widehat{\lambda}_i \binom{(p-1)}{j} |a_i|^{(p-1)-j} (\epsilon/\lambda_i)^j \\
&= \widehat{\lambda}_i (p-1)|a_i|^{p-2}\epsilon/\lambda_i + \sum_{j=2}^{(p-1)/2} \widehat{\lambda}_i (p-1)^j |a_i|^{(p-1)/2} (\epsilon/\lambda_i)^j \\
&\leq (p-1)\epsilon|a_i|^{p-2} + \frac{1}{100k}\epsilon|a_i|^{(p-1)/2}
\end{aligned}$$

Similarly, we can bound the second part,

$$\begin{aligned}
&\sum_{j=(p-1)}^{(p-1)/2} \widehat{\lambda}_i \binom{(p-1)}{j} |a_i|^{(p-1)-j} (\epsilon/\lambda_i)^j \\
&= \widehat{\lambda}_i \cdot 1 \cdot 1 \cdot (\epsilon/\lambda_i)^{p-1} + \sum_{j=(p-1)-1}^{(p-1)/2} \widehat{\lambda}_i \binom{(p-1)}{j} |a_i|^{(p-1)-j} (\epsilon/\lambda_i)^j \\
&\leq \frac{1}{100k}\epsilon + \frac{1}{100k}\epsilon|a_i|
\end{aligned}$$

Thus, putting it all together, we get

$$|A_i| \leq \epsilon|a_i| + \frac{1}{10k}\epsilon|a_i| + (p-1)\epsilon|a_i| + \frac{1}{100k}\epsilon$$

which implies that

$$|A_i|^2 \leq 2\left((\epsilon|a_i|)^2 + (\frac{1}{10k}\epsilon|a_i|)^2 + ((p-1)\epsilon|a_i|)^2 + (\frac{1}{100k}\epsilon)^2\right)$$

It remains to bound B_i,

$$\begin{aligned}
|B_i| &= |\widehat{\lambda}_i \|\widehat{v}_i^{\perp}\|_2 (a_i c_i + \|\widehat{v}_i^{\perp}\|_2 b_i)^{p-1}| \\
&\leq \widehat{\lambda}_i \|\widehat{v}_i^{\perp}\|_2 \sum_{j=0}^{p-1} \binom{(p-1)}{j} |a_i c_i|^{p-1-j} \|\widehat{v}_i^{\perp}\|_2^j \\
&\leq \widehat{\lambda}_i (\epsilon/\lambda_i) \sum_{j=0}^{p-1} \binom{(p-1)}{j} |a_i|^{p-1-j} (\epsilon/\lambda_i)^j \\
&= \widehat{\lambda}_i (\epsilon/\lambda_i) |a_i|^{p-1} + \widehat{\lambda}_i (\epsilon/\lambda_i) \sum_{j=1}^{p-1} \binom{(p-1)}{j} |a_i|^{p-1-j} (\epsilon/\lambda_i)^j
\end{aligned}$$

where the first term can be bounded as

$$\widehat{\lambda}_i (\epsilon/\lambda_i) |a_i|^{p-1} \leq (\epsilon + \lambda_i)(\epsilon/\lambda_i)|a_i|^{p-1} \leq \epsilon |a_i|^{p-1} + \frac{1}{100k}\epsilon |a_i|^{p-1}$$

The second term can be divided into two parts $j = 1, \cdots, (p-1)/2$ and $j = (p-1)/2, \cdots, (p-1)$. For the first part,

$$\begin{aligned}
&(\epsilon/\lambda_i) \sum_{j=1}^{(p-1)/2} \widehat{\lambda}_i \binom{(p-1)}{j} |a_i|^{(p-1)-j} (\epsilon/\lambda_i)^j \\
&= (\epsilon/\lambda_i)\widehat{\lambda}_i (p-1)|a_i|^{p-2}\epsilon/\lambda_i + (\epsilon/\lambda_i) \sum_{j=2}^{(p-1)/2} \widehat{\lambda}_i (p-1)^j |a_i|^{(p-1)/2} (\epsilon/\lambda_i)^j \\
&\leq \frac{1}{100k}\epsilon |a_i|^{p-2} + \frac{1}{100k}\epsilon |a_i|^{(p-1)/2} \qquad \text{by } \epsilon \ll \lambda_i/(kp)
\end{aligned}$$

Similarly, we can bound the second part,

$$\begin{aligned}
&(\epsilon/\lambda_i) \sum_{j=(p-1)}^{(p-1)/2} \widehat{\lambda}_i \binom{(p-1)}{j} |a_i|^{(p-1)-j} (\epsilon/\lambda_i)^j \\
&= (\epsilon/\lambda_i)\widehat{\lambda}_i \cdot 1 \cdot 1 \cdot (\epsilon/\lambda_i)^{p-1} + (\epsilon/\lambda_i) \sum_{j=(p-1)-1}^{(p-1)/2} \widehat{\lambda}_i \binom{(p-1)}{j} |a_i|^{(p-1)-j} (\epsilon/\lambda_i)^j \\
&\leq \frac{\phi}{k}\epsilon + \frac{1}{100k}\epsilon |a_i| \qquad \text{by } \epsilon \ll \lambda_i/kp
\end{aligned}$$

where $\phi = 2k(\epsilon/\lambda_k)^{p-1}$. Putting it all together, we have

$$\begin{aligned}
& |B_i| \leq \epsilon |a_i|^2 + \frac{1}{100k^2}\epsilon |a_i| + \frac{\phi}{k}\epsilon \\
\implies & \sum_{i=1}^{r} |B_i| \leq \epsilon\kappa + \frac{1}{100k^2}\epsilon \sum_{i=1}^{t} |a_i| + \phi\epsilon \\
\implies & (\sum_{i=1}^{r} |B_i|)^2 \leq 2\left((\epsilon\kappa)^2 + (\frac{1}{100k}\epsilon \sum_{i=1}^{t} |a_i|)^2 + (\phi\epsilon)^2 \right) \\
\implies & (\sum_{i=1}^{r} |B_i|)^2 \leq 2\left((\epsilon\kappa)^2 + (\frac{1}{100k}\epsilon)^2 \kappa k + (\phi\epsilon)^2 \right)
\end{aligned}$$

Recall that $\kappa = \sum_{i=1}^{t} |a_i|^2 \leq 1$. Overall, we have

$$\begin{aligned}
& \left\| \sum_{i=1}^{r} \widehat{E}_i(I, u, \cdots, u) \right\|_2^2 \\
&= 2\sum_{i=1}^{r} |A_i|^2 + 2(\sum_{i=1}^{r} |B_i|)^2 \\
&\leq 4\left(\epsilon^2 \cdot \kappa + (\frac{1}{10k}\epsilon)^2 \kappa + (p-1)^2\epsilon^2\kappa + (\frac{1}{100\sqrt{k}}\epsilon)^2 \right) \\
&+ 4\left((\epsilon\kappa)^2 + (\frac{1}{100k}\epsilon)^2 \kappa k + (\phi\epsilon)^2 \right) \\
&\leq 4p^2\epsilon^2\kappa + 4\phi^2\epsilon^2
\end{aligned}$$

which gives the desired bound.

Part 2. For any $j \in [k] \backslash [i]$, we have

$$\begin{aligned}\left|\sum_{i=1}^{r} \widehat{E}_i(v_j, u, \cdots, u)\right| &\leq \sum_{i=1}^{r} |\widehat{E}_i(v_j, u, \cdots, u)| \\ &= \sum_{i=1}^{r} \widehat{\lambda}_i |\widehat{v}_i^\top u|^{p-1} |\widehat{v}_i^\top v_j| \\ &\leq \sum_{i=1}^{t} \widehat{\lambda}_i |\widehat{v}_i^\top u|^{p-1} \frac{\epsilon}{\sqrt{n}\lambda_i},\end{aligned}$$

where the first step follows by triangle inequality, the second step follows by $v_j \cdot v_i, \forall i \neq j$, and the last step follows by part 3 of definition of ϵ-close. Let's try to bound $\sum_{i=1}^{r} \frac{\widehat{\lambda}_i}{\lambda_i} |\widehat{v}_i^\top u|^{p-1}$,

$$\begin{aligned}&\sum_{i=1}^{r} \frac{\widehat{\lambda}_i}{\lambda_i} |\widehat{v}_i^\top u|^{p-1} \\ \leq &\sum_{i=1}^{r} \frac{\widehat{\lambda}_i}{\lambda_i} (|v_i^\top u| + |(v_i - \widehat{v}_i)^\top u|)^{p-1} \\ \leq &\sum_{i=1}^{r} \frac{\widehat{\lambda}_i}{\lambda_i} (|v_i^\top u| + \|v_i - \widehat{v}_i\|_2)^{p-1} \\ \leq &\sum_{i=1}^{r} (1 + \epsilon/\lambda_i)(|v_i^\top u| + \epsilon/\lambda_i)^{p-1} \\ \leq &\, 2\sum_{i=1}^{r} |v_i^\top u|^2 + 2k(\epsilon/\lambda_k)^{p-1} \\ = &\, 2\kappa + \phi.\end{aligned}$$

which gives the desired bound $(2\kappa + \phi)\epsilon/\sqrt{n}$.

□

Corollary 22.9.13. *Let $\widehat{E}_i = \lambda_i v_i^{\otimes p} - \widehat{\lambda}_i \widehat{v}_i^{\otimes p} \in \mathbb{R}^{n^p}, \forall i \in [k]$. For any $c_0 \geq 1$, for any $\epsilon \leq \lambda_k/(2c_0 k)$, if $\{\widehat{\lambda}_i, \widehat{v}_i\}_{i=1}^k$ is ϵ-close to $\{\lambda_i, v_i\}_{i=1}^k$, then for all $r \in [k]$, for any unit vector $u \in \mathbb{R}^n$, $|u^\top v_{r+1}| \geq 1 - \frac{1}{c_0^2 p^2 k}$, then*

1. $\left\|\sum_{i=1}^{r}\widehat{E}_i(I,u,\cdots,u)\right\|_2 \le 2p\epsilon\kappa^{1/2}+2\phi\epsilon \le 4\epsilon/c_0.$

2. $\forall j\in[k]\backslash[i], \left|\sum_{i=1}^{r}\widehat{E}_j(v_j,u,\cdots,u)\right| \le (2\kappa\epsilon+\phi\epsilon)/\sqrt{n} \le 4\epsilon/(c_0\sqrt{n}).$

where $\kappa=\sum_{i=1}^{r}|u^\top v_i|^2$ *and* $\phi=2k(\epsilon/\lambda_k)^{p-1}$.

Proof. Using Fact 22.9.1, we have for any $i\in[r]$,

$$\kappa=\sum_{i=1}^{r}|u^\top v_i|^2 \le k\cdot 1/(c_0^2p^2k) \le 1/c_0^2p^2.$$

which implies $2p\epsilon\sqrt{\kappa}\le 2/c_0$. We also can bound ϕ,

$$\phi=2k(\epsilon/\lambda_k)^{p-1} \le 2k(1/(2c_0k))^2 \le 1/c_0.$$

□

22.9.3 Main result

We first extend the original robust tensor power method analysis to general order $p \geq 3$, i.e., Theorem 22.3.4. Second, we combine Theorem 22.3.4 with our importance sampling techniques, i.e., Corollary 22.7.1 and Lemma 22.8.8.

Theorem 22.3.4. (Main, Abitrary order robust tensor power method) For any $p \geq 3$, $k \geq 1$, for any tensor $A = A^* + E \in \mathbb{R}^{n^p}$, where $A^* = \sum_{i=1}^k \lambda_i v_i^{\otimes p}$ with $\lambda_i > 0$ and orthonormal basis vectors $\{v_1, \ldots, v_k\} \subseteq \mathbb{R}^n$, $n \geq k$. Let λ_1, λ_k be the largest and smallest values in $\{\lambda_i\}_{i=1}^k$ and $\{\widehat{\lambda}_i, \widehat{v}_i\}_{i=1}^k$ be outputs of the robust tensor power method. For any sufficiently large constant $c_0 \geq 100$, there exists a sufficiently small constant $c > 0$, for any $\epsilon \in (0, c\lambda_k/(c_0 p^2 k n^{(p-2)/2})$ if E satisfies that

$$\|E\|_2 \leq \epsilon/(c_0\sqrt{n}) \tag{22.3}$$

and $T = \Omega(\log(\lambda_1 n/\epsilon)), L = \Omega(k\log(k))$, then with probability at least $9/10$, there exists a permutation $\pi : [k] \to [k]$, such that $\forall i \in [k]$,

$$|\lambda_i - \widehat{\lambda}_{\pi(i)}| \leq \epsilon, \quad \|v_i - \widehat{v}_{\pi(i)}\|_2 \leq \epsilon/\lambda_i. \tag{22.4}$$

Proof. We use $E \in \mathbb{R}^{n^p}$ to denote the original noise, use $\widehat{E}_i = \lambda_i v_i^{\otimes p} - \widehat{\lambda}_i \widehat{v}_i^{\otimes p} \in \mathbb{R}^{n^p}$ to denote the deflation noise, use $\overline{E} \in \mathbb{R}^{n^p}$ to denote the sketch noise and use $\widetilde{E}$ to denote the "true" noise that contains both original noise, deflation noise and sketch noise. Thus, in $t+1$ step, we actually observe tensor $A^* + \widetilde{E}$, where $\widetilde{E} = E + \sum_{i=1}^t \widehat{E}_i + \overline{E}$. To prove Theorem 22.3.4, in fact we don't need to worry about sketch noise $\overline{E}$. Instead of assuming it is $\overline{E} = 0$, we prove a strong theorem by assuming sketch noise $\overline{E}$ is bounded, i.e., $\|\overline{E}\|_2 \leq \epsilon/(c_0\sqrt{n})$.

Base case $i = 1$**.** Consider the first step, we get $\widehat{\lambda}_1 \in \mathbb{R}, \widehat{v}_1 \in \mathbb{R}^n$, we first show $\|\widehat{v}_1 - v_1\|_2$ is bounded(as Part 2 of Definition 22.9.1) and then show $|\widehat{\lambda}_1 - \lambda_1|$ is bounded(as Part 1 of Definition 22.9.1). At the end, we show $|\widehat{v}_i^\top v_j|$ is bounded(as Part 3 of Definition 22.9.1).

Bounding $|\widehat{v}_1 - v_1|$**.** So, applying Lemma 22.9.8, we have

$$\tan\theta(u_0, v_1) = \sin\theta(u_0, v_1)/\cos(\theta(u_0, v_1)) \leq \sqrt{n}.$$

Let t^* denote the situation where $|u_{t^*}^\top v_1| = 1 - \frac{1}{c_0^2 p^2 k^2}$. We know

$$\|u_{t^*} - v_1\|_2^2 = 2 - 2|u_{t^*}^\top v_1| = 2/(c_0^2 p^2 k^2).$$

We can upper bound

$$\|u_{t^*} - v_1\|_2 \leq \tan\theta(u_{t^*}, v_1) \leq 0.8 \tan\theta(u_{t^*-1}, v_1) \leq \cdots \leq 0.8^{t^*} \tan\theta(u_0, v_1) \leq 0.8^{t^*}\sqrt{n}$$

Thus, we need to set $t^* = \Omega(\log(nkpc_0)) = \Omega(\log c_0 n)$.

Now, we need to analyze $\|u_T - v_1\|_2$,

$$\|u_T - v_1\|_2 \leq 0.8(\tan\theta(u_T, v_1) + 18\epsilon/(c_0\lambda_1)) \leq \cdots \leq 0.8^{T-t^*} \tan(u_{t^*}, v_1) + 5 \cdot 18\epsilon/(c_0\lambda_1)$$

To guarantee $\|u_T - v_1\|_2 \leq \epsilon/\lambda_1$, we need to choose $T - t^* = \Omega(n\lambda_1/\epsilon)$ and $c_0 \geq 100$. Thus, we get the desired property in Part 2 of Definition 22.9.1.

Bounding $|\widehat{\lambda}_1 - \lambda_1|$. It remains to bound $|\widehat{\lambda}_1 - \lambda_1|$.

$$
\begin{aligned}
|\widehat{\lambda}_1 - \lambda_1| &= |[A^* + \widetilde{E}](\widehat{v}_1, \cdots, \widehat{v}_1) - \lambda_1| \\
&\le |\widetilde{E}(\widehat{v}_1, \cdots, \widehat{v}_1)| + |A^*(\widehat{v}_1, \cdots, \widehat{v}_1) - \lambda_1| && \text{by triangle inequality} \\
&= |\widetilde{E}(\widehat{v}_1, \cdots, \widehat{v}_1)| + \left| \left[\sum_{i=1}^k \lambda_i v_i^{\otimes p} \right] (\widehat{v}_1, \cdots, \widehat{v}_1) - \lambda_1 \right| && \text{by } A^* = \sum_{i=1}^k \lambda_i v_i^{\otimes p} \\
&\le \underbrace{|\widetilde{E}(\widehat{v}_1, \cdots, \widehat{v}_1)|}_{A} + \underbrace{|\lambda_1 |v_1^\top \widehat{v}_1|^p - \lambda_1|}_{B} + \underbrace{\sum_{j=2}^k \lambda_j |v_j^\top \widehat{v}_1|^p}_{C}. && \text{by triangle inequality}
\end{aligned}
$$

For the term A, we have

$$
\begin{aligned}
A &= |\widetilde{E}(\widehat{v}_1, \cdots, \widehat{v}_1)| \\
&\le |E(\widehat{v}_1, \cdots, \widehat{v}_1)\| + |\overline{E}(\widehat{v}_1, \cdots, \widehat{v}_1)| \\
&\le \|E\|_2 + |\overline{E}(\widehat{v}_1, \cdots, \widehat{v}_1)| \\
&\le \epsilon/(c_0\sqrt{n}) + \epsilon/(c_0\sqrt{n}) \\
&\le \epsilon/12 && \text{by } c_0 \ge 100, n \ge 1
\end{aligned}
$$

It remains to upper bound the second term B and the third term C.

$$
\begin{aligned}
B &= \lambda_1 - \lambda_1 (1 - \frac{1}{2}\|v_1 - \widehat{v}_1\|_2^2)^p && \text{by } v_1^\top \widehat{v}_1 = 1 - \frac{1}{2}\|v_1 - \widehat{v}_1\|_2^2 \\
&\le \lambda_1 p \frac{1}{2}\|v_1 - \widehat{v}_1\|_2^2 && \text{by } \|v_1 - \widehat{v}_1\|_2^2 \ll 1 \\
&\le p\epsilon^2/(2\lambda_1) && \text{by } \|v_1 - \widehat{v}_1\|_2 \le \epsilon/\lambda_1 \\
&\le \epsilon/12. && \text{by } p\epsilon/(2\lambda_1) \le 1/12
\end{aligned}
$$

For the term C,

$$
C = \sum_{j=2}^k \lambda_j |v_j^\top \widehat{v}_1|^p \le \sum_{j=2}^k \lambda_j (\epsilon/(\sqrt{n}\lambda_j))^p = \epsilon \sum_{j=2}^k (\epsilon/(\lambda_j\sqrt{n}))^{p-1} \le \epsilon/4.
$$

where the first inequality follows by Part 3 of Definition 22.9.1, and the last step follows by $(\epsilon/\lambda_k)^{p-1} \leq 1/(4k)$. It suffices to set $\epsilon < \frac{1}{4}k^{1/(p-1)}\lambda_k$ Thus, putting it all together, we have

$$|\widehat{\lambda}_1 - \lambda_1| \leq A + B + C \leq \epsilon.$$

Bounding $|\widehat{v}_1^\top v_j|$. We consider any $j \in \{2, \cdots, k\}$. We choose t^* to be smallest integer such that $|v_1^\top u_{t^*}| \geq 1 - \frac{1}{c_0^2 p^2 k^2}$. This implies $|v_j^\top u_{t^*}| \leq \frac{1}{c_0 pk}$. By Part 3 of Lemma 22.9.9, we have

$$|v_j^\top u_{t^*}|/|v_1^\top u_{t^*}| \leq \ 0.8|v_j^\top u_{t^*-1}|/|v_1^\top u_{t^*-1}| \leq \cdots \leq 0.8^{t^*} \cdot u_0 \leq 0.8^{t^*} \cdot 1/(1/\sqrt{n})$$

Thus, we need to choose $t^* = \Omega(\log c_0 n)$.

Let analyze $T > t^*$,

$$|v_j^\top u_T|/|v_1^\top u_T| \leq 0.8^{T-t^*}|v_j^\top u_{t^*}|/|v_1^\top u_{t^*}| + 5 \cdot 18\epsilon/(c_0\lambda_1\sqrt{n}).$$

Thus, we need to choose $T = \Omega(\log(n\lambda_1/\epsilon))$ and $c_0 \geq 100$ to guarantee that $|v_j^\top u_T| \leq \epsilon/(\lambda_1\sqrt{n})$.

The case for $i = r+1$. Conditioned on all the first t cases are holding, let's consider $t+1$ case, then the "true" noise $\widetilde{E} = E + \sum_{i=1}^r E_i + \overline{E} \in \mathbb{R}^{n^p}$. Similarly as the base case $t = 1$, we first show how to bound $\|\widehat{v}_{r+1} - v_{r+1}\|_2$(as Part 2 of Definition 22.9.1), and then we show how to bound $|\widehat{\lambda}_{r+1} - \lambda_{r+1}|$(as Part 1 of Definition 22.9.1), at the end we show how to bound $|v_{r+1}^\top v_j|$(as Part 3 of Definition 22.9.1).

Bounding $\|\widehat{v}_{r+1} - v_{r+1}\|_2$. The proof is identical to the Base case, the difference is, we need to choose $T = \Omega(\log(n\lambda_{t+1}/\epsilon))$.

Bounding $|\widehat{\lambda}_{r+1} - \lambda_{r+1}|$. We redefine A^* and $\widetilde{E}$ to be $A^* = \sum_{i=t+1}^{k} \lambda_i v_i^{\otimes p}$ and $\widetilde{E} = E + \overline{E} + \sum_{i=1}^{t} \widehat{E}_i$. Then rewrite $|\widehat{\lambda}_{t+1} - \lambda_{t+1}|$ as follows

$$
\begin{aligned}
& |\widehat{\lambda}_{r+1} - \lambda_{r+1}| \\
= & |[A^* + \widetilde{E}](\widehat{v}_{r+1}, \cdots, \widehat{v}_{r+1}) - \lambda_{r+1}| \\
\leq & |\widetilde{E}(\widehat{v}_{r+1}, \cdots, \widehat{v}_{r+1})| + |A^*(\widehat{v}_{r+1}, \cdots, \widehat{v}_{r+1}) - \lambda_{r+1}| && \text{by triangle inequality} \\
= & |\widetilde{E}(\widehat{v}_{r+1}, \cdots, \widehat{v}_{r+1})| + \left| \left[\sum_{i=r+1}^{k} \lambda_i v_i^{\otimes p} \right] (\widehat{v}_{r+1}, \cdots, \widehat{v}_{r+1}) - \lambda_{r+1} \right| && \text{by } A^* = \sum_{i=r+1}^{k} \lambda_i v_i^{\otimes p} \\
\leq & \underbrace{|\widetilde{E}(\widehat{v}_{r+1}, \cdots, \widehat{v}_{r+1})|}_{A} + \underbrace{|\lambda_{r+1} |v_{r+1}^\top \widehat{v}_{r+1}|^p - \lambda_{r+1}|}_{B} + \underbrace{\sum_{j=r+2}^{k} \lambda_j |v_j^\top \widehat{v}_{r+1}|^p}_{C}. && \text{by triangle inequality}
\end{aligned}
$$

We need to analyze A,

$$
\begin{aligned}
A = & |\widetilde{E}(\widehat{v}_{r+1}, \cdots, \widehat{v}_{r+1})| \\
= & |E(\widehat{v}_{r+1}, \cdots, \widehat{v}_{r+1})| + |\overline{E}(\widehat{v}_{r+1}, \cdots, \widehat{v}_{r+1})| + |\sum_{i=1}^{r} \widehat{E}_i(\widehat{v}_{r+1}, \cdots, \widehat{v}_{r+1})| \\
\leq & \epsilon/(c_0\sqrt{n}) + \epsilon/(c_0\sqrt{n}) + 4\epsilon/(c_0\sqrt{n}) \\
\leq & \epsilon/12. && \text{by } c_0 \geq 100, n \geq 1
\end{aligned}
$$

Bounding $|\widehat{v}_{r+1}^\top v_j|$. We consider any $j \in \{r+2, \cdots, k\}$. The proof is identical to the base. □

Theorem 22.9.14 (Arbitrary even order sublinear time robust tensor power method)**.** *There exists some sufficiently small constant $\gamma \in (0,1)$ such that any $p \in [3, n^\gamma]$, $k \in [1, n^\gamma]$. There exists some sufficiently small constant $\alpha \in (0, 1 - 20\gamma)$ such that for any tensor $A = A^* + E \in \mathbb{R}^{n^p}$, where $A^* = \sum_{i=1}^{k} \lambda_i v_i^{\otimes p}$ with $\lambda_i > 0$ and orthonormal basis vectors*

$\{v_1, \ldots, v_k\} \subseteq \mathbb{R}^n$. *Let* λ_1, λ_k $(\geq 1/n^\gamma)$ *be the largest and smallest values in* $\{\lambda_i\}_{i=1}^k$ *and* $\{\widehat{\lambda}_i, \widehat{v}_i\}_{i=1}^k$ *be outputs of the "importance sampling" robust tensor power method. For any sufficiently large constant* $c_0 \geq 100$*, there exists a sufficiently small constant* $c > 0$*, for any* $\epsilon \in (0, c\lambda_k/(c_0 p^2 k n^{(p-2)/2})$ *if* E *satisfies that* $\|E\|_2 \leq \epsilon/(c_0\sqrt{n})$ *and if* $T = \Omega(\log(\lambda_1 n/\epsilon)), L = \Omega(k\log(k)), m = \Omega(TLk^2), b = \Omega(mnc_0^2/\epsilon^2)$ *then our algorithm uses* $O(nk)$ *spaces, runs in* $O(n^{p-\alpha})$ *time and then with probability at least* $9/10$*, there exists a permutation* $\pi : [k] \to [k]$*, such that* $\forall i \in [k]$*,*

$$|\lambda_i - \widehat{\lambda}_{\pi(i)}| \leq \epsilon, \qquad \|v_i - \widehat{v}_{\pi(i)}\|_2 \leq \epsilon/\lambda_i.$$

Proof. It follows by combining Theorem 22.3.4, Corollary 22.7.1 and Theorem 22.8.9 . The total number of iterations is $m = \Omega(TLk^2)$, thus we need to take the union bound over all the iterations. The total running time is $O(mb) = n^{p-\alpha'}(\log \lambda_1 n)^2/\lambda_k^2$, where α' is some sufficiently small constant. □

22.10 Algorithm

Algorithm 22.3 Subroutine for approximate tensor contraction $A(I, v, w)$

function ApproxTIvw($A, v, w, n, B, \{\widehat{b}_i\}$)
  $\widetilde{q}, \widetilde{r} \leftarrow$ CumProb(v, w)
  for $d = 1 \to B$ **do**
    $\mathcal{L} \leftarrow$ GenRandTuples($\sum_{i=1}^{n} \widehat{b}_i, \widetilde{q}, \widetilde{r}$)
    for $i = 1 \to n$ **do**
      $s_i^{(d)} \leftarrow 0$
      for $\ell = 1 \to \widehat{b}_i$ **do**
        $(j, k) \leftarrow \mathcal{L}_{(i-1)b+\ell}$
        $s_i^{(d)} \leftarrow s_i^{(d)} + \frac{1}{q_j r_k} A_{i,j,k} \cdot u_j \cdot u_k$
      end for
    end for
  end for
  $\widehat{A}(I, v, w)_i \leftarrow \underset{d \in [B]}{\text{median}}\ s_i^{(d)} / \widehat{b}_i,\ \forall i \in [n]$
  return $\widehat{A}(I, v, w)$

Algorithm 22.4 Subroutine for approximate tensor contraction $A(u, v, w)$

function ApproxTuvw($A, u, v, w, n, B, \widehat{b}$)
  $\widetilde{p}, \widetilde{q}, \widetilde{r} \leftarrow$ CumProb(u, v, w)
  for $d = 1 \to B$ **do**
    $\mathcal{L} \leftarrow$ GenRandTuples($\widehat{b}, \widetilde{p}, \widetilde{q}, \widetilde{r}$).
    $s^{(d)} \leftarrow 0$
    for $(i, j, k) \in \mathcal{L}$ **do**
      $s^{(d)} \leftarrow s^{(d)} + \frac{1}{p_i q_j r_k} A_{i,j,k} \cdot u_i \cdot u_j \cdot u_k$
    end for
    $s^{(d)} \leftarrow s^{(d)} / \widehat{b}$
  end for
  $\widehat{A}(u, v, w) \leftarrow \underset{d \in [B]}{\text{median}}\ s^{(d)}$
  return $\widehat{A}(u, v, w)$

Algorithm 22.5 Our main algorithm

function ImportanceSamplingRobustTensorPowerMethod(A, n, B, b)
if s_i are known, where $\|A_{i,*,*}\|_F^2 \lesssim s_i$ **then**
    $\widehat{b}_i \leftarrow b \cdot s_i / \|A\|_F^2, \forall i \in [n]$
else
    $\widehat{b}_i \leftarrow b/n, \forall i \in [n]$
end if
$\widehat{b} = \sum_{i=1}^n \widehat{b}_i$
for $\ell = 1 \to L$ **do**
    for $t = 1 \to T$ **do**
        $u^{(\ell)} \leftarrow \text{ApproxTIvw}(A, u^{(\ell)}, u^{(\ell)}, n, B, \{\widehat{b}_i\})$
        $u^{(\ell)} \leftarrow u^{(\ell)} / \|u^{(\ell)}\|_2$
    end for
    $\lambda^{(\ell)} \leftarrow \text{ApproxTuvw}(A, u^{(\ell)}, u^{(\ell)}, u^{(\ell)}, n, B, \widehat{b})$
end for
$\ell^* \leftarrow \arg\max_{\ell \in [L]} \lambda^{(\ell)}$
$u^* \leftarrow u^{(\ell^*)}$
for $t = 1 \to T$ **do**
    $u^* \leftarrow \text{ApproxTIvw}(A, u^*, u^*, n, B, \{\widehat{b}_i\})$
    $u^* \leftarrow u^* / \|u^*\|_2$
end for
$\lambda^* \leftarrow \text{ApproxTuvw}(A, u^*, u^*, u^*, n, B, \widehat{b})$
return λ^*, u^*

22.11 Additional Experiment Results

The following sections gather all of our experiment results we collected to compare our sampling-based robust tensor power method to a sketching-based robust tensor power method. Our algorithm has the following parameters to set:

1. T: Number of power iterations
2. L: The number of starting vectors of the robust tensor power method
3. B: The number of sketches used in sketching, or the number of repetitions of sampling
4. b: The size of the sketch, or the number of indices sampled

In all of our experiments, we fix $T = 30$, $L = 50$, and change B and b for each input tensor. We observe the squared residual Frobenius norm to evaluate the performance of each algorithm. We only compute the first eigenvalue and eigenvector (rank-1 recovery) of each tensor. This enables us to run a large collection of tensors within a reasonable time.

22.11.1 Synthetic tensors with inverse decaying eigenvalues

The tables shown in this section use synthetic tensors generated with eigenvalues decaying as $\lambda_i = \frac{1}{i}$. All tensors have rank $k = 100$ and added noise with $\sigma = 0.01$.

Sketching based robust power method										
	Squared residual norm				**Running time (s)**				**Preprocessing time (s)**	
b \ B	10	30	50		10	30	50		10	30	50
2^{10}	1.011	0.9263	0.6358		0.6034	2.407	4.382		5.386	15.96	26.47
2^{12}	1.01	0.4912	0.4451		1.342	4.603	8.012		5.972	17.34	28.72
2^{14}	0.4394	0.4161	0.4085		4.921	15.64	27.93		8.877	24.81	40.87
2^{16}	0.4089	0.4029	0.4006		22.48	69.67	115.2		13.79	34.86	55.65

Table 22.3: Sketching based robust power method: $n = \mathbf{1200}$, **inverse** decay, $\|\mathbf{T}\|_F^2$ =1.01

Importance sampling based robust power method (without prescanning)										
	Squared residual norm				**Running time (s)**				**Preprocessing time (s)**	
b \ B	10	30	50		10	30	50		10	30	50
$5n$	0.4075	0.4021	0.4		2.611	8.325	16.34		0.0	0.0	0.0
$10n$	0.4031	0.4007	0.3997		4.764	13.79	27.56		0.0	0.0	0.0
$20n$	0.4046	0.3997	0.3995		8.519	24.61	49.8		0.0	0.0	0.0
$30n$	0.3999	0.3995	0.399		12.32	35.52	71.59		0.0	0.0	0.0
$40n$	0.4009	0.3994	0.3989		16.15	46.25	94.76		0.0	0.0	0.0

Table 22.4: Importance sampling based robust power method, without prescanning: $n = \mathbf{1200}$, **inverse** decay, $\|\mathbf{T}\|_F^2$ =1.01

Importance sampling based robust power method (with prescanning)											
	Squared residual norm				**Running time (s)**				**Preprocessing time (s)**		
b \ B	10	30	50		10	30	50		10	30	50
$5n$	0.408	0.4025	0.4005		3.621	11.9	20.49		2.234	2.235	2.241
$10n$	0.4051	0.4003	0.3997		6.302	20.56	35.24		2.225	2.223	2.226
$20n$	0.4057	0.3991	0.3993		11.67	37.89	64.52		2.225	2.226	2.225
$30n$	0.4027	0.3993	0.399		16.99	54.83	93.14		2.225	2.225	2.226
$40n$	0.4024	0.399	0.3988		22.23	72.13	122.2		2.223	2.225	2.225

Table 22.5: Importance sampling based robust power method with prescanning: $n = \mathbf{1200}$, **inverse** decay, $\|\mathbf{T}\|_F^2$ =1.01

Algorithm 22.2 Sampling Procedure for Rank-2

procedure SAMPLINGRANK2(A, n, b, α) ▷ Lemma 22.8.5
  $\mathcal{S} \leftarrow S_1 \leftarrow S_2 \leftarrow S_3 \leftarrow S_4 \leftarrow \emptyset$
  for $i \in [n]$ **do**
    if $|A_{i,i,i}| \geq 3/n$ **then**
      $S_1 \leftarrow S_1 \cup i$
    else
      $S_2 \leftarrow S_2 \cup i$
    end if
  end for ▷ $|S_1| = O(n^{1/3})$
  for $i \in S_1$ **do** ▷ **Part 1.**
    Choose b i.i.d. samples from slice i, add to $\mathcal{S}$
  end for ▷ # samples $= O(bn^{1/3})$
  for $i \in [n]$ **do** ▷ **Part 2.**
    Choose $O(b/n^{\alpha})$ i.i.d. samples from slice i, add to $\mathcal{S}$
  end for ▷ # samples $= O(bn^{1-\alpha})$
  Randomly choose a set $S_3 \subseteq S_2$ such that $|S_3| = O(n^{1/3+\alpha} \log n)$ ▷ **Part 3.**
  for $i \in S_3$ **do**
    for $l \in [n]$ **do**
      if $|A_{i,l,l}|^2 \geq 1/(4n^{1+3\alpha})$ **then** ▷ # $l = O(n^{1/3+3\alpha})$
        Choose b i.i.d. samples from slice l, add to $\mathcal{S}$
      end if
    end for
  end for ▷ # samples $= O(bn^{2/3+4\alpha})$
  for $i \in [n]$ **do** ▷ **Part 4.**
    for $j \in [n]$ **do**
      if $|A_{i,j,j}|^2 \geq 1/(4n^{2/3+2\alpha})$ **then**
        $S_4 \leftarrow S_4 \cup \{i, j\}$
      end if
    end for
  end for
  for $i \in S_4$ **do** ▷ $|S_4| = O(n^{2/3+2\alpha})$
    Choose b i.i.d samples from slice i, add to $\mathcal{S}$
  end for ▷ # samples $= O(bn^{2/3+2\alpha})$
  return $\mathcal{S}$
end procedure

22.11.2 Synthetic tensors with inverse square decaying eigenvalues

The tables shown in this section use synthetic tensors generated with eigenvalues decaying as $\lambda_i = \frac{1}{i^2}$. All tensors have rank $k = 100$ and added noise with $\sigma = 0.01$.

Sketching based robust power method										
	Squared residual norm				**Running time (s)**				**Preprocessing time (s)**	
b \ B	10	30	50		10	30	50		10	30	50
2^{10}	1.01	1.014	0.5437		0.6114	2.423	4.374		5.361	15.85	26.08
2^{12}	1.02	0.2271	0.1549		1.344	4.563	8.022		5.978	17.23	28.31
2^{14}	0.1513	0.1097	0.1003		4.928	15.51	27.87		8.788	24.72	40.4
2^{16}	0.1065	0.09242	0.08936		22.28	69.7	116.3		13.76	34.74	55.34

Table 22.6: Sketching based robust power method: $n = \mathbf{1200}$, **inverse-square** decay, $\|\mathbf{T}\|_F^2$ =1.01

Importance sampling based robust power method (without prescanning)										
	Squared residual norm				**Running time (s)**				**Preprocessing time (s)**	
b \ B	10	30	50		10	30	50		10	30	50
$5n$	0.08684	0.08637	0.08639		2.595	8.3	15.46		0.0	0.0	0.0
$10n$	0.08784	0.08671	0.08627		4.42	13.68	25.84		0.0	0.0	0.0
$20n$	0.08704	0.087	0.08618		8.02	24.51	46.37		0.0	0.0	0.0
$30n$	0.08697	0.08645	0.08625		11.63	35.35	66.71		0.0	0.0	0.0
$40n$	0.08653	0.08664	0.08611		15.19	46.12	87.24		0.0	0.0	0.0

Table 22.7: Importance sampling based robust power method, without prescanning: $n = \mathbf{1200}$, **inverse-square** decay, $\|\mathbf{T}\|_F^2$ =1.01

Importance sampling based robust power method (with prescanning)										
	Squared residual norm				**Running time (s)**				**Preprocessing time (s)**	
b \ B	10	30	50		10	30	50		10	30	50
$5n$	0.08657	0.08684	0.08636		3.1	10.47	18		2.234	2.236	2.234
$10n$	0.08741	0.08677	0.08668		5.427	17.43	30.26		2.232	2.233	2.233
$20n$	0.08648	0.08624	0.08634		9.843	31.42	54.49		2.226	2.226	2.226
$30n$	0.08635	0.08634	0.08615		14.33	45.4	63.85		2.226	2.224	2.227
$40n$	0.08622	0.08652	0.08619		18.68	59.32	82.83		2.225	2.225	2.225

Table 22.8: Importance sampling based robust power method with prescanning: $n = \mathbf{1200}$, **inverse-square** decay, $\|\mathbf{T}\|_F^2$ =1.01

22.11.3 Synthetic tensors with linearly decaying eigenvalues

The tables shown in this section use synthetic tensors generated with eigenvalues decaying as $\lambda_i = 1 - \frac{i-1}{k}$. All tensors have rank $k = 100$ and added noise $\sigma = 0.01$. Due to the slowly decaying eigenvalues, we expect to see large residual values of rank-1 decomposition, and both algorithms have to work harder to get a better recovery. Our sampling based method works better especially when the dimension is large.

Sketching based robust power method										
	Squared residual norm				**Running time (s)**				**Preprocessing time (s)**	
b \ B	10	30	50		10	30	50		10	30	50
2^{10}	1.01	1.01	1.01		0.6153	2.457	4.381		5.364	15.78	26.25
2^{12}	1.01	1.01	1.01		1.356	4.704	7.994		5.975	17.24	28.48
2^{14}	1.01	1.01	0.9829		4.942	15.84	27.82		8.851	24.62	40.5
2^{16}	0.9835	0.9813	0.9811		22.66	72.01	115.1		13.75	35.13	55.59

Table 22.9: Sketching based robust power method: $n = \mathbf{1200}$, **linear** decay, $\|\mathbf{T}\|_F^2$ =1.01

Importance sampling based robust power method (without prescanning)										
	Squared residual norm				**Running time (s)**				**Preprocessing time (s)**	
b \ B	10	30	50		10	30	50		10	30	50
$5n$	1.01	0.9884	0.9879		2.627	8.699	17.61		0.0	0.0	0.0
$10n$	1.005	0.9881	0.9865		4.419	14.43	29.98		0.0	0.0	0.0
$20n$	0.9924	0.9853	0.9847		8.05	25.88	53.99		0.0	0.0	0.0
$30n$	0.9899	0.984	0.982		11.67	37.32	77.92		0.0	0.0	0.0
$40n$	0.9878	0.9831	0.9822		15.26	48.55	102		0.0	0.0	0.0

Table 22.10: Importance sampling based robust power method, without prescanning: $n = \mathbf{1200}$, **linear** decay, $\|\mathbf{T}\|_F^2$ =1.01

Importance sampling based robust power method (with prescanning)										
	Squared residual norm				**Running time (s)**				**Preprocessing time (s)**	
b \ B	10	30	50		10	30	50		10	30	50
$5n$	1.01	0.991	0.9882		3.668	11.64	19.48		2.234	2.234	2.234
$10n$	1.009	0.9883	0.9851		6.512	20.14	33.45		2.225	2.225	2.226
$20n$	0.9918	0.9852	0.9828		11.9	36.43	61.01		2.225	2.225	2.226
$30n$	0.9873	0.9849	0.9826		17.14	52.89	87.36		2.226	2.226	2.226
$40n$	0.9882	0.9829	0.9822		22.38	69.01	114.6		2.226	2.227	2.226

Table 22.11: Importance sampling based robust power method with prescanning: $n = \mathbf{1200}$, **linear** decay, $\|\mathbf{T}\|_F^2$ =1.01

22.11.4 Results from real-life datasets

Dataset	# Documents	# Vocabulary	# Words	Short Description
Wiki	114274	10000	58508288	Wikipedia articles, preprocessed by authors of [WTSA15]
Enron [3]	186501	10000	16971591	Enron Email Dataset, preprocessed by authors of [WTSA15]
AP [4]	2246	10473	435838	Associated Press [BNJ01]
NIPS [5]	1500	12419	1900000	NIPS full papers, preprocessed by UCI
KOS [6]	3430	6906	467714	KOS blog, preprocessed by UCI
NSF [7]	49078	22170	4406190	NSF research award abstracts 1990-2003

Table 22.12: List of six real-life datasets

We used the two same datasets as the previous work [WTSA15]: Wiki and Enron, as well as four additional real-life datasets, shown in Table 22.12. We use the program from the authors of [TWZS15] to build a $K \times K \times K$ tensor A_{LDA} for each dataset. Due to limited time our experiments only use $K = 200$. We expect to gain more over the sketching-based method on larger tensors.

Since these tensors comes from real-world applications and are not normalized, their Frobenius norm is usually very large.

[3] http://www.cs.cmu.edu/~enron/
[4] http://www.cs.princeton.edu/~blei/lda-c/
[5] https://archive.ics.uci.edu/ml/machine-learning-databases/bag-of-words
[6] https://archive.ics.uci.edu/ml/machine-learning-databases/bag-of-words
[7] http://archive.ics.uci.edu/ml/datasets/NSF+Research+Award+Abstracts+1990-2003

Sketching based robust power method: dataset **Enron**								
	Squared residual norm			**Running time (s)**			**Preprocessing time (s)**	
b \ B	10	30		10	30		10	30
2^{10}	4.733e+07	4.608e+07		0.3667	0.7942		0.2429	0.2378
2^{11}	4.677e+07	4.58e+07		0.4434	1.486		0.2468	0.3238
2^{12}	4.581e+07	4.559e+07		0.995	3.243		0.3647	0.6283
2^{13}	4.568e+07	4.546e+07		2.023	6.017		0.5745	0.6458
2^{14}	4.556e+07	4.542e+07		4.561	13.97		0.9698	1.061

Table 22.13: Sketching based robust power method: dataset **Enron**, $\|\mathbf{T}\|_F^2$ =4.96e+07

Importance sampling based robust power method (without prescanning): dataset **Enron**								
	Squared residual norm			**Running time (s)**			**Preprocessing time (s)**	
b \ B	10	30		10	30		10	30
$5n$	4.579e+07	4.557e+07		0.3784	1.139		0.0	0.0
$10n$	4.563e+07	4.559e+07		0.5689	1.603		0.0	0.0
$20n$	4.56e+07	4.545e+07		1.003	2.706		0.0	0.0
$30n$	4.555e+07	4.543e+07		1.362	3.717		0.0	0.0
$40n$	4.564e+07	4.544e+07		1.693	4.753		0.0	0.0

Table 22.14: Importance sampling based robust power method, without prescanning: dataset **Enron**, $\|\mathbf{T}\|_F^2$ =4.96e+07

Importance sampling based robust power method (with prescanning): dataset **Enron**								
	Squared residual norm			**Running time (s)**			**Preprocessing time (s)**	
b \ B	10	30		10	30		10	30
$5n$	4.605e+07	4.545e+07		0.4014	1.176		0.0105	0.01043
$10n$	4.573e+07	4.55e+07		0.6255	1.773		0.01036	0.01044
$20n$	4.561e+07	4.545e+07		1.318	2.923		0.01041	0.01039
$30n$	4.563e+07	4.546e+07		1.459	4.041		0.01101	0.01104
$40n$	4.56e+07	4.544e+07		1.841	5.092		0.01039	0.01102

Table 22.15: Importance sampling based robust power method with prescanning: dataset **Enron**, $\|\mathbf{T}\|_F^2$ =4.96e+07

Sketching based robust power method: dataset **AP**								
	Squared residual norm			**Running time (s)**			**Preprocessing time (s)**	
b \ B	10	30		10	30		10	30
2^{10}	7.065e+06	6.703e+06		0.2316	0.8128		0.172	0.2472
2^{11}	6.674e+06	6.613e+06		0.4404	1.417		0.2419	0.3072
2^{12}	6.656e+06	6.57e+06		0.9689	2.936		0.3586	0.4257
2^{13}	6.585e+06	6.573e+06		2.058	6.172		0.5826	0.6592
2^{14}	6.57e+06	6.555e+06		4.558	13.99		0.9601	1.059

Table 22.16: Sketching based robust power method: dataset **AP**, $\|\mathbf{T}\|_F^2$ =6.905e+06

Importance sampling based robust power method (without prescanning): dataset **AP**								
	Squared residual norm			**Running time (s)**			**Preprocessing time (s)**	
b \ B	10	30		10	30		10	30
$5n$	6.859e+06	6.805e+06		0.3762	1.548		0.0	0.0
$10n$	6.631e+06	6.904e+06		0.5469	1.577		0.0	0.0
$20n$	6.572e+06	6.72e+06		0.9535	2.587		0.0	0.0
$30n$	6.64e+06	6.557e+06		1.263	3.53		0.0	0.0
$40n$	6.565e+06	6.564e+06		1.613	4.478		0.0	0.0

Table 22.17: Importance sampling based robust power method, without prescanning: dataset **AP**, $\|\mathbf{T}\|_F^2$ =6.905e+06

Importance sampling based robust power method (with prescanning): dataset **AP**								
	Squared residual norm			**Running time (s)**			**Preprocessing time (s)**	
b \ B	10	30		10	30		10	30
$5n$	6.58e+06	6.893e+06		0.4007	1.233		0.01045	0.01103
$10n$	6.604e+06	6.567e+06		0.7035	2.011		0.01061	0.01051
$20n$	6.803e+06	6.563e+06		1.166	3.246		0.01042	0.01061
$30n$	6.58e+06	6.573e+06		1.588	4.815		0.01103	0.01122
$40n$	6.66e+06	6.555e+06		1.989	5.601		0.01084	0.01103

Table 22.18: Importance sampling based robust power method with prescanning: dataset **AP**, $\|\mathbf{T}\|_F^2$ =6.905e+06

Sketching based robust power method: dataset **NSF**								
	Squared residual norm			**Running time (s)**			**Preprocessing time (s)**	
b \ B	10	30		10	30		10	30
2^{10}	4.342e+06	3.983e+06		0.2329	1.118		0.1731	0.2938
2^{11}	4.183e+06	3.906e+06		0.4415	1.434		0.2439	0.3061
2^{12}	3.929e+06	3.869e+06		0.9758	2.952		0.3554	0.4268
2^{13}	3.885e+06	3.86e+06		2.045	6.015		0.6591	0.6489
2^{14}	3.865e+06	3.854e+06		4.591	13.89		1.081	1.047

Table 22.19: Sketching based robust power method: dataset **NSF**, $\|\mathbf{T}\|_F^2$ =4.765e+06

Importance sampling based robust power method (without prescanning): dataset **NSF**								
	Squared residual norm			**Running time (s)**			**Preprocessing time (s)**	
b \ B	10	30		10	30		10	30
$5n$	3.899e+06	3.885e+06		0.3701	1.166		0.0	0.0
$10n$	3.891e+06	3.862e+06		0.5678	1.698		0.0	0.0
$20n$	3.896e+06	3.861e+06		1.029	2.927		0.0	0.0
$30n$	3.889e+06	3.856e+06		1.415	3.941		0.0	0.0
$40n$	3.867e+06	3.857e+06		1.823	5.287		0.0	0.0

Table 22.20: Importance sampling based robust power method, without prescanning: dataset **NSF**, $\|\mathbf{T}\|_F^2$ =4.765e+06

Importance sampling based robust power method (with prescanning): dataset **NSF**								
	Squared residual norm			**Running time (s)**			**Preprocessing time (s)**	
b \ B	10	30		10	30		10	30
$5n$	3.938e+06	3.902e+06		0.5219	1.566		0.01039	0.01108
$10n$	3.898e+06	3.867e+06		0.8981	2.568		0.01104	0.01043
$20n$	3.947e+06	3.864e+06		1.561	4.546		0.01097	0.01057
$30n$	3.889e+06	3.852e+06		2.207	6.477		0.011	0.01107
$40n$	3.891e+06	3.853e+06		2.827	8.446		0.01103	0.01065

Table 22.21: Importance sampling based robust power method with prescanning: dataset **NSF**, $\|\mathbf{T}\|_F^2$ =4.765e+06

Sketching based robust power method: dataset **NIPS**							
	Squared residual norm			**Running time (s)**		**Preprocessing time (s)**	
b \ B	10	30		10	30	10	30
2^{10}	5.649e+09	5.432e+09		0.2383	0.8686	0.1724	0.2625
2^{11}	5.522e+09	5.383e+09		0.45	1.52	0.2417	0.3247
2^{12}	5.394e+09	5.366e+09		1.264	3.304	0.5082	0.6358
2^{13}	5.375e+09	5.353e+09		2.07	6.068	0.5766	0.7604
2^{14}	5.36e+09	5.35e+09		4.591	13.84	0.9817	1.044

Table 22.22: Sketching based robust power method: dataset **NIPS**, $\|\mathbf{T}\|_F^2$ =6.058e+09

Importance sampling based robust power method (without prescanning): dataset **NIPS**							
	Squared residual norm			**Running time (s)**		**Preprocessing time (s)**	
b \ B	10	30		10	30	10	30
$5n$	5.648e+09	5.403e+09		0.3205	0.9938	0.0	0.0
$10n$	5.417e+09	5.347e+09		0.4739	1.753	0.0	0.0
$20n$	5.352e+09	5.349e+09		0.8345	2.464	0.0	0.0
$30n$	5.361e+09	5.35e+09		1.19	3.317	0.0	0.0
$40n$	5.352e+09	5.349e+09		1.532	4.153	0.0	0.0

Table 22.23: Importance sampling based robust power method, without prescanning: dataset **NIPS**, $\|\mathbf{T}\|_F^2$ =6.058e+09

Importance sampling based robust power method (with prescanning): dataset **NIPS**							
	Squared residual norm			**Running time (s)**		**Preprocessing time (s)**	
b \ B	10	30		10	30	10	30
$5n$	6.019e+09	5.943e+09		0.4818	1.401	0.01039	0.01057
$10n$	5.352e+09	5.389e+09		0.797	2.197	0.01063	0.01105
$20n$	5.387e+09	5.349e+09		1.335	3.849	0.01042	0.01039
$30n$	5.363e+09	5.346e+09		1.895	5.491	0.0104	0.01042
$40n$	5.352e+09	5.382e+09		2.523	7.019	0.01064	0.011

Table 22.24: Importance sampling based robust power method with prescanning: dataset **NIPS**, $\|\mathbf{T}\|_F^2$ =6.058e+09

Sketching based robust power method: dataset **KOS**								
	Squared residual norm			**Running time (s)**			**Preprocessing time (s)**	
b \ B	10	30		10	30		10	30
2^{10}	5.015e+04	4.807e+04		0.2356	0.8581		0.1781	0.2507
2^{11}	4.858e+04	4.787e+04		0.4482	1.417		0.2411	0.3072
2^{12}	4.797e+04	4.773e+04		0.9652	2.947		0.3556	0.4265
2^{13}	4.776e+04	4.764e+04		2.046	5.985		0.5722	0.6517
2^{14}	4.771e+04	4.761e+04		4.453	13.87		0.968	1.052

Table 22.25: Sketching based robust power method: dataset **KOS**, $\|\mathbf{T}\|_F^2$ =5.016e+04

Importance sampling based robust power method (without prescanning): dataset **KOS**								
	Squared residual norm			**Running time (s)**			**Preprocessing time (s)**	
b \ B	10	30		10	30		10	30
$5n$	4.783e+04	4.766e+04		0.3968	1.181		0.0	0.0
$10n$	4.817e+04	4.767e+04		0.6259	1.704		0.0	0.0
$20n$	4.775e+04	4.766e+04		1.039	2.79		0.0	0.0
$30n$	4.771e+04	4.762e+04		1.357	3.57		0.0	0.0
$40n$	4.767e+04	4.762e+04		1.76	4.892		0.0	0.0

Table 22.26: Importance sampling based robust power method, without prescanning: dataset **KOS**, $\|\mathbf{T}\|_F^2$ =5.016e+04

Importance sampling based robust power method (with prescanning): dataset **KOS**								
	Squared residual norm			**Running time (s)**			**Preprocessing time (s)**	
b \ B	10	30		10	30		10	30
$5n$	4.773e+04	4.766e+04		0.4808	1.35		0.01061	0.01099
$10n$	4.792e+04	4.765e+04		0.7363	2.024		0.01108	0.01044
$20n$	4.795e+04	4.766e+04		1.336	3.308		0.01056	0.01104
$30n$	4.785e+04	4.765e+04		1.728	4.631		0.01039	0.0104
$40n$	4.767e+04	4.764e+04		2.295	5.773		0.01063	0.01102

Table 22.27: Importance sampling based robust power method with prescanning: dataset **KOS**, $\|\mathbf{T}\|_F^2$ =5.016e+04

Part V

Open Problems

Chapter 23

Open Problems

In this chapter we give a long list of open problems . Some of these problems have appeared in other places and might be well-known to researchers in the community; nevertheless we include them since they are related to this thesis in some way.

To encourage people to solve these problems, we offer cash for each problem. The reward of each problem is not decided by the importance of the problem, but rather determined by the time when it is solved. To decide the order of time, we will compare the arXiv ID. For the i-th problem solved, we will pay $0.01 \times 2^{i-1}$ dollars.

23.1 Linear Programs

Linear programs are among most important optimization problems from both theoretical and practical view. In a recent result, [CLS19] show how to solve linear programs in the current matrix multiplication time. It would be interesting if this is still true if we are in a world where $n \times n$ matrix multiplication can be done in $O(n^2)$ time.

Open Problem 1. *Is it possible to solve linear programs in the matrix multiplication time?*

Another question is whether we can solve linear programs faster on sparse input. [LS14, LS15] show how to solve linear program in $O(\sqrt{d} \cdot \text{nnz}(A) + d^{2.5})$ time, where $\text{nnz}(A)$ is the number of non-zeros in A and d is the number of constraints.

Open Problem 2. *Is it possible to solve linear programs in the input sparsity time?*

There are other well-known open problems in the theory of linear programming. One important question is whether there is a strongly polynomial time algorithm.

Open Problem 3. *Does LP admit a strongly polynomial time algorithm?*

We concluded open problems on linear programs with a problem originally proposed by Seve Smale [Sma98] in 1998.

Open Problem 4 (Problem 9 in [Sma98])**.** *Is there a polynomial-time algorithm over the real numbers which decides the feasibility of the linear system of inequalities $Ax \geq b$?*

23.2 Matrix Multiplication

Let ω denote the exponent of matrix multiplication. Let α denote the dual exponent of the matrix multiplication.

Open Problem 5. *Is it possible that* $\omega \geq 2.5 - \alpha/2$*?*

For the current value of $\omega \sim 2.38$ and $\alpha \sim 0.31$, this is true. If $\omega = 2$, then $\alpha = 1$, it is also true. For the motivation of this inequality, we refer readers to [CLS19, LSZ19].

We also state a very well-known problem about matrix multiplication

Open Problem 6. *What is the right answer of* ω*?*

23.3 Ordinary Differential Equations

The following open problem is conjectured by Yin Tat Lee.

Open Problem 7. *[LSV18] gives an algorithm for sampling logconcave densities under m-strong convexity and M-smoothness. It takes $\kappa^{1.5}$ iterations (where $\kappa := M/m$), $O(1)$ gradient evaluation per iteration, and the total time is $\kappa^{1.5}d$. Is that possible to give an algorithm that takes κ iterations, and $O(1)$ gradient evaluation per iteration?*

23.4 Matrix Factorization

We discuss several open problems about matrix/tensor factorization in this section.

23.4.1 Weighted Factorization

Recall the definition of weighted low rank approximation

Definition 23.4.1. Given a matrix $A \in \mathbb{R}^{n\times n}$ and a matrix $W \in \mathbb{R}^{n\times n}$. Let W denote a rank-r weight matrix. For integer $k \geq 1$ and approximation ratio $\alpha > 1$, the goal is to find a rank-k matrix $B \in \mathbb{R}^{n\times n}$ such that

$$\|W \circ (B - A)\|_F \leq \alpha \cdot \min_{\text{rank}-k\ A'} \|W \circ (A' - A)\|_F.$$

We assume the entries of A and W are integers in the range $\{-M, -M+1, \cdots, M\}$ for an integer $M \leq 2^{\text{poly}(n)}$, i.e., that the entries of A and W can be specified with $\text{poly}(n)$ bits.

Open Problem 8. *For $k = O(1)$ and $\alpha = O(1)$, [RSW16] showed an $n^{O(r)}$ upper bound and a $2^{\Omega(r)}$ lower bound under Feige-*ETH*. Is it possible to close the gap?*

The above question was originally asked by David P. Woodruff (see Open Problem 9 in [CFHW17]).

Open Problem 9. *If W has r distinct columns (which is a stronger assumption than W has* rank-r*), [RSW16] showed $2^{\widetilde{O}(r^2)}$ upper bound and $2^{\Omega(r)}$ lower bound under Feige-*ETH*, is it possible to close the gap?*

23.4.2 Nonnegative Matrix Factorization

Nonnegative matrix factorization was theoretically studied by [AGKM12] and [Moi13]. The formal definition is

Definition 23.4.2 (NMF)**.** Given an $n \times n$ matrix A and a parameter k, the goal is to output two nonnegative $n \times k$ matrices U and $V^\top$ such that $UV = A$.

Open Problem 10. *There is an algorithm for NMF that takes $n^{O(k^2)}$ time due to [Moi13]. Under* ETH*, any algorithm that solves NMF takes $n^{\Omega(k)}$ time. The question is, can we close the gap?*

23.4.3 Tensor factorization

Tensor factorization/decomposition has many applications in computer science. We give a formal problem definition.

Definition 23.4.3. Given a $n \times n \times n$ tensor A and a parameter k, the goal is to output three $n \times k$ matrices U, V, W such that

$$\|U \otimes V \otimes W - A\|_F \leq O(1) \min_{\mathrm{rank}-k\ A_k} \|A_k - A\|_F.$$

Open Problem 11. *[SWZ19b] propose an algorithm that takes $\mathrm{nnz}(A) + n \cdot \mathrm{poly}(k) + 2^{O(k^2)}$ time. From the other perspective, [SWZ19b] show that there is no algorithm that takes $2^{o(k)}$ time unless Feige-*ETH *is false. For the situation $k = \omega(\log n)$, can we close the gap between $2^{O(k^2)}$ and $2^{\Omega(k)}$?*

In fact, all three open problems are equivalent in some sense. If someone can solve one of them, then probably also can improve the remaining ones. From the lower bound

side, all the current hardness results are based on ETH-type assumption. We don't know how to boost the current reduction. From the algorithmic side, all the current results are based on polynomial system solver, which has exponential dependence on the number of variables. For the target rank-k matrices, we don't know how to write down a system that only involves $o(k^2)$ variables.

23.5 Continuous Fourier Transform

23.5.1 One dimensional

We say $x(t) : \mathbb{R} \to \mathbb{C}$ is a k-Fourier-sparse signal if $x(t)$ can be written as $x(t) = \sum_{j=1}^{k} v_j e^{-2\pi \mathbf{i} f_j t}$ where $v_j \in \mathbb{C}$ and $f_j \in \mathbb{R}$, $\forall j \in [k]$. Let $\|x\|_T = (\frac{1}{T}\int_0^T |x(t)|^2 \mathrm{d}t)^{1/2}$.

Open Problem 12. *[CKPS16] showed that : for any k-Fourier-sparse signal $x(t) : \mathbb{R} \to \mathbb{C}$ and any duration $T > 0$, we have*

$$\max_{t \in [0,T]} |x(t)|^2 \leq O(k^4 \log^3 k) \cdot \|x\|_T^2.$$

[CKPS16] also showed that $\Omega(k^2)$ is necessary. Can we close the gap?

23.5.2 High dimensional

We say $x(t) : \mathbb{R}^d \to \mathbb{C}$ is a d-dimensional k-Fourier-sparse signal if $x(t)$ can be written as $x(t) = \sum_{j=1}^{k} v_j e^{-2\pi \mathbf{i} f_j^\top t}$ where $v_j \in \mathbb{C}$ and $f_j \in \mathbb{R}^d$, $\forall j \in [k]$. Let $\|x\|_T = (\frac{1}{T^d}\int_{[0,T]^d} |x(t)|^2 \mathrm{d}t)^{1/2}$.

Open Problem 13. *[Son17] showed that : for any d-dimensional k-Fourier-sparse signal $x(t) : \mathbb{R}^d \to \mathbb{C}$ and any duration $T > 0$, we have*

$$\max_{t \in [0,T]^d} |x(t)|^2 \leq k^{O(d)} \cdot \|x\|_T^2.$$

Is it possible to improve the upper bound $k^{O(d)}$, e.g. to $\mathrm{poly}(k, d)$? Is it possible to prove a lower bound of $k^{\Omega(d)}$?

We provide the details of [Son17] in the Chapter 31.

Currently, [PS15] and [CKPS16] are working for $d = 1$ case.

Open Problem 14. *Can we generalize [PS15] and [CKPS16] to arbitrary dimension d?*

23.6 Discrete Fourier Transform

23.6.1 High dimensional

We introduce some notations for high dimensional Fourier transform. We assume that the universe size $n = p^d$ for any positive integer p. Let $\omega = e^{2\pi \mathbf{i}/p}$ where $\mathbf{i} = \sqrt{-1}$. We define the normalized d-dimensional Fourier transform

$$\widehat{x}_f = \frac{1}{\sqrt{n}} \sum_{t \in [p]^d} x_t \cdot \omega^{f^\top t}, \forall f \in [p]^d$$

and the inverse Fourier transform is

$$x_t = \frac{1}{\sqrt{n}} \sum_{f \in [p]^d} \widehat{x}_f \cdot \omega^{-f^\top t}, \forall t \in [p]^d.$$

One important goal for high dimensional Fourier transform is to achieve dimension-free results. We suggest some problems for any $d \geq 1$.

Open Problem 15. *For the noiseless setting, [KVZ19] give an algorithm that uses*

$$O(k^3 \log^2 \log^2 n)$$

samples and $O(k^3 \log^2 k \log^2 n)$ time. This is already dimension-free! If the running time has to be $\mathrm{poly}(k \log n)$, what is the best sample complexity we can achieve?

Open Problem 16. *For the noisy setting, [NSW19a] give an algorithm that uses*

$$O(k \log k \log n)$$

samples and $\widetilde{O}(n)$ running time. Is $O(k \log k \log n)$ the right answer for sample complexity?

Open Problem 17. *For the noisy setting, can we get $\mathrm{poly}(k \log n)$ samples and $\mathrm{poly}(k \log n)$ running times?*

For $d = O(1)$, the state-of-the-art result [Kap16] takes $k \log n \cdot \log \log n$ samples and $k \operatorname{poly}(\log n)$ running times.

Open Problem 18. *For the noisy setting, for $d = O(1)$, what is the right answer for sample complexity?*

23.7 Discrete time continuous frequency Fourier transform

There is a long line of work doing Fourier transform in the discrete setting [GMS05, Iwe08, Iwe10, HIKP12a, HIKP12b, Iwe13, IKP14, IK14, Kap16, Kap17, CI17, NSW18, LN19, NSW19a]. We provide the formal definition here. For ease of discussion, we call it discrete-discrete setting.

Definition 23.7.1 (discrete time discrete frequency Fourier transform, [HIKP12a])**.** Given a vector $\widehat{x} \in \mathbb{C}^n$, an integer k and an approximation ratio $\alpha > 1$. design an algorithm that takes samples from $x \in \mathbb{C}^n$, and outputs a $O(k)$-sparse vector $y \in \mathbb{C}^n$ such that

$$\|y - \widehat{x}\|_2 \leq \alpha \cdot \min_{k-\text{sparse } z} \|z - \widehat{x}\|_2.$$

In general, we hope the number of samples and running time are both $k \operatorname{poly}(\log n)$.

To the best of our knowledge, [PS15] is the first work that defines the robust sparse Fourier transform in the continuous setting. For ease of discussion, we call it continuous-continuous setting.

Definition 23.7.2 (continuous time continuous frequency Fourier transform, [PS15])**.** Let x be a function $x(t) = x^*(t) + g(t)$ where $x^*(t) = \sum_{j=1}^k v_j e^{2\pi \mathbf{i} f_j t}$ (i.e. $x^*(t)$ is a k-Fourier-sparse signal) and $f_j \in [-F, F]$. Let $T > 0$. The goal is to design an algorithm that allows to take samples from $x(t)$ and outputs a $O(k)$-Fourier-sparse signal $y(t)$ such that

$$\frac{1}{T} \int_0^T |y(t) - x(t)|^2 \mathrm{d}t \leq \alpha \cdot \frac{1}{T} \int_0^T |g(t)|^2 \mathrm{d}t.$$

In general, we hope the number of samples and running time are $k \operatorname{poly}(\log FT)$.

For a function $x(t)$, we define

$$\|x(t)\|_T = \left(\frac{1}{T}\int_0^T |x(t)|^2 \mathrm{d}t\right)^{1/2}$$

The following definition is suggested by Jerry Li.

Definition 23.7.3 (discrete time continuous frequency Fourier transform)**.** Given k tones $(v_j, f_j) \in \mathbb{C} \times [0,1]$, we can access $x(t) = \sum_{j=1}^k v_j e^{-2\pi \mathbf{i} f_j t} + g(t)$, $\forall t \in [n]$. The goal is to output a $\mathrm{poly}(k \log n)$-sparse function $y(t)$ such that

$$\sum_{t=1}^n (y(t) - x(t))^2 \leq O(1) \cdot \sum_{t=1}^n g(t)^2.$$

Some interesting questions are:

Open Problem 19. *Can we generalize the continuous Fourier transform techniques in [PS15] to the setting where we are only allowed to take restricted samples (i.e., from integers), if we may assume a frequency gap? Are we still able to recover the continuous frequencies up to some precision, and then reconstruct the signal?*

Open Problem 20. *Can we generalize the continuous Fourier transform techniques in [CKPS16] to the setting where we are only allowed to take restricted samples and reconstruct the signal with finding the true continuous frequencies?*

The discrete time continuous frequency Fourier transform problem naturally arises in the study of learning low-rank Toeplitz covariance matrices. This is a well-studied problem in signal processing, for instance, in relationship to *direction-of-arrival* estimation. Here, we are given samples $X_1, \ldots, X_n$ drawn from a d-dimensional Gaussian with mean 0 and

covariance Σ, which we assume to be Toeplitz, and the goal is to approximately recover Σ. Moreover, we are allowed to access individual coordinates of each X_i one-by-one, and only pay a cost of 1 in the sample complexity for every coordinate we access.

Classical algorithms (such as ruler-based methods, or matrix pencil-based methods), either require a number of samples which is polynomial in the ambient dimension, or are not very tolerant to noise. In [ELMM19], it was demonstrated that by solving the discrete time continuous frequency Fourier transform problem, one can obtain an algorithm which is both robust to noise, and which requires a number of samples which depends only polylogarithmically in the ambient dimension, when Σ is low rank. The aforementioned paper gives an algorithm for this problem which is sample-efficient, but has exponential running time. By instead using hashing-based techniques, one could hope to obtain a polynomial-time algorithm for this problem, which would in turn yield the first efficient algorithm for learning low-rank Toeplitz covariance matrices using only $\operatorname{poly}\log(d)$ samples.

23.8 Applications of Fourier transform

23.8.1 Speeding up linear programs

It is known that Subsampled Randomized Hadamard Transform gives the subspace embedding.

Theorem 23.8.1 ([AC06, Sar06, Tro11b, DMM06a, DMMS11, DMIMW12, LDFU13]). *Let $S = \frac{\sqrt{n}}{\sqrt{r}} P H_n D$, where D is an $n \times n$ diagonal matrix with i.i.d. diagonal entries $D_{i,i}$ in which $D_{i,i} = 1$ with probability $1/2$, and $D_{i,i} = -1$ with probability $1/2$. H_n refers to the Hadamard matrix of size n, which we assume is a power of 2. Here the (i,j)-th entry of H_n is given by $(-1)^{\langle i,j \rangle}/\sqrt{n}$ where $\langle i,j \rangle = \sum_{z=1}^{\log n} i_z \cdot j_z$ and $(i_{\log n}, \cdots, i_1)$ and $(j_{\log n}, \cdots, j_1)$ are the binary representations of i and j respectively. Then $r \times n$ matrix P samples r coordinates of an n-dimensional vector uniformly at random, where*

$$r = \Omega(\epsilon^{-2} \log d \cdot (\sqrt{d} + \sqrt{\log n})^2)$$

Then with probability at least .99, for any fixed $n \times d$ matrix U with orthonormal columns,

$$\| I_d - U^\top S^\top S U \|_2 \leq \epsilon.$$

Further, for any vector $x \in \mathbb{R}^n$, Sx can be computed in $O(n \log r)$ time.

An interesting question is

Open Problem 21. *If we don't use diagonal matrix D, can we show subspace embedding guarantee presented in Theorem 23.8.1, or the guarantee presented in Lemma 3.9.1?*

If the above conjecture holds, then we have more power to design fast algorithm for several fundamental problems, i.e. linear programs.

23.8.2 Proving convergence result for deep neural network

Open Problem 22. *Is the frequency gap assumption in continuous Fourier transform ([PS15]) somehow equivalent to data-separable assumption in deep learning theory ([AZLS18, AZLS19, SY19])?*

The above problem seems strange in some sense, since Fourier transform and deep learning are completely different problems. The motivation of the stating that problem is : 1) Fourier transform seems to be related to kernel regression with random Fourier feature, 2) a recent work shows that an infinite width one-hidden-layer neural network is somehow equivalent to Kernel regression [ADH+19].

23.9 An Over-parameterization Theory of Neural Networks

Over-parameterization theory for deep neural networks becomes extremely popular over the last few years. There is a long line (still growing very quickly) of work proving that (stochastic) gradient descent algorithm is able to find the global minimum if the network is wide enough [LL18, DZPS19, AZLS18, AZLS19, DLL+19, ZCZG18, SY19].

Formally speaking, the existing results show that as long as the width m is at least polynomial of number of input data n, then (S)GD-type algorithm can work in the following sense: first randomly pick a weight matrix to be the initialization point, update the weight matrix according to gradient direction over each iteration, and eventually find the global minimum.

Yin Tat Lee [Lee18] conjectured the following

Open Problem 23. *If m is nearly in n (ignore the polylog factors), then we can prove convergence result for one-hidden layer neural network.*

Next, we want to ask the open questions for multiple layer neural networks. Let L denote the number of layers of the neural network, let n denote the number of input data points, let d denote the dimension of input data points, let δ denote the failure probability, and let m denote the width of the neural network.

Open Problem 24. *For a deep neural network with ReLU activation function, what is the smallest m that suffices to show convergence?*

Similarly, we are also curious about the right answer for RNNs,

Open Problem 25. *For a recurrent neural network with ReLU activation function, what is the smallest m that suffices to show convergence?*

[AZLS18] proved a large polynomial dependence on all the parameters, we believe it is improvable. However, we are not sure about the right answer. Note that all the above three open problems are about training error.

23.10 Matrix Chernoff and Discrepancy

23.10.1 Matrix Chernoff

Open Problem 26. *[KS18] show that $O(\epsilon^{-2}\log^2 n)$ spanning trees suffice to produce a $(1\pm\epsilon)$-spectral graph sparsifier, and that $\Omega(\epsilon^{-2}\log n)$ spanning trees are necessary. Which one is tight? Can we prove matching upper and lower bound?*

For more background on this problem, we refer the reader to [BSST13, KS18].

We consulted some experts[1]. They are not aware of any form of the following Chernoff-type result for hyperbolic polynomials. They believed the following result is interesting.

Open Problem 27. *Given vectors $x_1,\cdots,x_n\in\mathbb{R}^d$ and a polynomial h with respect to e, let us define the hyperbolic norm $\|x\|_h=\max_{i\in[n]}|\lambda_i(x)|$ where $h(te-x)=h(e)\cdot\prod_{i=1}^n(t-\lambda_i(x))$, and suppose $\|x_i\|_h\le 1$ for all $i\in[n]$. If we sample i.i.d. $\epsilon_1,\cdots,\epsilon_n$ uniformly from $\{-1,1\}$, then does*

$$\left\|\sum_{i=1}^n\epsilon_i x_i\right\|_h\le O(\sqrt{n\log n})$$

hold with constant probability?

23.10.2 Discrepancy

Open Problem 28 (Matrix Spencer)**.** *For any symmetric matrices $A_1,\cdots,A_n\in\mathbb{R}^{n\times n}$ with $\|A_i\|\le 1$, do there exist signs $\epsilon_1,\cdots,\epsilon_n\in\{-1,1\}$ such that*

$$\left\|\sum_{i=1}^n\epsilon_i A_i\right\|=O(\sqrt{n}).$$

[1] Personal communication with James Renegar

The above problem is called the Matrix Spencer conjecture, see e.g. [Mek04] and [RR19].

[MSS15b] resolved the Kadison-Singer conjecture, a long-standing open problem, but their proof is non-constructive. From the algorithmic perspective, a major open question is the following:

Open Problem 29. *Given positive semidefinite matrices* $A_1, \cdots, A_m \in \mathbb{R}^{n \times n}$ *and* $\epsilon > 0$ *with* $\sum_{i=1}^m A_i = I_n$ *and* $\mathrm{tr}[A_i] \leq \epsilon$ *for all* $i \in [m]$*, is there a polynomial time algorithm to find signs* $x_1, \ldots, x_m \in \{-1, 1\}$ *such that* $\left\| \sum_{i=1}^m x_i A_i \right\| \leq O(\sqrt{\epsilon})$*?*

In order to describe the next open problems, we provide some definitions:

Definition 23.10.1 ([Brä18])**.** Given a degree d polynomial h that is hyperbolic with respect to $e \in \mathbb{R}^n$, The open hyperbolicity cone of h with respect to e is the set

$$\Gamma_{++}(h, e) = \{x \in \mathbb{R}^n : \lambda_i(x) > 0 \ \forall i \in [n]\}.$$

We also define the closed hyperbolicity cone

$$\Gamma_{+}(h, e) = \{x \in \mathbb{R}^n : \lambda_i(x) \geq 0 \ \forall i \in [n]\},$$

where $h(te - x) = h(e) \cdot \prod_{i=1}^n (t - \lambda_i(x))$.

The trace, rank and spectral radius (with respect to e) of $x \in \mathbb{R}^n$ are defined as for matrices

$$\mathrm{tr}[x] = \sum_{i=1}^n \lambda_i(x), \quad \mathrm{rank}_h(x) = |\{i \in [n] : \lambda_i(x) \neq 0\}|, \quad \text{and} \quad \|x\|_h = \max_{i \in [n]} \lambda_i(x).$$

Recently, [Brä18] generalized Kadison-Singer to hyperbolic polynomials and also higher rank. Note that [MSS15b] can be viewed as the special case when h is the determinant polynomial with rank-1 matrices. [KLS19] generalized [MSS15b] from a different perspective. It would be interesting to combine [KLS19] and [Brä18] to show the following result.

Open Problem 30. *Let h be a polynomial that is hyperbolic with respect to $e \in \mathbb{R}^n$. Let $v_1, \cdots, v_m \in \Gamma_+(h,e)$ with $\mathrm{rank}_h(v_i) \leq 1$, and*

$$\sigma^2 = \left\| \sum_{i=1}^m \mathrm{tr}[v_i] v_i \right\|_h .$$

Then there exists a choice of signs $\epsilon_i \in \{-1, 1\}$ and some constant $c > 0$ such that

$$\left\| \sum_{i=1}^m \epsilon_i v_i \right\|_h \leq c \cdot \sigma.$$

We also wonder if similar techniques can be used to sparsify hyperbolicity cones:

Open Problem 31. *Given $\epsilon > 0$, a polynomial h that is hyperbolic with respect to $e \in \mathbb{R}^n$ and $v_1, \cdots, v_m \in \Gamma_+(h,e)$ with $v_1 + \cdots + v_m = e$, is it always possible to find coefficients $s \in \mathbb{R}^m_{\geq 0}$ such that $|\mathrm{supp}(s)| \leq n/\epsilon^2$ and $\|e - \sum_{i=1}^m s_i v_i\|_h = O(\epsilon)$?*

The above problem can be viewed as a generalization of [BSS12].

23.11 Edit Distance and Longest Common Subsequence

In this section we consider two of the most ubiquitous measures of similarity between a pair of strings: the longest common subsequence (LCS) and the edit distance. The LCS of two n-character strings A and B is simply their longest (not necessarily contiguous) common substring. Edit distance is the minimum number of character insertions, deletions, and substitutions required to transform A and B. In fact, under a slightly more restricted definition that does not allow substitutions, the two measures are complements to each other and the problems of computing them are exactly equivalent.

Under plausible fine-grained complexity assumptions such as SETH, neither edit distance nor LCS can be computed much faster [AWW14, ABW15, BI15, BK15, AHWW16].

For edit distance, an $\widetilde{O}(n + \Delta^2)$-time algorithm of [LMS98] (where Δ is the true edit distance between the strings) implies a linear-time $\sqrt{n}$-approximation algorithm. The approximation factor has been significantly improved in a series of works to $O(n^{3/7})$ [BYJKK04], to $O(n^{0.34})$ [BES06], to $O(2^{\widetilde{O}(\sqrt{\log n})})$ [AO09], and finally to polylogarithmic [AKO10]. A recent work of Boroujeni, Ehsani, Ghodsi, HajiAghayi and Seddighin [BEG+18] obtains a constant factor approximation quantum algorithm for edit distance that runs in truly subquadratic time.

We state three open problems related to edit distance. Two of the problems were originally posted by Aviad Rubinstein [Rub18].

Open Problem 32. *Is there a constant-factor approximation to edit distance that runs in near-linear time?*

Currently, the best approximation ratio is $(3 + \epsilon)$ [RSSS19].

Open Problem 33. *Is there a $(1+\epsilon)$-approximation algorithm for edit distance in truly subquadratic (or near-linear) time?*

Note that there is a more concrete plan to make progress on approximation ratio of edit distance problem. For more details, we refer the readers to [RSSS19].

Note that the current subquadratic time constant approximation edit distance algorithm [CDG+18] is a randomized algorithm. It would be interesting to derandomize the algorithm.

Open Problem 34. *Is there a deterministic subquadratic time constant approximation edit distance algorithm?*

Now let us come to the LCS problem. The textbook dynamic programming algorithm for computing LCS (or edit distance) runs in $O(n^2)$ time, and a slightly faster $O(n^2/\log^2(n))$-time algorithm is known due to Masek and Paterson [MP80]. There is a trivial algorithm that gives 2-approximation for binary strings. Finding faster algorithms is a central and long standing open problem both in theory and in practice (e.g. Problem 35 of [Knu72]).

Rubinstein and Song [RS19b] proposed a truly subquadratic time $(2-\epsilon)$-approximation algorithm for binary strings. Let $|\Sigma|$ be the alphabetical size. Then one natural question is if we can generalize the algorithm for $|\Sigma| = 3, 4$ or some constants.

Open Problem 35. *For any constant $|\Sigma|$, is there a truly subquadratic time algorithm that gives $(|\Sigma| - \epsilon)$-approximation for some constants $\epsilon > 0$?*

Another interesting question is about de-randomization.

Open Problem 36. *Can we de-randomize the* LCS *algorithm in [RSSS19]?*

23.12 Map-Reduce

Many modern parallel systems, such as MapReduce, Hadoop, and Spark, can be modeled well by the MPC model. The MPC model captures well coarse-grained computation on large data - data is distributed to processors, each of which has a sublinear (in the input data) amount of memory and we alternate between rounds of computation and rounds of communication, where each machine can communicate an amount of data as large as the size of its memory.

We consider one fundamental graph problem: connectivity. On an undirected graph with n and m edges, $O(\log n)$ round connectivity algorithm has been known for over 35 years. In 2018, [ASS+18] proposed a new algorithm that only needs $O(\log D \cdot \log\log_{m/n} n)$ where D is the diameter of the graph. Very recently, Behnezhad, Dhulipala, Esfandiari, Lacki and Mirrokni [BDE+19] improved the number of rounds to $O(\log D + \log\log_{m/n} n)$. An interesting open question is

Open Problem 37. *Is there an $O(\log D)$ rounds algorithm for graph connectivity?*

The above problem was originally suggested by Peilin Zhong. In next a few paragraphs, we discuss several other open problems which are proposed by Xiaorui Sun.

Roughgarden, Vassilvitskii and Wang [RVW18] showed that if there is an $\omega(1)$-round MPC lower bound for graph connectivity, then $\mathsf{NC}_1 \neq \mathsf{P}$. Since NC_1 vs P is a well-known hard problem, it is unlikely to prove an $\omega(1)$-round unconditional lower bound for connectivity. Hence, we ask if an conditional lower bound is possible.

Open Problem 38. *Is there an $\omega(1)$-round conditional lower bound for graph connectivity?*

Furthermore, people actually conjectured that any algorithm solve graph connectivity problem requires $\Omega(\log n)$ rounds. This conjecture has been use to show other graph problems are unlikely to have constant round algorithms (Yaroslavtsev and Vadapalli [YV18], Ghaffari, Kuhn and Uitto [GKU19]).

Open Problem 39. *Is there an $\Omega(\log n)$-round conditional lower bound for graph connectivity?*

In a more broader setting, it will be interesting to understand the complexity of MPC. In particular, if all the polynomial-time solvable problem have an efficient MPC algorithm. Since linear programming is P-Complete, we ask

Open Problem 40. *Is there an $O(n^{o(1)})$ rounds algorithm for linear programming?*

Dynamic programming in MPC has been studied recently. Im, Moseley and Sun [IMS17a] and Hajiaghayi, Seddighin and Sun [HSS19] showed that many classic dynamic programming problems have constant rounds approximation algorithms. These problems include weighted interval scheduling, longest increasing subsequence, edit distance and longest common subsequence. But it is unclear if these problems can be solved exactly in constant rounds.

Open Problem 41. *Can dynamic programming problems be solved exactly in $O(1)$ rounds?*

Part VI

More Algorithms for Fundamental Problems

Chapter 24

ℓ_∞ Regression

Sketching has emerged as a powerful technique for speeding up problems in numerical linear algebra, such as regression. In the overconstrained regression problem, one is given an $n \times d$ matrix A, with $n \gg d$, as well as an $n \times 1$ vector b, and one wants to find a vector $\widehat{x}$ so as to minimize the residual error $\|Ax - b\|_2$. Using the sketch and solve paradigm, one first computes $S \cdot A$ and $S \cdot b$ for a randomly chosen matrix S, then outputs $x' = (SA)^\dagger Sb$ so as to minimize $\|SAx' - Sb\|_2$.

The sketch-and-solve paradigm gives a bound on $\|x' - x^*\|_2$ when A is well-conditioned. Our main result is that, when S is the subsampled randomized Fourier/Hadamard transform, the error $x' - x^*$ behaves as if it lies in a "random" direction within this bound: for any fixed direction $a \in \mathbb{R}^d$, we have with $1 - d^{-c}$ probability that

$$\langle a, x' - x^* \rangle \lesssim \frac{\|a\|_2 \|x' - x^*\|_2}{d^{\frac{1}{2}-\gamma}}, \tag{24.1}$$

where $c, \gamma > 0$ are arbitrary constants. This implies $\|x' - x^*\|_\infty$ is a factor $d^{\frac{1}{2}-\gamma}$ smaller than $\|x' - x^*\|_2$. It also gives a better bound on the generalization of x' to new examples: if rows of A correspond to examples and columns to features, then our result gives a better bound for the error introduced by sketch-and-solve when classifying fresh examples. We show that not all oblivious subspace embeddings S satisfy these properties. In particular, we give

counterexamples showing that matrices based on Count-Sketch or leverage score sampling do not satisfy these properties.

We also provide lower bounds, both on how small $\|x' - x^*\|_2$ can be, and for our new guarantee (24.1), showing that the subsampled randomized Fourier/Hadamard transform is nearly optimal. Our lower bound on $\|x' - x^*\|_2$ shows that there is an $O(1/\epsilon)$ separation in the dimension of the optimal oblivious subspace embedding required for outputting an x' for which $\|x' - x^*\|_2 \leq \epsilon \|Ax^* - b\|_2 \cdot \|A^\dagger\|_2$, compared to the dimension of the optimal oblivious subspace embedding required for outputting an x' for which $\|Ax' - b\|_2 \leq (1+\epsilon)\|Ax^* - b\|_2$, that is, the former problem requires dimension $\Omega(d/\epsilon^2)$ while the latter problem can be solved with dimension $O(d/\epsilon)$. This explains the reason known upper bounds on the dimensions of these two variants of regression have differed in prior work.

This part is based upon the following previous publication

- Eric Price, Zhao Song, David P. Woodruff
  Fast Regression with an ℓ_∞ Guarantee.
  ICALP 2017 [PSW17]

24.1 Introduction

Oblivious subspace embeddings (OSEs) were introduced by Sarlos [Sar06] to solve linear algebra problems more quickly than traditional methods. An OSE is a distribution of matrices $S \in \mathbb{R}^{m \times n}$ with $m \ll n$ such that, for any d-dimensional subspace $U \subset \mathbb{R}^n$, with "high" probability S preserves the norm of every vector in the subspace. OSEs are a generalization of the classic Johnson-Lindenstrauss lemma from vectors to subspaces. Formally,

we require that with probability $1 - \delta$,

$$\|Sx\|_2 = (1 \pm \epsilon) \|x\|_2$$

simultaneously for all $x \in U$, that is, $(1 - \epsilon) \|x\|_2 \leq \|Sx\|_2 \leq (1 + \epsilon) \|x\|_2$.

A major application of OSEs is to regression. The regression problem is, given $b \in \mathbb{R}^n$ and $A \in \mathbb{R}^{n \times d}$ for $n \geq d$, to solve for

$$x^* = \operatorname*{argmin}_{x \in \mathbb{R}^d} \|Ax - b\|_2 \tag{24.2}$$

Because A is a "tall" matrix with more rows than columns, the system is overdetermined and there is likely no solution to $Ax = b$, but regression will find the closest point to b in the space spanned by A. The classic answer to regression is to use the Moore-Penrose pseudoinverse: $x^* = A^\dagger b$ where

$$A^\dagger = (A^\top A)^{-1} A^\top$$

is the "pseudoinverse" of A (assuming A has full column rank, which we will typically do for simplicity). This classic solution takes $O(nd^{\omega-1} + d^\omega)$ time, where $\omega < 2.373$ is the matrix multiplication constant [CW90, Wil12, LG14]: $nd^{\omega-1}$ time to compute $A^\top A$ and d^ω time to compute the inverse.

OSEs speed up the process by replacing (24.2) with

$$x' = \operatorname*{argmin}_{x \in \mathbb{R}^d} \|SAx - Sb\|_2$$

for an OSE S on $d + 1$-dimensional spaces. This replaces the $n \times d$ regression problem with an $m \times d$ problem, which can be solved more quickly since $m \ll n$. Because $Ax - b$ lies in

the $d+1$-dimensional space spanned by b and the columns of A, with high probability S preserves the norm of $SAx - Sb$ to $1 \pm \epsilon$ for all x. Thus,

$$\|Ax' - b\|_2 \leq \frac{1+\epsilon}{1-\epsilon} \|Ax^* - b\|_2 .$$

That is, S produces a solution x' which preserves the *cost* of the regression problem. The running time for this method depends on (1) the reduced dimension m and (2) the time it takes to multiply S by A. We can compute these for "standard" OSE types:

- If S has i.i.d. Gaussian entries, then $m = O(d/\epsilon^2)$ is sufficient (and in fact, $m \geq d/\epsilon^2$ is required [NN14]). However, computing SA takes $O(mnd) = O(nd^2/\epsilon^2)$ time, which is worse than solving the original regression problem (one can speed this up using fast matrix multiplication, though it is still worse than solving the original problem).

- If S is a subsampled randomized Hadamard transform (SRHT) matrix with random sign flips (see Theorem 2.4 in [Woo14a] for a survey, and also see [AC06, Sar06, DMM06c, DMMS11, KW11, Tro11b, DMIMW12, LDFU13, AL13, Bou14, HR16, CNW16]), then m increases to $\widetilde{O}(d/\epsilon^2 \cdot \log n)$, where $\widetilde{O}(f) = f \operatorname{poly}(\log(f))$. But now, we can compute SA using the fast Hadamard transform in $O(nd \log n)$ time. This makes the overall regression problem take $O(nd \log n + d^\omega/\epsilon^2)$ time.

- If S is a random sparse matrix with random signs (the "Count-Sketch" matrix), then $m = d^{1+\gamma}/\epsilon^2$ suffices for $\gamma > 0$ a decreasing function of the sparsity [CW13, MM13, NN13a, BDN15, Coh16a]. (The definition of a Count-Sketch matrix is, for any $s \geq 1$, $S_{i,j} \in \{0, -1/\sqrt{s}, 1/\sqrt{s}\}$, $\forall i \in [m], j \in [n]$ and the column sparsity of matrix S is s. Independently in each column s positions are chosen uniformly at random without

replacement, and each chosen position is set to $-1/\sqrt{s}$ with probability 1/2, and $+1/\sqrt{s}$ with probability 1/2.) Sparse OSEs can benefit from the sparsity of A, allowing for a running time of $\widetilde{O}(\text{nnz}(A)) + \widetilde{O}(d^\omega/\epsilon^2)$, where $\text{nnz}(A)$ denotes the number of non-zeros in A.

When n is large, the latter two algorithms are substantially faster than the naïve $nd^{\omega-1}$ method.

24.1.1 Our Contributions

Despite the success of using subspace embeddings to speed up regression, often what practitioners are interested is not in preserving the cost of the regression problem, but rather in the *generalization* or *prediction* error provided by the vector x'. Ideally, we would like for any future (unseen) example $a \in \mathbb{R}^d$, that $\langle a, x' \rangle \approx \langle a, x^* \rangle$ with high probability.

Ultimately one may want to use x' to do classification, such as regularized least squares classification (RLSC) [RYP03], which has been found in cases to do as well as support vector machines but is much simpler [ZP04]. In this application, given a training set of examples with multiple (non-binary) labels identified with the rows of an $n \times d$ matrix A, one creates an $n \times r$ matrix B, each column indicating the presence or absence of one of the r possible labels in each example. One then solves the multiple response regression problem $\min_X \|AX - B\|_F$, and uses X to classify future examples. A commonly used method is for a future example a, to compute $\langle a, x_1 \rangle, \ldots, \langle a, x_r \rangle$, where $x_1, \ldots, x_r$ are the columns of X. One then chooses the label i for which $\langle a, x_i \rangle$ is maximum.

For this to work, we would like the inner products $\langle a, x'_1 \rangle, \ldots, \langle a, x'_r \rangle$ to be close to

$\langle a, x_1^* \rangle, \ldots, \langle a, x_r^* \rangle$, where X' is the solution to $\min_X \|SAX - SB\|_F$ and X^* is the solution to $\min_X \|AX - B\|_F$. For any $O(1)$-accurate OSE on $d+r$ dimensional spaces [Sar06], which also satisfies so-called approximate matrix multiplication with error $\epsilon' = \epsilon/\sqrt{(d+r)}$, we get that

$$\|x' - x^*\|_2 \leq O(\epsilon) \cdot \|Ax^* - b\|_2 \cdot \left\|A^\dagger\right\|_2 \tag{24.3}$$

where $\left\|A^\dagger\right\|$ is the spectral norm of $A^\dagger$, which equals the reciprocal of the smallest singular value of A. To obtain a generalization error bound for an unseen example a, one has

$$|\langle a, x^* \rangle - \langle a, x' \rangle| = |\langle a, x^* - x' \rangle| \leq \|x^* - x'\|_2 \|a\|_2 = O(\epsilon)\|a\|_2 \|Ax^* - b\|_2 \left\|A^\dagger\right\|_2, \tag{24.4}$$

which could be tight if given only the guarantee in (24.3). However, if the difference vector $x' - x^*$ were distributed in a uniformly random direction subject to (24.3), then one would expect an $\widetilde{O}(\sqrt{d})$ factor improvement in the bound. This is what our main theorem shows:

Theorem 24.1.1 (Main Theorem, informal)**.** *Suppose $n \leq \text{poly}(d)$ and matrix $A \in \mathbb{R}^{n \times d}$ and vector $b \in \mathbb{R}^n$ are given. Let $S \in \mathbb{R}^{m \times n}$ be a subsampled randomized Hadamard transform matrix with $m = d^{1+\gamma}/\epsilon^2$ rows for an arbitrarily small constant $\gamma > 0$. For $x' = \text{argmin}_{x \in \mathbb{R}^d} \|SAx - Sb\|_2$ and $x^* = \text{argmin}_{x \in \mathbb{R}^d} \|Ax - b\|_2$, and any fixed $a \in \mathbb{R}^d$,*

$$|\langle a, x^* \rangle - \langle a, x' \rangle| \leq \frac{\epsilon}{\sqrt{d}} \|a\|_2 \|Ax^* - b\|_2 \left\|A^\dagger\right\|_2. \tag{24.5}$$

with probability $1 - 1/d^C$ for an arbitrarily large constant $C > 0$. This implies that

$$\|x^* - x'\|_\infty \leq \frac{\epsilon}{\sqrt{d}} \|Ax^* - b\|_2 \left\|A^\dagger\right\|_2. \tag{24.6}$$

with $1 - 1/d^{C-1}$ probability.

If $n > \text{poly}(d)$, then by first composing S with a Count-Sketch OSE with $\text{poly}(d)$ rows, one can achieve the same guarantee.

(Here γ is a constant going to zero as n increases; see Theorem 24.3.3 for a formal statement of Theorem 24.1.1.)

Notice that Theorem 24.1.1 is considerably stronger than that of (24.4) provided by existing guarantees. Indeed, in order to achieve the guarantee (24.6) in Theorem 24.1.1, one would need to set $\epsilon' = \epsilon/\sqrt{d}$ in existing OSEs, resulting in $\Omega(d^2/\epsilon^2)$ rows. In contrast, we achieve only $d^{1+\gamma}/\epsilon^2$ rows. We can improve the bound in Theorem 24.1.1 to $m = d/\epsilon^2$ if S is a matrix of i.i.d. Gaussians; however, as noted, computing $S \cdot A$ is slower in this case.

Note that Theorem 24.1.1 also *makes no distributional assumptions* on the data, and thus the data could be heavy-tailed or even adversarially corrupted. This implies that our bound is still useful when the rows of A are not sampled independently from a distribution with bounded variance.

The ℓ_∞ bound (24.6) of Theorem 24.1.1 is achieved by applying (24.5) to the standard basis vectors $a = e_i$ for each $i \in [d]$ and applying a union bound. This ℓ_∞ guarantee often has a more natural interpretation than the ℓ_2 guarantee—if we think of the regression as attributing the observable as a sum of various factors, (24.6) says that the contribution of each factor is estimated well. One may also see our contribution as giving a way for estimating the pseudoinverse $A^\dagger$ *entrywise*. Namely, we get that $(SA)^\dagger S \approx A^\dagger$ in the sense that each entry is within additive $O(\epsilon\sqrt{\frac{\log d}{d}}\left\|A^\dagger\right\|_2)$. There is a lot of work on computing entries of inverses of a matrix, see, e.g., [ADL+12, LAKD08].

Another benefit of the ℓ_∞ guarantee is when the regression vector x^* is expected to be k-*sparse* (e.g. [Lee12]). In such cases, thresholding to the top k entries will yield an ℓ_2 guarantee a factor $\sqrt{\frac{k}{d}}$ better than (24.3).

One could ask if Theorem 24.1.1 also holds for sparse OSEs, such as the Count-Sketch. Surprisingly, we show that one cannot achieve the generalization error guarantee in Theorem 24.1.1 with high probability, say, $1 - 1/d$, using such embeddings, despite the fact that such embeddings do approximate the cost of the regression problem up to a $1 + \epsilon$ factor with high probability. This shows that the generalization error guarantee is achieved by some subspace embeddings but not all.

Theorem 24.1.2 (Not all subspace embeddings give the ℓ_∞ guarantee; informal version of Theorem 24.8.1)**.** *The Count-Sketch matrix with $d^{1.5}$ rows and sparsity $d^{.25}$—which is an OSE with exponentially small failure probability—with constant probability will have a result x' that does not satisfy the ℓ_∞ guarantee (24.6).*

We can show that Theorem 24.1.1 holds for S based on the Count-Sketch OSE T with $d^{O(C)}/\epsilon^2$ rows with $1 - 1/d^C$ probability. We can thus compose the Count-Sketch OSE with the SRHT matrix and obtain an $O(\text{nnz}(A)) + \text{poly}(d/\epsilon)$ time algorithm to compute $S \cdot TA$ achieving (24.6). We can also compute $R \cdot S \cdot T \cdot A$, where R is a matrix of Gaussians, which is more efficient now that STA only has $d^{1+\gamma}/\epsilon^2$ rows; this will reduce the number of rows to d/ϵ^2.

Another common method of dimensionality reduction for linear regression is *leverage score sampling* [DMIMW12, LMP13, PKB14, CMM15], which subsamples the rows of A by choosing each row with probability proportional to its "leverage scores". With $O(d \log(d/\delta)/\epsilon^2)$ rows taken, the result x' will satisfy the ℓ_2 bound (24.3) with probability $1 - \delta$. However, it does not give a good ℓ_∞ bound:

Theorem 24.1.3 (Leverage score sampling does not give the ℓ_∞ guarantee; informal version

of Theorem 24.9.1). *Leverage score sampling with $d^{1.5}$ rows—which satisfies the ℓ_2 bound with exponentially small failure probability—with constant probability will have a result x' that does not satisfy the ℓ_∞ guarantee (24.6).*

Finally, we show that the $d^{1+\gamma}/\epsilon^2$ rows that SRHT matrices use is roughly optimal:

Theorem 24.1.4 (Lower bounds for ℓ_2 and ℓ_∞ guarantees; informal versions of of Theorem 24.5.1 and Corollary 24.5.3)**.** *Any sketching matrix distribution over $m \times n$ matrices that satisfies either the ℓ_2 guarantee (24.3) or the ℓ_∞ guarantee (24.6) must have $m \gtrsim \min(n, d/\epsilon^2)$.*

Notice that our result shows the necessity of the $1/\epsilon$ separation between the results originally defined in Equation (3) and (4) of Theorem 12 of [Sar06]. If we want to output some vector x' such that $\|Ax' - b\|_2 \leq (1+\epsilon)\|Ax^* - b\|_2$, then it is known that $m = \Theta(d/\epsilon)$ is necessary and sufficient. However, if we want to output a vector x' such that $\|x' - x^*\|_2 \leq \epsilon\|Ax^* - b\|_2 \cdot \|A^\dagger\|_2$, then we show that $m = \Theta(d/\epsilon^2)$ is necessary and sufficient.

Comparison to Gradient Descent While this work is primarily about sketching methods, one could instead apply iterative methods such as gradient descent, after appropriately preconditioning the matrix, see, e.g., [AMT10, ZF13, CW13]. That is, one can use an OSE with constant ϵ to construct a preconditioner for A and then run conjugate gradient using the preconditioner. This gives an overall dependence of $\log(1/\epsilon)$.

The main drawback of this approach is that one loses the ability to save on storage space or number of passes when A appears in a stream, or to save on communication or rounds when A is distributed. Given increasingly large data sets, such scenarios are now quite common, see, e.g., [CW90] for regression algorithms in the data stream model. In

situations where the entries of A appear sequentially, for example, a row at a time, one does not need to store the full $n \times d$ matrix A but only the $m \times d$ matrix SA.

Also, iterative methods can be less efficient when solving multiple response regression, where one wants to minimize $\|AX - B\|$ for a $d \times t$ matrix X and an $n \times t$ matrix B. This is the case when ϵ is constant and t is large, which can occur in some applications (though there are also other applications for which ϵ is very small). For example, conjugate gradient with a preconditioner will take $\widetilde{O}(ndt)$ time while using an OSE directly will take only $\widetilde{O}(nd + d^2t)$ time (since one effectively replaces n with $O(d)$ after computing $S \cdot A$), separating t from d. Multiple response regression, arises, for example, in the RLSC application above.

Proof Techniques **Theorem 24.1.1.** As noted in Theorem 24.1.2, there are some OSEs for which our generalization error bound does not hold. This hints that our analysis is non-standard and cannot use generic properties of OSEs as a black box. Indeed, in our analysis, we have to consider matrix products of the form $S^\top S(UU^\top S^\top S)^k$ for our random sketching matrix S and a fixed matrix U, where k is a positive integer. We stress that it is the *same matrix* S appearing multiple times in this expression, which considerably complicates the analysis, and does not allow us to appeal to standard results on approximate matrix product (see, e.g., [Woo14a] for a survey). The key idea is to recursively reduce $S^\top S(UU^\top S^\top S)^k$ using a property of S. We use properties that only hold for specifics OSEs S: first, that each column of S is unit vector; and second, that for all pairs (i, j) and $i \neq j$, the inner product between S_i and S_j is at most $\frac{\sqrt{\log n}}{\sqrt{m}}$ with probability $1 - 1/\operatorname{poly}(n)$.

Theorems 24.8.1 and 24.9.1. To show that Count-Sketch does not give the ℓ_∞ guar-

antee, we construct a matrix A and vector b as in Figure 24.1, which has optimal solution x^* with all coordinates $1/\sqrt{d}$. We then show, for our setting of parameters, that there likely exists an index $j \in [d]$ satisfying the following property: the jth column of S has disjoint support from the kth column of S for all $k \in [d+\alpha] \setminus \{j\}$ except for a single $k > d$, for which S_j and S_k share exactly one common entry in their support. In such cases we can compute x'_j explicitly, getting $|x'_j - x^*_j| = \frac{1}{s\sqrt{\alpha}}$. By choosing suitable parameters in our construction, this gives that $\|x' - x^*\|_\infty \gg \frac{1}{\sqrt{d}}$. The lower bound for leverage score sampling follows a similar construction.

Theorem 24.5.1 and Corollary 24.5.3. The lower bound proof for the ℓ_2 guarantee uses Yao's minimax principle. We are allowed to fix an $m \times n$ sketching matrix S and design a distribution over $[A\ b]$. We first write the sketching matrix $S = U\Sigma V^\top$ in its singular value decomposition (SVD). We choose the $d+1$ columns of the adjoined matrix $[A, b]$ to be random orthonormal vectors. Consider an $n \times n$ orthonormal matrix R which contains the columns of V as its first m columns, and is completed on its remaining $n - m$ columns to an arbitrary orthonormal basis. Then $S \cdot [A, b] = V^\top RR^\top \cdot [A, b] = [U\Sigma I_m, 0] \cdot [R^\top A, R^\top b]$. Notice that $[R^\top A, R^\top b]$ is equal in distribution to $[A, b]$, since R is fixed and $[A, b]$ is a random matrix with $d+1$ orthonormal columns. Therefore, $S \cdot [A, b]$ is equal in distribution to $[U\Sigma G, U\Sigma h]$ where $[G, h]$ corresponds to the first m rows of an $n \times (d+1)$ uniformly random matrix with orthonormal columns.

A key idea is that if $n = \Omega(\max(m, d)^2)$, then by a result of Jiang [Jia06], any $m \times (d+1)$ submatrix of a random $n \times n$ orthonormal matrix has $o(1)$ total variation distance to a $d \times d$ matrix of i.i.d. $N(0, 1/n)$ random variables, and so any events that

would have occurred had G and h been independent i.i.d. Gaussians, occur with the same probability for our distribution up to an $1 - o(1)$ factor, so we can assume G and h are independent i.i.d. Gaussians in the analysis.

The optimal solution x' in the sketch space equals $(SA)^\dagger Sb$, and by using that SA has the form $U\Sigma G$, one can manipulate $\|(SA)^\dagger Sb\|$ to be of the form $\|\widetilde{\Sigma}^\dagger(\Sigma R)^\dagger \Sigma h\|_2$, where the SVD of G is $R\widetilde{\Sigma}T$. We can upper bound $\|\widetilde{\Sigma}\|_2$ by $\sqrt{r/n}$, since it is just the maximum singular value of a Gaussian matrix, where r is the rank of S, which allows us to lower bound $\|\widetilde{\Sigma}^\dagger(\Sigma R)^\dagger \Sigma h\|_2$ by $\sqrt{n/r}\|(\Sigma R)^\dagger \Sigma h\|_2$. Then, since h is i.i.d. Gaussian, this quantity concentrates to $\frac{1}{\sqrt{r}}\|(\Sigma R)^\dagger \Sigma h\|$, since $\|Ch\|^2 \approx \|C\|_F^2/n$ for a vector h of i.i.d. $N(0, 1/n)$ random variables. Finally, we can lower bound $\|(\Sigma R)^\dagger \Sigma\|_F^2$ by $\|(\Sigma R)^\dagger \Sigma R R^\top\|_F^2$ by the Pythagorean theorem, and now we have that $(\Sigma R)^\dagger \Sigma R$ is the identity, and so this expression is just equal to the rank of ΣR, which we prove is at least d. Noting that $x^* = 0$ for our instance, putting these bounds together gives $\|x' - x^*\| \geq \sqrt{d/r}$. The last ingredient is a way to ensure that the rank of S is at least d. Here we choose another distribution on inputs A and b for which it is trivial to show the rank of S is at least d with large probability. We require S be good on the mixture. Since S is fixed and good on the mixture, it is good for both distributions individually, which implies we can assume S has rank d in our analysis of the first distribution above.

24.1.2 Notation

For a positive integer, let $[n] = \{1, 2, \ldots, n\}$. For a vector $x \in \mathbb{R}^n$, define $\|x\|_2 = (\sum_{i=1}^n x_i^2)^{\frac{1}{2}}$ and $\|x\|_\infty = \max_{i \in [n]} |x_i|$. For a matrix $A \in \mathbb{R}^{m \times n}$, define $\|A\|_2 = \sup_x \|Ax\|_2/\|x\|_2$ to be the spectral norm of A and $\|A\|_F = (\sum_{i,j} A_{i,j}^2)^{1/2}$ to be the Frobenius norm of A. We

use $A^\dagger$ to denote the Moore-Penrose pseudoinverse of $m \times n$ matrix A, which if $A = U\Sigma V^\top$ is its SVD (where $U \in \mathbb{R}^{m\times n}$, $\Sigma \in \mathbb{R}^{n\times n}$ and $V \in \mathbb{R}^{n\times}$ for $m \geq n$), is given by $A^\dagger = V\Sigma^{-1}U^\top$.

In addition to $O(\cdot)$ notation, for two functions f, g, we use the shorthand $f \lesssim g$ (resp. $\gtrsim$) to indicate that $f \leq Cg$ (resp. $\geq$) for an absolute constant C. We use $f \eqsim g$ to mean $cf \leq g \leq Cf$ for constants c, C.

Definition 24.1.1 (Subspace Embedding)**.** A $(1 \pm \epsilon)$ ℓ_2-subspace embedding for the column space of an $n \times d$ matrix A is a matrix S for which for all $x \in \mathbb{R}^d$, $\|SAx\|_2^2 = (1 \pm \epsilon)\|Ax\|_2^2$.

Definition 24.1.2 (Approximate Matrix Product)**.** Let $0 < \epsilon < 1$ be a given approximation parameter. Given matrices A and B, where A and B each have n rows, the goal is to output a matrix C so that $\|A^\top B - C\|_F \leq \epsilon\|A\|_F\|B\|_F$. Typically C has the form $A^\top S^\top SB$, for a random matrix S with a small number of rows. In particular, this guarantee holds for the subsampled randomized Hadamard transform S with $O(\epsilon^{-2})$ rows [DMMS11].

24.2 Warmup: Gaussians OSEs

We first show that if S is a Gaussian random matrix, then it satisfies the generalization guarantee. This follows from the rotational invariance of the Gaussian distribution.

Theorem 24.2.1. *Suppose $A \in \mathbb{R}^{n\times d}$ has full column rank. If the entries of $S \in \mathbb{R}^{m\times n}$ are i.i.d. $N(0, 1/m)$, $m = O(d/\epsilon^2)$, then for any vectors a, b and $x^* = A^\dagger b$, we have, with probability $1 - 1/\operatorname{poly}(d)$,*

$$|a^\top (SA)^\dagger Sb - a^\top x^*| \lesssim \frac{\epsilon\sqrt{\log d}}{\sqrt{d}} \|a\|_2 \|b - Ax^*\|_2 \left\|A^\dagger\right\|_2.$$

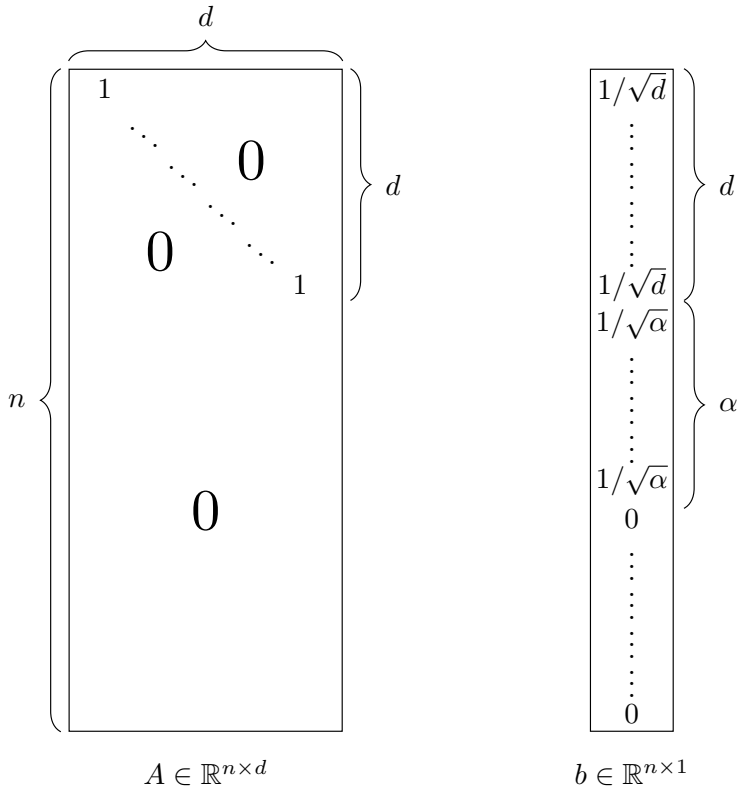


Figure 24.1: Our construction of A and b for the proof that Count-Sketch does not obey the ℓ_∞ guarantee. $\alpha < d$.

Because SA has full column rank with probability 1, $(SA)^\dagger SA = I$. Therefore

$$|a^\top (SA)^\dagger Sb - a^\top x^*| = |a^\top (SA)^\dagger S(b - Ax^*)| = |a^\top (SA)^\dagger S(b - AA^\dagger b)|.$$

Thus it suffices to only consider vectors b where $A^\dagger b = 0$, or equivalently $U^\top b = 0$. In such cases, SU will be independent of Sb, which will give the result. The proof is in Appendix 24.6.

24.3 SRHT Matrices

We first provide the definition of the subsampled randomized Hadamard transform (SRHT): let $S = \frac{1}{\sqrt{rn}} P H_n D$. Here, D is an $n \times n$ diagonal matrix with i.i.d. diagonal entries $D_{i,i}$, for which $D_{i,i}$ in uniform on $\{-1, +1\}$. The matrix H_n is the Hadamard matrix of size $n \times n$, and we assume n is a power of 2. Here, $H_n = [H_{n/2},\ H_{n/2}; H_{n/2},\ -H_{n/2}]$ and $H_1 = [1]$. The $r \times n$ matrix P samples r coordinates of an n dimensional vector uniformly at random.

For other subspace embeddings, we no longer have that SU and Sb are independent. To analyze them, we start with a claim that allows us to relate the inverse of a matrix to a power series.

Claim 24.3.1. *Let $S \in \mathbb{R}^{m \times n}$, $A \in \mathbb{R}^{n \times d}$ have SVD $A = U\Sigma V^\top$, and define $T \in \mathbb{R}^{d \times d}$ by*

$$T = I_d - U^\top S^\top S U.$$

Suppose SA has linearly independent columns and $\|T\|_2 \leq 1/2$. Then

$$(SA)^\dagger S = V\Sigma^{-1} \left(\sum_{k=0}^{\infty} T^k \right) U^\top S^\top S. \tag{24.7}$$

Proof.

$$\begin{aligned}
(SA)^\dagger S &= (A^\top S^\top S A)^{-1} A^\top S^\top S \\
&= (V\Sigma U^\top S^\top S U \Sigma V^\top)^{-1} V \Sigma U^\top S^\top S \\
&= V\Sigma^{-1} (U^\top S^\top S U)^{-1} U^\top S^\top S \\
&= V\Sigma^{-1} (I_d - T)^{-1} U^\top S^\top S \\
&= V\Sigma^{-1} \left(\sum_{k=0}^{\infty} T^k \right) U^\top S^\top S,
\end{aligned}$$

where in the last equality, since $\|T\|_2 < 1$, the von Neumann series $\sum_{k=0}^{\infty} T^k$ converges to $(I_d - T)^{-1}$. □

We then bound the kth term of this sum:

Lemma 24.3.2. Let $S \in \mathbb{R}^{r \times n}$ be the subsampled randomized Hadamard transform, and let a be a unit vector. Then with probability $1 - 1/\mathrm{poly}(n)$, we have

$$\begin{aligned}&|a^\top S^\top S(UU^\top S^\top S)^k b| \\ &= O(\log^k n) \cdot (O(d(\log n)/r) + 1)^{\frac{k-1}{2}} \cdot (\sqrt{d}\|b\|_2(\log n)/r + \|b\|_2(\log^{\frac{1}{2}} n)/r^{\frac{1}{2}})\end{aligned}$$

Hence, for r at least $d \log^{2k+2} n \log^2(n/\epsilon)/\epsilon^2$, this is at most $O(\|b\|_2 \epsilon/\sqrt{d})$ with probability at least $1 - 1/\mathrm{poly}(n)$.

We defer the proof of this lemma to the next section, and now show how the lemma lets us prove that SRHT matrices satisfy the generalization bound with high probability:

Theorem 24.3.3. *Suppose $A \in \mathbb{R}^{n \times d}$ has full column rank with $\log n = d^{o(1)}$. Let $S \in \mathbb{R}^{m \times n}$ be a subsampled randomized Hadamard transform with $m = O(d^{1+\alpha}/\epsilon^2)$ for $\alpha = \Theta(\sqrt{\frac{\log\log n}{\log d}})$. For any vectors a, b and $x^* = A^\dagger b$, we have*

$$|a^\top (SA)^\dagger Sb - a^\top x^*| \lesssim \frac{\epsilon}{\sqrt{d}} \|a\|_2 \|b - Ax^*\|_2 \left\|\Sigma^{-1}\right\|_2$$

with probability $1 - 1/\mathrm{poly}(d)$.

Proof. Define $\Delta = \Theta\left(\frac{1}{\sqrt{m}}\right) (\log^c d) \|a\|_2 \|b - Ax^*\|_2 \|\Sigma^{-1}\|_2$. For a constant $c > 0$, we have that S is a $(1 \pm \gamma)$-subspace embedding (Definition 24.1.1) for $\gamma = \sqrt{\frac{d \log^c n}{m}}$ with probability $1 - 1/\mathrm{poly}(d)$ (see, e.g., Theorem 2.4 of [Woo14a] and references therein), so $\|SUx\|_2 =$

$(1 \pm \gamma) \|Ux\|_2$ for all x, which we condition on. Hence for $T = I_d - U^\top S^\top S U$, we have $\|T\|_2 \le (1+\gamma)^2 - 1 \lesssim \gamma$. In particular, $\|T\|_2 < 1/2$ and we can apply Claim 24.3.1.

As in Section 24.2, SA has full column rank if S is a subspace embedding, so $(SA)^\dagger SA = I$ and we may assume $x^* = 0$ without loss of generality.

By the approximate matrix product (Definition 24.1.2), we have for some c that

$$|a^\top V \Sigma^{-1} U^\top S^\top S b| \le \frac{\log^c d}{\sqrt{m}} \|a\|_2 \|b\|_2 \left\|\Sigma^{-1}\right\|_2 \le \Delta \tag{24.8}$$

with $1 - 1/\mathrm{poly}(d)$ probability. Suppose this event occurs, bounding the $k = 0$ term of (24.7). Hence it suffices to show that the $k \ge 1$ terms of (24.7) are bounded by Δ.

By approximate matrix product (Definition 24.1.2), we also have with $1 - 1/d^2$ probability that

$$\left\|U^\top S^\top S b\right\|_F \le \frac{\log^c d}{\sqrt{m}} \left\|U^\top\right\|_F \|b\|_2 \le \frac{\log^c d \sqrt{d}}{\sqrt{m}} \|b\|_2 .$$

Combining with $\|T\|_2 \lesssim \gamma$ we have for any k that

$$|a^\top V \Sigma^{-1} T^k U^\top S^\top S b| \lesssim \gamma^k (\log^c d) \frac{\sqrt{d}}{\sqrt{m}} \|a\|_2 \left\|\Sigma^{-1}\right\|_2 \|b\|_2 .$$

Since this decays exponentially in k at a rate of $\gamma < 1/2$, the sum of all terms greater than k is bounded by the kth term. As long as

$$m \gtrsim \frac{1}{\epsilon^2} d^{1+\frac{1}{k}} \log^c n, \tag{24.9}$$

we have $\gamma = \sqrt{\frac{d \log^c n}{m}} < \epsilon d^{-1/(2k)} / \log^c n$, so that

$$\sum_{k' \ge k} |a^\top V \Sigma^{-1} T^{k'} U^\top S^\top S b| \lesssim \frac{\epsilon}{\sqrt{d}} \|a\|_2 \left\|\Sigma^{-1}\right\|_2 \|b\|_2 .$$

On the other hand, by Lemma 24.3.2, increasing m by a C^k factor, we have for all k that

$$|a^\top V^\top \Sigma^{-1} U^\top S^\top S (UU^\top S^\top S)^k b| \lesssim \frac{1}{2^k} \frac{\epsilon}{\sqrt{d}} \|a\|_2 \|b\|_2 \left\|\Sigma^{-1}\right\|_2$$

with probability at least $1 - 1/\operatorname{poly}(d)$, as long as $m \gtrsim d \log^{2k+2} n \log^2(d/\epsilon)/\epsilon^2$. Since the T^k term can be expanded as a sum of 2^k terms of this form, we get that

$$\sum_{k'=1}^{k} |a^\top V \Sigma^{-1} T^k U^\top S^\top S b| \lesssim \frac{\epsilon}{\sqrt{d}} \|a\|_2 \|b\|_2 \left\|\Sigma^{-1}\right\|_2$$

with probability at least $1 - 1/\operatorname{poly}(d)$, as long as $m \gtrsim d(C \log n)^{2k+2} \log^2(d/\epsilon)/\epsilon^2$ for a sufficiently large constant C. Combining with (24.9), the result holds as long as

$$m \gtrsim \frac{d \log^c n}{\epsilon^2} \max((C \log n)^{2k+2}, d^{\frac{1}{k}})$$

for any k. Setting $k = \Theta(\sqrt{\frac{\log d}{\log \log n}})$ gives the result. □

Combining Different Matrices. In some cases it can make sense to combine different matrices that satisfy the generalization bound.

Theorem 24.3.4. Let $A \in \mathbb{R}^{n \times d}$, and let $R \in \mathbb{R}^{m \times r}$ and $S \in \mathbb{R}^{r \times n}$ be drawn from distributions of matrices that are ϵ-approximate OSEs and satisfy the generalization bound (24.6). Then RS satisfies the generalization bound with a constant factor loss in failure probability and approximation factor.

We defer the details to Appendix 24.7.

24.4 Proof of Lemma 24.3.2

Proof. Each column S_i of the subsampled randomized Hadamard transform has the same distribution as $\sigma_i S_i$, where σ_i is a random sign. It also has $\langle S_i, S_i \rangle = 1$ for all i and $|\langle S_i, S_j \rangle| \lesssim \frac{\sqrt{\log(1/\delta)}}{\sqrt{r}}$ with probability $1 - \delta$, for any δ and $i \neq j$. See, e.g., [LDFU13].

By expanding the following product into a sum, and rearranging terms, we obtain

$$\begin{aligned}
& a^\top S^\top S (UU^\top S^\top S)^k b \\
= & \sum_{i_0, j_0, i_1, j_1, \cdots, i_k, j_k} a_{i_0} b_{j_k} \sigma_{i_0} \sigma_{i_1} \cdots \sigma_{i_k} \sigma_{j_0} \sigma_{j_1} \cdots \sigma_{j_k} \\
& \cdot \langle S_{i_0}, S_{j_0} \rangle (UU^\top)_{j_0, i_1} \langle S_{i_1}, S_{j_1} \rangle \cdots (UU^\top)_{j_{k-1}, i_k} \langle S_{i_k}, S_{j_k} \rangle \\
= & \sum_{i_0, j_k} a_{i_0} b_{j_k} \sigma_{i_0} \sigma_{j_k} \sum_{j_0, i_1, j_1, \cdots, i_k} \sigma_{i_1} \cdots \sigma_{i_k} \sigma_{j_0} \sigma_{j_1} \cdots \sigma_{j_{k-1}} \\
& \cdot \langle S_{i_0}, S_{j_0} \rangle (UU^\top)_{j_0, i_1} \langle S_{i_1}, S_{j_1} \rangle \cdots (UU^\top)_{j_{k-1}, i_k} \langle S_{i_k}, S_{j_k} \rangle \\
= & \sum_{i_0, j_k} \sigma_{i_0} \sigma_{j_k} Z_{i_0, j_k}
\end{aligned}$$

where Z_{i_0, j_k} is defined to be

$$Z_{i_0, j_k} = a_{i_0} b_{j_k} \sum_{\substack{i_1, \cdots i_k \\ j_0, \cdots j_{k-1}}} \prod_{c=1}^{k} \sigma_{i_c} \prod_{c=0}^{k-1} \sigma_{j_c} \cdot \prod_{c=0}^{k} \langle S_{i_c}, S_{j_c} \rangle \prod_{c=1}^{k} (UU^\top)_{i_{c-1}, j_c}$$

Note that Z_{i_0, j_k} is independent of σ_{i_0} and σ_{j_k}. We observe that in the above expression if $i_0 = j_0$, $i_1 = j_1$, $\cdots$, $i_k = j_k$, then the sum over these indices equals $a^\top (UU^\top) \cdots (UU^\top) b = 0$, since $\langle S_{i_c}, S_{j_c} \rangle = 1$ in this case for all c. Moreover, the sum over all indices conditioned on $i_k = j_k$ is equal to 0. Indeed, in this case, the expression can be factored into the form $\zeta \cdot U^\top b$, for some random variable ζ, but $U^\top b = 0$.

Let W be a matrix with $W_{i,j} = \sigma_i \sigma_j Z_{i,j}$. We need Khintchine's inequality:

Fact 24.4.1 (Khintchine's Inequality)**.** *Let $\sigma_1, \ldots, \sigma_n$ be i.i.d. sign random variables, and let $z_1, \ldots, z_n$ be real numbers. Then there are constants $C, C' > 0$ so that*

$$\Pr\left[\left|\sum_{i=1}^{n} z_i \sigma_i\right| \geq Ct\|z\|_2\right] \leq e^{-C't^2}.$$

We note that Khintchine's inequality sometimes refers to bounds on the moment of $|\sum_i z_i\sigma_i|$, though the above inequality follows readily by applying a Markov bound to the high moments.

We apply Fact 24.4.1 to each column of W, so that if W_i is the i-th column, we have by a union bound that with probability $1 - 1/\text{poly}(n)$, $\|W_i\|_2 = O(\|Z_i\|_2\sqrt{\log n})$ simultaneously for all columns i. It follows that with the same probability, $\|W\|_F^2 = O(\|Z\|_F^2 \log n)$, that is, $\|W\|_F = O(\|Z\|_F\sqrt{\log n})$. We condition on this event in the remainder.

Thus, it remains to bound $\|Z\|_F$. By squaring Z_{i_0,j_0} and using that $\mathbf{E}[\sigma_i\sigma_j] = 1$ if $i = j$ and 0 otherwise, we have,

$$\mathop{\mathbf{E}}_{\sigma}[Z_{i_0,j_k}^2] = a_{i_0}^2 b_{j_k}^2 \sum_{\substack{i_1,\cdots i_k \\ j_0,\cdots j_{k-1}}} \prod_{c=0}^{k} \langle S_{i_c}, S_{j_c}\rangle^2 \prod_{c=1}^{k} (UU^\top)_{i_{c-1},j_c}^2 \tag{24.10}$$

We defer to Appendix 24.10 the proof that

$$\mathop{\mathbf{E}}_{S}[\|Z\|_F^2] \leq (O(d(\log n)/r) + 1)^{k-1} \cdot (d\|b\|_2^2(\log^2 n)/r^2 + \|b\|_2^2(\log n)/r)$$

Note that we also have the bound:

$$(O(d(\log n)/r) + 1)^{k-1} \leq (e^{O(d(\log n)/r)})^{k-1} \leq e^{O(kd(\log n)/r)} \leq O(1)$$

for any $r = \Omega(kd \log n)$.

Having computed the expectation of $\|Z\|_F^2$, we now would like to show concentration. Consider a specific

$$Z_{i_0,j_k} = a_{i_0} b_{j_k} \sum_{i_k} \sigma_{i_k} \langle S_{i_k}, S_{j_k} \rangle \cdots \sum_{j_1} \sigma_{j_1} (UU^\top)_{j_1,i_2} \sum_{i_1} \sigma_{i_1} \langle S_{i_1}, S_{j_1} \rangle \sum_{j_0} \sigma_{j_0} \langle S_{i_0}, S_{j_0} \rangle (UU^\top)_{j_0,i_1}.$$

By Fact 24.4.1, for each fixing of i_1, with probability $1 - 1/\mathrm{poly}(n)$, we have

$$\sum_{j_0} \sigma_{j_0} \langle S_{i_0}, S_{j_0} \rangle (UU^\top)_{j_0,i_1} = O(\sqrt{\log n}) \left(\sum_{j_0} \langle S_{i_0}, S_{j_0} \rangle^2 (UU^\top)_{j_0,i_1}^2 \right)^{\frac{1}{2}}. \tag{24.11}$$

Now, we can apply Khintchine's inequality for each fixing of j_1, and combine this with (24.11). With probability $1 - 1/\mathrm{poly}(n)$, again we have

$$\begin{aligned} & \sum_{i_1} \sigma_{i_1} \langle S_{i_1}, S_{j_1} \rangle \sum_{j_0} \sigma_{j_0} \langle S_{i_0}, S_{j_0} \rangle (UU^\top)_{j_0,i_1} \\ = & \sum_{i_1} \sigma_{i_1} \langle S_{i_1}, S_{j_1} \rangle O(\sqrt{\log n}) \left(\sum_{j_0} \langle S_{i_0}, S_{j_0} \rangle^2 (UU^\top)_{j_0,i_1}^2 \right)^{\frac{1}{2}} \\ = & O(\log n) \left(\sum_{i_1} \langle S_{i_1}, S_{j_1} \rangle^2 \sum_{j_0} \langle S_{i_0}, S_{j_0} \rangle^2 (UU^\top)_{j_0,i_1}^2 \right)^{\frac{1}{2}} \end{aligned}$$

Thus, we can apply Khintchine's inequality recursively over all the $2k$ indexes $j_0, i_1, j_1, \cdots, j_{k-1}, i_k$, from which it follows that with probability $1 - 1/\mathrm{poly}(n)$, for each such i_0, j_k, we have $Z_{i_0,j_k}^2 = O(\log^k n) \mathop{\mathbf{E}}_S [Z_{i_0,j_k}^2]$, using (24.18). We thus have with this probability, that $\|Z\|_F^2 = O(\log^k n) \mathop{\mathbf{E}}_S [\|Z\|_F^2]$, completing the proof. □

24.5 Lower bound for ℓ_2 and ℓ_∞ guarantee

We prove a lower bound for the ℓ_2 guarantee, which immediately implies a lower bound for the ℓ_∞ guarantee.

Definition 24.5.1. Given a matrix $A \in \mathbb{R}^{n \times d}$, vector $b \in \mathbb{R}^n$ and matrix $S \in \mathbb{R}^{r \times n}$, denote $x^* = A^\dagger b$. We say that an algorithm $\mathcal{A}(A, b, S)$ that outputs a vector $x' = (SA)^\dagger Sb$ "succeeds" if the following property holds: $\|x' - x^*\|_2 \lesssim \epsilon \|b\|_2 \cdot \|A^\dagger\|_2 \cdot \|Ax^* - b\|_2$.

Theorem 24.5.1. Suppose Π is a distribution over $\mathbb{R}^{m \times n}$ with the property that for any $A \in \mathbb{R}^{n \times d}$ and $b \in \mathbb{R}^n$, $\Pr_{S \sim \Pi}[\mathcal{A}(A, b, S) \text{ succeeds }] \geq 19/20$. Then $m \gtrsim \min(n, d/\epsilon^2)$.

Proof. The proof uses Yao's minimax principle. Let $\mathcal{D}$ be an arbitrary distribution over $\mathbb{R}^{n \times (d+1)}$, then $\mathop{\mathbb{E}}_{(A,b) \sim \mathcal{D}} \mathop{\mathbb{E}}_{S \sim \Pi}[\mathcal{A}(A, b, S) \text{ succeeds }] \geq 1 - \delta$. Switching the order of probabilistic quantifiers, an averaging argument implies the existence of a fixed matrix $S_0 \in \mathbb{R}^{m \times n}$ such that

$$\mathop{\mathbb{E}}_{(A,b) \sim \mathcal{D}}[\mathcal{A}(A, b, S_0) \text{ succeeds }] \geq 1 - \delta.$$

Thus, we must construct a distribution $\mathcal{D}_{\text{hard}}$ such that

$$\mathop{\mathbb{E}}_{(A,b) \sim \mathcal{D}_{\text{hard}}}[\mathcal{A}(A, b, S_0) \text{ succeeds }] \geq 1 - \delta,$$

cannot hold for any $\Pi_0 \in \mathbb{R}^{m \times n}$ which does not satisfy $m = \Omega(d/\epsilon^2)$. The proof can be split into three parts. First, we prove a useful property. Second, we prove a lower bound for the case $\text{rank}(S) \geq d$. Third, we show why $\text{rank}(S) \geq d$ is necessary.

(I) We show that $[SA, Sb]$ are independent Gaussian, if both $[A, b]$ and S are or-

thonormal matrices. We can rewrite SA in the following sense,

$$\begin{aligned}
\underbrace{S}_{m\times n}\cdot\underbrace{A}_{n\times d} &= \underbrace{S}_{m\times n}\underbrace{R}_{n\times n}\underbrace{R^\top}_{n\times n}\underbrace{A}_{n\times d} \\
&= S\begin{bmatrix} S^\top & \overline{S}^\top \end{bmatrix}\begin{bmatrix} S \\ \overline{S} \end{bmatrix} A \\
&= \begin{bmatrix} I_m & 0 \end{bmatrix}\begin{bmatrix} S \\ \overline{S} \end{bmatrix} A \\
&= \begin{bmatrix} I_m & 0 \end{bmatrix}\underbrace{\widetilde{A}}_{n\times d} \\
&= \underbrace{\widetilde{A}_m}_{m\times d}
\end{aligned} \tag{24.12}$$

where $\overline{S}$ is the complement of the orthonormal basis S, I_m is a $m \times m$ identity matrix, and $\widetilde{A}_m$ is the left $m \times d$ submatrix of $\widetilde{A}$. Thus, using [Jia06] as long as $m = o(\sqrt{n})$ (because of $n = \Omega(d^3)$) the total variation distance between $[SA, Sb]$ and a random Gaussian matrix is small, i.e.,

$$D_{TV}([SA, Sb], H) \leq 0.01 \tag{24.13}$$

where each entry of H is i.i.d. Gaussian $\mathcal{N}(0, 1/n)$.

(II) Here we prove the theorem in the case when S has rank $r \geq d$ (we will prove this is necessary in part III. Writing $S = U\Sigma V^\top$ in its SVD, we have

$$\underbrace{S}_{m\times n}A = \underbrace{U}_{m\times r}\underbrace{\Sigma}_{r\times r}\underbrace{V^\top}_{r\times n}RR^\top A = U\Sigma G \tag{24.14}$$

where $R = \begin{bmatrix} V & \overline{V} \end{bmatrix}$. By a similar argument in Equation (24.12), as long as $r = o(\sqrt{n})$ we have that G also can be approximated by a Gaussian matrix, where each entry is sampled from i.i.d. $\mathcal{N}(0, 1/n)$. Similarly, $Sb = U\Sigma h$, where h also can be approximated by a Gaussian matrix, where each entry is sampled from i.i.d. $\mathcal{N}(0, 1/n)$.

Since U has linearly independent columns, $(U\Sigma G)^\dagger U\Sigma h = (\Sigma G)^\dagger U^\top U\Sigma h = (\Sigma G)^\dagger \Sigma h$.

The $r \times d$ matrix G has SVD $G = \underbrace{R}_{r\times d}\underbrace{\widetilde{\Sigma}}_{d\times d}\underbrace{T}_{d\times d}$, and applying the pseudo-inverse property again, we have

$$\begin{aligned}
\|(SA)^\dagger Sb\|_2 &= \|(\Sigma G)^\dagger \Sigma h\|_2 \\
&= \|(\Sigma R\widetilde{\Sigma}T)^\dagger \Sigma h\|_2 \\
&= \|T^\dagger(\Sigma R\widetilde{\Sigma})^\dagger \Sigma h\|_2 \\
&= \|(\Sigma R\widetilde{\Sigma})^\dagger \Sigma h\|_2 \\
&= \|\widetilde{\Sigma}^\dagger(\Sigma R)^\dagger \Sigma h\|_2,
\end{aligned}$$

where the the first equality follows by Equation (24.14), the second equality follows by the SVD of G, the third and fifth equality follow by properties of the pseudo-inverse[1] when T has orthonormal rows and $\widetilde{\Sigma}$ is a diagonal matrix, and the fourth equality follows since $\|T^\dagger\|_2 = 1$ and T is an orthonormal basis.

Because each entry of $G = R\widetilde{\Sigma}T \in \mathbb{R}^{r\times d}$ is sampled from an i.i.d. Gaussian $\mathcal{N}(0,1)$, using the result of [Ver10] we can give an upper bound for the maximum singular value of G: $\|\widetilde{\Sigma}\| \lesssim \sqrt{\frac{r}{n}}$ with probability at least .99. Thus,

$$\begin{aligned}
\|\widetilde{\Sigma}^\dagger(\Sigma R)^\dagger \Sigma h\|_2 &\geq \sigma_{\min}(\widetilde{\Sigma}^\dagger)\cdot\|(\Sigma R)^\dagger \Sigma h\|_2 \\
&= \frac{1}{\sigma_{\max}(\widetilde{\Sigma})}\|(\Sigma R)^\dagger \Sigma h\|_2 \\
&\gtrsim \sqrt{n/r}\|(\Sigma R)^\dagger \Sigma h\|_2.
\end{aligned}$$

[1] https://en.wikipedia.org/wiki/Moore-Penrose_pseudoinverse

Because h is a random Gaussian vector which is independent of $(\Sigma R)^\dagger \Sigma$, by Claim 24.5.2, $\mathbb{E}_h[\|(\Sigma R)^\dagger \Sigma h\|_2^2] = \frac{1}{n} \cdot \|(\Sigma R)^\dagger \Sigma\|_F^2$, where each entry of h is sampled from i.i.d. Gaussian $\mathcal{N}(0, 1/n)$. Then, using the Pythagorean Theorem,

$$\begin{aligned}
\|(\Sigma R)^\dagger \Sigma\|_F^2 &= \|(\Sigma R)^\dagger \Sigma R R^\top\|_F^2 + \|(\Sigma R)^\dagger \Sigma (I - RR^\top)\|_F^2 \\
&\geq \|(\Sigma R)^\dagger \Sigma R R^\top\|_F^2 \\
&= \|(\Sigma R)^\dagger \Sigma R\|_F^2 \\
&= \text{rank}(\Sigma R) \\
&= \text{rank}(SA) \\
&= d.
\end{aligned}$$

Thus, $\|x' - x^*\|_2 \gtrsim \sqrt{d/r} \geq \sqrt{d/m} = \epsilon$.

(III) Now we show that we can assume that $\text{rank}(S) \geq d$.

We sample A, b based on the following distribution $\mathcal{D}_{\text{hard}}$: with probability $1/2$, A, b are sampled from $\mathcal{D}_1$; with probability $1/2$, A, b are sampled from $\mathcal{D}_2$. In distribution $\mathcal{D}_1$, A is a random orthonormal basis and d is always orthogonal to A. In distribution $\mathcal{D}_2$, A is a $d \times d$ identity matrix in the top-d rows and 0s elsewhere, while b is a random unit vector. Then, for any (A, b) sampled from $\mathcal{D}_1$, S needs to work with probability at least $9/10$. Also for any (A, b) sampled from $\mathcal{D}_2$, S needs to work with probability at least $9/10$. The latter two statements follow since overall S succeeds on $\mathcal{D}_{\text{hard}}$ with probability at least $19/20$.

Consider the case where A, b are sampled from distribution $\mathcal{D}_2$. Then $x^* = b$ and $\text{OPT} = 0$. Then consider x' which is the optimal solution to $\min_x \|SAx - Sb\|_2^2$, so $x' = (SA)^\dagger Sb = (S_L)^\dagger S_L b$, where S can be decomposed into two matrices $S_L \in \mathbb{R}^{r \times d}$ and $S_R \in$

$\mathbb{R}^{r\times(n-d)}$, $S = \begin{bmatrix} S_L & S_R \end{bmatrix}$. Plugging x' into the original regression problem, $\|Ax' - b\|_2^2 = \|A(S_L)^\dagger S_L b - b\|_2^2$, which is at most $(1+\epsilon)\,\mathrm{OPT} = 0$. Thus $\mathrm{rank}(S_L)$ is d. Since S_L is a submatrix of S, the rank of S is also d. □

It remains to define several tools which are used in the main proof of the lower bound.

Claim 24.5.2. *For any matrix $A \in \mathbb{R}^{n\times d}$, if each entry of a vector $g \in \mathbb{R}^d$ is chosen from an i.i.d Gaussian $\mathcal{N}(0,\sigma^2)$, then $\mathop{\mathbb{E}}_g[\|Ag\|_2^2] = \sigma^2\|A\|_F^2$.*

Proof.

$$\begin{aligned}
\mathop{\mathbb{E}}_g[\|Ag\|_2^2] &= \mathop{\mathbb{E}}_g\left[\sum_{i=1}^n (\sum_{j=1}^d A_{ij}g_j)^2\right] \\
&= \mathop{\mathbb{E}}_g\left[\sum_{i=1}^n (\sum_{j=1}^d A_{ij}^2 g_j^2 + \sum_{j\neq j'} A_{ij}A_{ij'}g_j g_{j'})\right] \\
&= \sum_{i=1}^n \sum_{j=1}^d A_{ij}^2\sigma^2 \\
&= \sigma^2\|A\|_F^2.
\end{aligned}$$

□

Let $g_1, g_2, \cdots, g_t$ be i.i.d. $\mathcal{N}(0,1)$ random variables. The random variables $\sum_{i=1}^t g_i^2$ are $\mathcal{X}^2$ with t degree of freedom. Furthermore, the tail bounds are known (see Lemma A.1.6).

Definition 24.5.2. Given a matrix $A \in \mathbb{R}^{n\times d}$, vector $b \in \mathbb{R}^n$ and matrix $S \in \mathbb{R}^{r\times n}$, denote $x^* = A^\dagger b$. We say that an algorithm $\mathcal{B}(A,b,S)$ that outputs a vector $x' = (SA)^\dagger Sb$ "succeeds" if the following property holds:

$$\|x' - x^*\|_\infty \lesssim \frac{\epsilon}{\sqrt{d}}\|b\|_2 \cdot \|A^\dagger\|_2 \cdot \|Ax^* - b\|_2.$$

Applying $\|x' - x\|_\infty \geq \frac{1}{\sqrt{d}}\|x' - x\|_2$ to Theorem 24.5.1 ,we obtain the ℓ_∞ lower bound as a corollary,

Corollary 24.5.3. *Suppose* Π *is a distribution over* $\mathbb{R}^{m\times n}$ *with the property that for any* $A \in \mathbb{R}^{n\times d}$ *and* $b \in \mathbb{R}^n$,

$$\Pr_{S\sim\Pi}[\mathcal{B}(A, b, S) \text{ succeeds }] \geq 9/10.$$

Then $m \gtrsim \min(n, d/\epsilon^2)$.

24.6 Proof for Gaussian case

Lemma 24.6.1. *If the entries of* $S \in \mathbb{R}^{m\times n}$ *are i.i.d.* $N(0, 1/m)$, $m = O(d/\epsilon^2)$, *and* $U^\top b = 0$, *then*

$$|a^\top (SA)^\dagger Sb| \lesssim \frac{\epsilon\sqrt{\log d}}{\sqrt{d}} \|a\|_2 \|b\|_2 \left\|\Sigma^{-1}\right\|_2$$

for any vectors a, b *with probability* $1 - 1/\mathrm{poly}(d)$.

Proof. With probability 1, the matrix SA has linearly independent columns, and so $(SA)^\dagger$ is

$$\begin{aligned}
&= (A^\top S^\top SA)^{-1} A^\top S^\top \\
&= (V\Sigma U^\top S^\top SU\Sigma V^\top)^{-1} V\Sigma U^\top S^\top \\
&= V\Sigma^{-1}(U^\top S^\top SU)^{-1}\Sigma^{-1} V^\top V\Sigma U^\top S^\top \\
&= V\Sigma^{-1}(U^\top S^\top SU)^{-1} U^\top S^\top.
\end{aligned}$$

Hence, we would like to bound

$$X = a^\top V\Sigma^{-1}(U^\top S^\top SU)^{-1} U^\top S^\top Sb.$$

It is well-known (stated, for example, explicitly in Theorem 2.3 of [Woo14a]) that with probability $1-\exp(-d)$, the singular values of SU are $(1\pm\epsilon)$ for $m = O(d/\epsilon^2)$. We condition on this event. It follows that

$$\begin{aligned} & \|V\Sigma^{-1}(U^\top S^\top SU)^{-1}U^\top S\|_2 \\ = \; & \|\Sigma^{-1}(U^\top S^\top SU)^{-1}U^\top S\|_2 \\ \leq \; & \|\Sigma^{-1}\|_2\|(U^\top S^\top SU)^{-1}\|_2\|U^\top S\|_2 \\ \leq \; & \|\Sigma^{-1}\|_2 \cdot \frac{1}{1-\epsilon} \cdot (1+\epsilon) \\ = \; & O(\|\Sigma^{-1}\|_2), \end{aligned}$$

where the first equality uses that V is a rotation, the first inequality follows by sub-multiplicativity, and the second inequality uses that the singular values of SU are in the range $[1-\epsilon, 1+\epsilon]$. Hence, with probability $1-\exp(-d)$,

$$\|a^\top V\Sigma^{-1}(U^\top S^\top SU)^{-1}U^\top S^\top\|_2 = O(\|\Sigma^{-1}\|_2\|a\|_2). \tag{24.15}$$

The main observation is that since $U^\top b = 0$, SU is statistically independent from Sb. Hence, Sb is distributed as $N(0, \|b\|_2^2 I_m)$, conditioned on the vector $a^\top V\Sigma^{-1}(U^\top S^\top SU)^{-1}U^\top S^\top$. It follows that conditioned on the value of $a^\top V\Sigma^{-1}(U^\top S^\top SU)^{-1}U^\top S^\top$, X is distributed as

$$N(0, \|b\|_2^2\|a^\top V\Sigma^{-1}(U^\top S^\top SU)^{-1}U^\top S^\top\|_2^2/m),$$

and so using (24.15) , with probability $1 - 1/\mathrm{poly}(d)$, we have

$$|X| = O(\epsilon\sqrt{\log d}\|a\|_2\|b\|_2\|\Sigma^{-1}\|_2/\sqrt{d}).$$

□

24.7 Combining Different Matrices

In some cases it can make sense to combine different matrices that satisfy the generalization bound.

Theorem 24.3.4. Let $A \in \mathbb{R}^{n \times d}$, and let $R \in \mathbb{R}^{m \times r}$ and $S \in \mathbb{R}^{r \times n}$ be drawn from distributions of matrices that are ϵ-approximate OSEs and satisfy the generalization bound (24.6). Then RS satisfies the generalization bound with a constant factor loss in failure probability and approximation factor.

Proof. For any vectors a, b, and $x^* = A^\dagger b$ we want to show

$$|a^\top (RSA)^\dagger RSb - a^\top x^*| \lesssim \frac{\epsilon}{\sqrt{d}} \|a\|_2 \|b - Ax^*\|_2 \left\|A^\dagger\right\|_2$$

As before, it suffices to consider the $x^* = 0$ case. We have with probability $1 - \delta$ that

$$|a^\top (SA)^\dagger Sb| \lesssim \frac{\epsilon}{\sqrt{d}} \|a\|_2 \|b\|_2 \left\|A^\dagger\right\|_2;$$

suppose this happens. We also have by the properties of R, applied to SA and Sb, that

$$|a^\top (RSA)^\dagger RSb - a^\top (SA)^\dagger Sb| \lesssim \frac{\epsilon}{\sqrt{d}} \|a\|_2 \|Sb\|_2 \left\|(SA)^\dagger\right\|_2.$$

Because S is an OSE, we have $\|Sb\|_2 \leq (1 + \epsilon)$ and $\left\|(SA)^\dagger\right\|_2 \gtrsim (1 - \epsilon) \left\|A^\dagger\right\|_2$. Therefore

$$|a^\top (RSA)^\dagger RSb| \lesssim \frac{\epsilon}{\sqrt{d}} \|a\|_2 \|b\|_2 \left\|A^\dagger\right\|_2$$

□

We describe a few of the applications of combining sketches.

24.7.1 Removing dependence on n via Count-Sketch

One of the limitations of the previous section is that the choice of k depends on n. To prove that theorem, we have to assume that $\log d > \log\log n$. Here, we show an approach to remove that assumption.

The main idea is instead of applying matrix $S \in \mathbb{R}^{m \times n}$ to matrix $A \in \mathbb{R}^{n \times d}$ directly, we pick two matrices $S \in \mathbb{R}^{m \times \mathrm{poly}(d)}$ and $C \in \mathbb{R}^{\mathrm{poly}(d) \times n}$, e.g. S is FastJL matrix and C is Count-Sketch matrix with $s = 1$. We first compute $C \cdot A$, then compute $S \cdot (CA)$. The benefit of these operations is S only needs to multiply with a matrix (CA) that has $\mathrm{poly}(d)$ rows, thus the assumption we need is $\log d > \log\log(\mathrm{poly}(d))$ which is always true. The reason for choosing C as a Count-Sketch matrix with $s = 1$ is: (1) $\mathrm{nnz}(CA) \leq \mathrm{nnz}(A)$ (2) The running time is $O(\mathrm{poly}(d) \cdot d + \mathrm{nnz}(A))$.

24.7.2 Combining Gaussians and SRHT

By combining Gaussians with SRHT matrices, we can embed into the optimal dimension $O(d/\epsilon^2)$ with fast $\widetilde{O}(nd \log n + d^\omega/\epsilon^4)$ embedding time.

24.7.3 Combining all three

By taking Gaussians times SRHT times Count-Sketch, we can embed into the optimal dimension $O(d/\epsilon^2)$ with fast $O(\mathrm{nnz}(A) + d^4 \mathrm{poly}(\frac{1}{\epsilon}, \log d))$ embedding time.

24.8 Count-Sketch does not obey the ℓ_∞ guarantee

Here we demonstrate an A and a b such that Count-Sketch will not satisfy the ℓ_∞ guarantee with constant probability, so such matrices cannot satisfy the generalization guar-

antee (24.6) with high probability.

Theorem 24.8.1. Let $S \in \mathbb{R}^{m \times n}$ be drawn as a Count-Sketch matrix with s nonzeros per column. There exists a matrix $A \in \mathbb{R}^{n \times d}$ and $b \in \mathbb{R}^n$ such that, if $s^2 d \lesssim m \lesssim \sqrt{d^3 s}$, then the "true" solution $x^* = A^\dagger b$ and the approximation $x' = (SA)^\dagger Sb$ have large ℓ_∞ distance with constant probability:

$$\|x' - x^*\|_\infty \gtrsim \sqrt{\frac{d}{ms}} \|b\|_2 .$$

Plugging in $m = d^{1.5}$ and $s = d^{0.25}$ we find that

$$\|x' - x^*\|_\infty \gtrsim 1/d^{3/8} \|b\|_2 \gg 1/\sqrt{d} \|b\|_2 ,$$

even though such a matrix is an OSE with probability exponential in s. Therefore there exists a constant c for which this matrix does not satisfy the generalization guarantee (24.6) with $1 - \frac{c}{d}$ probability.

Proof. We choose the matrix A to be the identity on its top d rows: $A = \begin{bmatrix} I_d \\ 0 \end{bmatrix}$. Choose some $\alpha \geq 1$, set the value of the first d coordinates of vector b to be $\frac{1}{\sqrt{d}}$ and set the value to be $1/\sqrt{\alpha}$ for the next α coordinates, with the remaining entries all zero. Note that $\|b\|_2 = \sqrt{2}$, $x^* = (1/\sqrt{d}, \ldots, 1/\sqrt{d})$, and $\|Ax^* - b\|_2 = 1$.

Let S_k denote the kth column vector of matrix $S \in \mathbb{R}^{m \times n}$. We define two events, Event I, $\forall k' \in [d]$ and $k' \neq k$, we have $\text{supp}(S_{k'}) \cap \text{supp}(S_k) = \emptyset$; Event II, $\exists$ a unique $k' \in \{d+1, d+2, \cdots, d+\alpha\}$ such that $|\text{supp}(S_{k'}) \cap \text{supp}(S_k)| = 1$, and all other k' have $\text{supp}(S_{k'}) \cap \text{supp}(S_k) = \emptyset$. Using Claim 24.8.2, with probability at least .99 there exists a k for which both events hold.

Given the constructions of A and b described early, it is obvious that

$$Ax - b = \left[x_1 - \tfrac{1}{\sqrt{d}}, \cdots, x_d - \tfrac{1}{\sqrt{d}}, -\tfrac{1}{\sqrt{\alpha}}, \cdots, -\tfrac{1}{\sqrt{\alpha}}, 0, \cdots, 0\right]^\top.$$

Conditioned on event I and II are holding, then denote $\text{supp}(S_j) = \{i_1, i_2, \cdots, i_s\}$. Consider the terms involving x_j in the quadratic form

$$\min_x \|SAx - Sb\|_2^2.$$

it can be written as $(s-1)(x_j - 1/\sqrt{d})^2 + (x_j - 1/\sqrt{d} \pm 1/\sqrt{\alpha})^2$. Hence the optimal x' will have $x'_j = \frac{1}{\sqrt{d}} \pm \frac{1}{s\sqrt{\alpha}}$, which is different from the desired $1/\sqrt{d}$ by $\frac{1}{s\sqrt{\alpha}}$. Plugging in our requirement of $\alpha \approx m^2/(s^3d^2)$, we have

$$\|x' - x^*\|_\infty \geq \frac{1}{s\sqrt{\alpha}} \gtrsim c\sqrt{\frac{sd^2}{m^2}} \gtrsim \frac{1}{\sqrt{d}}$$

where the last inequality follows by $m \lesssim \sqrt{sd^3}$. Thus, we get the result. □

Claim 24.8.2. *If $m = \Omega(s^2 d)$, $m = o(d^2)$, $\alpha < d$, and $\alpha = O(\frac{m^2}{s^3 d^2})$, with probability at least .99 there exists a $k \in [d]$ for which both event I and II hold.*

Proof. If $m = \Omega(s^2 d)$, then for any i in $\{1, 2, ..., d\}$, let X_i be an indicator that the entries of column i are disjoint from all i' in $[d]\backslash\{i\}$. Then $\mathbb{E}[X_i] \geq .9999$, so by Markov's inequality, with probability .99, we have $.99d$ columns having this property (indeed, the expected value of $d - X$ is at most $.0001d$, so $\Pr[d - X \geq .01d] \leq \frac{\mathbb{E}[d-X]}{.01d} \leq \frac{.0001d}{.01d} = .01$). Define Event E to be that $.99d$ columns of first d columns have the property that the entries of that column

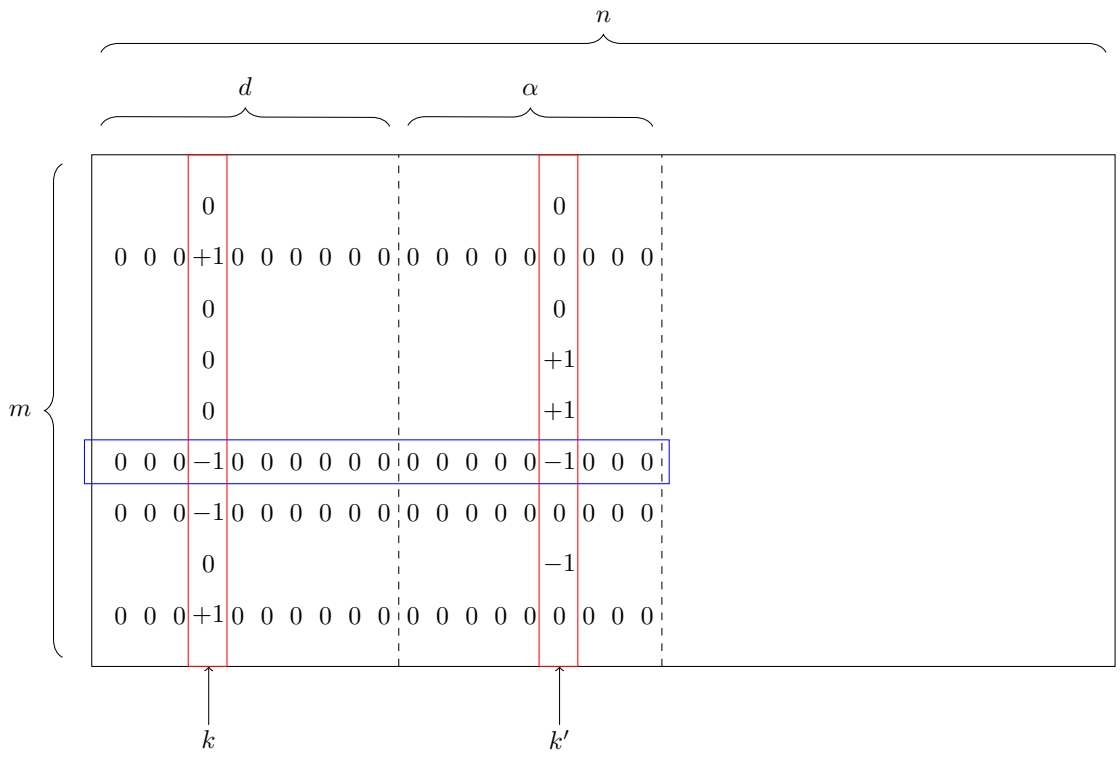


Figure 24.2: Count Sketch matrix $S \in \mathbb{R}^{m \times n}$. Event I, for any $k' \in [d]$ and $k' \neq k$, $\text{supp}(S_k) \cap \text{supp}(S_{k'}) = \emptyset$. Event II, there exists a unique $k' \in \{d+1, d+2, \cdots, d+\alpha\}$ such that S_k and $S_{k'}$ intersect at exactly one location(row index).

are disjoint from all the other $d-1$ columns. Let S be the set of these $.99d$ columns. Let N be the union of supports of columns in S.

Each column i in $\{d+1, ..., d+\alpha\}$ chooses s non-zero entries. Define event $F($ which is similar as event $E)$ to be that $.99\alpha$ columns of the next α columns have the property that the entries of that column are disjoint from all the other $\alpha - 1$ columns. By the same argument, since $\alpha < d$, with probability .99, we have $.99\alpha$ columns in $\{d+1, ..., d+\alpha\}$ being disjoint from other columns in $\{d+1, ..., d+\alpha\}$. Condition on event F holding. Let L be the multiset union of supports of all columns in $\{d+1, ..., d+\alpha\}$. Then L has size $\alpha \cdot s$. Let

M be the union of supports of all columns in $\{d+1, ..., d+\alpha\}$, that is, the set union rather than the multiset union. Note that $|M| \geq .99\alpha \cdot s$ because of $.99\alpha$ columns are disjoint from each other.

The intersection size x of N and M is hyper-geometrically distributed with expectation

$$\mathbb{E}[x] = \frac{s|S| \cdot |M|}{m}.$$

By a lower tail bound for the hypergeometric distribution [2] ,

$$\Pr[x \leq (p-t)n] \leq \exp(-2t^2 n),$$

where $p = s \cdot |S|/m$ and $n = |M|$, so

$$\Pr[x \leq \mathbb{E}[x] - t \cdot |M|] \leq \exp(-2t^2 \cdot |M|) \leq 0.01,$$

where the last inequality follows by setting $t = \Theta(1/\sqrt{|M|})$. Thus, we get with probability .99, the intersection size is at least $\frac{s|S| \cdot |M|}{m} - \Theta(\sqrt{|M|})$.

Now let W be the distinct elements in $L \backslash M$, so necessarily $|W| \leq .01\alpha \cdot s$. By an upper tail bound for the hypergeometric distribution, the intersection size y of W and N satisfies

$$\Pr[y \geq (p+t)n] \leq \exp(-2t^2 n),$$

where $p = s \cdot |S|/m$ and $n = |W|$, we again get

$$\Pr[y \geq \mathbb{E}[y] + t \cdot |W|] \leq \exp(-2t^2 \cdot |W|).$$

[2] https://en.wikipedia.org/wiki/Hypergeometric_distribution

If $|W| = 0$, then $y = 0$. Otherwise, we can set $t = \Theta(1/\sqrt{|W|})$ so that this probability is less than .01, and we get with probability .99, the intersection size y is at most $s \cdot |S| \cdot |W|/m + \Theta(\sqrt{|W|})$. Note that we have that $\Theta(\sqrt{|M|})$ and $\Theta(\sqrt{|W|})$ are bounded by $\Theta(\sqrt{s \cdot \alpha})$. Setting $\alpha = O(\frac{m^2}{s^3 d^2})$ suffices to ensure y is at most $(1.01)s \cdot |S| \cdot |W|/m$, and earlier that x is at least $.99 \cdot s \cdot |S| \cdot |M|/m$.

The probability one of the $|S|$ blocks in N has two or more intersections with M is less than $\binom{x}{2}$ times the probability two random distinct items in the intersection land in the block. This probability is

$$\frac{\binom{x}{2} \cdot \binom{s}{2}}{\binom{s \cdot |S|}{2}} = \Theta(x^2/|S|^2) = \Theta(x^2/d^2) = \Theta(m^2/(d^4 s^2)).$$

So the expected number of such blocks is $\Theta(m^2 sd/(d^4 s^2)) = \Theta(m^2/(d^3 s))$ which is less than $(.99 \cdot s \cdot |S| \cdot |M|)/(2m) \leq X/2$ if $m = o(d^2)$, which we have. So, there are at least $x/2$ blocks which have intersection size exactly 1 with N. Note that the number of intersections of the $|S|$ blocks with W is at most y, which is at most $(1.01)s \cdot |S| \cdot |W|/m \leq (1.01)s \cdot |S| \cdot \frac{1}{99} \cdot |M|/m < x/2$, and therefore there exists a block, that is, a column among the first d columns, which intersects M in exactly one position and does not intersect W. This is our desired column. Thus, we complete the proof. □

24.9 Leverage score sampling does not obey the ℓ_∞ guarantee

Not only does Count-Sketch fail, but so does leverage score sampling, which is a technique that takes a subsample of rows of A with rescaling. In this section we show an A and a b such that leverage score sampling will not satisfy the ℓ_∞ guarantee. We start with a formal definition of leverage scores.

Definition 24.9.1 (Leverage Scores)**.** Given an arbitrary $n \times d$ matrix A, with $n > d$, let U denote the $n \times d$ matrix consisting of the d left singular vectors of A, let $U_{(i)}$ denote the $i-$th row of the matrix U, so $U_{(i)}$ is a row vector. Then the leverage scores of the rows of A are given by $l_i = \|U_{(i)}\|_2^2$, for $i \in [n]$.

The leverage score sampling matrix can be thought of as a square diagonal matrix $D \in \mathbb{R}^{n \times n}$ with diagonal entries chosen from some distribution. If $D_{ii} = 0$, it means we do not choose the i-th row of matrix A., If $D_{ii} > 0$, it means we choose that row of the matrix A and also rescale that row. We show that the leverage score sampling matrix cannot achieve ℓ_∞ guarantee, nor can it achieve our notion of generalization error.

Theorem 24.9.1. Let $D \in \mathbb{R}^{n \times n}$ be a leverage score sampling matrix with m nonzeros on the diagonal. There exists a matrix $A \in \mathbb{R}^{n \times d}$ and a vector $b \in \mathbb{R}^n$ such that, if $m \lesssim d\sqrt{d}$, then the "true" solution $x^* = A^\dagger b$ and the approximation $x' = (DA)^\dagger Db$ have large ℓ_∞ distance with constant probability:

$$\|x' - x^*\|_\infty \gtrsim \frac{1}{\sqrt{d}} \|b\|_2 .$$

Therefore there exists a constant c for which this matrix does not satisfy the generalization guarantee (24.6) with $1 - \frac{c}{d}$ probability.

Proof. We choose the matrix A to be the identity on its top d rows, and L scaled identity matrices $\frac{1}{\sqrt{\alpha d}} I_d$ for the next dL rows, where L satisfies $\frac{1}{d} + \frac{1}{\alpha d} L = 1$ (to normalize each column of A), which implies $L = \alpha(d-1)$. Choose some $\beta \in [1, d)$. Set the value of the first d coordinates of vector b to be $\frac{1}{\sqrt{d}}$ and set the value to be $\frac{1}{\sqrt{\beta}}$ for the next β coordinates, with the remaining entries all zero. Note that $\|b\|_2 = \sqrt{2}$.

First, we compute $\|Ax-b\|_2^2$. Because β is less than d, there are two kinds of x_j: one involves the following term,

$$(\frac{1}{\sqrt{d}}x_j - \frac{1}{\sqrt{d}})^2 + (L-1)(\frac{1}{\sqrt{\alpha d}}x_j)^2, \tag{24.16}$$

where the optimal x_j should be set to $1/d$. The other involves the term:

$$(\frac{1}{\sqrt{d}}x_j - \frac{1}{\sqrt{d}})^2 + (\frac{1}{\sqrt{\alpha d}}x_j - \frac{1}{\sqrt{\beta}})^2 + (L-1)(\frac{1}{\sqrt{\alpha d}}x_j)^2, \tag{24.17}$$

where the optimal x_j should be set to $1/d + 1/\sqrt{\alpha\beta d}$. Because we are able to choose α, β such that $\alpha\beta \gtrsim d$, then

$$x_j = 1/d + 1/\sqrt{\alpha\beta d} \lesssim 1/d.$$

Second, we compute $\|DAx - Db\|_2^2$. With high probability, there exists a j satisfying Equation (24.17), but after applying leverage score sampling, the middle term of Equation (24.17) is removed. Let $p_1 = \frac{1}{d}$ denote the leverage score of each of the top d rows of A, and let $p_2 = \frac{1}{\alpha d}$ denote the leverage score of each of the next Ld rows of A. We need to discuss the cases $m > d$ and $m \leq d$ separately.

If $m > d$, then the following term involves x_j,

$$\begin{aligned}
& (\frac{1}{\sqrt{p_1}}\frac{1}{\sqrt{d}}x_j - \frac{1}{\sqrt{p_1}}\frac{1}{\sqrt{d}})^2 + \frac{m-d}{d}\cdot(\frac{1}{\sqrt{p_2}}\frac{1}{\sqrt{\alpha d}}x_j)^2 \\
= \; & \frac{1}{p_1}(\frac{1}{\sqrt{d}}x_j - \frac{1}{\sqrt{d}})^2 + \frac{m-d}{d}\cdot\frac{1}{p_2}(\frac{1}{\sqrt{\alpha d}}x_j)^2 \\
= \; & d\left((\frac{1}{\sqrt{d}}x_j - \frac{1}{\sqrt{d}})^2 + \frac{m-d}{d}\alpha(\frac{1}{\sqrt{\alpha d}}x_j)^2\right).
\end{aligned}$$

where the optimal x_j should be set to

$$\begin{aligned} x_j &= \frac{1/d}{1/d + (m-d)\alpha/(\alpha d^2)} \\ &= \frac{1}{1 + (m-d)/d} \\ &\gtrsim \frac{1}{(m-d)/d} \\ &\gg \frac{1}{\sqrt{d}}. \qquad \text{by } \; m \ll d\sqrt{d} \end{aligned}$$

If $m \leq d$, then the term involving x_j is $(\frac{1}{\sqrt{p_1}}\frac{1}{\sqrt{d}}x_j - \frac{1}{\sqrt{p_1}}\frac{1}{\sqrt{d}})^2$ where the optimal x_j should be set to be $1 \gg 1/\sqrt{d}$.

Third, we need to compute $\|Ax^* - b\|_2^2$ and $\sigma_{\min}(A)$. It is easy to see that $\sigma_{\min}(A)$ because A is an orthonormal matrix. The upper bound for $\|Ax^* - b\|_2^2 = 2$, and the lower bound is also a constant, which can be proved in the following way:

$$\|Ax^* - b\|_2^2 = \sum_{j=1}^{\beta}(24.16) + \sum_{j=\beta+1}^{d}(24.17) \geq d(\frac{1}{\sqrt{d}}\frac{1}{d} - \frac{1}{\sqrt{d}})^2 \gtrsim d \cdot \frac{1}{d} = 1.$$

□

24.10 Bounding $\mathbb{E}[\|Z\|_F^2]$

Before getting into the proof details, we define the key property of S being used in the rest of the proofs.

Definition 24.10.1 (All Inner Product Small(AIPS) Property)**.** For any matrix $S \in \mathbb{R}^{r \times n}$, if for all $i, j \in [n]$ with $i \neq j$ we have

$$|\langle S_i, S_j \rangle| = O(\sqrt{\log n}/\sqrt{r}),$$

we say that S satisfies the "AIPS" property.

Claim 24.10.1. *If $S \in \mathbb{R}^{r \times n}$ is a subsampled Hadamard transform matrix, then the* AIPS *property holds with probability at least* $1 - 1/\operatorname{poly}(n)$.

Proof. From the structure of S, for any $i \neq j$, we have with probability $1 - 1/\operatorname{poly}(n)$ such that $|\langle S_i, S_j \rangle| = O(\sqrt{\log n}/\sqrt{r})$. Applying a union bound over $O(n^2)$ pairs, we obtain that

$$\Pr[\text{ AIPS holds }] \geq 1 - 1/\operatorname{poly}(n).$$

□

The main idea for bounding $\mathbb{E}[\|Z\|_F^2]$ is to rewrite it as $\mathbb{E}[\|Z\|_F^2] = \mathbb{E}[\ \|Z\|_F^2 \mid \text{AIPS holds }] + \mathbb{E}[\ \|Z\|_F^2 \mid \text{AIPS does not hold }]$. Because Pr[AIPS does not hold] is at most $1/\operatorname{poly}(n)$, the first term dominates the second term, which means we only need to pay attention to the first term. We repeatedly apply this idea until all the S are removed.

We start by boundinbg $\mathbb{E}[\|Z\|_F^2]$ by squaring Z_{i_0,j_0} and using that $\mathbb{E}[\sigma_i \sigma_j] = 1$ if $i = j$ and 0 otherwise. Then, we obtain,

$$\mathop{\mathbb{E}}_{\sigma}[Z_{i_0,j_k}^2] = a_{i_0}^2 b_{j_k}^2 \sum_{i_1,\cdots i_k, j_0, \cdots j_{k-1}} \prod_{c=0}^{k} \langle S_{i_c}, S_{j_c} \rangle^2 \prod_{c=1}^{k} (UU^\top)_{i_{c-1},j_c}^2. \tag{24.18}$$

We thus have,

$$\sum_{i_0, j_k, j_k \neq i_k} a_{i_0}^2 \langle S_{i_0}, S_{j_0} \rangle^2 b_{j_k}^2 \langle S_{i_k}, S_{j_k} \rangle^2 = a_{j_0}^2 \|b\|_2^2 O((\log n)/r) + \|a\|_2^2 \|b\|_2^2 O((\log^2 n)/r^2) \stackrel{\text{def}}{=} C_{j_0},$$

where the first equality is from our conditioning, and the second equality is the definition

of C_{j_0}. Hence, $\mathop{\mathbb{E}}_{S}[\|Z\|_F^2]$ is

$$
\begin{aligned}
&= \sum_{i_1,\cdots i_k,j_0,\cdots j_{k-1}} \prod_{c=1}^{k-1} \langle S_{i_c}, S_{j_c}\rangle^2 \prod_{c=1}^{k} (UU^\top)^2_{j_{c-1},i_c} \cdot \sum_{i_0,j_k,j_k \neq i_k} a_{i_0}^2 \langle S_{i_0}, S_{j_0}\rangle^2 b_{j_k}^2 \langle S_{i_k}, S_{j_k}\rangle^2 \\
&= \sum_{i_1,\cdots i_k,j_0,\cdots j_{k-1}} \prod_{c=1}^{k-1} \langle S_{i_c}, S_{j_c}\rangle^2 \prod_{c=1}^{k} (UU^\top)^2_{j_{c-1},i_c} C_{j_0} \\
&= \sum_{i_1,\cdots i_k,j_0,\cdots j_{k-1}} \langle S_{i_{k-1}}, S_{j_{k-1}}\rangle^2 (UU^\top)^2_{j_{k-1},i_k} \cdot \prod_{c=1}^{k-2} \langle S_{i_c}, S_{j_c}\rangle^2 \prod_{c=1}^{k-1} (UU^\top)^2_{j_{c-1},i_c} C_{j_0},
\end{aligned}
$$

where the first equality follows from (24.18), the second equality by definition of C_{j_0}, and the final equality by factoring out $c = k-1$ from one product and $c = k-2$ from the other product.

The way to bound the term $\langle S_{i_{k-1}}, S_{j_{k-1}}\rangle$ is by separating the diagonal term where $i_{k-1} = j_{k-1}$ and the non-diagonal term where $i_{k-1} \neq j_{k-1}$. We now use the aforementioned property of S, namely, that $\langle S_{i_{k-1}}, S_{j_{k-1}}\rangle = 1$, if $i_{k-1} = j_{k-1}$, while for $i_{k-1} \neq j_{k-1}$, we have with probability $1 - 1/\mathrm{poly}(n)$ that $|\langle S_{i_{k-1}}, S_{j_{k-1}}\rangle| = O(\sqrt{\log n}/\sqrt{r})$ conditioned on AIPS holding.

Conditioned on AIPS holding, we can recursively reduce the number of terms in the

product:

$$
\begin{aligned}
& \|Z\|_F^2 \\
= & \sum_{i_1,\cdots i_k,j_0,\cdots j_{k-1},i_{k-1}\neq j_{k-1}} O((\log n)/r)\cdot (UU^\top)^2_{j_{k-1},i_k}\cdot \prod_{c=1}^{k-2}\langle S_{i_c},S_{j_c}\rangle^2 \prod_{c=1}^{k-1}(UU^\top)^2_{j_{c-1},i_c}C_{j_0} \\
+ & \sum_{i_1,\cdots i_k,j_0,\cdots j_{k-1},i_{k-1}=j_{k-1}} 1\cdot (UU^\top)^2_{j_{k-1},i_k}\cdot \prod_{c=1}^{k-2}\langle S_{i_c},S_{j_c}\rangle^2 \prod_{c=1}^{k-1}(UU^\top)^2_{j_{c-1},i_c}C_{j_0} \\
\leq & \sum_{i_1,\cdots i_k,j_0,\cdots j_{k-1}} O((\log n)/r)\cdot (UU^\top)^2_{j_{k-1},i_k}\cdot \prod_{c=1}^{k-2}\langle S_{i_c},S_{j_c}\rangle^2 \prod_{c=1}^{k-1}(UU^\top)^2_{j_{c-1},i_c}C_{j_0} \\
+ & \sum_{i_1,\cdots i_k,j_0,\cdots j_{k-1},i_{k-1}=j_{k-1}} 1\cdot (UU^\top)^2_{j_{k-1},i_k}\cdot \prod_{c=1}^{k-2}\langle S_{i_c},S_{j_c}\rangle^2 \prod_{c=1}^{k-1}(UU^\top)^2_{j_{c-1},i_c}C_{j_0},
\end{aligned}
$$

where the first equality follows from the property just mentioned, and the inequality follows by including back the tuples of indices for which $i_{k-1} = j_{k-1}$, using that each summand is non-negative.

Our next step will be to bound the term $(UU^\top)^2_{j_{k-1},i_k}$. We have, $\|Z\|_F^2$ is

$$\begin{aligned}
&\leq \sum_{i_k,j_{k-1}} (UU^\top)^2_{i_k,j_{k-1}} \sum_{\substack{i_1,\cdots,i_{k-1}\\ j_0,\cdots,j_{k-2}}} O((\log n)/r) \\
&\cdot \prod_{c=1}^{k-2} \langle S_{i_c}, S_{j_c}\rangle^2 \prod_{c=1}^{k-1} (UU^\top)^2_{j_{c-1},i_c} C_{j_0} \\
&+ \sum_{\substack{i_1,\cdots,i_k\\ j_0,\cdots,j_{k-1}\\ i_{k-1}=j_{k-1}}} 1 \cdot (UU^\top)^2_{j_{k-1},i_k} \prod_{c=1}^{k-2} \langle S_{i_c}, S_{j_c}\rangle^2 \prod_{c=1}^{k-1} (UU^\top)^2_{j_{c-1},i_c} C_{j_0} \\
&= \underbrace{O(d(\log n)/r) \sum_{\substack{i_1,\cdots,i_{k-1}\\ j_0,\cdots,j_{k-2}}} \prod_{c=1}^{k-2} \langle S_{i_c}, S_{j_c}\rangle^2 \prod_{c=1}^{k-1} (UU^\top)^2_{j_{c-1},i_c} C_{j_0}}_{A} \\
&+ \underbrace{\sum_{\substack{i_1,\cdots,i_k\\ j_0,\cdots,j_{k-1}\\ i_{k-1}=j_{k-1}}} 1 \cdot (UU^\top)^2_{j_{k-1},i_k} \prod_{c=1}^{k-2} \langle S_{i_c}, S_{j_c}\rangle^2 \prod_{c=1}^{k-1} (UU^\top)^2_{j_{c-1},i_c} C_{j_0}}_{B},
\end{aligned}$$

where the equality uses that $\sum_{i_k,j_{k-1}} (UU^\top)^2_{i_k,j_{k-1}} = \|UU^\top\|_F^2 = d$. We first upper bound

term B:

$$
\begin{aligned}
&= \sum_{\substack{i_1,\cdots,i_k \\ j_0,\cdots,j_{k-1} \\ i_{k-1}=j_{k-1}}} 1 \cdot (UU^\top)^2_{j_{k-1},i_k} \prod_{c=1}^{k-2} \langle S_{i_c}, S_{j_c} \rangle^2 \prod_{c=1}^{k-1} (UU^\top)^2_{j_{c-1},i_c} C_{j_0} \\
&= \sum_{\substack{i_1,\cdots,i_{k-1} \\ j_0,j_1,\cdots,j_{k-1} \\ i_{k-1}=j_{k-1}}} C_{j_0} \prod_{c=1}^{k-2} \langle S_{i_c}, S_{j_c} \rangle^2 \prod_{c=1}^{k-1} (UU^\top)^2_{j_{c-1},i_c} \sum_{i_k} (UU^\top)^2_{j_{k-1},i_k} \\
&= \sum_{\substack{i_1,\cdots,i_{k-1} \\ j_0,j_1,\cdots,j_{k-1} \\ i_{k-1}=j_{k-1}}} C_{j_0} \prod_{c=1}^{k-2} \langle S_{i_c}, S_{j_c} \rangle^2 \prod_{c=1}^{k-1} (UU^\top)^2_{j_{c-1},i_c} |e_{j_{k-1}} UU^\top|^2 \\
&\leq \sum_{\substack{i_1,\cdots,i_{k-1} \\ j_0,j_1,\cdots,j_{k-1} \\ i_{k-1}=j_{k-1}}} C_{j_0} \prod_{c=1}^{k-2} \langle S_{i_c}, S_{j_c} \rangle^2 \prod_{c=1}^{k-1} (UU^\top)^2_{j_{c-1},i_c} 1 \\
&= \sum_{\substack{i_1,\cdots,i_{k-1} \\ j_0,j_1,\cdots,j_{k-2}}} C_{j_0} \prod_{c=1}^{k-2} \langle S_{i_c}, S_{j_c} \rangle^2 \prod_{c=1}^{k-1} (UU^\top)^2_{j_{c-1},i_c},
\end{aligned}
$$

where the first equality is the definition of B, the second equality follows by separating out the index i_k, the third equality uses that $\sum_{i_k} (UU^\top)^2_{j_{k-1},i_k} = \|e_{j_{k-1}} UU^\top\|_2^2$, that is, the squared norm of the j_{k-1}-th row of $UU^\top$, the inequality follows since all rows of a projection matrix $UU^\top$ have norm at most 1, and the final equality uses that j_{k-1} no longer appears in the expression.

We now merge our bounds for the terms A and B in the following way:

$$\begin{aligned}
& \|Z\|_F^2 \\
\leq\ & A + B \\
\leq\ & O(d(\log n)/r) \sum_{\substack{i_1,\cdots,i_{k-1} \\ j_0,\cdots,j_{k-2}}} \prod_{c=1}^{k-2} \langle S_{i_c}, S_{j_c} \rangle^2 \prod_{c=1}^{k-1} (UU^\top)^2_{j_{c-1},i_c} C_{j_0} \\
+\ & \sum_{\substack{i_1,\cdots,i_{k-1} \\ j_0,j_1,\cdots,j_{k-2}}} C_{j_0} \prod_{c=1}^{k-2} \langle S_{i_c}, S_{j_c} \rangle^2 \prod_{c=1}^{k-1} (UU^\top)^2_{j_{c-1},i_c} \\
=\ & (O(d(\log n)/r) + 1) \sum_{\substack{i_1,\cdots,i_{k-1} \\ j_0,\cdots,j_{k-2}}} \prod_{c=1}^{k-2} \langle S_{i_c}, S_{j_c} \rangle^2 \prod_{c=1}^{k-1} (UU^\top)^2_{j_{c-1},i_c} C_{j_0} \\
\leq\ & \cdots \\
\leq\ & (O(d(\log n)/r) + 1)^2 \sum_{\substack{i_1,\cdots,i_{k-2} \\ j_0,\cdots,j_{k-3}}} \prod_{c=1}^{k-3} \langle S_{i_c}, S_{j_c} \rangle^2 \prod_{c=1}^{k-2} (UU^\top)^2_{j_{c-1},i_c} C_{j_0} \\
\leq\ & \cdots \\
\leq\ & (O(d(\log n)/r) + 1)^{k-1} \sum_{i_1,j_0} \prod_{c=1}^{1} (UU^\top)^2_{j_{c-1},i_c} C_{j_0} \\
\leq\ & (O(d(\log n)/r) + 1)^{k-1} \left(d\|b\|_2^2 (\log^2 n)/r^2 + \|b\|_2^2 (\log n)/r\right),
\end{aligned}$$

where the first two inequalities and first equality are by definition of A and B above. The first inequality follows by induction, since at this point we have replaced k with $k-1$, and can repeat the argument, incurring another multiplicative factor of $O(d(\log n)/r) + 1$. Repeating the induction in this way we arrive at the last inequality. Finally, the last

inequality follows by plugging in the definition of C_{j_0}, using that $\sum_{i_1,j_0}(UU^\top)^2_{j_0,i_1} = d$, and

$$\sum_{j_0,i_1}(UU^\top)^2_{j_0,i_1}a^2_{j_0} = \sum_{j_0} a^2_{j_0}\sum_{i_1}(UU^\top)^2_{j_0,i_1} = \sum_{j_0} a^2_{j_0}\|e_{j_0}UU^\top\|_2^2 \leq 1,$$

where the inequality follows since each row of $UU^\top$ has norm at most 1, and a is a unit vector. The final result is that

$$\|Z\|_F^2 \leq (O(d(\log n)/r) + 1)^{k-1}\,(d\|b\|_2^2(\log^2 n)/r^2 + \|b\|_2^2(\log n)/r).$$

Chapter 25

Symmetric Norm Regression

We provide efficient algorithms for overconstrained linear regression problems with size $n \times d$ when the loss function is a symmetric norm (a norm invariant under sign-flips and coordinate-permutations). An important class of symmetric norms are Orlicz norms, where for a function G and a vector $y \in \mathbb{R}^n$, the corresponding Orlicz norm $\|y\|_G$ is defined as the unique value α such that $\sum_{i=1}^n G(|y_i|/\alpha) = 1$. When the loss function is an Orlicz norm, our algorithm produces a $(1+\epsilon)$-approximate solution for an arbitrarily small constant $\epsilon > 0$ in input-sparsity time, improving over the previously best-known algorithm which produces a $d \cdot \operatorname{polylog} n$-approximate solution. When the loss function is a general symmetric norm, our algorithm produces a $\sqrt{d} \cdot \operatorname{polylog} n \cdot \operatorname{mmc}(\ell)$-approximate solution in input-sparsity time, where $\operatorname{mmc}(\ell)$ is a quantity related to the symmetric norm under consideration. To the best of our knowledge, this is the first input-sparsity time algorithm with provable guarantees for the general class of symmetric norm regression problem. Our results shed light on resolving the universal sketching problem for linear regression, and the techniques might be of independent interest to numerical linear algebra problems more broadly.

This part is based upon the following previous publication

- Zhao Song, Ruosong Wang, Lin F. Yang, Hongyang Zhang, Peilin Zhong
  Efficient Symmetric Norm Regression via Linear Sketching.

Manuscript 2019 [SWY+19]

25.1 Introduction

Linear regression is a fundamental problem in machine learning. For a data matrix $A \in \mathbb{R}^{n \times d}$ and a response vector $b \in \mathbb{R}^n$ with $n \gg d$, the overconstrained linear regression problem can be formulated as solving the following optimization problem:

$$\min_{x \in \mathbb{R}^d} \mathcal{L}(Ax - b), \tag{25.1}$$

where $\mathcal{L} : \mathbb{R}^n \to \mathbb{R}$ is a loss function. Via the technique of linear sketching, we have witnessed many remarkable speedups for linear regression for a wide range of loss functions. Such technique involves designing a sketching matrix $S \in \mathbb{R}^{r \times n}$, and showing that by solving a linear regression instance on the data matrix SA and the response vector Sb, which is usually much smaller in size, one can obtain an approximate solution to the original linear regression instance in (25.1). Sarlós showed in [Sar06] that by taking S as a Fast Johnson-Lindenstrauss Transform matrix [AC06], one can obtain $(1+\epsilon)$-approximate solutions to the least square regression problem ($\mathcal{L}(y) = \|y\|_2^2$) in $O(nd \log n + \mathrm{poly}(d/\epsilon))$ time. The running time was later improved to $O(\mathrm{nnz}(A) + \mathrm{poly}(d/\epsilon))$ [CW13, MM13, NN13a, LMP13, Coh16a]. Here $\mathrm{nnz}(A)$ is the number of non-zero entries in the data matrix A, which could be much smaller than nd for sparse data matrices. This technique was later generalized to other loss functions. By now, we have $\widetilde{O}(\mathrm{nnz}(A) + \mathrm{poly}(d/\epsilon))$[1] time algorithms for ℓ_p norms ($\mathcal{L}(y) = \|y\|_p^p$) [DDH+09, MM13, WZ13, CP15, WW19], the quantile loss function [YMM13], and M-estimators [CW15b, CW15a].

Despite we have successfully applied the technique of linear sketching to many different loss functions, ideally, it would be more desirable to design algorithms that work

[1] Throughout the paper, we use $\widetilde{O}(f)$ to denote $f \,\mathrm{polylog}\, f$.

Table 25.1: M-estimators

HUBER	$\begin{cases} x^2/2 & \lvert x\rvert \le c \\ c(\lvert x\rvert - c/2) & \lvert x\rvert > c \end{cases}$
$\ell_1 - \ell_2$	$2(\sqrt{1 + x^2/2} - 1)$
"FAIR"	$c^2 \left(\lvert x\rvert/c - \log(1 + \lvert x\rvert/c)\right)$

for a wide range of loss functions, instead of designing a new sketching algorithm for every specific loss function. Naturally, this leads to the following problem, which is the linear regression version of the universal sketching problem[2] studied in streaming algorithms [BO10, BCWY16]. We note that similar problems are also asked and studied for various algorithmic tasks, including principal component analysis [SWZ18], sparse recovery [NSW19b], approximate nearest neighbor search [ANN+17, ANN+18] and mean estimation with statistical queries [FGV17, LNRW19].

Question 25.1.1. *Is that possible to design sketching algorithms for linear regression, that work for a wide range of loss functions?*

Prior to our work, [CW15b, CW15a] studied this problem in terms of M-estimators, where the loss function employs the form $\mathcal{L}(y) = \sum_{i=1}^{n} G(y_i)$ for some function G. See Table 25.1 for a list a M-estimators. However, much less is known for the case where the loss function $\mathcal{L}(\cdot)$ is a norm, except for ℓ_p norms. Recently, Andoni et al. [ALS+18] tackle Problem 25.1.1 for Orlicz norms, which can be seen as a scale-invariant version of M-estimators. For a function G and a vector $y \in \mathbb{R}^n$ with $y \neq 0$, the corresponding *Orlicz*

[2] https://sublinear.info/index.php?title=Open_Problems:30.

norm $\|y\|_G$ is defined as the unique value α such that

$$\sum_{i=1}^{n} G(|y_i|/\alpha) = 1. \tag{25.2}$$

When $y = 0$, we define $\|y\|_G$ to be 0. Note that Orlicz norms include ℓ_p norms as special cases, by taking $G(z) = |z|^p$ for some $p \geq 1$. Under certain assumptions on the function G, [ALS+18] obtains the first input-sparsity time algorithm for solving Orlicz norm regression. More precisely, in $\widetilde{O}(\mathrm{nnz}(A) + \mathrm{poly}(d \log n))$ time, their algorithm obtains a solution $\widehat{x} \in \mathbb{R}^d$ such that $\|A\widehat{x} - b\|_G \leq d \cdot \mathrm{polylog}\, n \cdot \min_{x \in \mathbb{R}^d} \|Ax - b\|_G$.

There are two natural problems left open by the work of [ALS+18]. First, the algorithm in [ALS+18] has approximation ratio as large as $d \cdot \mathrm{polylog}\, n$. Although this result is interesting from a theoretical point of view, such a large approximation ratio is prohibited for machine learning applications in practice. Is it possible to obtain an algorithm that runs in $\widetilde{O}(\mathrm{nnz}(A) + \mathrm{poly}(d/\epsilon))$ time, with approximation ratio $1+\epsilon$, for arbitrarily small ϵ, similar to the case of ℓ_p norms? Moreover, although Orlicz norm includes a wide range of norms, many other important norms, e.g., top-k norms (the sum of absolute values of the leading k coordinates of a vector), max-mix of ℓ_p norms (e.g. $\max\{\|x\|_2, c\|x\|_1\}$ for some $c > 0$), and sum-mix of ℓ_p norms (e.g. $\|x\|_2 + c\|x\|_1$ for some $c > 0$), are not Orlicz norms. More complicated examples include the k-support norm [AFS12] and the box-norm [MPS14], which have found applications in sparse recovery. In light of Problem 25.1.1, it is natural to ask whether it is possible to apply the technique of linear sketching to a broader class of norms. In this paper, we obtain affirmative answers to both problems, and make progress towards finally resolving Problem 25.1.1.

Notations. For a matrix $A \in \mathbb{R}^{n \times d}$, we use $A_i \in \mathbb{R}^d$ to denote its i-th row, viewed as a column vector. For n real numbers $x_1, x_2, \cdots, x_n$, we define $\mathrm{diag}(x_1, x_2, \cdots, x_n) \in \mathbb{R}^{n \times n}$ to be the diagonal matrix where the i-th diagonal entry is x_i. For a vector $x \in \mathbb{R}^n$ and $p \geq 1$, we use $\|x\|_p$ to denote its ℓ_p norm, and $\|x\|_0$ to denote its ℓ_0 norm, i.e., the number of non-zero entries in x. For two vectors $x, y \in \mathbb{R}^n$, we use $\langle x, y \rangle$ to denote their inner product. For any $n \in \mathbb{N}_+$, we use $[n]$ to denote the set $\{1, 2, \cdots, n\}$. For $0 \leq p \leq 1$, we define $\mathrm{Ber}(p)$ to be the Bernoulli distribution with parameter p. We use $\mathbb{S}^{n-1}$ to denote the unit ℓ_2 sphere in $\mathbb{R}^n$, i.e., $\mathbb{S}^{n-1} = \{x \in \mathbb{R}^n \mid \|x\|_2 = 1\}$. We use $\mathbb{R}_{\geq 0}$ to denote the set of all non-negative real numbers, i.e., $\mathbb{R}_{\geq 0} = \{x \in \mathbb{R} \mid x \geq 0\}$.

25.1.1 Our Contributions

Algorithm for Orlicz Norms. Our first contribution is a unified algorithm which produces $(1+\epsilon)$-approximate solutions to the linear regression problem in (25.1), when the loss function $\mathcal{L}(\cdot)$ is an Orlicz norm. Before introducing our results, we first give our assumptions on the function G appeared in (25.2).

Assumption 25.1.2. *We assume the function $G : \mathbb{R} \to \mathbb{R}_{\geq 0}$ satisfies the following properties:*

1. *G is a strictly increasing convex function on $[0, \infty)$;*
2. *$G(0) = 0$, and for all $x \in \mathbb{R}$, $G(x) = G(-x)$;*
3. *There exists some $C_G > 0$, such that for all $0 < x < y$, $G(y)/G(x) \leq C_G (y/x)^2$.*

The first two conditions in Assumption 25.1.2 are necessary to make sure the corresponding Orlicz norm $\|\cdot\|_G$ is indeed a norm, and the third condition requires the function G

to have at most quadratic growth, which can be satisfied by all M-estimators in Table 25.1 and is also required by prior work [ALS+18]. Notice that our assumptions are weaker than those in [ALS+18]. In [ALS+18], it is further required that $G(x)$ is a linear function when $x > 1$, and G is twice differentiable on an interval $(0, \delta_G)$ for some $\delta_G > 0$. Given our assumptions on G, our main theorem is summarized as follow.

Theorem 25.1.3. *For a function G that satisfies Assumption 25.1.2, there exists an algorithm that, on any input $A \in \mathbb{R}^{n\times d}$ and $b \in \mathbb{R}^n$, finds a vector x^* in time $\widetilde{O}(\mathrm{nnz}(A) + \mathrm{poly}(d/\epsilon))$, such that with probability at least 0.9, $\|Ax^* - b\|_G \leq (1+\epsilon)\min_{x\in\mathbb{R}^d}\|Ax - b\|_G$.*

To the best of our knowledge, this is the first input-sparsity time algorithm with $(1+\epsilon)$-approximation guarantee, that goes beyond ℓ_p norms, the quantile loss function, and M-estimators. See Table 25.2 for a more comprehensive comparison with previous results.

Algorithm for Symmetric Norms. We further study the case when the loss function $\mathcal{L}(\cdot)$ is a symmetric norm. Symmetric norm is a more general class of norms, which includes all norms that are invariant under sign-flips and coordinate-permutations. Formally, we define symmetric norms as follow.

Definition 25.1.1. A norm $\|\cdot\|_\ell$ is called a *symmetric norm*, if

$$\|(y_1, y_2, \ldots, y_n)\|_\ell = \|(s_1 y_{\sigma_1}, s_2 y_{\sigma_2}, \ldots, s_n y_{\sigma_n})\|_\ell$$

for any permutation σ and any assignment of $s_i \in \{-1, 1\}$.

Symmetric norm includes ℓ_p norms and Orlicz norms as special cases. It also includes all examples provided in the introduction, i.e., top-k norms, max-mix of ℓ_p norms,

Table 25.2: Comparison between input-sparsity time linear regression algorithms

Reference	Loss Function	Approximation Ratio
[DDH+09, MM13, WZ13, CP15, WW19]	ℓ_p norms	$1+\epsilon$
[YMM13]	Quantile loss function	$1+\epsilon$
[CW15b, CW15a]	M-estimators	$1+\epsilon$
[ALS+18]	Orlicz norms	$d \cdot \operatorname{polylog} n$
Theorem 25.1.3	Orlicz norms	$1+\epsilon$
Theorem 25.1.4	Symmetric norms	$\sqrt{d} \cdot \operatorname{polylog} n \cdot \operatorname{mmc}(\ell)$

sum-mix of ℓ_p norms, the k-support norm [AFS12] and the box-norm [MPS14], as special cases. Understanding this general set of loss functions can be seen as a preliminary step to resolve Problem 25.1.1. Our main result for symmetric norm regression is summarized in the following theorem.

Theorem 25.1.4. *Given a symmetric norm $\|\cdot\|_\ell$, there exists an algorithm that, on any input $A \in \mathbb{R}^{n\times d}$ and $b \in \mathbb{R}^n$, finds a vector x^* in time $\widetilde{O}(\operatorname{nnz}(A) + \operatorname{poly}(d))$, such that with probability at least 0.9, $\|Ax^* - b\|_\ell \leq \sqrt{d} \cdot \operatorname{polylog} n \cdot \operatorname{mmc}(\ell) \cdot \min_{x\in\mathbb{R}^d} \|Ax - b\|_\ell$.*

In the above theorem, $\operatorname{mmc}(\ell)$ is a characteristic of the symmetric norm $\|\cdot\|_\ell$, which has been proven to be essential in streaming algorithms for symmetric norms [BBC+17]. See Definition 25.3.3 for the formal definition of $\operatorname{mmc}(\ell)$, and Section 25.3 for more details about $\operatorname{mmc}(\ell)$. In particular, for ℓ_p norms with $p \leq 2$, top-k norms with $k \geq n/\operatorname{polylog} n$, max-mix of ℓ_2 norm and ℓ_1 norm ($\max\{\|x\|_2, c\|x\|_1\}$ for some $c > 0$), sum-mix of ℓ_2 norm and ℓ_1 norm ($\|x\|_2 + c\|x\|_1$ for some $c > 0$), the k-support norm, and the box-norm, $\operatorname{mmc}(\ell)$ can all be upper bounded by $\operatorname{polylog} n$, which implies our algorithm has approximation ratio $\sqrt{d} \cdot \operatorname{polylog} n$ for all these norms. This clearly demonstrates the generality of our algorithm.

25.1.2 Technical Overview

Similar to previous works on using linear sketching to speed up solving linear regression, our core technique is to provide efficient dimensionality reduction methods for Orlicz norms and general symmetric norms. In this section, we discuss the techniques behind our results.

Row Sampling Algorithm for Orlicz Norms. Compared to prior work on Orlicz norm regression [ALS+18] which is based on random projection[3], our new algorithm is based on row sampling. For a given matrix $A \in \mathbb{R}^{n \times d}$, our goal is to output a *sparse* weight vector $w \in \mathbb{R}^n$ with at most $\mathrm{poly}(d \log n/\epsilon)$ non-zero entries, such that with high probability, for all $x \in \mathbb{R}^d$,

$$(1-\epsilon)\|Ax-b\|_G \leq \|Ax-b\|_{G,w} \leq (1+\epsilon)\|Ax-b\|_G. \tag{25.3}$$

Here, for a weight vector $w \in \mathbb{R}^n$ and a vector $y \in \mathbb{R}^n$, the *weighted Orlicz norm* $\|y\|_{G,w}$ is defined as the unique value α such that $\sum_{i=1}^n w_i G(|y_i|/\alpha) = 1$. See Definition 25.2.1 for the formal definition of weighted Orlicz norm. To obtain a $(1+\epsilon)$-approximate solution to Orlicz norm regression, by (25.3), it suffices to solve

$$\min_{x \in \mathbb{R}^d} \|Ax-b\|_{G,w}. \tag{25.4}$$

Since the vector $w \in \mathbb{R}^n$ has at most $\mathrm{poly}(d \log n/\epsilon)$ non-zero entries, and we can ignore all rows of A with zero weights, there are at most $\mathrm{poly}(d \log n/\epsilon)$ remaining rows in A in the optimization problem in (25.4). Furthermore, as we show in Lemma 25.2.1, $\|\cdot\|_{G,w}$ is

[3]Even for ℓ_p norms with $p < 2$, embeddings based on random projections will necessarily induce a distortion factor polynomial in d, as shown in [WW19].

a seminorm, which implies we can solve the optimization problem in (25.4) in $\mathrm{poly}(d \log n/\epsilon)$ time, by simply solving a convex program with size $\mathrm{poly}(d \log n/\epsilon)$. Thus, we focus on how to obtain the weight vector $w \in \mathbb{R}^n$ in the remaining part. Furthermore, by taking $\mathbf{Ab}$ to be a matrix whose first d columns are A and last column is b, to satisfy (25.3), it suffices to find a weight vector w such that for all $x \in \mathbb{R}^{d+1}$,

$$(1-\epsilon)\|\mathbf{Ab}x\|_G \leq \|\mathbf{Ab}x\|_{G,w} \leq (1+\epsilon)\|\mathbf{Ab}x\|_G. \tag{25.5}$$

Hence, we ignore the response vector b in the remaining part of the discussion.

We obtain the weight vector w via importance sampling. We compute a set of sampling probabilities $\{p_i\}_{i=1}^n$ for each row of the data matrix A, and sample the rows of A according to these probabilities. The i-th entry of the weight vector w is then set to be $w_i = 1/p_i$ with probability p_i and $w_i = 0$ with probability $1 - p_i$. However, unlike ℓ_p norms, Orlicz norms are not "entry-wise" norms, and it is not even clear that such a sampling process gives an unbiased estimation. Our key insight here is that for a vector Ax with unit Orlicz norm, if for all $x \in \mathbb{R}^d$,

$$(1-\epsilon)\sum_{i=1}^n G((Ax)_i) \leq \sum_{i=1}^n w_i G((Ax)_i) \leq (1+\epsilon)\sum_{i=1}^n G((Ax)_i), \tag{25.6}$$

then (25.5) holds, which follows from the convexity of the function G. See Lemma 25.2.5 and its proof for more details. Therefore, it remains to develop a way to define and calculate $\{p_i\}_{i=1}^n$, such that the total number of sampled rows is small.

Our method for defining and computing sampling probabilities p_i is inspired by row sampling algorithms for ℓ_p norms [DDH+09]. Here, the key is to obtain an upper bound on the contribution of each entry to the summation $\sum_{i=1}^n G((Ax)_i)$. Indeed, suppose for some

vector $u \in \mathbb{R}^n$ such that $G(Ax)_i \le u_i$ for all $x \in \mathbb{R}^d$ with $\|Ax\|_G = 1$, we can then sample each row of A with sampling probability proportional to u_i. Now, by standard concentration inequalities and a net argument, (25.6) holds with high probability. It remains to upper bound the total number of sampled rows, which is proportional to $\sum_{i=1}^n u_i$.

We use the case of ℓ_2 norm, i.e., $G(x) = x^2$, as an example to illustrate our main ideas for choosing the vector $u \in \mathbb{R}^n$. Suppose $U \in \mathbb{R}^{n \times d}$ is an orthonormal basis matrix of the column space of A, then the *leverage score*[4] is defined to be the squared ℓ_2 norm of each row of U. Indeed, leverage score gives an upper bound on the contribution of each row to $\|Ux\|_2^2$, since by Cauchy-Schwarz inequality, for each row U_i of U, we have $\langle U_i, x\rangle^2 \le \|U_i\|_2^2\|x\|_2^2 = \|U_i\|_2^2\|Ux\|_2^2$, and thus we can set $u_i = \|U_i\|_2^2$. It is also clear that $\sum_{i=1}^n u_i = d$.

For general Orlicz norms, leverage scores are no longer upper bounds on $G((Ux)_i)$. Inspired by the role of orthonormal bases in the case of ℓ_2 norm, we first define well-conditioned basis for general Orlicz norms as follow.

Definition 25.1.2. Let $\|\cdot\|_G$ be an Orlicz norm induced by a function G which satisfies Assumption 25.1.2. We say $U \in \mathbb{R}^{n \times d}$ is a well-conditioned basis with condition number $\kappa_G = \kappa_G(U)$ if for all $x \in \mathbb{R}^d$, $\|x\|_2 \le \|Ux\|_G \le \kappa_G\|x\|_2$.

Given this definition, when $\|Ux\|_G = 1$, by Cauchy-Schwarz inequality and monotonicity of G, we can show that $G((Ux)_i) \le G(\|U_i\|_2\|x\|_2) \le G(\|U_i\|_2\|Ux\|_G) \le G(\|U_i\|_2)$. This also leads to our definition of Orlicz norm leverage scores.

[4] See, e.g., [Mah11a], for a survey on leverage scores.

Definition 25.1.3. Let $\|\cdot\|_G$ be an Orlicz norm induced by a function G which satisfies Assumption 25.1.2. For a given matrix $A \in \mathbb{R}^{n\times d}$ and a well-conditioned basis U of the column space of A, the *Orlicz norm leverage score* of the i-th row of A is defined to be $G(\|U_i\|_2)$.

It remains to give an upper bound on the summation of Orlicz norm leverage scores of all rows. Unlike the ℓ_2 norm, it is not immediately clear how to use the definition of well-conditioned basis to obtain such an upper bound for general Orlicz norms. To achieve this goal, we use a novel probabilistic argument. Suppose one takes x to be a vector with i.i.d. Gaussian random variables. Then each entry of Ux has the same distribution as $\|U_i\|_2 \cdot g_i$, where $\{g_i\}_{i=1}^n$ is a set of standard Gaussian random variables. Thus, with constant probability, $\sum_{i=1}^n G((Ux)_i)$ is an upper bound on the summation of Orlicz norm leverage scores. Furthermore, by the growth condition of the function G, we have $\sum_{i=1}^n G((Ux)_i) \leq C_G\|Ux\|_G^2$. Now by Definition 25.1.2, $\|Ux\|_G \leq \kappa_G\|x\|_2$, and $\|x\|_2 \leq O(\sqrt{d})$ with constant probability by tail inequalities of Gaussian random variables. This implies an upper bound on the summation of Orlicz norm leverage scores. See Lemma 25.2.2 and its proof for more details.

Our approach for constructing well-conditioned bases is inspired by [SW11]. In Lemma 25.2.3, we show that given a subspace embedding Π which embeds the column space of A with Orlicz norm $\|\cdot\|_G$ into the ℓ_2 space with distortion κ, then one can construct a well-conditioned basis with condition number $\kappa_G \leq \kappa$. The running time is dominated by calculating ΠA and doing a QR-decomposition on ΠA. To this end, we can use the oblivi-

ous subspace embedding for Orlicz norms in Corollary 25.3.4[5] to construct well-conditioned bases. The embedding in Corollary 25.3.4 has $O(d)$ rows and $\kappa = \text{poly}(d \log n)$, and calculating ΠA can be done in $\widetilde{O}(\text{nnz}(A) + \text{poly}(d))$ time. Using such an embedding to construct the well-conditioned basis, our row sampling algorithm produces a vector w that satisfies (25.6) with $\|w\|_0 \leq \text{poly}(d \log n/\epsilon)$ in time $\widetilde{O}(\text{nnz}(A) + \text{poly}(d))$.

Oblivious Subspace Embeddings for Symmetric Norms. To obtain a faster algorithm for linear regression when the loss function is a general symmetric norm, we show that there exists a distribution over embedding matrices, such that if S is a random matrix drawn from that distribution, then for any $n \times d$ matrix A, with constant probability, for all $x \in \mathbb{R}^d$, $\|Ax\|_\ell \leq \|SAx\|_2 \leq \text{poly}(d \log n) \cdot \text{mmc}(\ell) \cdot \|Ax\|_\ell$. Moreover, the embedding matrix S is *sparse*, and calculating SA requires only $\widetilde{O}(\text{nnz}(A) + \text{poly}(d))$ time. Another favorable property of S is that it is an *oblivious subspace embeeding*, meaning the distribution of S does not depend on A. To achieve this goal, it is sufficient to construct a random diagonal matrix D such that for any fixed vector $x \in \mathbb{R}^n$,

$$\Pr[\|Dx\|_2 \geq \Omega(1/\text{poly}(d \log n)) \cdot \|x\|_\ell] \geq 1 - \exp(-\Omega(d \log n)), \tag{25.7}$$

and

$$\Pr[\|Dx\|_2 \leq \text{poly}(d \log n) \cdot \text{mmc}(\ell) \cdot \|x\|_\ell] \geq 1 - O(1/d). \tag{25.8}$$

Our construction is inspired by the sub-sampling technique in [IW05], which was used for sketching symmetric norms in data streams [BBC+17]. Throughout the discussion, we

[5] Alternatively, we can use the oblivious subspace embedding in [ALS+18] for this step. However, as we have discussed, the oblivious subspace embedding in [ALS+18] requires stronger assumptions on the function $G : \mathbb{R} \to \mathbb{R}_{\geq 0}$ than those in Assumption 25.1.2, which restricts the class of Orlicz norms to which our algorithm can be applied.

use $\xi^{(q)} \in \mathbb{R}^n$ to denote a vector with q non-zero entries and each entry is $1/\sqrt{q}$. Let us start with a special case where the vector $x \in \mathbb{R}^n$ has s non-zero entries and each non-zero entry is 1. It is easy to see $\|x\|_\ell = \sqrt{s}\|\xi^{(s)}\|_\ell$. Now consider a random diagonal matrix D which corresponds to a sampling process, i.e., each diagonal entry is set to be 1 with probability p and 0 with probability $1-p$. Our goal is to show that $\sqrt{1/p}\|\xi^{(1/p)}\|_\ell \cdot \|Dx\|_2$ is a good estimator of $\|x\|_\ell$. If $p = \Theta(d \log n/s)$, then with probability at least $1 - \exp(-\Omega(d \log n))$, Dx will contain at least one non-zero entry from x, in which case (25.7) is satisfied. However, we do not know s in advance. Thus, we use $t = O(\log n)$ different matrices $D_1, D_2, \cdots, D_t$, where D_i has sampling probability $1/2^i$. Clearly at least one such D_j can establish (25.7). For the upper bound part, if p is much smaller than $1/s$, then Dx will never contain a non-zero entry from x. Otherwise, in expectation Dx will contain ps non-zero entries, in which case our estimation will be roughly $\sqrt{s}\|\xi^{(1/p)}\|_\ell$, which can be upper bounded by $O(\log n \cdot \mathrm{mmc}(\ell) \cdot \sqrt{s}\|\xi^{(s)}\|_\ell)$. At this point, (25.8) follows from Markov's inequality. See Section 25.7.5 for the formal argument, and Section 25.3 for a detailed discussion on $\mathrm{mmc}(\ell)$.

To generalize the above argument to general vectors, for a vector $x \in \mathbb{R}^n$, we conceptually partition its entries into $\Theta(\log n)$ groups, where the i-th group contains entries with magnitude in $[2^i, 2^{i+1})$. By averaging, at least one group of entries contributes at least $\Omega(1/\log n)$ fraction to the value of $\|x\|_\ell$. To establish (25.7), we apply the lower bound part of the argument in the previous paragraph to this "contributing" group. To establish (25.8), we apply the upper bound part of the argument to all groups, which will only induce an additional $O(\log n)$ factor in the approximation ratio, by triangle inequality.

Since our oblivious subspace embedding embeds a given symmetric norm into the ℓ_2 space, in order to obtain an approximate solution to symmetric norm regression, we only need

to solve a least square regression instance with much smaller size. This is another advantage of our subspace embedding, since the least square regression problem is a well-studied problem in optimization and numerical linear algebra, for which many efficient algorithms are known, both in theory and in practice.

25.2 Linear Regression for Orlicz Norms

In this section, we introduce our results for Orlicz norm regression.

Weighted Orlicz Norm. We first give the definition of weighted Orlicz norm.

Definition 25.2.1. For a function G that satisfies Assumption 25.1.2 and a weight vector $w \in \mathbb{R}^n$ such that $w_i \geq 0$ for all $i \in [n]$, for a vector $x \in \mathbb{R}^n$, if $\sum_{i=1}^n w_i \cdot |x_i| = 0$, then the weighted Orlicz norm $\|x\|_{G,w}$ is defined to be 0. Otherwise, the weighted Orlicz norm $\|x\|_{G,w}$ is defined as the unique value $\alpha > 0$ such that $\sum_{i=1}^n w_i G(|x_i|/\alpha) = 1$.

When $w_i = 1$ for all $i \in [n]$, we have $\|x\|_{G,w} = \|x\|_G$ where $\|x\|_G$ is the (unweighted) Orlicz norm. It is well known that $\|\cdot\|_G$ is a norm. We show in the following lemma that $\|\cdot\|_{G,w}$ is a seminorm.

Lemma 25.2.1. *For a function G that satisfies Assumption 25.1.2 and a weight vector $w \in \mathbb{R}^n$ such that $w_i \geq 0$ for all $i \in [n]$, for all $x, y \in \mathbb{R}^n$, we have (i) $\|x\|_{G,w} \geq 0$, (ii) $\|x+y\|_{G,w} \leq \|x\|_{G,w} + \|y\|_{G,w}$, and (iii) $\|ax\|_{G,w} = |a| \cdot \|x\|_{G,w}$ for all $a \in \mathbb{R}$.*

Leverage Scores and Well-Conditioned Bases for Orlicz Norms. The following lemma establishes an upper bound on the summation of Orlicz norm leverage scores defined in Definition 25.1.3.

Lemma 25.2.2. *Let $\|\cdot\|_G$ be an Orlicz norm induced by a function G which satisfies Assumption 25.1.2. Let $U \in \mathbb{R}^{n \times d}$ be a well-conditioned basis with condition number κ_G as in Definition 25.1.2. Then we have $\sum_{i=1}^n G(\|U_i\|_2) \leq O(C_G d \kappa_G^2)$,*

Now we show that given a subspace embedding which embeds the column space of A with Orlicz norm $\|\cdot\|_G$ into the ℓ_2 space with distortion κ, then one can construct a well-conditioned basis with condition number $\kappa_G \leq \kappa$.

Lemma 25.2.3. *Let $\|\cdot\|_G$ be an Orlicz norm induced by a function G which satisfies Assumption 25.1.2. For a given matrix $A \in \mathbb{R}^{n\times d}$ and an embedding matrix $\Pi \in \mathbb{R}^{s\times n}$, suppose for all $x \in \mathbb{R}^d$, $\|Ax\|_G \leq \|\Pi Ax\|_2 \leq \kappa\|Ax\|_G$. Let $Q \cdot R = \frac{1}{\kappa}\Pi A$ be a QR-decomposition of $\frac{1}{\kappa}\Pi A$. Then AR^{-1} is a well-conditioned basis (see Definition 25.1.2) with $\kappa_G(AR^{-1}) \leq \kappa$.*

The following lemma shows how to estimate Orlicz norm leverage scores given a change of basis matrix $R \in \mathbb{R}^{d\times d}$, in $\widetilde{O}(\mathrm{nnz}(A) + \mathrm{poly}(d))$ time.

Lemma 25.2.4. *Let $\|\cdot\|_G$ be an Orlicz norm induced by a function G which satisfies Assumption 25.1.2. For a given matrix $A \in \mathbb{R}^{n\times d}$ and $R \in \mathbb{R}^{d\times d}$, there exists an algorithm that outputs $\{u_i\}_{i=1}^n$ such that with probability at least 0.99, $u_i = \Theta(G(\|(AR^{-1})_i\|_2))$ for all $1 \leq i \leq n$. The algorithm runs in $\widetilde{O}(\mathrm{nnz}(A) + \mathrm{poly}(d))$ time.*

The Row Sampling Algorithm. Based on the notion of Orlicz norm leverage scores and well-conditioned bases, we design a row sampling algorithm for Orlicz norms.

Lemma 25.2.5. *Let $\|\cdot\|_G$ be an Orlicz norm induced by a function G which satisfies Assumption 25.1.2. Let $U \in \mathbb{R}^{n\times d}$ be a well-conditioned basis with condition number $\kappa_G = \kappa_G(U)$ as in Definition 25.1.2. For sufficiently small ϵ and δ, and sufficiently large constant C, let $\{p_i\}_{i=1}^n$ be a set of sampling probabilities satisfying $p_i \geq \min\left\{1, C\left(\log(1/\delta) + d\log(1/\epsilon)\right)\epsilon^{-2}G\left(\|U_i\|_2\right)\right\}$. Let w be a vector whose i-th entry is set to be $w_i = 1/p_i$ with probability p_i and $w_i = 0$ with*

probability $1-p_i$*, then with probability at least* $1-\delta$*, for all* $x \in \mathbb{R}^d$*, we have* $(1-\epsilon)\|Ux\|_G \leq \|Ux\|_{G,w} \leq (1+\epsilon)\|Ux\|_G$.

Solving Linear Regression for Orlicz Norms. Now we combine all ingredients to give an algorithm for Orlicz norm regression. We use $\mathbf{Ab} \in \mathbb{R}^{n\times(d+1)}$ to denote a matrix whose first d columns are A and the last column is b. The algorithm is described in Figure 25.1, and we prove its running time and correctness in Theorem 25.2.6. We assume we are given an embedding matrix Π, such that for all $x \in \mathbb{R}^{d+1}$, $\|\mathbf{Ab}x\|_G \leq \|\Pi\mathbf{Ab}x\|_2 \leq \kappa\|\mathbf{Ab}x\|_G$. The construction of Π and the value κ will be given in Corollary 25.3.4. In Appendix 25.8.1, we use Theorem 25.2.6 and Corollary 25.3.4 to formally prove Theorem 25.1.3.

1. For the given embedding matrix Π, calculate $\Pi\mathbf{Ab}$ and invoke QR-decomposition on $\Pi\mathbf{Ab}/\kappa$ to obtain $Q \cdot R = \Pi\mathbf{Ab}/\kappa$.
2. Invoke Lemma 25.2.4 to obtain $\{u_i\}_{i=1}^n$ such that $u_i = \Theta(G(\|(\mathbf{Ab}R^{-1})_i\|_2))$.
3. For a sufficiently large constant C, let $\{p_i\}_{i=1}^n$ be a set of sampling probabilities with $p_i \geq \min\left\{1, C\cdot d\cdot\epsilon^{-2}\log(1/\epsilon)\cdot G\left(\|(\mathbf{Ab}R^{-1})_i\|_2\right)\right\}$, and w be a vector whose i-th entry $w_i = 1/p_i$ with probability p_i and $w_i = 0$ with probability $1-p_i$.
4. Calculate $x^* = \mathrm{argmin}_{x\in\mathbb{R}^d}\|Ax-b\|_{G,w}$. Return x^*.

Figure 25.1: Algorithm for Orlicz norm regression

Theorem 25.2.6. *Let* $\|\cdot\|_G$ *be an Orlicz norm induced by a function* G *which satisfies Assumption 25.1.2. Given an embedding matrix* Π*, such that for all* $x \in \mathbb{R}^d$*,* $\|\mathbf{Ab}x\|_G \leq \|\Pi\mathbf{Ab}x\|_2 \leq \kappa\|\mathbf{Ab}x\|_G$*, with probability at least* 0.9*, the algorithm in Figure 25.1 outputs* $x^* \in \mathbb{R}^d$ *in time* $\mathrm{poly}(d\kappa/\epsilon) + \mathcal{T}_{\mathrm{QR}}(\Pi\mathbf{Ab})$*, such that* $\|Ax^*-b\|_G \leq (1+\epsilon)\min_{x\in\mathbb{R}^d}\|Ax-b\|_G$*. Here,* $\mathcal{T}_{\mathrm{QR}}(\Pi\mathbf{Ab})$ *is the running time for calculating* $\Pi\mathbf{Ab}$ *and invoking QR-decomposition*

on $\Pi\mathbf{Ab}$.

25.3 Linear Regression for Symmetric Norms

In this section, we introduce SymSketch, a subspace embedding for symmetric norms.

Definition of SymSketch. We first formally define SymSketch. Due to space limitation, we give the definition of Gaussian embeddings, CountSketch embeddings and their compositions in Appendix 25.7.1.1.

Definition 25.3.1 (Symmetric Norm Sketch (SymSketch))**.** Let $t = \Theta(\log n)$. Let $\widetilde{D} \in \mathbb{R}^{n(t+1)\times n}$ be a matrix defined as $\widetilde{D} = \begin{bmatrix} w_0 D_0 \\ w_1 D_1 \\ \vdots \\ w_t D_t \end{bmatrix}$, where for each $i \in \{0, 1, \ldots, t\}$, $D_i = \mathrm{diag}(z_{i,1}, z_{i,2}, \ldots, z_{i,n}) \in \mathbb{R}^{n\times n}$ and $z_{i,j} \sim \mathrm{Ber}(1/2^i)$ for each $j \in [n]$. Moreover, $w_i = \|(\underbrace{1, 1, \ldots, 1}_{2^i \text{ ones}}, 0, \ldots, 0)\|_\ell$. Let $\Pi \in \mathbb{R}^{O(d)\times n(t+1)}$ be a composition of Gaussian embedding and CountSketch embedding (Definition 25.7.4) with $\epsilon = 0.1$, and $S = \Pi\widetilde{D}$. We say $S \in \mathbb{R}^{O(d)\times n}$ is a SymSketch.

Modulus of Concentration. Now we give the definition of $\mathrm{mmc}(\ell)$ for a symmetric norm.

Definition 25.3.2 ([BBC+17])**.** Let $\mathcal{X}$ denote the uniform distribution over $\mathbb{S}^{n-1}$. The *median* of a symmetric norm $\|\cdot\|_\ell$ is the unique value M_ℓ such that $\Pr_{x\sim\mathcal{X}}[\|x\|_\ell \geq M_\ell] \geq 1/2$ and $\Pr_{x\sim\mathcal{X}}[\|x\|_\ell \leq M_\ell] \geq 1/2$.

Definition 25.3.3 ([BBC+17])**.** For a given symmetric norm $\|\cdot\|_\ell$, we define the *modulus of concentration* to be $\mathrm{mc}(\ell) = \max_{x\in\mathbb{S}^{n-1}} \|x\|_\ell / M_\ell$, and define the *maximum modulus of concentration* to be $\mathrm{mmc}(\ell) = \max_{k\in[n]} \mathrm{mc}(\ell^{(k)})$, where $\|\cdot\|_{\ell^{(k)}}$ is a norm on $\mathbb{R}^k$ which is defined to be $\|(x_1, x_2, \cdots, x_k)\|_{\ell^{(k)}} = \|(x_1, x_2, \cdots, x_k, 0, \cdots, 0)\|_\ell$.

It has been shown in [BBC+17] that $\mathrm{mmc}(\ell) = \Theta(n^{1/2-1/p})$ for ℓ_p norms when $p > 2$, $\mathrm{mmc}(\ell) = \Theta(1)$ for ℓ_p norms when $p \leq 2$, $\mathrm{mmc}(\ell) = \widetilde{\Theta}(\sqrt{n/k})$ for top-k norms, and $\mathrm{mmc}(\ell) = O(\log n)$ for the k-support norm [AFS12] and the box-norm [MPS14]. We show that $\mathrm{mmc}(\ell)$ is upper bounded by $O(1)$ for max-mix of ℓ_2 norm and ℓ_1 norm and sum-mix of ℓ_2 norm and ℓ_1 norm.

Lemma 25.3.1. *For a real number $c > 0$, let $\|x\|_{\ell_a} = \|x\|_2 + c\|x\|_1$ and $\|x\|_{\ell_b} = \max\{\|x\|_2, c\|x\|_1\}$. We have $\mathrm{mmc}(\ell_a) = O(1)$ and $\mathrm{mmc}(\ell_b) = O(1)$.*

Moreover, we show that for an Orlicz norm $\|\cdot\|_G$ induced by a function G which satisfies Assumption 25.1.2, $\mathrm{mmc}(\ell)$ is upper bounded by $O(\sqrt{C_G}\log n)$.

Lemma 25.3.2. *For an Orlicz norm $\|\cdot\|_G$ on $\mathbb{R}^n$ induced by a function G which satisfies Assumption 25.1.2, $\mathrm{mmc}(\ell)$ is upper bounded by $O(\sqrt{C_G}\log n)$.*

Subspace Embedding. The following theorem shows that SymSketch is a subspace embedding.

Theorem 25.3.3. *Let $S \in \mathbb{R}^{O(d)\times n}$ be a SymSketch as defined in Definition 25.3.1. For a given matrix $A \in \mathbb{R}^{n\times d}$, with probability at least 0.9, for all $x \in \mathbb{R}^d$,*

$$\Omega\left(\frac{1}{\sqrt{d}\log^3 n}\right)\cdot\|Ax\|_\ell \leq \|SAx\|_2 \leq O\left(d^2\,\mathrm{mmc}(\ell)\cdot\log^{5/2} n\right)\cdot\|Ax\|_\ell.$$

Furthermore, the running time of computing SA is $\widetilde{O}(\mathrm{nnz}(A) + \mathrm{poly}(d))$.

Combine Theorem 25.3.3 with Lemma 25.3.2, we have the following corollary.

Corollary 25.3.4. *Let $\|\cdot\|_G$ be an Orlicz norm induced by a function G which satisfies Assumption 25.1.2. Let $S \in \mathbb{R}^{O(d)\times n}$ be a* SymSketch *as defined in Definition 25.3.1. For a given matrix $A \in \mathbb{R}^{n\times d}$, with probability at least 0.9, for all $x \in \mathbb{R}^d$,*

$$\Omega\left(\frac{1}{\sqrt{d}\log^3 n}\right)\cdot\|Ax\|_\ell \le \|SAx\|_2 \le O\left(\sqrt{C_G}d^2\log^{7/2} n\right)\cdot\|Ax\|_\ell.$$

Furthermore, the running time of computing SA is $\widetilde{O}(\mathrm{nnz}(A)+\mathrm{poly}(d))$.

25.4 Conclusion

In this paper, we give efficient algorithms for solving the overconstrained linear regression problem, when the loss function is a symmetric norm. For the special case when the loss function is an Orlicz norm, our algorithm produces a $(1+\epsilon)$-approximate solution in $\widetilde{O}(\mathrm{nnz}(A) + \mathrm{poly}(d/\epsilon))$ time. When the loss function is a general symmetric norm, our algorithm produces a $\sqrt{d} \cdot \mathrm{polylog}\, n \cdot \mathrm{mmc}(\ell)$-approximate solution in $\widetilde{O}(\mathrm{nnz}(A) + \mathrm{poly}(d))$ time.

In light of Problem 25.1.1, there are a few interesting problems that remain open. Is that possible to design an algorithm that produces $(1+\epsilon)$-approximate solutions to the linear regression problem, when the loss function is a general symmetric norm? Furthermore, is that possible to use the technique of linear sketching to speed up the overconstrained linear regression problem, when the loss function is a general norm? Answering these problems could lead to a better understanding of Problem 25.1.1.

25.5 Preliminaries

Notations. For a matrix $A \in \mathbb{R}^{n\times d}$, we use A_i to denote its i-th row, A^i to denote its i-th column, $\|A\|_F$ to denote the Frobenius norm of A, and $\|A\|_2$ to denote the spectral norm of A. For any $n' \leq n$, we define $\xi^{(n')} \in \mathbb{R}^n$ to be a vector $\xi^{(n')} = \frac{1}{\sqrt{n'}}(1, 1, \cdots, 1, 0, 0, \cdots, 0)$.

ϵ-nets. We use the standard upper bound on size of ϵ-nets.

Definition 25.5.1. For a given set $\mathcal{S}$ and a norm $\|\cdot\|$, we say $\mathcal{N} \subseteq \mathcal{S}$ is a ϵ-net of $\mathcal{S}$ if for any $s \in \mathcal{S}$, there exists some $\overline{s} \in \mathcal{N}$ such that $\|s - \overline{s}\| \leq \epsilon$.

Lemma 25.5.1 ([Woj96, II.E, 10])**.** *Given a matrix $A \in \mathbb{R}^{n\times d}$ and a norm $\|\cdot\|$, let $\mathcal{S}$ be the unit $\|\cdot\|$-norm ball in the column space of A, i.e., $\mathcal{S} = \{Ax \mid \|Ax\| = 1\}$. For $\epsilon \in (0, 1)$, there exists an ϵ-net $\mathcal{N}$ of $\mathcal{S}$ with size $|\mathcal{N}| \leq (1 + 1/\epsilon)^d$.*

25.6 Missing Proofs in Section 25.2

In this section, we give missing proofs in Section 25.2.

We first show that if a function G satisfies Assumption 25.1.2, then G has at least linear growth. We will use this fact in later proofs.

Lemma 25.6.1. *Given a function G that satisfies property $\mathcal{P}$, then for any $0 < x \leq y$, $y/x \leq G(y)/G(x)$.*

Proof. Due to the convexity of G and $G(0) = 0$, for any $y > x > 0$, we have

$$G(x) \leq G(y)x/y + G(0)(1 - x/y) = G(y)x/y.$$

□

25.6.1 Proof of Lemma 25.2.1

Proof. The first condition is clear from the definition of $\|x\|_{G,w}$.

Now we prove the second condition. When $\|x + y\|_{G,w} = 0$, the triangle inequality clearly holds since $\|x\|_{G,w} \geq 0$ and $\|y\|_{G,w} \geq 0$. When $\|x\|_{G,w} = 0$ and $\|x + y\|_{G,w} \neq 0$, for any $\alpha > 0$, we have

$$\sum_{i=1}^{n} w_i G(|x_i + y_i|/\alpha) = \sum_{i|w_i>0} w_i G(|x_i + y_i|/\alpha) = \sum_{i|w_i>0} w_i G(|y_i|/\alpha) = \sum_{i=1}^{n} w_i G(|y_i|/\alpha),$$

which implies $\|x + y\|_{G,w} = \|y\|_{G,w}$. Similarly, the second condition also holds if $\|y\|_{G,w} = 0$ and $\|x + y\|_{G,w} \neq 0$. If $\|x + y\|_{G,w} \neq 0$, $\|x\|_{G,w} \neq 0$ and $\|y\|_{G,w} \neq 0$, by definition of $\|\cdot\|_{G,w}$, we have

$$\sum_{i=1}^{n} w_i G(x_i/\|x\|_{G,w}) = 1$$

and

$$\sum_{i=1}^{n} w_i G(y_i/\|y\|_{G,w}) = 1.$$

Thus,

$$\begin{aligned}
&\sum_{i=1}^{n} w_i G\left(\frac{x_i + y_i}{\|x\|_{G,w} + \|y\|_{G,w}}\right) \\
\leq &\sum_{i=1}^{n} w_i G\left(\frac{|x_i| + |y_i|}{\|x\|_{G,w} + \|y\|_{G,w}}\right) && (G \text{ is increasing}) \\
\leq &\sum_{i=1}^{n} w_i \left(\frac{\|x\|_{G,w}}{\|x\|_{G,w} + \|y\|_{G,w}} \cdot G(|x_i|/\|x\|_{G,w}) + \frac{\|y\|_{G,w}}{\|x\|_{G,w} + \|y\|_{G,w}} \cdot G(|y_i|/\|y\|_{G,w})\right) \\
& && (G \text{ is convex}) \\
=&1,
\end{aligned}$$

which implies $\|x + y\|_{G,w} \leq \|x\|_{G,w} + \|y\|_{G,w}$.

For the third condition, for any $a \in \mathbb{R}$ and $x \in \mathbb{R}^n$, if $\|x\|_{G,w} = 0$ then $\|ax\|_{G,w} = 0$. If $a = 0$, we have $\|ax\|_{G,w} = 0$. Otherwise, we have

$$\sum_{i=1}^{n} w_i G(x_i/\|x\|_{G,w}) = 1,$$

which implies

$$\sum_{i=1}^{n} w_i G\left(\frac{a \cdot x_i}{|a|\|x\|_{G,w}}\right) = 1,$$

and thus $\|ax\|_{G,w} = |a|\|x\|_{G,w}$. □

25.6.2 Proof of Lemma 25.2.2

Proof. Let $g \in \mathbb{R}^d$ be a vector whose entries are i.i.d. Gaussian random variables with zero mean and standard deviation 10^2. We show that with probability at least 0.8,

$$\sum_{i=1}^{n} G(\|U_i\|_2) \leq O\left(\sum_{i=1}^{n} G(\langle U_i, g\rangle)\right) \leq O\left(\max\{1, C_G\|Ug\|_G^2\}\right) \leq O(C_G d \kappa_G^2).$$

We divide our proofs into three parts.

Part I We will show that with probability at least 0.9,

$$\sum_{i=1}^{n} G(\|U_i\|_2) \leq O\left(\sum_{i=1}^{n} G(\langle U_i, g\rangle)\right).$$

For each $i \in [n]$, $\langle U_i, g\rangle$ has the same distribution as $10^2 \cdot \|U_i\|_2 \cdot \mathcal{N}(0,1)$. For each $i \in [n]$, we let B_i be the random variable such that

$$B_i = \begin{cases} 1 & |\langle U_i, g\rangle| \leq \|U_i\|_2 \\ 0 & \text{otherwise} \end{cases}.$$

By tail inequalities of standard Gaussian random variables, $\Pr[B_i = 1] \leq 0.01$. Thus,

$$\mathbb{E}\left[B_i \cdot G(\|U_i\|_2)\right] \leq 0.01 \cdot G(\|U_i\|_2),$$

which implies

$$\mathbb{E}\left[\sum_{i=1}^{n} B_i \cdot G(\|U_i\|_2)\right] \leq 0.01 \cdot \sum_{i=1}^{n} G(\|U_i\|_2),$$

By the monotonicity of G, since

$$G(\langle U_i, g\rangle) \geq (1 - B_i) G(\|U_i\|_2),$$

we have

$$\sum_{i=1}^{n} G(\langle U_i, g\rangle) \geq \sum_{i=1}^{n} (1 - B_i) G(\|U_i\|_2).$$

By Markov's inequality, with probability at least 0.9, we have

$$\sum_{i=1}^{n} B_i \cdot G(\|U_i\|_2) \leq 0.1 \sum_{i=1}^{n} G(\|U_i\|_2),$$

which implies

$$\sum_{i=1}^{n} G(\langle U_i, g\rangle) \geq 0.9 \sum_{i=1}^{n} G(\|U_i\|_2).$$

Part II We will show that

$$\sum_{i=1}^{n} G(\langle U_i, g\rangle) \leq \max\{1, C_G \cdot \|Ug\|_G^2\}.$$

When $\|Ug\|_G \leq 1$, by monotonicity of G, we must have

$$\sum_{i=1}^{n} G(\langle U_i, g\rangle) \leq 1.$$

When $\|Ug\|_G \geq 1$, we have

$$\sum_{i=1}^{n} G(\langle U_i, g\rangle / \|Ug\|_G) = 1.$$

Since

$$G(\langle U_i, g\rangle) \leq G(\langle U_i, g\rangle / \|Ug\|_G) \cdot C_G \|Ug\|_G^2$$

and

$$\sum_{i=1}^{n} G(\langle U_i, g\rangle / \|Ug\|_G) = 1,$$

we must have

$$\sum_{i=1}^{n} G(\langle U_i, g\rangle) \leq \sum_{i=1}^{n} G(\langle U_i, g\rangle / \|Ug\|_G) \cdot C_G \cdot \|Ug\|_G^2 \leq C_G \cdot \|Ug\|_G^2.$$

Part III We will show that $\|Ug\|_G^2 \leq O(C_G d \kappa_G^2)$. By definition of a well-conditioned basis and tail inequalities of Gaussian random variables, with probability at least 0.9, we have

$$\|Ug\|_G \leq \kappa_G \|g\|_2 \leq O(\kappa_G \sqrt{d}).$$

Thus, applying a union bound over three parts of the proof, we have with probability at least 0.8,

$$\sum_{i=1}^{n} G(\|U_i\|_2) \leq O(C_G d \kappa_G^2). \tag{25.9}$$

However, the condition in (25.9) is deterministic. Thus, the condition in (25.9) always holds. □

25.6.3 Proof of Lemma 25.2.3

Proof. Notice that for any $x \in \mathbb{R}^d$,

$$\|AR^{-1}x\|_G \leq \|\Pi AR^{-1}x\|_2 = \kappa\|Qx\|_2 = \kappa\|x\|_2$$

and

$$\|AR^{-1}x\|_G \geq \frac{1}{\kappa}\|\Pi AR^{-1}x\|_2 = \|Qx\|_2 = \|x\|_2.$$

□

25.6.4 Proof of Lemma 25.2.4

Proof. In Theorem 2.13 of [Woo14b], it has been shown how to calculate $\{l_i\}_{i=1}^n$ such that $l_i = \Theta(\|(AR^{-1})_i\|_2)$ in $\widetilde{O}(\text{nnz}(A) + \text{poly}(d))$ time with probability at least 0.99. We simply take $u_i = G(l_i)$. By Lemma 25.6.1 and the growth condition of G, we must have $u_i = \Theta(G(\|(AR^{-1})_i\|_2))$. □

25.6.5 Proof of Lemma 25.2.5

Proof. By homogeneity, we only need to prove that with probability $1 - \delta$, for all x which satisfies $\|Ux\|_G = 1$,

$$(1-\epsilon)\|Ux\|_G \leq \|Ux\|_{G,w} \leq (1+\epsilon)\|Ux\|_G.$$

We first prove that for any fixed $x \in \mathbb{R}^d$ such that $\|Ux\|_G = 1$, with probability $1 - \delta(1 + 4/\epsilon)^{-d}$,

$$(1-\epsilon/4)\|Ux\|_G \leq \|Ux\|_{G,w} \leq (1+\epsilon/4)\|Ux\|_G.$$

Let $x \in \mathbb{R}^d$ that satisfies $\|Ux\|_G = 1$ and $y = Ux$. Let Z_i be a random variable which denotes the value of $w_i G(y_i)$ and $Z = \sum_{i=1}^n Z_i$.

We will first show that if $Z \in [1-\epsilon/4, 1+\epsilon/4]$, then $\|y\|_{G,w} \in [1-\epsilon/4, 1+\epsilon/4]$. There are three cases:

1. If $\|y\|_{G,w} = 1$, then $\|y\|_{G,w}$ is already in $[1-\epsilon/4, 1+\epsilon/4]$.

2. If $\|y\|_{G,w} > 1$, then by Lemma 25.6.1, we have

$$\sum_{i=1}^n w_i G(y_i) \geq \sum_{i=1}^n w_i \|y\|_{G,w} \cdot G(y_i/\|y\|_{G,w}).$$

   Since

$$\sum_{i=1}^n w_i \cdot G(y_i/\|y\|_{G,w}) = 1,$$

   we must have

$$\|y\|_{G,w} \leq \sum_{i=1}^n w_i G(y_i) = Z \leq 1 + \epsilon/4.$$

3. If $\|y\|_{G,w} < 1$, then by Lemma 25.6.1, we have

$$1 = \sum_{i=1}^n w_i G(y_i/\|y\|_{G,w}) \geq 1/\|y\|_{G,w} \cdot \sum_{i=1}^n w_i G(y_i),$$

   which implies

$$\|y\|_{G,w} \geq \sum_{i=1}^n w_i G(y_i) = Z \geq 1 - \epsilon/4.$$

Thus, it suffices to prove that

$$\Pr\left[Z \in [1-\epsilon/4, 1+\epsilon/4]\right] \geq 1 - \delta(1+4/\epsilon)^{-d}.$$

Consider the expectation of Z, we have

$$\begin{aligned}\mathbb{E}[Z] &= \sum_{i=1}^{n} \mathbb{E}[Z_i] \\ &= \sum_{i=1}^{n} \mathbb{E}[w_i] \cdot G((Ux)_i) \\ &= \sum_{i=1}^{n} G((Ux)_i) \\ &= 1,\end{aligned}$$

where the last equality follows since $\|Ux\|_G = 1$.

Notice that $|Z_i - \mathbb{E}(Z_i)|$ is always upper bounded by

$$\begin{aligned}w_i G(y_i) &= w_i G((Ux)_i) \\ &\leq w_i G(\|U_i\|_2 \cdot \|x\|_2) \\ &\leq w_i G\left(\|U_i\|_2\right) \\ &\leq G\left(\|U_i\|_2\right)/p_i \\ &\leq \frac{\epsilon^2}{C\left(\log(1/\delta) + d\log(1/\epsilon)\right)},\end{aligned}$$

where the first inequality follows from Cauchy-Schwarz inequality, the second inequality follows from the definition of well-conditioned basis in Definition 25.1.2 and monotonicity of G, the third inequality follows from definition of w_i and the last inequality follows from the choice of p_i.

Consider the variance of Z, we have:

$$\begin{aligned}
\mathbb{V}(Z) &= \sum_{i|p_i<1} \mathbb{V}(Z_i) \\
&\leq \sum_{i|p_i<1} \mathbb{E}(Z_i^2) \\
&= \sum_{i|p_i<1} \left(G((Ux)_i)\right)^2 / p_i \\
&\leq \left(\sum_{i|p_i<1} G((Ux)_i)\right) \cdot \max_{i|p_i<1} G((Ux)_i)/p_i \\
&\leq \frac{\epsilon^2}{C\left(\log(1/\delta) + d\log(1/\epsilon)\right)},
\end{aligned}$$

where the second inequality follows from Hölder's inequality and the last inequality follows from the upper bound of $G((Ux)_i)/p_i$ and $\|Ux\|_G = 1$.

Thus, by Bernstein inequality, we have:

$$\Pr\left(|Z - 1| > \epsilon/4\right) \leq (1 + 4/\epsilon)^{-d}\delta.$$

Thus, for a fixed x, with probability at least $1 - (1 + 4/\epsilon)^{-d}\delta$, we have

$$(1 - \epsilon/4)\|Ux\|_G \leq \|Ux\|_{G,w} \leq (1 + \epsilon/4)\|Ux\|_G.$$

Let $\mathcal{S}$ be the unit $\|\cdot\|_G$-norm ball in the column space of U, i.e., $\mathcal{S} = \{Ux \mid \|Ux\|_G = 1\}$. According to Lemma 25.5.1, there exists an $\epsilon/4$-net $\mathcal{N}$ of $\mathcal{S}$ with $|\mathcal{N}| \leq (1 + 4/\epsilon)^d$. We use $\mathcal{E}$ to denote the event that for all $y \in \mathcal{N}$, $\|y\|_{G,w} \in [1 - \epsilon/4, 1 + \epsilon/4]$. By taking union bound over all vectors in $\mathcal{N}$, we have $\Pr[\mathcal{E}] \geq 1 - \delta$.

Conditioned on $\mathcal{E}$, now we show that for all $y \in \mathcal{S}$, $\|y\|_{G,w} \in [1 - \epsilon, 1 + \epsilon]$. Consider a

fixed vector $y \in \mathcal{S}$, since $\mathcal{N}$ is an $\epsilon/4$-net of $\mathcal{S}$, we can choose a vector $u^{(1)} \in \mathcal{N}$ such that

$$\|y - u^{(1)}\|_G \leq \epsilon/4.$$

Thus, we have that

$$\begin{aligned}\|y\|_{G,w} &\leq \|u^{(1)}\|_{G,w} + \|y - u^{(1)}\|_{G,w} \\ &\leq (1 + \epsilon/4) + \|y - u^{(1)}\|_{G,w}.\end{aligned}$$

Let $\alpha^{(1)} = 1/\|y - u^{(1)}\|_G$. Then we have $\alpha^{(1)}(y - u^{(1)}) \in \mathcal{S}$. Thus, there exist $u^{(2)} \in \mathcal{N}$ such that

$$\|u^{(2)} - \alpha^{(1)}(y - u^{(1)})\|_G \leq \epsilon/4.$$

It implies that

$$\begin{aligned}\|(y - u^{(1)}) - u^{(2)}/\alpha^{(1)}\|_G &\leq \epsilon/(4\alpha^{(1)}) \\ &\leq (\epsilon/4)^2.\end{aligned}$$

Thus,

$$\begin{aligned}\|y - u^{(1)}\|_{G,w} &\leq \|u^{(2)}\|_{G,w}/\alpha^{(1)} + \|y - u^{(1)} - u^{(2)}/\alpha^{(1)}\|_{G,w} \\ &\leq (1 + \epsilon/4)\epsilon/4 + \|y - u^{(1)} - u^{(2)}/\alpha^{(1)}\|_{G,w}.\end{aligned}$$

Let $\alpha^{(2)} = 1/\|y - u^{(1)} - u^{(2)}/\alpha^{(1)}\|_G$. Then we can repeat the above argument and get

$$\begin{aligned}\|y\|_{G,w} &\leq (1 + \epsilon/4) + (1 + \epsilon/4)\epsilon/4 + (1 + \epsilon/4)(\epsilon/4)^2 + \cdots \\ &= (1 + \epsilon/4)/(1 - \epsilon/4) \\ &\leq 1 + \epsilon.\end{aligned}$$

By applying the above upper bound on $\|\alpha^{(1)}(u^{(1)} - y)\|_{G,w}$, we can get

$$\begin{aligned}
\|y\|_{G,w} &\geq \|u^{(1)}\|_{G,w} - \|u^{(1)} - y\|_{G,w} \\
&\geq (1 - \epsilon/4) - \|u^{(1)} - y\|_{G,w} \\
&\geq (1 - \epsilon/4) - \frac{1+\epsilon}{\alpha^{(1)}} \\
&\geq 1 - \epsilon/2 - \epsilon^2/4 \\
&\geq 1 - \epsilon.
\end{aligned}$$

Thus, conditioned on $\mathcal{E}$, which holds with probability $1 - \delta$, we have $\|y\|_{G,w} \in [1 - \epsilon, 1 + \epsilon]$ for all $y = Ux$ with $\|y\|_G = 1$. □

25.6.6 Proof of Theorem 25.2.6

Proof. We first analyze the running time of the algorithm. In Step 1, we calculate $\Pi\mathbf{Ab}$ and invoke QR-decomposition on $\Pi\mathbf{Ab}$. In Step 2, we apply the algorithm in Lemma 25.2.4, which runs in $\widetilde{O}(\mathrm{nnz}(A) + \mathrm{poly}(d))$ time. Obtaining the weight vector $w \in \mathbb{R}^n$ in Step 3 requires $O(n)$ time.

Since for all $x \in \mathbb{R}^d$,

$$\|\mathbf{Ab}x\|_G \leq \|\Pi\mathbf{Ab}x\|_2 \leq \kappa\|\mathbf{Ab}x\|_G.$$

we have

$$\begin{aligned}
\mathbb{E}[\|w\|_0] = \sum_{i=1}^{n} p_i &= \sum_{i=1}^{n} O\left(d\log(1/\epsilon)/\epsilon^2 \cdot G\left(\|(\mathbf{Ab}R^{-1})_i\|_2\right)\right) \\
&\leq O\left(d\log(1/\epsilon)/\epsilon^2 \cdot C_G d\left(\kappa_G(\mathbf{Ab}R^{-1})\right)^2\right) && \text{(Lemma 25.2.2)} \\
&\leq O\left(C_G d^2\kappa^2\log(1/\epsilon)/\epsilon^2\right). && \text{(Lemma 25.2.3)}
\end{aligned}$$

By Markov's inequality, with constant probability we have $\|w\|_0 \leq O\left(C_G d^2 \kappa^2 \log(1/\epsilon)/\epsilon^2\right)$. Moreover, in order to solve $\min_x \|Ax - b\|_{G,w}$, we can ignore all rows of A with zero weights, and thus there are at most $O\left(C_G d^2 \kappa^2 \log(1/\epsilon)/\epsilon^2\right)$ remaining rows in A. Furthermore, as we show in Lemma 25.2.1, $\|\cdot\|_{G,w}$ is a seminorm, which implies we can solve $\min_x \|Ax - b\|_{G,w}$ in $\mathrm{poly}(C_G d\kappa/\epsilon)$ time, by simply solve a convex program with size $O\left(C_G d^2 \kappa^2 \log(1/\epsilon)/\epsilon^2\right)$.

Now we prove the correctness of the algorithm. The algorithm in Lemma 25.2.4 succeeds with constant probability. By Lemma 25.2.5, with constant probability, simultaneously for all $x \in \mathbb{R}^{d+1}$,

$$(1 - \epsilon/3)\|\mathbf{Ab}R^{-1}x\|_G \leq \|\mathbf{Ab}R^{-1}x\|_{G,w} \leq (1 + \epsilon/3)\|\mathbf{Ab}R^{-1}x\|_G.$$

Equivalently, with constant probability, simultaneously for all $x \in \mathbb{R}^{d+1}$,

$$(1 - \epsilon/3)\|\mathbf{Ab}x\|_G \leq \|\mathbf{Ab}x\|_{G,w} \leq (1 + \epsilon/3)\|\mathbf{Ab}x\|_G.$$

Since $x^* = \mathrm{argmin}_x \|Ax - b\|_{G,w}$, for all $x \in \mathbb{R}^d$, we have

$$\begin{aligned}
\|Ax^* - b\|_G &\leq 1/(1 - \epsilon/3)\|Ax^* - b\|_{G,w} \\
&\leq 1/(1 - \epsilon/3)\|Ax - b\|_{G,w} \\
&\leq (1 + \epsilon/3)/(1 - \epsilon/3)\|Ax - b\|_G \\
&\leq (1 + \epsilon)\|Ax - b\|_G
\end{aligned}$$

for sufficiently small ϵ. Thus, x^* is a $(1 + \epsilon)$-approximate solution to $\min_x \|Ax - b\|_G$.

Note that the failure probability of the algorithm can be reduced to an arbitrarily small constant by independent repetitions and taking the best solution found among all repetitions. □

25.7 Missing Proofs in Section 25.3

In this section, we give missing proofs in Section 25.3.

Without loss of generality, throughout this section, for the symmetric norm $\|\cdot\|_\ell$ under consideration, we assume $\|\xi^{(1)}\|_\ell = 1$.

25.7.1 Background

25.7.1.1 Known ℓ_2 Oblivious Subspace Embeddings

In this section, we recall some known ℓ_2 subspace embeddings.

Definition 25.7.1. We say $S \in \mathbb{R}^{t\times n}$ is an ℓ_2 subspace embedding for the column space of $A \in \mathbb{R}^{n\times d}$, if for all $x \in \mathbb{R}^d$,

$$(1-\epsilon)\|Ax\|_2 \le \|SAx\|_2 \le (1+\epsilon)\|Ax\|_2.$$

Definition 25.7.2. A CountSketch embedding is defined to be $\Pi = \Phi D \in \mathbb{R}^{m\times n}$ with $m = \Theta(d^2/\epsilon^2)$, where D is an $n \times n$ random diagonal matrix with each diagonal entry independently chosen to be $+1$ or -1 with equal probability, and $\Phi \in \{0,1\}^{m\times n}$ is an $m \times n$ binary matrix with $\Phi_{h(i),i} = 1$ and all remaining entries being 0, where $h : [n] \to [m]$ is a random map such that for each $i \in [n]$, $h(i) = j$ with probability $1/m$ for each $j \in [m]$.

Theorem 25.7.1 ([CW13])**.** *For a given matrix $A \in \mathbb{R}^{n\times d}$ and $\epsilon \in (0, 1/2)$. Let $\Pi \in \mathbb{R}^{\Theta(d^2/\epsilon^2)\times n}$ be a CountSketch embedding. With probability at least 0.9999, Π is an ℓ_2 subspace embedding for the column space of A. Furthermore, ΠA can be computed in $O(\mathrm{nnz}(A))$ time.*

Definition 25.7.3. A Gaussian embedding S is defined to be $\frac{1}{\sqrt{m}} \cdot G \in \mathbb{R}^{m\times n}$ with $m = \Theta(d/\epsilon^2)$, where each entry of $G \in \mathbb{R}^{m\times n}$ is chosen independently from the standard Gaussian distribution.

Theorem 25.7.2 ([Woo14b]). *For a given matrix $A \in \mathbb{R}^{n\times d}$ and $\epsilon \in (0, 1/2)$. Let $S \in \mathbb{R}^{\Theta(d/\epsilon^2)\times n}$ be a Gaussian embedding. With probability at least 0.9999, S is an ℓ_2 subspace embedding for the column space of A.*

Definition 25.7.4. A composition of Gaussian embedding and CountSketch embedding is defined to be $S' = S\Pi$, where $\Pi \in \mathbb{R}^{\Theta(d^2/\epsilon^2)\times n}$ is a CountSketch embedding and $S \in \mathbb{R}^{\Theta(d/\epsilon^2)\times\Theta(d^2/\epsilon^2)}$ is a Gaussian embedding.

The following corollary directly follows from the above two theorems.

Corollary 25.7.3. *For a given matrix $A \in \mathbb{R}^{n\times d}$ and $\epsilon \in (0, 1/2)$. Let $S' \in \mathbb{R}^{\Theta(d/\epsilon^2)\times n}$ be a composition of Gaussian embedding and CountSketch embedding. With probability at least 0.9998, S' is an ℓ_2 subspace embedding for the column space of A. Furthermore, $S'A$ can be computed in $O(\mathrm{nnz}(A) + d^4/\epsilon^4)$ time.*

We remark that all ℓ_2 subspace embeddings introduced in this section are *oblivious*, meaning that the distribution of the embedding matrix does not depend on the matrix A.

25.7.1.2 Properties of Symmetric Norms

General Properties. We first introduce several general properties of symmetric norms.

Lemma 25.7.4 (Lemma 2.1 in [BBC+17]). *For any symmetric norm $\|\cdot\|_\ell$ and $x, y \in \mathbb{R}^n$ such that for all $i \in [n]$ we have $|x_i| \leq |y_i|$, then $\|x\|_\ell \leq \|y\|_\ell$.*

Lemma 25.7.5 (Fact 2.2 in [BBC+17]). *Suppose $\|\xi^{(1)}\|_\ell = 1$, for any vector $x \in \mathbb{R}^n$,*

$$\|x\|_\infty \leq \|x\|_\ell \leq \|x\|_1.$$

Lemma 25.7.6 (Lemma 3.12 in [BBC+17]). *Let $\|\cdot\|_\ell$ be a symmetric norm. Then*

$$\Omega(M_\ell/\sqrt{\log n}) \le \|\xi^{(n)}\|_\ell \le O(M_\ell),$$

where M_ℓ is as defined in Definition 25.3.2.

Modulus of Approximation. We need the following quantity of a symmetric norm.

Definition 25.7.5. The *maximum modulus of approximation* of a symmetric norm $\|\cdot\|_\ell$ is defined as

$$\mathrm{mma}(\ell, r) = \max_{1\le a\le b\le ar\le n} \frac{M_{\ell(ar)}}{M_{\ell(b)}},$$

where $\|\cdot\|_{\ell(k)}$ is a norm on $\mathbb{R}^k$ which is defined to be

$$\|(x_1, x_2, \cdots, x_k)\|_{\ell(k)} = \|(x_1, x_2, \cdots, x_k, 0, \cdots, 0)\|_\ell,$$

and $M_{\ell(k)}$ is as defined in Definition 25.3.2.

Intuitively, $\mathrm{mma}(\ell, r)$ characterizes how well the original symmetric norm can be approximated by a lower dimensional induced norm. We show in the following lemma that $\mathrm{mma}(\ell, r) \le O(\sqrt{r \log n})$ for any symmetric norm.

Lemma 25.7.7. *For any symmetric norm $\|\cdot\|_\ell$ and $r \in [n]$, $\mathrm{mma}(\ell, r) \le O(\sqrt{r \log n})$.*

Proof. By Lemma 25.7.6, for any $i \in [n]$, $\Omega(M_{\ell(i)}/\sqrt{\log n}) \le \|\xi^{(i)}\|_\ell \le O(M_{\ell(i)})$. Let $ar = c_1 b + c_2$, where c_1, c_2 are non-negative integers with $c_1 \le ar/b$ and $c_2 \le b$. Observe that we can rewrite $\xi^{(ar)}$ as

$$\xi^{(ar)} = \left[\frac{\sqrt{b}}{\sqrt{ar}} \cdot \left(\underbrace{\xi^{(b)}, \xi^{(b)}, \xi^{(b)}, \ldots, \xi^{(b)}}_{c_1 \text{ times}}\right), \frac{\sqrt{c_2}}{\sqrt{ar}}\xi^{(c_2)}, 0, \ldots, 0\right].$$

Therefore, by triangle inequality, we have

$$\begin{aligned}
\|\xi^{(ar)}\|_\ell &\le \frac{\sqrt{b}}{\sqrt{ar}} \cdot c_1 \cdot \|\xi^{(b)}\|_\ell + \frac{\sqrt{c_2}}{\sqrt{ar}} \cdot \|\xi^{(c_2)}\|_\ell \\
&\le \sqrt{\frac{b}{ar}} \cdot \frac{ar}{b} \cdot \|\xi^{(b)}\|_\ell + \frac{\sqrt{b}}{\sqrt{ar}} \cdot \|\xi^{(b)}\|_\ell && (c_1 \le ar/b \text{ and } c_2 \le b) \\
&\le \sqrt{r} \cdot \|\xi^{(b)}\|_\ell + \|\xi^{(b)}\|_\ell && (a \le b \le ar) \\
&\le 2\sqrt{r} \cdot \|\xi^{(b)}\|_\ell. && (r \ge 1)
\end{aligned}$$

Now we apply Lemma 25.7.6 on both sides, which implies

$$\frac{M_{\ell^{(ar)}}}{\sqrt{\log n}} \le O(\sqrt{r} \cdot M_{\ell^{(b)}})$$

as desired. □

Properties of SymSketch. Now we introduce several properties of SymSketch.

The following lemma shows that for a data matrix $A \in \mathbb{R}^d$, calculating SA requires $\widetilde{O}(\mathrm{nnz}(A) + \mathrm{poly}(d))$ time for a SymSketch S.

Lemma 25.7.8. *For a given matrix $A \in \mathbb{R}^{n\times d}$, let $S \in \mathbb{R}^{O(d)\times n}$ be a SymSketch as in Definition 25.3.1. SA can be computed in $O(\mathrm{nnz}(A)) + \mathrm{poly}(d)$ time in expectation, and in $O(\mathrm{nnz}(A)\log n) + \mathrm{poly}(d)$ time in the worst case.*

Proof. Since S is a SymSketch, $S = \Pi\widetilde{D} = \Pi \cdot \begin{bmatrix} w_0 D_0 \\ w_1 D_1 \\ \vdots \\ w_t D_t \end{bmatrix}$, where $\Pi \in \mathbb{R}^{O(d)\times O(n\log n)}$. Since D_i is a diagonal matrix, $\mathrm{nnz}(D_i A) \le \mathrm{nnz}(A)$, and thus $\mathrm{nnz}(\widetilde{D}A) \le (t+1)\cdot\mathrm{nnz}(A) = O(\mathrm{nnz}(A)\log n)$, which implies $\widetilde{D}A$ can be computed in $(t+1)\cdot\mathrm{nnz}(A) = O(\mathrm{nnz}(A)\log n)$ time.

On the other hand, the expected number of nonzero entries of $D_i A$ is $2^{-i}\,\mathrm{nnz}(A)$. Thus, $\widetilde{D}A$ has $O(\mathrm{nnz}(A))$ nonzero entries in expectation, which implies $\widetilde{D}A$ can be computed in $O(\mathrm{nnz}(A))$ time.

Finally, notice that Π is a composition of Gaussian embedding and CountSketch embedding, which implies $\Pi\widetilde{D}A$ can be computed in $\mathrm{nnz}(\widetilde{D}A) + \mathrm{poly}(d)$ time. □

The following lemma shows that with constant probability, for all $x \in \mathbb{R}^n, \|Sx\|_2 \leq \mathrm{poly}(n)\|x\|_2$.

Lemma 25.7.9. *Let $S \in \mathbb{R}^{O(d)\times n}$ be a SymSketch as defined in Definition 25.3.1, then with probability at least 0.9999 $\|S\|_2 \leq \mathrm{poly}(n)$.*

Proof. Notice that $S = \Pi\widetilde{D}$, since $\|S\|_2 \leq \|\Pi\|_2 \cdot \|\widetilde{D}\|_2$, it suffices to bound $\|\Pi\|_2$ and $\|\widetilde{D}\|_2$. Since Π is a composition of Gaussian embedding and CountSketch embedding (Definition 25.7.4), with probability at least $0.9999, \|\Pi\|_2 \leq \|\Pi\|_F \leq \mathrm{poly}(n)$. Now consider $\widetilde{D} = \begin{bmatrix} w_0 D_0 \\ w_1 D_1 \\ \vdots \\ w_t D_t \end{bmatrix}$. By Lemma 25.7.5, for all $j \in [t], w_j \leq \mathrm{poly}(n)$. Furthermore, $\|D_j\|_2 \leq 1$ and $t = \Theta(\log n)$, which implies $\|\widetilde{D}\|_2 \leq \mathrm{poly}(n)$. □

Throughout this whole section we assume that for any non-zero vector $x \in \mathbb{R}^n$, we have $1 \leq |x_j| \leq \mathrm{poly}(n)$ for all $j \in [n]$. Notice that this assumption is without loss of generality, as shown in the following lemma.

Lemma 25.7.10. *For any non-zero vector $x \in \mathbb{R}^n$, let $\overline{x} \in \mathbb{R}^n$ be a vector with $\overline{x} = \frac{\text{poly}(n) \cdot x}{\|x\|_\infty}$, and $x' \in \mathbb{R}^n$ where*

$$x'_i = \begin{cases} \overline{x}_i & \text{if } \overline{x}_i \geq 1 \\ 0 & \text{otherwise} \end{cases}.$$

For a symmetric norm $\|\cdot\|_\ell$, suppose $\|S\|_2 \leq \text{poly}(n)$ and

$$\alpha \|x'\|_\ell \leq \|Sx'\|_2 \leq \beta \|x'\|_\ell$$

for some $\alpha, \beta \in [1/\text{poly}(n), \text{poly}(n)]$, then

$$\Omega(\alpha)\|x\|_\ell \leq \|Sx\|_2 \leq O(\beta)\|x\|_\ell.$$

Proof. By triangle inequality and Lemma 25.7.4, we have

$$\|\overline{x}\|_\ell - n \leq \|x'\|_\ell \leq \|\overline{x}\|_\ell.$$

By Lemma 25.7.5,

$$\|x'\|_\ell \geq \|x'\|_\infty = \|\overline{x}\|_\infty = \text{poly}(n),$$

we have

$$(1 - 1/\text{poly}(n))\|\overline{x}\|_\ell \leq \|x'\|_\ell \leq \|\overline{x}\|_\ell.$$

Notice that $\|S\overline{x}\|_2 = \|Sx' + S(\overline{x} - x')\|_2$. By triangle inequality we have

$$\|Sx'\|_2 - \|S\|_2\|\overline{x} - x'\|_2 \leq \|S\overline{x}\|_2 \leq \|Sx'\|_2 + \|S\|_2\|\overline{x} - x'\|_2.$$

By the given conditions, we have

$$(1 - 1/\text{poly}(n))\|Sx'\|_2 \leq \|S\overline{x}\|_2 \leq (1 + 1/\text{poly}(n))\|Sx'\|_2,$$

which implies

$$\Omega(\alpha)\|\overline{x}\|_\ell \leq \|S\overline{x}\|_2 \leq O(\beta)\|\overline{x}\|_\ell.$$

Since $\overline{x} = \frac{\text{poly}(n)\cdot x}{\|x\|_\infty}$, we have

$$\Omega(\alpha)\|x\|_\ell \leq \|Sx\|_2 \leq O(\beta)\|x\|_\ell.$$

□

By Lemma 25.7.10, we can focus on those non-zero vectors $x \in \mathbb{R}^n$ such that $1 \leq |x_j| \leq \text{poly}(n)$ for all $j \in [n]$.

Definition 25.7.6. For a given vector $x \in \mathbb{R}^n$, suppose for all $j \in [n]$,

$$1 \leq |x_j| \leq \text{poly}(n).$$

Let $g = \Theta(\log n)$. For each $i \in \{0, 1, \ldots, g\}$, we define

$$L_i(x) = \{j \mid 2^i \leq |x_j| < 2^{i+1}\}.$$

For each $i \in \{0, 1, \ldots, g\}$, we define $V_i(x) \in \mathbb{R}^n$ to be the vector

$$V_i(x) = (\underbrace{2^i, 2^i, \ldots, 2^i}_{|L_i(x)|}, 0, \ldots, 0).$$

For each $i \in \{0, 1, \ldots, g\}$, we say a level i to be *contributing* if

$$\|V_i(x)\|_\ell \geq \Omega\left(1/g\right) \cdot \|x\|_\ell.$$

Lemma 25.7.11. *Let $g = \Theta(\log n)$. For a given vector $x \in \mathbb{R}^n$ such that for all $j \in [n], 1 \leq |x_j| \leq 2^g$, there exists at least one level $i \in \{0, 1, \cdots, g\}$ which is contributing.*

Proof. If none of $i \in \{0, 1, \cdots, g\}$ is contributing, then $\|x\|_\ell \leq \sum_{i=0}^{g} \|V_i(x)\|_\ell \leq 1/(2g) \cdot \sum_{i=0}^{g} \|x\|_\ell \leq \frac{1}{2}\|x\|_\ell$, which leads to a contradiction. □

25.7.2 Proof of Lemma 25.3.1

Proof. Consider a fixed $n' \in [n]$. By Lemma 25.7.6, we have

$$M_{\ell_a^{(n')}} = \Omega\left(\|\xi^{(n')}\|_{\ell_a^{(n')}}\right) = \Omega\left(1 + c\sqrt{n'}\right)$$

and

$$M_{\ell_b^{(n')}} = \Omega\left(\|\xi^{(n')}\|_{\ell_b^{(n')}}\right) = \Omega\left(\max\left(1, c\sqrt{n'}\right)\right).$$

It is also straightforward to verify that

$$\max_{x \in \mathbb{S}^{n'-1}} \|x\|_{\ell_a} = 1 + c\sqrt{n'}$$

and

$$\max_{x \in \mathbb{S}^{n'-1}} \|x\|_{\ell_b} = \max\left(1, c\sqrt{n'}\right).$$

Taking the ratio between $\max_{x \in \mathbb{S}^{n'-1}} \|x\|_\ell$ and $M_{\ell^{(n')}}$ for $\ell \in \{\ell_a, \ell_b\}$, we complete the proof. □

25.7.3 Proof of Lemma 25.3.2

Proof. Let $\overline{G}(x) = \sqrt{x} \cdot G^{-1}(1/x)$, where $G^{-1}(1/x)$ is the unique value in $[0, \infty)$ such that $G(G^{-1}(1/x)) = 1/x$. We first show that $\overline{G}(x)$ is an approximately decreasing function for $x \in (0, \infty)$. Let m, n be two real numbers with $0 < m \leq n$. We have $1/n \leq 1/m$, which implies $0 < G^{-1}(1/n) \leq G^{-1}(1/m)$ by monotonicity of G. By the third condition in Assumption 25.1.2, we have

$$\frac{G(G^{-1}(1/m))}{G(G^{-1}(1/n))} \leq C_G \cdot \left(\frac{G^{-1}(1/m)}{G^{-1}(1/n)}\right)^2,$$

which implies

$$\sqrt{n} \cdot G^{-1}(1/n) \leq \sqrt{C_G} \cdot \sqrt{m} \cdot G^{-1}(1/m).$$

Hence $\overline{G}(n) \le \sqrt{C_G} \cdot \overline{G}(m)$.

We are now ready to prove the lemma. Recall that for the Orlicz norm $\|\cdot\|_\ell = \|\cdot\|_G$, we have

$$\mathrm{mmc}(\ell) = \max_{n' \in [n]} \mathrm{mc}(\ell^{(n')}) = \max_{n' \in [n]} \frac{\max_{x \in \mathbb{S}^{n'-1}} \|x\|_{\ell^{(n')}}}{M_{\ell^{(n')}}}.$$

By Lemma 25.7.6, we have,

$$\Omega(1) \cdot \|\xi^{(n')}\|_{\ell^{(n')}} \le M_{\ell^{(n')}} \le O(\sqrt{\log n}) \cdot \|\xi^{(n')}\|_{\ell^{(n')}}.$$

Thus $\|\xi^{(n')}\|_{\ell^{(n')}}$ provides an approximation to $M_{\ell^{(n')}}$. By definition of $\|\cdot\|_G$, we have

$$\|\xi^{(n')}\|_{\ell^{(n')}} = \frac{1}{\sqrt{n'} \cdot G^{-1}(1/n')} = \frac{1}{\overline{G}(n')}.$$

Hence

$$\Omega(1) \le M_{\ell^{(n')}} \cdot \overline{G}(n') \le O(\sqrt{\log n}).$$

Next, we compute $\max_{x \in \mathbb{S}^{n'-1}} \|x\|_{\ell^{(n')}}$. For an arbitrary unit vector $x \in \mathbb{S}^{n'-1}$, we denote

$$B_j = \{i \in [n] : |x_i| \in [1/2^j, 1/2^{j-1})\}$$

and $b_j = |B_j|$. For each j, let $x^{B_j} \in \mathbb{R}^n$ be the vector such that

$$x_i^{B_j} = \begin{cases} x_i & \text{if } j \in B_j \\ 0 & \text{otherwise} \end{cases}.$$

Note that non-zero coordinates in x^{B_j} have magnitude close to each other (within a factor

of 2), we thus have

$$\begin{aligned}\|x^{B_j}\|_\ell &= \|x^{B_j}\|_2 \cdot \left\| \frac{x^{B_j}}{\|x^{B_j}\|_2} \right\|_\ell \\ &\le \frac{\sqrt{b_j}}{2^{j-1}} \cdot \left\| \frac{x^{B_j}}{\|x^{B_j}\|_2} \right\|_\ell \\ &\le \frac{\sqrt{b_j}}{2^{j-2}} \cdot \|\xi^{(b_j)}\|_{\ell^{(b_j)}} \\ &= \frac{\sqrt{b_j}}{2^{j-2}} \cdot \frac{1}{\overline{G}(b_j)}.\end{aligned}$$

Similarly,

$$\begin{aligned}\|x^{B_j}\|_\ell &\ge \frac{\sqrt{b_j}}{2^{j+2}} \cdot \|\xi^{(b_j)}\|_{\ell^{(b_j)}} \\ &= \frac{\sqrt{b_j}}{2^{j+2}} \cdot \frac{1}{\overline{G}(b_j)} \\ &\ge \frac{\sqrt{b_j}}{2^{j+2}} \cdot \frac{1}{\sqrt{C_G} \cdot \overline{G}(1)}.\end{aligned}$$

We claim there exists an constant $c > 0$ such that

$$\sum_{\substack{j > c \log n \\ b_j > 0}} \|x^{B_j}\|_\ell \le \sum_{j' \le c \log n} \|x^{B^{j'}}\|_\ell.$$

To show this, by Lemma 25.6.1, for any $b \ge 1$,

$$b = \frac{G(G^{-1}(1))}{G(G^{-1}(1/b))} \ge \frac{G^{-1}(1)}{G^{-1}(1/b)},$$

which implies

$$G^{-1}(1/b) \ge \frac{G^{-1}(1)}{b}.$$

Next, since $\|x\|_2 = 1$, there exists an $0 \le \widetilde{j} \le 4 \log n$ such that $b_{\widetilde{j}} \ge 1$. Therefore,

$$\|x^{B_{\widetilde{j}}}\|_\ell \ge \frac{1}{2^{\widetilde{j}+2}} \cdot \frac{1}{\sqrt{C_G} \cdot \overline{G}(1)}.$$

Thus, we have

$$\begin{aligned}
\sum_{\substack{j>c\log n\\ b_j>0}} \|x^{B_j}\|_\ell &= \sum_{\substack{j>c\log n\\ b_j>0}} \|x^{B_j}\|_2 \cdot \left\| \frac{x^{B_j}}{\|x^{B_j}\|_2} \right\|_\ell \\
&\le \sum_{\substack{j>c\log n\\ b_j>0}} \frac{\sqrt{n}}{2^{j-2}} \cdot \|\xi^{(b_j)}\|_{\ell^{(b_j)}} \\
&\le \sum_{\substack{j>c\log n\\ b_j>0}} \frac{\sqrt{n}}{2^{j-2}} \cdot \frac{1}{\sqrt{b_j} G^{-1}(1/b_j)} \\
&\le n \cdot \frac{\sqrt{n}}{2^{c\log n-2}} \cdot \frac{b_j}{\sqrt{b_j} G^{-1}(1)} \\
&\le n \cdot 2^{\widetilde{j}+2} \cdot \frac{\sqrt{n}}{2^{c\log n-2}} \cdot \sqrt{C_G n} \cdot \|x^{B_{\widetilde{j}}}\|_\ell \\
&\le \|x^{B_{\widetilde{j}}}\|_\ell \\
&\le \sum_{j'\le c\log n} \|x^{B^{j'}}\|_\ell
\end{aligned}$$

for some sufficiently large constant c.

Let

$$j^* = \mathrm{argmax}_{j\le c\log n} \|x^{B_j}\|_\ell,$$

we have

$$\begin{aligned}
\|x^{B_{j^*}}\|_\ell \le \|x\|_\ell &\le \sum_{j\le c\log n} \|x^{B_j}\|_\ell + \sum_{\substack{j>c\log n\\ b_j>0}} \|x^{B_j}\|_\ell \\
&\le 2 \sum_{j\le c\log n} \|x^{B_j}\|_\ell \\
&\le O(\log n) \cdot \|x^{B_{j^*}}\|_\ell.
\end{aligned}$$

Thus,

$$\begin{aligned}
\max_{x\in\mathbb{S}^{n'-1}} \|x\|_{\ell^{(n')}} &\le O(\log n') \max_{x\in\mathbb{S}^{n'-1}} \|x^{B_{j^*}}\|_\ell \\
&\le O(\log n') \max_{x\in\mathbb{S}^{n'-1}} \|x^{B_{j^*}}\|_2 \cdot \left\| \frac{x^{B_{j^*}}}{\|x^{B_{j^*}}\|_2} \right\|_\ell \\
&\le O(\log n') \max_{x\in\mathbb{S}^{n'-1}} \left\| \frac{x^{B_j}}{\|x^{B_j}\|_2} \right\|_\ell \\
&\le O(\log n') \max_{b_{j^*}\le n'} \|\xi^{(b_j)}\|_{\ell^{(b_j)}} \\
&\le O(\log n') \max_{b_{j^*}\le n'} \frac{1}{\overline{G}(b_{j^*})} \\
&\le \frac{O(\sqrt{C_G}\log n')}{\overline{G}(n')}.
\end{aligned}$$

Thus, we have

$$\begin{aligned}
\mathrm{mmc}(\ell) &= \max_{n'\in[n]} \frac{\max_{x\in\mathbb{S}^{n'-1}} \|x\|_{\ell^{(n')}}}{M_{\ell^{(n')}}} \\
&\le O(\sqrt{C_G}\log n).
\end{aligned}$$

□

25.7.4 Contraction Bound of SymSketch

In this section we give the contraction bound of SymSketch. We first show that for a fixed vector $x \in \mathbb{R}^n$, $\|\widetilde{D}x\|_2 \ge 1/\operatorname{poly}(d\log n)\cdot\|x\|_\ell$ with probability $1-2^{-\Theta(d\log n)}$.

Lemma 25.7.12. *Let $\widetilde{D}$ be the matrix defined in Definition 25.3.1. For any fixed $x\in\mathbb{R}^n$, with probability $1-2^{-\Theta(d\log n)}$,*

$$\|\widetilde{D}x\|_2 \ge 1/\alpha_0 \cdot \|x\|_\ell,$$

where $\alpha_0 = O(\mathrm{mma}(\ell,d)\cdot\log^{5/2} n) = O(\sqrt{d}\log^3 n)$.

Proof. The lemma follows from the following two claims. Recall that $t = \Theta(\log n)$.

Claim 25.7.13. *For any fixed $x \in \mathbb{R}^n$. If there is a contributing level $i^* \in \{0, 1, 2, \ldots, g\}$ such that $|L_{i^*}(x)| = \Theta(d \log n) \cdot 2^j$ for some $j \in [t]$, then with probability at least $1 - 2^{-\Theta(d \log n)}$,*

$$\|w_j D_j x\|_2 \geq \Omega\left(\frac{1}{\mathrm{mma}(\ell, d) \cdot \log^{5/2} n}\right) \|x\|_\ell.$$

Proof. Let y_h be a random variable such that

$$y_h = \begin{cases} 1 & \text{if the } h\text{-th diagonal entry of } D_j \text{ is } 1 \\ 0 & \text{otherwise} \end{cases}.$$

Let $Y = \sum_{h \in L_{i^*}(x)} y_h$. By Chernoff bound, we have

$$\Pr[Y \geq \Omega(d \log n)] \geq 1 - 2^{-\Theta(d \log n)}.$$

Conditioned on $Y \geq \Omega(d \log n)$, we have

$$\begin{aligned}
\frac{\|w_j D_j x\|_2}{\|x\|_\ell} &= \frac{w_j \|D_j x\|_2}{\|x\|_\ell} \\
&\geq \frac{2^{i^*} w_j \sqrt{d \log n}}{\|x\|_\ell} \\
&\geq \frac{2^{i^*} \sqrt{2^j} M_{\ell(2^j)} \sqrt{d \log n}}{2^{i^*+1} g M_{\ell(|L_{i^*}(x)|)} \sqrt{\log n} \sqrt{|L_{i^*}(x)|}} \\
&\geq \Omega(1/\log^{3/2} n) \cdot \frac{M_{\ell(2^j)}}{M_{\ell(|L_{i^*}(x)|)}} \\
&= \Omega(1/\log^{3/2} n) \cdot \frac{M_{\ell(2^j)}}{M_{\ell(|L_{i^*}(x)|/\log n)}} \cdot \frac{M_{\ell(|L_{i^*}(x)|/\log n)}}{M_{\ell(|L_{i^*}(x)|)}} \\
&\geq \Omega\left(\frac{1}{\mathrm{mma}(\ell, d) \cdot \mathrm{mma}(\ell, \log n) \cdot \log^{3/2} n}\right) \\
&\geq \Omega\left(\frac{1}{\mathrm{mma}(\ell, d) \cdot \log^{5/2} n}\right).
\end{aligned}$$

Here the first inequality follows from the fact that there are at least $\Omega(d \log n)$ coordinates sampled from $L_{i^*}(x)$. The second inequality follows from Lemma 25.7.6 and the fact that level i^* is a contributing level. The third inequality follows from $|L_{i^*}(x)| = \Theta(d \log n) \cdot 2^j$ and $g = \Theta(\log n)$. The forth inequality follows from Definition 25.7.5. The last inequality follows from Lemma 25.7.7. □

Claim 25.7.14. *For any fixed $x \in \mathbb{R}^n$. If there is a contributing level $i^* \in \{0, 1, 2, \ldots, g\}$ such that $|L_{i^*}(x)| = O(d \log n)$, then we have*

$$\|w_0 D_0 x\|_2 \geq \Omega\left(\frac{1}{\mathrm{mma}(\ell, d) \cdot \log^{5/2} n}\right) \|x\|_\ell.$$

Proof.

$$\begin{aligned}
\frac{\|w_0 D_0 x\|_2}{\|x\|_\ell} &= \frac{w_0 \|x\|_2}{\|x\|_\ell} \\
&\geq \frac{2^{i^*} w_0 \sqrt{|L_{i^*}(x)|}}{\|x\|_\ell} \\
&\geq \frac{2^{i^*} M_{\ell(1)} \sqrt{|L_{i^*}(x)|}}{2^{i^*+1} g M_{\ell(|L_{i^*}(x)|)} \sqrt{\log n} \sqrt{|L_{i^*}(x)|}} \\
&\geq \Omega(1/\log^{3/2} n) \cdot \frac{M_{\ell(1)}}{M_{\ell(|L_{i^*}(x)|/\log n)}} \cdot \frac{M_{\ell(|L_{i^*}(x)|/\log n)}}{M_{\ell(|L_{i^*}(x)|)}} \\
&\geq \Omega\left(\frac{1}{\mathrm{mma}(\ell, d) \cdot \log^{5/2} n}\right)
\end{aligned}$$

The first inequality follows from the fact that we only consider the contribution of the coordinates in $L_{i^*}(x)$. The second inequality follows from Lemma 25.7.6 and the fact that level i^* is a contributing level. The third inequality follows from $g = \Theta(\log n)$. The last inequality follows from Definition 25.7.5 and Lemma 25.7.7. □

By Claim 25.7.13 and Claim 25.7.14, since any vector $x \in \mathbb{R}^n$ contains at least one contributing level, with probability at least $1 - 2^{-\Theta(d \log n)}$ we have $\|\widetilde{D}x\|_2 \geq \Omega(1/(\mathrm{mma}(\ell, d) \cdot \log^{5/2} n)) \cdot \|x\|_\ell$. We complete the proof by combining this with Lemma 25.7.7. □

Now we show how to combine the contraction bound in Lemma 25.7.12 with a net argument to give a contraction bound for all vectors in a subspace.

Lemma 25.7.15. *Let $S \in \mathbb{R}^{O(d) \times n}$ be a random matrix. For any $\alpha_0 = \mathrm{poly}(n)$ and $A \in \mathbb{R}^{n \times d}$, if*

1. *$\|S\|_2 \leq \mathrm{poly}(n)$ holds with probability at least 0.999;*
2. *for any fixed $x \in \mathbb{R}^n$, $\|Sx\|_2 \geq 1/\alpha_0 \cdot \|x\|_\ell$ holds with probability $1 - e^{-Cd \log n}$ for a sufficiently large constant C,*

then with probability at least 0.998, for all $y \in \mathbb{R}^n$ in the column space of A,

$$\|Sy\|_2 \geq \Omega(1/\alpha_0)\|y\|_\ell.$$

Proof. For the matrix $A \in \mathbb{R}^{n \times d}$, we define the set $\mathcal{B} = \{y \mid y = Ax, \|y\|_2 = 1\}$. We define $\mathcal{N} \subset \mathbb{R}^n$ to be an ϵ-net of $\mathcal{B}$ as in Definition 25.5.1. By Lemma 25.5.1, we have $|\mathcal{N}| \leq (1 + 1/\epsilon)^d$, and for all $y \in \mathcal{B}$, there exists $z \in \mathcal{N}$ such that $\|y - z\|_2 \leq \epsilon$. We take $\epsilon = 1/\mathrm{poly}(n)$ here.

Due to the second condition, since $|\mathcal{N}| \leq e^{O(d \log n)}$, by taking union bound over all vectors in $\mathcal{N}$, we know that with probability $1 - e^{-\Theta(d \log n)}$, for all $z \in \mathcal{N}$, $\|Sz\|_2 \geq 1/\alpha_0 \cdot \|z\|_\ell$.

Now, for any vector $y \in \mathcal{B}$, there exists $z \in \mathcal{N}$ such that $\|y - z\|_2 \leq 1/\operatorname{poly}(n)$, and we define $w = y - z$.

$$
\begin{aligned}
\|Sy\|_2 &= \|S(z+w)\|_2 \\
&\geq \|Sz\|_2 - \|Sw\|_2 && \text{(triangle inequality)} \\
&\geq 1/\alpha_0 \cdot \|z\|_\ell - \|Sw\|_2 && \text{(by the second condition)} \\
&\geq 1/\alpha_0 \cdot \|z\|_\ell - \|S\|_2 \|w\|_2 && (\|Sw\|_2 \leq \|S\|_2 \cdot \|w\|_2) \\
&\geq 1/\alpha_0 \cdot \|z\|_\ell - \operatorname{poly}(n) \cdot \|w\|_2 && \text{(by the first condition)} \\
&\geq 1/\alpha_0 \cdot \|y - w\|_\ell - \operatorname{poly}(n) \cdot \|w\|_2 && (y = z + w) \\
&\geq 1/\alpha_0 \cdot \|y\|_\ell - 1/\alpha_0 \cdot \|w\|_\ell - \operatorname{poly}(n) \cdot \|w\|_2 && \text{(triangle inequality)} \\
&\geq 1/\alpha_0 \cdot \|y\|_\ell - 1/\alpha_0 \cdot \sqrt{n}\|w\|_2 - \operatorname{poly}(n) \cdot \|w\|_2 && \text{(Lemma 25.7.5)} \\
&\geq 1/\alpha_0 \cdot \|y\|_\ell - (1/\alpha_0 \cdot \sqrt{n} + \operatorname{poly}(n))\epsilon && (\|w\|_2 \leq \epsilon) \\
&\geq 0.5/\alpha_0 \cdot \|y\|_\ell.
\end{aligned}
$$

□

Lemma 25.7.16. *For a given matrix $A \in \mathbb{R}^{n \times d}$. Let $S \in \mathbb{R}^{O(d) \times n}$ be a SymSketch as defined in Definition 25.3.1. With probability at least 0.995, for all $x \in \mathbb{R}^d$, $\|SAx\|_2 \geq 1/\alpha_0 \cdot \|Ax\|_\ell$ where $\alpha_0 = O(\sqrt{d}\log^3 n)$.*

Proof. By Lemma 25.7.12 and Lemma 25.7.9, the two conditions in Lemma 25.7.15 are satisfied. By Lemma 25.7.15, with probability at least 0.998, for all $x \in \mathbb{R}^d$, $\|\widetilde{D}Ax\|_2 \geq \Omega(1/\alpha_0)\|Ax\|_\ell$. Since $\Pi \in \mathbb{R}^{O(d) \times n(t+1)}$ is a composition of Gaussian embedding and CountSketch embedding with $\epsilon = 0.1$, by Corollary 25.7.3, with probability at least 0.999, for all

$x \in \mathbb{R}^d$, $\|\Pi\widetilde{D}Ax\|_2 \geq \Omega(\|\widetilde{D}Ax\|_2)$. By a union bound, we know that with probability at least 0.995, for all $x \in \mathbb{R}^d$, $\|SAx\|_2 \geq \Omega(1/\alpha_0)\|Ax\|_\ell$. □

25.7.5 Dilation Bound of SymSketch

In this section we give the dilation bound of SymSketch. We first show that for any fixed $x \in \mathbb{R}^n$, with high probability, $\|\widetilde{D}x\|_2 \leq \mathrm{poly}(d \log n) \cdot \mathrm{mmc}(\ell) \cdot \|x\|_\ell$.

Lemma 25.7.17. *Let $\widetilde{D}$ be the matrix defined in Definition 25.3.1. For any fixed vector $x \in \mathbb{R}^n$, with probability $1 - \delta$, $\|\widetilde{D}x\|_2 \leq \alpha_1/\delta \cdot \|x\|_\ell$, where $\alpha_1 = O(\mathrm{mmc}(\ell) \log^{5/2} n)$.*

Proof. Consider a fixed vector $x \in \mathbb{R}^n$. Recall that $t = \Theta(\log n)$. Let $c > 0$ be a fixed constant. We define the j-heavy level set H_j as

$$H_j = \left\{ i \;\middle|\; |L_i(x)| \geq c\frac{\delta 2^j}{\log^2 n}, 0 \leq i \leq g \right\}.$$

Let $\overline{H}_j$ be the j-light level set, i.e., $\overline{H}_j = \{0, 1, \ldots, g\} \backslash H_j$. Notice that

$$\sum_{i \in \overline{H}_j} |L_i(x)| \cdot 2^{-j} \leq g \cdot c\delta 2^j / \log^2 n \cdot 2^{-j} \leq O(\delta / \log n).$$

By Markov's inequality, with probability at least $1 - \delta/(2t)$, no element from a j-light level is sampled by D_j, i.e., for all $i \in \overline{H}_j, k \in L_i(x)$, the k-th diagonal entry of D_j is 0. By taking union bound over all $j \in [t]$, with probability at least $1 - \delta/2$, for all $j \in [t]$, no element from a j-light level is sampled by D_j. Let ζ denote this event. We condition on this event in the remaining part of the proof. In the following analysis, we show an upper bound of $\|w_j D_j x\|_2^2$ for each $j \in [t]$. Let H_j be the set of j-heavy levels.

Consider a fixed $j \in [t]$, we have

$$\begin{aligned}
\mathop{\mathbb{E}}_{D_j}\left[\|w_j D_j x\|_2^2 \;\middle|\; \zeta\right] &= w_j^2 \mathop{\mathbb{E}}_{D_j}\left[\|D_j x\|_2^2 \middle| \zeta\right] \\
&= w_j^2 \mathop{\mathbb{E}}_{D_j}\left[\sum_{h=1}^{n} (D_j(h,h))^2 x_h^2 \;\middle|\; \zeta\right] \\
&= w_j^2 \mathop{\mathbb{E}}_{D_j}\left[\sum_{i=0}^{g} \sum_{h \in L_i(x)} (D_j(h,h))^2 x_h^2 \;\middle|\; \zeta\right] \\
&= w_j^2 \frac{1}{2^j} \sum_{i \in H_j} \sum_{h \in L_i(x)} x_h^2 \\
&\leq w_j^2 \frac{1}{2^j} \sum_{i \in H_j} |L_i(x)| \cdot (2^{i+1})^2.
\end{aligned}$$

Claim 25.7.18. $w_j^2 2^{-j} \leq O((M_{\ell^{(2^j)}})^2)$.

Proof.

$$\begin{aligned}
w_j^2 2^{-j} &= (\|(1,1,\ldots,1,0,\ldots,0)\|_\ell)^2 \cdot 2^{-j} \\
&= \left(\left\|\frac{1}{\sqrt{2^j}}(1,1,\ldots,1,0,\ldots,0)\right\|_\ell \sqrt{2^j}\right)^2 \cdot 2^{-j} \\
&= (\|\xi^{(2^j)}\|_\ell)^2 \cdot 2^j \cdot 2^{-j} \\
&= (\|\xi^{(2^j)}\|_\ell)^2 \\
&\leq O(M_{\ell^{(2^j)}}),
\end{aligned}$$

where the third step follows from the definition of $\xi^{(2^j)}$, and the last step follows from Lemma 25.7.6. □

Using the above claim, we have

$$
\begin{aligned}
& w_j^2 \sum_{i \in H_j} |L_i(x)| \cdot 2^{2i-j} \\
= & \sum_{i \in H_j} \frac{w_j^2 2^{-j}}{(M_{\ell(|L_i(x)|)})^2} \cdot |L_i(x)| \cdot (M_{\ell(|L_i(x)|)})^2 \cdot 2^{2i} \\
\leq & O\left(\sum_{i \in H_j} \left(\frac{M_{\ell(2^j)}}{M_{\ell(|L_i(x)|)}} \right)^2 \cdot |L_i(x)| \cdot (M_{\ell(|L_i(x)|)})^2 \cdot 2^{2i} \right) \\
\leq & O\left(\sum_{i \in H_j} \left(\frac{M_{\ell(2^j)}}{M_{\ell(|L_i(x)|)}} \right)^2 w_{\log |L_i(x)|}^2 \log n \cdot 2^{2i} \right) \\
= & \log n \cdot \underbrace{\sum_{i \in H_j, |L_i(x)| \leq 2^j} \left(\frac{M_{\ell(2^j)}}{M_{\ell(|L_i(x)|)}} \right)^2 \cdot w_{\log |L_i(x)|}^2 \cdot 2^{2i}}_{\diamondsuit} \\
& + \log n \cdot \underbrace{\sum_{i \in H_j, |L_i(x)| > 2^j} \left(\frac{M_{\ell(2^j)}}{M_{\ell(|L_i(x)|)}} \right)^2 \cdot w_{\log |L_i(x)|}^2 \cdot 2^{2i}}_{\heartsuit},
\end{aligned}
$$

where the second step follows from $w_j^2 2^{-j} \leq O((M_{\ell(2^j)})^2)$ (Claim 25.7.18), and the third step follows from $|L_i(x)| \cdot (M_{\ell(|L_i(x)|)})^2 \leq O(w_{\log |L_i(x)|}^2 \log n)$ (Lemma 25.7.6). It remains to upper bound $\diamondsuit$ and $\heartsuit$.

To given an upper bound for $\diamondsuit$, we have

$$\begin{aligned}
\diamondsuit &\leq O\left(\sum_{i\in H_j, |L_i(x)|\leq 2^j} \mathrm{mma}^2(\ell, \log^2 n/\delta)\cdot w^2_{\log|L_i(x)|}\cdot 2^{2i}\right)\\
&\leq O\left(\mathrm{mma}^2(\ell,\log^2 n/\delta)\left(\sum_{i=0}^{g} w_{\log|L_i(x)|}\cdot 2^i\right)^2\right)\\
&\leq O\left(\mathrm{mma}^2(\ell,\log^2 n/\delta)\right)\|x\|_\ell^2\\
&\leq O(\log^3 n/\delta)\|x\|_\ell^2,
\end{aligned}$$

where the first step follows from the definition of mma, the second step follows from Minkowski inequality, the third step follows from the definition of $L_i(x)$, $w_{\log|L_i(x)|}$ and triangle inequality, the last step follows from Lemma 25.7.7.

To give an upper bound for $\heartsuit$, we have

$$\begin{aligned}
\heartsuit &\leq O\left(\log n\cdot\sum_{i\in H_j, |L_i(x)|>2^j} \mathrm{mmc}^2(\ell) w^2_{\log|L_i(x)|}\cdot 2^{2i}\right)\\
&\leq O\left(\log n\cdot\mathrm{mmc}^2(\ell)\cdot\left(\sum_{i=0}^{g} w_{\log|L_i(x)|}\cdot 2^i\right)^2\right)\\
&\leq O\left(\log n\cdot\mathrm{mmc}^2(\ell)\cdot\|x\|_\ell^2\right),
\end{aligned}$$

where the first step follows from $(M_{\ell(2^j)}/M_{\ell(|L_i(x)|)})^2 \leq O(\log n\cdot\mathrm{mmc}^2(\ell))$ (Lemma 3.14 in [BBC+17]).

Putting it all together, we have

$$\mathop{\mathbb{E}}_{D_j}[\|w_j D_j x\|_2^2|\zeta] \leq \log n\cdot(\diamondsuit+\heartsuit)\leq O(\log^4 n/\delta+\log^2 n\cdot\mathrm{mmc}^2(\ell))\|x\|_\ell^2.$$

Thus,

$$\mathop{\mathbb{E}}_{\widetilde{D}}[\|\widetilde{D}x\|_2^2|\zeta] \leq \sum_{j=0}^{t} \mathop{\mathbb{E}}_{D_j}[\|w_j D_j x\|_2^2|\zeta] \leq O(\log^5 n/\delta + \log^3 n \cdot \mathrm{mmc}^2(\ell))\|x\|_\ell^2.$$

By Markov's inequality, conditioned on ζ, with probability at least $1 - \delta/2$,

$$\|\widetilde{D}x\|_2^2 \leq O(\log^5 n/\delta + \log^3 n \cdot \mathrm{mmc}^2(\ell))\|x\|_\ell^2/\delta.$$

Since ζ holds with probability at least $1 - \delta/2$, with probability at least $1 - \delta$, we have

$$\|\widetilde{D}x\|_2 \leq O(\log^{5/2} n/\delta \cdot \mathrm{mmc}(\ell)) \cdot \|x\|_\ell.$$

□

Now we show how to use the dilation bound for a fixed vector in Lemma 25.7.17 to prove a dilation bound for all vectors in a subspace. We need the following existential result in our proof.

Lemma 25.7.19 ([Aue30])**.** *Given a matrix $A \in \mathbb{R}^{n\times m}$ and a norm $\|\cdot\|$, there exists a basis matrix $U \in \mathbb{R}^{n\times d}$ of the column space of A, such that*

$$\sum_{i=1}^{d} \|U^i\| \leq d,$$

and for all $x \in \mathbb{R}^d$,

$$\|x\|_\infty \leq \|Ux\|.$$

Lemma 25.7.20. *Given a matrix $A \in \mathbb{R}^{n\times d}$. Let $S \in \mathbb{R}^{O(d)\times n}$ be a* SymSketch *as defined in Definition 25.3.1. With probability at least 0.99, for all $x \in \mathbb{R}^d$,*

$$\|SAx\|_2 \leq O(\alpha_1 d^2)\|Ax\|_\ell,$$

where $\alpha_1 = O(\mathrm{mmc}(\ell) \cdot \log^{5/2} n)$.

Proof. Recall that $S = \Pi\widetilde{D}$. Let U be a basis matrix of the column space of A as in Lemma 25.7.19. By Lemma 25.7.17, for a fixed $i \in [d]$, with probability at least $1 - 1/(100d)$, $\|\widetilde{D}U^i\|_2 \leq O(\alpha_1 d)\|U^i\|_\ell$. By taking a union bound over $i \in [d]$, with probability at least 0.999, for all $i \in [d]$, $\|\widetilde{D}U^i\|_2 \leq \alpha_1 d\|U^i\|_\ell$. Thus, for any $x \in \mathbb{R}^d$,

$$\begin{aligned}
\|\widetilde{D}Ux\|_2 \leq & \sum_{i=1}^d |x_i| \cdot \|\widetilde{D}U^i\|_2 \\
\leq & \|x\|_\infty \cdot \sum_{i=1}^d \|\widetilde{D}U^i\|_2 \\
\leq & \|Ux\|_\ell \cdot \sum_{i=1}^d \|\widetilde{D}U^i\|_2 \\
\leq & O(\alpha_1 d) \cdot \|Ux\|_\ell \cdot \sum_{i=1}^d \|U^i\|_\ell \\
\leq & O(\alpha_1 d^2) \cdot \|Ux\|_\ell,
\end{aligned}$$

where the first step follows from triangle inequality, the second step follows from $|x_i| \leq \|x\|_\infty$ for all $i \in [d]$, the third step follows from $\|x\|_\infty \leq \|Ux\|_\ell$, the fourth step follows from $\|\widetilde{D}U^i\|_2 \leq O(\alpha_1 d)\|U^i\|_\ell$, the last step follows from $\sum_{i=1}^d \|U^i\|_\ell \leq d$.

By Corollary 25.7.3, with probability at least 0.999, Π is an ℓ_2 subspace embedding with $\epsilon = 0.1$ for the column space of $\widetilde{D}U$. Thus, with probability at least 0.99, for all $x \in \mathbb{R}^d$, $\|SAx\|_2 \leq O(\alpha_1 d^2)\|Ax\|_\ell$. □

25.7.6 Proof of Theorem 25.3.3

Proof. It directly follows from Lemma 25.7.16, Lemma 25.7.20 and Lemma 25.7.8. □

25.8 Missing Proofs of Main Theorems

25.8.1 Proof of Theorem 25.1.3

Let $S \in \mathbb{R}^{O(d)\times n}$ be a SymSketch as defined in Definition 25.3.1, and $\Pi = O(\sqrt{d}\log^3 n)\cdot S$. By Corollary 25.3.4, for a given matrix $A \in \mathbb{R}^{n\times d}$, with probability at least 0.9, for all $x \in \mathbb{R}^d$,

$$\|Ax\|_G \le \|\Pi Ax\|_2 \le \kappa\|Ax\|_G,$$

where $\kappa = O(\sqrt{C_G}d^{5/2}\log^{13/2} n)$. We prove Theorem 25.1.3 by combining Theorem 25.2.6 with the embedding matrix Π constructed above.

25.8.2 Proof of Theorem 25.1.4

Let $S \in \mathbb{R}^{O(d)\times n}$ be a SymSketch as defined in Definition 25.3.1. For a given data matrix $A \in \mathbb{R}^{n\times d}$ and response vector $b \in \mathbb{R}^n$, we calculate $x^* = \operatorname{argmin}_x \|SAx - Sb\|_2$ and return x^*. The algorithm runs in $O(\mathrm{nnz}(A) + \mathrm{poly}(d))$ time, since by Lemma 25.7.8, the expected running time for calculating SA is $O(\mathrm{nnz}(A) + \mathrm{poly}(d))$, and $x^* = (SA)^+Sb$ can be calculated in $\mathrm{poly}(d)$ time.

To see the correctness, let $\overline{x} = \operatorname{argmin}_x \|Ax - b\|_\ell$. With probability at least 0.99, we have

$$\begin{aligned}
\|Ax^* - b\|_\ell &\le O(\sqrt{d}\log^3 n)\|SAx^* - Sb\|_2 \\
&\le O(\sqrt{d}\log^3 n)\|SA\overline{x} - Sb\|_2 \\
&\le O(\sqrt{d}\log^3 n)\|\widetilde{D}A\overline{x} - \widetilde{D}b\|_\ell \\
&= O(\sqrt{d}\log^{11/2} n)\cdot \mathrm{mmc}(\ell)\cdot\|A\overline{x} - b\|_\ell.
\end{aligned}$$

The first step follows by applying Lemma 25.7.16 on $\mathbf{Ab}$, where we use $\mathbf{Ab} \in \mathbb{R}^{n \times (d+1)}$ to denote a matrix whose first d columns are A and the last column is b. The second step follows from the fact that $x^* = \text{argmin}_x \|SAx - Sb\|_2$. The third step follows by Definition 25.3.1 and Corollary 25.7.3. The last step follows by applying Lemma 25.7.17 on $A\overline{x} - b$.

Chapter 26

Kronecker Product Regression

We study Kronecker product regression, in which the design matrix to a regression problem is a Kronecker product of two or more matrices. Formally, given $A_i \in \mathbb{R}^{n_i \times d_i}$ for $i = 1, 2, \ldots, q$ where $n_i \gg d_i$ for each i, and $b \in \mathbb{R}^{n_1 n_2 \cdots n_q}$, let $\mathcal{A} = A_i \otimes A_2 \otimes \cdots \otimes A_q$. Then for $p \in [1, 2]$, the goal is to find $x \in \mathbb{R}^{d_1 \cdots d_q}$ that approximately minimizes $\|\mathcal{A}x - b\|_p$. Recently, Diao, Song, Sun, and Woodruff (AISTATS, 2018) gave an algorithm which is faster than forming the Kronecker product $\mathcal{A} \in \mathbb{R}^{n_1 \cdots n_q \times d_1 \cdots d_q}$. Specifically, for $p = 2$ they achieve a running time of $O(\sum_{i=1}^{q} \text{nnz}(A_i) + \text{nnz}(b))$, where $\text{nnz}(A_i)$ is the number of non-zero entries in A_i. Note that $\text{nnz}(b)$ can be as large as $\Theta(n_1 \cdots n_q)$. For $p = 1$, $q = 2$ and $n_1 = n_2$, they achieve a worse bound of $O(n_1^{3/2}\text{poly}(d_1 d_2) + \text{nnz}(b))$. In this work, we provide significantly faster algorithms. For $p = 2$, our running time is $O(\sum_{i=1}^{q} \text{nnz}(A_i))$, which has no dependence on $\text{nnz}(b)$. For $p < 2$, our running time is $O(\sum_{i=1}^{q} \text{nnz}(A_i) + \text{nnz}(b))$, which matches the prior best running time for $p = 2$. We also consider the related all-pairs regression problem, where given $A \in \mathbb{R}^{n \times d}, b \in \mathbb{R}^n$, we want to solve $\min_{x \in \mathbb{R}^d} \|\bar{A}x - \bar{b}\|_p$, where $\bar{A} \in \mathbb{R}^{n^2 \times d}, \bar{b} \in \mathbb{R}^{n^2}$ consist of all pairwise differences of the rows of A, b. We give an $O(\text{nnz}(A))$ time algorithm for $p \in [1, 2]$, improving the $\Omega(n^2)$ time required to form $\bar{A}$. Finally, we initiate the study of Kronecker product low rank approximation and low-trank approximation. For input $\mathcal{A}$ as above, we give an $O(\sum_{i=1}^{q} \text{nnz}(A_i))$ time algorithm, which is much faster than computing $\mathcal{A}$.

2023

This part is based upon the following previous publication

- Huaian Diao, Rajesh Jayaram, Zhao Song, Wen Sun, David P. Woodruff
  Optimal Sketching for Kronecker Product Regression and Low Rank Approximation.
  Manuscript 2019 [DJS+19]

26.1 Introduction

In the q-th order Kronecker product regression problem, one is given $A_i \in \mathbb{R}^{n_i \times d_i}$ for each $i \in \{1, 2, \ldots, q\}$, together with $b \in \mathbb{R}^{n_1 n_2 \cdots n_q}$, and the goal is to obtain a solution to the optimization problem:

$$\min_{x \in \mathbb{R}^{d_1 d_2 \cdots d_q}} \|(A_1 \otimes A_2 \cdots \otimes A_q)x - b\|_p,$$

where $p \in [1, 2]$, and for a vector $x \in \mathbb{R}^n$ the ℓ_p norm is defined by $\|x\|_p = (\sum_{i=1}^n |x_i|^p)^{1/p}$. For $p = 2$, this is least squares regression, and for $p = 1$ this is least absolute deviation regression.

Kronecker product regression is a special case of ordinary regression, in which the design matrix is highly structured. Namely, it is the Kronecker product of two or more matrices. Kronecker product matrices naturally arise in applications such as spline regression, signal processing, and multivariate data fitting. We refer the reader to [VL92, VLP93, GVL13] for further background and applications of Kronecker product regression. As discussed in [DSSW18], Kronecker product regression also arises in structured blind deconvolution problems [OY05], and the bivariate problem of surface fitting and multidimensional density smoothing [EM06].

A recent work of Diao, Song, Sun, and Woodruff [DSSW18] utilizes *sketching* techniques to output an $x \in \mathbb{R}^{d_1 d_2 \cdots d_q}$ with objective function at most $(1+\epsilon)$-times larger than optimal, for both least squares and least absolute deviation Kronecker product regression. Importantly, their time complexity is faster than the time needed to explicitly compute $A_1 \otimes \cdots \otimes A_q$. We note that sketching itself is a powerful tool for compressing extremely high dimensional data, and has been used in a number of tensor related problems, e.g., [SWZ16, LHW17, DSSW18, SWZ19b].

For least squares regression, [DSSW18] achieve $O(\sum_{i=1}^{q} \text{nnz}(A_i) + \text{nnz}(b) + \text{poly}(d/\epsilon))$ time, where $\text{nnz}(C)$ for a matrix C denotes the number of non-zero entries of C. Note that the focus is on the over-constrained regression setting, when $n_i \gg d_i$ for each i, and so the goal is to have a small running time dependence on the n_i. We remark that over-constrained regression has been the focus of a large body of work over the past decade, which primarily attempts to design fast regression algorithms in the big data (large sample size) regime, see, e.g., [Mah11b, Woo14b] for surveys.

Observe that explicitly computing $A_1 \otimes \cdots \otimes A_q$ would take $\prod_{i=1}^{q} \text{nnz}(A_i)$ time, which can be as large as $\prod_{i=1}^{q} n_i d_i$, and so the results of [DSSW18] offer a large computational advantage. Unfortunately, since $b \in \mathbb{R}^{n_1 n_2 \cdots n_q}$, we can have $\text{nnz}(b) = \prod_{i=1}^{q} n_i$, and therefore $\text{nnz}(b)$ is likely to be the dominant term in the running time. This leaves open the question of whether it is possible to solve this problem in time *sub-linear* in $\text{nnz}(b)$, with a dominant term of $O(\sum_{i=1}^{q} \text{nnz}(A_i))$.

For least absolute deviation regression, the bounds of [DSSW18] achieved are still an improvement over computing $A_1 \otimes \cdots \otimes A_q$, though worse than the bounds for least squares regression. The authors focus on $q = 2$ and the special case $n = n_1 = n_2$. Here, they obtain

a running time of $O(n^{3/2}\,\mathrm{poly}(d_1 d_2/\epsilon) + \mathrm{nnz}(b))$[1]. This leaves open the question of whether an *input-sparsity* $O(\mathrm{nnz}(A_1) + \mathrm{nnz}(A_2) + \mathrm{nnz}(b) + \mathrm{poly}(d_1 d_2/\epsilon))$ time algorithm exists.

In this work, we also study the related all-pairs regression problem. Given $A \in \mathbb{R}^{n\times d}, b \in \mathbb{R}^n$, the goal is to approximately solve the ℓ_p regression problem $\min_x \|\bar{A}x - \bar{b}\|_p$, where $\bar{A} \in \mathbb{R}^{n^2\times d}$ is the matrix formed by taking all pairwise differences of the rows of A (and $\bar{b}$ is defined similarly). For $p = 1$, this is known as the *rank regression estimator*, which has a long history in statistics. It is closely related to the renowned Wilconxon rank test [WL09], and enjoys the desirable property of being robust with substantial efficiency gain with respect to heavy-tailed random errors, while maintaining high efficiency for Gaussian errors [WKL09, WL09, WPB+18, Wan19a]. In many ways, it has properties more desirable in practice than that of the Huber M-estimator [WPB+18, Wan19b]. Recently, the all-pairs loss function was also used by [WPB+18] as an alternative approach to overcoming the challenges of tuning parameter selection for the Lasso algorithm. However, the rank regression estimator is computationally intensive to compute, even for moderately sized data, since the standard procedure (for $p = 1$) is to solve a linear program with $O(n^2)$ constraints. In this work, we demonstrate the first highly efficient algorithm for this estimator.

Finally, in addition to regression, we extend our techniques to the Low Rank Approximation (LRA) problem. Here, given a large data matrix A, the goal is to find a low rank matrix B which well-approximates A. LRA is useful in numerous applications, such as

[1]We remark that while the $\mathrm{nnz}(b)$ term is not written in the Theorem of [DSSW18], their approach of leverage score sampling from a well-conditioned basis requires one to sample from a well conditioned basis of $[A_1 \otimes A_2, b]$ for a subspace embedding. As stated, their algorithm only sampled from $[A_1 \otimes A_2]$. To fix this omission, their algorithm would require an additional $\mathrm{nnz}(b)$ time to leverage score sample from the augmented matrix.

compressing massive datasets to their primary components for storage, denoising, and fast matrix-vector products. Thus, designing fast algorithms for approximate LRA has become a large and highly active area of research; see [Woo14b] for a survey. For an incomplete list of recent work using sketching techniques for LRA, see [CW13, MM13, NN13a, BW14, CW15b, CW15a, RSW16, BWZ16, SWZ17, MW17, CGK+17c, LHW17, SWZ18, BW18, SWZ19b, BBB+19b, IVWW19] and the references therein.

Motivated by the importance of LRA, we initiate the study of low-rank approximation of Kronecker product matrices. Given q matrices $A_1, \cdots A_q$ where $A_i \in \mathbb{R}^{n_i \times d_i}$, $n_i \gg d_i$, $A = \otimes_{i=1}^{q} A_i$, the goal is to output a rank-k matrix $B \in \mathbb{R}^{n \times d}$ such that $\|B - A\|_F^2 \leq (1+\epsilon)\,\mathrm{OPT}_k$, where OPT_k is the cost of the best rank-k approximation, $n = n_1 \cdots n_q$, and $d = d_1 \cdots d_q$. Here $\|A\|_F^2 = \sum_{i,j} A_{i,j}^2$. The fastest general purpose algorithms for this problem run in time $O(\mathrm{nnz}(A) + \mathrm{poly}(dk/\epsilon))$ [CW13]. However, as in regression, if $A = \otimes_{i=1}^{q} A_i$, we have $\mathrm{nnz}(A) = \prod_{i=1}^{q} \mathrm{nnz}(A_i)$, which grows very quickly. Instead, one might also hope to obtain a running time of $O(\sum_{i=1}^{q} \mathrm{nnz}(A_i) + \mathrm{poly}(dk/\epsilon))$.

26.1.1 Our Contributions

Our main contribution is an input sparsity time $(1+\epsilon)$-approximation algorithm to Kronecker product regression for every $p \in [1, 2]$, and $q \geq 2$. Given $A_i \in \mathbb{R}^{n_i \times d_i}$, $i = 1, \ldots, q$, and $b \in \mathbb{R}^n$ where $n = \prod_{i=1}^{q} n_i$, together with accuracy parameter $\epsilon \in (0, 1/2)$ and failure probability $\delta > 0$, the goal is to output a vector $x' \in \mathbb{R}^d$ where $d = \prod_{i=1}^{q} d_i$ such that $\|(A_1 \otimes \cdots \otimes A_q)x' - b\|_p \leq (1+\epsilon)\min_x \|(A_1 \otimes \cdots \otimes A_q)x - b\|_p$ holds with probability at least $1 - \delta$. For $p = 2$, our algorithm runs in $\widetilde{O}(\sum_{i=1}^{q} \mathrm{nnz}(A_i)) + \mathrm{poly}(d\delta^{-1}/\epsilon))$ time[2]. Notice

[2]For a function $f(n, d, \epsilon, \delta)$, $\widetilde{O}(f) = O(f \cdot \mathrm{poly}(\log n))$, where we assume $n \geq d, \epsilon^{-1}, \delta^{-1}$

that this is *sub-linear* in the input size, since it does not depend on $\mathrm{nnz}(b)$. For $p < 2$, the running time is $\widetilde{O}((\sum_{i=1}^{q} \mathrm{nnz}(A_i) + \mathrm{nnz}(b) + \mathrm{poly}(d/\epsilon)) \log(1/\delta))$.

Observe that in both cases, this running time is significantly faster than the time to write down $A_1 \otimes \cdots \otimes A_q$. For $p = 2$, up to logarithmic factors, the running time is the same as the time required to simply read each of the A_i. Moreover, in the setting $p < 2$, $q = 2$ and $n_1 = n_2$ considered in [DSSW18], our algorithm offers a substantial improvement over their running time of $O(n^{3/2} \mathrm{poly}(d_1 d_2/\epsilon))$.

We empirically evaluate our Kronecker product regression algorithm on exactly the same datasets as those used in [DSSW18]. For $p \in \{1, 2\}$, the accuracy of our algorithm is nearly the same as that of [DSSW18] while the running time is significantly faster.

For the all-pairs (or rank) regression problem, we first note that for $A \in \mathbb{R}^{n \times d}$, one can rewrite $\bar{A} \in \mathbb{R}^{n^2 \times d}$ as the difference of Kronecker products $\bar{A} = A \otimes \mathbf{1}^n - \mathbf{1}^n \otimes A$ where $\mathbf{1}^n \in \mathbb{R}^n$ is the all ones vector. Since $\bar{A}$ is not a Kronecker product itself, our earlier techniques for Kronecker product regression are not directly applicable. Therefore, we utilize new ideas, in addition to careful sketching techniques, to obtain an $\widetilde{O}(\mathrm{nnz}(A) + \mathrm{poly}(d/\epsilon))$ time algorithm for $p \in [1, 2]$, which improves substantially on the $O(n^2 d)$ time required to even compute $\bar{A}$, by a factor of at least n.

Our main technical contribution for both our ℓ_p regression algorithm and the rank regression problem is a novel and highly efficient ℓ_p sampling algorithm. Specifically, for the rank-regression problem we demonstrate, for a given $x \in \mathbb{R}^d$, how to independently sample s entries of a vector $\bar{A}x = y \in \mathbb{R}^{n^2}$ from the ℓ_p distribution $(|y_1|^p / \|y\|_p^p, \ldots, |y_{n^2}|^p / \|y\|_p^p)$ in $\widetilde{O}(nd + \mathrm{poly}(ds))$ time. For the ℓ_p regression problem, we demonstrate the same result when

$y = (A_1 \otimes \cdots \otimes A_q)x - b \in \mathbb{R}^{n_1 \cdots n_q}$, and in time $\widetilde{O}(\sum_{i=1}^{q} \text{nnz}(A_i) + \text{nnz}(b) + \text{poly}(ds))$. This result allows us to sample a small number of rows of the input to use in our sketch. Our algorithm draws from a large number of disparate sketching techniques, such as the dyadic trick for quickly finding heavy hitters [CM04, KNPW11, LNNT16, NS19], and the precision sampling framework from the streaming literature [AKO10].

For the Kronecker Product Low-Rank Approximation (LRA) problem, we give an input sparsity $O(\sum_{i=1}^{q} \text{nnz}(A_i) + \text{poly}(dk/\epsilon))$-time algorithm which computes a rank-k matrix B such that $\|B - \otimes_{i=1}^{q} A_i\|_F^2 \leq (1+\epsilon) \min_{\text{rank}-k\ B'} \|B' - \otimes_{i=1}^{q} A_i\|_F^2$. Note again that the dominant term $\sum_{i=1}^{q} \text{nnz}(A_i)$ is substantially smaller than the $\text{nnz}(A) = \prod_{i=1}^{q} \text{nnz}(A_i)$ time required to write down the Kronecker Product A, which is also the running time of state-of-the-art general purpose LRA algorithms [CW13, MM13, NN13a]. Thus, our results demonstrate that substantially faster algorithms for approximate LRA are possible for inputs with a Kronecker product structure. Our technical contributions involve demonstrating that useful properties of known sketching matrices hold also for the Kronecker product of these matrices.

In addition, motivated by [VL00], we use our techniques to solve the low-trank approximation problem, where we are given an arbitrary matrix $A \in \mathbb{R}^{n^q \times n^q}$, and the goal is to output a trank-k matrix $B \in \mathbb{R}^{n^q \times n^q}$ such that $\|B - A\|_F$ is minimized. Here, the trank of a matrix B is the smallest integer k such that B can be written as a summation of k matrices, where each matrix is the Kronecker product of q matrices with dimensions $n \times n$. Compressing a matrix A to a low-trank approximation yields many of the same benefits as LRA, such as compact representation, fast matrix-vector product, and fast matrix multiplication, and thus is applicable in many of the settings where LRA is used. Using similar

sketching ideas, we provide an $O(\sum_{i=1}^{q} \text{nnz}(A_i) + \text{poly}(d_1 \cdots d_q/\epsilon))$ time algorithm for this problem under various loss functions. Our results for low-trank approximation can be found in Section 26.11.

26.2 Preliminaries

Notation. For a tensor $A \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, we use $\|A\|_p$ to denote the entry-wise ℓ_p norm of A, i.e., $\|A\|_p = (\sum_{i_1} \sum_{i_2} \sum_{i_3} |A_{i_1,i_2,i_3}|^p)^{1/p}$. For $n \in \mathbb{N}$, let $[n] = \{1, 2, \ldots, n\}$. For a matrix A, let $A_{i,*}$ denote the i-th row of A, and $A_{*,j}$ the j-th column. For $a, b \in \mathbb{R}$ and $\epsilon \in (0, 1)$, we write $a = (1 \pm \epsilon)b$ to denote $(1 - \epsilon)b \leq a \leq (1 + \epsilon)b$. We now define various sketching matrices used by our algorithms.

p-stable Transforms. We will utilize the well-known p-stable distribution, $\mathcal{D}_p$ (see [Nol07, Ind06] for further discussion), which exist for $p \in (0, 2]$. For $p \in (0, 2)$, $X \sim \mathcal{D}_p$ is defined by its characteristic function $\mathbb{E}_X[\exp(\sqrt{-1}tX)] = \exp(-|t|^p)$, and can be efficiently generated to a fixed precision [Nol07, KNW10b]. For $p = 2$, $\mathcal{D}_2$ is just the standard Gaussian distribution, and for $p = 1$, $\mathcal{D}_1$ is the *Cauchy* distribution. The distribution $\mathcal{D}_p$ has the property that if $z_1, \ldots, z_n \sim D_p$ are i.i.d., and $a \in \mathbb{R}^n$, then $\sum_{i=1}^{n} z_i a_i \sim z\|a\|_p$ where $\|a\|_p = (\sum_{i=1}^{n} |a_i|^p)^{1/p}$, and $z \sim \mathcal{D}_p$. This property will allow us to utilize sketches with entries independently drawn from $\mathcal{D}_p$ to preserve the ℓ_p norm.

Definition 26.2.1 (Dense p-stable Transform, [CDMI+13])**.** Let $p \in [1, 2]$. Let $S = \sigma \cdot C \in \mathbb{R}^{m \times n}$, where σ is a scalar, and each entry of $C \in \mathbb{R}^{m \times n}$ is chosen independently from $\mathcal{D}_p$.

We will also need a sparse version of the above.

Definition 26.2.2 (Sparse p-Stable Transform, [MM13, CDMI+13])**.** Let $p \in [1, 2]$. Let $\Pi = \sigma \cdot SC \in \mathbb{R}^{m \times n}$, where σ is a scalar, $S \in \mathbb{R}^{m \times n}$ has each column chosen independently

and uniformly from the m standard basis vectors of $\mathbb{R}^m$, and $C \in \mathbb{R}^{n \times n}$ is a diagonal matrix with diagonals chosen independently from the standard p-stable distribution. For any matrix $A \in \mathbb{R}^{n \times d}$, ΠA can be computed in $O(\mathrm{nnz}(A))$ time.

One nice property of p-stable transformations is that they provide *low-distortion ℓ_p embeddings*.

Lemma 26.2.1 (Theorem 1.4 of [WW19]; see also Theorem 2 and 4 of [MM13] for earlier work [3])**.** *Fix $A \in \mathbb{R}^{n \times d}$, and let $S \in \mathbb{R}^{k \times n}$ be a sparse or dense p-stable transform for $p \in [1, 2)$, with $k = \Theta(d^2/\delta)$. Then with probability $1 - \delta$, for all $x \in \mathbb{R}^d$:*

$$\|Ax\|_p \leq \|SAx\|_p \leq O(d \log d)\|Ax\|_p$$

We simply call a matrix $S \in \mathbb{R}^{k \times n}$ a low distortion ℓ_p embedding for $A \in \mathbb{R}^{n \times d}$ if it satisfies the above inequality for all $x \in \mathbb{R}^d$.

Leverage Scores & Well Condition Bases. We now introduce the notions of ℓ_2 leverage scores and well-conditioned bases for a matrix $A \in \mathbb{R}^{n \times d}$.

Definition 26.2.3 (ℓ_2-Leverage Scores, [Woo14b, BSS12])**.** Given a matrix $A \in \mathbb{R}^{n \times d}$, let $A = Q \cdot R$ denote the QR factorization of matrix A. For each $i \in [n]$, we define $\sigma_i = \frac{\|(AR^{-1})_i\|_2^2}{\|AR^{-1}\|_F^2}$, where $(AR^{-1})_i \in \mathbb{R}^d$ is the i-th row of matrix $(AR^{-1}) \in \mathbb{R}^{n \times d}$. We say that $\sigma \in \mathbb{R}^n$ is the ℓ_2 leverage score vector of A.

[3] In discussion with the authors of these works, the original $O((d \log d)^{1/p})$ distortion factors stated in these papers should be replaced with $O(d \log d)$; as we do not optimize the poly(d) factors in our analysis, this does not affect our bounds.

Algorithm 26.1 Our ℓ_2 Kronecker Product Regression Algorithm

procedure ℓ_2 KRONECKER REGRESSION($(\{A_i, n_i, d_i\}_{i\in[q]}, b)$) ▷ Theorem 26.3.1
  $d \leftarrow \prod_{i=1}^q d_i$, $n \leftarrow \prod_{i=1}^q n_i$, $m \leftarrow \Theta(d/(\delta\epsilon^2))$.
  Compute approximate leverage scores $\widetilde{\sigma}_i(A_j)$ for all $j \in [q]$, $i \in [n_j]$. ▷ Lemma 26.7.3
  Construct diagonal leverage score sampling matrix $D \in \mathbb{R}^{n\times n}$, with m non-zero entries ▷ Proposition 26.7.4
  Compute (via the psuedo-inverse)
    $$\widehat{x} = \arg\min_{x\in\mathbb{R}^d} \|D(A_1 \otimes A_2 \otimes \cdots \otimes A_q)x - Db\|_2$$
  return $\widehat{x}$
end procedure

Definition 26.2.4 ((ℓ_p, α, β) Well-Conditioned Basis, [Cla05])**.** Given a matrix $A \in \mathbb{R}^{n\times d}$, we say $U \in \mathbb{R}^{n\times d}$ is an (ℓ_p, α, β) well-conditioned basis for the column span of A if the columns of U span the columns of A, and if for any $x \in \mathbb{R}^d$, we have $\alpha\|x\|_p \leq \|Ux\|_p \leq \beta\|x\|_p$, where $\alpha \leq 1 \leq \beta$. If $\beta/\alpha = d^{O(1)}$, then we simply say that U is an ℓ_p well conditioned basis for A.

Fact 26.2.2 ([WW19, MM13])**.** *Let $A \in \mathbb{R}^{n\times d}$, and let $SA \in \mathbb{R}^{k\times d}$ be a low distortion ℓ_p embedding for A (see Lemma 26.2.1), where $k = O(d^2/\delta)$. Let $SA = QR$ be the QR decomposition of SA. Then AR^{-1} is an ℓ_p well-conditioned basis with probability $1 - \delta$.*

26.3 Kronecker Product Regression

We first introduce our algorithm for $p = 2$. Our algorithm for $1 \leq p < 2$ is given in Section 26.3.1. Our regression algorithm for $p = 2$ is formally stated in Algorithm 26.1. Recall that our input design matrix is $A = \otimes_{i=1}^q A_i$, where $A_i \in \mathbb{R}^{n_i\times d_i}$, and we are also given $b \in \mathbb{R}^{n_1\cdots n_q}$. Let $n = \prod_{i=1}^q n_i$ and $d = \prod_{i=1}^q d_i$. The crucial insight of the algorithm is that one can approximately compute the leverage scores of A given only good approximations to the leverage scores of each A_i. Applying this fact gives a efficent algorithm for sampling rows of A with probability proportional to the leverage scores. Following standard arguments, we

Algorithm 26.2 Our ℓ_p Kronecker Product Regression Algorithm, $1 \leq p < 2$

procedure $O(1)$-APPROXIMATE ℓ_p REGRESSION($\{A_i, n_i, d_i\}_{i\in[q]}$) ▷ Theorem 26.3.2
  $d \leftarrow \prod_{i=1}^{q} d_i$, $n \leftarrow \prod_{i=1}^{q} n_i$.
  for $i = 1, \ldots, q$ **do**
    $s_i \leftarrow O(qd_i^2)$
    Generate sparse p-stable transform $S_i \in \mathbb{R}^{s_i \times n}$ (def 26.2.2) ▷ Lemma 26.2.1
    Take the QR factorization of $S_i A_i = Q_i R_i$ to obtain $R_i \in \mathbb{R}^{d_i \times d_i}$ ▷ Fact 26.2.2
    Let $Z \in \mathbb{R}^{d \times \tau}$ be a dense p-stable transform for $\tau = \Theta(\log(n))$ ▷ Definition 26.2.1
    for $j = 1, \ldots, n_i$ **do**
      $a_{i,j} \leftarrow \text{median}_{\eta\in[\tau]}\{(|(A_i R_i^{-1} Z)_{j,\eta}|/\theta_p)^p\}$, where θ_p is the median of $\mathcal{D}_p$.
    end for
  end for
  Define a distribution $\mathcal{D} = \{q'_1, q'_1, \ldots, q'_n\}$ by $q'_{\sum_{i=1}^{q} j_i \prod_{l=1}^{j-1} n_l} = \prod_{i=1}^{q} a_{i,j_i}$.
  Let $\Pi \in \mathbb{R}^{n\times n}$ denote a diagonal sampling matrix, where $\Pi_{i,i} = 1/q_i^{1/p}$ with probability $q_i = \min\{1, r_1 q'_i\}$ and 0 otherwise, where $r_1 = \Theta(d^3/\epsilon^2)$. ▷ [DDH+09]
  Let $x' \in \mathbb{R}^d$ denote the solution of
    $\min_{x\in\mathbb{R}^d} \|\Pi(A_1 \otimes A_2 \otimes \cdots \otimes A_q)x - \Pi b\|_p$
  return x' ▷ x' is an $O(1)$ approx: Lemma 26.8.4
end procedure
procedure $(1+\epsilon)$-APPROXIMATE ℓ_p REGRESSION($x' \in \mathbb{R}^d$)
  Implicitly define $\rho = (A_1 \otimes A_2 \otimes \cdots \otimes A_q)x' - b \in \mathbb{R}^n$
  Via Lemma 26.8.7, compute a diagonal sampling matrix $\Sigma \in \mathbb{R}^{n\times n}$ such that $\Sigma_{i,i} = 1/\alpha_i^{1/p}$ with probability $\alpha_i = \min\{1, \max\{q_i, r_2|\rho_i|^p/\|\rho\|_p^p\}\}$ where $r_2 = \Theta(d^3/\epsilon^3)$.
  Compute $\widehat{x} = \arg\min_{x\in\mathbb{R}^d} \|\Sigma(A_1 \otimes A_2 \otimes \cdots \otimes A_q) - \Sigma b\|_p$ (via convex optimization methods, e.g., [BCLL18])
  return $\widehat{x}$
end procedure

will show that by restricting the regression problem to the sampled rows, we can obtain our desired $(1 \pm \epsilon)$-approximate solution efficiently.

Our main theorem for this section is stated below. A full proof of the theorem can be found in the supplementary, namely, Section 26.7.

Theorem 26.3.1 (Kronecker product ℓ_2 regression)**.** *Let $D \in \mathbb{R}^{n \times n}$ be the diagonal row sampling matrix generated via Proposition 26.7.4, with $m = \Theta(1/(\delta\epsilon^2))$ non-zero entries, and let $A = \otimes_{i=1}^{q} A_i$, where $A_i \in \mathbb{R}^{n_i \times d_i}$, and $b \in \mathbb{R}^n$, where $n = \prod_{i=1}^{q} n_i$ and $d = \prod_{i=1}^{q} d_i$. Then let $\widehat{x} = \arg\min_{x \in \mathbb{R}^d} \|DAx - Db\|_2$, and let $x^* = \arg\min_{x' \in \mathbb{R}^d} \|Ax - b\|_2$. Then with probability $1 - \delta$, we have*

$$\|A\widehat{x} - b\|_2 \leq (1 + \epsilon)\|Ax^* - b\|_2.$$

Moreover, the total running time required to compute $\widehat{x}$ is $\widetilde{O}(\sum_{i=1}^{q} \mathrm{nnz}(A_i) + (dq/(\delta\epsilon))^{O(1)})$.

26.3.1 Kronecker Product ℓ_p Regression

We now consider ℓ_p regression for $1 \leq p < 2$. Our algorithm is stated formally in Algorithm 26.2. Below is our main theorem, which demonstrates that Algorithm 26.2 approximately solves the Kronecker ℓ_p regression problem with input sparsity running time.

Theorem 26.3.2 (Main result, ℓ_p $(1+\epsilon)$-approximate regression)**.** *Fix $1 \leq p < 2$. Then for any constant $q = O(1)$, given matrices $A_1, A_2, \cdots, A_q$, where $A_i \in \mathbb{R}^{n_i \times d_i}$, let $n = \prod_{i=1}^{q} n_i$, $d = \prod_{i=1}^{q} d_i$. Let $\widehat{x} \in \mathbb{R}^d$ be the output of Algorithm 26.2. Then*

$$\|(A_1 \otimes A_2 \otimes \cdots \otimes A_q)\widehat{x} - b\|_p \leq (1 + \epsilon) \min_{x \in \mathbb{R}^n} \|(A_1 \otimes A_2 \otimes \cdots \otimes A_q)x - b\|_p$$

holds with probability at least $1-\delta$. In addition, our algorithm takes $\widetilde{O}\left(\left(\sum_{i=1}^{q} \mathrm{nnz}(A_i) + \mathrm{nnz}(b) + (d/\epsilon)^{O(1)}\right)\right.$ time to output $\widehat{x} \in \mathbb{R}^d$.

We give a complete proof of the above theorem in the supplementary material, namely, in Section 26.8. Our high level approach follows that of [DDH+09]. Namely, we first obtain

a vector x' which is an $O(1)$-approximate solution to the optimal solution. This is done by first constructing (implicitly) a matrix $U \in \mathbb{R}^{n \times d}$ that is a well-conditioned basis for the design matrix $A_1 \otimes \cdots \otimes A_q$. We then efficiently sample rows of U with probability proportional to their ℓ_p norm (which must be done without even explicitly computing most of U). We then use the results of [DDH+09] to demonstrate that solving the regression problem constrained to these sampled rows gives a solution $x' \in \mathbb{R}^d$ such that $\|(A_1 \otimes \cdots \otimes A_q)x' - b\|_p \leq 8 \min_{x \in \mathbb{R}^d} \|(A_1 \otimes \cdots \otimes A_q)x' - b\|_p$.

We define the *residual error* $\rho = (A_1 \otimes \cdots \otimes A_q)x' - b \in \mathbb{R}^n$ of x'. Our goal is to sample additional rows $i \in [n]$ with probability proportional to their residual error $|\rho_i|^p / \|\rho\|_p^p$, and solve the regression problem restricted to the sampled rows. However, we cannot afford to compute even a small fraction of the entries in ρ (even when b is dense, and certainly not when b is sparse). So to carry out this sampling efficiently, we design an involved, multi-part sketching and sampling routine (described in Section 26.8). This sampling technique is the main technical contribution of this section, and relies on a number of techniques, such as the Dyadic trick for quickly finding heavy hitters in the streaming literature, and a careful pre-processing step to avoid a poly(d)-blow up in the runtime. Given these samples, we can obtain the solution $\widehat{x}$ after solving the regression problem on the sampled rows, and the fact that this gives a $(1+\epsilon)$ approximate solution will follow from Theorem 6 of [DDH+09].

26.4 All-Pairs Regression

Given a matrix $A \in \mathbb{R}^{n \times d}$ and $b \in \mathbb{R}^n$, let $\bar{A} \in \mathbb{R}^{n^2 \times d}$ be the matrix such that $\bar{A}_{i+(j-1)n,*} = A_{i,*} - A_{j,*}$, and let $\bar{b} \in \mathbb{R}^{n^2}$ be defined by $\bar{b}_{i+(j-1)n} = b_i - b_j$. Thus, $\bar{A}$ consists of all pairwise differences of rows of A, and $\bar{b}$ consists of all pairwise differences of rows of b,.

The ℓ_p all pairs regression problem on the inputs A, b is to solve $\min_{x \in \mathbb{R}^d} \|\bar{A}x - \bar{b}\|_p$.

First note that this problem has a close connection to Kronecker product regression. Namely, the matrix $\bar{A}$ can be written $\bar{A} = A \otimes \mathbf{1}^n - \mathbf{1}^n \otimes A$, where $\mathbf{1}^n \in \mathbb{R}^n$ is the all 1's vector. Similarly, $\bar{b} = b \otimes \mathbf{1}^n - \mathbf{1}^n \otimes b$. For simplicity, we now drop the superscript and write $\mathbf{1} = \mathbf{1}^n$.

Our algorithm is given formally in Figure 26.3. We generate sparse p-stable sketches $S_1, S_2 \in \mathbb{R}^{k \times n}$, where $k = (d/(\epsilon\delta))^{O(1)}$. We compute $M = (S_1 \otimes S_2)(F \otimes \mathbf{1} - \mathbf{1} \otimes F) = S_1 F \otimes S_2 \mathbf{1} - S_1 \mathbf{1} \otimes S_2 F$, where $F = [A, b]$. We then take the QR decomposition $M = QR$. Finally, we sample rows of $(F \otimes \mathbf{1} - \mathbf{1} \otimes F)R^{-1}$ with probability proportional to their ℓ_p norms. This is done by an involved sampling procedure described in Lemma 26.4.2, which is similar to the sampling procedure used in the proof of Theorem 26.3.2. Finally, we solve the regression problem $\min_x \|\Pi(\bar{A}x - \bar{b})\|_p$, where Π is the diagonal row-sampling matrix constructed by the sampling procedure. We summarize the guarantee of our algorithm in the following theorem.

Theorem 26.4.1. *Given $A \in \mathbb{R}^{n \times d}$ and $b \in \mathbb{R}^n$, for $p \in [1, 2]$, let $\bar{A} = A \otimes \mathbf{1} - \mathbf{1} \otimes A \in \mathbb{R}^{n^2 \times d}$ and $\bar{b} = b \otimes \mathbf{1} - \mathbf{1} \otimes b \in \mathbb{R}^{n^2}$. Then there is an algorithm for that outputs $\widehat{x} \in \mathbb{R}^d$ such that with probability $1 - \delta$ we have $\|\bar{A}\widehat{x} - \bar{b}\|_p \leq (1 + \epsilon) \min_{x \in \mathbb{R}^d} \|\bar{A}x - \bar{b}\|_p$. The running time is $\widetilde{O}(\text{nnz}(A) + (d/(\epsilon\delta))^{O(1)})$.*

The theorem crucially utilizes our fast ℓ_p sampling routine, which is described in Figure 26.5 in the supplementary. A full discussion and proof of the lemma can be found in the supplementary material 26.9.1.

Algorithm 26.3 Our All-Pairs Regression Algorithm

procedure ALL-PAIRS REGRESSION(A, b)
  $F = [A, b] \in \mathbb{R}^{n \times d+1}$. $r \leftarrow \text{poly}(d/\epsilon)$
  Generate $S_1, S_2 \in \mathbb{R}^{k \times n}$ sparse p-stable transforms for $k = \text{poly}(d/(\epsilon\delta))$.
  Sketch $(S_1 \otimes S_2)(F \otimes \mathbf{1} - \mathbf{1} \otimes F)$.
  Compute QR decomposition: $(S_1 \otimes S_2)(F \otimes \mathbf{1} - \mathbf{1} \otimes F) = QR$.
  Let $M = (F \otimes \mathbf{1} - \mathbf{1} \otimes F)R^{-1}$, and $\sigma_i = \|M_{i,*}\|_p^p / \|M\|_p^p$.
  Obtain row sampling diagonal matrix $\Pi \in \mathbb{R}^{n \times n}$ such that $\Pi_{i,i} = 1/\widetilde{q}_i^{1/p}$ independently with probability $q_i \geq \min\{1, r\sigma_i\}$, where $\widetilde{q}_i = (1 \pm \epsilon^2)q_i$. ▷ Lemma 26.4.2
  return $\widehat{x}$, where $\widehat{x} = \arg\min_{x \in \mathbb{R}^d} \|\Pi(\bar{A}x - \bar{b})\|_p$.
end procedure

Lemma 26.4.2 (Fast ℓ_p sampling)**.** *Given $R \in \mathbb{R}^{d+1 \times d+1}$ and $F = [A, b] \in \mathbb{R}^{n \times d+1}$, there is an algorithm that, with probability $1 - n^{-c}$ for any constant c, produces a diagonal matrix $\Pi \in \mathbb{R}^{n^2 \times n^2}$ such that $\Pi_{i,i} = 1/\widetilde{q}_i^{1/p}$ with probability $q_i \geq \min\{1, r\|M_{i,*}\|_p^p / \|M\|_p^p\}$ and $\Pi_{i,i} = 0$ otherwise, where $r = \text{poly}(d/\epsilon)$ and $M = (F \otimes \mathbf{1} - \mathbf{1} \otimes F)R^{-1}$, and $\widetilde{q}_i = (1 \pm \epsilon^2)q_i$ for all $i \in [n^2]$. The total time required is $\widetilde{O}(\text{nnz}\, A + \text{poly}(d/\epsilon))$.*

26.5 Low Rank Approximation of Kronecker Product Matrices

We now consider low rank approximation of Kronecker product matrices. Given q matrices $A_1, A_2, \ldots, A_q$, where $A_i \in \mathbb{R}^{n_i \times d_i}$, the goal is to output a rank-k matrix $B \in \mathbb{R}^{n \times d}$, where $n = \prod_{i=1}^q n_i$ and $d = \prod_{i=1}^q d_i$, such that $\|B - A\|_F \leq (1 + \epsilon)\,\text{OPT}_k$, where $\text{OPT}_k = \min_{\texttt{rank}-k\ A'} \|A' - A\|_F$, and $A = \otimes_{i=1}^q A_i$. Our approach employs the Count-Sketch distribution of matrices [CW13, Woo14b]. A count-sketch matrix S is generated as follows. Each column of S contains exactly one non-zero entry. The non-zero entry is placed in a uniformly random row, and the value of the non-zero entry is either 1 or -1 chosen uniformly

at random.

Our algorithm is as follows. We sample q independent Count-Sketch matrices $S_1, \dots S_q$, with $S_i \in \mathbb{R}^{k_i \times n_i}$, where $k_1 = \cdots = k_q = \Theta(qk^2/\epsilon^2)$. We then compute $M = (\otimes_{i=1}^{q} S_i)A$, and let $U \in \mathbb{R}^{k \times d}$ be the top k right singular vectors of M. Finally, we output $B = AU^\top U$ in factored form (as $q+1$ separate matrices, $A_1, A_2, \dots, A_q, U$), as the desired rank-k approximation to A. The following theorem demosntrates the correctness of this algorithm. A full proof can be found in Section 26.10.

Theorem 26.5.1. *For any constant $q \geq 2$, there is an algorithm which runs in time $O(\sum_{i=1}^{q} \operatorname{nnz}(A_i) + d\operatorname{poly}(k/\epsilon))$ and outputs a rank k-matrix B in factored form such that $\|B - A\|_F \leq (1+\epsilon)\operatorname{OPT}_k$ with probability 9/10.* [4]

26.6 Numerical Simulations

In our numerical simulations, we compare our algorithms to two baselines: (1) brute force, i.e., directly solving regression without sketching, and (2) the methods based sketching developed in [DSSW18]. All methods were implemented in Matlab on a Linux machine. We remark that in our implementation, we simplified some of the steps of our theoretical algorithm, such as the residual sampling algorithm (Alg. 26.4). We found that in practice, even with these simplifications, our algorithms already demonstrated substantial improvements over prior work.

Following the experimental setup in [DSSW18], we generate matrices $A_1 \in \mathbb{R}^{300 \times 15}$,

[4] To amplify the probability, we can sketch A and $AU^\top U$ with a sparse JL matrix (e.g., Lemma 26.10.1 with $k_i = \Theta(qk^2/(\delta\epsilon^2))$ for each i) in input sparsity time to estimate the cost of a given solution. We can then repeat $\log(1/\delta)$ times and take the minimum to get failure probability $1 - \delta$.

$A_2 \in \mathbb{R}^{300\times 15}$, and $b \in \mathbb{R}^{300^2}$, such that all entries of A_1, A_2, b are sampled i.i.d. from a normal distribution. Note that $A_1 \otimes A_2 \in \mathbb{R}^{90000\times 225}$. We define T_{bf} to be the time of the brute force algorithm, T_{old} to be the time of the algorithms from [DSSW18], and T_{ours} to be the time of our algorithms. We are interested in the time ratio with respect to the brute force algorithm and the algorithms from [DSSW18], defined as, $r_t = T_{\text{ours}}/T_{\text{bf}}$, and $r'_t = T_{\text{ours}}/T_{\text{old}}$. The goal is to show that our methods are significantly faster than both baselines, i.e., both r_t and r'_t are significantly less than 1.

We are also interested in the quality of the solutions computed from our algorithms, compared to the brute force method and the method from [DSSW18]. Denote the solution from our method as x_{our}, the solution from the brute force method as x_{bf}, and the solution from the method in [DSSW18] as x_{old}. We define the relative residual percentage $r_e$and r'_e to be:

$$r_e = 100\frac{|\|\mathcal{A}x_{\text{ours}} - b\| - \|\mathcal{A}x_{\text{bf}} - b\||}{\|\mathcal{A}x_{\text{bf}} - b\|}, \quad r'_e = 100\frac{|\|\mathcal{A}x_{\text{old}} - b\| - \|\mathcal{A}x_{\text{bf}} - b\||}{\|\mathcal{A}x_{\text{bf}} - b\|}$$

Where $\mathcal{A} = A_1 \otimes A_2$. The goal is to show that r_e is close zero, i.e., our approximate solution is comparable to the optimal solution in terms of minimizing the error $\|\mathcal{A}x - b\|$.

Throughout the simulations, we use a moderate input matrix size so that we can accommodate the brute force algorithm and to compare to the exact solution. We consider varying values of m, where M denotes the size of the sketch (number of rows) used in either the algorithms of [DSSW18] or the algorithms in this paper. We also include a column m/n in the table, which is the ratio between the size of the sketch and the original matrix $A_1 \otimes A_2$. Note in this case that $n = 90000$.

Table 26.1: Results for ℓ_2 and ℓ_1-regression with respect to different sketch sizes m.

	m	m/n	r_e	r'_e	r_t	r'_t
ℓ_2	8100	.09	2.48%	1.51%	0.05	0.22
	12100	.13	1.55%	0.98%	0.06	0.24
	16129	.18	1.20%	0.71%	0.07	0.08
ℓ_1	2000	.02	7.72%	9.10%	0.02	0.59
	4000	.04	4.26%	4.00%	0.03	0.75
	8000	.09	1.85%	1.6%	0.07	0.83
	12000	.13	1.29%	0.99%	0.09	0.79
	16000	.18	1.01%	0.70%	0.14	0.90

Simulation Results for ℓ_2 We first compare our algorithm, Alg. 26.1, to baselines under the ℓ_2 norm. In our implementation, $\min_x \|Ax - b\|_2$ is solved by Matlab backslash $A\backslash b$. Table 26.1 summarizes the comparison between our approach and the two baselines. The numbers are averaged over 5 random trials. First of all, we notice that our method in general provides slightly less accurate solutions than the method in [DSSW18], i.e., $r_e > r'_e$ in this case. However, comparing to the brute force algorithm, our method still generates relatively accurate solutions, especially when m is large, e.g., the relative residual percentage w.r.t. the optimal solution is around 1% when $m \approx 16000$. On the other hand, as suggested by our theoretical improvements for ℓ_2, our method is significantly faster than the method from [DSSW18], consistently across all sketch sizes m. Note that when $m \approx 16000$, our method is around 10 times faster than the method in [DSSW18]. For small m, our approach is around 5 times faster than the method in [DSSW18].

Simulation Results for ℓ_1 We compare our algorithm, Alg. 26.2, to two baselines under the ℓ_1-norm. The first is a brute-force solution, and the second is the algorithm for [DSSW18].

For $\min_x \|Ax - b\|_1$, the brute for solution is obtained via a Linear Programming solver in Gurobi [GO16]. Table 26.1 summarizes the comparison of our approach to the two baselines under the ℓ_1-norm. The statistics are averaged over 5 random trials. Compared to the Brute Force algorithm, our method is consistently around 10 times faster, while in general we have relative residual percentage around 1%. Compared to the method from [DSSW18], our approach is consistently faster (around 1.3 times faster). Note our method has slightly higher accuracy than the one from [DSSW18] when the sketch size is small, but slightly worse accuracy when the sketch size increases.

26.7 Missing Proofs from Section 26.3

In this section, we prove correctness of our ℓ_2 Kronecker product regression algorithm. Specifically, we prove Theorem 26.3.1. To prove correctness, we need to establish several facts about the leverage scores of a Kronecker product.

Proposition 26.7.1. *Let $U_i \in \mathbb{R}^{n_i \times d_i}$ be an orthonormal basis for $A_i \in \mathbb{R}^{n_i \times d_i}$. Then $U = \otimes_{i=1}^q U_i$ is an orthonormal basis for $A = \otimes_{i=1}^q A_i$.*

Proof. Note that the column norm of each column of U is the product of column norms of the U_i's, which are all 1. Thus U has unit norm columns. It suffices then to show that all the singular values of U are 1 or -1, but this follows from the fact that the singular values of U are the product of singular values of the U_i's, which completes the proof. □

Corollary 26.7.2. *Let $A = \otimes_{i=1}^q A_i$, where $A_i \in \mathbb{R}^{n_i \times d_i}$. Fix any $\vec{i} = (i_1, \ldots, i_q) \in [n_1] \times [n_2] \times \cdots \times [n_q]$, and let $\vec{i}$ index into a row of A in the natural way. Then the $\vec{i}$-th leverage score of A is equal to $\prod_{j=1}^q \sigma_{i_j}(A_j)$, where $\sigma_t(B)$ is the t-th leverage score of a matrix B.*

Proof. Note $U = \otimes_{i=1}^{q} U_i$ is an orthonormal basis for $A = \otimes_{i=1}^{q} A_i$ by the prior Proposition. Now if $U_{\vec{i},*}$ is the $\vec{i}$-th row of U, then by fundamental properties of Kronecker products [VL00], we have $\|U_{\vec{i},*}\|_2 = \prod_{j=1}^{q} \|(U_j)_{i_j,*}\|_2$, which completes the proof. Note here that we used the fact that leverage scores are independent of the choice of orthonormal basis [Woo14b]. □

Proposition 26.7.3 (Theorem 29 of [CW13])**.** *Given a matrix $A \in \mathbb{R}^{n \times d}$, let $\sigma \in \mathbb{R}^n$ be the ℓ_2 leverage scores of A (see definition 26.2.3). Then there is an algorithm which computes values $\widetilde{\sigma}_1, \widetilde{\sigma}_2, \ldots, \widetilde{\sigma}_n$ such that $\widetilde{\sigma}_i = (1 \pm \epsilon)\sigma_i$ simultaneously for all $i \in [n]$ with probability $1 - 1/n^c$ for any constant $c \geq 1$. The runtime is $\widetilde{O}(\mathrm{nnz}(A) + \mathrm{poly}(d/\epsilon))$.*

Proposition 26.7.4. *Given $A = \otimes_{i=1}^{q} A_i$, where $A_i \in \mathbb{R}^{n_i \times d_i}$, there is an algorithm which, with probability $1 - 1/n^c$ for any constant $c \geq 1$, outputs a diagonal matrix $D \in \mathbb{R}^{n \times n}$ with m non-zeros entries, such that $D_{i,i} = 1/(m\widetilde{\sigma}_i)$ is non-zero with probability $\widetilde{\sigma}_i \in (1 \pm 1/10)\sigma_i(A)$. The time required is $\widetilde{O}(\sum_{i=1}^{q} \mathrm{nnz}(A_i) + \mathrm{poly}(dq/\epsilon) + mq)$.*

Proof. By Proposition 26.7.3, we can compute approximate leverage scores of each A_i up to error $\Theta(1/q)$ in time $\widetilde{O}(\mathrm{nnz}(A_i) + \mathrm{poly}(d/\epsilon))$ with high probability. To sample a leverage score from A, it suffices to sample one leverage score from each of the A_i's by Corollary 26.7.2. The probability that a given row $\vec{i} = (i_1, \ldots, i_q) \in [n_1] \times [n_2] \times \cdots \times [n_q]$ of A is chosen is $\prod_{j=1}^{q} \widetilde{\sigma}(A_j)_{i_j} = (1 \pm \Theta(1/q))^q \sigma_{\vec{i}}(A) = (1 \pm 1/10)\sigma_{\vec{i}}(A)$ as needed. Obtaining a sample takes $\widetilde{O}(1)$ time per A_i (since a random number needs to be generated to $O(\log(n))$-bits of precision in expectation and with high probability to obtain this sample), thus $O(q)$ time overall, so repeating the sampling M times gives the desired additive mq runtime. □

The $q = 1$ version of the following result can be found in [CW13, SWZ19b].

Proposition 26.7.5. *Let $D \in \mathbb{R}^{n \times n}$ be the diagonal row sampling matrix generated via Proposition 26.7.4, with $m = \Theta(1/(\delta\epsilon^2))$ non-zero entries. Let $A = \otimes_{i=1}^{q} A_i$ as above, and let $U \in \mathbb{R}^{n \times r}$ be an orthonormal basis for the column span of A, where $r = \mathrm{rank}(A)$. Then for any matrix B with n rows, we have*

$$\Pr\left[\|U^\top D^\top DB - U^\top B\|_F \leq \epsilon \|U\|_F \|B\|_F\right] \geq 1 - \delta$$

Proof. By definition of leverage scores and Proposition 26.7.4, D is a matrix which sample each row $U_{i,*}$ of U with probability at least $(9/10)\|U_{i,*}\|_2/\|U\|_F$. Taking the average of m such rows, we obtain the approximate matrix product result with error $O(1/\sqrt{\delta m})$ with probability $1 - \delta$ by Theorem 2.1 of [KV17]. □

We our now ready to prove the main theorem of this section, Theorem 26.3.1

Theorem 26.3.1(Kronecker product ℓ_2 regression) *Let $D \in \mathbb{R}^{n \times n}$ be the diagonal row sampling matrix generated via Proposition 26.7.4, with $m = \Theta(1/(\delta\epsilon^2))$ non-zero entries, and let $A = \otimes_{i=1}^{q} A_i$, where $A_i \in \mathbb{R}^{n_i \times d_i}$, and $b \in \mathbb{R}^n$, where $n = \prod_{i=1}^{q} n_i$ and $d = \prod_{i=1}^{q} d_i$. Then we have let $\widehat{x} = \arg\min_{x \in \mathbb{R}^d} \|DAx - Db\|_2$, and let $x^* = \arg\min_{x' \in \mathbb{R}^d} \|Ax - b\|_2$. Then with probability $1 - \delta$, we have*

$$\|A\widehat{x} - b\|_2 \leq (1 + \epsilon)\|Ax^* - b\|_2$$

Moreover, the total runtime requires to compute $\widehat{x}$ is

$$\widetilde{O}\left(\sum_{i=1}^{q} \mathrm{nnz}(A_i) + (dq/(\delta\epsilon))^{O(1)}\right).$$

Proof. Let U be an orthonormal basis for the column span of A. By Lemma 3.3 of [CW09], we have $\|A(\widehat{x}-x^*)\|_2 \leq 2\sqrt{\epsilon}\|Ax^*-b\|_2$. Note that while Lemma 3.3 of [CW09] uses a different sketching matrix D than us, the only property required for the proof of Lemma 3.3 is that $|U^\top D^\top DB - U^\top B\|_F \leq \sqrt{\epsilon/d}\|A\|_F\|B\|_F$ with probability at least $1-\delta$ for any fixed matrix B, which we obtain by Proposition 26.7.5 by having $O(d/(\delta\epsilon^2))$ non-zeros on the diagonal of D). By the normal equations, we have $A^\top(Ax^*-b)=0$, thus $\langle A(\widehat{x}-x^*),(Ax^*-b)\rangle = 0$, and so by the Pythagorean theorem we have

$$\|A\widehat{x}-b\|_2^2 = \|Ax^*-b\|_2^2 + \|A(\widehat{x}-x^*)\|_2^2 \leq (1+4\epsilon)\|Ax^*-b\|_2^2$$

Which completes the proof after rescaling of ϵ. The runtime required to obtain the matrix D is $\widetilde{O}(\sum_{i=1}^q \text{nnz}(A_i) + \text{poly}(dq/\epsilon))$ by Proposition 26.7.4, where we set D to have $m = \Theta(d/(\delta\epsilon^2))$ non-zero entries on the diagonal. Once D is obtained, one can compute $D(A+b)$ in time $O(md)$, thus the required time is $O(\delta^{-1}(d/\epsilon)^2)$. Finally, computing $\widehat{x}$ once DA, Db are computed requires a single pseudo-inverse computation, which can be carried out in $O(\delta^{-1}d^3/\epsilon^2)$ time (since DA now has only $O(\delta^{-1}(d/\epsilon)^2)$ rows).

□

26.8 Missing Proofs from Section 26.3.1

We now give a complete proof of Theorem 26.3.2. Our high level approach follows that of [DDH+09]. Namely, we first obtain a vector x' which is a $O(1)$ approximate solution to the optimal, and then use the *residual error* $\rho \in \mathbb{R}^d$ of x' to refine x' to a $(1\pm\epsilon)$ approximation $\widehat{x}$. The fact that x' is a constant factor approximation follows from our Lemma 26.8.4. Given x', by Lemma 26.8.7 we can efficiently compute the matrix Σ which samples from

the coordinates of the residual error $\rho = (A_1 \otimes \cdots \otimes A_q)x' - b$ in the desired runtime. The sampling lemma is the main technical lemma, and requires a careful multi-part sketching and sampling routine. Given this Σ, the fact that $\widehat{x}$ is a $(1+\epsilon)$ approximate solution follows directly from Theorem 6 of [DDH+09]. Our main theorem and its proof is stated below. The proof will utilize the lemmas and sampling algorithm developed in the secitons which follow.

Theorem 26.3.2 (Main result, ℓ_p, $1+\epsilon$-approximation)**.** *Fix $1 \leq p < 2$. Then for any constant $q = O(1)$, given matrices $A_1, A_2, \cdots, A_q$, where $A_i \in \mathbb{R}^{n_i \times d_i}$, let $n = \prod_{i=1}^{q} n_i$, $d = \prod_{i=1}^{q} d_i$. Let $\widehat{x} \in \mathbb{R}^d$ be the output of Algorithm 26.2. Then*

$$\|(A_1 \otimes A_2 \otimes \cdots \otimes A_q)\widehat{x} - b\|_p \leq (1+\epsilon) \min_{x \in \mathbb{R}^n} \|(A_1 \otimes A_2 \otimes \cdots \otimes A_q)x - b\|_p$$

holds with probability at least $1 - \delta$. In addition, our algorithm takes

$$\widetilde{O}\left(\left(\sum_{i=1}^{q} \operatorname{nnz}(A_i) + \operatorname{nnz}(b) + (d/\epsilon)^{O(1)}\right) \log(1/\delta)\right)$$

time to output $\widehat{x} \in \mathbb{R}^d$.

Proof. By Lemma 26.8.4, the output x' in line 16 of algorithm 26.3.1 is an 8 approximation of the optimal solution, and x' is obtained in time $\widetilde{O}(\sum_{i=1}^{q} \operatorname{nnz}(A_i) + (dq/\epsilon)^{O(1)})$. We then obtain the residual error $\rho = (A_1 \otimes \cdots \otimes A_q)x' - b$ (implicitly). By Theorem 6 of [DDH+09], if we let $\Sigma \in \mathbb{R}^{n \times n}$ be a row sampling matrix where $\Sigma_{i,i} = 1/\alpha_i^{1/p}$ with probability $\alpha_i = \min\{1, \max\{q_i, r_2 \frac{|\rho_i|^p}{\|\rho\|_p^p}\}$, where q_i is the row sampling probability used in the sketch Π from which x' was obtained, and$r_2 = O(d^3/\epsilon^2 \log(1/\epsilon))$, then the solution to $\min_x \|\Sigma(A_1 \otimes \cdots \otimes A_q)x - \Sigma b\|_p$ will be a $(1+\epsilon)$ approximately optimal solution. By Lemma 26.8.7, we can obtain such a matrix Σ in time $\widetilde{O}(\sum_{i=1}^{q} \operatorname{nnz}(A_i) + q \operatorname{nnz}(b) + (d \log(n)/(\epsilon\delta)^{O(q^2)})$, which completes the proof of correctness. Finally, note that we can solve the sketched regression problem

$\min_x \|\Sigma(A_1 \otimes \cdots \otimes A_q)x - \Sigma b\|_p$ which has $O((d\log(n)/\epsilon)^{O(q^2)}(1/\delta))$ constraints and d variables in time $O((d\log(n)/\epsilon)^{O(q^2)}(1/\delta))$ using linear programming for $p = 1$ (see [CLS19] for the state of the art linear program solver), or more generally interior point methods for convex programming for $p > 1$ (see [BCLL18] for the state of the art ℓ_p solver).

Now to boost the failure probability from a $O(1/\delta)$ to $\log(1/\delta)$ dependency, we do the following. We run the above algorithm with $\delta = 1/10$, so that our output $\widehat{x} \in \mathbb{R}^d$ is a $(1+\epsilon)$ approximation with probability $9/10$. Now note that we actually have an $(1+\epsilon)$ estimate of the cost $\|(A_1 \otimes A_2 \otimes \cdots \otimes A_q)\widehat{x} - b\|_p$ of the solution $\widehat{x}$, which is simply given by $\|\Sigma(A_1 \otimes A_2 \otimes \cdots \otimes A_q)\widehat{x} - \Sigma b\|_p$ where Σ is the sampling matrix used to compute $\widehat{x}$. Thus we can simply repeat the above process $O(\log(1/\delta))$ times, and take the solution with the minimal cost overall. □

We start by defining a tensor operation which will be useful for our analysis.

Definition 26.8.1 ($((\cdot,\ldots,\cdot),\cdot)$ operator for tensors and matrices)**.** Given tensor $A \in \mathbb{R}^{d_1 \times d_2 \times \cdots \times d_q}$ and matrices $B_i \in \mathbb{R}^{n_i \times d_i}$ for $i \in [q]$, we define the tensor $((B_1, B_2, \ldots, B_q), A) \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_q}$:

$$((B_1, B_2, \ldots, B_q), A)_{i_1,\ldots,i_q} = \sum_{i'_1=1}^{d_1} \sum_{i'_2=1}^{d_2} \cdots \sum_{i'_q=1}^{d_q} A_{i'_1,i'_2,\ldots,i'_q} \prod_{\ell=1}^{q} (B_\ell)_{i_\ell,i'_\ell}$$

Observe for the case of $q = 2$, we just have $((B_1, B_2), A) = B_1 A B_2^\top \in \mathbb{R}^{n_1 \times n_2}$.

Using the above notation, we first prove a result about reshaping tensors.

Lemma 26.8.1 (Reshaping)**.** *Given matrices $A_1, A_2, \cdots, A_q \in \mathbb{R}^{n_i \times d_i}$ and a tensor $B \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_q}$, let $n = \prod_{i=1}^q n_i$ and let $d = \prod_{i=1}^d d_i$. Let b denote the vectorization of B.*

For any tensor $X \in \mathbb{R}^{d_1 \times d_2 \times \cdots \times d_q}$, we have $\|((A_1, A_2, \cdots, A_q), X) - B\|_\xi$ is equal to $\|(A_1 \otimes A_2 \otimes \cdots \otimes A_q)x - b\|_\xi$ where ξ is any entry-wise norm (such as an ℓ_p-norm) and x is the vectorization of X. See Definition 26.8.1 of the $((\cdot, \ldots, \cdot), \cdot)$ tensor operator.

Observe, for the case of $q = 2$, this is equivalent to the statement that $\|A_1 X A_2^\top - B\|_\xi = \|(A_1 \otimes A_2)x - b\|_\xi$.

Proof. For the pair $x \in \mathbb{R}^d$, $X \in \mathbb{R}^{d_1 \times d_2 \times \cdots \times d_q}$, the connection is the following: $\forall i_1 \in [d_1], \ldots, i_q \in [d_q]$,

$$x_{i_1 + \sum_{l=2}^{q} (i_l - 1) \cdot \prod_{t=1}^{l-1} d_t} = X_{i_1, \cdots, i_q}.$$

Similarly, for $b \in \mathbb{R}^n$, $B \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_q}$, for any $j_1, \in [n_1], \ldots, j_q \in [n_q]$,

$$b_{j_1 + \sum_{l=2}^{q} (j_l - 1) \cdot \prod_{t=1}^{l-1} n_t} = B_{j_1, j_2, \cdots, j_q}.$$

For simplicity, for any $(i_1, \ldots, i_q) \in [d_1] \times \cdots \times [d_q]$ and $(j_1, \ldots, j_q) \in [n_1] \times \cdots \times [n_q]$ we define $\vec{i} = i_1 + \sum_{l=2}^{q} (i_l - 1) \cdot \prod_{t=1}^{l-1} d_t$ and similarly $\vec{j} = j_1 + \sum_{l=2}^{q} (j_l - 1) \cdot \prod_{t=1}^{l-1} n_t$. Then we can simplify the above relation and write $x_{\vec{i}} = X_{i_1, i_2, \cdots, i_q}$, and $b_{\vec{j}} = B_{j_1, j_2, \cdots, j_q}$.

For a matrix Z, let $Z_{i,*}$ denote the i-th row of Z. We consider the $\vec{j}$-th entry of $(A_1 \otimes A_2 \otimes \cdots \otimes A_q)x$,

$$\begin{aligned}
((A_1 \otimes A_2 \otimes \cdots \otimes A_q)x)_{\vec{j}} &= \left\langle (A_1 \otimes A_2 \otimes \cdots \otimes A_q)_{\vec{j},*} \cdot x \right\rangle \\
&= \sum_{i_1=1}^{d_1} \sum_{i_2=1}^{d_2} \cdots \sum_{i_q=1}^{d_q} \left(\prod_{l=1}^{q} (A_l)_{j_l, i_l} \right) \cdot x_{\vec{i}} \\
&= \sum_{i_1=1}^{d_1} \sum_{i_2=1}^{d_2} \cdots \sum_{i_q=1}^{d_q} \left(\prod_{l=1}^{q} (A_l)_{j_l, i_l} \right) \cdot X_{i_1, i_2, \cdots, i_q} \\
&= ((A_1, A_2, \cdots, A_q), X)_{j_1, \ldots, j_q}.
\end{aligned}$$

Where the last equality is by Definition (26.8.1). Since we also have $b_{\vec{j}} = B_{j_1,\ldots,j_q}$, this completes the proof of the Lemma. □

26.8.1 Sampling from an ℓ_p-Well-Conditioned Base

In this Section, we discuss the first half of Algorithm 26.2 which computes $x' \in \mathbb{R}^d$, which we will show is a $O(1)$-approximate solution to the optimal. First note that by Lemma 26.2.1 together with fact 26.2.2, we know that $A_i R_i^{-1}$ is an ℓ_p well conditioned basis for A_i (recall this means that $A_i R_i^{-1}$ is a (α, β, p) well conditioned basis for A, and $\beta/\alpha = d_i^{O(1)}$) with probability $1 - O(1/q)$, and we can then union bound over this occurring for all $i \in [q]$. Given this, we now prove that $(A_1 R_1^{-1} \otimes A_2 R_2^{-1} \otimes \cdots \otimes A_q R_q^{-1})$ is a well conditioned basis for $(A_1 \otimes A_2 \otimes \cdots \otimes A_q)$.

Lemma 26.8.2. *Let $A_i \in \mathbb{R}^{n_i \times d_i}$ and $R_i \in \mathbb{R}^{d_i \times d_i}$. Then if $A_i R_i^{-1}$ is a (α_i, β_i, p) well-conditioned basis for A_i for $i = 1, 2, \ldots, q$, we have for all $x \in \mathbb{R}^{d_1 \cdots d_q}$:*

$$\prod_{i=1}^{q} \alpha_i \|x\|_p \leq \|(A_1 R_1^{-1} \otimes A_2 R_2^{-1} \otimes \cdots \otimes A_q R_q^{-1})x\|_p \leq \prod_{i=1}^{q} \beta_i \|x\|_p$$

Proof. We first consider the case of $q = 2$. We would like to prove

$$\alpha_1 \alpha_2 \|x\|_p \leq \|(A_1 R_1^{-1} \otimes A_2 R_2^{-1})x\|_p \leq \beta_1 \beta_2 \|x\|_p,$$

First note, by the reshaping Lemma 26.8.1, this is equivalent to

$$\alpha_1 \alpha_2 \|X\|_p \leq \|A_1 R_1^{-1} X (R_2^{-1} A_2)^\top\|_p \leq \beta_1 \beta_2 \|X\|_p.$$

Where $X \in \mathbb{R}^{d_1 \times d_2}$ is the tensorization of x. We first prove one direction. Let $U_1 = A_1 R_1^{-1}$

and $U_2 = A_2 R_2^{-1}$. We have

$$\begin{aligned}\|U_1 X U_2^\top\|_p^p &= \sum_{i_2=1}^{n_2} \|U_1 (X U_2^\top)_{i_2}\|_p^p \\ &\leq \sum_{i_2=1}^{n_2} \beta_1^p \|(X U_2^\top)_{i_2}\|_p^p \\ &= \beta_1^p \|X U_2^\top\|_p^p \\ &\leq \beta_1^p \beta_2^p \|X\|_p^p,\end{aligned}$$

where the first step follows from rearranging, the second step follows from the well-conditioned property of U_1, the third step follows from rearranging again, the last step follows from the well-conditioned property of U_2. Similarly, we have

$$\begin{aligned}\|U_1 X U_2^\top\|_p^p &= \sum_{i_2=1}^{n_2} \|U_1 (X U_2^\top)_{i_2}\|_p^p \\ &\geq \sum_{i_2=1}^{n_2} \alpha_1^p \|(X U_2^\top)_{i_2}\|_p^p \\ &= \alpha_1^p \|X U_2^\top\|_p^p \\ &\geq \alpha_1^p \alpha_2^p \|X\|_p^p,\end{aligned}$$

where again the first step follows from rearranging, the second step follows from the well-conditioned property of U_1, the third step follows from rearranging again, the last step follows from the well-conditioned property of U_2.

In general, for arbitrary $q \geq 2$, similarly using our reshaping lemma, we have

$$\begin{aligned}\|(\otimes_{i=1}^q (A_i R_i^{-1}))x\|_p &\geq \prod_{i=1}^q \alpha_i \|x\|_p, \\ \|(\otimes_{i=1}^q (A_i R_i^{-1}))x\|_p &\leq \prod_{i=1}^q \beta_i \|x\|_p.\end{aligned}$$

□

Putting this together with fact 26.2.2, and noting $d = d_1 \cdots d_q$, we have

Corollary 26.8.3. *Let $A_i R_i^{-1}$ be as in algorithm 26.2. Then we have for all $x \in \mathbb{R}^{d_1 \cdots d_q}$:*

$$(1/d)^{O(1)} \|x\|_p \leq \|(A_1 R_1^{-1} \otimes \cdots \otimes A_q R_q^{-1})x\|_p \leq d^{O(1)} \|x\|_p,$$

In other words, $(A_1 R_1^{-1} \otimes \cdots \otimes A_q R_q^{-1})$ is a well conditioned ℓ_p basis for $(A_1 \otimes \cdots \otimes A_q)$

From this, we can obtain the following result.

Lemma 26.8.4. *Let $x' \in \mathbb{R}^d$ be the output of the $O(1)$-Approximate ℓ_p Regression Procedure in Algorithm 26.2. Then with probability $99/100$ we have*

$$\|(A_1 \otimes \cdots \otimes A_q)x' - b\|_p \leq 8 \min_x \|(A_1 \otimes \cdots \otimes A_q)x - b\|_p$$

Moreover, the time required to compute x' is $\widetilde{O}(\sum_{i=1}^q \operatorname{nnz}(A_i) + (dq/\epsilon)^{O(1)})$.

Proof. By Theorem 6 of [DDH+09], if we let Π be a diagonal row sampling matrix such that $\Pi_{i,i} = 1/q_i^{1/p}$ with probability $q_i \geq \min\{1, r_1 \frac{\|U_{i,*}\|_p^p}{\|U\|_p^p}\}$, where U is a ℓ_p well-conditioned basis for $(A_1 \otimes \cdots \otimes A_q)$ and $r_1 = O(d^3)$, then the solution x' to

$$\min_x \|\Pi((A_1 \otimes \cdots \otimes A_q)x - b\|$$

will be a 8-approximation. Note that we can solve the sketched regression problem $\min_x \|\Pi((A_1 \otimes \cdots \otimes A_q)x' - b\|$ which has $O(\text{poly}(d/\epsilon))$ constraints and d variables in time $\text{poly}(d/\epsilon)$ using linear programming for $p = 1$ (see [CLS19] for the state of the art linear program solver), or

more generally interior point methods for convex programming for $p > 1$ (see [BCLL18] for the state of the art ℓ_p solver).

Then by Corollary 26.8.3, we know that setting $U = (A_1 R_1^{-1} \otimes \cdots \otimes A_q R_q^{-1})$ suffices, so now we must sample rows of U. To do this, we must approximately compute the norms of the rows of U. Here, we use the fact that $\|\cdot\|_p^p$ norm of a row of $(A_1 R_1^{-1} \otimes \cdots \otimes A_q R_q^{-1})$ is the product of the row norms of the $A_i R_i^{-1}$ that correspond to that row. Thus it suffices to sample a row j_i from each of the $A_i R_i^{-1}$'s with probability at least $\min\{1, r_1 \|(A_i R_i^{-1})_{j_i,*}\|_p^p / \|A_i R_i^{-1}\|_p^p\}$ for each $i \in [q]$.

To do this, we must estimate all the row norms $\|(A_i R_i^{-1})_{j_i,*}\|_p^p$ to $(1 \pm 1/10)$ error. This is done in steps $7 - 10$ of Algorithm 26.2, which uses dense p-stable sketches $Z \in \mathbb{R}^{d \times \tau}$, and computes $(A_i R_i^{-1} Z)$, where $\tau = \Theta(\log(n))$. Note that computing $R_i^{-1} Z \in \mathbb{R}^{d \times \tau}$ requires $\widetilde{O}(d^2)$. Once computed, $A_i(R_i^{-1} Z)$ can be computed in $\widetilde{O}(\text{nnz}(A_i))$ time. We then take the median of the coordinates of $(A_i R_i^{-1} Z)$ (normalized by the median of the p-stable distribution $\mathcal{D}_p$, which can be efficiently approximated to $(1 \pm \epsilon)$ in $O(\text{poly}(1/\epsilon))$ time, see Appendix A.2 of [KNW10b] for details) as our estimates for the row norms. This is simply the Indyk median estimator [Ind06], and gives a $(1 \pm 1/10)$ estimate $a_{i,j}$ of all the row norms $\|(A_i R_i^{-1})_{j,*}\|_p^p$ with probability $1 - 1/\text{poly}(n)$. Then it follows by Theorem 6 of [DDH+09] that x' is a 8-approximation of the optimal solution with probability 99/100 (note that we amplified the probability by increasing the sketch sizes S_i by a constant factor), which completes the proof.

□

26.8.2 ℓ_p Sampling From the residual of a $O(1)$-factor approximation

By Lemma 26.8.4 in the prior section, we know that the x' first returned by the in algorithm 26.2 is a 8-approximation. We now demonstrate how we can use this $O(1)$ approximation to obtain a $(1+\epsilon)$ approximation. The approach is again to sample rows of $(A_1 \otimes \cdots \otimes A_q)$. But instead of sampling rows with the well-conditioned leverage scores q_i, we now sample the i-th row with probability $\alpha_i = \min\{1, \max\{q_i, r_2|\rho_i|^p/\|\rho\|_p^p\}\}$, where $\rho = (A_1 \otimes \cdots \otimes A_q)x' - b \in \mathbb{R}^n$ is the *residual error* of the $O(1)$-approximation x'. Thus we must now determine how to sample quickly from the residuals $|\rho_i|^p/\|\rho\|_p^p$. Our sampling algorithm will need a tool originally developed in the streaming literature.

Count-sketch for heavy hitters with the Dyadic Trick. We now introduce a sketch S which finds the ℓ_2 heavy hitters in a vector x efficently. This sketch S is known as count-sketch for heavy hitters with the Dyadic Trick. To build S we first stack $\Theta(\log(n))$ copies of the *count sketch matrix* $S^i \in \mathbb{R}^{k' \times n}$ [CW13]. The matrix S^i is constructed as follows. S^i has exactly one non-zero entry per column, which is placed in a uniformly random row, and given the value 1 or -1 uniformly at random. For S^i, let $h_i : [n] \to [k']$ be such that $h_i(t)$ is the row with the non-zero entry in the t-th column of S^i, and let $g_i : [n] \to \{1, -1\}$ be such that the value of that non-zero entry is $g_i(t)$. Note that the h_i, g_i can be implemented as 4-wise independent hash functions. Fix any $x \in \mathbb{R}^n$. Then given $S^1x, S^2x, \cdots, S^{\Theta(\log(n))}x$, we can estimate the value of any coordinate x_j by $\text{median}_{i \in \Theta\log(n)}\{g_i(j)(S^ix)_{h_i(j)}\}$.

It is well-known that this gives an estimate of x_j with additive error $\Theta(1/\sqrt{k'})\|x\|_2$ with probability $1 - 1/\operatorname{poly}(n)$ for all $j \in [n]$ [CCFC04]. However, naively, to find the heaviest coordinates in x, that is all coordinates x_j with $|x_j| \geq \Theta(1/\sqrt{k'})\|x\|_2$, one would

need to query $O(n)$ estimates. This is where the Dyadic trick comes in [CM04]. We repeat the above process $\Theta(\log(n))$ times, with matrices $S^{(i,j)}$, for $i, j \in \Theta(\log(n))$. Importantly, however, in $S^{(i,j)}$, for all $t, t' \in [n]$ such that the first j most significant bits in their binary identity representation are the same, we set $h_{(i,j)}(t) = h_{(i,j)}(t')$, effectively collapsing these identities to one. To find a heavy item, we can then query the values of the *two* identities from $S^{(1,1)}, S^{(2,1)}, \cdots, S^{(\Theta(\log(n)),1)}$, and recurse into all the portions which have size at least $\Theta(1/\sqrt{k'})\|x\|_2$. It is easy to see that we recurse into at most $O(k')$ such pieces in each of the $\Theta(\log(n))$ levels, and it takes $O(\log(n))$ time to query a single estimate, from which the desired runtime of $O(k' \log^2(n))$ is obtained. For a further improvement on size k of the overall sketched required to quickly compute Q, see [LNNT16]. We summarize this construction below in definition 26.8.2.

Definition 26.8.2 (Count-sketch for heavy hitters with Dyadic Trick [CCFC04, LNNT16])**.** There is a randomized sketch $S \in \mathbb{R}^{k \times n}$ with $k = O(\log^2(n)/\epsilon^2)$ such that, for a fixed vector $x \in \mathbb{R}^n$, given $Sx \in \mathbb{R}^k$, one can compute a set $Q \subset [n]$ with $|Q| = O(1/\epsilon^2)$ such that $\{i \in [n] \mid |x_i| \geq \epsilon \|x\|_2\} \subseteq Q$ with probability $1 - 1/\operatorname{poly}(n)$. Moreover, Sx can be computed in $O(\log^2(n) \operatorname{nnz}(x))$ time. Given Sx, the set Q can be computed in time $O(k)$.

We begin with some notation. For a vector $y \in \mathbb{R}^n$, where $n = n_1 \cdots n_q$, one can index any entry of y_i via $\vec{i} = (i_1, i_2, \cdots, i_q) \in [n_1] \times \cdots \times [n_q]$ via $i = i_1 + \sum_{j=2}^{q} (i_j - 1) \prod_{l=1}^{i_j - 1} n_l$. It will useful to index into such a vector y interchangably via a vector $y_{\vec{i}}$ and an index y_j with $j \in [n]$. For any set of subsets $T_i \subset [n_i]$, we can define $y_{T_1 \times \cdots T_q} \in \mathbb{R}^n$ as y restricted to the $\vec{i} \in T_1 \times \cdots \times T_q$. Here, by restricted, we mean the coordinates in y that are not in this set are set equal to 0. Similarly, for a $y \in \mathbb{R}^{n_i}$ and $S \subset [n_i]$, we can define y_S as y restricted

to the coordinates in S. Note that in Algorithm 26.4, $\mathbb{I}_n$ denotes the $n \times n$ identity matrix for any integer n. We first prove a proposition on the behavior of Kronecker products of p-stable vectors, which we will need in our analysis.

Proposition 26.8.5. *Let $Z_1, Z_2, \cdots, Z_q$ be independent vectors with entries drawn i.i.d. from the p-stable distribution, with $Z_i \in \mathbb{R}^{n_i}$. Now fix any $i \in [q]$, and any $x \in \mathbb{R}^n$, where $n = n_1 n_2 \cdots n_q$. Let $e_j \in \mathbb{R}^{n_i}$ be the j-th standard basis column vector for any $j \in [n_i]$. Let $\Gamma(i,j) = [n_1] \times [n_2] \times \cdots \times [n_{i-1}] \times \{j\} \times [n_{i+1}] \times \cdots \times [n_q]$. Define the random variable*

$$\mathcal{X}_{i,j}(x) = |(Z_1 \otimes Z_1 \otimes \cdots \otimes Z_{i-1} \otimes e_j^\top \otimes Z_{i+1} \otimes \cdots \otimes Z_q)x|^p.$$

Then for any $\lambda > 1$, with probability at least $1 - O(q/\lambda)$ we have

$$\|x_{\Gamma(i,j)}\|_p^p / \lambda^q \leq \mathcal{X}_{i,j}(x) \leq (\lambda \log(n))^q \|x_{\Gamma(i,j)}\|_p^p$$

Proof. First observe that we can reshape $y = x_\Gamma \in \mathbb{R}^m$ where $m = n/n_i$, and re-write this random variable as $\mathcal{X}_{i,j}(x) = |(Z_1 \otimes Z_2 \otimes \cdots \otimes Z_{q-1})y|^p$. By reshaping Lemma 26.8.1, we can write this as $|(Z_1 \otimes Z_2 \otimes \cdots \otimes Z_{q-2})Y Z_{q-1}^\top|^p$, where $Y \in \mathbb{R}^{m/n_{q-1} \times n_{q-1}}$. We first prove a claim. In the following, for a matrix A, let $\|A\|_p^p = \sum_{i,j} |A_{i,j}|^p$.

Claim 26.8.6. *Let Z be any p-stable vector and X a matrix. Then for any $\lambda > 1$, with probability $1 - O(1/\lambda)$, we have*

$$\lambda^{-1} \|X\|_p^p \leq \|XZ\|_p^p \leq \log(n)\lambda \|X\|_p^p.$$

Proof. By p-stability, each entry of $|(XZ)_i|^p$ is distributed as $|z_i|^p \|X_{i,*}\|_p^p$, where z_i is again p-stable (but the $z_i's$ are not independent). Now p-stables have tails that decay at the rate $\Theta(1/x^p)$ (see Chapter 1.5 of [Nol07]), thus $\Pr[|z_i|^p > x] = O(1/x)$ for any $x > 0$. We

can condition on the fact that $z_i < \lambda \cdot n^{10}$ for all i, which occurs with probability at least $1 - n^{-9}/\lambda$ by a union bound. Conditioned on this, we have $\mathbb{E}[|z_i|^p] = O(\log(n))$ (this can be seen by integrating over the truncated tail $O(1/x)$), and the upper bound then follows from a application of Markov's inequality.

For the lower bound Let Y_i be an indicator random variable indicating the event that $|z_i|^p < 2/\lambda$. Now p-stables are anti-concentrated, namely, their pdf is upper bounded by a constant everywhere. It follows that $\Pr[Y_i] < c/\lambda$ for some constant c. By Markov's inequality $\Pr[\sum_i Y_i \|X_{i,*}\|_p^p > \|X\|_p^p/2] < O(1/\lambda)$. Conditioned on this, the remaining $\|X\|_p^p/2$ of the ℓ_p mass shrinks by less than a $2/\lambda$ factor, thus $\|XZ\|_p^p > (\|X\|_p^p/2)(2/\lambda) = \|X\|_p^p/\lambda$ as needed. □

By the above claim, we have $\|Y\|_p/\lambda^{1/p} \leq \|YZ_{q-1}^\top\|_p \leq (\log(n)\lambda)^{1/p}\|Y\|_p$ with probability $1 - O(1/\lambda)$. Given this, we have $\mathcal{X}_{i,j}(x) = |(Z_1 \otimes Z_2 \otimes \cdots \otimes Z_{q-2})y'|^p$, where $\|Y\|_p/\lambda^{1/p} \leq \|y'\|_p \leq (\log(n)\lambda)^{1/p}\|Y\|_p$. We can inductively apply the above argument, each time getting a blow up of $(\log(n)\lambda)^{1/p}$ in the upper bound and $(1/\lambda)^p$ in the lower bound, and a failure probability of $(1/\lambda)$. Union bounding over all q steps of the induction, the proposition follows.

□

Lemma 26.8.7. *Fix any $r_2 \geq 1$, and suppose that $x' = \min_x \|\Pi(A_1 \otimes \cdots \otimes A_q)x - \Pi b\|_p$ and $\Pi \in \mathbb{R}^{n \times n}$ is a row sampling matrix such that $\Pi_{i,i} = 1/q_i^{1/p}$ with probability q_i. Define the residual error $\rho = (A_1 \otimes \cdots \otimes A_q)x' - b \in \mathbb{R}^n$. Then Algorithm 26.4, with probability $1 - \delta$, succeeds in outputting a row sampling matrix $\Sigma \in \mathbb{R}^{n \times n}$ such that $\Sigma_{i,i} = 1/\alpha_i^{1/p}$ with probability $\alpha_i = \min\{1, \max\{q_i, r_3|\rho_i|^p/\|\rho\|_p^p\}\}$ for some $r_3 \geq r_2$, and otherwise $\Sigma_{i,i} = 0$. The*

algorithm runs in time

$$\widetilde{O}\left(\sum_{i=1}^{q} \operatorname{nnz}(A_i) + q \operatorname{nnz}(b) + (r_2 \log(n)/\delta)^{O(q^2)}\right).$$

Proof. The algorithm is given formally in figure 26.4. We analyze the runtime and correctness here.

Proof of Correctness. The approach of the sampling algorithm is as follows. Recall that we can index into the coordinates of $\rho \in \mathbb{R}^n$ via $\vec{a} = (a_1, \ldots, a_q)$ where $a_i \in [n_i]$. We build the coordinates of $\vec{a}$ one by one. To sample a $\vec{a} \in \prod_{i=1}^{q} [n_i]$, we can first sample $a_1 \in [n_1]$ from the distribution $\Pr[a_1 = j] = \sum_{\vec{u}:u_1=j} |\rho_{\vec{u}}|^p / (\sum_{\vec{u}} |\rho_{\vec{u}}|^p)$. Once we fix a_1, we can sample a_2 from the conditional distribution distribution $\Pr[a_2 = j] = \sum_{\vec{u}:u_2=j,u_1=a_1} |\rho_{\vec{u}}|^p / (\sum_{\vec{u}:u_1=a_1} |\rho_{\vec{u}}|^p)$, and so on. For notation, given a vector $\vec{a} = (a_1, \ldots, a_{i-1})$, let $\Delta(\vec{a}) = \{\vec{u} \in [n_1] \times \cdots \times [n_q] \mid a_j = y_j \text{ for all } j = 1, 2, \ldots, i-1\}$. Then in general, when we have sampled $\vec{a} = (a_1, \ldots, a_{i-1})$ for some $i \leq q$, we need to sample $a_i \leftarrow j \in [n_k]$ with probability

$$\Pr[a_i = j] = \sum_{\vec{u} \in \Delta(\vec{a}):u_i=j} |\rho_{\vec{u}}|^p \Big/ \left(\sum_{\vec{u} \in \Delta(\vec{a})} |\rho_{\vec{u}}|^p\right).$$

We repeat this process to obtain the desired samples. Note that to sample efficiently, we will have to compute these aforementioned sampling probabilities approximately. Because of the error in approximating, instead of returning r_2 samples, we over-sample and return $r_3 = \Theta(r_2 \log^{q^2}(n))$ samples.

The first step is of the algorithm is to generate the p-stable vectors $Z^{i,j} \in \mathbb{R}^{n_i}$ for $i \in [q]$ and $j = 1, 2, \ldots, \Theta(\log(n))$. We can pre-compute and store $Z^{i,j} A_i$ for $i \in [q]$, which takes $\widetilde{O}(\sum_{i=1}^{q} \operatorname{nnz}(A_i))$ time. We set $w^j \leftarrow \left((\mathbb{I}_{n_1}) \otimes (\bigotimes_{k=2}^{q} Z^{k,j})\rho\right) \in \mathbb{R}^{n_1}$ and define $w \in \mathbb{R}^{n_1}$

by $w_l = \text{median}_{j\in[\tau]}\{|w_l^j|\}$ for $l \in [n_1]$. Observe that w_l^j is an estimate of $\sum_{\vec{u}:u_1=l} |\rho_{\vec{u}}|^p$. By Proposition 26.8.5, it is a $(c\log(n))^q$ approximation with probability at least $3/4$ for some constant c. Taking the median of $\Theta(\log(n))$ repetitions, we have that

$$c^{-q} \cdot \sum_{\vec{u}:u_1=l} |\rho_{\vec{u}}|^p \leq |w_l|^p \leq (c\log(n))^q \cdot \sum_{\vec{u}:u_1=l} |\rho_{\vec{u}}|^p$$

with probability $1 - 1/\operatorname{poly}(n)$, and we can then union bound over all such estimates every conducted over the course of the algorithm. We call the above estimate $|w_l|^p$ a $O((c\log(n))^q)$-error estimate of $\sum_{\vec{u}:u_1=l} |\rho_{\vec{u}}|^p$. Given this, we can correctly and independently sample the first coordinate of each of the $\Theta(r_3)$ samples. We now describe how to sample the i-th coordinate. So in general, suppose we have sampled $(a_1, ..., a_{i-1})$ so far, and we need to now sample $a_i \in [n_i]$ conditioned on $(a_1, ..., a_{i-1})$. We first consider

$$W^{i,k} = \left((\bigotimes_{k=1}^{i-1} e_{a_k}^\top) \otimes (\mathbb{I}_{n_i}) \otimes (\bigotimes_{k=i+1}^{q} Z^{k,j})\rho \right) \in \mathbb{R}^{n_i}$$

Note that the j-th coordinate $W_j^{i,k}$ for $W^{i,k}$ is an estimate of $\sum_{\vec{u}\in\Delta(\vec{a}):u_i=j} |\rho_{\vec{u}}|^p$. Again by By Proposition 26.8.5, with probability $1 - 1/\operatorname{poly}(n)$, we will have $|W_j^{i,k}|^p$ is a $O((c\log(n))^q)$-error estimate of $\sum_{\vec{u}\in\Delta(\vec{a}):u_i=j} |\rho_{\vec{u}}|^p$ or at least one $k \in [\tau]$. Our goal will now be to find all $j \in [n_i]$ such that $\sum_{\vec{u}\in\Delta(\vec{a}):u_i=j} |\rho_{\vec{u}}|^p \geq \Theta((c\log(n))^q/r_3^8) \sum_{\vec{u}\in\Delta(\vec{a})} |\rho_{\vec{u}}|^p$. We call such a j a *heavy hitter*.

Let $Q_i \subset [n_i]$ be the set of heavy hitters. To find all the heavy hitters, we use the count-sketch for heavy hitters with the Dyadic trick of definition 26.8.2. We construct this count-sketch of def 26.8.2 $S^i \in \mathbb{R}^{k'\times n_i}$ where $k' = O(\log^2(n)r_3^{16})$. We then compute $S^i W^{i,k}$, for $k = 1, 2, \ldots, \tau$, and obtain the set of heavy hitters $h \in H_{i,k} \subset [n_i]$ which satisfy $|W_j^{i,k}|^p \geq \Theta(1/r_3^8)\|W^{i,k}\|_p^p$. By the above discussion, we know that for each $j \in Q_i$, we

will have $|W_j^{i,k}|^p \geq \Theta(1/r_3^{16})\|W^{i,k}\|_p^p$ for at least one $k \in [\tau]$ with high probability. Thus $H_i = \cup_{k=1}^{\tau} H_{i,k} \supseteq Q_i$.

We now will decide to either sample a heavy hitter $\xi \in H_i$, or a non-heavy hitter $\xi \in [n_i] \setminus H_i$. By Proposition 26.8.5, we can compute a $O((c\log(n))^{-q})$-error estimate $\beta_i = \text{median}_{j\in\tau} \left|\left((\bigotimes_{k=1}^{i-1} e_{a_k}^\top) \otimes (\bigotimes_{k=i}^{q} Z^{k,j})\rho\right)\right|^p$ of $\sum_{\vec{u}\in\Delta(\vec{a})} |\rho_{\vec{u}}|^p$, meaning:

$$O(c^{-q}) \sum_{\vec{u}\in\Delta(\vec{a})} |\rho_{\vec{u}}|^p \leq \beta_i \leq O((c\log n)^q) \sum_{\vec{u}\in\Delta(\vec{a})} |\rho_{\vec{u}}|^p.$$

Again, by Proposition 26.8.5, we can compute a $O((c\log(n))^{-q})$-error estimate $\gamma_i \leftarrow \text{median}_{j\in[\tau]} \left((\bigotimes_{k=1}^{i-1} e_{a_k}^\top \right.$ of $\sum_{h\in[n_i]\setminus H_i} \sum_{\vec{u}\in\Delta(\vec{a}):u_i=j} |\rho_{\vec{u}}|^p$. It follows that

$$O(c^{-2q})\frac{\sum_{h\in[n_i]\setminus H_i} \sum_{\vec{u}\in\Delta(\vec{a}):u_i=j} |\rho_{\vec{u}}|^p}{\sum_{\vec{u}\in\Delta(\vec{a})} |\rho_{\vec{u}}|^p} \leq \frac{\gamma_i}{\beta_i} \leq O((c\log n)^{2q})\frac{\sum_{h\in[n_i]\setminus H_i} \sum_{\vec{u}\in\Delta(\vec{a}):u_i=j} |\rho_{\vec{u}}|^p}{\sum_{\vec{u}\in\Delta(\vec{a})} |\rho_{\vec{u}}|^p}$$

In other words, γ_i/β_i is a $O((c\log(n))^{2q})$-error approximation of the true probability that we should sample a non-heavy item. Thus with probability $1 - \gamma_i/\beta_i$, we choose to sample a heavy item.

To sample a heavy item, for each $\xi \in H_i$, by Proposition 26.8.5, we can compute an $O((c\log(n))^{-q})$-error estimate $\text{median}_{j\in\tau} \left|\left((\bigotimes_{k=1}^{i-1} e_{a_k}^\top) \otimes (e_\xi^\top) \otimes (\bigotimes_{k=i+1}^{q} Z^{k,j})\rho\right)\right|^p$ of $\sum_{\vec{u}\in\Delta(\vec{a}):u_i=\xi} |\rho_{\vec{u}}|^p$, meaning

$$\left(O(c^{-q}) \sum_{\vec{u}\in\Delta(\vec{a}):u_i=\xi} |\rho_{\vec{u}}|^p\right) \leq \text{median}_{j\in\tau} \left| \left((\bigotimes_{k=1}^{i-1} e_{a_k}^\top) \otimes (e_\xi^\top) \otimes (\bigotimes_{k=i+1}^{q} Z^{k,j})\rho \right) \right|^p$$
$$\leq \left(O((c\log n)^q) \sum_{\vec{u}\in\Delta(\vec{a}):u_i=\xi} |\rho_{\vec{u}}|^p\right)$$

Thus we can choose to sample a heavy item $\xi \in H_i$ from the distribution given by

$$\Pr\left[\text{sample } a_i \leftarrow \xi\right] = \frac{\text{median}_{j\in\tau} \left| \left((\bigotimes_{k=1}^{i-1} e_{a_k}^\top) \otimes (e_\xi^\top) \otimes (\bigotimes_{k=i+1}^{q} Z^{k,j})\rho \right) \right|^p}{\sum_{\xi'\in H_i} \text{median}_{j\in\tau} \left| \left((\bigotimes_{k=1}^{i-1} e_{a_k}^\top) \otimes (e_{\xi'}^\top) \otimes (\bigotimes_{k=i+1}^{q} Z^{k,j})\rho \right) \right|^p}$$

Which gives a $O((c\log(n))^{2q})$-error approximation to the correct sampling probability for a heavy item.

In the second case, with probability γ_i/β_i, we choose to not sample a heavy item. In this case, we must now sample a item from $[n_i] \setminus H_i$. To do this, we partition $[n_i]$ randomly into $\Omega_1, \ldots, \Omega_\eta$ for $\eta = 1/r_3^2$. Now there are two cases. First suppose that we have

$$\frac{\sum_{j\in[n_i]\setminus H_i} \sum_{\vec{u}\in\Delta(\vec{a}):u_i=j} |\rho_{\vec{u}}|^p}{\sum_{\vec{u}\in\Delta(\vec{a})} |\rho_{\vec{u}}|^p} \leq \Theta(1/r_3^3)$$

Now recall that γ_i/β_i was a $O((c\log(n))^{2q})$-error estimate of the ratio on the left hand side of the above equation, and γ_i/β_i was the probability with which we choose to sample a non-heavy hitter. Since we only repeat the sampling process r_3 times, the probability that we ever sample a non-heavy item in this case is at most $\Theta(q(c\log(n))^{2q}/r_3^2) < \Theta(q/r_3)$, taken over all possible repetitions of this sampling in the algorithm. Thus we can safely ignore this case, and condition on the fact that we never sample a non-heavy item in this

case. Otherwise, $\sum_{j\in[n_i]\setminus H_i}\sum_{\vec{u}\in\Delta(\vec{a}):u_i=j}|\rho_{\vec{u}}|^p > \Theta(1/r_3^3)\sum_{\vec{u}\in\Delta(\vec{a})}|\rho_{\vec{u}}|^p$, and it follows that $\sum_{\vec{u}\in\Delta(\vec{a}):u_i=j'}|\rho_{\vec{u}}|^p \leq \Theta(1/r_3^5\sum_{j\in[n_i]\setminus H_i})\sum_{\vec{u}\in\Delta(\vec{a}):u_i=j}|\rho_{\vec{u}}|^p$ for all $j'\in[n_i]\setminus H_i$, since we removed all $\Theta(1/r_3^8)$ heavy hitters from $[n_i]$ originally. Thus by Chernoff bounds, with high probability we have that $\sum_{j\in\Omega_i\setminus H_i}(\sum_{\vec{u}\in\Delta(\vec{a}):u_i=j}|\rho_{\vec{u}}|^p) = \Theta(1/\eta\sum_{j\in[n_i]\setminus H_i}\sum_{\vec{u}\in\Delta(\vec{a}):u_i=j}|\rho_{\vec{u}}|^p)$, which we can union bound over all repetitions.

Given this, by choosing $t\sim[\eta]$ uniformly at random, and then choosing $j\in\Omega_t\setminus H_i$ with probability proportional to its mass in $\Omega_t\setminus H_i$, we get a $\Theta(1)$ approximation of the true sampling probability. Since we do not know its exact mass, we instead sample from the distribution $\{\frac{\theta_j}{\sum_{j'\in\Omega_t\setminus H_i}\theta_{j'}}\}_{j\in\Omega_t\setminus H_i}$, where

$$\theta_j = \underset{l\in[\tau]}{\text{median}}\left(\left|(\bigotimes_{k=1}^{i-1}e_{a_k}^\top)\otimes(e_j^\top)\otimes(\bigotimes_{k=i+1}^{q}Z^{k,l})\rho\right|^p\right)$$

Again by Proposition 26.8.5, this gives a $O((c\log(n))^{2q})$-error approximation to the correct sampling probability. Note that at each step of sampling a coorindate of $\vec{a}$ we obtained at most $O((c\log(n))^{2q})$-error in the sampling probability. Thus, by oversampling by a $O((c\log(n))^{2q^2})$ factor, we can obtain the desired sampling probabilities. This completes the proof of correctness. Note that to improve the failure probability to $1-\delta$, we can simply scale r_3 by a factor of $1/\delta$.

Proof of Runtime. We now analyze the runtime. At every step $i=1,2,\ldots,q$ of the sampling, we compute $v_i^j \leftarrow S^i\left((\bigotimes_{k=1}^{i-1}e_{a_k}^\top)\otimes(\mathbb{I}_{n_i})\otimes(\bigotimes_{k=i+1}^{q}Z^{k,j})\rho\right)\in\mathbb{R}^{n_i}$ for $j=1,2,\ldots\Theta(\log(n))$. This is equal to

$$S^i\left((\bigotimes_{k=1}^{i-1}(A_k)_{a_k,*})\otimes(A_i)\otimes(\bigotimes_{k=i+1}^{q}Z^{k,j}A_k)x' - (\bigotimes_{k=1}^{i-1}e_{a_k}^\top)\otimes(\mathbb{I}_{n_i})\otimes(\bigotimes_{k=i+1}^{q}Z^{k,j})b\right)$$

We first consider the term inside of the parenthesis (excluding S^i). Note that the term $(\bigotimes_{k=i+1}^{q} Z^{k,j} A_k)$ was already pre-computed, and is a vector of length at most d, this this requires a total of $\widetilde{O}(\sum_{i=1}^{q} \operatorname{nnz}(A_i) + d)$ time. Note that these same values are used for every sample. Given this pre-computation, we can rearrage the first term to write $(\bigotimes_{k=1}^{i-1}(A_k)_{a_k,*}) \otimes (A_i) X' (\bigotimes_{k=i+1}^{q} Z^{k,j} A_k)^\top$ where X' is a matrix formed from x' so that x' is the vectorization of X' (this is done via reshaping Lemma 26.8.1). The term $y = X'(\bigotimes_{k=i+1}^{q} Z^{k,j} A_k)^\top$ can now be computed in $O(d)$ time, and then we reshape again to write this as $(\bigotimes_{k=1}^{i-1}(A_k)_{a_k,*}) Y A_i^\top$ where Y again is a matrix formed from y. Observe that $\zeta = \operatorname{vec}(\bigotimes_{k=1}^{i-1}(A_k)_{a_k,*} Y) \in \mathbb{R}^{d_i}$ can be computed in time $O(qd)$, since each entry is a dot product of a column $Y_{*,j} \in \mathbb{R}^{d_1 \cdot d_2 \cdots d_{i-1}}$ of Y with the $d_1 \cdot d_2 \cdots d_{i-1}$ dimensional vector $\bigotimes_{k=1}^{i-1}(A_k)_{a_k,*}$, which can be formed in $O(d_1 \cdot d_2 \cdots d_{i-1} q)$ time, and there are a total of d_i columns of Y.

Given this, The first entire term $S^i(\bigotimes_{k=1}^{i-1}(A_k)_{a_k,*}) \otimes (A_i) \otimes (\bigotimes_{k=i+1}^{q} Z^{k,j} A_k) x'$ can be rewritten as $S^i A_i \zeta$, where $\zeta = \zeta_{\vec{a}} \in \mathbb{R}^{d_i}$ can be computed in $O(dq)$ time for each sample $\vec{a}$. Thus if we recompute the value $S_i A_i \in \mathbb{R}^{k \times n}$, where $k = \widetilde{O}(r_3^{16})$, which can be done in time $\widetilde{O}(\operatorname{nnz} A_i)$, then every time we are sampling the i-th coordinate of some $\vec{a}$, computing the value of $S^i A_i \zeta_{\vec{a}}$ can be done in time $O(k d_i^2) = r_3^{O(1)}$.

We now consider the second term. We do a similar trick, reshaping $b \in \mathbb{R}^n$ into $B \in \mathbb{R}^{(n_1 \cdots n_i) \times (n_i \cdots n_q)}$ and writing this term as $((\bigotimes_{k=1}^{i-1} e_{a_k}^\top) \otimes (\mathbb{I}_{n_i})) B (\bigotimes_{k=i+1}^{q} Z^{k,j})^\top$ and computing $b' = B(\bigotimes_{k=i+1}^{q} Z^{k,j})^\top \in \mathbb{R}^{(n_1 \cdots n_i)}$ in $\operatorname{nnz}(B) = \operatorname{nnz}(b)$ time. Let $B' \in \mathbb{R}^{(n_1 \cdots n_{i-1}) \times n_i}$ be such that $\operatorname{vec}(B') = b'$, and we reshape again to obtain $(\bigotimes_{k=1}^{i-1} e_{a_k}^\top) B' (\mathbb{I}_{n_i}) = (\bigotimes_{k=1}^{i-1} e_{a_k}^\top) B'$ Now note that so far, the value B' did not depend on the sample $\vec{a}$ at all. Thus for each $i = 1, 2, \ldots, q$, B' (which depends only on i) can be pre-computed in $\operatorname{nnz}(b)$ time. Given this, the value $(\bigotimes_{k=1}^{i-1} e_{a_k}^\top) B'$ is just a row $B'_{(a_1,\ldots,a_k),*}$ of B' (or a column of $(B')^\top$). We

first claim that $nnz(B') \leq \text{nnz}(b) = \text{nnz}(B)$. To see this, note that each entry of B' is a dot product $B_{j,*}(\bigotimes_{k=i+1}^{q} Z^{k,j})^\top$ for some row $B_{j,*}$ of B, and moreover there is a bijection between these dot products and entries of B'. Thus for every non-zero entry of B', there must be a unique non-zero row (and thus non-zero entry) of B. This gives a bijection from the support of B' to the support of B (and thus b) which completes the claim. Since $S^i(B'_{(a_1,\ldots,a_k),*})^\top$ can be computed in $\widetilde{O}(\text{nnz}(B'_{(a_1,\ldots,a_k),*}))$ time, it follows that $S^i(B'_{(a_1,\ldots,a_k),*})^\top$ can be computed for all rows $(B'_{(a_1,\ldots,a_k),*})$ of B in $\widetilde{O}(\text{nnz}(b))$ time. Given this precomputation, we note that $(\mathbb{I}_{n_i}) \otimes (\bigotimes_{k=i+1}^{q} Z^{k,j})b$ is just $S^i(B'_{(a_1,\ldots,a_k),*})^\top$ for some $(a_1,\ldots,a_k)$, which has already been pre-computed, and thus requires no addition time per sample. Thus, given a total of $\widetilde{O}(\sum_{i=1}^{q} \text{nnz}(A_i) + q\,\text{nnz}(b) + r_3^{O(1)})$ pre-processing time, for each sample we can compute v_i^j for all $i \in [q]$ and $j \in [\tau]$ in $\widetilde{O}(r_3^{O(1)})$ time, and thus $\widetilde{O}(r_3^{O(1)})$ time over all r_3 samples.

Given this, the procedure to compute the heavy hitters $H_{i,j}$ takes $\widetilde{O}(r_3^{16})$ time by Definition 26.8.2 for each sample and $i \in [q], j \in [\tau]$. By a identical pre-computation and rearrangement argument as above, each β_i^j (and thus β_i) can be computed in $\widetilde{O}(r_3^{O(1)})$ time per sample after pre-computation. Now note that γ_i is simply equal to median $j \in [\tau](\beta_i^j - (\bigotimes_{k=1}^{i-1} e_{a_k}^\top) \otimes (Z_{H_i}^{k,j}) \otimes (\bigotimes_{k=i+1}^{q} Z^{k,j})\rho)$. Since $(Z_{H_i}^{k,j})$ is sparse, the above can similar be computed in $O(d|H_i|) = \widetilde{O}(r_3^{O(1)})$ time per sample after pre-computation. To see this, note that the b term of $(\bigotimes_{k=1}^{i-1} e_{a_k}^\top) \otimes (Z_{H_i}^{k,j}) \otimes (\bigotimes_{k=i+1}^{q} Z^{k,j})\rho$ can be written as $(\bigotimes_{k=1}^{i-1} e_{a_k}^\top)B'''(Z_{H_i}^{k,j})^\top$, where $B''' \in \mathbb{R}^{n_1 \cdots n_{i-1} \times n_i}$ is a matrix that has already been pre-computed and does not depend on the given sample. Then this quantity is just the dot product of a row of B''' with $(Z_{H_i}^{k,j})^\top$, but since $(Z_{H_i}^{k,j})$ is $|H_i|$-sparse, so the claim for the b term follows. For the $(A_1 \otimes \cdots \otimes A_q)$ term, just as we demonstrated in the discussion of computing v_i^j, note that

this can be written as $(\bigotimes_{k=1}^{i-1}(A_k)_{a_k,*})Y((A_i)_{H_i,*})^\top$ for some matrix $Y \in \mathbb{R}^{d_1\cdots d_i \times d_{i-1}}$ that has already been precomputed. Since $(A_i)_{H_i,*}$ only has $O(|H_i|)$ non-zero rows, this whole product can be computed in time $O(d|H_i|)$ as needed.

Similarly, we can compute the sampling probabilities

$$\Pr\left[\text{sample } a_i \leftarrow j\right] = \frac{\text{median}_{j\in\tau}\left|\left((\bigotimes_{k=1}^{i-1} e_{a_k}^\top)\otimes(e_\xi^\top)\otimes(\bigotimes_{k=i+1}^{q} Z^{k,j})\rho\right)\right|^p}{\sum_{\xi'\in H_i}\text{median}_{j\in\tau}\left|\left((\bigotimes_{k=1}^{i-1} e_{a_k}^\top)\otimes(e_{\xi'}^\top)\otimes(\bigotimes_{k=i+1}^{q} Z^{k,j})\rho\right)\right|^p}$$

for each every item $\zeta \in H_i$ in $\widetilde{O}(r_3^{O(1)})$ time after pre-computation, and note $|H_i| = \widetilde{O}(r_3^{O(1)})$ by definition 26.8.2. Thus the total time to sample a heavy hitter in a given coordinate $i \in [q]$ for each sample $\widetilde{O}(r_3^{O(1)})$ per sample, for an overall time of $\widetilde{O}(qr_3^{O(1)})$ over all samples and $i \in [q]$.

Finally, we consider the runtime for sampling a non-heavy item. Note that $|\Omega_t| = O(n_i/\eta)$ with high probability for all $t \in [\eta]$ by chernoff bounds. Computing each

$$\theta_j = \underset{l\in[\tau]}{\text{median}}\left(\left|(\bigotimes_{k=1}^{i-1} e_{a_k}^\top)\otimes(e_j^\top)\otimes(\bigotimes_{k=i+1}^{q} Z^{k,l})\rho\right|^p\right)$$

takes $O(qd)$ time after pre-computation, and so we spend a total of $O(qdn_i/\eta)$ time sampling an item from $\Omega_t \setminus H_i$. Since we only ever sample a total of r_3 samples, and $\eta = \Theta(r_3^2)$, the total time for sampling non-heavy hitters over the course of the algorithm in coordinate i is $o(n_i) = o(\text{nnz}(A_i))$ as needed, which completes the proof of the runtime.

Computing the Sampling Probabilities α_i The above arguments demonstrate how to sample efficiently from the desired distribution. We now must describe how the sampling probabilities α_i can be computed. First note, for each sample that is sampled in the above

way, at every step we compute exactly the probability with which we decide to sample a coordinate to that sample. Thus we know exactly the probability that we choose a sample, and moreover we can compute each q_i in $O(d)$ time as in Lemma 26.8.4. Thus we can compute the maximum of q_i and this probability exactly. For each item sampled as a result of the leverage score sampling probabilities q_i as in Lemma 26.8.4, we can also compute the probability that this item was sampled in the above procedure, by using the same sketching vectors $Z^{i,k}$ and count-sketches S^i. This completes the proof of the Lemma.

□

26.9 Missing Proofs from Section 26.4

In this section, we prove the correctness of our all-pairs regression algorithm 26.3. Our main theorem, Theorem 26.4.1, relies crucially on the sample routine developed in Section 26.9.1. We first prove the theorem which utilizes this routine, and defer the description and proof of the routine to Section 26.9.1.

Recall first the high level description of our algorithm (given formally in Figure 26.3). We pick $S_1, S_2 \in \mathbb{R}^{k \times n}$and S are sparse p-stable sketches. We then compute $M = (S_1 \otimes S_2)(F \otimes \mathbf{1} - \mathbf{1} \otimes F) = S_1F \otimes S_2\mathbf{1} - S_1\mathbf{1} \otimes S_2F$, where $F = [A, b]$. We then take the QR decomposition $M = QR$. Finally, we sample rows of $(F \otimes \mathbf{1} - \mathbf{1} \otimes F)R^{-1}$ with probability proportional to their ℓ_p norms. This is done by the sampling procedure described in Section 26.9.1. Finally, we solve the regression problem $\min_x \|\Pi(\bar{A}x - \bar{b})\|_p$, where Π is the diagonal row-sampling matrix constructed by the sampling procedure.

We begin by demonstrating that $S_1 \otimes S_2$ is a poly(d) distortion embedding for the

column span of $[\bar{A}, \bar{b}]$.

Lemma 26.9.1. *Let $S_1, S_2 \in \mathbb{R}^{k \times n}$ be sparse p-stable transforms, where $k = \text{poly}(d/(\epsilon\delta))$. Then for all $x \in \mathbb{R}^{d+1}$, with probability $1 - \delta$ we have*

$$1/O(d^4 \log^4 d)\|[\bar{A}, \bar{b}]x\|_p \leq \|(S_1 \otimes S_2)[\bar{A}, \bar{b}]x\|_p \leq O(d^2 \log^2 d)\|[\bar{A}, \bar{b}]x\|_p$$

Proof. Let $F = [A, b]$. Then a basis for the columns of $[\bar{A}, \bar{b}]$ is given by $F \otimes \mathbf{1} - \mathbf{1} \otimes F$. We first condition on both S_1, S_2 being a low-distortion embedding for the $d + 2$ dimensional column-span of $[F, \mathbf{1}]$. Note that this holds with large constant probability by 26.2.1.

So for any $x \in \mathbb{R}^{d+1}$, we first show the upper bound

$$\begin{aligned}
\|(S_1 \otimes S_2)(F \otimes \mathbf{1} - \mathbf{1} \otimes F)x\|_p &= \|(S_1 F \otimes S_2 \mathbf{1})x - (S_1 \mathbf{1} \otimes S_2 F)\|_p \\
&= \|S_1 F x \mathbf{1}^\top S_2^\top - S_1 \mathbf{1} x^\top F^\top S_2^\top\|_p \\
&= \|S_1(F x \mathbf{1}^\top - \mathbf{1} x^\top F^\top) S_2^\top\|_p \\
&\leq O(d \log d)\|(F x \mathbf{1}^\top - \mathbf{1} x^\top F^\top) S_2^\top\|_p \\
&\leq O(d^2 \log^2 d)\|F x \mathbf{1}^\top - \mathbf{1} x^\top F^\top\|_p \\
&= O(d^2 \log^2 d)\|(F \otimes \mathbf{1} - \mathbf{1} \otimes F)x\|_p
\end{aligned}$$

Where the first equality follows by properties of the Kronecker product [VL00], the second by reshaping Lemma 26.8.1. The first inequality follows from the fact that each column of $(F x \mathbf{1}^\top - \mathbf{1} x^\top F^\top) S_2^\top$ is a vector in the column span of $[F, \mathbf{1}]$, and then using that S_1 is a low distortion embedding. The second inequality follows from the fact that each row of $(F x \mathbf{1}^\top - \mathbf{1} x^\top F^\top)$ is a vector in the *column span* of $[F, \mathbf{1}]$, and similarly using that S_2 is a low

distortion embedding. The final inequality follows from reshaping. Using a similar sequence of inequalities, we get the matching lower bound as desired. □

We now prove our main theorem.

Theorem 26.4.1 *Given $A \in \mathbb{R}^{n\times d}$ and $b \in \mathbb{R}^n$, for $p \in [1,2]$ there is an algorithm for the All-Pairs Regression problem that outputs $\widehat{x} \in \mathbb{R}^d$ such that with probability $1-\delta$ we have*

$$\|\bar{A}\widehat{x} - \bar{b}\|_p \leq (1+\epsilon)\min_{x\in\mathbb{R}^d} \|\bar{A}x - \bar{b}\|_p$$

Where $\bar{A} = A \otimes \mathbf{1} - \mathbf{1} \otimes A \in \mathbb{R}^{n^2\times d}$ and $\bar{b} = b \otimes \mathbf{1} - \mathbf{1} \otimes b \in \mathbb{R}^{n^2}$. For $p < 2$, the running time is $\widetilde{O}(nd + (d/(\epsilon\delta))^{O(1)})$, and for $p = 2$ the running time is $O(\mathrm{nnz}(A) + (d/(\epsilon\delta))^{O(1)})$.

Proof. We first consider the case of $p = 2$. Here, we can use the fact that the TensorSketch random matirx $S \in \mathbb{R}^{k\times n}$ is a subspace embedding for the column span of $[\bar{A}, \bar{b}]$ when $k = \Theta(d/\epsilon^2)$ [DSSW18], meaning that $\|S[\bar{A}, \bar{b}]\|_2 = (1 \pm \epsilon)\|[\bar{A}, \bar{b}]x\|_2$ for all $x \in \mathbb{R}^{d+1}$ with probability 9/10. Moreover, $S\bar{A}$ and $S\bar{b}$ can be computed in $O(\mathrm{nnz}(A)+\mathrm{nnz}(b)) = O(\mathrm{nnz}(A))$ by [DSSW18] since they are the difference of Kronecker products. As a result, we can simply solve the regression problem $\widehat{x} = \arg\min_x \|S\bar{A}x - S\bar{b}\|_2$ in $\mathrm{poly}(kd)$ time to obtain the desired $\widehat{x}$.

For $p < 2$, we use the algorithm in Figure 26.3, where the crucial leverage score sampling procedure to obtain Π in step 7 of Figure 26.3 is described in Lemma 26.4.2. Our high level approach follows the general ℓ_p sub-space embedding approach of [DDH+09]. Namely, we first compute a low-distortion embedding $(S_1 \otimes S_2)(F \otimes \mathbf{1} - \mathbf{1} \otimes F)$. By Lemma 26.9.1, using sparse-p stable transformations S_1, S_2, we obtain the desired $\mathrm{poly}(d)$ distortion

embedding into $\mathbb{R}^{k^2}$, where $k = \text{poly}(d/\epsilon)$. Note that computing $(S_1 \otimes S_2)(F \otimes \mathbf{1} - \mathbf{1} \otimes F)$ can be done in $O(\text{nnz}(A) + \text{nnz}(b) + n)$ time using the fact that $(S_1 \otimes S_2)(F \otimes \mathbf{1}) = S_1 F \otimes S_2 \mathbf{1}$. As shown in [DDH+09], it follows that $M = (F \otimes \mathbf{1} - \mathbf{1} \otimes F)R^{-1}$ is an ℓ_p well-conditioned basis for the column span of $(F \otimes \mathbf{1} - \mathbf{1} \otimes F)$ (see definition 26.2.4). Then by Theorem 5 of [DDH+09], if we let $\widehat{\Pi}$ be the diagonal row sampling matrix such that $\widehat{\Pi}_{i,i} = 1/q_i^{1/p}$ for each i with probability $q_i \geq \min\{1, r\|M_{i,*}\|_p^p / \|M\|_p^p\}$ (and $\widehat{\Pi}_{i,i} = 0$ otherwise) for $r = \text{poly}(d \log(1/\delta)/\epsilon)$, then with probability $1 - \delta$ we have $\|\widehat{\Pi}(F \otimes \mathbf{1} - \mathbf{1} \otimes F)x\|_p = (1 \pm \epsilon)\|(F \otimes \mathbf{1} - \mathbf{1} \otimes F)x\|_p$ for all $x \in \mathbb{R}^{d+1}$. First assume that we had such a matrix.

Since $(\bar{A}x - \bar{b})$ is in the column span of $(F \otimes \mathbf{1} - \mathbf{1} \otimes F)$ for any $x \in \mathbb{R}^{d+1}$, it follows that $\|\widehat{\Pi}(\bar{A}x - \bar{b})\|_p = (1 \pm \epsilon)\|(\bar{A}x - \bar{b})\|_p$ for all $x \in \mathbb{R}^d$, which completes the proof of correctness. By Lemma 26.4.2, we can obtain a row sampling matrix Π in time $\widetilde{O}(nd + \text{poly}(d/\epsilon))$, except that the entries of Π are instead equal to either 0 or $1/\widetilde{q}_i^{1/p}$ where $\widetilde{q}_i = (1 \pm \epsilon^2)q_i$. Now let $\widehat{\Pi}$ be the idealized row sampling matrices from above, with entries either 0 or $1/q_i^{1/p}$ as needed for Theorem 5 of [DDH+09]. Note that for any matrix Z each row of $\widehat{\Pi}Zx$ is equal to ΠZx times some constant $1 - \epsilon^2 < c < 1 + \epsilon^2$. It follows that $\|\Pi(\bar{A}x - \bar{b})\|_p = (1 \pm \epsilon^2)\|\widehat{\Pi}(\bar{A}x - \bar{b})\|_p$ for all $x \in \mathbb{R}^d$, and thus the objective function is changed by at most a $(1 \pm \epsilon^2)$ term, which is simply handled by a constant factor rescaling of ϵ.

Finally, we can solve the sketched regression problem $\|\Pi(\bar{A}x - \bar{b})\|_p$ which has $\text{poly}(d/\epsilon)$ constraints and d variables in time $\text{poly}(d/\epsilon)$ using linear programming for $p = 1$ (see [CLS19] for the state of the art linear program sovler), or more generally interior point methods for convex programming for $p > 1$ (see [BCLL18] for the state of the art ℓ_p solver. Finally, the failure probability bound holds by union bounding over all the aforementioned results, and noting that the lowest probability event was the even that $S_1 \otimes S_2$ was a low distortion

embedding via Lemma 26.9.1. This completes the proof of the theorem.

□

26.9.1 Proof of Fast Sampling Lemma 26.4.2

We now provide a full proof of the main technical lemma of Section 26.4. The sampling algorithm is given formally in Algorithm 26.5. The following proof of Lemma 26.4.2 analyzes each step in the process, demonstrating both correctness and the desired runtime bounds.

Lemma 26.4.2 *Given $R \in \mathbb{R}^{(d+1)\times(d+1)}$ and $F = [A, b] \in \mathbb{R}^{n\times(d+1)}$, there is an algorithm that, with probability $1-\delta$ for any $\delta > n^{-c}$ for any constant c, produces a diagonal matrix $\Pi \in \mathbb{R}^{n^2\times n^2}$ such that $\Pi_{i,i} = 1/\widetilde{q}_i^{1/p}$ with probability $q_i \geq \min\{1, r\|M_{i,*}\|_p^p/\|M\|_p^p\}$ and $\Pi_{i,i} = 0$ otherwise, where $r = \mathrm{poly}(d/\epsilon)$ and $M = (F\otimes \mathbf{1} - \mathbf{1}\otimes F)R^{-1}$, and $\widetilde{q}_i = (1\pm\epsilon^2)q_i$ for all $i \in [n^2]$. The total time required is $\widetilde{O}(\mathrm{nnz}\, A + \mathrm{poly}(d/\epsilon))$.*

Proof. Our proof proceeds in several steps. We analyze the runtime concurrently with out analysis of correctness.

Reducing the number of Columns of R^{-1} We begin by generating a matrix $G \in \mathbb{R}^{(d+1)\times\xi}$ of i.i.d. $\mathcal{N}(0, 1/\sqrt{\xi})$ Gaussian random variables. We then compute $Y \leftarrow R^{-1}G$ in $\widetilde{O}(d^2)$ time. We first claim that it suffices to instead ℓ_p sample rows of $C = (F\otimes\mathbf{1}-\mathbf{1}\otimes F)Y = MG$. Note that each entry $|C_{i,j}|^p$ is distributed as $g^p\|M_{i,*}\|_2^p$ where G $\mathcal{N}(0, 1/\sqrt{\xi})$ Gaussian, which holds by the 2-stability of Gaussian random variables. Note that $\mathbb{E}[|g|^p] = \Theta(1/\xi)$, so $\mathbb{E}[\|C_{i,*}\|_p^p] = \|M_{i,*}\|_2^p$, and by sub-exponential concentration (see Chapter 2 of [Wai19]), we have that $\|C_{i,*}\|_p^p = (1\pm 1/10)\|M_{i,*}\|_2^p$ with probability $1 - 1/\mathrm{poly}(n)$, and we can union

bound over this holding for all $i \in [n^2]$. By relationships between the p norms, we have $\|M_{i,*}\|_p^p/d < \|M_{i,*}\|_2^p < \|M_{i,*}\|_p^p$, thus this changes the overall sampling probabilities by a factor between $\Theta(1/d^2)$ and $\Theta(d^2)$. Thus, we can safely oversample by this factor (absorbing it into the value of r) to compensate for this change in sampling probabilities.

Sampling a row from C. To sample a row from C, the approach will be to sample an entry $C_{i,j}$ of C with probability proportional to $\|C_{i,j}\|_p^p/\|C\|_p^p$. For every (i,j) sampled, we sample the entire i-th row of j, so that the j-th row is indeed sampled with probability proportional to its norm. Thus, it suffices to sample entries of C such that each $C_{i,j}$ is chosen with probability at least $\min\{1, r\|C_{i,j}\|_p^p/\|C\|_p^p\}$. First note that the i-th column of $C = (F \otimes \mathbf{1} - \mathbf{1} \otimes F)Y$ can be rearranged into a $n \times n$ matrix via Lemma 26.8.1, given by $(FY_{*,i}\mathbf{1}^\top - \mathbf{1}Y_{*,i}^\top F^\top)$. To ℓ_p sample a coordinate from C, it suffices to first ℓ_p sample a column of one of the above matrices, and then ℓ_p sample an entry from that column.

To do this, we first compute $FY \in \mathbb{R}^{n \times \xi}$, which can be done in time $\widetilde{O}(\text{nnz}\, A)$ because Y only has $\xi = \Theta(\log(n))$ columns. We then compute $Z(FY_{*,i}\mathbf{1}^\top - \mathbf{1}Y_{*,i}^\top F^\top) \in \mathbb{R}^{1 \times n}$ for all $i \in [d]$, where $Z \in \mathbb{R}^{1 \times n}$ is a fixed vector of i.i.d. p-stable random variables. Once FY has been computed, for each $i \in [\xi]$ it takes $O(n)$ time to compute this n-dimensional vector, thus the total time required to compute all ξ vectors is $\widetilde{O}(n)$. We repeat this process $t = O(\log(n))$ times with different p-stable vectors $Z^1, \ldots, Z^\top$, and take the median of each coordinate of $Z^j(FY_{*,i}\mathbf{1}^\top - \mathbf{1}Y_{*,i}^\top F^\top) \in \mathbb{R}^n$, $j \in [t]$, divided by the median of the p-stable distribution (which can be approximated to $(1 \pm \epsilon)$ error in $\text{poly}(1/\epsilon)$ time, see Appendix A.2 of [KNW10b] for details of this). This is done in Step 7 of Algorithm 26.5. It is standard this this gives a $(1 \pm 1/10)$ approximation the the norm $\|(FY_{*,i}\mathbf{1}^\top - \mathbf{1}Y_{*,i}^\top F^\top)_{*,l}\|_p$ for each

$i \in [d], l \in [n]$ with probability $1 - 1/\operatorname{poly}(n)$ (See the Indyk median estimator [Ind06]).

Now let $\sigma_{i,l}$ be our estimate of the norm $\|(FY_{*,i}\mathbf{1}^\top - \mathbf{1}Y_{*,i}^\top F^\top)_{*,l}\|_p$, for all $i \in [\xi]$ and $l \in [n]$. We now sample a columns $(i,l) \in [\xi] \times [n]$, where each (i,l) is chosen with probability $\sigma_{i,l}/(\sum_{i',l'} \sigma_{i',l'})$. We repeat this process $\Theta(r)$ times, to obtain a *multi-set* $T \subset [\xi] \times [n]$ of sampled columns (i,l). We stress that T is a multi-set, because the same column (i,l) may have been chosen for multiple samples, and each time it is chosen we must independently sample one of the entries of that column. For any $(i,l) \in T$, we define $W^{(i,l)} = (FY_{*,i}\mathbf{1}^\top - \mathbf{1}Y_{*,i}^\top F^\top)_{*,l} = (FY_{*,i} - \mathbf{1}(FY)_{l,i})$.

ℓ_p Sampling an entry from $W^{(i,l)}$. Now fix any $(i,l) \in T$. We show how to ℓ_p sample an entry from the vector $W^{(i,l)} \in \mathbb{R}^n$. In other words, for a given $j \in [n]$, we want to sample $W_j^{(i,l)} \in [n]$ with probability at least $r|W_j^{(i,l)}|^p/\|W^{(i,l)}\|_p^p$. We do this in two steps. First, let $S_0 \in \mathbb{R}^{k \times n}$ be the count-sketch for heavy hitters of definition 26.8.2, where $k = \operatorname{poly}(r)$. Note that we can compute $S_0 FY$ and $S_0\mathbf{1}$ in time $\widetilde{O}(n)$, since $FY \in \mathbb{R}^{n \times \xi}$. Once this is done, for each $(i,l) \in T$ we can compute $S_0 W^{(i,l)}$ in $O(k)$ time by computing $(S_0\mathbf{1}(FY)_{l,i})$ (note that FY and $S_0\mathbf{1}$ are already computed), and subtracting it off from the i-th column of $S_0 FY$, so the total time is $\widetilde{O}(n + \operatorname{poly}(d/\epsilon))$ to compute $S_0 W^{(i,l)}$ for all $(i,l) \in |T|$. Now we can obtain the set $Q_0^{(i,l)} \subset [n]$ containing all the $\widetilde{\Omega}(1/\sqrt{k})$ heavy hitters in $W^{(i,l)}$ with high probability. We can then explicitly compute the value of $W_j^{(i,l)}$ for all $j \in Q_0^{(i,l)}$, and exactly compute the set

$$H^{(i,l)} = \left\{ j \in [n] \;\middle|\; |W_j^{(i,l)}|^p > \beta/r^{16}\|W^{(i,l)}\|_p^p \right\},$$

all in $\widetilde{O}(k)$ time via definition 26.8.2, where $\beta > 0$ is a sufficiently small constant (here we use the fact that $|x|_p \geq |x|_2$ for $p \leq 2$). Note that we use the same sketch S_0 to compute all

sets $Q_0^{(i,l)}$, and union bound the event that we get the heavy hitters over all poly(d/ϵ) trails.

We are now ready to show how we sample an index from $W^{(i,l)}$. First, we estimate the total ℓ_p norm of the items in $[n_i] \setminus H^{(i,l)}$ (again with the Indyk median estimator), and call this $\alpha_{(i,l)}$ as in Algorithm 26.5, which can be computed in $O(|H^{(i,l)}|)$ additional time (by subtracting off the $|H^{(i,l)}|$ coordinates $ZW_\zeta^{(i,l)}$ for all heavy hitters $\zeta \in H^{(i,l)}$ from our estimate $\sigma_{(i,l)}$), and with probability $\alpha_{(i,l)}/\sigma_{(i,l)}$, we choose to sample one of the items of $H^{(i,l)}$, which we can then sample from the distribution $|W_j^{(i,l)}|^p/(\sum_{j\in H^{(i,l)}} |W_j^{(i,l)}|^p)$. Since all the $\sigma_{(i,l)}, \alpha_{(i,l)}$'s were constant factor approximations, it follows that we sampled such an item with probability $\Omega(r|W_{j'}^{(i,l)}|^p/\|C\|_p^p)$ as needed. Otherwise, we must sample an entry from $[n] \setminus H^{(i,l)}$. To do this, we first randomly partition $[n]$ into $\eta = \Theta(r^4/\epsilon^4)$ subsets $\Omega_1, \Omega_2, \ldots, \Omega_\eta$.

We now make the same argument made in the proof of Lemma 26.8.7, considering two cases. In the first case, the ℓ_p mass of $[n]\setminus H^{(i,l)}$ drops by a $1/r^2$ factor after removing the heavy hitters. In this case, $\alpha_{(i,l)}/\sigma_{(i,l)} = O(1/r^2)$, thus we will never *not* sample a heavy hitter with probability $1-O(1/r)$, which we can safely ignore. Otherwise, the ℓ_p drops by less than a $1/r^2$ factor, and it follows that all remaining items must be at most a β/r^{14} heavy hitter over the remaining coordinates $[n]\setminus H^{(i,l)}$ (since if they were any larger, they would be β/r^{16} heavy hitters in $[n]$, and would have been removed in $H^{(i,l)}$). Thus we can assume we are in the second case. So by Chernoff bounds, we have $\sum_{j\in\Omega_t} |W_j^{(i,l)}|_p = \Theta(\frac{1}{\eta}\sum_{j\in[n]\setminus H^{(i,l)}} |W_j^{(i,l)}|_p)$ with probability greater than $1-\exp(-\Omega(r))$. We can then union bound over this event occurring for all $t \in [\eta]$ and all $(i,l) \in T$. Given this, if we uniformly sample a $t \sim [\eta]$, and then ℓ_p sample a coordinate $j \in \Omega_t$, we will have sampled this coordinate with the correct probability up to a constant factor. We now sample such a t uniformly from η.

To do this, we generate a diagonal matrix $D \in \mathbb{R}^{n \times n}$, where $D_{i,i} = 1/u_i^{1/p}$, where $u_1, \ldots, u_n$ are i.i.d. exponential random variables. For any set $\Gamma \subset [n]$, let D_Γ be D with all diagonal entries (j, j) such that $j \notin \Gamma$ set equal to 0. Now let $S \in \mathbb{R}^{k' \times n}$ be a second instance of count-sketch for heavy hitters of definition 26.8.2, where we set $k' = \text{poly}(k)$ from above. It is known that returning $j^* = \arg\max_{j \in \Omega_t \setminus H^{(i,l)}} |(DW^{(i,l)})_j|$ is a perfect ℓ_p sample from $\Omega_t \setminus H^{(i,l)}$ [JW18]. Namely, $\Pr[j^* = j] = |W_j^{(i,l)}|^p / \|W_{\Omega_t \setminus H^{(i,\ell)}}\|_p^p$ for any $j \in \Omega_t \setminus H^{(i,\ell)}$. Thus it will suffice to find this j^*. To find j^*, we compute $S(DW)_{\Omega_t \setminus H^{(i,\ell)}}$. Note that since FY has already been computed, to do this we need only compute $SD_{\Omega_t \setminus H^{(i,\ell)}} FY_{*,i}$ and $SD_{\Omega_t \setminus H^{(i,\ell)}} \mathbf{1}(FY)_{\ell,i}$, which takes total time $\widetilde{O}(|\Omega_t \setminus H^{(i,\ell)}|) = \widetilde{O}(n/\eta)$. We then obtain a set $Q^{(i,l)} \subset \Omega_t \setminus H^{(i,\ell)}$ which contains all j with $|(DW^{(i,l)})_j| \geq \widetilde{\Omega}(1/\sqrt{k'}) \|(DW)_{\Omega_t \setminus H^{(i,\ell)}}\|_2$.

As noted in [JW18], the value $\max_{j \in \Omega_t \setminus H^{(i,l)}} |(DW^{(i,l)})_j|$ is distributed identically to $\|W_{\Omega_t \setminus H^{(i,\ell)}}\|_p / u^{1/p}$ where u is again an exponential random variable. Since exponential random variables have tails that decay like $e^{-\Omega(x)}$, it follows that with probability $1 - \exp(-\Omega(r))$ that we have $\max_{j \in \Omega_t \setminus H^{(i,l)}} |(DW^{(i,l)})_j| = \Omega(\|W_{\Omega_t \setminus H^{(i,\ell)}}\|_p / r)$, and we can then union bound over the event that this occurs for all $(i, l) \in T$ and Ω_t. Given this it follows that $(DW^{(i,l)})_{j^*} = \Omega(\|W_{\Omega_t \setminus H^{(i,\ell)}}\|_p / r)$. Next, for any constant $c \geq 2$, by Proposition 1 of [JW18], we have $\|((DW)_{\Omega_t \setminus H^{(i,\ell)}})_{\texttt{tail}(c \log(n))}\|_2 = \widetilde{O}(\|W^{(i,l)}_{\Omega_t \setminus H^{(i,\ell)}}\|_p)$ with probability $1 - n^{-c}$, where for a vector x, $x_{(\overline{[t]})}$ is x but with the top t largest (in absolute value) entries set equal to 0. Since there are at most $c \log(n)$ coordinates in $(DW)_{\Omega_t \setminus H^{(i,\ell)}}$ not counted in $((DW)_{\Omega_t \setminus H^{(i,\ell)}})_{\texttt{tail}(c \log(n))}$, and since $(DW)_{j^*}$ is the largest coordinate in all of $(DW)_{\Omega_t \setminus H^{(i,\ell)}}$, by putting together all of the above it follows that $(DW)_{j^*}$ is a $\widetilde{\Omega}(1/r)$-heavy hitter in $(DW)_{\Omega_t \setminus H^{(i,\ell)}}$. Namely, that $|(DW)_{j^*}| \geq \widetilde{\Omega}(\|(DW)_{\Omega_t \setminus H^{(i,\ell)}}\|_2 / r)$. Thus, we conclude that $j^* \in Q^{(i,l)}$.

Given that $j^* \in Q^{(i,l)}$, we can then compute the value $(DW^{(i,l)})_j = D_{j,j}(FY_{j,i} - FY_{l,i})$ in $O(1)$ time to find the maximum coordinate j^*. Since $|Q^{(i,l)}| = O(k') = O(\text{poly}(d/\epsilon))$, it follows that the total time required to do this is $\widetilde{O}(n/\eta + \text{poly}(d/\epsilon))$. Since we repeat this process for each $(i,l) \in T$, and $|T| = \Theta(r)$ whereas $\eta = \Theta(r^4)$, it follows that the total runtime for this step is $\widetilde{O}(n/r^3 + \text{poly}(d/\epsilon))$. By [JW18], the result is a perfect ℓ_p sample from $(DW)_{\Omega_t \setminus H^{(i,\ell)}}$, which is the desired result. To complete the proof, we note that the only complication that remains is that we utilize the same scaling matrix D to compute the sampled used in each of the columns $W^{(i,l)}$ for each $(i,l) \in T$. However, note that for $t \neq t'$, we have that D_{Ω_t} and D_{Ω_t} are independent random variables. Thus it suffices to condition on the fact that the $t \in [\eta]$ that is sampled for each of the $|T|$ repetitions of sampling a Ω_t are distinct. But this occurs with probability at least $1/r$, since $|T| = \Theta(r)$ and $\eta = \Theta(r^4)$. Conditioned on this, all $|T|$ samples are independent, and each sample is an entry $C_{i,j}$ of C such that the probability that a given (i,j) is chosen is $|C_{i,j}|^p / \|C\|_p^p$. Repeating this sampling $\Theta(r)$ times, we get that each $C_{i,j}$ is sampled with probability at least $\min\{1, r|C_{i,j}|^p/\|C\|_p^p\}$, which completes the proof of correctness. Note that the dominant runtime of the entire procedure was $\widetilde{O}(\text{nnz}(A) + \text{poly}(d/\epsilon))$ as stated, and the probability of success was $1 - \exp(-r) + 1/\text{poly}(n)$, which we can be amplified to any $1 - \delta$ for $\delta > 1/n^c$ for some constant c by increasing the value of r by $\log(1/\delta)$ and the number of columns of the sketch G to $\log(1/\delta)$, which does not effect the $\widetilde{O}(\text{nnz}(A) + \text{poly}(d/\epsilon))$ runtime.

Computing approximations $\widetilde{q_i}$ for q_i. It remains now how to compute the approximate sampling probabilities $\widetilde{q_i}$ for $\Theta(r)$ rows of C that were sampled. Note that to sample an entry, in C, we first sampled the $n \times 1$ submatrix $W^{(i,l)}$ of C which contained it, where the

probability that we sample this submatrix is known to us. Next, if the entry of C was a heavy hitter in $W^{(i,l)}$, we exactly compute the probability that we sample this entry, and sample it with this probability. If the entry j of $W^{(i,l)}$ is not a heavy hitter, we first sample an Ω_t uniformly with probability exactly $1/\eta$. The last step is sampling a coordinate from $W^{(i,l)}_{\Omega_t \setminus H^{(i,l)}}$ via exponential scaling. However, we do not know the exact probability of this sampling, since this will be equal to $|W^{(i,l)}_j|^p / \|W^{(i,l)}_{\Omega_t \setminus H^{(i,l)}}\|_p^p$, and we do not know $\|W^{(i,l)}_{\Omega_t \setminus H^{(i,l)}}\|_p^p$ exactly. Instead, we compute it approximately to error $(1 \pm \epsilon^2)$ as follows. For each $(i,l) \in T$ and $\alpha = 1, 2, \ldots, \Theta(\log(n)/\epsilon^4)$, we compute $Z^{(\alpha)} W^{(i,l)}_{\Omega_t \setminus H^{(i,l)}}$, where $Z \in \mathbb{R}^{1 \times |\Omega_t \setminus H^{(i,l)}|}$ is a vector of p-stable random variables. Again, we use the Indyk median estimator [Ind06], taking the median of these $\Theta(\log(n)/\epsilon^4)$ repetitions, to obtain an estimate of $\|W^{(i,l)}_{\Omega_t \setminus H^{(i,l)}}\|_p^p$ with high probability to $(1 \pm \epsilon^2)$ relative error. Each repetition requires $O(|\Omega_t \setminus H^{(i,l)}|)$ additional time, and since $|\Omega_t \setminus H^{(i,l)}||T| = o(\epsilon^4 n / r^3)$, it follows that the total computational time is at most an additive $o(n)$, thus computing the $\widetilde{q}_i$'s to error $(1 \pm \epsilon^2)$ does not effect the overall runtime.

□

26.10 Missing Proofs from Section 26.5

We first give the proof of the main theorem. The proof relies on Lemma 26.10.2, which the rest of this section will be devoted to proving.

Theorem 26.5.1 *For any constant $q \geq 2$, there is an algorithm which runs in time $O(\sum_{i=1}^{q} \operatorname{nnz}(A_i) + d\,\mathrm{poly}(k/\epsilon))$ and outputs a rank k-matrix B in factored form such that $\|B - A\|_F \leq (1+\epsilon)\,\mathrm{OPT}_k$ with probability $9/10$.*

Proof. By Lemma 26.10.2, we have $(1-\epsilon)\|A-AP\|_F^2 \leq \|M-MP\|_F^2+c \leq (1+\epsilon)\|A-AP\|_F^2$ for all rank k projection matrices P. In particular, we have

$$\min_P (1+\epsilon)\|A-AP\|_F^2 + c = (1+\epsilon)\,\mathrm{OPT}_k^2$$

where the minimum is taken over all rank k projection matrices. The minimizer P on the LHS is given by the projection onto the top k singular space of M. Namely, $MP = MU^\top U$ where U is the top k singular row vectors of M. Thus $\|M - MU^\top U\|_F^2 + c \leq (1+\epsilon)\,\mathrm{OPT}_k^2$. Moreover, we have $\|A - AU^\top U\|_F^2 \leq (1+2\epsilon)(\|M - MU^\top U\|_F^2 + c) \leq (1+4\epsilon)\,\mathrm{OPT}_k^2$. Thus $\|A - AU^\top U\|_F \leq (1+O(\epsilon))\,\mathrm{OPT}_k$ as needed.

For runtime, note that we first must compute $M = (\otimes_{i=1}^q S_i)(A_1 \otimes A_2) = S_1A_1 \otimes \cdots \otimes S_qA_q$. Now S_iA_i can be computed in $O(\mathrm{nnz}(A_i))$ time for each i [CW13]. One all S_iA_i are computed, their Kronecker product can be computed in time $O(qk_1k_2\cdots k_qd) = \mathrm{poly}(kd/\epsilon)$. Given $M \in \mathbb{R}^{k_1\cdots k_q \times d}$, the top k singular vectors U can be computed by computing the SVD of M, which is also done in time $\mathrm{poly}(kd/\epsilon)$. Once U is obtained, the algorithm can terminate, which yields the desired runtime. □

To complete the proof of the main theorem, we will need to prove Lemma 26.10.2. To do this, we begin by introducing two definitions.

Definition 26.10.1. A random matrix S is called a ϵ-subspace embedding for a rank k subspace $\mathcal{V}$ we have simultaneously for all $x \in \mathcal{V}$ that $\|Sx\|_2 = (1 \pm \epsilon)\|x\|_2$.

Definition 26.10.2. A random matrix S satisfies the ϵ-approximate matrix product property if, for any fixed matrices A, B, of the appropriate dimensions, we have $\Pr[\|A^\top S^\top SB - A^\top B\|_F \leq \epsilon\|A\|_F\|B\|_F] \geq 9/10$.

We now show that S is both a subspace embedding and satisfies approximate matrix product, where $S = \otimes_{i=1}^{q} S_i$ and $S_i \in \mathbb{R}^{k_i \times n_i}$ are count-sketch matrices.

Lemma 26.10.1. *If $S = (\otimes_{i=1}^{q} S_i)$ with $S_i \in \mathbb{R}^{k_i \times n_i}$, $k_1 = k_2 = \cdots = k_q = \Theta(qk^2/\epsilon^2)$, then S is an ϵ-subspace embedding for any fixed k dimensional subspace $\mathcal{V} \subset \mathbb{R}^n$ with probability $9/10$, and also satisfies the (ϵ/k)-approximate matrix product property.*

Proof. We first show that S satisfies the $O(\epsilon/k, 1/10, 2)$-JL moment property. Here, the (ϵ, δ, ℓ)-JL moment property means that for any fixed $x \in \mathbb{R}^n$ with $\|x\|_2 = 1$, we have $\mathbb{E}[(\|Sx\|_2^2 - 1)^2] \leq \epsilon^{\ell}\delta$, which will imply approximate matrix product by the results of [KN14].

We prove this by induction on q. Let $\bar{k} = k_1$. First suppose $S = (Q \otimes T)$, where $Q \in \mathbb{R}^{k_1 \times n_1}$ is a count-sketch, and $T \in \mathbb{R}^{k' \times n'}$ is any random matrix which satisfies $\mathbb{E}[\|Tx\|_2^2] = \|x\|_2^2$ ($T \in \mathbb{R}^{k' \times n'}$ is unbiased), and $\mathbb{E}[(\|Tx\|_2 - 1)^2] \leq 1 + c/\bar{k}$ for some value $c < \bar{k}$. Note that both of these properties are satisfied with $c = 4$ if $T \in \mathbb{R}^{k_2 \times n_2}$ is itself a count-sketch matrix [CW13]. Moreover, these are the only properties we will need about T, so we will. We now prove that $\mathbb{E}[\|(S \otimes T)x\|_2^2] = 1$ and $\mathbb{E}[\|(S \otimes T)x\|_2^4] \leq 1 + (c+4)/\bar{k}$ for any unit vector x.

Fix any unit $x \in \mathbb{R}^n$ now (here $n = n_1 n'$), and let $x^j \in \mathbb{R}^{n'}$ be the vector obtained by restricted x to the coordinates $jn_1 + 1$ to $(j+1)n_1$. For any $i \in [k_1], j \in [k']$, let $i_j = (i-1)k' + j$. Let $h_Q(i) \in [k_1]$ denote the row where the non-zero entry in the i-th column is placed in Q. Let $\sigma_Q(i) \in \{1, -1\}$ denote the sign of the entry $Q_{h_Q(i),i}$. Let $\delta_Q(i, j)$

indicate the event that $h_Q(i) = j$. First note that

$$\begin{aligned}
\mathbb{E}\left[\sum_{i,j}((Q \otimes T)x)_{ij}^2\right] &= \mathbb{E}\left[\sum_{i=1}^{k_1}\sum_{j=1}^{k'}\left(\sum_{\tau=1}^{n_1}\delta_Q(\tau,i)\sigma_Q(\tau)(Tx^\tau)_j\right)^2\right] \\
&= \mathbb{E}\left[\sum_{i=1}^{k_1}\sum_{j=1}^{k'}\sum_{\tau=1}^{n_1}\delta_Q(\tau,i)(Tx^\tau)_j^2\right] \\
&= \mathbb{E}\left[\sum_{\tau=1}^{n_1}\sum_{i=1}^{k_1}\sum_{j=1}^{k'}\delta_Q(\tau,i)(Tx^\tau)_j^2\right] \\
&= \mathbb{E}\left[\sum_{\tau=1}^{n_1}\|Tx^\tau\|_2^2\right] \\
&= \|x\|_2^2
\end{aligned}$$

Where the last equality follows because count-sketch T is unbiased for the base case, namely that $\mathbb{E}[\|Tx\|_2^2] = \|x\|_2^2$ for any x [Woo14b], or by induction. We now compute the second moment,

$$\begin{aligned}
\mathbb{E}\left[\left(\sum_{i,j}((Q \otimes T)x)_{ij}^2\right)^2\right] &= \mathbb{E}\left[\left(\sum_{i,j}\left(\sum_{\tau=1}^{n_1}\delta_Q(\tau,i)\sigma_Q(\tau)(Tx^\tau)_j\right)^2\right)^2\right] \\
&= \mathbb{E}\left[\left(\sum_{i,j}\sum_{\tau_1,\tau_2}^{n_1}\delta_Q(\tau_1,i)\sigma_Q(\tau_1)(Tx^{\tau_1})_j\delta_Q(\tau_2,i)\sigma_Q(\tau_2)(Tx^{\tau_2})_j\right)^2\right] \\
&= \sum_{\tau_1,\tau_2,\tau_3,\tau_4}^{n_1}\mathbb{E}\left[\left(\sum_{i,j}\delta_Q(\tau_1,i)\sigma_Q(\tau_1)(Tx^{\tau_1})_j\delta_Q(\tau_2,i)\sigma_Q(\tau_2)(Tx^{\tau_2})_j\right)\right. \\
&\quad \left.\cdot\left(\sum_{i,j}\delta_Q(\tau_3,i)\sigma_Q(\tau_3)(Tx^{\tau_3})_j\delta_Q(\tau_4,i)\sigma_Q(\tau_4)(Tx^{\tau_4})_j\right)\right].
\end{aligned}$$

We now analyze the above expectation. There are several cases for the expectation of each

term. First, we bound the sum of the expectations when $t_1 = t_2 = t_3 = t_4$ by

$$\begin{aligned}
&\sum_{\tau=1}^{n_1} \mathbb{E}\left[\left(\sum_{i,j} \delta_Q(\tau,i)\sigma_Q(\tau)(Tx^\tau)_j \delta_Q(\tau,i)\sigma_Q(\tau)(Tx^\tau)_j\right)\right. \\
&\left.\cdot\left(\sum_{i,j} \delta_Q(\tau,i)\sigma_Q(\tau)(Tx^\tau)_j \delta_Q(\tau,i)\sigma_Q(\tau)(Tx^\tau)_j\right)\right] \\
\leq &\sum_{\tau=1}^{n_1} \mathbb{E}\left[\|Tx^\tau\|_2^4\right] = 1 + c/\bar{k}
\end{aligned}$$

Where the last equation follows from the variance of count-sketch [CW13] for the base case, or by induction for $q \geq 3$. We now bound the sum of the expectations when $t_1 = t_2 \neq t_3 = t_4$ by

$$\begin{aligned}
&\sum_{\tau_1 \neq \tau_2} \mathbb{E}\left[\left(\sum_{i,j} \delta_Q(\tau_1,i)\sigma_Q(\tau_1)(Tx^{\tau_1})_j \delta_Q(\tau_1,i)\sigma_Q(\tau_1)(Tx^{\tau_1})_j\right)\right. \\
&\left.\cdot\left(\sum_{i,j} \delta_Q(\tau_2,i)\sigma_Q(\tau_2)(Tx^{\tau_2})_j \delta_Q(\tau_2,i)\sigma_Q(\tau_2)(Tx^{\tau_2})_j\right)\right] \\
&\leq \sum_{\tau_1 \neq \tau_2} \mathbb{E}[\|Tx^{\tau_1}\|_2^2 \|Tx^{\tau_2}\|_2^2/k_1] \\
&\leq \mathbb{E}[\|Tx\|_2^4/k_1] \leq (1 + c/\bar{k})/k_1.
\end{aligned}$$

We can similarly bound the sum of the terms with $t_1 = t_3 \neq t_2 = t_4$ and $t_1 = t_4 \neq t_3 = t_2$ by $(1 + c/\bar{k})/k_1$, giving a total bound on the second moment of

$$\mathbb{E}[\|(Q \otimes T)x\|_2^4] \leq 1 + c/\bar{k} + 3(1 + c/\bar{k})/k_1) \leq 1 + (4 + c)/\bar{k}$$

since any term with a $t_i \notin \{t_1, t_2, t_3, t_4\} \setminus \{t_i\}$ immediately has expectation 0. By induction, it follows that $\mathbb{E}[(\otimes_{i=1}^q S_i)x\|_2^2] = 1$ for any unit x, and $\mathbb{E}[(\otimes_{i=1}^q S_i)x\|_2^4] \leq 1 + (4q + c)/\bar{k}$, where c is the constant from the original variance of count-sketch. Setting $\bar{k} = k_1 = \cdots = k_q = \Theta(qk^2/\epsilon^2)$ with a large enough constant, this completes the proof that $S = (\otimes_{i=1}^q S_i)$

has the $O(\epsilon/k, 1/10, 2)$-JL moment property. Then by Theorem 21 of [KN14], we obtain the approximate matrix product property:

$$\Pr[\|A^\top S^\top SB - A^\top B\|_F \leq O(\epsilon/k)\|A\|_F\|B\|_F] \geq 9/10$$

for any two matrices A, B. Letting $A = B^\top = U$ where $U \in \mathbb{R}^{n\times k}$ is a orthogonal basis for any k-dimensional subspace $\mathcal{V} \subset \mathbb{R}^n$, it follows that

$$\|U^\top S^\top SU - I_k\|_F \leq O(\epsilon/k)\|U\|_F^2 \leq O(\epsilon),$$

where the last step follows because U is orthonormal, so $\|U\|_F^2 = k$. Since the Frobenius norm upper bounds the spectral norm $\|\cdot\|_2$, we have $\|U^\top S^\top SU - I_k\|_2 \leq O(\epsilon)$, from which it follows that all the eigenvalues of $U^\top S^\top SU$ are in $(1 - O(\epsilon), 1 + O(\epsilon))$, which implies $\|SUx\|_2 = (1 \pm O(\epsilon))\|x\|_2$ for all $x \in \mathbb{R}^n$, so for any $y \in \mathcal{V}$, let x_y be such that $y = Ux_y$, and then $\|Sy\|_2 = \|SUx_y\|_2 = (1 \pm O(\epsilon))\|x_y\|_2 = (1 \pm O(\epsilon))\|Ux_y\|_2 = (1 \pm O(\epsilon))\|y\|_2$, which proves that S is a subspace embedding for $\mathcal{V}$ (not the second to last inequality holds because U is orthonormal). □

Finally, we are ready to prove Lemma 26.10.2.

Lemma 26.10.2. *Let $S = (\otimes_{i=1}^q S_i)$ with $S_i \in \mathbb{R}^{k_i\times n_i}$, $k_1 = k_2 = \cdots = k_q = \Theta(qk^2/\epsilon^2)$. Then with probability $9/10$ SA is a Projection Cost Preserving Sketch (PCP) for A, namely for all rank k orthogonal projection matrix $P \in \mathbb{R}^{d\times d}$,*

$$(1 - \epsilon)\|A - AP\|_F^2 \leq \|SA - SAP\|_F^2 + c \leq (1 + \epsilon)\|A - AP\|_F^2$$

where $c \geq 0$ is some fixed constant independent of P (but may depend on A and SA).

Proof. To demonstrate that SA is a PCP, we show that the conditions of Lemma 10 of [CEM+15] hold, which imply this result. Our result follows directly from Theorem 12 of [CEM+15]. Note that all that is needed (as discussed below the theorem) for the proof is that S is an ϵ-subspace embedding for a fixed k-dimensional subspaces, and that S satisfies the $(\epsilon/\sqrt{k})$ approximate matrix product property. By Lemma 26.10.1, we have both ϵ-subspace embedding for S as well as a stronger (ϵ/k) approximate matrix product property. Thus Theorem 12 holds for the random matrix S when $k_1 = k_2 = \cdots = k_q = \Theta(qk^2/\epsilon^2)$, which completes the proof.

□

26.11 Entry-wise Norm Low T-rank Approximation

We now demonstrate our results for low trank approximation of arbitrary input matrices. Specifically, we study the following problem, defined in [VL00]: given $A \in \mathbb{R}^{n^2 \times n^2}$, the goal is to output a trank-k matrix $B \in \mathbb{R}^{n^2 \times n^2}$ such that

$$\|B - A\|_\xi \leq \alpha \cdot \mathrm{OPT}\,. \tag{26.1}$$

for some $\alpha \geq 1$, where $\mathrm{OPT} = \min_{\mathrm{trank}-k\ A'} \|A' - A\|_\xi$,, where the trank of a matrix B is defined as the smallest integer k such that B can be written as a summation of k matrices, where each matrix is the Kronecker product of q matrices with dimensions $n \times n$: $B = \sum_{i=1}^{k} U_i \otimes V_i$, where $U_i, V_i \in \mathbb{R}^{n \times n}$.

Using Lemma 26.8.1, we can rearrange the entries in $A \in \mathbb{R}^{n^2 \times n^2}$ to obtain $\overline{A} \in \mathbb{R}^{n^2 \times n^2}$, where the $(i + n(j-1))$'th row of $\bar{A}$ is equal to $vec((A_1)_{i,j} A_2)$, and also vectorize the matrix $U_i \in \mathbb{R}^{n \times n}$ and $V_i \in \mathbb{R}^{n \times n}$ to obtain vectors $u_i \in \mathbb{R}^{n^2}, v_i \in \mathbb{R}^{n^2}$. Therefore, for any entry-wise

norm ξ we have

$$\left\|\sum_{i=1}^{k} U_i \otimes V_i - A\right\|_{\xi} = \left\|\sum_{i=1}^{k} u_i v_i^{\top} - \overline{A}\right\|_{\xi}$$

Lemma 26.11.1 (Reshaping for Low Rank Approximation)**.** *There is a one-to-one mapping* $\pi : [n] \times [n] \times [n] \times [n] \to [n^2] \times [n^2]$ *such that for any pairs* $(U, u) \in \mathbb{R}^{n \times n} \times \mathbb{R}^{n^2}$ *and* $(V, v) \in \mathbb{R}^{n \times n} \times \mathbb{R}^{n^2}$*, if* $U_{i_1, j_1} = u_{i_1 + n(j_1 - 1)}$ *and* $V_{i_1, j_1} = v_{i_1 + n(j_1 - 1)}$*, then we have for* i_1, i_2, j_1, j_2

$$(U \otimes V)_{i_1 + n(i_2 - 1), j_1 + n(j_2 - 1)} = (u \cdot v^{\top})_{\pi(i_1, i_2, j_1, j_2)}$$

where $U \otimes V \in \mathbb{R}^{n^2 \times n^2}$ *and* $uv^{\top} \in \mathbb{R}^{n^2 \times n^2}$*.*

Proof. We have

$$\begin{aligned}(U \otimes V)_{i_1 + n(i_2 - 1), j_1 + n(j_2 - 1)} &= U_{i_1, j_1} V_{i_2, j_2} \\ &= u_{i_1 + n(j_1 - 1)} \cdot v_{i_2 + n(j_2 - 1)} \\ &= (uv^{\top})_{i_1 + n(j_1 - 1), i_2 + n(j_2 - 1)}\end{aligned}$$

where the first step follows from the definition of $\otimes$ product, the second step follows from the connection between U, V and u, v, the last step follows from the outer product. $\square$

Therefore, instead of using trank to define low-rank approximation of the $\otimes$ product of two matrices, we can just use the standard notion of rank to define it since both B and A' can be rearranged to have rank k.

Definition 26.11.1 (Based on Standard Notion of Rank)**.** Given two matrices $A_1, A_2 \times \mathbb{R}^{n \times n}$, let $\overline{A} \in \mathbb{R}^{n^2 \times n^2}$ denote the re-shaping of $A_1 \otimes A_2$. The goal is to output a rank-k matrix $\overline{B}$ such that

$$\|\overline{B} - \overline{A}\|_{\xi} \leq \alpha \, \mathrm{OPT}_{\xi, k}$$

where $\mathrm{OPT}_{\xi, k} = \min_{\mathrm{rank}-k \ \overline{A}'} \|\overline{A}' - \overline{A}\|_{\xi}$.

In other words, $\overline{B}$ can be written as $\overline{B} = \sum_{i=1}^{k} u_i v_i^\top$ where u_i, v_i are length n^2 vectors.

Combining the low-rank reshaping Lemma 26.11.1 with the main input-sparsity low-rank approximation of [CW13], we obtain our Frobenius norm low rank approximation result.

Theorem 26.11.2 (Frobenius norm low rank approximation, $p = 2$)**.** *For any $\epsilon \in (0, 1/2)$, there is an algorithm that runs in $n^2 \operatorname{poly}(k/\epsilon)$ and outputs a rank-k matrix $\overline{B}$ such that $\|\overline{B} - \overline{A}\|_F \leq (1+\epsilon) \operatorname{OPT}_{F,k}$ holds with probability at least $9/10$, where OPT_p is cost achieved by best rank-k solution under the ℓ_p-norm.*

Similarly, using the main ℓ_p low rank approximation algorithm of [SWZ17], we have

Theorem 26.11.3 (Entry-wise ℓ_p-norm low rank approximation, $1 \leq p \leq 2$)**.** *There is an algorithm that runs in $n^2 \operatorname{poly}(k)$ and outputs a rank-k matrix $\overline{B}$ such that $\|\overline{B} - \overline{A}\|_p \leq \operatorname{poly}(k \log n) \operatorname{OPT}_{p,k}$ holds with probability at least $9/10$, where OPT_p is cost achieved by best rank-k solution under the ℓ_p-norm.*

Applying the bi-criteria algorithm of [CGK+17c] gives us:

Theorem 26.11.4 (General $p > 1$, bicriteria algorithm)**.** *There is an algorithm that runs in $\operatorname{poly}(n, k)$ and outputs a rank-$\operatorname{poly}(k \log n)$ matrix $\overline{B}$ such that $\|\overline{B} - \overline{A}\|_p \leq \operatorname{poly}(k \log n) \operatorname{OPT}_p$ holds with probability at least $9/10$, where $\operatorname{OPT}_{p,k}$ is cost achieved by best rank-k solution under the ℓ_p-norm.*

Finally using the low-rank approximation algorithm for general loss functions given in [SWZ18], we obtain a very general result. The parameters for the loss function described in the following theorem are discussed in Section 26.12.

Theorem 26.11.5 (General loss function g)**.** *For any function g that satisfies Definition 26.12.1, 26.12.2, 26.12.3, there is an algorithm that runs in $O(n^2 \cdot T_{\mathrm{reg},g,n^2,k,n^2})$ time and outputs a* rank-$O(k \log n)$ *matrix $\overline{B} \in \mathbb{R}^{n^2 \times n^2}$ such that*

$$\|\overline{B} - \overline{A}\|_g \leq \mathrm{ati}_{g,k} \cdot \mathrm{mon}_g \cdot \mathrm{reg}_{g,k} \cdot O(k \log k) \cdot \mathrm{OPT}_{g,k},$$

holds with probability $1 - 1/\operatorname{poly}(n)$.

Hence, overall, the strategy is to first reshape $A = A_1 \otimes A_2$ into $\bar{A}$, then compute $\bar{B} = \sum_{i=1}^k u_i v_i^\top$ using any of the above three theorems depending on the desired norm, and finally reshape u_i and v_i back to $U_i \in \mathbb{R}^{n \times n}$ and $V_i \in \mathbb{R}^{n \times n}$. It is easy to verify that the guarantees from Theorems 26.11.5, 26.11.3, 26.11.2 are directly transferable to the guarantee of the trank $-k$ approximation shown in Eq 26.1.

26.12 Properties for General Loss Functions for Low trank Approximation

We re-state three general properties (defined in [SWZ18]), the first two of which are structural properties and are necessary and sufficient for obtaining a good approximation from a small subset of columns. The third property is needed for efficient running time.

Definition 26.12.1 (Approximate triangle inequality)**.** For any positive integer n, we say a function $g(x) : \mathbb{R} \to \mathbb{R}_{\geq 0}$ satisfies the $\mathrm{ati}_{g,n}$-approximate triangle inequality if for any $x_1, x_2, \cdots, x_n \in \mathbb{R}$ we have

$$g\left(\sum_{i=1}^n x_i\right) \leq \mathrm{ati}_{g,n} \cdot \sum_{i=1}^n g(x_i).$$

Definition 26.12.2 (Monotone property)**.** For any parameter $\mathrm{mon}_g \geq 1$, we say function $g(x) : \mathbb{R} \to \mathbb{R}_{\geq 0}$ is mon_g-monotone if for any $x, y \in \mathbb{R}$ with $0 \leq |x| \leq |y|$, we have $g(x) \leq \mathrm{mon}_g \cdot g(y)$.

Definition 26.12.3 (Regression property)**.** We say function $g(x) : \mathbb{R} \to \mathbb{R}_{\geq 0}$ has the $(\mathrm{reg}_{g,d}, T_{\mathrm{reg},g,n,d,m})$-regression property if the following holds: given two matrices $A \in \mathbb{R}^{n \times d}$ and $B \in \mathbb{R}^{n \times m}$, for each $i \in [m]$, let OPT_i denote $\min_{x \in \mathbb{R}^d} \|Ax - B_i\|_g$. There is an algorithm that runs in $T_{\mathrm{reg},g,n,d,m}$ time and outputs a matrix $X' \in \mathbb{R}^{d \times m}$ such that

$$\|AX'_i - B\|_g \leq \mathrm{reg}_{g,d} \cdot \mathrm{OPT}_i, \forall i \in [m]$$

and outputs a vector $v \in \mathbb{R}^d$ such that

$$\mathrm{OPT}_i \leq v_i \leq \mathrm{reg}_{g,d} \cdot \mathrm{OPT}_i, \forall i \in [m].$$

The success probability is at least $1 - 1/\operatorname{poly}(nm)$.

Algorithm 26.4 Algorithm to ℓ_p sample $\Theta(r_2)$ entires of $\rho = (A_1 \otimes \cdots \otimes A_q)x' - b$

procedure RESIDUAL ℓ_p SAMPLE(ρ, r_2)
  $r_3 \leftarrow \Theta(r_2 \log^{q^2}(n)/\delta)$.
  Generate i.i.d. p-stable vectors $Z^{1,j}, Z^{2,j}, \ldots, Z^{q,j} \in \mathbb{R}^n$ for $j \in [\tau]$ for $\tau = \Theta(\log(n))$
  $T \leftarrow \emptyset$ ▷ sample set to return
  Pre-compute and store $Z^{i,j}A_i \in \mathbb{R}^{1 \times d_i}$ for all $i \in [q]$ and $j \in [\tau]$
  Generate count-sketches for heavy hitters $S^i \in \mathbb{R}^{k \times n_i}$ of Definition 26.8.2 for all $i \in [q]$, where $k = O(\log^2(n) r_3^{O(1)})$.
  for $t = 1, 2, \ldots, r_3$ **do**
    $s = (s_1, \ldots, s_q) \leftarrow (\emptyset, \ldots, \emptyset)$ ▷ next sample to return
    $w^j \leftarrow \left((\mathbb{I}_{n_1}) \otimes (\bigotimes_{k=2}^q Z^{k,j})\rho\right) \in \mathbb{R}^{n_1}$ ▷ $\mathbb{I}_n \in \mathbb{R}^{n \times n}$ is identity
    Define $w \in \mathbb{R}^{n_1}$ by $w_l = \text{median}_{j \in [\tau]}\{|w_l^j|\}$ for $l \in [n_1]$
    Sample $j^* \in [n_1]$ from the distribution $\left(\frac{|w_1|^p}{\|w\|_p^p}, \frac{|w_2|^p}{\|w\|_p^p}, \ldots, \frac{|w_{n_1}|^p}{\|w\|_p^p}\right)$
    $s_1 \leftarrow j^*$
    for $i = 2, \ldots, q$ **do**
      for $j \in [\tau]$ **do**
        Write $e_{a_k}^\top \in \mathbb{R}^{1 \times n_k}$ as the standard basis vector
        $v_i^j \leftarrow S^i\left((\bigotimes_{k=1}^{i-1} e_{a_k}^\top) \otimes (\mathbb{I}_{n_i}) \otimes (\bigotimes_{k=i+1}^q Z^{k,j})\rho\right) \in \mathbb{R}^k$
        Compute heavy hitters $H_{i,j} \subset [n_i]$ from v_i^j ▷ Definition 26.8.2
        $\beta_i^j \leftarrow \left((\bigotimes_{k=1}^{i-1} e_{a_k}^\top) \otimes (\bigotimes_{k=i}^q Z^{k,j})\rho\right) \in \mathbb{R}$
      end for
      Define $\beta_i \in \mathbb{R}^{k'}$ by $\beta_i = \text{median}_{j \in [\tau]}\{|\beta_i^j|^p\}$
      $H_i = \cup_{j=1}^\tau H_{i,j}$
      $\gamma_i \leftarrow \text{median}_{j \in [\tau]}\left((\bigotimes_{k=1}^{i-1} e_{a_k}^\top) \otimes Z^{i,j}_{[n_i] \setminus H_i} \otimes (\bigotimes_{k=i+1}^q Z^{k,j})\rho\right) \in \mathbb{R}$
      if with probability $1 - \gamma_i/\beta_i$ **then**
        Draw $\xi \in H_i$ with probability

$$\frac{\text{median}_{j \in \tau}\left|\left((\bigotimes_{k=1}^{i-1} e_{a_k}^\top) \otimes (e_\xi^\top) \otimes (\bigotimes_{k=i+1}^q Z^{k,j})\rho\right)\right|^p}{\sum_{\xi' \in H_i} \text{median}_{j \in \tau}\left|\left((\bigotimes_{k=1}^{i-1} e_{a_k}^\top) \otimes (e_{\xi'}^\top) \otimes (\bigotimes_{k=i+1}^q Z^{k,j})\rho\right)\right|^p}$$

        $s_i \leftarrow \xi$
      else ▷ s_i was not sampled as a heavy hitter
        Randomly partition $[n_i]$ into $\Omega_1^i, \Omega_2^i, \ldots, \Omega_\eta^i$ with $\eta = \Theta(r_3^2)$
        Sample $t \sim [\eta]$ uniformly at random
        for $j \in \Omega_t \setminus H_i$ **do**
          $\theta_j = \text{median}_{l \in [\tau]}\left(|(\bigotimes_{k=1}^{i-1} e_{a_k}^\top) \otimes (e_j^\top) \otimes (\bigotimes_{k=i+1}^q Z^{k,l})\rho|^p\right)$
        end for
        Sample $s_i \leftarrow j^*$ from the distribution $\{\frac{\theta_j}{\sum_{j' \in \Omega_t \setminus H_i} \theta_{j'}}\}_{j \in \Omega_t \setminus H_i}$
      end if
    end for
    $T \leftarrow S \cup s$ where $s = (s_1, \ldots, s_q)$
  end for
  return sample set T

Algorithm 26.5 Algorithm to ℓ_p sample $\Theta(r)$ rows of $M = (F \otimes \mathbf{1} - \mathbf{1} \otimes F)R^{-1}$

procedure ℓ_p SAMPLE($F = [A, b] \in \mathbb{R}^{n \times d}, R^{-1} \in \mathbb{R}^{d+1 \times d+1}, r$)
  Generate a matrix $G \in \mathbb{R}^{d+1 \times \xi}$ of i.i.d. $\mathcal{N}(0, 1/\sqrt{\xi})$ Gaussian random variables, with $\xi = \Theta(\log(n))$
  $Y \leftarrow R^{-1}G \in \mathbb{R}^{d+1 \times \xi}$
  $C \leftarrow (F \otimes \mathbf{1} - \mathbf{1} \otimes F)Y$
  Reshape i-th column $C_{*,i}$ into $(FY_{*,i}\mathbf{1}^\top - \mathbf{1}(Y_{*,i})^\top F^\top) \in \mathbb{R}^{n \times n}$
  Generate $Z \in \mathbb{R}^{t \times n}$ i.i.d. p-stable for $t = \Theta(\log(n))$ ▷ Definition 26.2.1
  For all $(i, l) \in [\xi] \times [n]$, set

$$\sigma_{i,l} \leftarrow \underset{\tau \in [t]}{\text{median}} \left(\frac{\left| (Z(FY_{*,i}\mathbf{1}^\top - \mathbf{1}(Y_{*,i})^\top F^\top)_{\tau,l} \right|^p}{(\text{median}(\mathcal{D}_p))^p} \right)$$

▷ Indyk Estimator [Ind06]
  Set $W^{(i,l)} \leftarrow (FY_{*,i}\mathbf{1}^\top - \mathbf{1}Y_{*,i}^\top F^\top)_{*,l} = FY_{*,i} - \mathbf{1}(FY)_{l,i} \in \mathbb{R}^n$
  for $j = 1, \ldots, \Theta(r)$ **do**
    Sample (i, l) from distribution $\sigma_{i,l} / \left(\sum_{i',l'} \sigma_{i',l'} \right)$.
  end for
  $T \leftarrow$ multi-set of samples (i, l)
  Generate $S_0 \in \mathbb{R}^{k \times n}$ $S \in \mathbb{R}^{k' \times n}$ count-sketches for heavy hitters with $k = r^{O(1)}, k' = k^{O(1)}$. ▷ Definition 26.8.2
  Generate $u_1, \ldots, u_n$ i.i.d. exponential variables.
  $D \leftarrow \texttt{Diag}(1/u_1^{1/p}, \ldots, 1/u_n^{1/p}) \in \mathbb{R}^{n \times n}$.
  for each sample $(i, l) \in T$ **do**
    Compute $S_0 W^{(i,l)}$ and obtain set of heavy hitters $Q_0^{(i,l)} \subset [n]$
    Compute $W_j^{(i,l)}$ exactly for all $j \in Q_0^{(i,l)}$, to obtain true heavy hitters $H^{(i,l)}$.
    Compute

$$\alpha_{i,l} \leftarrow \underset{\tau \in [t]}{\text{median}} \left(\frac{\left| Z_{\tau,*} W^{(i,l)} - \sum_{\zeta \in H^{(i,l)}} Z_{\tau,\zeta} W_\zeta^{(i,l)} \right|^p}{(\text{median}(\mathcal{D}_p))^p} \right)$$

    if With prob $1 - \alpha_{(i,l)}/\sigma_{(i,l)}$, sample a heavy item $j^* \leftarrow j$ **then**
      Sample a heavy item $j^* \leftarrow j$ from the distribution $|W_j^{(i,l)}|^p / \sum_{j \in H_{(i,l)}} |W_j^{(i,l)}|^p$.
      return The row $((l-1)n + j^*)$ ▷ Note that $C_{(l-1)n+j^*,*}$ contains $W_{j^*}^{(i,l)}$
    else
      Randomly partition $[n]$ into $\Omega_1, \Omega_2, \ldots, \Omega_\eta$ with $\eta = \Theta(r^4/\epsilon^4)$.
      Sample $t \sim [\eta]$ uniformly at random.
      Compute $S(DW^{(i,l)})_{\Omega_t \setminus H^{(i,l)}}$, and set $Q^{(i,l)} \subset \Omega_t \setminus H^{(i,l)}$ of heavy hitters.
      $j^* \leftarrow \arg\max_{j \in Q^{(i,l)}} (DW^{(i,l)})_j$
      return The row $((l-1)n + j^*)$ ▷ Note that $C_{(l-1)n+j^*,*}$ contains $W_{j^*}^{(i,l)}$
    end if
  end for
end procedure

Chapter 27

Dynamic k-means Clustering

We consider the k-means clustering problem in the dynamic streaming setting, where points from a discrete Euclidean space $\{1, 2, \ldots, \Delta\}^d$ can be dynamically inserted to or deleted from the dataset. For this problem, we provide a one-pass coreset construction algorithm using space $\widetilde{O}(k \cdot \mathrm{poly}(d, \log \Delta))$, where k is the target number of centers. To our knowledge, this is the first dynamic geometric data stream algorithm for k-means using space polynomial in dimension and nearly optimal (linear) in k.

This part is based upon the following previous publication

- Wei Hu, Zhao Song, Lin F. Yang, Peilin Zhong
  Nearly Optimal Dynamic k-Means Clustering for High-Dimensional Data.
  Manuscript 2018 [HSYZ18]

27.1 Introduction

Clustering is one of the central problems in unsupervised learning. The idea is to partition data points into clusters in the hope that points in the same cluster are similar to each other and points in different clusters are dissimilar. One of the most important approaches to clustering is *k-means*, which has been extensively studied for more than 60 years and has a wide range of applications (see e.g. [Jai10] for a survey). Given a set of points $Q \subset \mathbb{R}^d$, the k-means problem asks for a set of k *centers* $Z \subset \mathbb{R}^d$ such that the sum of squares of distances between data points to their closest centers is minimized, i.e., it tries to solve $\min_{Z \subset \mathbb{R}^d, |Z|=k} \text{cost}(Q, Z)$, where $\text{cost}(Q, Z)$ is a cost function defined as:

$$\text{cost}(Q, Z) := \sum_{q \in Q} \min_{z \in Z} \text{dist}^2(q, z).$$

Here $\text{dist}(\cdot, \cdot)$ stands for the Euclidean distance.

A major challenge in dealing with massive datasets is that the entire input data can be too large to be stored. A standard model of study in such settings is the *streaming* model, where data points arrive and are processed one at a time, and only a small amount of useful information (i.e., a *sketch*) about the data is maintained. See e.g. [Mut05b] for an introduction to the streaming model.

In this paper we study the k-means problem over *dynamic data streams* [Ind04], where data points from a discrete space $\{1, 2, \ldots, \Delta\}^d$ can be either inserted to or deleted from the dataset. A standard approach to solving k-clustering problems like k-means and k-median in the streaming setting is to maintain an ϵ-*coreset*, which is a small number of (weighted) points whose cost with respect to any k centers is a $(1 + \epsilon)$-approximation to the cost of the entire dataset on the same k centers. As a consequence, at the end of the stream, we

only need to find an approximate k-means solution on the coreset, which is automatically an approximate solution on the entire dataset. Hence our goal is to design an efficient method to maintain an ϵ-coreset over a dynamic data stream using as small space as possible.

27.1.1 Our Result

Theorem 27.1.1 (Main theorem, restatement of Theorem 27.6.22 in Section 27.6)**.** *Let $\epsilon \in (0, 1/2)$, $k, \Delta \in \mathbb{N}_+$, and $L = \log \Delta$. For dynamic data stream consisting of insertions and deletions of points in $[\Delta]^d$, there is an algorithm which uses a single pass over the stream and on termination outputs a weighted set S with a positive weight for each point therein, such that with probability at least 0.9, S is an ϵ-coreset for k-means of size $O(k\epsilon^{-2}d^4L^2 \log(kdL))$. The algorithm uses $\widetilde{O}(k) \cdot \mathrm{poly}(d, L, \epsilon^{-1})$ bits in the worst case.*

To our knowledge, this is the first algorithm for k-means in dynamic data streams that uses space *polynomial in data dimension d* and *nearly optimal (linear)*[1] *in the number of clusters k*. Previous algorithms for streaming k-means either require space exponential in d or only work for insertion-only streams.[2] See Section 27.1.3 and Section 27.7 for detailed discussions of previous results.

27.1.2 Our Techniques

At a high level our algorithm is based on a framework called *sensitivity sampling*, which was proposed by [FL11]. For a set $Q \subseteq [\Delta]^d$, the *sensitivity* of every point $q \in Q$ is

[1] It is easy to see that k points are needed in a coreset – when there are only k points in the dataset, the optimal k-means cost is 0, so a coreset has to contain all k points.

[2] It is also possible to obtain an $\widetilde{O}(k^2 \cdot \mathrm{poly}(d))$ space algorithm for dynamic streams by combining the techniques from [Che09] and [BFL+17].

defined as

$$s(q) := \max_{Z \subset \mathbb{R}^d, |Z|=k} \frac{\mathrm{dist}^2(q, Z)}{\sum_{p \in Q} \mathrm{dist}^2(p, Z)}.$$

Namely, $s(q)$ represents how "sensitive" the cost can be to the removal of point q. A crucial result shown by [FL11, BFL16] is that once we know a good *upper bound* on each point's sensitivity, there is a sampling method to construct an ϵ-coreset. Specifically, if we know an upper bound $s'(q) \geq s(q)$ for each $q \in Q$, we can sample q with probability $s'(q)/(\sum_{p \in Q} s'(p))$. Let R be a set of i.i.d. samples from this procedure with $|R| \geq \widetilde{\Omega}\left(\sum_{q \in Q} s'(q)/\epsilon^2\right)$, and each sample q is assigned a weight $\frac{\sum_{p \in Q} s'(p)}{|R| s'(q)}$. Then with high probability R is an ϵ-coreset for Q. Note that if $\sum_{q \in Q} s'(q) = \widetilde{O}(k \cdot \mathrm{poly}(d))$, then an $\widetilde{O}(k \cdot \mathrm{poly}(d))$-size ϵ-coreset can be constructed in this way. The formal description of this result is given in Theorem 27.2.1.

We give an efficient method to obtain sensitivity upper bounds $s'(\cdot)$ such that: (i) $\sum_{q \in Q} s'(q)$ is small, (ii) we can implement the sensitivity sampling procedure in the dynamic streaming setting. Then we are able to construct a coreset according to the previous paragraph.

The key intuition in our sensitivity estimation is the following. Imagine that there is a small region that is very dense, i.e., it contains a lot of points. Then the sensitivity of every point in that region must be low, because that point can be well represented by other points in the same small region. Therefore, the problem of finding sensitivity upper bound for a point boils down to figuring out the "right" region this point belongs to that can be considered "dense." Intuitively the sensitivity of this point depends on the size of this dense region – the smaller the size, the smaller the sensitivity.

We make this intuition formal by using a hierarchical grid structure similar to [FS05, BFL+17]. This structure is illustrated in Figure 27.1. The top-level (level 0) grid consists of

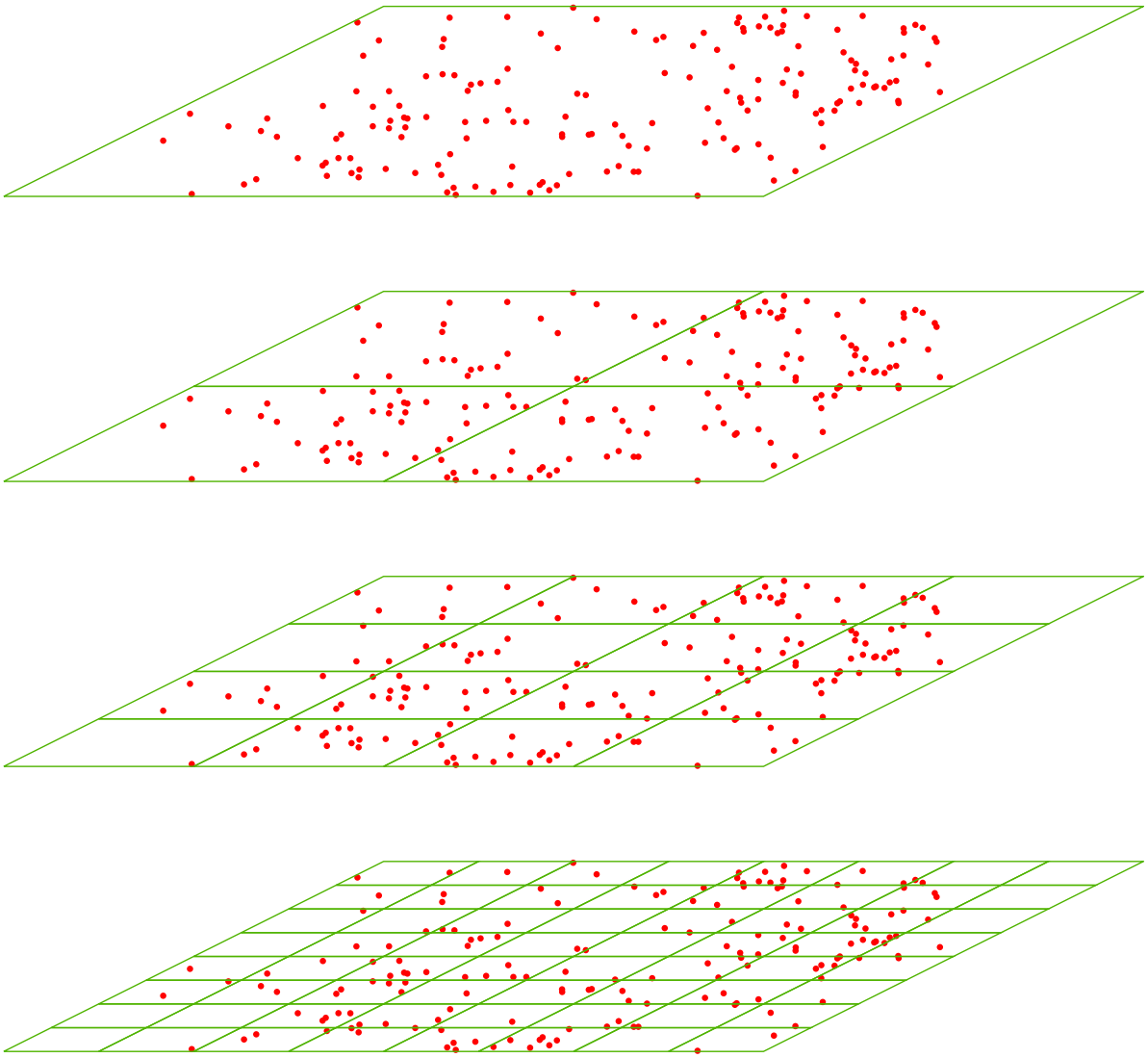


Figure 27.1: The grid structure over the point set. From top to bottom, three levels of grids are shown. Each cell splits into 2^d cells in the next level.

cells that are d-dimensional cubes of side-length Δ, and each cell in level $i-1$ splits into 2^d cells in level i. Each cell in level i has side-length $\Delta/2^i$. For a cell in level i, we say that it is *heavy* if it contains at least $T_i = \Theta\left(\frac{d^2}{k} \cdot \frac{\mathrm{OPT}}{(\Delta/2^i)^2}\right)$ points in Q, where OPT is the optimal cost of the k-means problem.[3] Since $T_i > T_{i-1}$, we know that if a cell in level i is heavy, then its parent cell in level $i-1$ is heavy as well. Therefore the set of all heavy cells in all levels form a tree. Now for a point $p \in Q$, denote by $c_i(p)$ the cell in level i that contains p, and then define j to be the smallest level index such that $c_j(p)$ is not heavy; then we show an upper bound on the sensitivity $s(p)$ solely based on this index number j, namely $s(p) \le s'(p) = \Theta(d^3/T_j)$. Furthermore, we prove that the sum of our sensitivity upper bounds is small: $\sum_{p\in Q} s'(p) = O\left(kd^3 \log \Delta\right)$, which satisfies our requirement. To establish these bounds we need the total number of heavy cells to be small, for which we apply a random shift of grid at the beginning, as illustrated in Figure 27.2.

To implement the above sensitivity sampling method in the dynamic streaming setting when the dataset is updated by insertions and deletions of points, the key difficulties are: 1) we do not know the value of OPT, and it changes when the underlying dataset is updated; 2) we need to compute the sensitivity upper bounds and to sample points at the same time using limited space.

Let us first assume OPT is known and give an algorithm to implement our sensitivity sampling procedure in the dynamic streaming setting. Our algorithm makes crucial use of the *k-set* data structure in [Gan05] for counting *distinct elements* in a dynamic stream. The k-set data structure ensures that if the number of distinct elements is at most some

[3] We assume for now that we know OPT. Our actual algorithm uses exponential search to guess the value of OPT.

predetermined parameter, it will return all distinct elements and their frequencies; otherwise it will return **FAIL**. We summarize its guarantee in Lemma 27.6.1. Note that in order to implement sensitivity sampling, we need to know which cells are heavy. Our algorithm dynamically tracks all heavy cells, using the k-set structure as a building block. Then the sensitivity sampling method has two stages: first sample a level i (with an appropriate probability for each level), and then uniformly sample a point from all points associated with level i, i.e., all points p such that $c_i(p)$ is not heavy and $c_{i-1}(p)$ is heavy. (Note that for all points associated with level i, they have the same sensitivity upper bounds, which means uniformly sampling a point from them is enough.) In order to do uniform sampling, we also maintain for each level i a uniformly random subset of points associated with i. Therefore it suffices to choose a point uniformly at random from this subset once i is chosen.

For the issue of not knowing OPT, we run in parallel multiple copies of our sampling algorithm for different guesses of OPT: $1, 2, 4 \ldots, \Delta^d \cdot d\Delta$. Our sampling algorithm ensures that when the guessed value is less than OPT but not too far away, the required space is small. For other guesses, the required space might be a lot, but since we have a space budget, our algorithm can return **FAIL** when the space runs out. Since at least one guess is accurate, at least one copy of the algorithm will succeed and output a small ϵ-coreset.

27.1.3 Related Work

It is well known that exactly solving k-means is NP-hard even for $k = 2$ or $d = 2$ [ADHP09, MNV09]. The most successful algorithm used in practice is Lloyd's algorithm, which is also known as "the k-means method" [Llo82]. Because of the NP-hardness, various attempts were made on approximation algorithms. [KMN+02] proved that a very simple

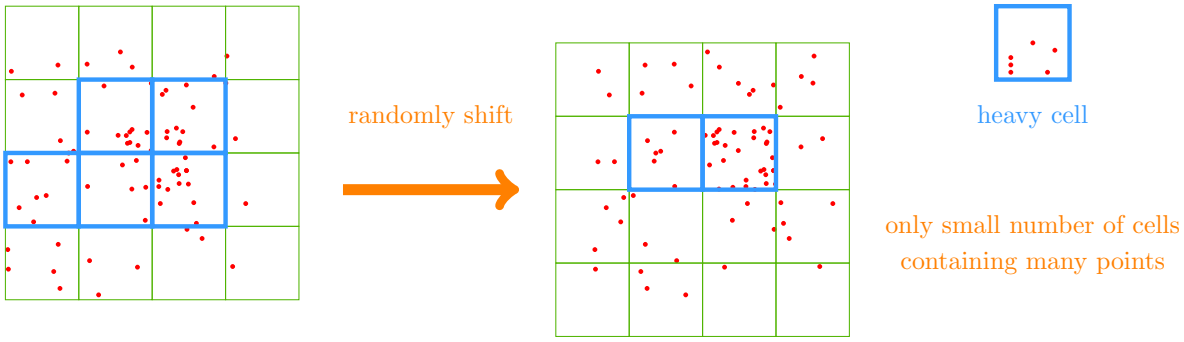


Figure 27.2: Random shift of grid brings down the number of heavy cells. In the left panel, we have a bad alignment of points and grids such that many cells contain lots of points. In the right panel, after the random shift, only two cells contain many points.

local search heuristic achieves $(9+\epsilon)$-approximation in polynomial time for any fixed $\epsilon > 0$. When d is a constant [FRS16, CAKM16] or k is a constant [FMS07, KSS10, FL11], $(1+\epsilon)$-approximation can be achieved in polynomial time.

There is a line of work studying k-means and k-median in insertion-only streams, e.g., [BS80a, GMMO00, COP03, BDMO03, AHPV04, HPM04, HPK05, Che09, FL11, FS12, AMR+12, BFL16]. There also have been a lot of interests in dynamic streaming algorithms for other problems, e.g. [BYJK+02, FKM+05, Bas08, KL11, AGM12a, AGM12b, GKP12, GKK12, AGM12b, BKS12a, CMS13, AGM13, Moi14a, BGS15, BHNT15, BS16, ACD+16, ADK+16, BWZ16, KLM+17, SWZ17, SWZ19b]. In addition, k-means and k-median were studied in various different settings, e.g., [CCGG98, IP11, BIRW16, BCMN14, SW18].

The most relevant papers are [FS05, FL11, BFL16, BFL+17]. [FS05] designed an algorithm to maintain an ϵ-coreset of size $k\epsilon^{-O(d)}$ for k-means and k-median. [FL11] introduced the sensitivity sampling framework for coreset construction, and their approach was further improved by [BFL16], but both of them only work for insertion-only streams and do

not apply to dynamic streams. [BFL+17] focused on the k-median problem and constructed a coreset of size $O(k \cdot \text{poly}(d, \log \Delta))$ in the dynamic streaming setting, but their technique heavily relies on k-median and cannot be extended to k-means. In Section 27.7 we explain in detail the limitations of previous approaches.

27.1.4 Roadmap

In Section 27.2, we introduce necessary notation and preliminaries. In Section 27.3, we give the offline version of our algorithm. Due to the page limit, missing proofs in Section 27.3 are given in Section 27.5, and we present our dynamic streaming algorithm and its analysis in Section 27.6.

27.2 Preliminaries

Notation. For $n \in \mathbb{N}_+$, let $[n] := \{1, 2, \ldots, n\}$. We define $\widetilde{O}(f)$ to be $O\left(f \cdot \log^{O(1)}(f)\right)$. For any $x \in \mathbb{R}_{\geq 0}, \epsilon \in (0,1)$, we use $(1 \pm \epsilon) \cdot x$ to denote the interval $((1-\epsilon) \cdot x, (1+\epsilon) \cdot x)$. For any $x \in \mathbb{R}$ and $a \in \mathbb{R}_{>0}$, $x \pm a$ denotes the interval $(x - a, x + a)$.

We denote by $\mathrm{dist}(\cdot, \cdot)$ the Euclidean distance in $\mathbb{R}^d$, i.e., for $p, q \in \mathbb{R}^d$, $\mathrm{dist}(p, q) := \|p - q\|_2$. For sets $P, Q \subseteq \mathbb{R}^d$ and a point $p \in \mathbb{R}^d$, we define $\mathrm{dist}(p, Q) = \mathrm{dist}(Q, p) := \min_{q \in Q} \mathrm{dist}(p, q)$ and $\mathrm{dist}(P, Q) := \min_{p \in P, q \in Q} \mathrm{dist}(p, q)$. For any two sets $Q, Z \subseteq \mathbb{R}^d$, we define $\mathrm{cost}(Q, Z) := \sum_{q \in Q} \mathrm{dist}^2(q, Z)$. We define $\mathrm{diam}(Q)$ to be Q's diameter, i.e., $\mathrm{diam}(Q) := \max_{p,q \in Q} \mathrm{dist}(p, q)$.

The dynamic streaming model. We consider the *dynamic streaming model*, defined below.

Definition 27.2.1 (Dynamic streaming model)**.** Let $Q \subseteq [\Delta]^d$ initially be an empty set. In the dynamic streaming model, there is a stream of update operations such that the t^{th} operation has the form $(p_t, \pm)$ which indicates that a point $p_t \in [\Delta]^d$ is inserted to or deleted from the set Q, where $+$ denotes insertion and $-$ denotes deletion. There is no invalid deletion during the stream.[4] An algorithm is allowed a single pass over the stream. At the end of the stream, the algorithm stores some information regarding Q. The space complexity of an algorithm in this model is defined as the total number of bits used by the algorithm during the stream.

[4] At any time during the stream, for any point $p \in [\Delta]^d$, the number of deletions of p so far is always no more than the number of insertions of p.

The goal of an algorithm in this model is to store some information which can be used for a certain computation task, while using as small space as possible. Although optimizing the running time is not required in this model, the algorithm in the current paper is actually efficient for each update.

In this paper we suppose that any two points in Q have different locations[5], i.e., Q is not a multiset. Our algorithm can be easily extended to allow multiple copies of a point by blowing up the total space by an $O(\log M)$ factor, where M is an upper bound on the number of copies.

k-means clustering. Now we introduce the *k-means clustering problem* and the notion of *coreset*.

Definition 27.2.2 (k-means clustering)**.** Given a point set $Q \subseteq [\Delta]^d$ and a parameter $k \in \mathbb{N}_+$ for the target number of centers, the goal of k-means clustering is to find a set of k points $Z \subseteq \mathbb{R}^d$ such that the objective function, $\mathrm{cost}(Q, Z) := \sum_{q \in Q} \mathrm{dist}^2(q, Z)$, is minimized. Each point in Z is called a *center*. OPT is defined to be the optimal cost of the k-means clustering problem.

However, solving the k-means problem exactly is NP-hard [ADHP09]. Oftentimes, we only need a good approximation. For the purpose of finding an approximate solution, an important concept is *coreset*, which is a small subset of (weighted) points whose k-means solution is a good approximate solution for the entire dataset. The formal definition is the following:

[5] At the end of the stream, for any point $p \in [\Delta]^d$, the number of insertions of p is at most one more than the number of deletions of p.

Definition 27.2.3 (Coreset for k-means)**.** Given $Q \subseteq [\Delta]^d$, $k \in \mathbb{N}_+$ and $\epsilon > 0$, a set of point-weight pairs $S = \{(s_1, w_1), (s_2, w_2), \ldots, (s_m, w_m)\} \subset [\Delta]^d \times \mathbb{R}_{>0}$ is an *ϵ-coreset* for Q, where w_i is the weight of s_i, if S satisfies

$$\forall Z \subset \mathbb{R}^d, |Z| = k : \left| \operatorname{cost}(S, Z) - \operatorname{cost}(Q, Z) \right| \leq \epsilon \cdot \operatorname{cost}(Q, Z),$$

where $\operatorname{cost}(S, Z) := \sum_{i=1}^m w_i \operatorname{dist}^2(s_i, Z)$. The size of the coreset is $|S|$.

The main problem studied in this paper is how to construct a small coreset for k-means over a dynamic data stream. The formal description is the following.

Definition 27.2.4 (Coreset for k-means over a dynamic stream)**.** Given a point set $Q \subseteq [\Delta]^d$ described by a dynamic stream of operations (Definition 27.2.1), a parameter $k \in \mathbb{N}_+$ for the target number of centers, and an error parameter $\epsilon \in (0, 0.5)$. The goal is to design an algorithm in the dynamic streaming model which can with probability at least 0.9 output a small size k-means ϵ-coreset (Definition 27.2.3) for Q using as small space as possible.

Sensitivity sampling based coreset construction. Let us briefly review the coreset construction framework proposed by [FL11, BFL16]. Given $Q \subseteq [\Delta]^d$ and $k \in \mathbb{N}_+$, the *sensitivity* of a point $p \in Q$ is defined as:

$$s(p) = \max_{Z \in \mathbb{R}^d, |Z| = k} \frac{\operatorname{dist}^2(p, Z)}{\sum_{q \in Q} \operatorname{dist}^2(q, Z)}.$$

The following theorem gives guarantee of a sensitivity sampling based coreset construction.

Theorem 27.2.1 ([FL11, BFL16])**.** *Given a set of points $Q \subseteq [\Delta]^d$ and a parameter k, let $s(p)$ denote the sensitivity of each point $p \in Q$. For each $p \in Q$, let $s'(p)$ be an upper*

bound on the sensitivity of p, i.e., $s'(p) \geq s(p)$, and let $t' = \sum_{p \in Q} s'(p)$. Consider a multiset S of m i.i.d. samples from Q, where each sample chooses $p \in Q$ with probability $s'(p)/t'$. For each sampled point p, a weight $w(p) \in (1 \pm \epsilon/2) \cdot t'/(ms'(p))$ is associated with p. If $m \geq \Omega(t'\epsilon^{-2}(\log |Q| \log t' + \log(1/\delta)))$, then with probability at least $1 - \delta$, $\{(p, w(p)) \mid p \in S\}$ is an ϵ-coreset (Definition 27.2.3) for Q.

According to the above theorem, if we can find a good sensitivity upper bound $s'(p)$ for each point p, then we are able to construct a coreset with size nearly linear in $t' = \sum_p s'(p)$. In section 27.3, we give an offline algorithm which can estimate a good sensitivity upper bound for each point, which readily implies an efficient offline coreset construction algorithm. In Section 27.6, we show how to implement this sensitivity sampling procedure over a dynamic stream. Notice that [BFL16] gave a sensitivity sampling framework that works for clustering with general loss functions, and our method can be extended to those problems as well.

27.3 An Offline Sensitivity Sampling Procedure

In this section, we consider the *offline* setting in which all the data points are given. In this setting, we design a coreset construction algorithm based on sensitivity sampling. In Section 27.6, we will show how to implement this algorithm in the dynamic streaming setting.

27.3.1 Randomly Shifted Grids

We consider data points from $[\Delta]^d$ and assume without loss of generality that $\Delta = 2^L$ for some positive integer L. The space $[\Delta]^d$ is partitioned by a hierarchical grid structure as follows. The first level (level 0) of the grid contains *cells* with side-length Δ such that all the data points are contained in a single cell. For each higher level, we refine the grid by splitting each cell into 2^d equal sized sub-cells. In the finest level, i.e., the L-th level, each cell contains a single point. We further randomly shift the boundary of the grids to achieve certain properties, which we will show later. Formally, our grid structure is defined as the following.

Definition 27.3.1 (Grids and cells)**.** Let $g_0 = \Delta$. Choose a vector v uniformly at random from $[0, \Delta]^d$. Partition the space $\mathbb{R}^d$ into a regular Cartesian grid G_0 with side-length g_0 and translate G_0 such that a vertex of this grid falls on v. The grid G_0 can be regarded as an infinite set of disjoint *cells*, where each cell $C \in G_0$ can be expressed as

$$[v_1 + n_1 g_0,\ v_1 + (n_1 + 1)g_0) \times \cdots \times [v_d + n_d g_0,\ v_d + (n_d + 1)g_0) \subset \mathbb{R}^d$$

for some $(n_1, n_2, \ldots, n_d) \in \mathbb{Z}^d$. (Note that each cell is a Cartesian product of intervals.)

For $i \geq 1$, we define the regular grid G_i as the grid with side-length $g_i = g_0/2^i$ aligned such that each cell in G_{i-1} contains 2^d cells in G_i. The finest grid is G_L where $L = \log_2 \Delta$. A cell of G_L has side-length 1 and thus contains at most one data point.

For convenience, we also define G_{-1} to be the regular grid with side-length $g_{-1} = 2\Delta$, and each cell in G_{-1} is a union of 2^d cells in G_0. Since the data points are in $[\Delta]^d$, there must be a single cell in G_{-1} which contains all the data points. Consider two cells $C \in G_i$ and $C' \in G_j$ for some $i, j \in \{-1, 0, 1, \ldots, L\}$. If $C' \subset C$, then C is an *ancestor* of C'. Furthermore, if $i = j - 1$, then C is the *parent* of C' and C' is a *child* of C. Thus every cell which is not from G_L has exactly 2^d children cells. For a point p (or a set P of points), $c_i(p)$ (or $c_i(P)$) denotes the cell C in grid G_i which contains p (or P). If i is clear from the context, we will just use $c(p)$ (or $c(P)$) for short.

27.3.2 Sensitivity Estimation and Coreset Construction

In Algorithm 27.1 we describe how to assign a sensitivity upper bound for every point. It needs an estimate o of the optimal k-means cost OPT. We will show how to enumerate the guesses o later. According to Theorem 27.2.1, it directly gives an offline coreset construction algorithm.

We also give an alternative sampling procedure in Algorithm 27.2 which is useful for the dynamic streaming model.

Theorem 27.3.1. *Suppose that for any $i \in \{0, 1, \ldots, L\}$ and for any cell $C \in G_i$ with $C \cap Q \neq \emptyset$, the estimated value z in line 27.1 of Algorithm 27.1 satisfies either $z \in |C \cap Q| \pm 0.1T_i(o)$ or $z \in (1 \pm 0.01) \cdot |C \cap Q|$, and for any Q_i, the estimated value $\widehat{q_i}$ in line 27.2*

Algorithm 27.1 Sensitivity Estimation

predetermined: a guess $o \in [1, \Delta^d \cdot d\Delta]$ of the optimal k-means cost OPT
input: a point set $Q \subseteq [\Delta]^d$, a parameter $k \in \mathbb{N}_+$
Impose randomly shifted grids $G_{-1}, G_0, G_1, \ldots, G_L$ (Definition 27.3.1).
Let $C \in G_{-1}$ be the cell which contains Q, i.e., $C = c(Q)$. Mark C as *heavy*.
for $i := 0 \to L - 1$ **do**
    Set the threshold value $T_i(o) = (d/g_i)^2 \cdot o/k \cdot 1/100$.
    for $C \in G_i$ with $C \cap Q \neq \emptyset$ **do**
        Let z be an estimated value of $|C \cap Q|$ up to some precision.
        If $z \geq T_i(o)$, mark C as heavy.
        Otherwise, if all the ancestors of C are marked as heavy, mark C as *crucial*.
    end for
end for
For $C \in G_L$, if all the ancestors of C are marked as heavy, mark C as crucial.
Initialize $Q_0 = Q_1 = \cdots = Q_L = \emptyset$.
For $p \in Q$, if $c_i(p)$ is marked as crucial, add p into set Q_i and set $s'(p) = 10d^3/T_i(o)$.
output: $Q_0, Q_1, \ldots, Q_L$ and $s'(\cdot)$

of Algorithm 27.2 satisfies either $\widehat{q_i} \in |Q_i| \pm 0.1\epsilon\gamma T_i(o)$ *or* $\widehat{q_i} \in (1 \pm 0.01\epsilon) \cdot |Q_i|$. *Suppose* $o \in (0, \mathrm{OPT}]$. *Then the set* S *output by Algorithm 27.2 is an* ϵ*-coreset (Definition 27.2.3) for* Q. *Furthermore, with probability at least* 0.93, $|S|$ *is at most*

$$O(k\epsilon^{-2}d^4L^2 \log(kdL) \cdot (\mathrm{OPT}/o + 1)).$$

27.3.3 Analysis

Now we give the proof of Theorem 27.3.1. All the missing proofs in this section are given in Section 27.5. Let us first state some simple facts.

Fact 27.3.2. *The point sets* $Q_0, Q_1, \ldots, Q_L$ *obtained by Algorithm 27.1 form a partition of* Q, *i.e., for all* $p \in Q$, *there is exactly one* $i \in \{0, 1, \ldots, L\}$ *such that* $p \in Q_i$.

Algorithm 27.2 Sensitivity Sampling Based Coreset Construction

predetermined: a guess o of the optimal k-means cost OPT, an error parameter $\epsilon \in (0, 0.5)$
input: a point set $Q \subset [\Delta]^d$, a parameter $k \in \mathbb{N}_+$
Let $Q_0, Q_1, \ldots, Q_L$ and $s'(\cdot)$ be the output of Algorithm 27.1.
Let $\widehat{q}_0, \widehat{q}_1, \ldots, \widehat{q}_L$ be the estimated values of $|Q_0|, |Q_1|, \ldots, |Q_L|$ respectively.
For $i \in \{0, 1, \ldots, L\}$, set $T_i(o) = (d/g_i)^2 \cdot o/k \cdot 1/100$ (same as in Algorithm 27.1).
Set $\gamma = \epsilon/(40^2 L d^3)$.
Let $I = \{i \mid 0 \le i \le L, \widehat{q}_i \ge \gamma T_i(o)\}$.
Let $Q^I = \bigcup_{i \in I} Q_i$. ▷ Only consider the levels with sufficient number of points.
Set $t' = \sum_{i \in I} \widehat{q}_i \cdot 10d^3/T_i(o)$. ▷ Total estimated sensitivities.
Set $m = \Theta(t' \epsilon^{-2} L d \log t')$ and initialize $S = \emptyset$. ▷ m is the total number of samples.
for $j = 1 \to m$ **do**
    Choose a random level $i \in I$ with probability $(\widehat{q}_i \cdot 10d^3/T_i(o))/t'$.
    Uniformly sample a point p from Q_i.
    Add $(p, t'/(ms'(p)))$ to set S.
end for
output: the set S

Fact 27.3.3. *For $C \in G_i$, if C is marked as heavy, then $|C \cap Q| \ge 0.9T_i(o)$; otherwise $|C \cap Q| \le 1.1T_i(o)$. Similarly, if $i \in I$, then $|Q_i| \ge 0.9\gamma T_i(o)$; otherwise $|Q_i| \le 1.1\gamma T_i(o)$.*

Fact 27.3.4. *For Q_i output by Algorithm 27.1, every point $p \in Q_i$ is assigned the same sensitivity upper bound $s'(p) = 10d^3/T_i(o)$.*

In line 27.2 of Algorithm 27.2, we set I to be the set of levels such that there are sufficient number of points in the crucial cells in those levels. The following lemma shows that the point set Q^I (line 27.2 of Algorithm 27.2) is a good representative of the point set Q, i.e., for any set of k centers Z, the k-means cost $\mathrm{cost}(Q, Z)$ is close to the $\mathrm{cost}(Q^I, Z)$.

Lemma 27.3.5. *Let Q^I and ϵ be the same as in Algorithm 27.2. If $o \in (0, \mathrm{OPT}]$, then for*

any $Z \subseteq \mathbb{R}^d$ with $|Z| = k$, we have:

$$\operatorname{cost}(Q^I, Z) \leq \operatorname{cost}(Q, Z) \leq (1 + \epsilon/10) \operatorname{cost}(Q^I, Z).$$

Next, instead of showing $s'(p)$ (output by Algorithm 27.1) is a sensitivity upper bound with respect ot Q, we show that $s'(p)$ is also an sensitivity upper bound with respect to Q^I. This is even stronger since Q^I is a subset of Q and we have:

$$\forall Z \subseteq \mathbb{R}^d : |Z| = k, \ \frac{\operatorname{dist}^2(p, Z)}{\sum_{q \in Q} \operatorname{dist}^2(q, Z)} \leq \frac{\operatorname{dist}^2(p, Z)}{\sum_{q \in Q^I} \operatorname{dist}^2(q, Z)}.$$

Lemma 27.3.6. *Let Q^I be the same as in Algorithm 27.2. If $o \in (0, \mathrm{OPT}]$, then for all $i \in \{0, 1, 2, \ldots, L\}$ and $p \in Q_i$, we have:*

$$\max_{Z \subseteq \mathbb{R}^d : |Z| = k} \frac{\operatorname{dist}^2(p, Z)}{\sum_{q \in Q^I} \operatorname{dist}^2(q, Z)} \leq 10 \frac{d^3}{T_i(o)} = s'(p).$$

Now, we explain the reason of imposing randomly shifted grids. We fix an optimal set $Z^* = \{z_1^*, z_2^*, \ldots, z_k^*\} \subset \mathbb{R}^d$ of k centers for the point set Q, i.e., $\operatorname{cost}(Q, Z^*) = \mathrm{OPT}$. We call a cell $C \in G_i$ a *center cell* if it is close to a center in Z^*, namely $\operatorname{dist}(C, Z^*) \leq g_i/(2d)$. We claim that there will not be too many center cells since we randomly shift the grids. In other words, each center of Z^* is far from the boundary of every gird.

Lemma 27.3.7. *With probability at least 0.94, the total number of center cells is at most $100kL$.*

This lemma is similar to Lemma 2.2 in [BFL+17]. For completeness, we also provide a proof in Section 27.5.

Consider the total estimated sensitivities, i.e., the sum of the sensitivity upper bounds over all the points. Due to Theorem 27.2.1, this sum determines the size of the coreset. We show that if the estimate o of the optimal k-means cost is close to OPT, then the total estimated sensitivities can not be too large.

Lemma 27.3.8. *Suppose the number of center cells is at most* $100kL$. *Let* $Q_0, Q_1, \ldots, Q_L, s'(\cdot)$ *be the output of Algorithm 27.1. Then the total estimated sensitivities satisfies*

$$\sum_{p\in Q} s'(p) \le 4000kLd^3 \cdot (\mathrm{OPT}/o + 1).$$

Since Q^I is a subset of Q, according to the above lemma, we also have $\sum_{p\in Q^I} s'(p) \le 4000d^3Lk \cdot (\mathrm{OPT}/o + 1)$. Now, we are ready to prove Theorem 27.3.1.

Proof of Theorem 27.3.1. By Lemma 27.3.7, with probability at least 0.94, the number of center cells is at most $100kL$. In the following, we condition on this event.

Algorithm 27.2 draws m i.i.d. samples. For each sample, a point $p \in Q_i \subseteq Q^I$ is chosen with probability

$$\frac{\widehat{q}_i/T_i(o)}{\sum_{j\in I}\widehat{q}_j/T_j(o)} \cdot \frac{1}{|Q_i|} = \frac{\frac{\widehat{q}_i}{|Q_i|}\cdot 20\frac{d^3}{T_i(o)}}{\sum_{j\in I}\sum_{p'\in Q_j}\frac{\widehat{q}_j}{|Q_j|}\cdot 20\frac{d^3}{T_j(o)}}.$$

Each sample p is given a weight

$$\frac{t'}{ms'(p)} = \frac{1}{m}\cdot\frac{\sum_{j\in I}\sum_{p'\in Q_j}\frac{\widehat{q}_j}{|Q_j|}\cdot 20\frac{d^3}{T_j(o)}}{\frac{\widehat{q}_i}{|Q_i|}\cdot 20\frac{d^3}{T_i(o)}}\cdot\frac{\widehat{q}_i}{|Q_i|} \in (1\pm\epsilon/4)\cdot\frac{1}{m}\cdot\frac{\sum_{j\in I}\sum_{p'\in Q_j}\frac{\widehat{q}_j}{|Q_j|}\cdot 20\frac{d^3}{T_j(o)}}{\frac{\widehat{q}_i}{|Q_i|}\cdot 20\frac{d^3}{T_i(o)}}.$$

Let $s''(p) = \frac{\widehat{q}_i}{|Q_i|} \cdot 20\frac{d^3}{T_i(o)}$. Since $\widehat{q}_i/|Q_i| \geq 1/2$, we know that $s''(p)$ is still a sensitivity upper bound of p with respect to Q^I by Lemma 27.3.6. According to Algorithm 27.2, we have $t' = \frac{1}{2}\sum_{p\in Q^I} s''(p)$. Therefore, if we set m to be a sufficiently large $\Omega(t'\epsilon^{-2}Ld\log t') = \Omega(t'\epsilon^{-2}(\log(\Delta^d)\log t' + \log(1/0.01)))$, then according to Theorem 27.2.1, S output by Algorithm 27.2 is an $\epsilon/2$-coreset for Q^I with probability at least 0.99. By Lemma 27.3.5, if S is an $\epsilon/2$-coreset for Q^I, then S is also an ϵ-coreset for Q. Thus, the correctness is proved, and the overall success probability is at least 0.93 obtained by a simple union bound. Now let us analyze the size of the coreset S. Since $\forall j \in I, \widehat{q}_j/|Q_j| \leq 2$, we have $\sum_{j\in I}\sum_{p'\in Q_j} s''(p') \leq 2t' \leq 4 \cdot 4000d^3Lk \cdot (\mathrm{OPT}/o + 1)$ by Lemma 27.3.8. Thus, the size of S is $m = O(k\epsilon^{-2}d^4L^2\log(kdL) \cdot (\mathrm{OPT}/o + 1))$. □

27.4 Conclusion

This paper gives the first k-means coreset construction in the dynamic streaming model using space polynomial in the dimension d and nearly optimal (linear) in k. The algorithm is based on sensitivity sampling, which we believe is a powerful tool and can have broader applications.

27.5 Missing Details in Section 27.3

Proof of Fact 27.3.2. Consider an arbitrary point $p \in Q$. Let $C_{-1}, C_0, \ldots, C_L$ be the cells which contain p, where $\forall i \in \{-1, 0, 1, \ldots, L\}$, the cell C_i is from the grid G_i. According to Algorithm 27.1, C_{-1} is marked as heavy and C_L cannot be marked as heavy. Let l be the largest integer such that all the cells $C_{-1}, C_0, \ldots, C_{l-1}$ are marked as heavy. Then the cell C_l must be marked as crucial, and all the cells $C_{l+1}, C_{l+2}, \ldots, C_L$ can not be crucial. Thus, we have $p \in Q_l$ and $\forall i \in \{0, 1, \ldots, L\} \setminus \{l\}$, $p \notin Q_i$. □

Facts 27.3.3 and 27.3.4 are obvious from the algorithms, so we omit the proofs.

The following claim is useful in the proofs.

Claim 27.5.1. *Let Q^I and γ be the same as mentioned in Algorithm 27.2. For $i \in \{0, 1, \ldots, L\}$ and any heavy cell $C \in G_{i-1}$, if all the ancestors of C are marked as heavy, then we have $|C \cap Q^I| \geq (1 - 5(L - i)\gamma) \cdot |C \cap Q|$.*

Proof. The proof is by induction. When $i = L$, consider a heavy cell $C \in G_{L-1}$ whose ancestors are also heavy. If there is no such cell C, then the claim holds directly for $i = L$. Otherwise, according to the construction of $Q_0, Q_1, \ldots, Q_L$, $\forall j \in \{0, 1, \ldots, L-1\}$, we have $Q_j \cap C = \emptyset$ and $(Q \cap C) \subseteq Q_L$. Since C is marked as heavy, we know that $|Q_L \cap C| = |Q \cap C| \geq 0.9T_{L-1}(o) \geq 0.9T_L(o)/4$. It implies that $\widehat{q}_L \geq \min(|Q_L| - 0.1\epsilon\gamma T_L(o), (1 - 0.01\epsilon)|Q_L|) \geq \min(|Q \cap C| - 0.1\epsilon\gamma T_L(o), (1 - 0.01\epsilon)|Q \cap C|) \geq \gamma T_L(o)$. Thus, we have $Q_L \subseteq Q^I$ which implies that $|C \cap Q^I| = |C \cap Q_L| = |C \cap Q|$.

Now assume the claim is true for $i + 1, i + 2, \ldots, L$. Consider a heavy cell $C \in G_{i-1}$ whose ancestors are also marked as heavy. If there is no such cell C, the claim holds directly

for i. Now consider the case when C exists. If level $i \in I$, i.e., $Q_i \subseteq Q^I$, we have

$$\begin{aligned}
|C \cap Q^I| &= \sum_{C' \in G_i : C' \text{ is a heavy child of } C} |C' \cap Q^I| + \sum_{C' \in G_i : C' \text{ is a crucial child of } C} |C' \cap Q^I| \\
&= \sum_{C' \in G_i : C' \text{ is a heavy child of } C} |C' \cap Q^I| + |C \cap Q_i| \\
&\geq (1 - 5(L - i - 1)\gamma) \sum_{C' \in G_i : C' \text{ is a heavy child of } C} |C' \cap Q| + |C \cap Q_i| \\
&\geq (1 - 5(L - i - 1)\gamma)|C \cap Q| \\
&\geq (1 - 5(L - i)\gamma)|C \cap Q|.
\end{aligned}$$

If level $i \notin I$, i.e., $Q_i \not\subseteq Q^I$, we have

$$\begin{aligned}
\sum_{C' \in G_i : C' \text{ is a heavy child of } C} |C' \cap Q| &\geq |C \cap Q| - |Q_i| \\
&\geq |C \cap Q| - 1.1\gamma T_i(o) \\
&\geq (1 - 5\gamma)|C \cap Q|.
\end{aligned}$$

Thus,

$$\begin{aligned}
|C \cap Q^I| &\geq \sum_{C' \in G_i : C' \text{ is a heavy child of } C} |C' \cap Q^I| \\
&\geq (1 - 5(L - i - 1)\gamma) \cdot \sum_{C' \in G_i : C' \text{ is a heavy child of } C} |C' \cap Q| \\
&\geq (1 - 5(L - i - 1)\gamma)(1 - 5\gamma)|C \cap Q| \\
&\geq (1 - 5(L - i)\gamma)|C \cap Q|.
\end{aligned}$$

□

Proof of Lemma 27.3.5. Since Q^I is a subset of Q, $\text{cost}(Q^I, Z) \leq \text{cost}(Q, Z)$ is trivial. In the following, we are trying to prove $\text{cost}(Q, Z) \leq (1 + \epsilon/10)\,\text{cost}(Q^I, Z)$.

We consider an level $i \notin I$, i.e., $Q_i \not\subseteq Q^I$. For a point $p \in Q_i$, all the ancestors of $c_i(p)$ must be heavy. By averaging argument, there must exist a point $q \in c_{i-1}(p) \cap Q^I$ such that

$$\begin{aligned}
\operatorname{dist}^2(p, Z) &\leq 2\operatorname{dist}^2(p, q) + 2\operatorname{dist}^2(q, Z) \\
&\leq 2dg_{i-1}^2 + 2\operatorname{dist}^2(q, Z) \\
&\leq 2dg_{i-1}^2 + 2\frac{1}{|c_{i-1}(p) \cap Q^I|} \sum_{q \in c_{i-1}(p) \cap Q^I} \operatorname{dist}^2(q, Z) \qquad (27.1)
\end{aligned}$$

where the first step follows from triangle inequality, the second step follows from definition of the grids, the last step follows from an averaging argument.

According to Claim 27.5.1, we have

$$|c_{i-1}(p) \cap Q^I| \geq \frac{1}{2} T_{i-1}(o).$$

Let $Q^N = Q \setminus Q^I$. We can lower bound $\text{cost}(Q, Z)$ in the following sense,

$$\begin{aligned}
\text{cost}(Q, Z) &= \text{cost}(Q^I, Z) + \text{cost}(Q^N, Z) \\
&= \text{cost}(Q^I, Z) + \sum_{i \notin I} \sum_{p \in Q_i} \text{dist}^2(p, z) \\
&\leq \text{cost}(Q^I, Z) + 2 \sum_{i \notin I} \sum_{p \in Q_i} \left(dg_{i-1}^2 + \frac{1}{|c_{i-1}(p) \cap Q^I|} \sum_{q \in c_{i-1}(p) \cap Q^I} \text{dist}^2(q, Z) \right) \\
&\leq \text{cost}(Q^I, Z) + 2 \sum_{i \notin I} \sum_{p \in Q_i} \left(dg_{i-1}^2 + \frac{2}{T_{i-1}(o)} \sum_{q \in c_{i-1}(p) \cap Q^I} \text{dist}^2(q, Z) \right) \\
&\leq \text{cost}(Q^I, Z) + 2 \sum_{i \notin I} \sum_{p \in Q_i} \left(dg_{i-1}^2 + \frac{2}{T_{i-1}(o)} \sum_{q \in Q^I} \text{dist}^2(q, Z) \right) \\
&\leq \text{cost}(Q^I, Z) + 2 \sum_{i \notin I} 1.1 \gamma T_i(o) \cdot \left(dg_{i-1}^2 + \frac{2}{T_{i-1}(o)} \sum_{q \in Q^I} \text{dist}^2(q, Z) \right) \\
&\leq \text{cost}(Q^I, Z) + 5L\gamma T_i(o) \cdot \left(dg_{i-1}^2 + \frac{2}{T_{i-1}(o)} \sum_{q \in Q^I} \text{dist}^2(q, Z) \right) \\
&= \text{cost}(Q^I, Z) + 5L\gamma T_i(o) \cdot \left(dg_{i-1}^2 + \frac{2}{T_{i-1}(o)} \text{cost}(Q^I, Z) \right) \\
&\leq \text{cost}(Q^I, Z) + 5L\gamma (d^3 o/(25k) + 8 \, \text{cost}(Q^I, Z)) \\
&\leq \text{cost}(Q^I, Z) + 5L\gamma (d^3 \, \text{cost}(Q, Z)/(25k) + 8 \, \text{cost}(Q^I, Z)).
\end{aligned}$$

where the second step follows from the definition of the cost, the third step follows from Eq. (27.1), the fourth step follows from $|c_{i-1}(p) \cap Q^I| \geq T_{i-1}(o)/2$, the fifth step follows from $(c_{i-1}(p) \cap Q^I) \subset Q^I$, the sixth step follows from $|Q_i| \leq 1.1\gamma T_i(o)$, the seventh step follows from $L + 1 - |I| \leq L + 1 \leq 2L$ and $2 \cdot 2 \cdot 1.1 \leq 5$, the ninth step follows from $T_i(o) = 4T_{i-1}(o)$ and $T_i(o) = (d/g_i)^2 \cdot o/k \cdot 1/100$, and the last step follows from $o \leq \text{OPT} \leq \text{cost}(Q, Z)$.

It implies that

$$\begin{aligned}\frac{\text{cost}(Q,Z)}{\text{cost}(Q^I,Z)} &\leq \frac{1+40L\gamma}{1-5L\gamma d^3/(25k)}\\ &\leq \frac{1+\epsilon/40}{1-\epsilon/40}\\ &\leq 1+\epsilon/10,\end{aligned}$$

where the second step follows from $\gamma \leq \epsilon/(40^2 L d^3)$, and the last step follows from $\epsilon < 1$. □

Proof of Lemma 27.3.6. Let $Z \subseteq \mathbb{R}^d$ be an arbitrary set of k centers. Fix a point $p \in Q$. Suppose p is in Q_i, i.e., p is in a crucial cell of G_i. Let $C = c_{i-1}(p)$, i.e., C is the parent cell of the crucial cell that contains p. By Algorithm 27.1, C and all of its ancestors must be heavy. By Claim 27.5.1, $C \cap Q^I$ cannot be empty. Thus, by an averaging argument, there is a point $p' \in C$ such that

$$\text{dist}^2(p', Z) \leq \frac{1}{|C \cap Q^I|} \sum_{q \in C \cap Q^I} \text{dist}^2(q, Z). \tag{27.2}$$

We have

$$\begin{aligned}
\frac{\operatorname{dist}^2(p,Z)}{\sum_{q\in Q^I}\operatorname{dist}^2(q,Z)} &\le 2\frac{\operatorname{dist}^2(p',Z)}{\sum_{q\in Q^I}\operatorname{dist}^2(q,Z)} + 2\frac{\operatorname{dist}^2(p,p')}{\sum_{q\in Q^I}\operatorname{dist}^2(q,Z)} \\
&\le 2\frac{\sum_{q\in C\cap Q^I}\operatorname{dist}^2(q,Z)}{|C\cap Q^I|\sum_{q\in Q^I}\operatorname{dist}^2(q,Z)} + 2\frac{dg_{i-1}^2}{\sum_{q\in Q^I}\operatorname{dist}^2(q,Z)} \\
&\le 2\frac{\sum_{q\in Q^I}\operatorname{dist}^2(q,Z)}{|C\cap Q^I|\sum_{q\in Q^I}\operatorname{dist}^2(q,Z)} + 2\frac{dg_{i-1}^2}{\sum_{q\in Q^I}\operatorname{dist}^2(q,Z)} \\
&= 2\frac{1}{|C\cap Q^I|} + 2\frac{dg_{i-1}^2}{\sum_{q\in Q^I}\operatorname{dist}^2(q,Z)} \\
&\le 2\frac{1}{|C\cap Q^I|} + 4\frac{dg_{i-1}^2}{\mathrm{OPT}} \\
&\le 2\frac{1}{|C\cap Q^I|} + 16\frac{dg_i^2}{\mathrm{OPT}} \\
&\le 2\frac{1}{|C\cap Q^I|} + \frac{d^3o}{T_i(o)k\,\mathrm{OPT}} \\
&\le 9\frac{1}{T_i(o)} + \frac{d^3o}{T_i(o)k\,\mathrm{OPT}} \\
&\le 9\frac{1}{T_i(o)} + \frac{d^3}{T_i(o)} \\
&\le 10\frac{d^3}{T_i(o)}
\end{aligned}$$

where the first step follows from triangle inequality, the second step follows from Eq. (27.2) and $p' \in c_{i-1}(p)$, the fifth step follows from $\sum_{q\in Q^I}\operatorname{dist}^2(q,Z) \ge (1-\epsilon)\operatorname{cost}(Q,Z) \ge \mathrm{OPT}/2$ (Lemma 27.3.5), the sixth step follows from $g_{i-1}^2 \le 4g_i^2$, the seventh step follows from $T_i(o) = \frac{d^2}{g_i^2}\frac{o}{100k}$, the eighth step follows from $T_i(o) = 4T_{i-1}(o) \le 4.5|C\cap Q^I|$ (Claim 27.5.1 and $|C\cap Q| \ge 0.9T_{i-1}(o)$), the ninth step follows from $1/k \le 1, o \le \mathrm{OPT}$. □

Proof of Lemma 27.3.7. Fix an $i \in \{0,1,\ldots,L\}$ and consider the grid G_i. For each optimal center z_j^*, we use $X_{j,\alpha}$ to denote the indicator random variable for the event that the distance

from z_j^* to the boundary in dimension α of the grid G_i is at most $g_i/(2d)$. Since in each dimension, if the center is close to a boundary, it contributes a factor at most 2 to the total number of center cells. It follows that the number of cells that have distance at most $g_i/(2d)$ to z_j^* is at most

$$N = 2^{\sum_{\alpha=1}^{d} X_{j,\alpha}}.$$

We denote $Y_{j,\alpha}$ to be $2^{X_{j,\alpha}}$, then

$$\mathbb{E}[N] = \mathbb{E}\left[\prod_{\alpha=1}^{d} Y_{j,\alpha}\right] = \prod_{\alpha=1}^{d} \mathbb{E}[Y_{j,\alpha}].$$

By using $\Pr[X_{j,\alpha} = 1] \leq (2g_i/(2d))/g_i = 1/d$, we obtain

$$\mathbb{E}[Y_{j,\alpha}] \leq \mathbb{E}[1 + X_{j,\alpha}] = 1 + \mathbb{E}[X_{j,\alpha}] \leq 1 + 1/d.$$

Thus $\mathbb{E}[N] = \prod_{\alpha=1}^{d} \mathbb{E}[Y_{j,\alpha}] \leq (1 + 1/d)^d \leq e$. The expected number of center cells in a single grid is at most $(1 + 1/d)^d k \leq ek \leq 3k$. By linearity of expectation, the expected number of center cells in all grids is at most $ek(L + 1) \leq 6kL$. By Markov's inequality, the probability that we have more than $100kL$ center cells in all grids is at most 0.06. □

Proof of Lemma 27.3.8. We have

$$\begin{aligned}
\sum_{p\in Q} s'(p) &= \sum_{i=0}^{L} \sum_{p\in Q_i} 10\frac{d^3}{T_i(o)} \\
&= 10d^3 \sum_{i=0}^{L} \frac{1}{T_i(o)} \left(\sum_{\text{center cell } C\in G_i} |C\cap Q_i| + \sum_{\text{non-center cell } C\in G_i} |C\cap Q_i| \right) \\
&\le 10d^3 \sum_{i=0}^{L} \frac{1}{T_i(o)} \left(\sum_{\text{center cell } C\in G_i} 1.1T_i(o) + \frac{\mathrm{OPT}}{g_i^2/(2d)^2} \right) \\
&\le 11d^3 \cdot (\#\text{ of center cells}) + 4000d^3Lk \cdot \frac{\mathrm{OPT}}{o} \\
&\le 1100d^3Lk + 4000d^3Lk \cdot \frac{\mathrm{OPT}}{o} \\
&\le 4000d^3Lk \cdot \left(\frac{\mathrm{OPT}}{o} + 1 \right),
\end{aligned}$$

where the first step follows from the definition of $s'(p)$, the third step follows from

1. if $C \in G_i$ and $C \cap Q_i \neq \emptyset$, then $|C \cap Q_i| = |C \cap Q| \le 1.1T_i(o)$;
2. if p is in a non-center cell $C \in G_i$, then $\mathrm{dist}^2(p, Z^*) \ge g_i^2/(2d)^2$,

the forth step follows from $g_i^2/d^2 = o/(100kT_i(o))$, the fifth step follows from that the total number of center cells is bounded by $100kL$. □

27.6 Coreset Construction over a Dynamic Stream

In this section, we show how to implement Algorithms 27.1 and 27.2 in the dynamic streaming setting.

First, we introduce a dynamic storage structure that allows us to insert and delete points or cells. We then use this data structure combined with hash functions to estimate the number of points falling into each cell. Lastly, we combine them with the sensitivity sampling procedure to obtain our final algorithm.

Algorithm 27.3 Point-cell storing procedure

1: STORING$(G_i, \alpha, \beta, \delta)$:
2: **Input:** $\{((p_1, l_1), \pm), ((p_2, l_2), \pm), \ldots\}$ ▷ $l_t \in [\widehat{m}]$. Only Algorithm 27.5 uses the case for $\widehat{m} > 1$.
3: Run DISTINCT$(\alpha, \delta/4)$ on $\{(c_i(p_1), \pm), (c_i(p_2), \pm), \ldots\}$ in parallel.
4: Set $r = \lceil \log(4\alpha/\delta) \rceil$ and $h_1, h_2, \ldots, h_r$, $\forall j \in [r]$, $h_j : G_i \to [2\alpha]$. ▷ h_j is pairwise independent.
5: Run $r \cdot 2\alpha$ copies of DISTINCT$(\beta, \delta/(2\alpha))$ in parallel.
6: ▷ Each copy is indexed by a pair $(j, b) \in [r] \times [2\alpha]$. The (j, b)-th copy is run on the sub-stream $\{((p'_1, l'_1), \pm), ((p'_2, l'_2), \pm), \ldots\}$, where each $((p'_t, l'_t), \pm)$ satisfies $h_j(c_i(p'_t)) = b$.
7: If line 27.3 returns **FAIL**, output **FAIL**; otherwise, let $\mathcal{C}, f : \mathcal{C} \to \mathbb{N}_+$ be the output of line 27.3.
8: ▷ $\mathcal{C} \subset G_i$ contains all the cells found, and $f(C)$ denotes the number of points in C.
9: Initialize $S \leftarrow \emptyset$.
10: **for** $C \in \mathcal{C}$ with $f(C) \le \beta$ **do**
11: Find $j \in [r]$ s.t. $\forall C' \in \mathcal{C}, h_j(C) \neq h_j(C')$ and the $(j, h_j(C))$-th copy in line 27.3 does not **FAIL**.
12: If such j does not exist, output **FAIL**; otherwise $S \leftarrow S \cup S_{j,h_j(C)}$.
13: ▷ Here $S_{j,h_j(C)}$ is the set of distinct points found by the $(j, h_j(C))$-th copy in line 27.3.
14: **end for**
15: **Output:** $\mathcal{C} \subset G_i, f : \mathcal{C} \to \mathbb{N}_+, S = \{(\widetilde{p}_1, \widetilde{l}_1), (\widetilde{p}_2, \widetilde{l}_2), \ldots\}$

27.6.1 The Dynamic Point-Cell Storing Data Structure

We introduce an algorithm that maintains a set of points and cells in a dynamic data stream. Before that, let us recall [Gan05]'s result for finding distinct elements, which we use as a subroutine in our algorithm.

Lemma 27.6.1 (Distinct elements [Gan05])**.** *Given parameters $M \geq 1, N \geq 1, s \geq 1, \delta \in (0, 1/2)$, there is an algorithm* DISTINCT(s, δ) *that requires $O(s(\log M + \log N)\log(s/\delta))$ bits to process a stream of insertion/deletion of data items. For each operation $(i, \pm)$ ($i \in [N]$), the algorithm takes $O(\log(s/\delta))$ time. M is an upper bound of the total frequency of all items during the stream. At the end of the stream, if the number of distinct elements is at most s, with probability at least $1 - \delta$ it returns all the distinct elements and their frequencies. It returns* ***FAIL*** *otherwise.*

We use DISTINCT as our sub-routine. We set the parameter M and N to be sufficiently large in our case, i.e., $M = N = \Delta^{2d}$. In Algorithm 27.3, we describe a method which can with probability at least $1-\delta$ output all the non-empty cells in grid G_i when the total number of non-empty cells is not too large (at most α). Furthermore, if the number of points in a particular cell is not too large (at most β), the algorithm can output all the points in that cell. Notice that Algorithm 27.3 is only a subroutine of our final algorithm and will only work on some sub-stream of the entire data stream.

The following lemma shows the guarantee of Algorithm 27.3.

Lemma 27.6.2. *Given parameters $i \in [L], \alpha, \beta \in \mathbb{N}_+, \delta \in (0, 0.5)$,* STORING$(G_i, \alpha, \beta, \delta)$ *(Algorithm 27.3) uses $O(\alpha\beta dL \cdot \log^2(\alpha\beta/\delta))$ bits to process a stream $\{((p_1, l_1), \pm), ((p_2, l_2), \pm), \ldots\}$ of insertion/deletion operations of data points. At the end of the stream, if the number of*

*non-empty cells in G_i is at most α, then with probability at least $1-\delta$ it returns the set $\mathcal{C}$ of all the non-empty cells, the number of points $f(C)$ in each cell $C \in \mathcal{C}$, and the set S of points in all the non-empty cells that contain at most β points. It returns **FAIL** otherwise.*

Proof. If the number of non-empty cells of G_i at the end of the stream is more than α, then according to line 27.3 of Algorithm 27.3 and Lemma 27.6.1, Algorithm 27.3 must output **FAIL**.

Now consider the case when the total number of non-empty cells is at most α. According to Lemma 27.6.1, with probability at least $1-\delta/4$, line 27.3 of Algorithm 27.3 will return the set $\mathcal{C}$ of all the non-empty cells of G_i and the number of points $f(C)$ for each cell $C \in \mathcal{C}$. For each $C \in \mathcal{C}$ with $|C| \leq \beta$, since $|\mathcal{C}| \leq \alpha$, the probability that $\exists j \in [r]$ such that $\forall C' \in \mathcal{C}, h_j(C') \neq h_j(C)$ is at least $1 - 1/2^r \geq 1 - \delta/(4\alpha)$ and furthermore the probability that the $(j, h_j(C))$-th copy of $\textsc{Distinct}(\beta, \delta/(2\alpha))$ in line 27.3 of Algorithm 27.3 will output all the points in C is at least $1-\delta/(2\alpha)$. By taking union bound, the overall probability that Algorithm 27.3 does not output **FAIL** is at most $1 - \delta/4 - (\delta/(2\alpha) + \delta/(4\alpha)) \cdot \alpha = 1 - \delta$.

According to Lemma 27.6.1, the space needed by line 27.3 of Algorithm 27.3 is $O(\alpha dL \cdot \log(\alpha/\delta))$ bits and the space needed of each copy in line 27.3 of Algorithm 27.3 is $O(\beta dL \cdot \log(\alpha\beta/\delta))$ bits. Thus, the total space needed is at most $O(\alpha\beta dL \cdot \log^2(\alpha\beta/\delta))$. □

27.6.2 Estimating the Number of Points in Each Cell

We use Algorithm 27.3 as a subroutine and design a dynamic streaming algorithm (Algorithm 27.4) that can estimate the number of points in each cell up to some precision. Furthermore, it also estimates the number of points $|Q_i|$ in crucial cells of each level i.

Algorithm 27.4 Estimating the number of points in each cell and in each level

PointsEstimation(o, ϵ, δ):
Input: a point set $Q \subseteq [\Delta]^d$ described by a stream $\{(p_1, \pm), (p_2, \pm), \ldots\}$
for $i \in \{0, 1, \ldots, L\}$, in parallel, **do**
  $T_i(o) \leftarrow (d/g_i)^2 \cdot o/(100k)$, $\alpha \leftarrow 10^{11} kLd \log(1/\delta)$, $\alpha' \leftarrow 10^{16}\epsilon^{-3}kL^2d^4$.
  ▷ $T_i(o)$: threshold for heaviness (Algortihm 27.1); α, α': parameters for Storing.
  Let $h_i : [\Delta]^d \to \{0, 1\}$ be a λ-wise independent hash function.
    ▷ $\lambda = 10\lceil (dL + \log(1/\delta) + 1)\rceil$; $\forall p \in [\Delta]^d, h_i(p) = 1$ w.p. $\min(4 \cdot 10^4 \lambda / T_i(o), 1)$.
  Run Storing$(G_i, \alpha, 1, 0.1\delta/L)$ on a sub-stream $\{((p'_1, 1), \pm), ((p'_2, 1), \pm), \ldots\}$ in parallel.
    ▷ Storing is defined in Algorithm 27.3. Here p'_j satisfies $h_i(p'_j) = 1$.
  If Storing returns **FAIL**, output **FAIL**.
  Let $\gamma \leftarrow \epsilon/(40^2 Ld^3)$. ▷ Threshold for discarding levels
  Let $h'_i : [\Delta]^d \to \{0, 1\}$ be a λ-wise independent hash function.
    ▷ $\forall p \in [\Delta]^d, h'_i(p) = 1$ w.p. $\min(4 \cdot 10^4 \epsilon^{-2}\gamma^{-1}\lambda / T_i(o), 1)$.
  Run Storing$(G_i, \alpha', 1, 0.1\delta/L)$ on sub-stream $\{((p''_1, 1), \pm), ((p''_2, 1), \pm), \ldots\}$ in parallel.
    ▷ Storing is defined in Algorithm 27.3. Here p''_j satisfies $h'_i(p''_j) = 1$.
  If Storing returns **FAIL**, output **FAIL**.
  Let $\mathcal{C}_i, f_i, S_i \leftarrow$ Storing$(G_i, \alpha, 1, 0.1\delta/L)$ in line 27.4. Let $\widehat{f}(C) = f_i(C) \cdot \min\{T_i(o)/(4 \cdot 10^4\lambda), 1\}$.
    ▷ Use $\widehat{f}(C)$ as an estimator for $|C \cap Q|$ and follow Algorithm 27.1 to determine whether C is marked as heavy, crucial or nothing. (And conceptually compute $Q_0, \ldots, Q_L$ for analysis.)
  Let $\mathcal{C}'_i, f'_i, S'_i \leftarrow$ Storing$(G_i, \alpha', 1, 0.1\delta/L)$ in line 27.4.
  Let $\widehat{q}_i = \min\left\{\epsilon^2\gamma T_i(o)/(4 \cdot 10^4\lambda) \cdot \sum_{C \in \mathcal{C}'_i : C \text{ is crucial}} f'_i(C), 1\right\}$.
end for
Output: $\widehat{q}_0, \widehat{q}_1, \ldots, \widehat{q}_L$ and $\widehat{f} : \bigcup_{i=0}^{L} G_i \to \mathbb{R}_+$

Now let us analyze Algorithm 27.4. We need the following high concentration bound in our analysis.

Theorem 27.6.3 ([BR94]). *Let λ be an even integer, and let X be the sum of n λ-wise*

independent random variables taking values in $[0,1]$. *Let* $\mu = \mathbb{E}[X]$ *and* $a > 0$. *Then we have*

$$\Pr\left[|X-\mu| > a\right] \leq 8 \cdot \left(\frac{\lambda\mu + \lambda^2}{a^2}\right)^{\lambda/2}.$$

Lemma 27.6.4 (Samples from each cell)**.** *In* PointsEstimation(o, ϵ, δ) *(Algorithm 27.4), with probability at least* $1 - \delta/10$, $\forall i \in \{0, 1, \ldots, L\}$ *with* $T_i(o) \geq 4 \cdot 10^4\lambda$, $\forall C \in G_i$, *we have either*

$$\sum_{p \in C \cap Q} h_i(p) \in |C \cap Q| \cdot \frac{4 \cdot 10^4\lambda}{T_i(o)} \pm 4 \cdot 10^3\lambda$$

or

$$\sum_{p \in C \cap Q} h_i(p) \in |C \cap Q| \cdot \frac{4 \cdot 10^4\lambda}{T_i(o)} \cdot (1 \pm 0.01).$$

Proof. Consider a cell $C \in G_i$. If $|C \cap Q| \leq T_i(o)$, then $\mu = \mathbb{E}\left[\sum_{p \in C \cap Q} h_i(p)\right] = 4 \cdot 10^4\lambda |C \cap Q|/T_i(o) \leq 4 \cdot 10^4\lambda$. According to Theorem 27.6.3, we have

$$\Pr\left[\left|\sum_{p \in C \cap Q} h_i(p) - \mu\right| > 4 \cdot 10^3\lambda\right] \leq 8 \cdot \left(\frac{4 \cdot 10^4\lambda^2 + \lambda^2}{(4 \cdot 10^3\lambda)^2}\right)^{\lambda/2} \leq 8 \cdot (1/2)^{\lambda/2} \leq (\delta/\Delta^d)^5.$$

If $|C \cap Q| > T_i(o)$, then $\mu = \mathbb{E}\left[\sum_{p \in C \cap Q} h_i(p)\right] = 4 \cdot 10^4\lambda |C \cap Q|/T_i(o) > 4 \cdot 10^4\lambda$. According to Theorem 27.6.3, we have

$$\Pr\left[\left|\sum_{p \in C \cap Q} h_i(p) - \mu\right| > 0.01\mu\right] \leq 8 \cdot \left(\frac{\lambda\mu + \lambda^2}{(0.01\mu)^2}\right)^{\lambda/2} \leq 8 \cdot (1/2)^{\lambda/2} \leq (\delta/\Delta^d)^5.$$

By taking union bound over all $i \in \{0, 1, \ldots, L\}$ and all the cells in G_i, the claim is proved. □

Lemma 27.6.5 (Estimating the number of points in each cell)**.** *If* PointsEstimation(o, ϵ, δ) *(Algorithm 27.4) does not output* ***FAIL****, then with probability at least* $1 - \delta/10$, $\forall i \in \{0, 1, \ldots, L\}, C \in G_i$, *either* $\widehat{f}(C) \in |C \cap Q| \pm 0.1T_i(o)$ *or* $\widehat{f}(C) \in (1 \pm 0.01)|C \cap Q|$.

Proof. Suppose PointsEstimation(o, ϵ, δ) does not output **FAIL**. According to Lemma 27.6.2, $\forall i \in \{0, 1, \ldots, L\}$ and $\forall C \in G_i$, $f_i(C) = \sum_{p \in C \cap Q} h_i(p)$. If $T_i(o) \le 4 \cdot 10^4 \lambda$, then $\widehat{f}(C) = f_i(C) = \sum_{p \in C \cap Q} h_i(p) = |C \cap Q|$.

Due to Lemma 27.6.4, with probability at least $1 - \delta/10$, $\forall i \in \{0, 1, \ldots, L\}$ with $T_i(o) > 4 \cdot 10^4 \lambda$ and $\forall C \in G_i$, either $f_i(C) \in |C \cap Q| \cdot \frac{4 \cdot 10^4 \lambda}{T_i(o)} \pm 4 \cdot 10^3 \lambda$ or $f_i(C) \in |C \cap Q| \cdot \frac{4 \cdot 10^4 \lambda}{T_i(o)} \cdot (1 \pm 0.01)$. Thus, either $\widehat{f}(C) \in |C \cap Q| \pm 0.1 T_i(o)$ or $\widehat{f}(C) \in (1 \pm 0.01)|C \cap Q|$. □

Lemma 27.6.6 (Samples from each Q_i)**.** *In* PointsEstimation(o, ϵ, δ) *(Algorithm 27.4), with probability at least* $1 - \delta/10$, $\forall i \in \{0, 1, \ldots, L\}$ *with* $T_i(o) \ge 4 \cdot 10^4 \epsilon^{-2} \gamma^{-1} \lambda$, $\forall C \in G_i$, *either*

$$\sum_{p \in Q_i} h_i'(p) \in |Q_i| \cdot \frac{4 \cdot 10^4 \epsilon^{-2} \gamma^{-1} \lambda}{T_i(o)} \pm 4 \cdot 10^3 \epsilon^{-1} \lambda$$

or

$$\sum_{p \in Q_i} h_i'(p) \in |Q_i| \cdot \frac{4 \cdot 10^4 \epsilon^{-2} \gamma^{-1} \lambda}{T_i(o)} \cdot (1 \pm 0.01\epsilon).$$

Proof. If $|Q_i| \le \gamma T_i(o)$, then $\mu = \mathbb{E}\left[\sum_{p \in Q_i} h_i'(p)\right] = 4 \cdot 10^4 \epsilon^{-2} \gamma^{-1} \lambda |Q_i| / T_i(o) \le 4 \cdot 10^4 \epsilon^{-2} \lambda$. According to Theorem 27.6.3, we have

$$\Pr\left[\left|\sum_{p \in Q_i} h_i'(p) - \mu\right| > 4 \cdot 10^3 \epsilon^{-1} \lambda\right] \le 8 \cdot \left(\frac{4 \cdot 10^4 \epsilon^{-2} \lambda^2 + \lambda^2}{(4 \cdot 10^3 \epsilon^{-1} \lambda)^2}\right)^{\lambda/2} \le 8 \cdot (1/2)^{\lambda/2} \le (\delta/\Delta^d)^5.$$

If $|Q_i| > \gamma T_i(o)$, then $\mu = \mathbb{E}\left[\sum_{p \in Q_i} h_i'(p)\right] = 4 \cdot 10^4 \epsilon^{-2} \gamma^{-1} \lambda |Q_i| / T_i(o) > 4 \cdot 10^4 \epsilon^{-2} \lambda$. According to Theorem 27.6.3, we have

$$\Pr\left[\left|\sum_{p \in Q_i} h_i'(p) - \mu\right| > 0.01\epsilon\mu\right] \le 8 \cdot \left(\frac{\lambda\mu + \lambda^2}{(0.01\epsilon\mu)^2}\right)^{\lambda/2} \le 8 \cdot (1/2)^{\lambda/2} \le (\delta/\Delta^d)^5.$$

By taking union bound over all $i \in \{0, 1, \ldots, L\}$, the claim is proved. □

Lemma 27.6.7 ($\widehat{q}_i$ can estimate $|Q_i|$ well)**.** *If* PointsEsitmation(o, ϵ, δ) *(Algorithm 27.4) does not output* ***FAIL****, then with probability at least* $1 - \delta/10$*,* $\forall i \in \{0, 1, \ldots, L\}$*, either* $\widehat{q}_i \in |Q_i| \pm 0.1\epsilon\gamma T_i(o)$ *or* $\widehat{q}_i \in (1 \pm 0.01\epsilon)|Q_i|$*.*

Proof. Suppose PointsEstimation(o, ϵ, δ) does not output **FAIL**. According to Lemma 27.6.2, $\forall i \in \{0, 1, \ldots, L\}$, $f_i'(C) = \sum_{p \in Q_i} h_i'(p)$. If $T_i(o) \le 4 \cdot 10^4 \epsilon^{-2}\gamma^{-1}\lambda$, then $\widehat{q}_i = \sum_{C \in G_i : C \text{ is crucial}} f_i'(C) = \sum_{p \in Q_i} h_i'(p) = |Q_i|$.

Due to Lemma 27.6.6, with probability at least $1 - \delta/10$, $\forall i \in \{0, 1, \ldots, L\}$ with $T_i(o) > 4 \cdot 10^4 \epsilon^{-2}\gamma^{-1}\lambda$, either

$$\sum_{C \in G_i : C \text{ is crucial}} f_i'(C) \in |Q_i| \cdot \frac{4 \cdot 10^4 \epsilon^{-2}\gamma^{-1}\lambda}{T_i(o)} \pm 4 \cdot 10^3 \epsilon^{-1}\lambda$$

or

$$\sum_{C \in G_i : C \text{ is crucial}} f_i'(C) \in |Q_i| \cdot \frac{4 \cdot 10^4 \epsilon^{-2}\gamma^{-1}\lambda}{T_i(o)} \cdot (1 \pm 0.01\epsilon).$$

Thus, either $\widehat{q}_i \in |Q_i| \pm 0.1\epsilon\gamma T_i(o)$ or $\widehat{q}_i \in (1 \pm 0.01\epsilon)|Q_i|$. □

Lemma 27.6.8 (Number of points sampled from non-center cells)**.** *In* PointsEsitmation(o, ϵ, δ) *(Algorithm 27.4), with probability at least* $1 - \delta/10$*,* $\forall i \in \{0, 1, \ldots, L\}$*, we have*

$$\sum_{\text{non-center cell } C \in G_i} \sum_{p \in C \cap Q} h_i(p) \le 4 \cdot 10^3 \lambda + 1.01 \cdot \frac{4 \cdot 10^4 \lambda}{T_i(o)} \sum_{\text{non-center cell } C \in G_i} |C \cap Q|$$

and

$$\sum_{\text{non-center cell } C \in G_i} \sum_{p \in C \cap Q} h_i'(p) \le 4 \cdot 10^3 \epsilon^{-1}\lambda + 1.01 \cdot \frac{4 \cdot 10^4 \epsilon^{-2}\gamma^{-1}\lambda}{T_i(o)} \sum_{\text{non-center cell } C \in G_i} |C \cap Q|.$$

Proof. The proof is exactly the same as the proofs of Lemmas 27.6.4 and 27.6.6. □

Lemma 27.6.9. $\forall i \in \{0, 1, \ldots, L\}$,

$$\sum_{\text{non-center cell } C \in G_i} \frac{|C \cap Q|}{T_i(o)} \leq 400k \cdot (\mathrm{OPT}/o).$$

Proof.

$$\begin{aligned}&\sum_{\text{non-center cell } C \in G_i} |C \cap Q| \\ \leq &\frac{\mathrm{OPT}}{(g_i/(2d))^2} = 400kT_i(o) \cdot \frac{\mathrm{OPT}}{o}.\end{aligned}$$

□

Lemma 27.6.10 (The success probability)**.** *Condition on the number of center cells of all the grids is at most* $100kL$*. If* $o \in (\mathrm{OPT}/16, \mathrm{OPT}]$*, then* PointsEstimation(o, ϵ, δ) *(Algorithm 27.4) does not output* ***FAIL*** *with probability at least* $1 - 3\delta/10$*.*

Proof. Let $o \geq \mathrm{OPT}/16$, according to Lemma 27.6.9 and Lemma 27.6.8, with probability at least $1 - \delta/10$, $\forall i \in \{0, 1, \ldots, L\}$, we have

$$\sum_{\text{non-center cell } C \in G_i} \sum_{p \in C \cap Q} h_i(p) \leq 4 \cdot 10^3 \lambda + 1.01 \cdot 4 \cdot 10^4 \lambda \cdot 6400k \leq 10^{11} kdL \log(1/\delta)$$

and

$$\sum_{\text{non-center cell } C \in G_i} \sum_{p \in C \cap Q} h_i'(p) \leq 4 \cdot 10^3 \epsilon^{-1} \lambda + 1.01 \cdot 4 \cdot 10^4 \epsilon^{-2} \gamma^{-1} \lambda \cdot 6400k \leq 10^{15} \cdot \epsilon^{-3} kL^2 d^4.$$

Since the number of center cells is at most $100kL$, the total number of cells which contains some p with $h_i(p) = 1$ is at most $10^{12} kdL \log(1/\delta) \leq \alpha$, and the total number of cells which contains some p with $h_i'(p) = 1$ is at most $10^{16} \epsilon^{-3} kL^2 d^4 \leq \alpha'$. According to Lemma 27.6.2, with probability at least $1 - 2\delta/10$, none the call of Storing will return **FAIL**. Thus the overall probability that Algorithm 27.4 does not output **FAIL** is at last $1 - 3\delta/10$. □

Lemma 27.6.11 (Space of Algorithm 27.4)**.** PointsEstimation(o, ϵ, δ) *(Algorithm 27.4) uses space at most* $O(k\epsilon^{-3}L^4d^5 \cdot \log^2(kLd/(\epsilon\delta)))$ *bits.*

Proof. The total space used is dominated by the space needed to run $L+1$ copies of Storing$(G_i, \alpha', 1, 0.1\delta/L)$ in line 27.4 of Algorithm 27.4. According to Lemma 27.6.2, the total space needed is $O(L \cdot \alpha' dL \log^2(L\alpha'/\delta)) = O(k\epsilon^{-3}L^4d^5 \cdot \log^2(kLd/(\epsilon\delta)))$ bits. □

27.6.3 Sensitivity Sampling over a Dynamic Stream

Since using Algorithm 27.4 we can estimate the number of points in each cell and the size of each Q_i, the only remaining thing for simulating Algorithm 27.2 is to draw samples based on their sensitivity upper bounds. In Algorithm 27.5, we show how to achieve this in a dynamic stream.

Now we analyze Algorithm 27.5.

Fact 27.6.12. *If* Sampling(o, ϵ, δ) *(Algorithm 27.5) does not output* ***FAIL****, then line 27.5 can be implemented, and* p *is a uniform sample drawn from* Q_i*.*

Proof. Although Q_i cannot be stored explicitly, $\forall p \in Q$, we are able to determine whether $p \in Q_i$ since we can use $\widehat{f}$ to find all the crucial cells and check whether p is in a crucial cell of G_i.

Suppose Sampling(o, ϵ, δ) does not output **FAIL**. According to Lemma 27.6.2 and the condition in line 27.5, $\forall j \in [\widehat{m}]$, we have $\{(p, j) \mid p \in Q_i, h_{i,j}(p) = 1\} \subseteq S_i$. Since $\forall x, y \in Q_i$, $\Pr[h_{i,j}(x) = 1] = \Pr[h_{i,j}(y) = 1]$, then the sample p in line 27.5 is drawn uniformly from Q_i. □

Algorithm 27.5 Sensitivity sampling over a dynamic stream

SAMPLING(o, ϵ, δ):
Input: a point set $Q \subseteq [\Delta]^d$ described by a stream $\{(p_1, \pm), (p_2, \pm), \ldots\}$
Run POINTSESTIMATION$(o, \epsilon, \delta/2)$ in parallel on the input stream. If it returns **FAIL**, output **FAIL**.
$\forall i \in \{0, 1, \ldots, L\}$, $T_i(o) \leftarrow (d/g_i)^2 \cdot o/k \cdot 1/100$. ▷ Threshold for heaviness (Algorithm 27.1).
$\widehat{m} \leftarrow \Theta\left(k\epsilon^{-3}L^4d^7 \log\left(\frac{dLk}{\delta}\right) \cdot \frac{1}{\delta}\right)$. ▷ $\widehat{m}$ hash functions needed for m independent samples.
$\lambda \leftarrow 10\lceil(dL + \log(1/\delta) + 1)\rceil$. ▷ λ is the independence parameter.
$\forall i \in \{0, 1, \ldots, L\}$, choose $h_{i,1}, h_{i,2}, \ldots, h_{i,\widehat{m}} : [\Delta]^d \to \{0, 1\}$.
▷ $\forall j \in [\widehat{m}]$, $h_{i,j}$ is λ-wise independent, $\forall p \in [\Delta]^d$, $h_{i,j}(p) = 1$ w.p. $\min\left\{1/(10^4 kLT_i(o)), 1\right\}$.
$\alpha \leftarrow \Theta(k\epsilon^{-3}L^3d^7 \log(dLk/\delta)\delta^{-1})$, $\beta \leftarrow \Theta(\epsilon^{-3}L^3d^7 \log(dLk/\delta)\delta^{-1})$. ▷ α, β: for STORING.
For $i \in \{0, 1, \ldots, L\}$, run STORING$(G_i, \alpha, \beta, 0.1\delta/L)$ (Algorithm 27.3) in parallel.
▷ Each instance is run on a new stream obtained by splitting each operation $(p_t, \pm)$ from the original input stream into a set of new operations $\{((p_t, j), \pm) \mid j \in [\widehat{m}], h_{i,j}(p_t) = 1\}$.
If any STORING returns **FAIL**, output **FAIL**.
$\widehat{q}_0, \widehat{q}_1, \ldots, \widehat{q}_L$, $\widehat{f} : \bigcup_{i=0}^{L} G_i \to \mathbb{R}_+$ $\leftarrow$ POINTSESTIMATION$(o, \epsilon, \delta/2)$ in line 27.5.
$\mathcal{C}_i, f_i, S_i \leftarrow$ STORING$(G_i, \alpha, \beta, 0.1\delta/L)$ in line 27.5.
For $i \in \{0, 1, \ldots, L\}$, if $\exists C \in \mathcal{C}_i$ marked as crucial in line 27.5, and $f_i(C) > \beta$, output **FAIL**.
$\gamma \leftarrow \epsilon/(40^2Ld^3)$. ▷ Threshold for discarding levels.
$I \leftarrow \{i \mid 0 \le i \le L, \widehat{q}_i \ge \gamma T_i(o)\}$.
$t' \leftarrow \sum_{i \in I} \widehat{q}_i \cdot 10d^3/T_i(o)$. ▷ Total estimated sensitivities.
$m \leftarrow \Theta(t'\epsilon^{-2}Ld \log(t'/\delta))$, $S \leftarrow \emptyset$. ▷ Total number of samples.
For $i \in \{0, 1, \ldots, L\}$, set $A_i = [\widehat{m}]$.
for $j = 1 \to m$ **do**
  Choose a random level $i \in I$ with probability $(\widehat{q}_i \cdot 10d^3/T_i(o))/t'$.
  Choose the minimum $j \in A_i$ s.t. $\exists p \in Q_i$, $(p, j) \in S_i$. If no such j, output **FAIL**.
  Uniformly choose a point p from the set $\{q \in Q_i \mid (q, j) \in S_i\}$.
  Update $A_i \leftarrow \{j + 1, j + 2, \ldots, \widehat{m}\}$.
  Add $(p, t'/(ms'(p)))$ to set S. ▷ $s'(p) = 10d^3/T_i(o)$.
end for
Output: the set S

Lemma 27.6.13 (Correctness of Algorithm 27.5)**.** *Suppose $o \in (0, \mathrm{OPT}]$. If* SAMPLING(o, ϵ, δ) *(Algorithm 27.5) does not output **FAIL**, then with probability at least $1 - \delta/5$, the output S by* SAMPLING(o, ϵ, δ) *is an ϵ-coreset for Q. Furthermore, the size $|S|$ is at most $O(k\epsilon^{-2}L^2d^4 \log(kLd) \cdot (\mathrm{OPT}/o + 1))$.*

Proof. Due to Lemma 27.6.5 and Lemma 27.6.7, with probability at least $1-\delta/10$, $\forall C \in \bigcup_{i=0}^{L} G_i$, either $\widehat{f}(C) \in |C \cap Q| \pm 0.1T_i(o)$ or $\widehat{f}(C) \in (1 \pm 0.01)|C \cap Q|$ and $\forall i \in \{0, 1, \ldots, L\}$, either $\widehat{q}_i \in |Q_i| \pm 0.1\epsilon\gamma T_i(o)$ or $\widehat{q}_i \in (1 \pm 0.01\epsilon)|Q_i|$, where $Q_0, Q_1, \ldots, Q_L$ are defined by using the estimation $\widehat{f}(\cdot)$ (Algorithm 27.1 or Algorithm 27.4). According to Fact 27.6.12, the sampling procedure can be implemented. Then by Theorem[6] 27.3.1, with probability at least $1-\delta/10$, the output set S is an ϵ-coreset for Q. By taking union bound, with probability $1-\delta/5$, the set S is an ϵ-coreset for Q. □

Now let us consider the success probability of Algorithm 27.5. Since we know the success probability of PointsEstimation$(o, \epsilon, \delta/2)$ in line 27.5 of Algortihm 27.5 and the success probability of Storing$(G_i, \alpha, \beta, 0.1\delta/L)$ in line 27.5, we only need to analyze the success probability in line 27.5 of Algorithm 27.5. To make line 27.5 succeed, we need to find enough samples from Q_i, i.e., we hope that $\sum_{j=1}^{\widehat{m}} \mathbf{1}(|\{p \in Q_i \mid h_{i,j}(p) = 1\}| > 0)$ is large. In the following analysis, we will show that $\sum_{j=1}^{\widehat{m}} \mathbf{1}(|\{p \in Q_i \mid h_{i,j}(p) = 1\}| > 0)$ is large. First, we show that the number of samples drawn from level i is bounded.

Lemma 27.6.14 (Number of samples from each level)**.** *Let I be the set computed in* Sampling(o, ϵ, δ) *(Algortihm 27.5). With probability at least $1-\delta/10$, $\forall i \in I$, the number of times that i is chosen in line 27.5 of Algorithm 27.5 is at most*

$$O\left(\epsilon^{-2} L d^4 \log(t'/\delta) \cdot \frac{L}{\delta} \cdot \frac{\widehat{q}_i}{T_i(o)}\right).$$

Proof. For $i \in I$, the expected number of times that i is chosen is $O(m \cdot \widehat{q}_i \cdot d^3/T_i(o)/t')$. By Markov's inequality, with probability at least $1-\delta/(20L)$, the number of times that i

[6]The proof is slightly different since Theorem 27.3.1 only claims a constant success probability. See Section 27.3.3 for the detailed proof of Theorem 27.3.1.

is chosen in line 27.5 of Algorithm 27.5 is at most $O\left(\epsilon^{-2} L d^4 \log(t'/\delta) \cdot \frac{L}{\delta} \cdot \frac{\widehat{q}_i}{T_i(o)}\right)$. By taking union bound over all $i \in I$, we complete the proof. □

Lemma 27.6.15 (Bounding t')**.** *Consider $o \geq \mathrm{OPT}/16$. Condition on $\widehat{f} : \bigcup_{i=0}^{L} G_i \to \mathbb{R}_+$ (in Algorithm 27.5) is good (Lemma 27.6.4), $\widehat{q}_0, \widehat{q}_1, \ldots, \widehat{q}_L$ (in Algorithm 27.5) are good (Lemma 27.6.6), and the number of center cells is at most $100kL$, then we have $t' \leq 10^6 d^3 Lk$.*

Proof. We have

$$\begin{aligned}
t' &= \sum_{i \in I} \widehat{q}_i \cdot 10d^3/T_i(o) \\
&\leq \sum_{i \in I} |Q_i| \cdot 20d^3/T_i(o) \\
&\leq \sum_{i=0}^{L} |Q_i| \cdot 20d^3/T_i(o) \\
&\leq 2 \cdot 4000 d^3 Lk \cdot 20 \\
&\leq 10^6 d^3 Lk,
\end{aligned}$$

where the first inequality follows by $\forall i \in I, \widehat{q}_i \geq \gamma T_i(o)$ and either $\widehat{q}_i \in |Q_i| \pm 0.1\epsilon\gamma T_i(o)$ or $\widehat{q}_i \in (1 \pm 0.01\epsilon)|Q_i|$, the second inequality follows by $I \subseteq \{0, 1, \ldots, L\}$, the third inequality follows by Lemma 27.3.8 and $o \geq \mathrm{OPT}/16$. □

Lemma 27.6.16 (Number of crucial points in each level)**.** *Consider $o \geq \mathrm{OPT}/16$. Conditioning on $\widehat{f} : \bigcup_{i=0}^{L} G_i \to \mathbb{R}_+$ (in Algorithm 27.5) is good (Lemma 27.6.5) and the number of center cells is at most $100kL$, we have:*

$$\frac{|Q_i|}{T_i(o)} \leq 7000kL.$$

Proof. We have

$$\begin{aligned}\frac{|Q_i|}{T_i(o)} &= \sum_{\text{center crucial cell } C\in G_i} \frac{|C\cap Q|}{T_i(o)} + \sum_{\text{non-center crucial cell } C\in G_i} \frac{|C\cap Q|}{T_i(o)} \\ &\leq 110kL + 400k(\mathrm{OPT}\,/o) \\ &\leq 7000kL,\end{aligned}$$

where the first inequality follows by that the number of center cells is at most $100kL$, the number points in a crucial cell is at most $1.1T_i(o)$ and Lemma 27.6.9. □

Lemma 27.6.17 (The number of samples is large)**.** *Consider $o \geq \mathrm{OPT}\,/16$. Conditioning on $\widehat{f} : \bigcup_{i=0}^{L} G_i \to \mathbb{R}_+$ (in Algorithm 27.5) is good (Lemma 27.6.4), $\widehat{q}_0, \widehat{q}_1, \ldots, \widehat{q}_L$ (in Algorithm 27.5) are good (Lemma 27.6.6), and the number of center cells is at most $100kL$, with probability at least $1 - \delta/10$, $\forall i \in I$, we have:*

$$\sum_{j=1}^{\widehat{m}} \mathbf{1}(|\{p \in Q_i \mid h_{i,j}(p) = 1\}| > 0) \geq \Omega\left(\epsilon^{-2} Ld^4 \log\left(\frac{dLk}{\delta}\right) \cdot \frac{L}{\delta} \cdot \frac{\widehat{q}_i}{T_i(o)}\right).$$

Proof. Consider a fixed $i \in I$. $\forall j \in [\widehat{m}]$, by union bound, we have

$$\Pr_{h_{i,j}}[\exists p \in Q_i, h_{i,j}(p) = 1] \leq \frac{|Q_i|}{10^4 kLT_i(o)} < 1,$$

where the inequality follows by Lemma 27.6.16. Thus $\lceil 10^4 kLT_i(o)/|Q_i|\rceil \leq 2\cdot 10^4 kLT_i(o)/|Q_i|$. Let $b = 10 \cdot \lceil 10^4 kLT_i(o)/|Q_i|\rceil$. Let $r = \lfloor \widehat{m}/b \rfloor \geq \widehat{m} \cdot |Q_i|/(2 \cdot 10^5 kLT_i(o)) - 1$. Since $i \in I$, we have $|Q_i|/T_i(o) \geq \frac{1}{2}\gamma$. Since $\widehat{m} \geq 10^9 kL/\gamma$, we have $r \geq \widehat{m}/(4 \cdot 10^5 kL) \cdot |Q_i|/T_i(o)$. For $s \in [r]$, we can define a random variable Y_s,

$$Y_s = \sum_{j=(s-1)\cdot b+1}^{s\cdot b} \sum_{p\in Q_i} h_{i,j}(p).$$

We have $\mathbb{E}[Y_s] = b \cdot |Q_i|/(10^4 kLT_i(o))$. Thus, $E[Y_s] \in [10, 20]$. Since Y_s is a sum of several (at least) pairwise independent unit random variables, $\mathbb{V}[Y_s] \leq 20$. Thus, by Chebyshev's inequality, we have

$$\Pr\left[|Y_s - \mathbb{E}[Y_s]| \geq 9\right] \leq 20/81 \leq 0.25.$$

Define X_s be the random variable such that $X_s = \mathbf{1}(Y_s \geq 1)$. Then $\mathbb{E}\left[\sum_{s\in[r]} X_s\right] \geq 0.75r$. By Chernoff bound, we know that

$$\Pr\left[\sum_{s\in[r]} X_s \leq 0.5r\right] \leq 2^{-r/20} \leq 0.01\delta/L,$$

where the last inequality follows by $r \geq \widehat{m}/(4 \cdot 10^5 kL) \cdot \gamma/2 \geq 20\log(100L/\delta)$.

Since $\widehat{m}$ is sufficiently large, i.e.,

$$\widehat{m} \geq \Omega\left(k\epsilon^{-2}L^3d^4 \log\left(\frac{dLk}{\delta}\right) \cdot \frac{1}{\delta\gamma}\right) \geq \Omega\left(k\epsilon^{-3}L^4d^7 \log\left(\frac{dLk}{\delta}\right) \cdot \frac{1}{\delta}\right),$$

we have

$$r \geq \widehat{m}/(4 \cdot 10^5 kL) \cdot |Q_i|/T_i(o) \geq \Omega\left(\epsilon^{-2}Ld^4 \log\left(\frac{dLk}{\delta}\right) \cdot \frac{L}{\delta} \cdot \frac{\widehat{q}_i}{T_i(o)}\right),$$

where the last inequality follows by $\widehat{q}_i = \Theta(|Q_i|)$. Notice that

$$\Pr\left[\sum_{j=1}^{\widehat{m}} \mathbf{1}(|\{p \in Q_i \mid h_{i,j}(p) = 1\}| > 0) \geq 0.5r\right] \geq \Pr\left[\sum_{s\in[r]} X_s \geq 0.5r\right] \geq 1 - 0.01\delta/L.$$

Thus, with probability at least $1 - 0.01\delta/L$,

$$\sum_{j=1}^{\widehat{m}} \mathbf{1}(|\{p \in Q_i \mid h_{i,j}(p) = 1\}| > 0) \geq \Omega\left(\epsilon^{-2}Ld^4 \log\left(\frac{dLk}{\delta}\right) \cdot \frac{L}{\delta} \cdot \frac{\widehat{q}_i}{T_i(o)}\right).$$

By taking the union bound over $i \in I$, we complete the proof. □

Lemma 27.6.18 (Sampling stage succeeds)**.** *Consider $o \geq \mathrm{OPT}/16$. Conditioning on $\widehat{f} : \bigcup_{i=0}^{L} G_i \to \mathbb{R}_+$ (in Algorithm 27.5) is good (Lemma 27.6.4), $\widehat{q}_0, \widehat{q}_1, \ldots, \widehat{q}_L$ (in Algorithm 27.5) are good (Lemma 27.6.6), and the number of center cells is at most $100kL$, if* SAMPLING(o, ϵ, δ) *does not output* ***FAIL*** *before line 27.5 of Algorithm 27.5, then with probability at least $1 - \delta/5$, it will not output* ***FAIL****.*

Proof. According to Lemma 27.6.14 and Lemma 27.6.15, with probability at least $1 - \delta/10$, $\forall i \in I$, the sampling procedure will not request too many samples from level i. According to Lemma 27.6.17, with probability at least $1 - \delta/10$, the number of samples needed for each level $i \in I$ is enough. Thus, with probability at least $1 - \delta/5$, the algorithm will not output **FAIL**. □

Lemma 27.6.19 (Samples can fit into the space)**.** *Suppose $o \geq \mathrm{OPT}/16$. Conditioning on $\widehat{f} : \bigcup_{i=0}^{L} G_i \to \mathbb{R}_+$ (in Algorithm 27.5) is good (Lemma 27.6.4), if the total number of center cells is at most $100kL$, with probability at least $1 - \delta/5$,* SAMPLING(o, ϵ, δ) *(Algorithm 27.5) will not output* ***FAIL*** *in line 27.5 nor line 27.5.*

Proof. Consider $i \in \{0, 1, \ldots, L\}$ and a cell $C \in G_i$ which is marked as crucial by POINTSESTIMATION$(o, \epsilon, \delta/2)$.

$$\begin{aligned}\mathbb{E}\left[\sum_{j=1}^{\widehat{m}} \sum_{p \in C \cap Q} h_{i,j}(p)\right] &= \widehat{m} \cdot |C \cap Q|/(10^4 kLT_i(o)) \\ &\leq O(\epsilon^{-3} L^3 d^7 \log(dLk/\delta)\delta^{-1}).\end{aligned}$$

By Theorem 27.6.3,

$$\Pr\left[\sum_{j=1}^{\widehat{m}} \sum_{p \in C \cap Q} h_{i,j}(p) > \mathbb{E}\left[\sum_{j=1}^{\widehat{m}} \sum_{p \in C \cap Q} h_{i,j}(p)\right] + \epsilon^{-3} L^3 d^7 \log(dLk/\delta)\delta^{-1}\right] \leq (\delta/\Delta^d)^5.$$

Thus, by taking union bound, with probability at last $1-\delta/10$, $\forall i \in \{0,1,\ldots,L\}$, any cell $C \in G_i$ which is marked as crucial, the total number of points sampled in C is at most β. Thus, with probability at least $1-\delta/20$, SAMPLING(o,ϵ,δ) (Algorithm 27.5) will not output **FAIL** in line 27.5.

Consider $i \in \{0,1\ldots,L\}$. Let us analyze the number of cells in G_i which contain at least 1 sample points. The number of points which are not in the center cell is at most

$$\frac{\mathrm{OPT}}{(g_i/(2d))^2} \leq 400kT_i(o)\cdot \mathrm{OPT}/o \leq 6400kT_i(o).$$

Thus, the expected number of sampled points in non-center cell is at most $O(k\epsilon^{-3}L^3d^7\log(dLk/\delta)\cdot 1/\delta)$. Since the number of center cells is at most $100kL$, by Theorem 27.6.3, with probability at least $1-\delta/(100L)$, the number of cells in G_i which contain at least 1 sample points is $O(k\epsilon^{-3}L^3d^7\log(dLk/\delta)\cdot 1/\delta)$. By taking union bound over all $i \in \{0,1,\ldots,L\}$, with probability at least $1-\delta/20$, $\forall i \in \{0,1,\ldots,L\}$, the number of sampled cell in G_i is at most α.

Due to Lemma 27.6.2, with probability at least $1-\delta/10$, none of the STORING$(G_i,\alpha,\beta,0.1\delta/L)$ in line 27.5 of Algorithm 27.5 will output **FAIL**. By union bound over all the failure probabilities, we complete the proof. □

Lemma 27.6.20 (The overall success probability)**.** *Suppose* $o \geq \mathrm{OPT}/16$ *and the total number of center cells is at most* $100kL$*. With probability at least* $1-4\delta/5$*,* SAMPLING(o,ϵ,δ) *(Algorithm 27.5) will not output* ***FAIL****.*

Proof. Due to Lemma 27.6.10, POINTSESTIMATION$(o,\epsilon,\delta/2)$ in line 27.5 will not output **FAIL** with probability at least $1-3\delta/20$. By Lemma 27.6.5 and Lemma 27.6.6, with

probability at least $1-\delta/5$, $\forall i \in \{0,1,\ldots,L\}$, either $\widehat{q}_i \in |Q_i| \pm 0.1\epsilon\gamma T_i(o)$ or $\widehat{q}_i \in (1 \pm 0.01\epsilon)|Q_i|$, and $\forall C \in G_i$, either $\widehat{f}(C) \in |C \cap Q| \pm 0.1T_i(o)$ or $\widehat{f}(C) \in (1 \pm 0.01)|C \cap Q|$. Then by Lemma 27.6.19, with probability at least $1-\delta/5$, SAMPLING(o,ϵ,δ) will not output **FAIL** in line 27.5 nor line 27.5. Finally, according to Lemma 27.6.18, with probability at least $1-\delta/5$, the algorithm does not output **FAIL**. By taking union bound over all the bad events, with probability at least $1-3\delta/20-\delta/5-\delta/5-\delta/5 \geq 1-4\delta/5$, SAMPLING$(o,\epsilon,\delta)$ (Algorithm 27.5) will not output **FAIL**. □

Lemma 27.6.21 (Total space of Algorithm 27.5)**.** SAMPLING(o,ϵ,δ) *uses space at most* $O(k\epsilon^{-6}L^8d^{15}\delta^{-2}\cdot\log^4(kLd/(\epsilon\delta)))$ *bits.*

Proof. According to Lemma 27.6.11, POINTSESTIMATION$(o,\epsilon,\delta/2)$ in line 27.5 of Algorithm 27.5 takes the space $O(k\epsilon^{-3}L^4d^5\cdot\log(kLd/(\epsilon\delta)))$ bits. According to Lemma 27.6.2, line 27.5 takes the total space:

$$O(L\cdot\alpha\beta dL\cdot\log^2(\alpha\beta/\delta)) = O(k\epsilon^{-6}L^8d^{15}\delta^{-2}\cdot\log^4(kLd/(\epsilon\delta))).$$

□

27.6.4 The Final Algorithm

Finally, we use exponential search to enumerate the guesses o. In Algorithm 27.6, we show the details of how to run Algorithm 27.5 with different guesses in parallel. Our main theorem is the following.

Theorem 27.6.22. *Suppose a point set $Q \subseteq [\Delta]^d$ is given by a stream of insertion/deletion operations in the dynamic streaming model (Definition 27.2.1). Let $L = \log\Delta$. For given*

Algorithm 27.6 Coreset construction over a dynamic stream

1: DynamicCoreset(ϵ):
2: **input:** a point set $Q \subseteq [\Delta]^d$ described by a stream $\{(p_1, \pm), (p_2, \pm), \ldots\}$
3: Impose randomly shifted grids $G_{-1}, G_0, G_1, \ldots, G_L$ (Definition 27.3.1).
4: Run Distinct($10000k, 0.001$) (Lemma 27.6.1) over the input stream in parallel.
5: If line 27.6 does not output **FAIL**, we output the entire point set Q.
6: For each $u \in [2dL]$, let $o_u = 2^u \cdot 50k$ and run Sampling($o_u, \epsilon, 0.001/(dL)$) (Algorithm 27.5) over the input stream in parallel.
7: Set a threshold $h = \Theta(k\epsilon^{-2}L^2d^4 \log(kLd))$.
8: Find the smallest u^* such that Sampling($o_{u^*}, \epsilon, 0.001/(dL)$) in line 27.6 does not output **FAIL**, and the returned set S^* has size at most h, i.e., $|S^*| \leq h$.
9: If no such u^*, output **FAIL**. Otherwise, output S^* returned by Sampling($o_{u^*}, \epsilon, 0.001/(dL)$).

$\epsilon \in (0, 1/2)$, Algorithm 27.6 uses a single pass over the stream and on termination outputs a k-means ϵ-coreset S (Definition 27.2.3) for Q with probability at least 0.9. Furthermore, the size of the coreset is at most $O(k\epsilon^{-2}d^4L^2 \log(kdL))$. The total space used by the algorithm is $\widetilde{O}(k) \cdot \mathrm{poly}(dL/\epsilon)$ bits.

We divide the proof of Theorem 27.6.22 into the following two lemmas.

Lemma 27.6.23 (Correctness and success probability)**.** *With probability at least 0.9, DynamicCoreset(ϵ) (Algorithm 27.6) outputs an ϵ-coreset for Q and the size of the coreset is at most $O(k\epsilon^{-2}L^2d^4 \log(kLd))$.*

Proof. If $|Q| \leq 10000k$, then according to Lemma 27.6.1, with probability at least 0.999, the entire data set Q will be returned by line 27.6 in Algorithm 27.6.

Consider the case when $|Q| \geq 10000k$. According to Lemma 27.3.7, with probability at least 0.94, the total number of center cells is upper bounded by $100kL$. Now, we condition on this happens. Since $|Q| \geq 10000k$, we know that $\mathrm{OPT} \geq 1000k$. There exists $u \in [2dL]$ such that $o_u \in [\mathrm{OPT}/16, \mathrm{OPT}]$. According to Lemma 27.6.20, with probability

at least 0.999, SAMPLING$(o_u, \epsilon, 0.001/(dL))$ will not output **FAIL**. By Lemma 27.6.13, with probability at least 0.999, the set S returned by SAMPLING$(o_u, \epsilon, 0.001/(dL))$ is an ϵ-coreset and $|S| \leq O(k\epsilon^{-2}L^2d^4)$. Thus, with probability at least 0.998, DYNAMICCORESET(ϵ) (Algorithm 27.6) will not output **FAIL**. Consider another $u' < u$. If SAMPLING$(o_{u'}, \epsilon, 0.001/(dL))$ does not output **FAIL**, and the set S' returned has size at most h, then according to Lemma 27.6.13, with probability at least $1 - 0.001/(dL)$, S' is an ϵ-coreset for Q. By taking union bound over all the such u', then with probability at least 0.999, S^* returned by SAMPLING$(o_{u^*}, \epsilon, 0.001/(dL))$ is an ϵ-coreset for Q. By taking union bound over all the bad events, we complete the proof. □

Lemma 27.6.24 (Total space needed for Algortihm 27.6)**.** DYNAMICCORESET(ϵ) *(Algorithm 27.6) uses space at most* $O(k\epsilon^{-6}L^{11}d^{18}\log^4(kLd/\epsilon))$ *bits.*

Proof. DYNAMICCORESET(ϵ) (Algorithm 27.6) runs $\Theta(dL)$ copies of SAMPLING$(o_u, \epsilon, 0.001/(dL))$. By Lemma 27.6.21, the total space needed is

$$O(dL \cdot k\epsilon^{-6}L^8d^{15} \cdot (dL)^2 \cdot \log^4(kLd/\epsilon)) = O(k\epsilon^{-6}L^{11}d^{18}\log^4(kLd/\epsilon))$$

bits. □

Proof of Theorem 27.6.22. Lemma 27.6.23 shows the correctness and the success probability of the algorithm. Lemma 27.6.24 shows the total space needed by the algorithm. □

27.7 Why Do Previous Techniques Fail?

In this section, we describe some previous techniques in more detail and explain why they fail in our setting.

Uniform sampling method. [FIS05] is one of the early papers using sampling procedures to solve dynamic streaming geometric problems. They showed that it is possible to use point samples from a dynamic point set to solve several geometric problems, e.g., Euclidean Minimum Spanning Tree. However, they only showed how to implement *uniform* sampling by using counting distinct elements and subsampling procedure as subroutines. In our setting, we require different sampling probabilities for different points. Although the bottom-level uniform sampling scheme of ours is similar to theirs, our overall sampling method is more complicated.

Estimating the cost. [Ind04] used a critical observation to *estimate* the cost of k-median, that is: let Z be a set of centers and P be the point set, then $\text{cost}(Z,P) = \int_0^\infty |P - B(Z,r)|\mathrm{d}r$, where $B(Z,r)$ is the union of balls of radius r centered at all points in Z. Then this integration is approximated by a summation with logarithmic levels, i.e., $\int_0^\infty |P - B(Z,r)|\mathrm{d}r \approx \sum_{i=0}^{\infty} \left|P - B(Z,r^{(i+1)})\right| \cdot \left(r^{(i+1)} - r^{(i)}\right)$, where $r^{(i)} = O(\epsilon(1+\epsilon)^i)$. The critical part is to estimate $\left|P - B(Z,r^{(i)})\right|$. [Ind04] constructed a counting data structure based on grids with side length $O\left(r^{(i)}\right)$. Then every input point is snapped to a grid point. To obtain sufficiently accurate estimates for all $\left|P - B(Z,r^{(i)})\right|$, the data structure needs to query $|Z|/\epsilon^{O(d)}$ many grid points per Z. Such a data structure is implemented using pair-wise independent hash functions, and uses memory $|Z|/\epsilon^{O(d)}$. Notice that this method only

gives an *estimate* of the cost, and does not construct a coreset. In order to obtain a k-median solution, an exhaustive search is needed. Furthermore, this technique fails to extend to k-means which does not have an integration formula for the cost function.

Exponential size coreset. [FS05] constructed an ϵ-coreset of size $k\epsilon^{-O(d)}$ for k-means and k-median. They also used the same grid structure as we use. A cell is marked as "heavy" if the cell contains enough points such that moving all points in the cell to the center of this cell incurs too much error in the optimal cost of k-means/median. Since the side-lengths of cells decrease as level increases, the number of points required to have this effect becomes larger. Eventually, all cells are non-heavy after some level. As such we also have a tree of heavy cells. The coreset is constructed by looking at each heavy cell and assigning each point in its non-heavy children cells to its center. It turns out that if we want an ϵ-coreset, the threshold of non-heavy cells is exponential in d, i.e., each non-heavy cell in level i cannot contain more than $\widetilde{O}\left(\epsilon^{O(d)} \cdot \mathrm{OPT}/2^i\right)$ points. This small threshold gives rise to $\widetilde{O}\left(1/\epsilon^{O(d)}\right)$ many heavy cells.

Insertion-only streams. Many of the previous *insertion-only* streaming coreset construction algorithms (e.g., [FS12]) heavily depend on a "merge-reduce" technique, i.e. reading some points in the stream, constructing a coreset, reading another part, constructing a new coreset, and then merging the two coresets. This procedure is repeated until the stream ends. This technique works well in the insertion-only streaming model, but it fails immediately when deletions are allowed. Although [BFL16] gave a new framework other than merge-reduce, their algorithm relies on a non-deleting structure of data streams as well.

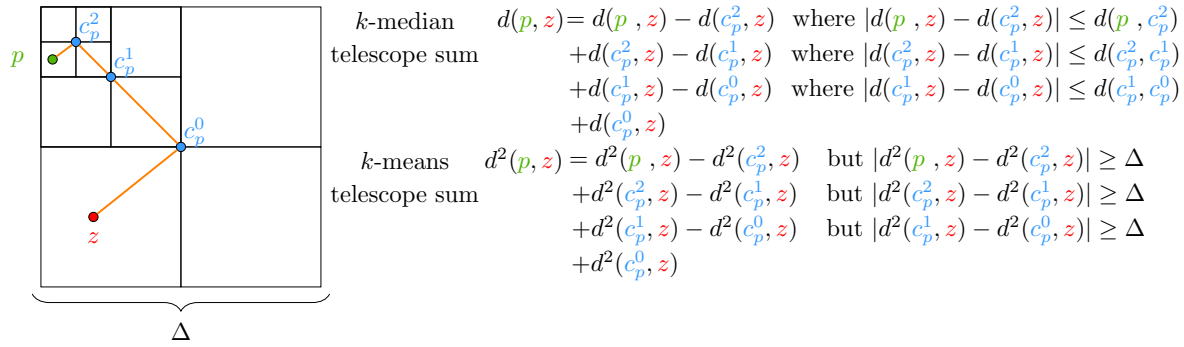


Figure 27.3: Telescope sum [BFL+17] fails for k-means. In the k-median problem, for a fixed set of centers Z, the total cost can be written as a telescope sum $\sum_{p\in P}(\mathrm{dist}(c_p^i, Z) - \mathrm{dist}(c_p^{i-1}, Z))$. For each piece, $|\mathrm{dist}(c_p^i, Z) - \mathrm{dist}(c_p^{i-1}, Z)|$ is always upper bounded by $\mathrm{dist}(c_p^{i-1}, c_p^i)$ which is independent from the choice of Z. However, in the k-means problem, the telescope sum of the total cost is $\sum_{p\in P}(\mathrm{dist}(c_p^i, Z)^2 - \mathrm{dist}(c_p^{i-1}, Z)^2)$. For each piece, the upper bound of $|\mathrm{dist}(c_p^i, Z)^2 - \mathrm{dist}(c_p^{i-1}, Z)^2|$ may depend on the location of Z, and it can be larger than Δ in the worst case.

Algorithm for k-median only. Although some k-median coreset construction techniques can be easily extended to k-means (see e.g. [FS12, BFL16]), those constructions can only be implemented in the insertion-only streaming model. [BFL+17] gave a k-median coreset construction in the dynamic streaming model, but their construction cannot be extended to k-means directly, as we explain now. Their k-median algorithm heavily relies on writing the cost of each point as a telescope sum. For example, we consider the 1-median problem. let z be a candidate center point and $p \in P$ be a point, then $\mathrm{dist}(p, z) = \mathrm{dist}(p, z) - \mathrm{dist}(c_p^{L-1}, z) + \mathrm{dist}(c_p^{L-1}, z) - \mathrm{dist}(c_p^{L-2}, z) + \cdots - \mathrm{dist}(c_p^0, z)$, where each c_p^i is the center of the cell in the i-th level containing p. Therefore, the total 1-median cost $\sum_{p\in P}\mathrm{dist}(p, z)$ of point set P on z can be split into L pieces, i.e., $\sum_{p\in P}(\mathrm{dist}(c_p^i, z) - \mathrm{dist}(c_p^{i-1}, z))$ for each $i \in [L]$. [BFL+17] estimated each of the L pieces by sampling points, i.e., let S^i be the samples in the i-th level, then the estimator of $\sum_{p\in P}(\mathrm{dist}(c_p^i, z) - \mathrm{dist}(c_p^{i-1}, z))$ is $\sum_{p\in S^i}(\mathrm{dist}(c_p^i, z) - \mathrm{dist}(c_p^{i-1}, z))/\zeta_p^i$ where ζ_p^i is the probability that point p is sampled. A crucial observation is that we have

$|\operatorname{dist}(c^i, z) - \operatorname{dist}(c^{i-1}, z)| \leq \Delta/2^i$ – the cell size in level i which is independent of the location of z. Using this nice upper bound on $|\operatorname{dist}(c^i, z) - \operatorname{dist}(c^{i-1}, z)|$, [BFL+17] applied Bernstein inequality to get high concentration of the estimator $\sum_{p \in S^i} (\operatorname{dist}(c_p^i, z) - \operatorname{dist}(c_p^{i-1}, z))/\zeta_p^i$ with only $\widetilde{O}(1/\epsilon^2)$ samples per level. However, this framework does not work for 1-means even though one can still write the telescope sum structure $\sum_p (\operatorname{dist}^2(c_p^i, z) - \operatorname{dist}^2(c_p^{i-1}, z))$ and can still setup an estimator $\sum_{p \in S^i} (\operatorname{dist}^2(c_p^i, z) - \operatorname{dist}^2(c_p^{i-1}, z))/\zeta_p^i$. But $|\operatorname{dist}^2(c_p^i, z) - \operatorname{dist}^2(c_p^{i-1}, z)|$ is not upper bounded by the cell size anymore. Instead, it depends on the location of z. For example, it can be as large as $|\operatorname{dist}^2(c_p^i, z) - \operatorname{dist}^2(c_p^{i-1}, z)| \geq \Delta$. See Figure 27.3. If we apply Bernstein inequality here, we will need too many samples to save any space.

Chapter 28

Convergence Result of one-hidden Layer Neural Network

We improve the over-parametrization size over the two beautiful results [Li and Liang' 2018] and [Du, Zhai, Poczos and Singh' 2019] in deep learning theory.

This part is based upon the following previous publication

- Zhao Song, Xin Yang

  Quadratic Suffices for Over-parametrization via Matrix Chernoff Bound.

  Manuscript 2019 [SY19]

28.1 Introduction

Over-parameterization theory for deep neural networks becomes extremely popular over the last few years. There is a long line (still growing very quickly) of work proving that (stochastic) gradient descent algorithm is able to find the global minimum if the network is wide enough [LL18, DZPS19, AZLS18, AZLS19, DLL+19, ZCZG18]. One fundamental question for over-parameterization is, how wide should the neural network be?

Formally speaking, the existing results show that as long as the width m is at least polynomial of number of input data n, then (S)GD-type algorithm can work in the following sense: first randomly pick a weight matrix to be the initialization point, update the weight matrix according to gradient direction over each iteration, and eventually find the global minimum. It is conjectured [Lee18] that $m = \Omega(n \operatorname{poly}(\log(n/\delta)))$ is the right answer, where δ is the failure probability. The randomness is from the random initialization and also algorithm itself, but not from data. There are other work relied on input data to be random [BG17, Tia17, ZSJ+17, Son17, LY17, ZSD17, DLT+18, GLM18, BJW19], however over-parameterization theory does not allow that assumption and it only needs to make very mild assumption on data, e.g. separable. The breakthrough result by Li and Liang [LL18] is the first one that is able to explain why the greedy algorithm works very well in practice for ReLU neural network from over-parameterization perspective. The state-of-the-art result for one-hidden-layer neural network with ReLU activation function is due to Du, Zhai, Poczos and Singh [DZPS19]. Their beautiful result proved that $m = \Omega(n^6 \operatorname{poly}(\log n, 1/\delta))$ is sufficient. We improve the result [DZPS19] from two perspectives : one is the dependence on failure probability, and the other is the dependence on the number of input data. More precisely, we show that $m = \Omega(n^4 \operatorname{poly}(\log(n/\delta)))$ is sufficient via a careful concentration analysis.

More interestingly, when the input data have certain property, we can improve the bound to $m = \Omega(n^2 \operatorname{poly}(\log(n/\delta)))$ via a more careful concentration analysis for random variables.

The study of concentration of sums of random variables dates back to Central Limit Theorems, the first modern concentration bounds were probably proposed by Bernstein [Ber24]. Chernoff bound is an extremely popular variant, which was introduced by Rubin and published by Chernoff [Che52]. Chernoff bound is a fundamental tool in Theoretical Computer Science and has been used in almost every randomized algorithm paper without even stating it. One common statement is the following: given a list of independent random variables $x_1, \cdots, x_m \in [0, 1]$ with mean μ, then

$$\Pr\left[\left|\frac{1}{m}\sum_{i=1}^{m} x_i - \mu\right| > \epsilon\right] \leq 2\exp(-\Omega(m\epsilon^2))$$

In many applications, we are not just dealing with scalar random variables. A natural generalization of the Chernoff bound appeared in the works of Rudelson [Rud99], Ahlswede-Winter [AW02], and Tropp [Tro12]. They proved that a similar concentration phenomenon is true even for matrix random variables. Given a list of independent complex Hermitian random matrices $X_1, \cdots, X_m \in \mathbb{C}^{n\times n}$ with mean μ and $\|X_i\| \leq 1$, $\forall i \in [m]$, then

$$\Pr\left[\left\|\frac{1}{m}\sum_{i=1}^{m} X_i - \mu\right\| > \epsilon\right] \leq 2n\exp(-\Omega(m\epsilon^2))$$

For a more detailed survey and recent progress on the topic Matrix Chernoff bound. We refer the readers to [Tro15, GLSS18, KS18].

In this work, we draw an interesting connection between deep learning theory and Matrix Chernoff bound : we can view the width of neural network as the number of independent random matrices.

28.1.1 Our Result

We start with the definition of Gram matrix, which can be found in [DZPS19].

Definition 28.1.1 (Data-dependent function H)**.** Given a collection of data $\{x_1, \cdots, x_n\} \subset \mathbb{R}^d$. For any vector $w \in \mathbb{R}^d$, we define symmetric matrix $H(w) \in \mathbb{R}^{n \times n}$ as follows

$$H(w)_{i,j} = x_i^\top x_j \mathbf{1}_{w^\top x_i \geq 0, w^\top x_j \geq 0}, \forall (i,j) \in [n] \times [n].$$

Then we define continuous Gram matrix $H^{\mathrm{cts}} \in \mathbb{R}^{n \times n}$ in the following sense

$$H^{\mathrm{cts}} = \mathop{\mathbb{E}}_{w \sim N(0,I)}[H(w)].$$

Similarly, we define discrete Gram matrix $H^{\mathrm{dis}} \in \mathbb{R}^{n \times n}$ in the following sense

$$H^{\mathrm{dis}} = \frac{1}{m} \sum_{r=1}^m H(w_r).$$

We use $N(0, I)$ to denote Gaussian distribution. We use $\mathbb{E}_w$ to denote $\mathbb{E}_{w \sim N(0,I)}$ and $\Pr_w$ to denote $\Pr_{w \sim N(0,I)}$. We introduce some mild data-dependent assumption. Without loss of generality, we can assume that $\|x_i\|_2 \leq 1$, $\forall i \in [n]$.

Assumption 28.1.1 (Data-dependent assumption)**.** *We made the following data-dependent assumption:*

1. *Let* $\lambda = \lambda_{\min}(\mathbb{E}_w[H(w)])$ *and* $\lambda \in (0, 1]$.
2. *Let* $\alpha \in [1, n]$ *and* $\gamma \in [0, 1)$ *be the parameter such that* [1]

$$\Pr_w \left[\left\| H(w) - \mathop{\mathbb{E}}_w[H] \right\| \leq \alpha \right] \geq 1 - \gamma.$$

[1] For simplicity, let us assume $\gamma = 0$.

3. Let $\beta \in [1, n^2]$ be the parameter such that

$$\left\| \mathop{\mathbb{E}}_{w} \left[\Big(H(w) - \mathop{\mathbb{E}}_{w}[H(w)] \Big) \Big(H(w) - \mathop{\mathbb{E}}_{w}[H(w)] \Big)^\top \right] \right\| \leq \beta.$$

4. Let $\theta \in [0, \sqrt{n}]$ be parameter such that

$$|x_i^\top x_j| \leq \theta / \sqrt{n}, \forall i \neq j.$$

The first assumption is from [DZPS19]. For more detailed discussion about that assumption, we refer the readers to [DZPS19]. The last assumption is similar to assumption in [AZLS18, AZLS19], where they assumed that for $i \neq j$, $\|x_i - x_j\|_2 \geq \theta'$. If we think of $\|x_i\|_2 = 1, \forall i \in [n]$, then we know that $(\theta')^2 \leq 2 - 2x_i^\top x_j$. It indicates $(\theta')^2 + 2\theta/\sqrt{n} \leq 2$. The second and the third assumption are motivated by Matrix Chernoff bound. The reason for introducing these Matrix Chernoff-type assumption is, the goal is to bound the spectral norm of the sums of random matrices in several parts of the proof. One way is to relax the spectral norm to the Frobenious norm, and bound each entry of the matrix, and finally union bound over all entries in the matrix. This could potentially lose a $\sqrt{n}$ factor compared to applying Matrix Chernoff bound. We feel these assumptions can indicate how the input data affect the over-parameterization size m in a more clear way.

We state our result for the concentration of sums of independent random matrices:

Proposition 28.1.2 (Informal of Theorem 28.5.1)**.** *Assume Part 1,2 and 3 of Assumption 28.1.1. If $m = \Omega((\lambda^{-2}\beta + \lambda^{-1}\alpha)\log(n/\delta))$, then*

$$\Pr_{w_1, \cdots, w_m \in N(0,I)}[\|H^{\mathrm{dis}} - H^{\mathrm{cts}}\|_2 \leq \lambda/4] \geq 1 - \delta.$$

Table 28.1: Summary of Convergence Result

Reference	m	λ	α	θ
[DZPS19]	$\lambda^{-4}n^6 \operatorname{poly}(\log n, 1/\delta)$	Yes	No	No
Theorem 28.1.3	$\lambda^{-4}n^4 \log^3(n/\delta)$	Yes	No	No
Theorem 28.1.4	$\lambda^{-4}n^3 \log^3(n/\delta) \cdot \alpha$	Yes	Yes	No
Theorem 28.1.5	$\lambda^{-4}n^2 \log^3(n/\delta) \cdot \alpha(\alpha + \theta^2)$	Yes	Yes	Yes

Proposition 28.1.2 is a direct improvement compared to Lemma 3.1 in [DZPS19], which requires $m = \Omega(\lambda^{-2}n^2 \log(n/\delta))$. Proposition 28.1.2 is better when input data points have some good properties, e.g., $\beta, \alpha = o(n^2)$. However the result in [DZPS19] always needs to pay n^2 factor, no matter what the input data points are.

We state our convergence result as follows:

Theorem 28.1.3 (Informal of Theorem 28.4.5)**.** *Assume Part 1 of Assumption 28.1.1. Let m denote the width of neural network, let n denote the number of input data points. If $m = \Omega(\lambda^{-4}n^4 \operatorname{poly}(\log(n/\delta)))$, then gradient descent is able to find the global minimum from a random initialization point with probability $1 - \delta$.*

Theorem 28.1.3 is a direct improvement compared to Theorem 4.1 in [DZPS19], which requires $m = \Omega(\lambda^{-4}n^6 \operatorname{poly}(\log n, 1/\delta))$.

If we also allow Part 2 of Assumption 28.1.1, we can slightly improve Theorem 28.1.3 from n^4 to n^3,

Theorem 28.1.4 (Informal of Theorem 28.5.5)**.** *Assume Part 1 and 2 of Assumption 28.1.1. If $m = \Omega(\lambda^{-4}n^3\alpha \operatorname{poly}(\log(n/\delta)))$, then gradient descent is able to find the global minimum from a random initialization point with probability $1 - \delta$.*

Except for m, Theorem 4.1 in [DZPS19] requires step size η to be $\Theta(\lambda/n^2)$. Theorem 28.1.4 only needs step size η to be $\Theta(\lambda/(\alpha n))$.

Further, if we also allow Part 4 of Assumption 28.1.1, we can slightly improve Theorem 28.1.3 from n^4 to n^2,

Theorem 28.1.5 (Informal of Theorem 28.6.4)**.** *Assume Part 1, 2 and 4 of Assumption 28.1.1. If $m = \Omega(\lambda^{-4} n^2 \alpha(\theta^2 + \alpha) \operatorname{poly}(\log(n/\delta)))$, the gradient descent is able to find the global minimum from a random initialization point with probability $1 - \delta$.*

28.1.2 Technical Overview

We follow the exact same optimization framework as Du, Zhai, Poczos and Singh [DZPS19]. We improve the bound on m by doing a careful concentration analysis for random variables without changing the high-level optimization framework.

We briefly summarize the optimization framework here: the minimal eigenvalue λ of H^{cts}, as introduced in [DZPS19], turns out to be closely related with the convergence rate. As time evolves, the weights w in the network may vary; however if w stay in a ball of radius R that only depends on the number of data n and λ, and particularly does not depend on the number of neurons m, then we are still able to lower bound the minimal eigenvalue of $H(w)$. On the other hand, we want to upper bound D, the actual move of w, with high probability. It turns out D is proportional to $\frac{1}{\sqrt{m}}$. We require $D < R$ in order to control the convergence rate. In this way we derive a lower bound of m.

In order to bound $\|H\|$, [DZPS19] relax it to Frobenius norm and then relax it to entry-wise L1 norm,

$$\|H\| \leq \|H\|_F \leq \|H\|_1.$$

Then they can bound each term of $H_{i,j}$ individually via Markov inequality.

One key observation is that $\|H\|_1$ is a quite loose bound for $\|H\|_F$, in the sense that $\|H\|_1 = \|H\|_F$ holds only if H contains at most 1 non-zero entry. This means we can work on the Frobenius norm directly, and we shall be able to obtain a tighter estimation. By definition of H, it can be written as a summation of m independent matrices $A_1, A_2, \cdots, A_m \in \mathbb{R}^{n \times n}$,

$$H = \frac{1}{m} \sum_{r=1}^{m} A_r$$

In order to bound $\|H\|_F$, for each i, j, we regard each H_{ij} as summation of m independent random variables, then apply Bernstein bound to obtain experiential tail bound on the concentration of H_{ij}. Finally, by taking a union bound over all the n^2 pairs we obtain a tighter bound for $\|H\|_F$.

We shall mention that $\|H\|_F$ is also a loose upper bound of $\|H\|$, i.e., $\|H\|_F = \|H\|$ only if H is a rank-1 matrix. Hence, if the condition number of H is small, which may happen as a property of the data, then we may benefit from bounding $\|H\|$ directly. We achieve this by apply matrix Chernoff bound, which states the spectral norm of summation of m independent matrices concentrates under certain conditions.

We shall stress that mutually independence plays a very important role in our argument. Throughout the whole paper we are dealing with summations of the form $\sum_{r=1}^{m} y_r$ where $\{y_m\}_{r=1}^{m}$ are independent random variables. Previous argument mainly applies Markov inequality, which pays a factor of $1/\delta$ around the mean for error probability δ. But we can obtain much tighter concentration bound by taking advantage of independence as in Bernstein inequality and Hoeffding inequality. This allows us to improve the dependency on δ from $1/\delta$ to $\log 1/\delta$.

We also make use of matrix spectral norm to deal with summation of the form $\|\sum_{i=1}^{n} a_i x_i\|_2$ where $\{a_i\}_{i=1}^{n}$ are scalars and $\{x_i\}_{i=1}^{n}$ are vectors. Naively applying triangle inequality leads to an upper bound proportional to $\|a\|_1$, which can be as large as $\sqrt{n}\|a\|_2$. Instead, we observe that the matrix formed by $\begin{pmatrix} x_1 & \cdots & x_n \end{pmatrix} := X$ has good singular value property, which allows us to obtain the bound $\|X\|_2 \cdot \|a\|_2$. Therefore, this bound does not rely on number of inputs explicitly.

Roadmap We provide some basic definitions and probability tools in Section 28.2. We define the optimization problem in Section 28.3. We present our quartic result in Section 28.4. We improve it to cubic and quadratic in Section 28.5 and Section 28.6.

28.2 Preliminaries

28.2.1 Notation

We use $[n]$ to denote $\{1, 2, \cdots, n\}$. We use ϕ to denote ReLU activation function, i.e., $\phi(x) = \max\{x, 0\}$. For an event $f(x)$, we define $\mathbf{1}_{f(x)}$ such that $\mathbf{1}_{f(x)} = 1$ if $f(x)$ holds and $\mathbf{1}_{f(x)} = 0$ otherwise. For a matrix A, we use $\|A\|$ to denote the spectral norm of A. We define $\|A\|_F = (\sum_i \sum_j A_{i,j}^2)^{1/2}$ and $\|A\|_1 = \sum_i \sum_j |A_{i,j}|$.

28.3 Problem Formulation

Our problem formulation is the same as [DZPS19]. We consider a two-layer ReLU activated neural network with m neurons in the hidden layer:

$$f(W, x, a) = \frac{1}{\sqrt{m}} \sum_{r=1}^{m} a_r \phi(w_r^\top x),$$

where $x \in \mathbb{R}^d$ is the input, $w_1, \cdots, w_m \in \mathbb{R}^d$ are weight vectors in the first layer, $a_1, \cdots, a_m \in \mathbb{R}$ are weights in the second layer. For simplicity, we only optimize W but not optimize a and W at the same time.

Recall that the ReLU function $\phi(x) = \max\{x, 0\}$. Therefore for $r \in [m]$, we have

$$\frac{f(W, x, a)}{\partial w_r} = \frac{1}{\sqrt{m}} a_r x \mathbf{1}_{w_r^\top x \geq 0}. \tag{28.1}$$

We apply the gradient descent to optimize the weight matrix W in the following standard way,

$$W(k+1) = W(k) - \eta \frac{\partial L(W(k))}{\partial W(k)}. \tag{28.2}$$

We define objective function L as follows

$$L(W) = \frac{1}{2} \sum_{i=1}^{n} (y_i - f(W, x_i, a))^2.$$

We can compute the gradient of L in terms of w_r

$$\frac{\partial L(W)}{\partial w_r} = \frac{1}{\sqrt{m}} \sum_{i=1}^{n} (f(W, x_i, a_r) - y_i) a_r x_i \mathbf{1}_{w_r^\top x_i \geq 0}. \tag{28.3}$$

We consider the ordinary differential equation defined by

$$\frac{\mathrm{d} w_r(t)}{\mathrm{d} t} = -\frac{\partial L(W)}{\partial w_r}. \tag{28.4}$$

At time t, let $u(t) = (u_1(t), \cdots, u_n(t)) \in \mathbb{R}^n$ be the prediction vector where each $u_i(t)$ is defined as

$$u_i(t) = f(W(t), a, x_i). \tag{28.5}$$

28.4 Quartic Suffices

28.4.1 Bounding the difference between continuous and discrete

Lemma 28.4.1 (Lemma 3.1 in [DZPS19]). *We define $H^{\mathrm{cts}}, H^{\mathrm{dis}} \in \mathbb{R}^{n\times n}$ as follows*

$$H^{\mathrm{cts}}_{i,j} = \mathop{\mathbb{E}}_{w\sim\mathcal{N}(0,I)}\left[x_i^\top x_j \mathbf{1}_{w^\top x_i\geq 0, w^\top x_j\geq 0}\right],$$
$$H^{\mathrm{dis}}_{i,j} = \frac{1}{m}\sum_{r=1}^m \left[x_i^\top x_j \mathbf{1}_{w_r^\top x_i\geq 0, w_r^\top x_j\geq 0}\right].$$

Let $\lambda = \lambda_{\min}(H^{\mathrm{cts}})$. If $m = \Omega(\lambda^{-2}n^2\log(n/\delta))$, we have

$$\|H^{\mathrm{dis}} - H^{\mathrm{cts}}\|_F \leq \frac{\lambda}{4}, \text{ and } \lambda_{\min}(H^{\mathrm{dis}}) \geq \frac{3}{4}\lambda.$$

hold with probability at least $1-\delta$.

For the completeness, we still provide a proof here.

Proof. For every fixed pair (i,j), $H^{\mathrm{dis}}_{i,j}$ is an average of independent random variables, i.e.

$$H^{\mathrm{dis}}_{i,j} = \frac{1}{m}\sum_{r=1}^m x_i^\top x_j \mathbf{1}_{w_r^\top x_i\geq 0, w_r^\top x_j\geq 0}.$$

Then the expectation of $H^{\mathrm{dis}}_{i,j}$ is

$$\begin{aligned}\mathbb{E}[H^{\mathrm{dis}}_{i,j}] &= \frac{1}{m}\sum_{r=1}^m \mathop{\mathbb{E}}_{w_r\sim N(0,I_d)}\left[x_i^\top x_j \mathbf{1}_{w_r^\top x_i\geq 0, w_r^\top x_j\geq 0}\right] \\ &= \mathop{\mathbb{E}}_{w\sim N(0,I_d)}\left[x_i^\top x_j \mathbf{1}_{w^\top x_i\geq 0, w^\top x_j\geq 0}\right] \\ &= H^{\mathrm{cts}}_{i,j}.\end{aligned}$$

For $r\in[m]$, let $z_r = \frac{1}{m}x_i^\top x_j \mathbf{1}_{w_r^\top x_i\geq 0, w_r^\top x_j\geq 0}$. Then z_r is a random function of w_r, hence $\{z_r\}_{r\in[m]}$ are mutually independent. Moreover, $-\frac{1}{m}\leq z_r\leq \frac{1}{m}$. So by Hoeffding inequality(Lemma A.1.5) we have for all $t>0$,

$$\Pr\left[|H^{\mathrm{dis}}_{i,j} - H^{\mathrm{cts}}_{i,j}|\geq t\right] \leq 2\exp\left(-\frac{2t^2}{4/m}\right) = 2\exp(-mt^2/2)$$

Setting $t = (\frac{1}{m} 2 \log(2n^2/\delta))^{1/2}$, we can apply union bound on all pairs (i, j) to get with probability at least $1 - \delta$, for all $i, j \in [n]$,

$$|H^{\mathrm{dis}}_{i,j} - H^{\mathrm{cts}}_{i,j}| \leq \Big(\frac{2}{m} \log(2n^2/\delta)\Big)^{1/2} \leq 4\Big(\frac{\log(n/\delta)}{m}\Big)^{1/2}.$$

Thus we have

$$\|H^{\mathrm{dis}} - H^{\mathrm{cts}}\|^2 \leq \ \|H^{\mathrm{dis}} - H^{\mathrm{cts}}\|_F^2 = \ \sum_{i=1}^{n} \sum_{j=1}^{n} |H^{\mathrm{dis}}_{i,j} - H^{\mathrm{cts}}_{i,j}|^2 \leq \frac{1}{m} 16 n^2 \log(n/\delta).$$

Hence if $m = \Omega(\lambda^{-2} n^2 \log(n/\delta))$ we have the desired result. □

We define the event

$$A_{i,r} = \Big\{ \exists u : \|u - \widetilde{w}_r\|_2 \leq R, \mathbf{1}_{x_i^\top \widetilde{w}_r \geq 0} \neq \mathbf{1}_{x_i^\top u \geq 0} \Big\}.$$

Note this event happens if and only if $|\widetilde{w}_r^\top x_i| < R$. Recall that $\widetilde{w}_r \sim \mathcal{N}(0, I)$. By anti-concentration inequality of Gaussian (Lemma A.1.11), we have

$$\Pr[A_{i,r}] = \Pr_{z \sim \mathcal{N}(0,1)}[|z| < R] \leq \frac{2R}{\sqrt{2\pi}}. \tag{28.6}$$

28.4.2 Bounding changes of H when w is in a small ball

We improve the Lemma 3.2 in [DZPS19] from the two perspective : one is the probability, and the other is upper bound on spectral norm.

Lemma 28.4.2 (perturbed w)**.** *Let $R \in (0,1)$. If $\widetilde{w}_1, \cdots, \widetilde{w}_m$ are i.i.d. generated $\mathcal{N}(0, I)$. For any set of weight vectors $w_1, \cdots, w_m \in \mathbb{R}^d$ that satisfy for any $r \in [m]$, $\|\widetilde{w}_r - w_r\|_2 \leq R$, then the $H : \mathbb{R}^{m \times d} \to \mathbb{R}^{n \times n}$ defined*

$$H(w)_{i,j} = \frac{1}{m} x_i^\top x_j \sum_{r=1}^m \mathbf{1}_{w_r^\top x_i \geq 0, w_r^\top x_j \geq 0}.$$

Then we have

$$\|H(w) - H(\widetilde{w})\|_F < 2nR,$$

holds with probability at least $1 - n^2 \cdot \exp(-mR/10)$.

Proof. The random variable we care is

$$\begin{aligned}
\sum_{i=1}^n \sum_{j=1}^n |H(\widetilde{w})_{i,j} - H(w)_{i,j}|^2 = & \frac{1}{m^2} \sum_{i=1}^n \sum_{j=1}^n \left| x_i^\top x_j \sum_{r=1}^m (\mathbf{1}_{\widetilde{w}_r^\top x_i \geq 0, \widetilde{w}_r^\top x_j \geq 0} - \mathbf{1}_{w_r^\top x_i \geq 0, w_r^\top x_j \geq 0}) \right|^2 \\
\leq & \frac{1}{m^2} \sum_{i=1}^n \sum_{j=1}^n \left(\sum_{r=1}^m \mathbf{1}_{\widetilde{w}_r^\top x_i \geq 0, \widetilde{w}_r^\top x_j \geq 0} - \mathbf{1}_{w_r^\top x_i \geq 0, w_r^\top x_j \geq 0} \right)^2 \\
= & \frac{1}{m^2} \sum_{i=1}^n \sum_{j=1}^n \Big(\sum_{r=1}^m s_{r,i,j} \Big)^2,
\end{aligned}$$

where the last step follows from for each r, i, j, we define

$$s_{r,i,j} := \mathbf{1}_{\widetilde{w}_r^\top x_i \geq 0, \widetilde{w}_r^\top x_j \geq 0} - \mathbf{1}_{w_r^\top x_i \geq 0, w_r^\top x_j \geq 0}.$$

We consider i, j are fixed. We simplify $s_{r,i,j}$ to s_r.

Then s_r is a random variable that only depends on $\widetilde{w}_r$. Since $\{\widetilde{w}_r\}_{r=1}^m$ are independent, $\{s_r\}_{r=1}^m$ are also mutually independent.

If $\neg A_{i,r}$ and $\neg A_{j,r}$ happen, then

$$\left|\mathbf{1}_{\widetilde{w}_r^\top x_i \geq 0, \widetilde{w}_r^\top x_j \geq 0} - \mathbf{1}_{w_r^\top x_i \geq 0, w_r^\top x_j \geq 0}\right| = 0.$$

If $A_{i,r}$ or $A_{j,r}$ happen, then

$$\left|\mathbf{1}_{\widetilde{w}_r^\top x_i \geq 0, \widetilde{w}_r^\top x_j \geq 0} - \mathbf{1}_{w_r^\top x_i \geq 0, w_r^\top x_j \geq 0}\right| \leq 1.$$

So we have

$$\mathop{\mathbb{E}}_{\widetilde{w}_r}[s_r] \leq \mathop{\mathbb{E}}_{\widetilde{w}_r}\left[\mathbf{1}_{A_{i,r} \vee A_{j,r}}\right] \leq \Pr[A_{i,r}] + \Pr[A_{j,r}] \leq \frac{4R}{\sqrt{2\pi}} \leq 2R,$$

and

$$\mathop{\mathbb{E}}_{\widetilde{w}_r}\left[\left(s_r - \mathop{\mathbb{E}}_{\widetilde{w}_r}[s_r]\right)^2\right] = \mathop{\mathbb{E}}_{\widetilde{w}_r}[s_r^2] - \mathop{\mathbb{E}}_{\widetilde{w}_r}[s_r]^2 \leq \mathop{\mathbb{E}}_{\widetilde{w}_r}[s_r^2] \leq \mathop{\mathbb{E}}_{\widetilde{w}_r}\left[\left(\mathbf{1}_{A_{i,r} \vee A_{j,r}}\right)^2\right] \leq \frac{4R}{\sqrt{2\pi}} \leq 2R.$$

We also have $|s_r| \leq 1$. So we can apply Bernstein inequality (Lemma A.1.2) to get for all $t > 0$,

$$\begin{aligned}\Pr\left[\sum_{r=1}^m s_r \geq 2mR + mt\right] &\leq \Pr\left[\sum_{r=1}^m (s_r - \mathbb{E}[s_r]) \geq mt\right] \\ &\leq \exp\left(-\frac{m^2t^2/2}{2mR + mt/3}\right).\end{aligned}$$

Choosing $t = R$, we get

$$\Pr\left[\sum_{r=1}^m s_r \geq 3mR\right] \leq \exp\left(-\frac{m^2R^2/2}{2mR + mR/3}\right) \leq \exp\left(-mR/10\right).$$

Thus, we can have

$$\Pr\left[\frac{1}{m}\sum_{r=1}^m s_r \geq 2R\right] \leq \exp(-mR/10).$$

Therefore, we complete the proof. □

Table 28.2: Table of Parameters for the $m = \widetilde{\Omega}(n^4)$ result in Section 28.4. **Nt.** stands for notations.

Nt.	**Choice**	**Place**	**Comment**
λ	$:= \lambda_{\min}(H^{\text{cts}})$	Ass. 28.1.1	Data-dependent
R	λ/n	Eq. (28.11)	Maximal allowed movement of weight
D_{cts}	$\frac{\sqrt{n}\|y-u(0)\|_2}{\sqrt{m}\lambda}$	Lemma 28.4.3	Actual moving distance of weight, continuous case
D	$\frac{4\sqrt{n}\|y-u(0)\|_2}{\sqrt{m}\lambda}$	Lemma 28.4.6	Actual moving distance of weight, discrete case
η	λ/n^2	Eq. (28.11)	Step size of gradient descent
m	$\lambda^{-2}n^2\log(n/\delta)$	Lemma 28.4.1	Bounding discrete and continuous
m	$\lambda^{-4}n^4\log^3(n/\delta)$	Lemma 28.4.4 Claim 28.4.7	$D < R$ and $\|y-u(0)\|_2^2 = \widetilde{O}(n)$

28.4.3 Loss is decreasing while weights are not changing much

For simplicity of notation, we provide the following definition.

Definition 28.4.1. For any $s \in [0,t]$, we define matrix $H(s) \in \mathbb{R}^{n\times n}$ as follows

$$H(s)_{i,j} = \frac{1}{m}\sum_{r=1}^{m} x_i^\top x_j \mathbf{1}_{w_r(s)^\top x_i \geq 0, w_r(s)^\top x_j \geq 0}.$$

With H defined, it becomes more convenient to write the dynamics of predictions.

For each $i \in [n]$, we have

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}t} u_i(t) &= \sum_{r=1}^{m} \left\langle \frac{\partial f(W(t), a, x_i)}{\partial w_r(t)}, \frac{\mathrm{d} w_r(t)}{\mathrm{d}t} \right\rangle \\
&= \sum_{r=1}^{m} \left\langle \frac{\partial f(W(t), a, x_i)}{\partial w_r(t)}, -\frac{\partial L(w(t), a)}{\partial w_r(t)} \right\rangle \\
&= \sum_{r=1}^{m} \left\langle \frac{\partial f(W(t), a, x_i)}{\partial w_r(t)}, -\frac{1}{\sqrt{m}} \sum_{i=1}^{n} (f(W, x_i, a_r) - y_i) a_r x_i \mathbf{1}_{w_r^\top x_i \geq 0} \right\rangle \\
&= \sum_{j=1}^{n} (y_j - u_j(t)) \left\langle \frac{\partial f(W(t), a, x_i)}{\partial w_r(t)}, \frac{\partial f(W(t), a, x_j)}{\partial w_r(t)} \right\rangle \\
&= \sum_{j=1}^{n} (y_j - u_j(t)) H(t)_{i,j}
\end{aligned}$$

where the first step follows from (28.5) and the chain rule of derivatives, the second step uses (28.3), the third step uses (28.4), the fourth step uses (28.1) and (28.5), and the last step uses the definition of the matrix H.

Hence, we have the following compact expression for $\frac{\mathrm{d}}{\mathrm{d}t} u(t)$ as a whole vector:

$$\frac{\mathrm{d}}{\mathrm{d}t} u(t) = H(t) \cdot (y - u(t)).$$

Lemma 28.4.3 (Lemma 3.3 in [DZPS19])**.** *Suppose for $0 \leq s \leq t$, $\lambda_{\min}(H(w(s))) \geq \lambda/2$. Let D_{cts} be defined as*

$$D_{\mathrm{cts}} := \frac{\sqrt{n} \|y - u(0)\|_2}{\sqrt{m\lambda}}.$$

Then we have

$$\|y - u(t)\|_2^2 \leq \exp(-\lambda t) \cdot \|y - u(0)\|_2^2,$$

and

$$\|w_r(t) - w_r(0)\|_2 \leq D_{\mathrm{cts}}.$$

For the completeness, we still provide a proof.

Proof. Recall we can write the dynamics of predictions as

$$\frac{\mathrm{d}}{\mathrm{d}t}u(t) = H(t) \cdot (y - u(t)).$$

We can calculate the loss function dynamics

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\|y - u(t)\|_2^2 = & -2(y - u(t))^\top \cdot H(t) \cdot (y - u(t)) \\ \leq & -\lambda \|y - u(t)\|_2^2.\end{aligned}$$

Thus we have $\frac{\mathrm{d}}{\mathrm{d}t}(\exp(\lambda t)\|y - u(t)\|_2^2) \leq 0$ and $\exp(\lambda t)\|y - u(t)\|_2^2$ is a decreasing function with respect to t.

Using this fact we can bound the loss

$$\|y - u(t)\|_2^2 \leq \exp(-\lambda t)\|y - u(0)\|_2^2. \tag{28.7}$$

Now, we can bound the gradient norm. Recall for $0 \leq s \leq t$,

$$\begin{aligned}\left\|\frac{\mathrm{d}}{\mathrm{d}s}w_r(s)\right\|_2 = & \left\|\sum_{i=1}^n (y_i - u_i)\frac{1}{\sqrt{m}}a_r x_i \cdot \mathbf{1}_{w_r(s)^\top x_i \geq 0}\right\|_2 \\ \leq & \frac{1}{\sqrt{m}}\sum_{i=1}^n |y_i - u_i(s)| \\ \leq & \frac{\sqrt{n}}{\sqrt{m}}\|y - u(s)\|_2 \\ \leq & \frac{\sqrt{n}}{\sqrt{m}}\exp(-\lambda s)\|y - u(0)\|_2.\end{aligned} \tag{28.8}$$

where the first step follows from (28.3), the second step follows from triangle inequality and $a_r = \pm 1$ for $r \in [m]$ and $\|x_i\|_2 = 1$ for $i \in [n]$, the third step follows from Cauchy-Schwartz inequality, and the last step follows from (28.7).

Integrating the gradient, we can bound the distance from the initialization

$$\begin{aligned}\|w_r(t) - w_r(0)\|_2 &\le \int_0^t \left\| \frac{\mathrm{d}}{\mathrm{d}s} w_r(s) \right\|_2 \mathrm{d}s \\ &\le \frac{\sqrt{n}\|y - u(0)\|_2}{\sqrt{m}\lambda}.\end{aligned}$$

□

Lemma 28.4.4 (Lemma 3.4 in [DZPS19]). *If $D_{\mathrm{cts}} < R$. then for all $t \ge 0$, $\lambda_{\min}(H(t)) \ge \frac{1}{2}\lambda$. Moreover, for all $r \in [m]$,*

$$\|w_r(t) - w_r(0)\| \le D_{\mathrm{cts}},$$

and

$$\|y - u(t)\|_2^2 \le \exp(-\lambda t) \cdot \|y - u(0)\|_2^2.$$

For the completeness, we still provide a proof.

Proof. Assume the conclusion does not hold at time t. We argue there must be some $s \le t$ so that $\lambda_{\min}(H(s)) < \frac{1}{2}\lambda$.

If $\lambda_{\min}(H(t)) < \frac{1}{2}\lambda$, then we can simply take $s = t$.

Otherwise since the conclusion does not hold, there exists r so that

$$\|w_r(t) - w_r(0)\| \ge D_{\mathrm{cts}} \quad \text{or} \quad \|y - u(t)\|_2^2 > \exp(-\lambda t)\|y - u(0)\|_2^2.$$

Then by Lemma 28.4.3, there exists $s \le t$ such that

$$\lambda_{\min}(H(s)) < \frac{1}{2}\lambda.$$

By Lemma 28.4.2, there exists $t_0 > 0$ defined as

$$t_0 = \inf \left\{ t > 0 : \max_{r \in [m]} \|w_r(t) - w_r(0)\|_2^2 \geq R \right\}.$$

Thus at time t_0, there exists $r \in [m]$ satisfying $\|w_r(t_0) - w_r(0)\|_2^2 = R$.

By Lemma 28.4.2,

$$\lambda_{\min}(H(t')) \geq \frac{1}{2}\lambda, \forall t' \leq t_0.$$

However, by Lemma 28.4.3, this implies

$$\|w_r(t_0) - w_r(0)\|_2 \leq D_{\text{cts}} < R,$$

which is a contradiction. □

28.4.4 Convergence

Theorem 28.4.5. *Recall that* $\lambda = \lambda_{\min}(H^{\text{cts}}) > 0$. *Let* $m = \Omega(\lambda^{-4} n^4 \log(n/\delta))$, *we i.i.d. initialize* $w_r \in \mathcal{N}(0, I)$, a_r *sampled from* $\{-1, +1\}$ *uniformly at random for* $r \in [m]$, *and we set the step size* $\eta = O(\lambda/n^2)$ *then with probability at least* $1-\delta$ *over the random initialization we have for* $k = 0, 1, 2, \cdots$

$$\|u(k) - y\|_2^2 \leq (1 - \eta\lambda/2)^k \cdot \|u(0) - y\|_2^2. \tag{28.9}$$

Correctness We prove Theorem 28.4.5 by induction. The base case is $i = 0$ and it is trivially true. Assume for $i = 0, \cdots, k$ we have proved (28.9) to be true. We want to show (28.9) holds for $i = k + 1$.

From the induction hypothesis, we have the following Lemma stating that the weights should not change too much.

Lemma 28.4.6 (Corollary 4.1 in [DZPS19])**.** *If* (28.9) *holds for* $i = 0, \cdots, k$, *then we have for all* $r \in [m]$

$$\|w_r(k+1) - w_r(0)\|_2 \leq \frac{4\sqrt{n}\|y - u(0)\|_2}{\sqrt{m}\lambda} := D$$

For the completeness, we still provide the proof

Proof. We use the norm of gradient to bound this distance,

$$\begin{aligned}
\|w_r(k+1) - w_r(0)\|_2 &\leq \eta \sum_{i=0}^{k} \left\| \frac{\partial L(W(i))}{\partial w_r(i)} \right\|_2 \\
&\leq \eta \sum_{i=0}^{k} \frac{\sqrt{n}\|y - u(i)\|_2}{\sqrt{m}} \\
&\leq \eta \sum_{i=0}^{k} \frac{\sqrt{n}(1-\eta\lambda/2)^{i/2}}{\sqrt{m}} \|y - u(0)\|_2 \\
&\leq \eta \sum_{i=0}^{\infty} \frac{\sqrt{n}(1-\eta\lambda/2)^{i/2}}{\sqrt{m}} \|y - u(0)\|_2 \\
&= \frac{4\sqrt{n}\|y - u(0)\|_2}{\sqrt{m}\lambda},
\end{aligned}$$

where the first step follows from (28.2), the second step follows from (28.8), the third step follows from the induction hypothesis, the fourth step relaxes the summation to an infinite summation, and the last step follows from $\sum_{i=0}^{\infty}(1-\eta\lambda/2)^{i/2} = \frac{2}{\eta\lambda}$.

Thus, we complete the proof. □

Next, we calculate the different of predictions between two consecutive iterations, analogue to $\frac{\mathrm{d}u_i(t)}{\mathrm{d}t}$ term in Lemma 28.4.3. For each $i \in [n]$, we have

$$\begin{aligned}
u_i(k+1) - u_i(k) &= \frac{1}{\sqrt{m}} \sum_{r=1}^{m} a_r \cdot \big(\phi(w_r(k+1)^\top x_i) - \phi(w_r(k)^\top x_i)\big) \\
&= \frac{1}{\sqrt{m}} \sum_{r=1}^{m} a_r \cdot \left(\phi\left(\left(w_r(k) - \eta \frac{\partial L(W(k))}{\partial w_r(k)} \right)^\top x_i \right) - \phi(w_r(k)^\top x_i) \right).
\end{aligned}$$

Here we divide the right hand side into two parts. $v_{1,i}$ represents the terms that the pattern does not change and $v_{2,i}$ represents the term that pattern may changes. For each

$i \in [n]$, we define $v_{1,i}$ and $v_{2,i}$ as follows

$$v_{1,i} := \frac{1}{\sqrt{m}} \sum_{r \in S_i} a_r \cdot \left(\phi \left(\left(w_r(k) - \eta \frac{\partial L(W(k))}{\partial w_r(k)} \right)^\top x_i \right) - \phi(w_r(k)^\top x_i) \right),$$
$$v_{2,i} := \frac{1}{\sqrt{m}} \sum_{r \in \overline{S}_i} a_r \cdot \left(\phi \left(\left(w_r(k) - \eta \frac{\partial L(W(k))}{\partial w_r(k)} \right)^\top x_i \right) - \phi(w_r(k)^\top x_i) \right).$$

Thus, we can rewrite $u(k+1) - u(k) \in \mathbb{R}^n$ in the following sense

$$u(k+1) - u(k) = v_1 + v_2.$$

In order to analyze $v_1 \in \mathbb{R}^n$, we provide definition of H and $H^\perp \in \mathbb{R}^{n \times n}$ first,

$$H(k)_{i,j} = \frac{1}{m} \sum_{r=1}^m x_i^\top x_j \mathbf{1}_{w_r(k)^\top x_i \geq 0, w_r(k)^\top x_j \geq 0},$$
$$H(k)^\perp_{i,j} = \frac{1}{m} \sum_{r \in \overline{S}_i} x_i^\top x_j \mathbf{1}_{w_r(k)^\top x_i \geq 0, w_r(k)^\top x_j \geq 0}.$$

Then, we can rewrite $v_{1,i} \in \mathbb{R}$

$$\begin{aligned} v_{1,i} = & -\frac{\eta}{m} \sum_{j=1}^n x_i^\top x_j (u_j - y_j) \sum_{r \in S_i} \mathbf{1}_{w_r(k)^\top x_i \geq 0, w_r(k)^\top x_j \geq 0} \\ = & -\eta \sum_{j=1}^n (u_j - y_j)(H_{i,j}(k) - H^\perp_{i,j}(k)), \end{aligned}$$

which means vector $v_1 \in \mathbb{R}^n$ can be written as

$$v_1 = \eta (y - u(k))^\top (H(k) - H^\perp(k)). \tag{28.10}$$

We are ready to prove the induction hypothesis. We can rewrite $\|y - u(k+1)\|_2^2$ as follows:

$$\begin{aligned} \|y - u(k+1)\|_2^2 = & \|y - u(k) - (u(k+1) - u(k))\|_2^2 \\ = & \|y - u(k)\|_2^2 - 2(y - u(k))^\top (u(k+1) - u(k)) + \|u(k+1) - u(k)\|_2^2. \end{aligned}$$

We can rewrite the second term in the above Equation in the following sense,

$$\begin{aligned}
&(y-u(k))^\top (u(k+1)-u(k)) \\
=\ &(y-u(k))^\top (v_1+v_2) \\
=\ &(y-u(k))^\top v_1 + (y-u(k))^\top v_2 \\
=\ &\eta (y-u(k))^\top H(k)(y-u(k)) - \eta (y-u(k))^\top H(k)^\perp (y-u(k)) + (y-u(k))^\top v_2,
\end{aligned}$$

where the third step follows from Eq. (28.10).

We define

$$\begin{aligned}
C_1 =\ & -2\eta (y-u(k))^\top H(k)(y-u(k)), \\
C_2 =\ & 2\eta (y-u(k))^\top H(k)^\perp (y-u(k)), \\
C_3 =\ & -2(y-u(k))^\top v_2, \\
C_4 =\ & \|u(k+1)-u(k)\|_2^2.
\end{aligned}$$

Thus, we have

$$\begin{aligned}
\|y-u(k+1)\|_2^2 &= \|y-u(k)\|_2^2 + C_1 + C_2 + C_3 + C_4 \\
&\leq \|y-u(k)\|_2^2 (1 - \eta\lambda + 8\eta n R + 8\eta n R + \eta^2 n^2),
\end{aligned}$$

where the last step follows from Claim 28.4.8, 28.4.9, 28.4.10 and 28.4.11, which we will prove given later.

Choice of η and R. Next, we want to choose η and R such that

$$(1 - \eta\lambda + 8\eta n R + 8\eta n R + \eta^2 n^2) \leq (1 - \eta\lambda/2). \tag{28.11}$$

If we set $\eta = \frac{\lambda}{4n^2}$ and $R = \frac{\lambda}{64n}$, we have

$$8\eta n R + 8\eta n R = 16\eta n R \leq \eta\lambda/4, \quad \text{and} \quad \eta^2 n^2 \leq \eta\lambda/4.$$

This implies

$$\|y - u(k+1)\|_2^2 \leq \|y - u(k)\|_2^2 \cdot (1 - \eta\lambda/2)$$

holds with probability at least $1 - 2n\exp(-mR)$.

Over-parameterization size, lower bound on m. We require

$$D = \frac{4\sqrt{n}\|y - u(0)\|_2}{\sqrt{m}\lambda} < R = \frac{\lambda}{64n},$$

and

$$2n\exp(-mR) \leq \delta.$$

By Claim 28.4.7, it is sufficient to choose $m = \Omega(\lambda^{-4} n^4 \log(m/\delta)\log^2(n/\delta))$.

28.4.5 Technical claims

Claim 28.4.7. *For $0 < \delta < 1$, with probability at least $1 - \delta$,*

$$\|y - u(0)\|_2^2 = O(n \log(m/\delta) \log^2(n/\delta)).$$

Proof.

$$\begin{aligned}
\|y - u(0)\|_2^2 = & \sum_{i=1}^n (y_i - f(W(0), a, x_i))^2 \\
= & \sum_{i=1}^n \Big(y_i - \frac{1}{\sqrt{m}} \sum_{r=1}^m a_r \phi(w_r^\top x_i) \Big)^2 \\
= & \sum_{i=1}^n y_i^2 - 2 \sum_{i=1}^n \frac{y_i}{\sqrt{m}} \sum_{r=1}^m a_r \phi(w_r^\top x_i) + \sum_{i=1}^n \frac{1}{m} \Big(\sum_{r=1}^m a_r \phi(w_r^\top x_i) \Big)^2.
\end{aligned}$$

Fix $r \in [m]$ and $i \in [n]$. Since $w_r \sim N(0, I)$ and $\|x_i\|_2 = 1$, $w_r^\top x_i$ follows distribution $N(0, 1)$. From concentration of Gaussian distribution, we have

$$\Pr_{w_r}[w_r^\top x_i \geq \sqrt{2 \log(2mn/\delta)}] \leq \frac{\delta}{2mn}.$$

Let E_1 be the even that for all $r \in [m]$ and $i \in [n]$ we have

$$\phi(w_r^\top x_i) \leq \sqrt{2 \log(2mn/\delta)}.$$

Then by union bound, $\Pr[E_1] \geq 1 - \frac{\delta}{2}$,

Fix $i \in [n]$. For every $r \in [m]$, we define random variable $z_{i,r}$ as

$$z_{i,r} := \frac{1}{\sqrt{m}} \cdot a_r \cdot \phi(w_r^\top x_i) \cdot \mathbf{1}_{w_r^\top x_i \leq \sqrt{2 \log(2mn/\delta)}}.$$

Then $z_{i,r}$ only depends on $a_r \in \{-1,1\}$ and $w_r \sim N(0,I)$. Notice that $\mathbb{E}_{a_r,w_r}[z_{i,r}] = 0$, and $|z_{i,r}| \leq \sqrt{2\log(2mn/\delta)}$. Moreover,

$$\begin{aligned}
\mathop{\mathbb{E}}_{a_r,w_r}[z_{i,r}^2] &= \mathop{\mathbb{E}}_{a_r,w_r}\left[\frac{1}{m}a_r^2\phi^2(w_r^\top x_i)\mathbf{1}^2_{w_r^\top x_i \leq \sqrt{2\log(2mn/\delta)}}\right] \\
&= \frac{1}{m}\mathop{\mathbb{E}}_{a_r}[a_r^2]\cdot \mathop{\mathbb{E}}_{w_r}\left[\phi^2(w_r^\top x_i)\mathbf{1}^2_{w_r^\top x_i \leq \sqrt{2\log(2mn/\delta)}}\right] \\
&\leq \frac{1}{m}\cdot 1 \cdot \mathop{\mathbb{E}}_{w_r}[(w_r^\top x_i)^2] \\
&= \frac{1}{m},
\end{aligned}$$

where the second step uses independence between a_r and w_r, the third step uses $a_r \in \{-1,1\}$ and $\phi(t) = \max\{t,0\}$, and the last step follows from $w_r^\top x_i \sim N(0,1)$.

Now we are ready to apply Bernstein inequality (Lemma A.1.2) to get for all $t > 0$,

$$\Pr\left[\sum_{r=1}^m z_{i,r} > t\right] \leq \exp\left(-\frac{t^2/2}{m\cdot\frac{1}{m} + \sqrt{2\log(2mn/\delta)}\cdot t/3}\right).$$

Setting $t = \sqrt{2\log(2mn/\delta)}\cdot\log(4n/\delta)$, we have with probability at least $1 - \frac{\delta}{4n}$,

$$\sum_{r=1}^m z_{i,r} \leq \sqrt{2\log(2mn/\delta)}\cdot\log(4n/\delta).$$

Notice that we can also apply Bernstein inequality (Lemma A.1.2) on $-z_{i,r}$ to get

$$\Pr\left[\sum_{r=1}^m z_{i,r} < -t\right] \leq \exp\left(-\frac{t^2/2}{m\cdot\frac{1}{m} + \sqrt{2\log(2mn/\delta)}\cdot t/3}\right).$$

Let E_2 be the event that for all $i \in [n]$,

$$\left|\sum_{r=1}^m z_{i,r}\right| \leq \sqrt{2\log(2mn/\delta)}\cdot\log(4n/\delta).$$

By applying union bound on all $i \in [n]$, we have $\Pr[E_2] \geq 1 - \frac{\delta}{2}$.

If both E_1 and E_2 happen, we have

$$\begin{aligned}
\|y - u(0)\|_2^2 &= \sum_{i=1}^n y_i^2 - 2\sum_{i=1}^n \frac{y_i}{\sqrt{m}} \sum_{r=1}^m a_r \phi(w_r^\top x_i) + \sum_{i=1}^n \frac{1}{m} \Big(\sum_{r=1}^m a_r \phi(w_r^\top x_i) \Big)^2 \\
&= \sum_{i=1}^n y_i^2 - 2\sum_{i=1}^n y_i \sum_{r=1}^m z_{i,r} + \sum_{i=1}^n \Big(\sum_{r=1}^m z_{i,r} \Big)^2 \\
&\leq \sum_{i=1}^n y_i^2 + 2\sum_{i=1}^n |y_i| \sqrt{2\log(2mn/\delta)} \cdot \log(4n/\delta) + \sum_{i=1}^n \Big(\sqrt{2\log(2mn/\delta)} \cdot \log(4n/\delta) \Big)^2 \\
&= O(n \log(m/\delta) \log^2(n/\delta)),
\end{aligned}$$

where the second step uses E_1, the third step uses E_2, and the last step follows from $|y_i| = O(1), \forall i \in [n]$.

By union bound, this will happen with probability at least $1 - \delta$. □

Claim 28.4.8. *Let $C_1 = -2\eta(y - u(k))^\top H(k)(y - u(k))$. We have*

$$C_1 \leq -\|y - u(k)\|_2^2 \cdot \eta\lambda.$$

Proof. By Lemma 28.4.2 and our choice of $R < \frac{\lambda}{8n}$, We have $\|H(0) - H(k)\|_F \leq 2n \cdot \frac{\lambda}{8n} = \frac{\lambda}{4}$. Recall that $\lambda = \lambda_{\min}(H(0))$. Therefore

$$\lambda_{\min}(H(k)) \geq \lambda_{\min}(H(0)) - \|H(0) - H(k)\| \geq \lambda/2.$$

Then we have

$$(y - u(k))^\top H(k)(y - u(k)) \geq \|y - u(k)\|_2^2 \cdot \lambda/2.$$

Thus, we complete the proof. □

Claim 28.4.9. *Let* $C_2 = 2\eta(y - u(k))^\top H(k)^\perp (y - u(k))$. *We have*

$$C_2 \leq \|y - u(k)\|_2^2 \cdot 8\eta n R.$$

holds with probability $1 - n\exp(-mR)$.

Proof. Note that

$$C_2 \leq 2\eta \|y - u(k)\|_2^2 \|H(k)^\perp\|.$$

It suffices to upper bound $\|H(k)^\perp\|$. Since $\|\cdot\| \leq \|\cdot\|_F$, then it suffices to upper bound $\|\cdot\|_F$.

For each $i \in [n]$, we define y_i as follows

$$y_i = \sum_{r=1}^{m} \mathbf{1}_{r \in \overline{S}_i}.$$

Then we have

$$\begin{aligned}
\|H(k)^{\perp}\|_F^2 &= \sum_{i=1}^{n}\sum_{j=1}^{n}(H(k)_{i,j}^{\perp})^2 \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n}\Big(\frac{1}{m}\sum_{r\in\overline{S}_i} x_i^\top x_j \mathbf{1}_{w_r(k)^\top x_i\geq 0, w_r(k)^\top x_j\geq 0}\Big)^2 \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n}\Big(\frac{1}{m}\sum_{r=1}^{m} x_i^\top x_j \mathbf{1}_{w_r(k)^\top x_i\geq 0, w_r(k)^\top x_j\geq 0}\cdot \mathbf{1}_{r\in\overline{S}_i}\Big)^2 \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n}(\frac{x_i^\top x_j}{m})^2\Big(\sum_{r=1}^{m} \mathbf{1}_{w_r(k)^\top x_i\geq 0, w_r(k)^\top x_j\geq 0}\cdot \mathbf{1}_{r\in\overline{S}_i}\Big)^2 \\
&\leq \frac{1}{m^2}\sum_{i=1}^{n}\sum_{j=1}^{n}\Big(\sum_{r=1}^{m} \mathbf{1}_{w_r(k)^\top x_i\geq 0, w_r(k)^\top x_j\geq 0}\cdot \mathbf{1}_{r\in\overline{S}_i}\Big)^2 \\
&= \frac{n}{m^2}\sum_{i=1}^{n}\Big(\sum_{r=1}^{m} \mathbf{1}_{r\in\overline{S}_i}\Big)^2 \\
&= \frac{n}{m^2}\sum_{i=1}^{n} y_i^2.
\end{aligned}$$

Fix $i \in [n]$. The plan is to use Bernstein inequality to upper bound y_i with high probability.

First by Eq. (28.6) we have

$$\mathbb{E}[\mathbf{1}_{r\in\overline{S}_i}] \leq R.$$

We also have

$$\begin{aligned}
\mathbb{E}\left[(\mathbf{1}_{r\in\overline{S}_i} - \mathbb{E}[\mathbf{1}_{r\in\overline{S}_i}])^2\right] &= \mathbb{E}[\mathbf{1}_{r\in\overline{S}_i}^2] - \mathbb{E}[\mathbf{1}_{r\in\overline{S}_i}]^2 \\
&\leq \mathbb{E}[\mathbf{1}_{r\in\overline{S}_i}^2] \\
&\leq R.
\end{aligned}$$

Finally we have $|\mathbf{1}_{r\in\overline{S}_i} - \mathbb{E}[\mathbf{1}_{r\in\overline{S}_i}]| \leq 1$.

Notice that $\{\mathbf{1}_{r\in\overline{S}_i}\}_{r=1}^m$ are mutually independent, since $\mathbf{1}_{r\in\overline{S}_i}$ only depends on $w_r(0)$. Hence from Bernstein inequality (Lemma A.1.2) we have for all $t > 0$,

$$\Pr[y_i > m \cdot R + t] \leq \exp\left(-\frac{t^2/2}{m \cdot R + t/3}\right).$$

By setting $t = 3mR$, we have

$$\Pr[y_i > 4mR] \leq \exp(-mR).$$

Hence by union bound, with probability at least $1 - n\exp(-mR)$,

$$\|H(k)^\perp\|_F^2 \leq \frac{n}{m^2} \cdot n \cdot (4mR)^2 = 16n^2R^2.$$

Putting all together we have

$$\|H(k)^\perp\| \leq \|H(k)^\perp\|_F \leq 4nR$$

with probability at least $1 - n\exp(-mR)$.

□

Claim 28.4.10. *Let $C_3 = -2(y - u(k))^\top v_2$. Then we have*

$$C_3 \leq \|y - u(k)\|_2^2 \cdot 8\eta nR.$$

with probability at least $1 - n\exp(-mR)$.

Proof. We have

$$\text{LHS} \leq 2\|y - u(k)\|_2 \cdot \|v_2\|_2.$$

We can upper bound $\|v_2\|_2$ in the following sense

$$\begin{aligned}
\|v_2\|_2^2 &\leq \sum_{i=1}^n \left(\frac{\eta}{\sqrt{m}} \sum_{r \in \overline{S}_i} \left| (\frac{\partial L(W(k))}{\partial w_r(k)})^\top x_i \right| \right)^2 \\
&= \frac{\eta^2}{m} \sum_{i=1}^n \left(\sum_{r=1}^m \mathbf{1}_{r \in \overline{S}_i} \left| (\frac{\partial L(W(k))}{\partial w_r(k)})^\top x_i \right| \right)^2 \\
&\leq \frac{\eta^2}{m} \cdot \max_{r \in [m]} \left| \frac{\partial L(W(k))}{\partial w_r(k)} \right|^2 \cdot \sum_{i=1}^n \left(\sum_{r=1}^m \mathbf{1}_{r \in \overline{S}_i} \right)^2 \\
&\leq \frac{\eta^2}{m} \cdot (\frac{\sqrt{n}}{\sqrt{m}} \|u(k) - y\|_2)^2 \cdot \sum_{i=1}^n \left(\sum_{r=1}^m \mathbf{1}_{r \in \overline{S}_i} \right)^2 \\
&\leq \frac{\eta^2}{m} \cdot (\frac{\sqrt{n}}{\sqrt{m}} \|u(k) - y\|_2)^2 \cdot \sum_{i=1}^n (4mR)^2 \\
&= 16 n^2 R^2 \eta^2 \|u(k) - y\|_2^2,
\end{aligned}$$

where the first step follows from definition of v_2, the fourth step follows from $\max_{r \in [m]} |\frac{\partial L(W(k))}{\partial w_r(k)}| \leq \frac{\sqrt{n}}{\sqrt{m}} \cdot \|u(k) - y\|_2$, the fifth step follows from $\sum_{r=1}^m \mathbf{1}_{r \in \overline{S}_i} \leq 4mR$ with probability at least $1 - \exp(-mR)$. □

Claim 28.4.11. *Let $C_4 = \|u(k+1) - u(k)\|_2^2$. Then we have*

$$C_4 \leq \eta^2 n^2 \|y - u(k)\|_2^2.$$

Proof. We have

$$\begin{aligned}
\text{LHS} &\leq \eta^2 \sum_{i=1}^n \frac{1}{m} \left(\sum_{r=1}^m \left\| \frac{\partial L(W(k))}{\partial w_r(k)} \right\|_2 \right)^2 \\
&\leq \eta^2 n^2 \|y - u(k)\|_2^2.
\end{aligned}$$

□

28.5 Cubic Suffices

We prove a more general version of Lemma 28.4.1 in this section.

Theorem 28.5.1 (Data-dependent version, bounding the difference between discrete and continuous)**.** *Let H^{cts} and H^{dis} be defined as Definition 28.1.1. Let λ, α, β be satisfied Assumption 28.1.1. If*

$$m = \Omega((\lambda^{-2}\beta + \lambda^{-1}\alpha)\log(n/\delta)),$$

we have

$$\|H^{\mathrm{dis}} - H^{\mathrm{cts}}\|_2 \leq \lambda/4, \text{ and } \lambda_{\min}(H^{\mathrm{dis}}) \geq \frac{3}{4}\lambda$$

holds with probability at least $1 - \exp(-\Omega(\log(n/\delta)))$.

Proof. Recall the definition, we know

$$H^{\mathrm{cts}} = \mathop{\mathbb{E}}_{w}[H(w)], \quad \text{and} \quad H^{\mathrm{dis}} = \frac{1}{m}\sum_{r=1}^{m} H(w_r).$$

We define matrix $Y_r = H(w_r) - \mathbb{E}_w[H(w)]$. We know that, Y_r are all independent,

$$\mathbb{E}[Y_r] = 0, \quad \|Y_r\| \leq \alpha, \quad \Big\| \sum_{r=1}^{m} \mathbb{E}[Y_r Y_r^\top] \Big\| \leq m\beta.$$

Let $Y = \sum_{r=1}^{m} Y_r$. We apply Matrix Bernstein inequality (Lemma A.2.2) with $t = \sqrt{m\beta\log(n/\delta)} + \alpha\log(n/\delta)$,

$$\begin{aligned}
\Pr[\|Y\| \geq t] &\leq 2n\exp\Big(-\frac{t^2/2}{m\beta + \alpha t/3}\Big) \\
&\leq 2n\exp(-\log(n/\delta)) \\
&\leq \exp(-\Omega(\log(n/\delta))).
\end{aligned}$$

Thus, we have

$$\Pr\left[\left\|\frac{1}{m}\sum_{r=1}^{m} Y_r\right\| \geq \frac{1}{m}(\sqrt{m\beta\log(n/\delta)} + \alpha\log(n/\delta))\right] \leq \exp(-\Omega(\log(n/\delta))).$$

In order to guarantee that $\frac{1}{m}(\sqrt{m\beta\log(n/\delta)} + \alpha\log(n/\delta)) \leq \lambda$, we need

$$\sqrt{m} \geq \lambda^{-1}\sqrt{\beta\log(n/\delta)}$$

when the first term is the dominated one; we need

$$m \geq \lambda^{-1}\alpha\log(n/\delta).$$

Overall, we need

$$m \geq \Omega((\lambda^{-2}\beta + \lambda^{-1}\alpha)\log(n/\delta)).$$

Thus, we complete the proof. □

Lemma 28.5.2 (Stronger version of Lemma 3.3 in [DZPS19])**.** *Let Part 4 in Assumption 28.1.1 hold. Let* $D_{\mathrm{cts}} = \frac{\sqrt{\alpha}\|y-u(0)\|_2}{\sqrt{m\lambda}}$*. Suppose for* $0 \leq s \leq t$*,* $\lambda_{\min}(H(s)) \geq \lambda/2$*. Then we have*

$$\|y - u(t)\|_2^2 \leq \exp(-\lambda t)\cdot\|y - u(0)\|_2^2,$$

and

$$\|w_r(t) - w_r(0)\|_2 \leq D_{\mathrm{cts}}.$$

Proof. Recall we can write the dynamics of predictions as

$$\frac{\mathrm{d}}{\mathrm{d}t}u(t) = H(t)\cdot(y - u(t)).$$

Table 28.3: Table of Parameters for the $m = \widetilde{\Omega}(n^3)$ result in Section 28.5. **Nt.** stands for notations.

Nt.	**Choice**	**Place**	**Comment**
λ	$:= \lambda_{\min}(H^{\mathrm{cts}})$	Part 1 of Ass. 28.1.1	Data-dependent
α	Absolute	Part 2 of Ass. 28.1.1	Data-dependent
β	Variance	Part 3 of Ass. 28.1.1	Data-dependent
R	λ/n	Eq. (28.11)	Maximal allowed movement of weight
D_{cts}	$\frac{\sqrt{\alpha}\\|y-u(0)\\|_2}{\sqrt{m}\lambda}$	Lemma 28.5.2	Actual moving distance, continuous case
D	$\frac{4\sqrt{\alpha}\\|y-u(0)\\|_2}{\sqrt{m}\lambda}$	Theorem 28.5.5	Actual moving distance, discrete case
η	$\lambda/(\alpha n)$	Eq. (28.11)	Step size of gradient descent
m	$(\lambda^{-2}\beta + \lambda^{-1}\alpha)\log(n/\delta)$	Theorem 28.5.1	Bounding discrete and continuous
m	$\lambda^{-4}\alpha n^3 \log^3(n/\delta)$	Lemma 28.4.4 Claim 28.4.7	$D < R$ and $\\|y - u(0)\\|_2^2 = \widetilde{O}(n)$

We can calculate the loss function dynamics

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t}\|y-u(t)\|_2^2 = & -2(y-u(t))^\top \cdot H(t) \cdot (y-u(t)) \\ \leq & -\lambda \|y-u(t)\|_2^2. \end{aligned}$$

Thus we have $\frac{\mathrm{d}}{\mathrm{d}t}(\exp(\lambda t)\|y-u(t)\|_2^2) \leq 0$ and $\exp(\lambda t)\|y-u(t)\|_2^2$ is a decreasing function with respect to t.

Using this fact we can bound the loss

$$\|y-u(t)\|_2^2 \leq \exp(-\lambda t)\|y-u(0)\|_2^2.$$

Therefore, $u(t) \to y$ exponentially fast.

Now, we can bound the gradient norm. Recall for $0 \leq s \leq t$,

$$\left\| \frac{\mathrm{d}}{\mathrm{d}s} w_r(s) \right\|_2 = \left\| \sum_{i=1}^n (y_i - u_i) \frac{1}{\sqrt{m}} a_r x_i \cdot \mathbf{1}_{w_r(s)^\top x_i \geq 0} \right\|_2.$$

Define matrix $X_r \in \mathbb{R}^{d\times n}$ by setting the i-th column to be $\mathbf{1}_{w_r(s)^\top x_i \geq 0} \cdot x_i$, then $X_r^\top X_r = H(w_r(s))$, where $H(\cdot)$ is the matrix defined in Definition 28.1.1. Then we have $\|X_r^\top X_r\|_2 \leq \alpha$ by Part 4 in Assumption 28.1.1, which leads to $\|X_r\|_2 \leq \sqrt{\alpha}$. So we have

$$\begin{aligned}
\left\| \frac{\mathrm{d}}{\mathrm{d}s} w_r(s) \right\|_2 &= \frac{1}{\sqrt{m}} \|X_r(y - u(s))\|_2 \\
&\leq \frac{1}{\sqrt{m}} \|X_r\|_2 \|(y - u(s))\|_2 \\
&\leq \frac{\sqrt{\alpha}}{\sqrt{m}} \|y - u(s)\|_2 \\
&\leq \frac{\sqrt{\alpha}}{\sqrt{m}} \exp(-\lambda s) \|y - u(0)\|_2.
\end{aligned} \tag{28.12}$$

Integrating the gradient, we can bound the distance from the initialization

$$\begin{aligned}
\|w_r(t) - w_r(0)\|_2 \leq & \int_0^t \left\| \frac{\mathrm{d}}{\mathrm{d}s} w_r(s) \right\|_2 \mathrm{d}s \\
\leq & \frac{\sqrt{\alpha} \|y - u(0)\|_2}{\sqrt{m}\lambda}.
\end{aligned}$$

□

28.5.1 Technical claims

Claim 28.5.3. *Let* $C_3 = -2(y - u(k))^\top v_2$. *Then we have*

$$C_3 \le \|y - u(k)\|_2^2 \cdot 8\eta(\alpha n)^{1/2} R.$$

with probability at least $1 - n\exp(-mR)$.

Proof. We have

$$\text{LHS} \le 2\|y - u(k)\|_2 \cdot \|v_2\|_2.$$

We can upper bound $\|v_2\|_2$ in the following sense

$$\begin{aligned}
\|v_2\|_2^2 &\le \sum_{i=1}^n \left(\frac{\eta}{\sqrt{m}} \sum_{r \in \overline{S}_i} \left| \Big(\frac{\partial L(W(k))}{\partial w_r(k)} \Big)^\top x_i \right| \right)^2 \\
&= \frac{\eta^2}{m} \sum_{i=1}^n \left(\sum_{r=1}^m \mathbf{1}_{r \in \overline{S}_i} \left| \Big(\frac{\partial L(W(k))}{\partial w_r(k)} \Big)^\top x_i \right| \right)^2 \\
&\le \frac{\eta^2}{m} \cdot \max_{r \in [m]} \left| \frac{\partial L(W(k))}{\partial w_r(k)} \right|^2 \cdot \sum_{i=1}^n \left(\sum_{r=1}^m \mathbf{1}_{r \in \overline{S}_i} \right)^2 \\
&\le \frac{\eta^2}{m} \cdot \Big(\frac{\sqrt{\alpha}}{\sqrt{m}} \|u(k) - y\|_2 \Big)^2 \cdot \sum_{i=1}^n \left(\sum_{r=1}^m \mathbf{1}_{r \in \overline{S}_i} \right)^2 \\
&\le \frac{\eta^2}{m} \cdot \Big(\frac{\sqrt{\alpha}}{\sqrt{m}} \|u(k) - y\|_2 \Big)^2 \cdot \sum_{i=1}^n (4mR)^2 \\
&= 16\alpha n R^2 \eta^2 \|u(k) - y\|_2^2,
\end{aligned}$$

where the first step follows from definition of v_2, the fourth step follows from (28.12) and

$$\begin{aligned}
\max_{r \in [m]} \left| \frac{\partial L(W(k))}{\partial w_r(k)} \right| &= \max_{r \in [m]} \left| \frac{\mathrm{d} w_r(k)}{\mathrm{d} k} \right| \\
&\le \frac{\sqrt{\alpha}}{\sqrt{m}} \|y - u(k)\|_2,
\end{aligned}$$

the fifth step follows from $\sum_{r=1}^{m} \mathbf{1}_{r \in \overline{S}_i} \leq 4mR$ with probability at least $1 - \exp(-mR)$. $\square$

Claim 28.5.4. *Let $C_4 = \|u(k+1) - u(k)\|_2^2$. Then we have*

$$C_4 \leq \eta^2 \alpha n \|y - u(k)\|_2^2.$$

Proof. We have

$$\begin{aligned}
\text{LHS} &\leq \eta^2 \sum_{i=1}^{n} \frac{1}{m} \left(\sum_{r=1}^{m} \left\| \frac{\partial L(W(k))}{\partial w_r(k)} \right\|_2 \right)^2 \\
&\leq \eta^2 \sum_{i=1}^{n} \frac{1}{m} \left(\sum_{r=1}^{m} \frac{\sqrt{\alpha}}{\sqrt{m}} \|u(k) - y\|_2 \right)^2 \\
&\leq \eta^2 \alpha n \|y - u(k)\|_2^2.
\end{aligned}$$

$\square$

28.5.2 Main result

Theorem 28.5.5. *Recall that $\lambda = \lambda_{\min}(H^{\text{cts}}) > 0$. Let $m = \Omega(\lambda^{-4} n^3 \alpha \log^3(n/\delta))$, we i.i.d. initialize $w_r \in \mathcal{N}(0, I)$, a_r sampled from $\{-1, +1\}$ uniformly at random for $r \in [m]$, and we set the step size $\eta = O(\lambda/(\alpha n))$ then with probability at least $1 - \delta$ over the random initialization we have for $k = 0, 1, 2, \cdots$*

$$\|u(k) - y\|_2^2 \leq (1 - \eta\lambda/2)^k \cdot \|u(0) - y\|_2^2. \tag{28.13}$$

Proof. This proof, similar to the proof of Theorem 28.4.5, is again by induction. (28.13) trivially holds when $k = 0$, which is the base case.

If (28.13) holds for $k' = 0, \cdots, k$, then we claim that for all $r \in [m]$

$$\|w_r(k+1) - w_r(0)\|_2 \leq \frac{4\sqrt{\alpha}\|y - u(0)\|_2}{\sqrt{m}\lambda} := D \tag{28.14}$$

To see this, we use the norm of gradient to bound this distance,

$$\begin{aligned}
\|w_r(k+1) - w_r(0)\|_2 &\leq \eta \sum_{k'=0}^{k} \left\| \frac{\partial L(W(k'))}{\partial w_r(k')} \right\|_2 \\
&\leq \eta \sum_{k'=0}^{k} \frac{\sqrt{\alpha}\|y - u(k')\|_2}{\sqrt{m}} \\
&\leq \eta \sum_{k'=0}^{k} \frac{\sqrt{\alpha}(1 - \eta\lambda/2)^{k'/2}}{\sqrt{m}} \|y - u(0)\|_2 \\
&\leq \eta \sum_{k'=0}^{\infty} \frac{\sqrt{\alpha}(1 - \eta\lambda/2)^{k'/2}}{\sqrt{m}} \|y - u(0)\|_2 \\
&= \frac{4\sqrt{\alpha}\|y - u(0)\|_2}{\sqrt{m}\lambda},
\end{aligned}$$

where the first step follows from (28.2), the second step follows from (28.12), the third step follows from the induction hypothesis, the fourth step relaxes the summation to an infinite summation, and the last step follows from $\sum_{k'=0}^{\infty}(1-\eta\lambda/2)^{k'/2} = \frac{2}{\eta\lambda}$.

Then from Claim 28.5.4, it is sufficient to choose $\eta = \frac{\lambda}{4\alpha n}$ so that (28.13) holds for $k' = k+1$. This completes the induction step.

Over-parameterization size, lower bound on m.

We require

$$D = \frac{4\sqrt{\alpha}\|y-u(0)\|_2}{\sqrt{m}\lambda} < R = \frac{\lambda}{64n},$$

and

$$2n\exp(-mR) \leq \delta.$$

This implies that

$$\begin{aligned} m &= \Omega(\lambda^{-4}n^2\alpha\|y-u(0)\|_2^2) \\ &= \Omega(\lambda^{-4}n^3\alpha\log(m/\delta)\log^2(n/\delta)), \end{aligned}$$

where the last step follows from Claim 28.4.7. □

28.6 Quadratic Suffices

Lemma 28.6.1 (perturbed w)**.** *Let $R \in (0,1)$. Let Assumption 4 in 28.1.1 hold, i.e. for all $i \neq j$, $|x_i^\top x_j| \leq \theta/\sqrt{n}$. If $\widetilde{w}_1, \cdots, \widetilde{w}_m$ are i.i.d. generated $\mathcal{N}(0, I)$. For any set of weight vectors $w_1, \cdots, w_m \in \mathbb{R}^d$ that satisfy for any $r \in [m]$, $\|\widetilde{w}_r - w_r\|_2 \leq R$, then the $H : \mathbb{R}^{m \times d} \to \mathbb{R}^{n \times n}$ defined*

$$H(w)_{i,j} = \frac{1}{m} x_i^\top x_j \sum_{r=1}^m \mathbf{1}_{w_r^\top x_i \geq 0, w_r^\top x_j \geq 0}.$$

Then we have

$$\|H(w) - H(\widetilde{w})\|_F < 2\left(n(1+\theta^2)\right)^{1/2} R,$$

holds with probability at least $1 - n^2 \cdot \exp(-mR/10)$.

Proof. The random variable we care is

$$\begin{aligned}
\sum_{i=1}^n \sum_{j=1}^n |H(\widetilde{w})_{i,j} - H(w)_{i,j}|^2 = & \frac{1}{m^2} \sum_{i=1}^n \sum_{j=1}^n \left| x_i^\top x_j \sum_{r=1}^m (\mathbf{1}_{\widetilde{w}_r^\top x_i \geq 0, \widetilde{w}_r^\top x_j \geq 0} - \mathbf{1}_{w_r^\top x_i \geq 0, w_r^\top x_j \geq 0}) \right|^2 \\
= & B_1 + B_2,
\end{aligned}$$

where B_1, B_2 are defined as

$$\begin{aligned}
B_1 = & \frac{1}{m^2} \sum_{i=1}^n \left| \sum_{r=1}^m (\mathbf{1}_{\widetilde{w}_r^\top x_i \geq 0} - \mathbf{1}_{w_r^\top x_i \geq 0}) \right|^2, \\
B_2 = & \frac{1}{m^2} \sum_{i=1}^n \sum_{j \in [n] \setminus \{i\}} \left| x_i^\top x_j \sum_{r=1}^m (\mathbf{1}_{\widetilde{w}_r^\top x_i \geq 0, \widetilde{w}_r^\top x_j \geq 0} - \mathbf{1}_{w_r^\top x_i \geq 0, w_r^\top x_j \geq 0}) \right|^2.
\end{aligned}$$

We can further bound B_2 as

$$B_2 \le \frac{1}{m^2}\sum_{i=1}^{n}\sum_{j\in[n]\setminus\{i\}}\frac{\theta^2}{n}\left|\sum_{r=1}^{m}(\mathbf{1}_{\widetilde{w}_r^\top x_i\ge 0,\widetilde{w}_r^\top x_j\ge 0}-\mathbf{1}_{w_r^\top x_i\ge 0,w_r^\top x_j\ge 0})\right|^2$$
$$= \frac{\theta^2}{nm^2}\sum_{i=1}^{n}\sum_{j\in[n]\setminus\{i\}}\left|\sum_{r=1}^{m}(\mathbf{1}_{\widetilde{w}_r^\top x_i\ge 0,\widetilde{w}_r^\top x_j\ge 0}-\mathbf{1}_{w_r^\top x_i\ge 0,w_r^\top x_j\ge 0})\right|^2.$$

For each r,i,j, we define

$$s_{r,i,j} := \mathbf{1}_{\widetilde{w}_r^\top x_i\ge 0,\widetilde{w}_r^\top x_j\ge 0}-\mathbf{1}_{w_r^\top x_i\ge 0,w_r^\top x_j\ge 0}.$$

Then we can rewrite B_1 and B_2 as

$$B_1 = \frac{1}{m^2}\sum_{i=1}^{n}\Big(\sum_{r=1}^{m}s_{r,i,i}\Big)^2,$$
$$B_2 = \frac{\theta^2}{nm^2}\sum_{i=1}^{n}\sum_{j\in[n]\setminus\{i\}}\Big(\sum_{r=1}^{m}s_{r,i,j}\Big)^2.$$

Therefore it is sufficient to bound $\sum_{r=1}^{m}s_{r,i,j}$ simutaneously for all pair i,j. Using same technique in the proof of Theorem 28.4.2, we have

$$\Pr\left[\frac{1}{m}\sum_{r=1}^{m}s_{r,i,j}\ge 2R\right]\le\exp(-mR/10).$$

By applying union bound on all i,j pairs, we get with probability at least $1-\exp(-mR/10)$,

$$\|H(w)-H(\widetilde{w})\|_F^2 \le B_1+B_2 \le 4nR^2(1+\theta)^2.$$

which is precisely what we need. □

Table 28.4: Table of Parameters for the $m = \widetilde{\Omega}(n^2)$ result in Section 28.6. **Nt.** stands for notations.

Nt.	**Choice**	**Place**	**Comment**
λ	$:= \lambda_{\min}(H^{\text{cts}})$	Part 1 of Ass. 28.1.1	Data-dependent
α	Absolute	Part 2 of Ass. 28.1.1	Data-dependent
β	Variance	Part 3 of Ass. 28.1.1	Data-dependent
θ	Inner product	Part 4 of Ass. 28.1.1	Data-dependent
R	$\frac{\lambda}{\sqrt{n}} \cdot \min\{\frac{1}{\sqrt{\alpha}}, \frac{1}{\sqrt{1+\theta^2}}\}$	Eq. (28.15)	Maximal allowed movement of weight
D	$\frac{4\sqrt{\alpha}\|y-u(0)\|_2}{\sqrt{m}\lambda}$	Theorem 28.6.4	Actual moving distance, discrete case
η	$\lambda/(\alpha n)$	Eq. (28.11)	Step size of gradient descent
m	$(\lambda^{-2}\beta + \lambda^{-1}\alpha)\log(n/\delta)$	Theorem 28.5.1	Bounding discrete and continuous
m	$\lambda^{-4}\alpha(\alpha+\theta^2)n^2\log^3(n/\delta)$	Lemma 28.4.4 Claim 28.4.7	$D < R$ and $\|y-u(0)\|_2^2 = \widetilde{O}(n)$

Claim 28.6.2. *Assume* $R \leq \frac{\lambda}{64\sqrt{n}} \cdot \frac{1}{\sqrt{1+\theta^2}}$. *Let* $C_1 = -2\eta(y-u(k))^\top H(k)(y-u(k))$. *We have*

$$C_1 \leq -\|y-u(k)\|_2^2 \cdot \eta\lambda$$

Proof. By Lemma 28.6.1 and our choice of $R \leq \frac{\lambda}{64\sqrt{n}} \cdot \frac{1}{\sqrt{1+\theta^2}}$, We have

$$\|H(0) - H(k)\|_F \leq 2\left(n(1+\theta^2)\right)^{1/2} \cdot \frac{\lambda}{64\sqrt{n}} \cdot \frac{1}{\sqrt{1+\theta^2}} \leq \frac{\lambda}{4}.$$

Recall that $\lambda = \lambda_{\min}(H(0))$. Therefore

$$\lambda_{\min}(H(k)) \geq \lambda_{\min}(H(0)) - \|H(0) - H(k)\| \geq \lambda/2.$$

Then we have

$$(y-u(k))^\top H(k)(y-u(k)) \geq \|y-u(k)\|_2^2 \cdot \lambda/2.$$

Thus, we complete the proof. □

Claim 28.6.3. *Let* $C_2 = 2\eta(y - u(k))^\top H(k)^\perp (y - u(k))$. *We have*

$$C_2 \leq \|y - u(k)\|_2^2 \cdot 8\eta R \left(n(1+\theta^2)\right)^{1/2}.$$

holds with probability $1 - n\exp(-mR)$.

Proof. Note that

$$C_2 \leq 2\eta \cdot \|y - u(k)\|_2^2 \cdot \|H(k)^\perp\|.$$

It suffices to upper bound $\|H(k)^\perp\|$. Since $\|\cdot\| \leq \|\cdot\|_F$, then it suffices to upper bound $\|\cdot\|_F$.

For each $i \in [n]$, we define y_i as follows

$$y_i = \sum_{r=1}^{m} \mathbf{1}_{r \in \overline{S}_i}.$$

Then we have

$$\begin{aligned}
\|H(k)^\perp\|_F^2 = & \sum_{i=1}^{n}\sum_{j=1}^{n} (H(k)_{i,j}^\perp)^2 \\
= & \sum_{i=1}^{n}\sum_{j=1}^{n} \Big(\frac{1}{m}\sum_{r\in\overline{S}_i} x_i^\top x_j \mathbf{1}_{w_r(k)^\top x_i \geq 0, w_r(k)^\top x_j \geq 0}\Big)^2 \\
= & \sum_{i=1}^{n}\sum_{j=1}^{n} \Big(\frac{1}{m}\sum_{r=1}^{m} x_i^\top x_j \mathbf{1}_{w_r(k)^\top x_i \geq 0, w_r(k)^\top x_j \geq 0} \cdot \mathbf{1}_{r\in\overline{S}_i}\Big)^2 \\
= & \sum_{i=1}^{n}\sum_{j=1}^{n} \frac{|x_i^\top x_j|^2}{m^2}\Big(\sum_{r=1}^{m} \mathbf{1}_{w_r(k)^\top x_i \geq 0, w_r(k)^\top x_j \geq 0} \cdot \mathbf{1}_{r\in\overline{S}_i}\Big)^2 \\
= & B_1 + B_2,
\end{aligned}$$

where B_1 and B_2 are defined as:

$$B_1 := \sum_{i=1}^{n} \frac{1}{m^2} \Big(\sum_{r=1}^{m} \mathbf{1}_{w_r(k)^\top x_i \geq 0} \cdot \mathbf{1}_{r \in \overline{S}_i} \Big)^2,$$
$$B_2 := \sum_{i=1}^{n} \sum_{j \in [n] \setminus \{i\}} \frac{|x_i^\top x_j|^2}{m^2} \Big(\sum_{r=1}^{m} \mathbf{1}_{w_r(k)^\top x_i \geq 0, w_r(k)^\top x_j \geq 0} \cdot \mathbf{1}_{r \in \overline{S}_i} \Big)^2.$$

We bound B_1 and B_2 separately.

We first bound B_1.

$$\begin{aligned} B_1 &= \sum_{i=1}^{n} \frac{1}{m^2} \Big(\sum_{r=1}^{m} \mathbf{1}_{w_r(k)^\top x_i \geq 0} \cdot \mathbf{1}_{r \in \overline{S}_i} \Big)^2 \\ &\leq \frac{1}{m^2} \sum_{i=1}^{n} \Big(\sum_{r=1}^{m} \mathbf{1}_{r \in \overline{S}_i} \Big)^2 \\ &= \frac{1}{m^2} \sum_{i=1}^{n} y_i^2. \end{aligned}$$

Fix $i \in [n]$. The plan is to use Bernstein inequality to upper bound y_i with high probability.

First by Eq. (28.6) we have

$$\mathbb{E} \left[\mathbf{1}_{r \in \overline{S}_i} \right] \leq R.$$

We also have

$$\begin{aligned} \mathbb{E} \Big[\big(\mathbf{1}_{r \in \overline{S}_i} - \mathbb{E} \left[\mathbf{1}_{r \in \overline{S}_i} \right] \big)^2 \Big] &= \mathbb{E} \Big[\mathbf{1}_{r \in \overline{S}_i}^2 \Big] - \mathbb{E} \left[\mathbf{1}_{r \in \overline{S}_i} \right]^2 \\ &\leq \mathbb{E} \Big[\mathbf{1}_{r \in \overline{S}_i}^2 \Big] \\ &\leq R. \end{aligned}$$

Finally we have $|\mathbf{1}_{r \in \overline{S}_i} - \mathbb{E}[\mathbf{1}_{r \in \overline{S}_i}]| \leq 1$.

Notice that $\{\mathbf{1}_{r\in\overline{S}_i}\}_{r=1}^m$ are mutually independent, since $\mathbf{1}_{r\in\overline{S}_i}$ only depends on $w_r(0)$. Hence from Bernstein inequality (Lemma A.1.2) we have for all $t > 0$,

$$\Pr\left[y_i > m\cdot R + t\right] \le \exp\left(-\frac{t^2/2}{m\cdot R + t/3}\right).$$

By setting $t = 3mR$, we have

$$\Pr\left[y_i > 4mR\right] \le \exp(-mR).$$

Hence by union bound, with probability at least $1 - n\exp(-mR)$, for all $i \in [n]$,

$$y_i \le 4mR.$$

If this happens, we have

$$B_1 \le 16nR^2.$$

Next we bound B_2. We have

$$\begin{aligned}
B_2 &= \sum_{i=1}^n \sum_{j\in[n]\setminus\{i\}} \frac{|x_i^\top x_j|^2}{m^2}\Big(\sum_{r=1}^m \mathbf{1}_{w_r(k)^\top x_i\ge 0, w_r(k)^\top x_j\ge 0}\cdot \mathbf{1}_{r\in\overline{S}_i}\Big)^2 \\
&\le \frac{1}{m^2}\sum_{i=1}^n \left(\sum_{j\in[n]\setminus\{i\}} (x_i^\top x_j)^4\right)^{1/2} \cdot \left(\sum_{j\in[n]\setminus\{i\}} \Big(\sum_{r=1}^m \mathbf{1}_{w_r(k)^\top x_i\ge 0, w_r(k)^\top x_j\ge 0}\cdot \mathbf{1}_{r\in\overline{S}_i}\Big)^4\right)^{1/2} \\
&\le \frac{1}{m^2}\sum_{i=1}^n \left(\sum_{j\in[n]\setminus\{i\}} (x_i^\top x_j)^4\right)^{1/2} \left(\sum_{j\in[n]\setminus\{i\}} y_i^4\right)^{1/2} \\
&= \frac{\sqrt{n-1}}{m^2}\sum_{i=1}^n \left(\sum_{j\in[n]\setminus\{i\}} (x_i^\top x_j)^4\right)^{1/2} y_i^2 \\
&\le 16R^2\sqrt{n}\sum_{i=1}^n \left(\sum_{j\in[n]\setminus\{i\}} (x_i^\top x_j)^4\right)^{1/2},
\end{aligned}$$

where the last step happens when $y_i \leq 4mR$ for all $i \in [n]$.

Now, using the assumption $x_i^\top x_j \leq \frac{\theta}{\sqrt{n}}$ (Part 4 of Assumption 28.1.1), we have

$$B_2 \leq 16nR^2\theta^2.$$

Putting things together, we have with probability at least $1 - n\exp(-mR)$,

$$\begin{aligned} \|H(k)^\perp\|_F^2 &\leq B_1 + B_2 \\ &\leq 16nR^2(1+\theta^2). \end{aligned}$$

This gives us $\|H(k)^\perp\| \leq 4R\left(n(1+\theta^2)\right)^{1/2}$, which is precisely what we need.

□

28.6.1 Main result

Theorem 28.6.4. *Let $\lambda, \alpha, \beta, \theta$ be defined as Assumption 28.1.1. Let*

$$m = \Omega\left(\lambda^{-4} n^2 \alpha \max\{1+\theta^2, \alpha\} \log^3(n/\delta)\right).$$

We i.i.d. initialize $w_r \in \mathcal{N}(0, I)$, a_r sampled from $\{-1, +1\}$ uniformly at random for $r \in [m]$, and we set the step size $\eta = O(\lambda/(\alpha n))$ then with probability at least $1-\delta$ over the random initialization we have for $k = 0, 1, 2, \cdots$

$$\|u(k) - y\|_2^2 \leq (1 - \eta\lambda/2)^k \cdot \|u(0) - y\|_2^2.$$

Proof. **Choice of η and R.** We want to choose η and R such that

$$(1 - \eta\lambda + 8\eta R\left(n(1+\theta^2)\right)^{1/2} + 8\eta(\alpha n)^{1/2} R + \eta^2 \alpha n) \leq (1 - \eta\lambda/2). \tag{28.15}$$

Now, if we set $\eta = \frac{\lambda}{4\alpha n}$ and $R = \frac{\lambda}{64\sqrt{n}} \cdot \min\{\frac{1}{\sqrt{1+\theta^2}}, \frac{1}{\sqrt{\alpha}}\}$, we have

$$8\eta R\left(n(1+\theta^2)\right)^{1/2} + 8\eta(\alpha n)^{1/2} R \leq \frac{1}{4}\eta\lambda,$$

and $\eta^2 n^2 \leq \frac{1}{4}\eta\lambda$. This gives us

$$\|y - u(k+1)\|_2^2 \leq \|y - u(k)\|_2^2 (1 - \eta\lambda/2)$$

with probability at least $1 - 2n\exp(-mR)$.

Over-parameterization size, lower bound on m. By same analysis as in the proof of Theorem 28.5.5, we still have

$$\|w_r(k+1) - w_r(0)\|_2 \leq \frac{4\sqrt{\alpha}\|y - u(0)\|_2}{\sqrt{m}\lambda} := D$$

We require

$$D = \frac{4\sqrt{\alpha}\|y - u(0)\|_2}{\sqrt{m}\lambda} < R = \frac{\lambda}{64\sqrt{n}} \cdot \min\left\{\frac{1}{\sqrt{1+\theta^2}}, \frac{1}{\sqrt{\alpha}}\right\}$$

and

$$2n\exp(-mR) \leq \delta.$$

This implies that

$$\begin{aligned} m &= \Omega(\lambda^{-4} n\alpha \|y - u(0)\|_2^2 \max\{1+\theta^2, \alpha\}) \\ &= \Omega(\lambda^{-4} n^2 \alpha \cdot \max\{1+\theta^2, \alpha\} \cdot \log(m/\delta)\log^2(n/\delta)), \end{aligned}$$

where the last step follows from Claim 28.4.7. □

Chapter 29

Longest Common Subsequence I

Given a pair of n-character strings, the problems of computing their Longest Common Subsequence and Edit Distance have been extensively studied for decades. For exact algorithms, LCS and Edit Distance (with character insertions and deletions) are equivalent; the state of the art running time is (almost) quadratic in n, and this is tight under plausible fine-grained complexity assumptions. But for approximation algorithms the picture is different: there is a long line of works with improved approximation factors for Edit Distance, but for LCS (with binary strings) only a trivial 1/2-approximation was known. In this work we give a reduction from approximate LCS to approximate Edit Distance, yielding the first efficient $(1/2 + \epsilon)$-approximation algorithm for LCS for some constant $\epsilon > 0$.

This part is based on previous following publication

- Aviad Rubinstein, Zhao Song

  Reducing approximate Longest Common Subsequence to approximate Edit Distance.

  Manuscript 2019 [RS19b]

29.1 Introduction

In this work we consider two of the most ubiquitous measures of similarity between a pair of strings: the longest common subsequence (LCS) and the edit distance. The LCS of two n-character strings A and B is simply their longest (not necessarily contiguous) common substring. Edit distance is the minimum number of character insertions, deletions, and substitutions required to transform A to B. In fact, under a slightly more restricted definition that does not allow substitutions[1], the two measures are complements and the problems of computing them exactly are equivalent.

There is a textbook dynamic programming algorithm for computing LCS (or edit distance) than runs in $O(n^2)$ time, and a slightly faster $O(n^2/\log^2(n))$-time algorithm due to Masek and Paterson [MP80]. Finding faster algorithms is a central and long standing open problem both in theory and in practice (e.g. Problem 35 of [Knu72]). Under plausible fine-grained complexity assumptions such as SETH, neither problem can be computed much faster [AWW14, ABW15, BI15, BK15, AHWW16].

For (multiplicative) approximation, the two problems are no longer equivalent. For edit distance, there is a long sequence of approximation algorithms with improving factors [BYJKK04, BES06, AO12, AKO10, BEG+18]; in particular, [CDG+18] gives a constant factor approximation in truly sub-quadratic time. For LCS with alphabet size $|\Sigma|$, in contrast, there is a trivial $1/|\Sigma|$-approximation, and no better algorithms are known (for large alphabet[2] there are some hardness of approximation results [AB17, AR18, CGL+19] and also

[1] Since the definitions are equivalent up to a factor of 2 (each substitution is an insertion and a deletion), this difference is irrelevant as we consider constant factor approximations of edit distance.

[2] One may also consider the case of intermediate alphabet size, e.g. $\Sigma = \{A, C, G, T\}$. It is plausible that

approximation algorithms with non-trivial polynomial factors [HSSS19, RSSS19]).

In this work we focus on binary strings, where the trivial algorithm gives a $(1/|\Sigma| = 1/2)$-approximation. Breaking this $1/2$ barrier is a well-known open problem in this area. Our main result is a fine-grained reduction that implies obtaining a $1/2 + \epsilon$-approximation for binary LCS (for some constant $\epsilon > 0$) is no harder than approximating edit distance to within some constant factor.

Theorem 29.1.1 (Reduction: approximate ED implies approximate LCS)**.**

Suppose that there exists a constant c and an approximate edit distance algorithm that runs in time $T(n)$ and, given two binary strings A, B of length n, returns an estimate $\widetilde{\mathrm{ED}}(A, B) \in [\mathrm{ED}(A, B), c \cdot \mathrm{ED}(A, B) + o(n)]$. Then there exists a fixed constant $\epsilon = \epsilon(c) \in (0, 1/2)$ and a deterministic approximation algorithm for longest common subsequence that runs in deterministic $T(n) + O(n)$ and approximates $\mathrm{LCS}(A, B)$ to within a $(1/2 + \epsilon)$-approximation factor.

Remark 29.1.1. We state the above theorem in terms of estimating the edit distance or length of the LCS. If the edit distance algorithm can efficiently compute the transformation (this assumption is almost wlog by [CGKK18]), then our algorithm can also efficiently compute the common string.

As mentioned above, the recent breakthrough of [CDG+18] gives a constant factor approximation of edit distance in truly-subquadratic ($\widetilde{O}(n^{2-2/7})$) time. By plugging their algorithm into our reduction, we would obtain $(1/2+\epsilon)$-approximation algorithm for binary

our framework can give a $1/|\Sigma| + \epsilon$-approximation in this case as well, but the proof may become more complicated.

LCS with the same running time. By applying our reduction to the even more recent approximation algorithms for edit distance[3] that run in near-linear time [KS19, BR19], we obtain the following stronger corollary:

Corollary 29.1.2 (Approximate LCS)**.** *For every constant $\delta > 0$ there exists a constant $\epsilon > 0$ such that, given two binary strings $A, B \in \{0,1\}^n$, there is an algorithm that runs in $O(n^{1+\delta})$ time and $\mathrm{LCS}(A,B)$ to within a $(1/2+\epsilon)$-factor.*

Technical preview

The crux of our algorithm is analyzing first order statistics (counts of 0s and 1s) of the input strings (A, B) and their substrings. We begin with a few simple observations. Below, we normalize ED and LCS so that they're always between 0 and 1 (as opposed to 0 and n).

- If the strings are balanced, namely have the same number of 0s and 1s, we know that $\mathrm{LCS}(A,B) \in [1/2, 1]$. If the strings are very close, say $\mathrm{LCS}(A,B) \geq (1-\delta)$ for sufficiently small $\delta > 0$, we can use the assumed edit distance algorithm as a black box and find a common substring of length $\geq (1 - O(\delta))$. On the converse if the substring returned by the algorithm is shorter than $(1-O(\delta))$, we know that $\mathrm{LCS}(A,B) < 1-\delta$, and thus returning an all-1 string of length $1/2$ is a $(1/2+2\delta)$-approximation.

- If A is balanced and B has e.g. 10% 0s and 90% 1s, we know that $\mathrm{LCS}(A,B) \in [0.5, 0.6]$, so simply returning the all-1 string of length $1/2$ is a $5/6$-approximation. The same holds for most ways in which one or both strings are unbalanced.

[3]The near-linear time approximation algorithms for edit distance [KS19, BR19] also incur a sublinear additive error term, but that is OK for our reduction.

- However there is one difficult case when the string are *perfectly unbalanced*, e.g. A has 99% 0s and B has 99% 1s. Now the first order statistics over the entire strings only tell us that $\mathrm{LCS}(A, B) \in [0.01, 0.02]$, so the trivial approximation doesn't beat 1/2. On the other hand, the edit distance is at least 0.98, so even a 1.1-approximation algorithm for edit distance wouldn't give us a non-trivial guarantee for this case.

Our main technical contribution is a careful analysis of this last case (and its many sub-cases).

29.2 Preliminaries

For strings $x, y \in \{0,1\}^m$ for $m \leq n$, we use $1(x)$ to denote the number of 1 in x, $0(x)$ to denote the number of 0 in x, and $\mathrm{LCS}(x, y)$ to denote the length of their longest common subsequence. All of these function are normalized w.r.t. the length of the original input to our main algorithm, n; in particular we always have $0(x), 1(x), \mathrm{LCS}(x, y) \in [0, m/n]$.

Fact 29.2.1.

$$\mathrm{LCS}(A, B) \leq \min\{0(A), 0(B)\} + \min\{1(A), 1(B)\}.$$

Parameters $\alpha, \beta, \gamma, \delta$

In the proof we consider the following parameters:

α We define $\alpha := \min\{1(A), 1(B), 0(A), 0(B)\}$. Notice that α may be very small, and even approaching 0 as a function of n. We assume wlog that this minimum is attained by $1(A) = \alpha$.

β The parameter β will represent a robustness parameter for some of our bounds. We take $\beta = \Theta(\alpha)$, but it may be smaller by an arbitrary constant factor.

γ The parameter $\gamma \in (0, 1)$ is a constant that depends on the approximation factor c of the approximation algorithm for edit distance that we assume. We choose β sufficiently small such that $\gamma\alpha \gg \beta$.

δ The parameter δ represents the deviation from "perfectly unbalanced" case (see Lemma 29.3.1). It is an arbitrary small constant. In particular, $\delta\alpha \ll \beta$. It is sufficiently small

that for succinctness of representation we'll simply omit it (as if it were zero) after Lemma 29.3.1.

Subroutines

Our reduction will assume the availability of an algorithm ApproxED which takes as input two strings A, B of length n and outputs $1-\widetilde{\mathrm{ED}}(A,B)$ where $\widetilde{\mathrm{ED}}(A,B) \in [\mathrm{ED}(A,B), c \cdot \mathrm{ED}(A,B) + o(1)]$.

In addition, we also define three trivial algorithms; they all run in time linear in length of input string.

Definition 29.2.1 (Match)**.** Given input string A and B, and a symbol $\sigma \in \Sigma$. The algorithm $\textsc{Match}(A, B, \sigma)$ will output a string C where every character is σ and the length of C is $\min\{\sigma(A), \sigma(B)\}$. This algorithm takes $O(|A| + |B|)$ time.

Definition 29.2.2 (BestMatch)**.** Given input string A and B. The algorithm $\textsc{BestMatch}(A, B)$ will take the longest one of $\textsc{Match}(A, B, 0)$ and $\textsc{Match}(A, B, 1)$. This algorithm also takes $O(|A| + |B|)$ time.

Definition 29.2.3 (Greedy)**.** Given input string A_1, A_2 and B. The algorithm $\textsc{Greedy}(A_1, A_2, B)$ will find the optimal contiguous partition $B = B_1 \cup B_2$ so as to maximize $\textsc{BestMatch}(A_1, B_1) + \textsc{BestMatch}(A_2, B_2)$. This algorithm also takes $O(|A| + |B|)$ time.

Below, we slightly abuse notation and refer to the above algorithms (ApproxED, Match, BestMatch, Greedy) both when we want their output to be the actual common string, and the length. Which output we need will be clear from context.

29.3 Reducing to perfectly unbalanced case

In this section we formalize the intuition from the introduction that BestMatch gives a better-than-1/2-approximation unless $1(A) \approx 0(B)$.

Lemma 29.3.1 (Reduction to perfectly unbalanced case)**.**

If $|1(A) - 0(B)| > \delta \min\{0(A), 0(B), 1(A), 1(B)\}$*, then*

$$\textsc{BestMatch}(A, B) \geq (1/2 + \delta/6)\,\mathrm{LCS}(A, B).$$

Proof. Assume wlog[4] that $1(A) = \min\{0(A), 0(B), 1(A), 1(B)\}$. Then we have,

$$\begin{aligned}
&\textsc{BestMatch}(A, B) \\
&= \textsc{Match}(A, B, 0) \\
&= \min\{0(B), 0(A)\} \\
&= 0(B) && \text{(By assumption } 1(A) \leq 1(B)\text{)} \\
&> (1 + \delta)1(A) && \text{(By premise of lemma)} \\
&= (1 + \delta)\min\{1(A), 1(B)\} \\
&\geq (1 + \delta)\big(\mathrm{LCS}(A, B) - \min\{0(A), 0(B)\}\big) && \text{(Fact 29.2.1)} \\
&= (1 + \delta)\big(\mathrm{LCS}(A, B) - \textsc{BestMatch}(A, B)\big).
\end{aligned}$$

[4]This is wlog since $|1(A) - 0(B)| = |0(A) - 1(B)|$, so the premise of this lemma is symmetric.

Therefore,

$$\begin{aligned}\textsc{BestMatch}(A,B) &\geq \frac{1+\delta}{2+\delta}\,\mathrm{LCS}(A,B)\\ &\geq (1/2+\frac{\delta}{4+2\delta})\,\mathrm{LCS}(A,B)\\ &\geq (1/2+\delta/6)\,\mathrm{LCS}(A,B).\end{aligned}$$

□

We henceforth assume wlog that

$$|0(A)-1(B)| \leq \delta \min\{0(A),0(B),1(A),1(B)\}. \tag{29.1}$$

For ease of presentation, we henceforth omit δ from our calculations, i.e. we'll assume that $\delta = 0$. It will be evident that modifying any of our inequalities by factors in $[\pm\delta\alpha]$ will not affect the proofs.

Eq. (29.1) is very important in our analysis, but it does not rule out the perfectly balanced case, namely $0(A) \approx 0(B) \approx 1(A) \approx 1(B) \approx 1/2$.

Lemma 29.3.2 (Ruling out the perfectly balanced case)**.**

Let $\beta', \gamma > 0$ be sufficiently small constants[5]*. If $0(A) \in [1/2 \pm \beta']$, then*

$$\max\{\textsc{BestMatch}(A,B), \textsc{ApproxED}(A,B)\} \geq (1/2+\gamma)\,\mathrm{LCS}(A,B). \tag{29.2}$$

Proof. By Eq. (29.1), the premise implies that $1(B) \in [1/2 \pm \beta']$, and by symmetry also $1(A), 0(B) \in [1/2 \pm \beta']$. Therefore, $\textsc{BestMatch}(A,B) \geq 1/2 - \beta'$.

[5]In particular, in Eq. (29.3) we require that $\beta' + \gamma < \frac{1}{4c+2}$, where c is the approximation factor of the edit distance algorithm.

Suppose that $\textsc{BestMatch}(A, B)$ isn't big enough to satisfy Eq. (29.2) (otherwise we're done). Then,

$$\mathrm{LCS}(A, B) > 2\Big(\textsc{BestMatch}(A, B) - \gamma\Big) \geq 2\Big(1/2 - (\beta' + \gamma)\Big) = 1 - 2(\beta' + \gamma).$$

Thus also

$$\mathrm{ED}(A, B) = 1 - \mathrm{LCS}(A, B) \leq 2(\beta' + \gamma).$$

Therefore, by its approximation guarantee, we have that

$$\textsc{ApproxED}(A, B) \geq 1 - c \cdot \mathrm{ED}(A, B) - o(1) \geq 1 - 2c(\beta' + \gamma) - o(1) \geq 1/2 + \gamma. \quad (29.3)$$

(The latter inequality follows by choosing β' and γ sufficiently small.) □

Setting $\beta' = 10\beta$, we henceforth assume wlog that

$$0(A), 1(A), 0(B), 1(B) \notin [1/2 \pm 10\beta]. \quad (29.4)$$

29.4 Perfectly unbalanced strings

In this section we build on our assumptions from Eq. (29.1) and Eq. (29.4) from the previous section to complete the proof of our reduction.

Recall that we define $\alpha := 1(A) < 1/2$, and by Eq. (29.1), we also have $0(B) = \alpha$. We partition each string into three contiguous substrings, where the extreme left and right substring are each of length α:

$$A = L_A \cup M_A \cup R_A$$

$$B = L_B \cup M_B \cup R_B$$

$$|L_A| = |R_A| = |L_B| = |R_B| = \alpha$$

$$|M_A| = |M_B| = 1 - 2\alpha.$$

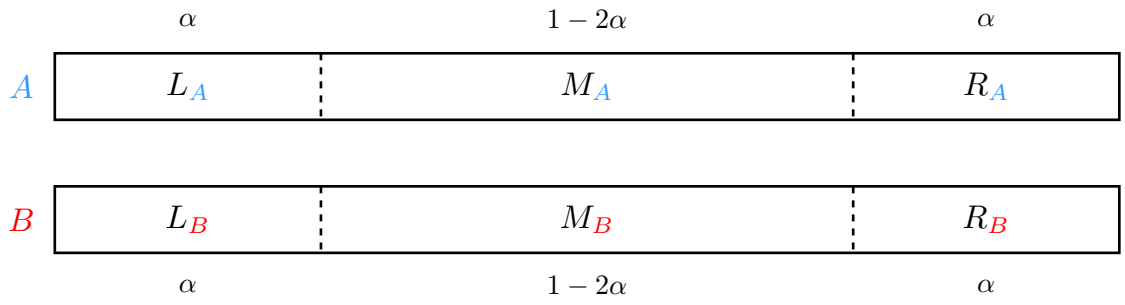


Figure 29.1

We consider six cases for the proportions of 1's and 0's in R_A, L_A, R_B, L_B as in Eq. (29.5). By Table 29.1, we know that those six cases cover all the possibilities.

Our six cases can be summarized in the following equation,

$$\begin{cases} 1(R_B) \leq \alpha/2+2\beta, 0(R_A) \leq \alpha/2+2\beta & \text{Case 1} \\ 1(L_B) \leq \alpha/2+2\beta, 0(L_A) \leq \alpha/2+2\beta & \text{Case 2} \\ 1(R_B) \leq \alpha/2+\beta, 1(L_B) \leq \alpha/2+\beta, 0(L_A) > \alpha/2+2\beta, 0(R_A) > \alpha/2+2\beta & \text{Case 3} \\ 1(R_B) > \alpha/2+2\beta, 1(L_B) > \alpha/2+2\beta, 0(L_A) \leq \alpha/2+\beta, 0(R_A) \leq \alpha/2+\beta & \text{Case 4} \\ 1(R_B) > \alpha/2+\beta, 0(L_A) > \alpha/2+\beta & \text{Case 5} \\ 1(L_B) > \alpha/2+\beta, 0(R_A) > \alpha/2+\beta & \text{Case 6} \end{cases} \tag{29.5}$$

Table 29.1: Fill all the six cases in Eq. (29.5) into the whole space. Note that $1+2+3$ means the combination of 1, 2 and 3 covers it. $5, 6$ means any one of them covers it.

	$0(R_A) \leq \alpha/2+\beta$, $0(L_A) \leq \alpha/2+\beta$	$0(R_A) \leq \alpha/2+\beta$, $0(L_A) > \alpha/2+\beta$	$0(R_A) > \alpha/2+\beta$, $0(L_A) \leq \alpha/2+\beta$	$0(R_A) > \alpha/2+\beta$, $0(L_A) > \alpha/2+\beta$
$1(R_B) \leq \alpha/2+\beta$, $1(L_B) \leq \alpha/2+\beta$	1,2	1	2	1+2+3
$1(R_B) \leq \alpha/2+\beta$, $1(L_B) > \alpha/2+\beta$	1	1	6	6
$1(R_B) > \alpha/2+\beta$, $1(L_B) \leq \alpha/2+\beta$	2	5	2	5
$1(R_B) > \alpha/2+\beta$, $1(L_B) > \alpha/2+\beta$	1+2+4	5	6	5,6

Case 1: $1(R_B) \leq \alpha/2 + 2\beta$**,** $0(R_A) \leq \alpha/2 + 2\beta$

We split this case into three sub-cases, as follows:

$$\begin{cases} 1(R_B) \in [\alpha/2 \pm 4\beta],\ 0(R_A) \in [\alpha/2 \pm 4\beta] & \text{Case 1(a)} \\ 1(R_B) < \alpha/2 - 4\beta,\ 0(R_A) \leq \alpha/2 + 2\beta & \text{Case 1(b)} \\ 1(R_B) \leq \alpha/2 + 2\beta,\ 0(R_A) < \alpha/2 - 4\beta & \text{Case 1(c)} \end{cases} \tag{29.6}$$

Case 1(a): $1(R_B) \in [\alpha/2 \pm 4\beta]$**,** $0(R_A) = [\alpha/2 \pm 4\beta]$

At a high level, we want to split the original problem into two subproblems:

left-middle $(L_A \cup M_A, L_B \cup M_B)$;

right (R_A, R_B).

Running BestMatch on the left-middle subproblem gives a (1/2)-approx; the right subproblem is (approximately) balanced so Lemma 29.3.2 (i.e. taking the better of BestMatch and ApproxED) gives better-than-1/2. The visualization of this case is presented in Figure 29.2.

We first want to upper bound $\text{LCS}(A, B)$ as roughly the sum of LCSs of the two subproblems, but in general this may not be the case. Fix an optimal matching μ corresponding to a longest common substring between A and B. Assume wlog (by symmetry w.r.t. switching A and B) that $\mu(R_A) \subseteq R_B$, i.e. the LCS does not match any R_A characters with characters from $L_B \cup M_B$. μ induces a new partition of B into two[6] contiguous

[6]Note that we do not define a $\widehat{M_B}$.

substrings $\widehat{L_B} \cup \widehat{R_B}$ such that $\mu(R_A) \subseteq \widehat{R_B} \subseteq R_B$. By optimality of μ, we have

$$\mathrm{LCS}(A, B) = \mathrm{LCS}(L_A \cup M_A, \widehat{L_B}) + \mathrm{LCS}(R_A, \widehat{R_B}). \tag{29.7}$$

Applying Fact 29.2.1 to both terms on the RHS, we have

$$\begin{aligned}\mathrm{LCS}(A, B) \leq & \underbrace{\min\{1(L_A \cup M_A), 1(\widehat{L_B})\} + \min\{0(L_A \cup M_A), 0(\widehat{L_B})\}}_{=X} \\ & + \underbrace{\min\{1(R_A), 1(\widehat{R_B})\} + \min\{0(R_A), 0(\widehat{R_B})\}}_{=Y}.\end{aligned} \tag{29.8}$$

We henceforth denote the left and right contributions to the bound on the LCS by X and Y respectively. (So $\mathrm{LCS}(A, B) \leq X + Y$.) We also define:

$$Z := \max\Big\{ \min\{1(L_A \cup M_A), 1(\widehat{L_B})\}, \min\{0(L_A \cup M_A), 0(\widehat{L_B})\}\Big\}.$$

(Observe that $Z \geq X/2$.)

We now prove a lower bound on the LCS that our algorithm can find.

$$\begin{aligned}\textsc{Greedy}(L_A \cup M_A, R_A, B) \geq & \max\Big\{ \min\{1(L_A \cup M_A), 1(\widehat{L_B})\}, \min\{0(L_A \cup M_A), 0(\widehat{L_B})\}\Big\} \\ & + \max\Big\{ \min\{1(R_A), 1(\widehat{R_B})\}, \min\{0(R_A), 0(\widehat{R_B})\}\Big\} \\ \geq & Z + Y/2.\end{aligned} \tag{29.9}$$

We break into sub-cases, depending on the value of Z.

Case 1(a-i): $Z > \alpha/2 + 10\beta$ In this case, observe that

$$\begin{aligned}
X - Z =& \min\{1(L_A \cup M_A), 1(\widehat{L_B}), 0(L_A \cup M_A), 0(\widehat{L_B})\} \\
\leq& 1(L_A \cup M_A) \\
=& \alpha - 1(R_A) && (\alpha = 1(A)) \\
\leq& \alpha/2 + 4\beta && \text{(Case 1(a) assumption)} \\
<& Z - 6\beta && \text{(Case 1(a-i) assumption)}
\end{aligned}$$

Therefore, $Z > X/2 + 3\beta$. Combining with Eq. (29.8) and (29.9), we have that

$$\textsc{Greedy}(L_A \cup M_A, R_A, B) \geq Z + Y/2 > X/2 + Y/2 + 3\beta \geq \mathrm{LCS}(A, B)/2 + 3\beta.$$

Case 1(a-ii): $Z \leq \alpha/2 + 10\beta$ By Eq. (29.7), we have

$$\begin{aligned}
\mathrm{LCS}(A, B) \leq& X + \mathrm{LCS}(R_A, \widehat{R_B}) \\
\leq& 2Z + \mathrm{LCS}(R_A, \widehat{R_B}) && (Z \geq X/2) \\
\leq& \alpha + 20\beta + \mathrm{LCS}(R_A, \widehat{R_B}) && \text{(Case 1(a-ii) assumption)} \\
\leq& \alpha + 20\beta + \mathrm{LCS}(R_A, R_B). && (\widehat{R_B} \subseteq R_B) \qquad (29.10)
\end{aligned}$$

For our purposes, this is effectively as good as bounding $\mathrm{LCS}(A, B)$ by the sum of LCSs of the left-middle and right subproblems.

We run BestMatch on the left-middle subproblem. We have that

$$\begin{aligned}
& \textsc{BestMatch}(L_A \cup M_A, L_B \cup M_B) \\
\geq\ & \min\{0(L_A \cup M_A), 0(L_B \cup M_B)\} \\
\geq\ & \alpha/2 - 4\beta && \text{(Case 1(a) assumption).}
\end{aligned}$$

We run BestMatch and ApproxED on R_A, R_B and take the better of the two outcomes. We apply Lemma 29.3.2 to strings R_A, R_B with $\beta' = 4\beta/\alpha$. (Notice that by Case 1(a) assumption, they are guaranteed to be approximately balanced to within $\pm 4\beta$, or a relative $\pm 4\beta/\alpha$.) We therefore have that

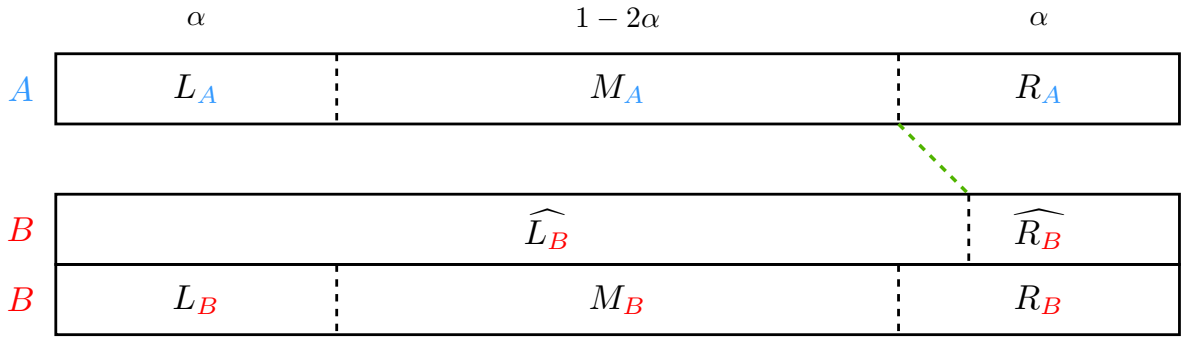
$$\begin{aligned}
&\max\{\textsc{BestMatch}(R_A, R_B), \textsc{ApproxED}(R_A, R_B)\} \\
&\geq (1/2 + \gamma)\,\mathrm{LCS}(R_A, R_B) && \text{(Lemma 29.3.2)} \\
&\geq \mathrm{LCS}(R_A, R_B)/2 + \gamma\alpha/2 - O(\beta\gamma) && (\mathrm{LCS}(R_A, R_B) \geq \alpha/2 - O(\beta)) \\
&\geq \mathrm{LCS}(R_A, R_B)/2 + \gamma\alpha/2 - O(\beta). && (\gamma \leq 1)
\end{aligned}$$

So in total, our algorithm finds a common substring of length at least

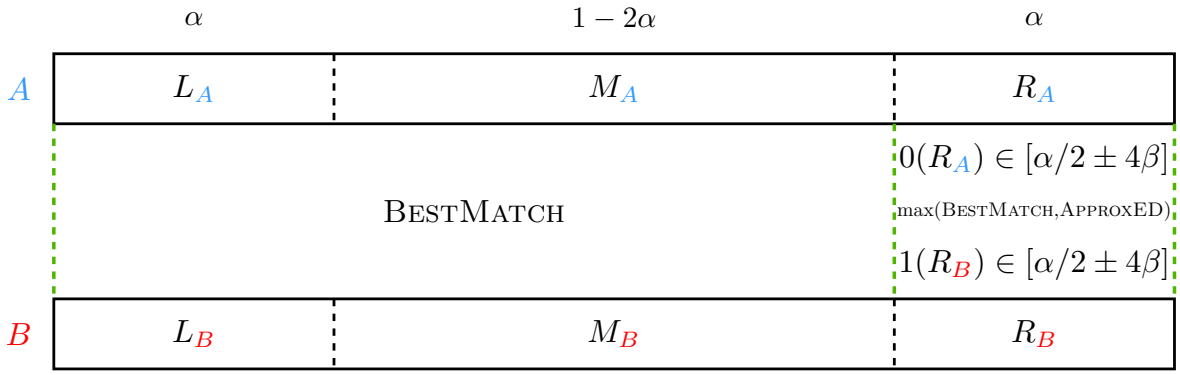
$$\begin{aligned}
&\big(\alpha + \mathrm{LCS}(R_A, R_B)\big)/2 + \gamma\alpha - O(\beta) \\
&\geq \mathrm{LCS}(A, B)/2 + \gamma\alpha/2 - O(\beta) && \text{(Eq. (29.10))} \\
&\geq \mathrm{LCS}(A, B)/2 + \frac{2}{6}\gamma\alpha && (\gamma\alpha \gg \beta) \\
&\geq (1/2 + \gamma/6)\,\mathrm{LCS}(A, B). && (\mathrm{LCS}(A, B) \leq 2\alpha)
\end{aligned}$$

Case 1(b): $1(R_B) < \alpha/2 - 4\beta$**,** $0(R_A) \leq \alpha/2 + 2\beta$

Fix an optimal matching μ. We further split this case into two sub-cases, depending on whether $\mu(R_A) \subseteq R_B$ or $R_B \subseteq \mu(R_A)$. (In Case 1(a) we could assume the former wlog by symmetry. Also notice that in general both may occur simultaneously.)



(a) Partition $(\widehat{L_B}, \widehat{R_B})$ created by GREEDY$(L_A \cup M_A, R_A, B)$



(b) BESTMATCH$(L_A \cup M_A, L_B \cup M_B)$ and max(BESTMATCH(R_A, R_B), APPROXED(R_A, R_B))

Figure 29.2: Visualization of Case 1(a) which is $0(R_A) \in [\alpha/2 \pm 4\beta]$ and $1(R_B) \in [\alpha/2 \pm 4\beta]$. If $0(\widehat{L_B}) > \alpha/2 + 10\beta$, we use GREEDY result. If $0(\widehat{L_B}) \leq \alpha/2 + 10\beta$, we use the result BESTMATCH+ max(BESTMATCH, APPROXED).

If $\mu(R_A) \subseteq R_B$, define the partition $\widehat{L_B}, \widehat{R_B}$ as in Case 1(a). We have

$$\begin{aligned}
\mathrm{LCS}(A, B) &= \mathrm{LCS}(L_A \cup M_A, \widehat{L_B}) + \mathrm{LCS}(R_A, \widehat{R_B}) \\
&\leq \underbrace{1(L_A \cup M_A)}_{\leq \alpha/2+2\beta} + \underbrace{0(\widehat{L_B}) + 0(\widehat{R_B})}_{\leq \alpha} + \underbrace{1(\widehat{R_B})}_{\leq \alpha/2-4\beta} && \text{(Fact 29.2.1)} \\
&\leq 2\alpha - 2\beta. && (29.11)
\end{aligned}$$

Similarly, if $\mu(R_A) \supseteq R_B$, we can define an analogous partition of A into $\widehat{L_A}, \widehat{R_A}$:

$$\begin{aligned}\text{LCS}(A, B) &= \text{LCS}(\widehat{L_A}, L_B \cup M_B) + \text{LCS}(\widehat{R_A}, R_B) \\ &\leq \underbrace{0(L_B \cup M_B)}_{\leq \alpha/2 - 4\beta} + \underbrace{1(\widehat{L_A}) + 1(\widehat{R_A})}_{\leq \alpha} + \underbrace{1(R_B)}_{\leq \alpha/2 - 4\beta} && \text{(Fact 29.2.1)} \\ &\leq 2\alpha - 8\beta. && (29.12)\end{aligned}$$

Either way, we have that $\text{LCS}(A, B) \leq 2\alpha - 2\beta$; therefore $\textsc{Match}(A, B, 0) = \alpha$ guarantees a better-than-1/2-approximation.

Case 1(c): $1(R_B) \leq \alpha/2 + 2\beta$**,** $0(R_A) < \alpha/2 - 4\beta$

Follows analogously to Case 1(b).

Case 2: $1(L_B) \leq \alpha/2 + \beta$**,** $0(L_A) \leq \alpha/2 + \beta$

We reverse the order of string A and B, then the proof is the same as Case 1.

Case 3: $1(R_B) \leq \alpha/2 + \beta$**,** $1(L_B) \leq \alpha/2 + \beta$**,** $0(L_A) > \alpha/2 + 2\beta$ **and** $0(R_A) > \alpha/2 + 2\beta$

We visualize this case in Figure 29.3a.

We show that simple applications of Match to the left, middle, and right substrings can guarantee a common string of at least $\alpha + 2\beta \geq \text{LCS}(A, B)/2 + 2\beta$.

For the middle substrings, observe that $1(M_A) = 1(A) - 1(R_A) - 1(L_A) > 4\beta$. We

can also lower bound $1(M_B)$ by:

$$\begin{aligned} 1(M_B) &= 1(B) - 1(L_B) - 1(R_B) \\ &> \alpha + 20\beta - 1(L_B) - 1(R_B) && \text{(Eq. (29.4))} \\ &\geq 18\beta && \text{(Premise of Case 3).} \end{aligned}$$

Therefore,

$$\textsc{Match}(M_A, M_B, 1) = \min\{1(M_A), 1(M_B)\} \geq 4\beta. \tag{29.13}$$

For the left substrings, observe that $0(L_B) = |L_B| - 1(L_B) > \alpha/2 - \beta$. Therefore,

$$\textsc{Match}(L_A, L_B, 0) = \min\{0(L_A), 0(L_B)\} \geq \alpha/2 - \beta. \tag{29.14}$$

Similarly,

$$\textsc{Match}(R_A, R_B, 0) = \min\{0(R_A), 0(R_B)\} \geq \alpha/2 - \beta. \tag{29.15}$$

Summing up Eq. (29.13),(29.14),(29.15), our algorithm obtains a common string of length at least $\alpha + 2\beta$.

Case 4: $1(R_B) > \alpha/2 + 2\beta$**,** $1(L_B) > \alpha/2 + 2\beta$**,** $0(L_A) \leq \alpha/2 + \beta$ **and** $0(R_A) \leq \alpha/2 + \beta$

We visualize this case in Figure 29.3b.

If we switch A and B, then the proof is the same as Case 3.

Case 5: $1(R_B) > \alpha/2 + \beta$**, and** $0(L_A) > \alpha/2 + \beta$

We visualize this case in Figure 29.3c.

We apply MATCH to two subproblems to obtain a common substring of length greater than $\alpha + 2\beta \geq \text{LCS}(A, B)/2 + 2\beta$.

Observe that $0(L_B \cup M_B) = 0(B) - 0(R_B) > \alpha/2 + \beta$.

$$\text{MATCH}(L_A, L_B \cup M_B, 0) = \min\{0(L_A), 0(L_B \cup M_B)\} > \alpha_/2 + \beta.$$

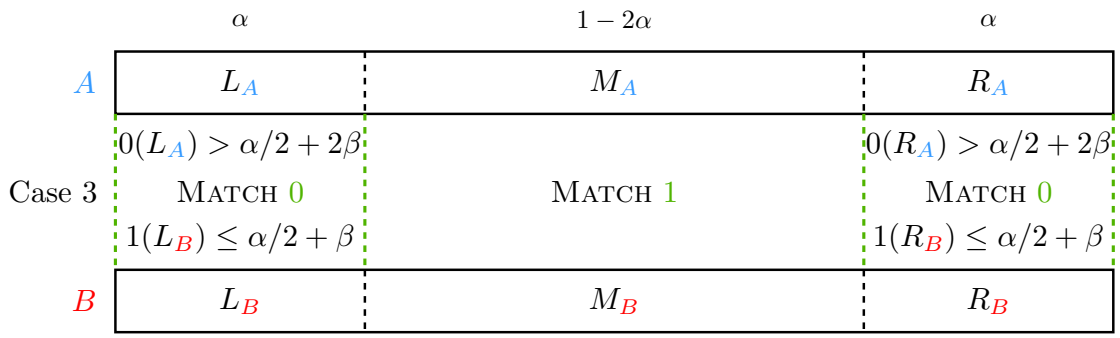
By an analogous argument,

$$\text{MATCH}(M_A \cup R_A, R_B, 1) = \min\{1(M_A \cup R_A), 1(R_B)\} > \alpha/2 + \beta.$$

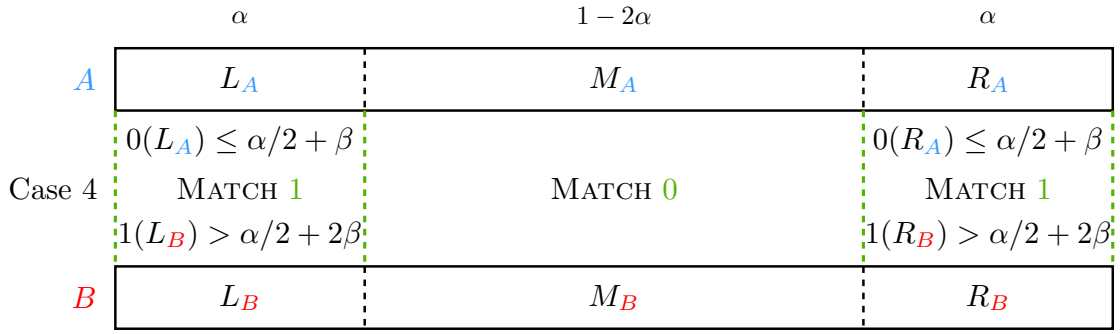
Case 6: $1(L_B) > \alpha/2 + \beta$**, and** $0(R_A) > \alpha/2 + \beta$

We visualize this case in Figure 29.3d.

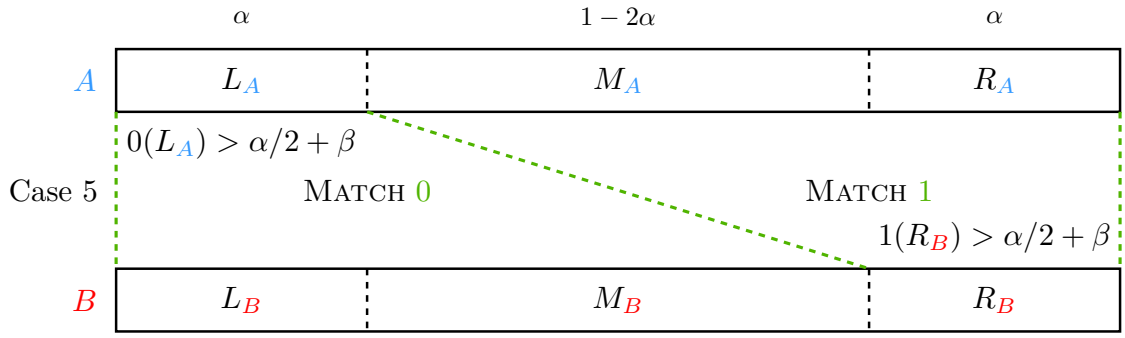
We reverse the oder of string A and B, then the proof is the same as Case 5.



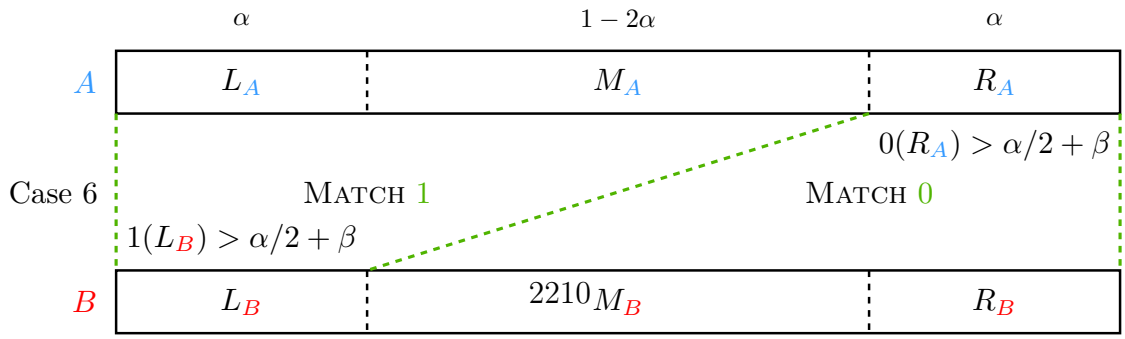
(a)



(b)



(c)



(d)

Figure 29.3: Case 3-6.

Algorithm 29.1 Approximate LCS algorithm

procedure ApproxLCS(A, B, α)
  Split A into three parts, L_A, M_A and R_A such that $|L_A| = |R_A| = \alpha$, similarly for B
  Choose β to be sufficiently small constant
  if $1(R_B) \leq \alpha/2 + 2\beta$ $0(R_A) \leq \alpha/2 + 2\beta$ **then** ▷ Case 1
    if $1(R_B) \in [\alpha/2 \pm 4\beta]$, $0(R_A) \in [\alpha/2 \pm 4\beta]$ **then** ▷ Case 1(a)
      $C, \widehat{L_B}, \widehat{R_B} \leftarrow$ Greedy($L_A \cup M_A, R_A, B$)
      $Z \leftarrow \max\{\min\{1(L_A \cup M_A), 1(\widehat{L_B})\}, \min\{0(L_A \cup M_A), 0(\widehat{L_B})\}\}$
      if $Z \leq \alpha/2 + 10\beta$ **then**
        $C \leftarrow$ BestMatch($L_A \cup M_A, L_B \cup M_B$)
          $+ \max\{$BestMatch(R_A, R_B), ApproxED(R_A, R_B)$\}$
      end if
    else if $1(R_B) < \alpha/2 - 4\beta$, $0(R_A) \leq \alpha/2 + 2\beta$ **then** ▷ Case 1(b)
      $C \leftarrow$ Match($A, B, 0$)
    else if $1(R_B) \leq \alpha/2 + 2\beta$, $0(R_A) < \alpha/2 - 4\beta$ **then** ▷ Case 1(c)
      Similar to Case 1(b)
    end if
  else if $1(L_B) \leq \alpha/2 + \beta$ $0(L_A) \leq \alpha/2 + \beta$ **then** ▷ Case 2
    Similar to Case 1
  else if $1(R_B), 1(L_B) \leq \alpha/2 + \beta$, $0(L_A), 0(R_A) > \alpha/2 + 2\beta$ **then** ▷ Case 3
    $C \leftarrow$ Match($L_A, L_B, 0$) + Match($M_A, M_B, 1$) + Match($R_A, R_B, 0$)
  else if $1(R_B), 1(L_B) > \alpha/2 + 2\beta$, $0(L_A), 0(R_A) \leq \alpha/2 + \beta$ **then** ▷ Case 4
    Similar to Case 3
  else if $1(R_B) > \alpha/2 + \beta$, $0(L_A) > \alpha/2 + \beta$ **then** ▷ Case 5
    $C \leftarrow$ Match($L_A, L_B \cup M_B, 0$) + Match($M_A \cup R_A, R_B, 1$)
  else if $1(L_B) > \alpha/2 + \beta$, $0(R_A) > \alpha/2 + \beta$ **then** ▷ Case 6
    $C \leftarrow$ Match($L_A \cup M_A, L_B, 1$) + Match($R_A, M_B \cup R_B, 0$)
  end if
  return C
end procedure

Chapter 30

Longest Common Subsequence II

Longest common subsequence (LCS) is a classic and central problem in combinatorial optimization. While LCS admits a quadratic time solution, recent evidence suggests that solving the problem may be impossible in truly subquadratic time. A special case of LCS wherein each character appears at most once in every string is equivalent to the longest increasing subsequence problem (LIS) which can be solved in quasilinear time. In this work, we present novel algorithms for approximating LCS in truly subquadratic time and LIS in truly sublinear time. Our approximation factors depend on the ratio of the optimal solution size over the input size. We denote this ratio by λ and obtain the following results for LCS and LIS without any prior knowledge of λ.

- A truly subquadratic time algorithm for LCS with approximation factor $O(\lambda^3)$.
- A truly sublinear time algorithm for LIS with approximation factor $O(\lambda^3)$.

Triangle inequality was recently used by Boroujeni *et al.* [BEG+18] and Chakraborty *et al.* [CDG+18] to present new approximation algorithms for edit distance. Our techniques for LCS extend the notion of triangle inequality to non-metric settings.

This part is based on previous following publication

- Aviad Rubinstein, Saeed Seddighin, Zhao Song, Xiaorui Sun

  Approximation Algorithms for LCS and LIS with Truly Improved Running Times. FOCS 2019 [RSSS19]

30.1 Introduction

In this paper, we consider three of the most classic problems in combinatorial optimization: the longest common subsequence (LCS), the edit distance (ED), and the longest increasing subsequence (LIS). The LCS of two strings A and B is simply their longest (not necessarily contiguous) common subsequence. The edit distance is defined as the minimum number of character deletions, insertions, and substitutions required to transform A into B. For the purpose of our discussion, we consider a more restricted definition of edit distance where substitutions are not allowed[1]. Longest increasing subsequence is equivalent to a special case of LCS where the input strings are permutations. All three problems are very fundamental and have been subject to a plethora of studies in the past few decades and specially in recent years [LMS98, BYJKK04, BES06, AO09, AKO10, SS10, BI15, ABW15, AHWW16, AB17, AR18, CGL+19, BEG+18, CDG+18, HSSS19].

If the strings have length n, both LCS and ED can be solved in quadratic time ($O(n^2)$) with dynamic programming. These running times are slightly improved to $O(n^2/\log^2(n))$ by Masek and Paterson [MP80], however, efforts to improve the running time to $O(n^{2-\Omega(1)})$ for either edit distance or LCS were all futile.

In recent years, our understanding of the source of complexity for these problems tremendously improved thanks to a sequence of fine-grained complexity developments [ABW15, AHWW16]. We now know that assuming the strong exponential time hypothesis (SETH) [ABW15], or even weaker assumptions such as the orthogonal vectors conjecture (OVC) [ABW15] or branching-program-SETH [AHWW16], there are no truly sub-quadratic[2] time algorithms for

[1] Alternatively, the cost of a substitution is doubled as it requires a deletion and an insertion.
[2] By *truly sub-quadratic* we mean $O(n^{2-\epsilon})$, for any constant $\epsilon > 0$

LCS. Similar results also hold for edit distance [BI15].

Indeed, the classic approach to break the quadratic barrier for these problems is approximation algorithms. Note that for (multiplicative) approximations, LCS and edit distance are no longer equivalent (much like we have a 2-approximation algorithm for Vertex Cover, but Independent Set is **NP**-hard to approximate within near-linear factors).

For edit distance, an $\widetilde{O}(n+\Delta^2)$-time algorithm of [LMS98] (where Δ is the true edit distance between the strings) implies a linear-time $\sqrt{n}$-approximation algorithm. The approximation factor has been significantly improved in a series of works to $O(n^{3/7})$ [BYJKK04], to $O(n^{0.34})$ [BES06], to $O(2^{\widetilde{O}(\sqrt{\log n})})$ [AO09][3], and finally to polylogarithmic [AKO10]. A recent work of Boroujeni *et al.* [BEG+18] obtains a constant factor approximation quantum algorithm for edit distance that runs in truly subquadratic time. Finally, the breakthrough of Chakraborty *et al.* [CDG+18] gave a classic (randomized) constant factor approximation for edit distance in truly subquadratic time. A key component in both of the latest constant factor approximation algorithms is the application of triangle inequality (for edit distance between certain substrings of the input). A particular challenge in extending these ideas to LCS is that LCS is not a metric and in particular does not satisfy the triangle inequality.

Our understanding of the complexity of approximate solutions for LCS is embarrassingly limited. For general strings, there are several linear-time $1/\sqrt{n}$-approximation algorithms based on sampling techniques. For alphabet size $|\Sigma|$, there is a trivial $1/|\Sigma|$-approximation algorithm that runs in linear time. Whether or not these approximation factors can be improved by keeping the running time subquadratic is one of the central

[3]We define $\widetilde{O}(f)$ to be $f \cdot \log^{O(1)}(f)$.

problems in fine-grained complexity. Very recently, both the general $1/\sqrt{n}$-approximation factor, and, for binary strings, the 1/2-approximation factor, have been slightly improved ([HSSS19] and [RS19b], respectively). There are a few fine-grained complexity results for approximate LCS, but they only hold against deterministic algorithms, and rely on very strong assumptions [AB17, AR18, CGL+19].

30.1.1 Our Results

For simplicity, we use $\mathsf{lcs}(A, B)$ to denote the size (not the whole sequence) of the longest common subsequence for two strings A and B. Similarly, we use $\mathrm{ED}(A, B)$ to denote the edit distance and $\mathsf{lis}(A)$ for the size of the longest common subsequence. We sometimes normalize the solution by the length of the strings so that the size of the solution remains in the interval $[0, 1]$. We refer to the normalized solutions by $\|\mathsf{lcs}(A, B)\| = \mathsf{lcs}(A, B)/n$ and $\|\mathrm{ED}(A, B)\| = \mathrm{ED}(A, B)/2n$ (here both strings have equal length n), and $\|\mathsf{lis}(A)\| = \mathsf{lis}(A)/n$. In this way, $\|\mathrm{ED}(A, B)\| + \|\mathsf{lcs}(A, B)\| = 1$ (assuming both strings have equal length).

As mentioned earlier, recent developments for edit distance are based on a simple but rather useful observation. Edit distance satisfies triangle inequality, or in other words, given three strings A_1, A_2, A_3 of length n such that $\|\mathrm{ED}(A_1, A_2)\| \leq \delta$ and $\|\mathrm{ED}(A_2, A_3)\| \leq \delta$ hold, we can easily imply that $\|\mathrm{ED}(A_1, A_3)\| \leq 2\delta$. While lcs does not satisfy the triangle inequality in any meaningful way, it does, *on average*, satisfy the following birthday-paradox-like property that we call *birthday triangle inequality*.

Property 30.1.1 (birthday triangle inequality). Consider three equal-length strings A_1, A_2, and A_3 such that $\|\mathsf{lcs}(A_1, A_2)\| \geq \lambda$ and $\|\mathsf{lcs}(A_2, A_3)\| \geq \lambda$. If the common subsequences

correspond to random indices of each string, we expect that $\|\mathsf{lcs}(A_1, A_3)\| \geq \lambda^2$.

Of course, this is not necessarily the case in general. More precisely, it is easy to construct examples[4] in which $\|\mathsf{lcs}(A_1, A_2)\| = 1/2$ and $\|\mathsf{lcs}(A_2, A_3)\| = 1/2$, but $\|\mathsf{lcs}(A_1, A_3)\| = 0$. Our first main result shows that while it only holds on average, we can algorithmically replace the triangle inequality for edit distance with the birthday triangle inequality *on worst case inputs*.

Theorem 30.1.2 (Main Theorem, formally stated as Theorem 30.2.1)**.** *Given strings A, B both of length n such that $\|\operatorname{LCS}(A, B)\| = \lambda$, we can approximate the length of the LCS between the two strings within an $O(\lambda^3)$ factor in subquadratic time. The approximation factor improves to $(1-\epsilon)\lambda^2$ when $1/\lambda$ is constant.*

We remark that our algorithm is actually able to output the whole sequence of the solution, but we only focus on estimating the size of the solution for simplicity. We begin by comparing our main theorem to previous work on edit distance. In this case, $1/\lambda$ is constant w.l.o.g.[5] and therefore the approximation factor of our algorithm is $(1-\epsilon)\lambda^2$. If $\delta = \|\operatorname{ED}(A, B)\|$, then our LCS algorithm outputs a transformation from A to B using at most $2n(1-(1-\epsilon)(1-\delta)^3)$ operations. Observe that when the strings are not overly close and $\delta = \Omega(1)$ by scaling ϵ, we already recover a $(3+\epsilon')$-approximation for edit distance in truly subquadratic time. For mildly far strings, say $\delta = 0.1$, a more careful look at the expansion of $(1-\delta)^3$ reveals that we save an additive $\Theta(\delta^2)$ in the approximation factor. For example, with $\delta = 0.1$ our approximation factor for edit distance is 2.71 instead of 3.

[4] For example, $A_1 = 0^{n/2}0^{n/2}, A_2 = 0^{n/2}1^{n/2}, A_3 = 1^{n/2}1^{n/2}$.

[5] When we use our solution to approximate edit distance, we can safely assume that $\|\mathsf{lcs}(A, B)\| = \Omega(1)$ since otherwise the edit distance of the two strings is very close to $2n$.

An interesting implication of our main result is for LCS over a large alphabet Σ, where the optimum $\|\operatorname{LCS}(A,B)\|$ may be much smaller than 1. This is believed to be the hardest regime for approximation algorithms (and indeed the only one for which we have any conditional hardness of approximation results [AB17, AR18, CGL+19]). Here, we consider instances that satisfy a mild balance assumption: we assume that there is a character that appears with frequency at least $1/|\Sigma|$ in both strings[6]. Then, our main theorem implies an $O(1/|\Sigma|^{3/4})$-approximate solution in truly subquadratic time (the first improvement over the trivial $1/|\Sigma|$ approximation in this regime).

Corollary 30.1.3 (LCS, formally stated as Corollary 30.2.3)**.** *Given a pair of strings (A,B) of length n over alphabet Σ that satisfy the balance condition, we can approximate their LCS within an $O(|\Sigma|^{3/4})$ factor in truly subquadratic time.*

Next, we show that a similar result can be obtained for LIS. Perhaps coincidentally, the approximation factor of our algorithm is also $O(\lambda^3)$ which is same to LCS, but the technique is completely different. Although LIS can be solved exactly in time $O(n \log n)$, there have been several attempts to approximate the size of LIS and related problems in sublinear time [Sch61, Fre75, DGL+99, EKK+00, Fis04, ACCL07, SS10]. The best known solution is due to the work of Saks and Seshadhri [SS10] that obtains a $(1+\epsilon)$-approximate algorithm for LIS in polylogarithmic time, when the solution size is at least a constant fraction of the input size [7]. In other words, if $\|\mathsf{lis}(A)\| = \lambda$ and $1/\lambda$ is constant, their algorithm approximates

[6] Note that in every instance in each string there is a character that appears with frequency at least $1/|\Sigma|$, but in general that may not be the same character.

[7] Their algorithm obtains an additive error of δn in time $2^{\widetilde{O}(1/\delta)}$. When the solution size is bounded by λn, one needs to set $\delta < \lambda$ in order to guarantee a multiplicative factor approximation.

$\mathsf{lis}(A)$ in polylogarithmic time. However, this only works if $1/\lambda$ is constant and even if $1/\lambda$ is logarithmically large, their method fails to run in sublinear time[8]. We complement the work of Saks and Seshadhri [SS10] by presenting a result for LIS similar to our result for LCS. More precisely, we show that when $\|\mathsf{lis}(A)\| = \lambda$, an $O(\lambda^3)$ approximation of LIS can be obtained in truly sublinear time. Although our approximation factor is worse than that of [SS10], our result works for any (not necessarily constant) λ.

Theorem 30.1.4 (LIS, formally stated as Theorem 30.4.8)**.** *Given an array A of n integer numbers such that $\|\mathsf{lis}(A)\| = \lambda$. We can approximate the length of the LIS for A in sublinear time within a factor $O(\lambda^3)$.*

If one favors the running time over the approximation factor, it is possible to improve the exponent of n in the running time down to any constant $\kappa > 0$ at the expense of incurring a larger multiplicative factor to the approximation.

30.1.2 Preliminaries

In LCS or edit distance, we are given two strings A and B as input. We assume for simplicity that the two strings have equal length and refer to that by n. In LCS, the goal is to find the largest subsequence of the characters which is shared between the two strings. In edit distance, the goal is to remove as few characters as possible from the two strings such that the remainders for the two strings are the same. We use $\mathsf{lcs}(A, B)$ and $\mathrm{ED}(A, B)$ to denote the size of the longest common subsequence and the edit distance of two strings A and B.

[8] There is a term $(1/\lambda)^{1/\lambda}$ in the running time.

In LIS, the input contains an array A of n integer numbers and the goal is to find a sequence of elements of A whose values (strictly) increase as their indices increase. For LIS, we denote the solution size for an array A by $\mathsf{lis}(A)$. We also use $\mathsf{lis}^{[\alpha,\beta]}(A)$ to denote the size of the longest increasing subsequence subject to elements whose values lie in range $[\alpha, \beta]$. Longest increasing subsequence is equivalent to LCS when the inputs are two permutations of n distinct characters.

Finally, we define a notation to denote the normalized solution sizes. For LCS, we denote the normalized solution size by $\|\mathsf{lcs}(A,B)\| = \mathsf{lcs}(A,B)/\sqrt{|A||B|}$ for A and B and we use $\|\operatorname{ED}(A,B)\| = \operatorname{ED}(A,B)/(2\sqrt{|A||B|})$ for edit distance. Note that, when the two strings have equal length we have $\|\operatorname{ED}(A,B)\| + \|\mathsf{lcs}(A,B)\| = 1$. Similarly, for longest increasing subsequence, we denote by $\|\mathsf{lis}(A)\| = \mathsf{lis}(A)/|A|$ the normalized solution size. We usually refer to the size of the input array by n.

Throughout this paper, we call an algorithm $f(\lambda)$-approximation for LCS if it is able to distinguish the following two cases: i) $\|\mathsf{lcs}(A,B)\| \geq \lambda$ or ii) $\|\mathsf{lcs}(A,B)\| < \lambda f(\lambda)$. A similar definition carries over to LIS. Once an $f(\lambda)$-approximation algorithm is provided for either LCS or LIS, one can turn it into an algorithm that outputs a solution with size $f(\lambda)(1-\epsilon)\lambda n$ provided that the optimal solution has a size λn. The algorithm is not aware of the value of λ but will start with $\lambda_0 = 1$ and iteratively multiply λ_0 by $1-\epsilon$ until a solution is found.

30.1.3 Techniques Overview

Our algorithm for LCS is closely related to the recent developments for edit distance [BEG+18, CDG+18]. We begin by briefly explaining the previous ideas for approxi-

mating edit distance and then we show how we use these techniques to obtain a solution for LCS. Finally, in Section 30.1.3.2 we outline our algorithm for LIS.

30.1.3.1 Summary of Previous ED Techniques

Indeed, edit distance is hard only if the two strings are far ($\|\operatorname{ED}(A,B)\| = \delta$ and $\delta = n^{-o(1)}$) otherwise the $O(n + (n\delta)^2)$ algorithm of Landau *et al.* [LMS98] computes the solution in truly subquadratic time. The algorithm of Chakraborty *et al.* [CDG+18] for edit distance has three main steps that we briefly discuss in the following.

Step 0 (window-compatible solutions): In the first step, they construct a set of windows W_A for string A and a set of windows W_B for string B. Each window is essentially a substring of the given string. Let k denote the total number of windows of $W_A \cup W_B$. For simplicity, let all the windows have the same size d and $n \simeq O(kd)$[9]. The construction features two key properties: 1) provided that the edit distances of the windows between W_A and W_B are available, one can recover a $1+\epsilon$ approximation of edit distance in time $\widetilde{O}(n+k^2)$ via dynamic programming. 2) $k^2 \times d^2 \simeq O(n^2)$. That is, if we naively compute the edit distance of every pair of windows, the overall running time would still asymptotically be the same as that of the classic algorithm.

In order to obtain a solution for edit distance, it suffices to know the distances between the windows. However, Chakraborty *et al.* [CDG+18] show that knowing the distances between most of the window pairs is enough to obtain an approximately optimal solution for edit distance. More precisely, if the distances are not correctly approximated for at

[9]The equality holds if we assume $\delta = \Omega(1)$.

most $O(k^{2-\Omega(1)})$ pairs, we can still obtain an approximate solution for edit distance. Step 1 provides estimates for the distances of the windows which is approximately correct except for $O(k^{2-\Omega(1)})$ many pairs and Step 2 shows how this can be used to obtain a solution for edit distance. Discretization simplifies the problem substantially. For a fixed $0 \leq \delta \leq 1$, they introduce a graph G_δ where the nodes correspond to the windows and an edge between window $w_i \in W_A$ and window $w_j \in W_B$ means that $\| \operatorname{ED}(w_i, w_j)\| \leq \delta$. If we are able to construct G_δ for logarithmically different choices of δ, we can as well estimate the distances within a $1+\epsilon$ factor for the windows. Therefore the problem boils down to constructing G_δ for a fixed given δ without computing the edit distance between all pairs of windows.

Step 1 (sparsification via triangle inequality): This step is the heart of the algorithm. Suppose we choose a high-degree vertex v from G_δ and discover all its incident edges by computing its edit distance to the rest of the windows. Triangle inequality implies that every pair of windows in $N(v)$ has a distance bounded by 2δ. Therefore by losing a factor 2 in the approximation, one can put all these edges in G_δ and not compute the edit distances explicitly. Although this does save some running time, in order to make sure the running time is truly subquadratic, we need to make a similar argument for paths of length 3 and thereby lose a factor 3 in the approximation. This method sparsifies the graph and what remains is to discover the edges of a sparse graph.

Step 2 (discovering the edges of the sparse graph): Step 1 uses triangle inequality and discovers many edges between the vertices of G_δ. However, it may not discover all the edges completely. When in the remainder graph, the degrees are small (and hence the graph

is sparse) triangle inequality does not offer an improvement and thus a different approach is required. Roughly speaking, Chakraborty *et al.* [CDG+18] subsample the windows of W_A into a smaller set S and discover all pairs of windows $w_i \in S$ and $w_j \in W_B$ such that edge (i, j) is not discovered in Step 1. Next, they compute the edit distance of each pair of windows $(w_i, w_j), w_i \in W_A, w_j \in W_B$ such that there exist two nearby windows (w_a, w_b) satisfying $w_a \in S, w_b \in W_B$ and the edge between w_a and w_b was missed in Step 1. The key observation is that even though this procedure does not discover all the edges, the approximated distances lead to an approximate solution for edit distance.

30.1.3.2 LCS

Our algorithm for LCS mimics the same guideline. In addition to this, Steps 0 and 2 of our algorithm are LCS analogues of the ones used by Chakraborty *et al.* [CDG+18]. The main novelty of our algorithm is Step 1 which is a replacement for triangle inequality. Recall that unlike edit distance, triangle inequality does not hold for LCS.

Challenge 30.1.5. How can we introduce a notion similar to triangle inequality to a non-metric setting such as LCS?

We introduce the notion of birthday triangle inequality to overcome the above difficulty. Given windows w_1, w_2, and w_3 of size d such that $\|\mathsf{lcs}(w_1, w_2)\| \geq \lambda$ and $\|\mathsf{lcs}(w_2, w_3)\| \geq \lambda$ hold, what can we say about the LCS of w_1 and w_3? In general, nothing! $\|\mathsf{lcs}(w_1, w_3)\|$ could be as small as 0. However, let us add some randomness to the setting. Think of the LCS of w_1 and w_2 as a matching from the characters of w_1 to w_2 and similarly the LCS for w_2 and w_3 as another matching between characters of w_2 and w_3. Assume (for the sake of the thought experiment) that the characters of w_2 appear randomly in each matching. Since

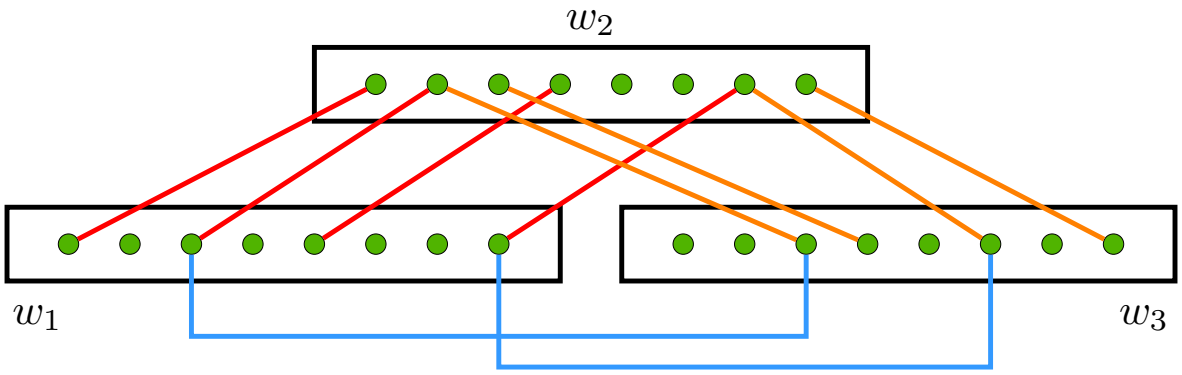


Figure 30.1: Birthday paradox for triangle inequality: let w_1, w_2, w_3 be three windows of length $d = 8$ and assume $\lambda = 1/2$. The LCS between w_1 and w_2 is $\lambda d = 4$ and the LCS between w_2 and w_3 is $\lambda d = 4$. Finally due to birthday paradox, we expect that the LCS between w_1 and w_3 is $\lambda^2 d = 2$.

$\|\mathsf{lcs}(w_1, w_2)\| \geq \lambda$, each character of w_2 appears with probability at least λ in the matching between w_1 and w_2. A similar argument implies that each character of w_2 appears with probability λ in the matching of w_2 and w_3. Thus, (assuming independence), each character of w_2 appears in both matchings with probability λ^2. This means that in expectation, there are $\lambda^2 d$ paths of length 2 between w_1 and w_3 which suggests $\|\mathsf{lcs}(w_1, w_3)\| \geq \lambda^2$ as shown in Figure 30.1. This is basically birthday paradox used for the sake of triangle inequality.

Replacing triangle inequality by birthday triangle inequality is particularly challenging since birthday triangle inequality only holds on average. In contrast, triangle inequality holds for any tuple of arbitrary strings. Most of our technical discussions is dedicated to proving that we can algorithmically use birthday triangle inequality to obtain a solution for the worst-case scenarios. The most inconvenient step of our analysis is to show that our algorithm estimates the LCS of most window pairs in the sparsification phase. While this is straightforward for edit distance, birthday triangle inequality requires a deeper analysis

of the underlying graph. In particular, we need to prove that if the undiscovered edges are too many, then birthday triangle inequality can be applied to certain neighborhoods of the graph.

There are two difficulties that we face here. On one hand, in order to apply birthday triangle inequality to a subgraph, we need to have enough structure for that subgraph to show the implication can be made. On the other hand, our assumptions cannot be too strong, otherwise such neighborhoods may not cover the edges of the graph. Therefore, the first challenge that we need to overcome is characterizing subgraphs in which birthday triangle inequality is guaranteed to be applicable. Our suggestion is the *bi-cliques* structure. Although combinatorial techniques seem unlikely to prove this, we use the Blakley-Roy inequality to show that in a large enough bi-clique, we can use birthday triangle inequality to imply a bound on the LCS of certain pairs. The second challenge is to prove that if the underlying graph is dense enough, the graph contains many bi-cliques that cover almost all the edges that we plan to discover. This is again a challenging graph theoretic problem. We leverage extremal graph theory tools such as Turan's theorem for cliques and bi-cliques to obtain this bound.

Similar to edit distance, we construct a set $W = W_A \cup W_B$ of k windows in Step 0 and aim to sparsify the edges of the lcs-graph in Step 1. Our construction ensures that $kd \simeq \Theta(n)$ and that knowing the LCS of the window pairs suffices to approximate the LCS of the two strings. For a threshold $0 \leq \lambda \leq 1$, define a matrix $\mathsf{O} : [k] \times [k] \to \{0, 1\}$ to be a matrix which identifies whether $\|\mathsf{lcs}(w_i, w_j)\| \geq \lambda$. In other words, $\mathsf{O}[i][j] = 1 \iff \|\mathsf{lcs}(w_i, w_j)\| \geq \lambda$. For an $0 < \alpha \leq 1$, we call a matrix O_α an α approximation of O if it meets the following two conditions:

$$\mathsf{O}_\alpha[i][j] = 0 \Longrightarrow \|\mathsf{lcs}(w_i, w_j)\| < \lambda \qquad \text{and} \qquad \mathsf{O}_\alpha[i][j] = 1 \Longrightarrow \|\mathsf{lcs}(w_i, w_j)\| \geq \alpha \cdot \lambda$$

Notice that O_α gives more flexibility than O for the cases that $\lambda\alpha \leq \mathsf{lcs}(w_i, w_j) < \lambda$. That is, both 0 and 1 are acceptable in these cases. Indeed an α approximation algorithm for the above problem is enough to obtain an α approximation algorithm for LCS. However, this is not necessary as Step 2 allows for incorrect approximation for up to $k^{2-\Omega(1)}$ many window pairs. Therefore, the problem of approximating LCS essentially boils down to approximating O for a given basket of windows $W = W_a \cup W_b$ and a fixed λ by allowing sufficiently small error in the output. A naive solution is to iterate over all pairs w_i and w_j and compute $\mathsf{lcs}(w_i, w_j)$ in time $O(d^2)$ and determine O accordingly. However, this amounts to a total running time of $O(k^2d^2)$ which is quadratic and not desirable. In order to save time, we need to compute the LCS of fewer than k^2 pairs of windows. To make this possible, we allow our algorithm to miss up to $O(k^{2-\Omega(1)})$ edges of the graph. Step 2 ensures that this does not hurt the approximation factor significantly.

We construct a graph from the windows wherein each vertex corresponds to a window and each edge identifies a pair with a large LCS (in terms of λ). Let us call this graph the lcs-graph and denote it by G_λ. The goal is to detect the edges of the graph by allowing false-positive. As we discussed earlier, the hard instances of the problem are the cases where the lcs-graph is dense for which we need a sparsifier. Roughly speaking, in our sparsification technique, our algorithm constructs another graph $\widehat{\mathsf{G}}_\lambda$ such that $\widehat{\mathsf{G}}_\lambda$ is valid in the sense that the edges of $\widehat{\mathsf{G}}_\lambda$ correspond to pairs of windows with large enough LCS. In addition to this, our algorithm guarantees that after the removal of the edges of $\widehat{\mathsf{G}}_\lambda$ from G_λ the remainder is

sparse. In other words, $|E(\mathsf{G}_\lambda) \setminus E(\widehat{\mathsf{G}}_\lambda)| = O(|V(\mathsf{G}_\lambda)|^{2-\Omega(1)})$. Of course, if the overall running time of the sparsification phase is truly subquadratic, the error of undiscovered edges can be addressed by the techniques of [CDG+18] in Step 2.

Below, we bring a formal definition for sparsification.

sparsification
input: Windows $w_1, w_2, \ldots, w_k$, parameters λ, and α.
solution: A matrix $\widehat{\mathsf{O}}_\alpha \in \{0,1\}^{k \times k}$ such that:

- $\widehat{\mathsf{O}}_\alpha[i][j] = 1 \implies \|\mathsf{lcs}(w_i, w_j)\| \geq \alpha \cdot \lambda$
- $\left|\left\{(i,j) \mid \|\mathsf{lcs}(w_i, w_j)\| \geq \lambda \text{ and } \widehat{\mathsf{O}}_\alpha[i][j] = 0\right\}\right| = k^{2-\Omega(1)}$

We present two sparsification techniques for LCS. The first one (Section 30.3.1), has an approximation factor of $(1-\epsilon) \cdot \lambda^2$. In Section 30.3.2 we present another sparsification technique that has a worse approximation factor $O(\lambda^3)$ but leaves fewer edges behind. Although the second sparsification technique has a worse approximation factor, it has the advantage that the number of edges that remain in the sparse graph is truly subquadratic regardless of the value of λ and therefore it extends our solution to the case that $\lambda = o(1)$ (see Section 30.3.2 for a detailed discussion).

Let us note one last algorithmic challenge to keep in mind before we begin to describe our sparsification techniques. For edit distance, if window pairs (w_1, w_2) and (w_2, w_3) are close, we are *guaranteed* that w_1 and w_3 are also close; for longest common subsequence, we will argue that (w_1, w_3) are *likely* to be have a long LCS (for a "random" choice of (w_1, w_3)). Nonetheless, in order to add (w_1, w_3) as an edge to our graph we have to *verify* that their

LCS is indeed long. If we were to verify an edge naively, we would need as much time as computing the LCS between (w_1, w_3) from scratch!

Sparsification 1, $(1-\epsilon)\lambda^2$-approximation: Similar to edit distance, applying birthday triangle inequality to paths of length 2 for LCS does not improve the running time significantly. Therefore, we need to use birthday triangle inequality for paths of length 3. To this end, we define the notion of constructive tuples as follows: a tuple $\langle w_i, w_a, w_b, w_j \rangle$ is an (ϵ, λ)-constructive tuple, if we have $\|\mathsf{lcs}(w_i, w_a)\| \geq \lambda$, $\|\mathsf{lcs}(w_i, w_j)\| \geq \lambda$, $\|\mathsf{lcs}(w_b, w_j)\| \geq \lambda$ and by taking the intersection of the three LCS matchings, we are able to imply $\|\mathsf{lcs}(w_a, w_b)\| \geq (1-\epsilon)\lambda^3$ (see Figure 30.2 for an example). Taking the intersection of the matchings can be done in linear time which is faster than computing the LCS.

Our sparsification technique here is simple but the analysis is very intricate. We subsample a set S of windows and compute the LCS of every window in S and all other windows. We set $|S| = k^\gamma \log k$, where $\gamma \in (0, 1)$. At this point, for some pairs, we already know their LCS. However, if neither w_i nor w_j is in S, we do not know if $\|\mathsf{lcs}(w_i, w_j)\| \geq \lambda$ or not. Therefore, for such pairs, we try to find windows $w_a, w_b \in S$ such that $\langle w_i, w_a, w_b, w_j \rangle$ is constructive. If such a constructive tuple is found for a pair of windows, then we conclude that their normalized LCS is at least $(1-\epsilon)\lambda^3$.

All that remains is to argue that this method discovers almost all the edges of the lcs-graph G_λ and the number of undiscovered edges is $k^{2-\Omega(1)}$. This is the most difficult part of the analysis. We note that proving the existence of **only one** constructive tuple is already non-trivial **even when G_λ is complete**. However, our goal is to show almost all the edges are discovered via constructive tuples when G_λ is dense (and of course not necessarily

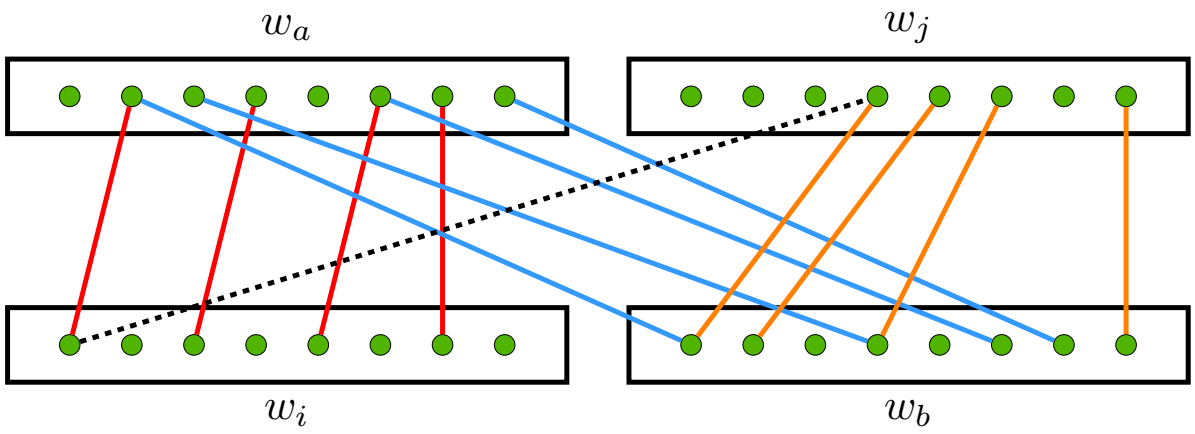


Figure 30.2: Let w_i, w_a, w_b and w_j denote four windows and each of them has length $d = 8$. This figure shows how the intersection of the edges of three windows are taken in order to construct a solution for the LCS of w_i and w_j. If the size of the intersection is large, then such a tuple is called constructive. The solid lines represent LCS between two strings, and the dashed line represents the intersection of the three LCSs.

complete).

Define a pair of windows w_i and w_j to be *well-connected*, if there are at least $k^{2-\gamma}$ different (w_a, w_b) pairs such that $\langle w_i, w_a, w_b, w_j \rangle$ is (ϵ, λ)-constructive. Since each window appears in S with probability $k^{\gamma-1} \log k$, for each well-connected pair we find one constructive tuple via our algorithm with high probablity. Therefore, we need to prove that the total number of pairs (w_i, w_j) such that (w_i, w_j) is not well-connected but $\|\mathsf{lcs}(w_i, w_j)\| \geq \lambda$ is subquadratic. Let us put these edges in a new graph NG_λ whose vertices are all the windows. We first leverage the Blakley-Roy inequality and a double counting technique to prove that if NG_λ has a large complete bipartite subgraph, then there is one constructive tuple which includes only the vertices of this subgraph (Lemma 30.3.2). Next, we apply the Turan's theorem to show that if NG_λ is dense, then it has a lot of large complete bipartite subgraphs. Finally, we use a probabilistic method to conclude that NG_λ cannot be too dense otherwise

there are a lot of constructive tuples in the graph which implies that at least one edge (w_i, w_j) in NG_λ is well-connected. This is not possible since all the well-connected pairs are detected in our sparsification algorithm with high probability.

The above argument proves that if we sparsify our graph using our sparsification algorithm, the remainder graph would have a subquadratic number of edges. Therefore after plugging Step 2 into the algorithm, the running time remains subquadratic. However, since Turan theorem gives us a weak bound, the running time of the algorithm using this sparsification is $O(n^{2-\Omega(\lambda)})$ and is only truly subquadratic when $1/\lambda$ is constant.

The above method also gives a nice insight into computing edit distance. As we show in Section 30.3, if our lcs-graph or similarly ED-graph is very dense, it contains bipartite complete subgraphs that lead to strong implications. For LCS, we obtain a bound of $(1-\epsilon)\lambda^2$ for the approximation factor of the sparsifier. For edit distance, this may lead to an approximation factor better than 3. In particular, if one gives a positive answer to Question 30.1.6 for some α, this technique improves the approximation factor of estimating edit distance in subquadratic time to $2+\alpha+\epsilon$. The authors are not aware of any counter-example for the question even when $\alpha = 0$ and it is quite possible that this technique improves the approximation factor for edit distance down to $2+\epsilon$.

Question 30.1.6. *Let X and Y be two sets of strings of length n such that for each string $A \in X$ and $B \in Y$ we have $\|\operatorname{ED}(A,B)\| \leq \delta$. If $1/\delta$ is constant and the sizes of X and Y are large enough constants (much larger than $1/\delta$) can we imply that there exist two strings in X or two strings in Y whose normalized distance is bounded by $(1+\alpha)\delta$?*

We remark that birthday triangle inequality proves Question 30.1.6 for $\alpha = 1 - O(\delta)$

even when $|X| = 1$. While we do not provide a formal proof, we believe that this already gives a $3 - o(1)$ approximation algorithm for edit distance in truly subquadratic time which slightly breaks the $3 + \epsilon$ barrier suggested in previous work [BEG+18, CDG+18].

Sparsification 2, $O(\lambda^3)$-approximation: In Section 30.3.1, we present another sparsification method that although gives us a slightly worse approximation factor $O(\lambda^3)$ it always leaves a truly subquadratic number of edges behind and therefore the running time of the algorithm would be truly subquadratic regardless of the parameter λ. This sparsification is based on a novel data structure.

Let $\mathsf{opt}_{i,a}$ denote the longest common subsequence of w_i and w_a (with some fixed tie-breaking rule, e.g. lexicographically first). Define $\mathsf{lcs}_{w_a}(w_i, w_j)$ to be the size of the longest common subsequence between $\mathsf{opt}_{i,a}$ and w_j. Notice that this definition is no longer symmetric. Let $\|\mathsf{lcs}_{w_a}(w_i, w_j)\|$ denote the relative value, i.e., $\|\mathsf{lcs}_{w_a}(w_i, w_j)\| = \mathsf{lcs}_{w_a}(w_i, w_j)/\sqrt{|w_i| \cdot |w_j|}$. The first ingredient of the algorithm is a data-structure, namely $\mathsf{lcs\text{-}cmp}$. After a preprocess of time $O(|w_a| \sum_{i \in S} |w_i|)$, $\mathsf{lcs\text{-}cmp}$ is able to answer queries of the following type in time $O(|w_i| + |w_j|)$:

- "for a $0 \leq \widetilde{\lambda} \leq 1$ either certify that $\|\mathsf{lcs}_{w_a}(w_i, w_j)\| \geq \Omega(\widetilde{\lambda}^2)$ or report that $\|\mathsf{lcs}_{w_a}(w_i, w_j)\| < O(\widetilde{\lambda})$".

In our sparsification, we repeat the following procedure k^γ times, where $\gamma \in (0, 1)$. We sample a window w_a uniformly at random and construct $\mathsf{lcs\text{-}cmp}(w_a, S)$ for $S = \{w_i | i \neq a \text{ and } |w_i| \geq |w_a|\}$. After the preprocessing step, we make a query for every pair of windows (w_i, w_j) such that $w_i, w_j \in S$ and determine if $\mathsf{lcs}_{w_a}(w_i, w_j)$ is at least $\Omega(\lambda^4)$ or upper bounded

by $O(\lambda^2)$ (here $\widetilde{\lambda} = \lambda^2$). If their LCS is at least $\Omega(\lambda^4)$ we report this pair as an edge in our lcs-graph. Finally, we use the Turan theorem to prove that the number of remaining edges in our graph is small.

To be more precise, we first construct a graph NG_λ that reflects the edges that are not detected via our sparsification. We give directions to the edges based on the length of the windows. If NG_λ is dense enough, then there is one vertex v in NG_λ with a large enough outgoing degree. We use the neighbors of v to construct another graph NF_λ with vertex set $N(v)$. An edge exists in NF_λ if $\max\{\|\mathsf{lcs}_{w_v(w_i,w_j)}\|, \|\mathsf{lcs}_{w_v(w_j,w_i)}\|\} \geq \Omega(\lambda^2)$. Edges of NF_λ are directed the same way as NG_λ. We prove that NF_λ has no large independent set. In other words, if we select a large enough set of vertices in NF_λ, then there is at least one edges between them. Next, we apply the Turan theorem to prove that NF_λ is dense. Finally, we imply that since NF_λ is dense, there is one vertex u in the neighbors of v such that there are a lot of 2-paths between v and u. This implies that the edge (u, v) should have been detected in our sparsification and therefore must not exist in NG_λ. This contradiction implies that NG_λ is sparse in the first place.

30.1.3.3 LIS

In this section, we present our result for longest increasing subsequence. More precisely, we show that when the solution size is lower bounded by $n\lambda$ ($\lambda \in [0, 1]$), one can approximate the solution within a factor $O(\lambda^3)$ in time $\widetilde{O}(\sqrt{n}/\lambda^7)$. This married with a simple sampling algorithm for the cases that $\lambda < n^{-\Omega(1)}$, provides an $O(\lambda^3)$-approximate algorithm with running time of $\widetilde{O}(n^{0.85})$ (without further dependence on λ). We further extend this result to reduce the running time to $\widetilde{O}(n^\kappa \operatorname{poly}(1/\lambda))$ for any $\kappa > 0$ by imposing

a multiplicative factor of poly$(1/\lambda)$ to the approximation.

Our algorithm heavily relies on sampling random elements of the array for which longest increasing subsequence is desired. Denote the input sequence by $A = \langle a_1, a_2, \ldots, a_n \rangle$. A naive approach to approximate the solution is to randomly subsample the elements of A to obtain a smaller array B and then compute the longest increasing subsequence of B to estimate the solution size for A. Let us first show why this approach alone fails to provide a decent approximation factor. First, consider an array $A = \langle 1, 2, \ldots, n \rangle$ which is strictly increasing. Based on A, we construct two inputs A' and A'' in the following way:

- A' is exactly equal to A except that a p fraction of the elements in A' are replaced by 0.
- A'' is exactly equal to A except that every block of length $\sqrt{n}$ is reversed in A''. In other words, $A'' = \langle \sqrt{n}, \sqrt{n}-1, \sqrt{n}-2, \ldots, 1, 2\sqrt{n}, 2\sqrt{n}-1, \ldots, \sqrt{n}+1, \ldots, n, n-1, n-2, \ldots, n-\sqrt{n}+1 \rangle$.

We subsample the two arrays A' and A'' with a rate of $1/\sqrt{n}$ to obtain two smaller arrays B' and B'' of size roughly $O(\sqrt{n})$. It is easy to prove that $\mathsf{lis}(B') = \Omega(\sqrt{n})$ and $\mathsf{lis}(B'') = \Omega(\sqrt{n})$ both hold even though $\mathsf{lis}(A') = \Omega(n)$ but $\mathsf{lis}(A'') = O(\sqrt{n})$. By setting $p = 1/e$ we can also make sure that $\mathsf{lis}(B')$ and $\mathsf{lis}(B'')$ are within a small multiplicative range even though the gap between $\mathsf{lis}(A')$ and $\mathsf{lis}(A'')$ is substantial.

The above observation shows that the problem is very elusive when random sampling is involved. We bring a remedy to this issue in the following. Divide the input array into $\sqrt{n}$ subarrays of size $\sqrt{n}$. We denote the subarrays by $\mathsf{sa}_1, \mathsf{sa}_2, \ldots, \mathsf{sa}_{\sqrt{n}}$ and fix an

optimal solution opt for the longest increasing subsequence of A. Define $\mathsf{sm}(\mathsf{sa}_i)$ to be the smallest number in sa_i that contributes to opt and $\mathsf{lg}(\mathsf{sa}_i)$ to be the largest number in sa_i that contributes to opt. Moreover, define $\mathsf{lis}^{[\ell,r]}$ to be the longest increasing subsequence of an array subject to the elements whose values lie within the interval $[\ell, r]$. This immediately implies

$$\mathsf{lis}(A) = \sum_{i=1}^{\sqrt{n}} \mathsf{lis}^{[\mathsf{sm}(\mathsf{sa}_i),\mathsf{lg}(\mathsf{sa}_i)]}(\mathsf{sa}_i).$$

Another observation that we make here is that since we assume $\|\mathsf{lis}(A)\| \geq \lambda$ and the size of each subarray is bounded by $\sqrt{n}$, then we have

$$\frac{\mathsf{lis}(A)}{\max_i \mathsf{lis}^{[\mathsf{sm}(\mathsf{sa}_i),\mathsf{lg}(\mathsf{sa}_i)]}(\mathsf{sa}_i)} \geq \sqrt{n}\lambda$$

which essentially means that in order to approximate $\mathsf{lis}(A)$ it suffices to compute $\mathsf{lis}^{[\mathsf{sm}(\mathsf{sa}_i),\mathsf{lg}(\mathsf{sa}_i)]}(\mathsf{sa}_i)$ for $\widetilde{O}(1/\lambda)$ many randomly sampled subarrays. This is quite helpful since this shows that we only need to sample $\widetilde{O}(1/\lambda)$ many subarrays and solve the problem for them. However, we do not know the values of $\mathsf{sm}(\mathsf{sa}_i)$ and $\mathsf{lg}(\mathsf{sa}_i)$ in advance. Therefore, the main challenge is to predict the values of $\mathsf{sm}(\mathsf{sa}_i)$ and $\mathsf{lg}(\mathsf{sa}_i)$ before we sample the subarrays.

Indeed, one needs to read the entire array to correctly compute $\mathsf{sm}(\mathsf{sa}_i)$ and $\mathsf{lg}(\mathsf{sa}_i)$ for each of the subarrays. However, we devise a method to approximately guess these values without losing too much in the size of the solution. Roughly speaking, we show that if we sample $k = O(1/(\lambda\epsilon))$ different elements from a subarray sa_i for some constant ϵ and denote

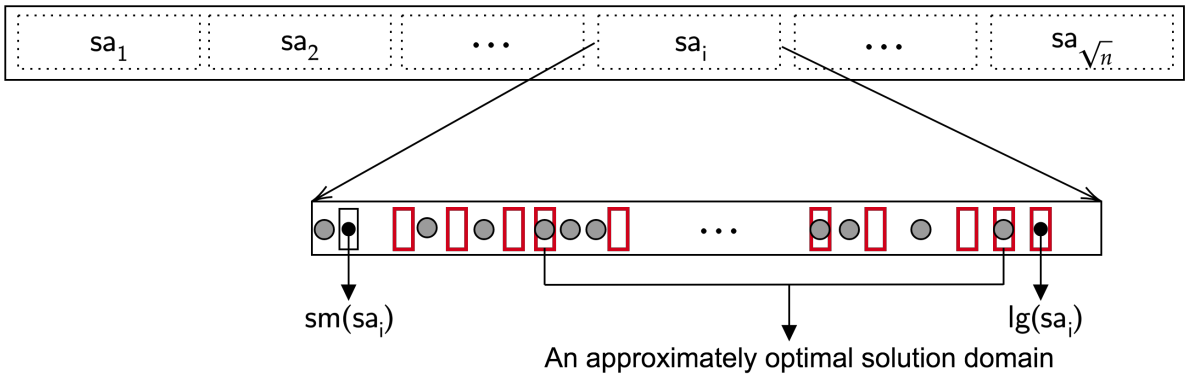


Figure 30.3: Red rectangles show the elements of sa_i that contribute to $\mathsf{lis}(A)$ and gray circles show the elements of sa that are sampled via our algorithm.

them by $a_{j_1}, a_{j_2}, \ldots, a_{j_k}$, then for at least one pair (α, β), $[a_{j_\alpha}, a_{j_\beta}]$ is approximately close to $[\mathsf{sm}(\mathsf{sa}_i), \mathsf{lg}(\mathsf{sa}_i)]$ up to a $(1-\epsilon)$ factor.

The above argument provides $O((1/(\lambda\epsilon))^2)$ candidate domain intervals for each sa_i. However, this does not provide a solution since we do not know which candidate domain interval approximates $[\mathsf{sm}(\mathsf{sa}_i), \mathsf{lg}(\mathsf{sa}_i)]$ for each sa_i. Of course, if we were to randomly choose one candidate interval for every subarray, we would make a correct guess for at least $O(\sqrt{n}(\lambda\epsilon)^2)$ subarrays which provides an approximation guarantee of $O(\lambda^2)$ for our algorithm. However, our assignments have to be monotone too. More precisely, let $[\widetilde{\mathsf{sm}}(\mathsf{sa}_i), \widetilde{\mathsf{lg}}(\mathsf{sa}_i)]$ be the guesses that our algorithm makes, then we should have

$$\widetilde{\mathsf{sm}}(\mathsf{sa}_1) \leq \widetilde{\mathsf{lg}}(\mathsf{sa}_1) \leq \widetilde{\mathsf{sm}}(\mathsf{sa}_2) \leq \widetilde{\mathsf{lg}}(\mathsf{sa}_2) \leq \ldots \leq \widetilde{\mathsf{sm}}(\mathsf{sa}_{\sqrt{n}}) \leq \widetilde{\mathsf{lg}}(\mathsf{sa}_{\sqrt{n}}).$$

Random sampling does not guarantee that the sampled intervals are monotone. To address this issue, we introduce the notion of *pseudo-solutions*. A pseudo-solution is an as-

signment of monotone intervals to subarrays in order to approximate $\mathsf{sm}(\mathsf{sa}_i)$ and $\mathsf{lg}(\mathsf{sa}_i)$. The *quality* of a pseudo solution with intervals $[\ell_1, r_1], [\ell_2, r_2], \ldots, [\ell_{\sqrt{n}}, r_{\sqrt{n}}]$ is equal to $\sum_i \mathsf{lis}^{[\ell_i, r_i]}(\mathsf{sa}_i)$. For a fixed pseudo-solution, this can be easily approximated via random sampling. Thus, our goal is to construct a pseudo-solution whose quality is at least an $O(\lambda^3)$ approximation of the size of the optimal solution. To this end, we present a greedy method in Section 30.4.2 to construct the desired pseudo-solution.

Finally, in Section 30.4.4, we show how the above ideas can be generalized to improve the running time down to $\widetilde{O}(n^{\kappa}\,\mathrm{poly}(1/\lambda))$ for any arbitrarily small $\kappa > 0$ by imposing a factor $\mathrm{poly}(1/\lambda)$ to the approximation guarantee.

30.2 Organization of the Paper

Our algorithm for LCS is explained in Section 30.6 (Step 0), Section 30.3 (Step 1), and Section 30.7 (Step 2). Since Steps 0 and 2 follow from previous work, we only bring Step 1 in the main body of the paper and defer the rest to the appendices.

In both our results for LCS and LIS, we assume that the goal is to find approximate solutions, provided that the solution size is at least $\lambda_0 n$. After the algorithms terminate, if the output is smaller than what we expect, we realize that the solution is smaller than $\lambda_0 n$. Therefore, we begin by setting $\lambda_0 = 1$ and iteratively multiply λ_0 by a $1 - \epsilon$ factor until we obtain a solution. This only adds a multiplicative factor of $\log 1/\lambda$ to the running time and a multiplicative factor of $1 - \epsilon$ to the approximation. Since we present two different sparsification techniques, we obtain two theorems: one is Theorem 30.2.1 and the other is Theorem 30.2.2.

Theorem 30.2.1. *Given strings A, B of length $|A| = |B| = n$ with $\|\operatorname{LCS}(A, B)\| = \lambda$, we can approximate the length of the LCS between the two strings within a factor $O(\lambda^3)$ in time $\widetilde{O}(n^{39/20})$.*

Proof. If $\lambda \leq n^{-20}$ we run the classic $O(n^2\lambda)$ time algorithm and get an exact solution in the desired time. Otherwise, we begin by setting $\lambda_0 = 1$ and iteratively multiply λ_0 by a factor $1-\epsilon$ until a solution is found with size at least $\Omega(\lambda^3)n$. This adds an overhead of $O(\log 1/\lambda)$ to the running time. By Choosing $d = \sqrt{n}$ we can bound the total running time of Steps 0, 1, and 2 by $\widetilde{O}(n^{13/10}\lambda^{-13}) \leq \widetilde{O}(n^{39/20})$. □

Theorem 30.2.2. *Given strings A, B of length $|A| = |B| = n$ with $\|\operatorname{LCS}(A, B)\| = \lambda$, we can approximate the length of the LCS between the two strings within a factor $(1 - \epsilon)\lambda^2$ in*

$n^{2-\Omega(\epsilon\lambda)}$ *time for any* $\epsilon > 0$.

Proof. The proof is identical to Theorem 30.2.1, we omit the details here. □

As an immediate corollary of Theorem 30.2.1, we present an algorithm that beats the $1/|\Sigma|$ approximation factor in truly subquadratic time, when the strings are balanced.

Corollary 30.2.3. *Given a pair of strings* (A, B) *of length* n *over alphabet* Σ *that satisfy the balance condition, we can approximate their* LCS *within an* $O(|\Sigma|^{3/4})$ *factor in time* $O(n^{39/20})$.

Proof. Since A and B are balanced, there is a character $\sigma \in \Sigma$ that appears at least $n/|\Sigma|$ times in both strings. Indeed, finding a solution of size $n/|\Sigma|$ by restricting our attention to only character σ can be done in time $O(n)$. If $\mathsf{lcs}(A, B) \leq n/|\Sigma|^{1/4}$ this already gives us an $O(|\Sigma|^{3/4})$ approximate solution. Otherwise, $\|\mathsf{lcs}(A, B)\| > 1/|\Sigma|^{1/4}$ and the approximation factor of our $O(\lambda^3)$-approximation algorithm would be bounded by $O(|\Sigma|^{3/4})$. □

Finally, we bring our results for LIS in Section 30.4. We show that

Theorem 30.2.4. *Given a length-*n *sequence* A *with* $\mathsf{lis}(A) = n\lambda$. *We can approximate the length of the* LIS *within a factor of* $O(\lambda^3)$ *in time* $\widetilde{O}(n^{17/20})$.

Proof. If $\lambda < n^{-1/20}$ we sample the array with a rate of $n^{-3/20}$ and compute the LIS for the sampled array. The running time of the algorithm is $\widetilde{O}(n^{17/20})$. The approximation factor is $O(n^{-3/20}) \geq O(\lambda^3)$. Otherwise, by Theorem 30.4.8, we estimate the size of LIS up to an $O(\lambda^3)$ approximation factor in time $\widetilde{O}(\lambda^{-7}\sqrt{n}) \leq \widetilde{O}(n^{17/20})$. □

30.3 LCS Step 1: Sparsification via Birthday Triangle Inequality

Recall that we are given two sets of windows W_A and W_B for the strings and our goal is to approximate the LCS of every pair of windows between W_A and W_B. For simplicity, we put all the windows in the same basket $W = W_1 \cup W_2$ and denote the windows by $w_1, w_2, \ldots, w_k$ where k is the total number of windows. Since the windows have different lengths, we define $\mathsf{w}_{\mathsf{max}} = \max_{i\in[k]} |w_i|$ to be the maximum length of the windows. Similarly, we also define $\mathsf{w}_{\mathsf{min}} = \min_{i\in[k]} |w_i|$ to be the minimum length of the windows. Let $\mathsf{w}_{\mathsf{gap}} = \mathsf{w}_{\mathsf{max}}/\mathsf{w}_{\mathsf{min}}$. Let $\mathsf{w}_{\mathsf{layers}}$ denote the number of different window sizes. Notations $\mathsf{w}_{\mathsf{gap}}$ and $\mathsf{w}_{\mathsf{layers}}$ will be used in the later analysis.

In order to approximate the LCS's we fix a $\lambda \in \{\epsilon\lambda_0, (1+\epsilon)\epsilon\lambda_0, (1+\epsilon)^2\epsilon\lambda_0, \ldots, 1\}$ and sparsify graph G_λ. In Section 30.3.1, we present a sparsification algorithm (Algorithm 30.1) which provides λ^2-approximation when λ is constant. The formal guarantee of the algorithm is provided in Theorem 30.3.6. In Section 30.3.2, we present a sparsification which provides $O(\lambda^3)$-approximation for any (potentially super-constant) λ.

30.3.1 Sparsification for O_{λ^2}

In our solution, we fix an arbitrary LCS for every pair of windows and refer to that as $\mathsf{opt}_{i,j}$ for two windows w_i and w_j. Note that we do not explicitly compute $\mathsf{opt}_{i,j}$ in our algorithm. Let us for simplicity, think of each $\mathsf{opt}_{i,j}$ as a matching between the characters of the two windows. Also, denote by $\left(\mathsf{opt}_{i,a} \cap \mathsf{opt}_{a,b} \cap \mathsf{opt}_{b,j}\right)$ a solution which is constructed for windows w_i and w_j by taking the intersection of $\left(\mathsf{opt}_{i,a} \cap \mathsf{opt}_{a,b} \cap \mathsf{opt}_{b,j}\right)$. More precisely,

we have

$$(x, y) \in \left(\mathsf{opt}_{i,a} \cap \mathsf{opt}_{a,b} \cap \mathsf{opt}_{b,j}\right) \Longleftrightarrow \exists \alpha, \beta \text{ such that}$$
$$(x, \alpha) \in \mathsf{opt}_{i,a} \text{ and}$$
$$(\alpha, \beta) \in \mathsf{opt}_{a,b} \text{ and}$$
$$(\beta, y) \in \mathsf{opt}_{b,j}.$$

Let $\left\|\left(\mathsf{opt}_{i,a} \cap \mathsf{opt}_{a,b} \cap \mathsf{opt}_{b,j}\right)\right\| = \frac{\left|\left(\mathsf{opt}_{i,a} \cap \mathsf{opt}_{a,b} \cap \mathsf{opt}_{b,j}\right)\right|}{\sqrt{|w_i||w_j|}}$.

Definition 30.3.1 ((ϵ, λ)-constructive)**.** We call a tuple $\langle w_i, w_a, w_b, w_j \rangle$ $(w_i \neq w_a \neq w_b \neq w_j)$ a *constructive* tuple, if $\left\|\left(\mathsf{opt}_{i,a} \cap \mathsf{opt}_{a,b} \cap \mathsf{opt}_{b,j}\right)\right\| \geq (1 - \epsilon)\lambda^3$.

The advantage of a constructive tuple is that if $\mathsf{opt}_{i,a}, \mathsf{opt}_{a,b}$, and $\mathsf{opt}_{b,j}$ are provided, one can construct a desirable solution for $\mathsf{opt}_{i,j}$ in linear time by taking the intersection of the given matchings. Our algorithm is actually based on the above observation. We bring our algorithm in the following:

We parameterize our algorithm by a value $0 < \gamma < 1$ to be set later. One may optimize the runtime of the algorithm by setting the value of γ in terms of the number of windows and the length of the windows. We first sample a set S of $O(k^\gamma \log k)$ windows. Next, we compute $\mathsf{opt}_{i,j}$ of every window $w_i \in S$ and every other window w_j (not necessarily in S). Finally, we find all the constructive tuples $\langle w_i, w_a, w_b, w_j \rangle$ such that $w_a, w_b \in S$ and update $\widehat{\mathsf{O}}_{\lambda^2}$ accordingly. This is shown in Algorithm 30.1.

The running time of our algorithm is equal to $O(k|S|\mathsf{w}_{\mathsf{max}}^2 + k^2|S|^2\mathsf{w}_{\mathsf{max}})$. The rest of this section is dedicated to proving that what remains in the lcs-graph is sparse.

We first introduce the notion of well-connected pairs.

Algorithm 30.1 Sparsification for O_{λ^2}

procedure QUADRATICSPARSIFICATION($w_1, w_2, \ldots, w_k, \lambda, \epsilon$) ▷ Theorem 30.3.6
  $\gamma \leftarrow 0.1$
  $S \leftarrow 40k^{\gamma} \log k$ i.i.d. samples of $[k]$
  $\widehat{\mathsf{O}}_{\lambda^2} \leftarrow \{0\}^{k \times k}$
  for $w_i \in S$ **do** ▷ Takes $|S| k \mathsf{w}_{\mathsf{max}}^2$ time
    for $j \leftarrow 1$ to k **do**
      $\mathsf{opt}_{i,j}, \mathsf{opt}_{j,i} \leftarrow \mathsf{lcs}(w_i, w_j)$
    end for
  end for
  for $w_i \in S$ **do** ▷ Takes $|S| k \mathsf{w}_{\mathsf{max}}$ time
    for $j \leftarrow 1$ to k **do**
      if $|\mathsf{opt}_{i,j}| > \lambda$ **then**
        $\widehat{\mathsf{O}}_{\lambda^2}[i][j] \leftarrow 1$
      end if
    end for
  end for
  for $i \leftarrow 1$ to k **do** ▷ Takes $k^2 |S|^2 \mathsf{w}_{\mathsf{max}}$ time
    for $j \leftarrow 1$ to k **do**
      for $w_a \in S$ **do**
        for $w_b \in S$ **do**
          if $\left\| \left(\mathsf{opt}_{i,a} \cap \mathsf{opt}_{a,b} \cap \mathsf{opt}_{b,j} \right) \right\| \geq (1 - \epsilon)\lambda^3$ **then**
            $\widehat{\mathsf{O}}_{\lambda^2}[i][j] \leftarrow 1$
          end if
        end for
      end for
    end for
  end for
  return $\widehat{\mathsf{O}}_{\lambda^2}$
end procedure

Definition 30.3.2 (well-connected pair)**.** We say a pair of windows (w_i, w_j) is *well-connected*, if there are at least $k^{2-\gamma}$ pairs of windows (w_a, w_b) such that $\langle w_i, w_a, w_b, w_j \rangle$ is (ϵ, λ)-constructive.

It follows from the definition that well-connected pairs are detected in our algorithm with high probability.

Lemma 30.3.1. *Let $\widehat{\mathsf{O}}_{\lambda^2} \in \{0,1\}^{k\times k}$ denote the output of Algorithm 30.1. With probability at least $1 - 1/\operatorname{poly}(k)$, for all $(i,j) \in [k] \times [k]$ such that (w_i, w_j) is a well-connected pair (Definition 30.3.2), we have $\widehat{\mathsf{O}}_{\lambda^2}[i][j] = 1$.*

Proof. We consider a fixed (i,j) such that pair (w_i, w_j) is well-connected. Let

$$Q_{i,j} = \left\{ (a,b) \;\middle|\; \left\| \mathsf{opt}_{i,a} \cap \mathsf{opt}_{a,b} \cap \mathsf{opt}_{b,j} \right\| \geq (1-\epsilon)\lambda^3 \right\}.$$

Conceptually, we divide the process of sampling S into two phases: we sample $20k^\gamma \log k$ windows in the first phase, and then we sample another $20k^\gamma \log k$ windows in the second phase.

For each $a \in [k]$, let $Q_{i,j,a} = \{b : (a,b) \in Q_{i,j}\}$. Since $\sum_{a\in[k]} |Q_{i,j,a}| = |Q_{i,j}| \geq k^{2-\gamma}$, there are at least

$$\frac{k^{2-\gamma} - k \cdot k^{1-\gamma}/2}{k} = \frac{k^{1-\gamma}}{2}$$

different number a's in $[k]$ such that $|Q_{i,j,a}| \geq \frac{k^{1-\gamma}}{2}$. Hence, in the first phase, there is a sampled number q such that $|Q_{i,j,q}| \geq k^{1-\gamma}/2$ with probability at least

$$1 - \left(1 - \frac{k^{1-\gamma}/2}{k}\right)^{20k^\gamma \log k} < \frac{1}{k^5}.$$

We fix such a q. In the second phase, there is a sampled number r such that $r \in Q_{i,j,q}$ with probability at least

$$1 - \left(1 - \frac{k^{1-\gamma}/2}{k}\right)^{20k^\gamma \log k} < \frac{1}{k^5}.$$

The lemma is obtained by a union bound on all the well-connected pairs (w_i, w_j). □

In what follows, we bound $|\{(i,j) \mid \|\mathsf{lcs}(w_i, w_j)\| \geq \lambda \text{ and } \widehat{\mathsf{O}}_{\lambda^2}[i][j] = 0\}|$. In other words, we prove an upper bound on the number of edges that remain in the graph.

Definition 30.3.3 (NG_λ)**.** Define a graph NG_λ with k vertices and the following edges:

$$E(\mathsf{NG}_\lambda) = \left\{ (i,j) \;\middle|\; \|\mathsf{lcs}(w_i, w_j)\| \geq \lambda \quad \text{and} \quad \widehat{\mathsf{O}}_{\lambda^2}[i][j] = 0 \right\}.$$

We first prove that every $K_{\Omega(\mathsf{w_{gap}}/(\epsilon\lambda^3)),\Omega(\mathsf{w_{gap}}/(\epsilon\lambda^3))}$ subgraph of NG_λ corresponds to at least one constructive tuple.

Lemma 30.3.2. *Let X and Y be two sets of windows such that for every $w_i \in X$ and $w_j \in Y$ there is an (i,j) edge in $E(\mathsf{NG}_\lambda)$, for every $w_i, w_{i'} \in X$, $|w_i| = |w_{i'}|$ and for every $w_j, w_{j'} \in Y$, $|w_j| = |w_{j'}|$. If $|X| \geq \Omega(\mathsf{w_{gap}}/(\epsilon\lambda^3))$ and $|Y| \geq \Omega(\mathsf{w_{gap}}/(\epsilon\lambda^3))$, then there exist $w_i, w_a, w_b, w_j \in X \cup Y$ such that $\langle w_i, w_a, w_b, w_j \rangle$ is (ϵ, λ)-constructive, i.e.,*

$$\left\| \mathsf{opt}_{i,a} \cap \mathsf{opt}_{a,b} \cap \mathsf{opt}_{b,j} \right\| \geq (1-\epsilon)\lambda^3.$$

Proof. By assumption, we know that $|X|, |Y| \geq \Omega(\mathsf{w_{gap}}/(\epsilon\lambda^3))$. Let t_1 denote the window size for each window in X, and t_2 denote the window size for each window in Y.

We carefully select $X' \subseteq X$ and $Y' \subseteq Y$ such that

1. $|X'| \geq \Omega(1/(\epsilon\lambda^3))$.
2. $|Y'| \geq \Omega(1/(\epsilon\lambda^3))$.
3. $|X'|t_1 = (1 \pm O(\epsilon))|Y'|t_2$.

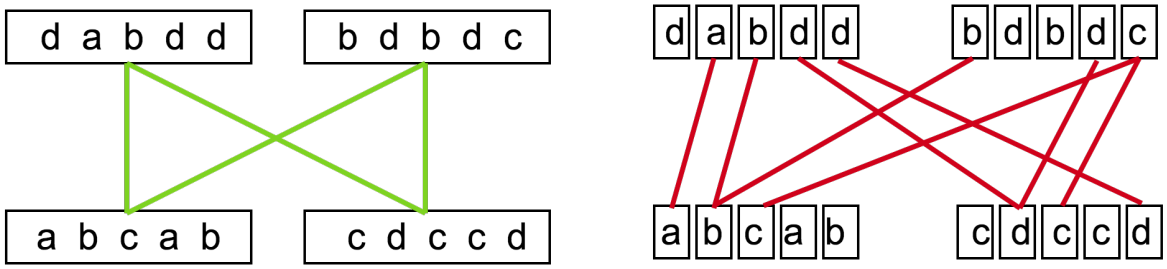


Figure 30.4: The graph on the left is an example of the string-based graph and the graph on the right is an example of the character-based graph.

To do this, if $|X|t_1 > (1 + O(\epsilon))|Y|t_2$, then we set $Y' = Y$ and select X' as an arbitrary subset of X with size $\left\lceil \frac{|Y|t_2}{t_1} \right\rceil$. if $|X|t_1 < (1 - O(\epsilon))|Y|t_2$, then we set $X' = X$ and select Y' an arbitrary subset of Y with size $\left\lceil \frac{|X|t_1}{t_2} \right\rceil$.

We define a window-based (bipartite) graph $G_W = (V_W, E_W)$ to be the subgraph of δ on $V_W = X' \cup Y'$ removing all the edges within X' and all the edges within Y'. Let $l_1 = |X'|$ and $l_2 = |Y'|$. The total number of nodes in window-based graph is $l_1 + l_2$, and the window-based graph is a bi-clique.

We also define a character-based (bipartite) graph $G_C = (V_C, E_C)$ as follows: Each window of X' has t_1 nodes in the character-based graph such that each node represents a character of the window. Similarly, each window of Y' has t_2 nodes in the character-based graph. Two nodes x, y in the character-based graph are adjacent iff $(x, y) \in \mathsf{opt}_{i,j}$ where w_i is the window containing character x and w_j is the window containing character y. Hence, the total number of nodes in character-based graph is $l_1 t_1 + l_2 t_2$. The total number of edges in character-based graph satisfy

$$l_1 l_2 \lambda \sqrt{t_1 t_2} \leq |E(G_C)| \leq l_1 l_2 \min\{t_1, t_2\}.$$

The average degree of the character-based graph is at least $2l_1l_2\lambda\sqrt{t_1t_2}/(l_1t_1+l_2t_2)$. By Blakley-Roy inequality (Lemma 30.5.3), the number of walks of length 3 in the character-based graph is at least

$$|V(G_C)|\cdot\left(2\frac{l_1l_2\lambda\sqrt{t_1t_2}}{l_1t_1+l_2t_2}\right)^3 = (l_1t_1+l_2t_2)\cdot\left(2\frac{l_1l_2\lambda\sqrt{t_1t_2}}{l_1t_1+l_2t_2}\right)^3$$
$$= 8\lambda^3\frac{(l_1l_2\sqrt{t_1t_2})^3}{(l_1t_1+l_2t_2)^2}.$$

But many of the 3-walks are degenerate. For the number of 2-walks, we can upper bound it by

$$\#\text{2-walks} \le (\text{max degree of } G_W)\cdot 2|E(G_C)| \le \max\{l_1,l_2\}\cdot 2(l_1l_2\min\{t_1,t_2\}) = 2(1+O(\epsilon))l_1^2l_2t_1.$$

Note that the #1-walks is $2|E(G_C)|$. We have

$$\begin{aligned}\frac{\#\text{2-walks}+\#\text{1-walks}}{\#\text{3-walks}} &\le \frac{2(1+O(\epsilon))l_1^2l_2t_1+2l_1l_2\min\{t_1,t_2\}}{\left(8\lambda^3\frac{(l_1l_2\sqrt{t_1t_2})^3}{(l_1t_1+l_2t_2)^2}\right)}\\ &< \frac{(l_1t_1+l_2t_2)^2l_1^2l_2t_1}{2\lambda^3l_1^3l_2^3t_1^{1.5}t_2^{1.5}}\\ &\le \frac{(1+O(\epsilon))l_1^4l_2t_1^3}{2\lambda^3l_1^{4.5}l_2^{1.5}t_1^3}\\ &= \frac{1+O(\epsilon)}{2\lambda^3l_1^{0.5}l_2^{0.5}}\\ &\le O(\epsilon),\end{aligned}$$

where the second step and third step follow from $l_1t_1 = (1\pm O(\epsilon))l_2t_2$, and the last step follows from $l_1 = \Omega(1/(\epsilon\lambda^3)), l_2 = \Omega(1/(\epsilon\lambda^3))$. Hence, the total number of 3-paths (3-walks that are not degenerate) is at least

$$8(1-O(\epsilon))\lambda^3\frac{(l_1l_2\sqrt{t_1t_2})^3}{(l_1t_1+l_2t_2)^2}.$$

On the other hand, the number of 4-tuples in window-based graph is $8\binom{l_1}{2}\binom{l_2}{2} \leq 2l_1^2 l_2^2$. Thus, there must exist a 4-tuple containing 3-walks with size at least,

$$\begin{aligned}
8(1-O(\epsilon))\lambda^3 \frac{(l_1 l_2 \sqrt{t_1 t_2})^3}{(l_1 t_1 + l_2 t_2)^2} \cdot \frac{1}{2l_1^2 l_2^2} &= 4(1-O(\epsilon))\lambda^3 \sqrt{t_1 t_2} \cdot \frac{l_1 l_2 t_1 t_2}{(l_1 t_1 + l_2 t_2)^2} \\
&= 4(1-O(\epsilon))\lambda^3 \sqrt{t_1 t_2} \cdot \frac{1}{\frac{l_1 t_1}{l_2 t_2} + 2 + \frac{l_2 t_2}{l_1 t_1}} \\
&\geq 4(1-O(\epsilon))\lambda^3 \sqrt{t_1 t_2} \cdot \frac{1}{4+O(\epsilon)} \\
&\geq (1-O(\epsilon))\lambda^3 \sqrt{t_1 t_2},
\end{aligned}$$

where the third step follows from $l_1 t_1 = (1 \pm O(\epsilon)) l_2 t_2$. Rescaling the factor of ϵ for $|X|$ and $|Y|$ completes the proof. □

Definition 30.3.4 (reconstructive tuple)**.** We say a tuple $\langle w_i, w_a, w_b, w_j \rangle$ is *reconstructive* if it is (ϵ, λ)-constructive and also $(i,a), (a,b), (b,j), (i,j) \in E(\mathsf{NG}_\lambda)$.

Before proving Lemma 30.3.5, we need to define a new graph NF_λ,

Definition 30.3.5 (NF_λ)**.** We construct a graph NF_λ in the following way: for each node $v \in \mathsf{NG}_\lambda$ we keep v in NF_λ with probability $p = k^{-\alpha}/4$ where $\alpha = 1 - \gamma/4$. For each $u, v \in \mathsf{NF}_\lambda$, if $(u,v) \in \mathsf{NG}_\lambda$ then we also draw an edge u, v in NF_λ.

It is obvious that $V(\mathsf{NF}_\lambda) \subseteq V(\mathsf{NG}_\lambda)$ and $E(\mathsf{NF}_\lambda) \subseteq E(\mathsf{NG}_\lambda)$.

Based on definition of NF_λ, we are able to show that if NG_λ is dense, so is NF_λ.

Claim 30.3.3 (NF_λ is a dense graph)**.** *Let* $\mathsf{w}_{\mathsf{layers}}$ *be the number of different window sizes. Let* NG_λ *denote a graph such that* $|E(\mathsf{NG}_\lambda)| \geq k^{2-\beta} \cdot \mathsf{w}_{\mathsf{layers}}^2$ *for some* β *and* NF_λ *be defined as Definition 30.3.5. If* $\gamma/4 - \beta = \Omega(1)$*, then with probability at least* 0.98*, we have*

$$|E(\mathsf{NF}_\lambda)| = \Omega(|V(\mathsf{NF}_\lambda)|^{2-4\beta/\gamma} \cdot \mathsf{w}_{\mathsf{layers}}^2)$$

Proof. Based on the sampling rate, we know that the following hold in expectation:

$$\mathbb{E}[|V(\mathsf{NF}_\lambda)|] = \frac{1}{4k^\alpha}|V(\mathsf{NG}_\lambda)|, \quad \text{and} \quad \mathbb{E}[|E(\mathsf{NF}_\lambda)|] = \frac{1}{4^2k^{2\alpha}}|E(\mathsf{NG}_\lambda)|.$$

Using standard Chernoff bound, we have with probability 0.99,

$$|V(\mathsf{NF}_\lambda)| \in [1/2, 2] \cdot \frac{1}{4k^\alpha}|V(\mathsf{NG}_\lambda)| = [1/2, 2] \cdot \frac{1}{4}k^{1-\alpha} = [1/2, 2] \cdot \frac{1}{4}k^{\gamma/4},$$

To show the concentration of $|E(\mathsf{NF}_\lambda)|$, we define a random variable $x_{u,v}$ such that $x_{u,v} = 1$ if $(u, v) \in E(\mathsf{NF}_\lambda)$, 0 otherwise. Then we have

$$\begin{aligned}
&\mathbb{E}\left[|E(\mathsf{NF}_\lambda)|^2\right] \\
=\ &\mathbb{E}\left[\left(\sum_{(u,v)\in E(\mathsf{NG}_\lambda)} x_{u,v}\right)^2\right] \\
=\ &\mathbb{E}\left[\sum_{(u,v)\in E(\mathsf{NG}_\lambda)} x_{u,v}^2\right] + \mathbb{E}\left[\sum_{(u,v),(u,v')\in E(\mathsf{NG}_\lambda)} \mathbf{1}_{v\neq v'} x_{u,v}x_{u,v'}\right] \\
&+ \mathbb{E}\left[\sum_{(u,v)\in E(\mathsf{NG}_\lambda)} \sum_{(u',v')\in E(\mathsf{NG}_\lambda)} \mathbf{1}_{u\neq v\neq u'\neq v'} x_{u,v}x_{u',v'}\right] \\
\leq\ &p^2|E(\mathsf{NG}_\lambda)| + p^3 \cdot \#\text{2-walks} + p^4 \cdot \left(|E(\mathsf{NG}_\lambda)|^2 - \#\text{2-walks} - |E(\mathsf{NG}_\lambda)|\right).
\end{aligned}$$

Since $\mathbb{E}[|E(\mathsf{NF}_\lambda)|] = p^2|E(\mathsf{NG}_\lambda)|$. Thus,

$$\begin{aligned}
\mathbb{V}[|E(\mathsf{NF}_\lambda)|] =\ &\mathbb{E}\left[(|E(\mathsf{NF}_\lambda)|)^2\right] - (\mathbb{E}[|E(\mathsf{NF}_\lambda)|])^2 \\
&\leq p^2|E(\mathsf{NG}_\lambda)| + p^3 \cdot \#\text{2-walks} \\
&\leq p^2|E(\mathsf{NG}_\lambda)| + p^3|V(\mathsf{NG}_\lambda)| \cdot |E(\mathsf{NG}_\lambda)| \\
&\leq 2p^3|V(\mathsf{NG}_\lambda)| \cdot |E(\mathsf{NG}_\lambda)|. \qquad \text{by } p|V(\mathsf{NG}_\lambda)| \geq 1
\end{aligned}$$

In order to use Chebyshev's inequality, we need to make sure

$$p^2|E(\mathsf{NG}_\lambda)| > 1, \quad p^2|E(\mathsf{NG}_\lambda)| \geq 100(2p^3|V(\mathsf{NG}_\lambda)||E(\mathsf{NG}_\lambda)|)^{1/2}.$$

By $p = k^{\gamma/4-1}/4$ and $|E(\mathsf{NG}_\lambda)| \geq k^{2-\beta}$, we need

$$k^{\gamma/2-\beta}/16 > 1,$$

and the second condition is equivalent to

$$p|E(\mathsf{NG}_\lambda)| \geq 20000|V(\mathsf{NG}_\lambda)|.$$

We just need $k^{\gamma/4-\beta} \geq 80000$.

Using Chebyshev's inequality, we have

$$\Pr\left[\left||E(\mathsf{NF}_\lambda)| - \mathbb{E}[|E(\mathsf{NF}_\lambda)|]\right| \geq \tau\sqrt{\mathbb{V}[|E(\mathsf{NF}_\lambda)|]}\right] \leq \frac{1}{\tau^2}$$

combining with $\sqrt{\mathbb{V}[|E(\mathsf{NF}_\lambda)|]} \leq \frac{1}{100}\,\mathbb{E}[|E(\mathsf{NF}_\lambda)|]$ together implies

$$\Pr\left[\left||E(\mathsf{NF}_\lambda)| - \mathbb{E}[|E(\mathsf{NF}_\lambda)|]\right| \geq \tau\frac{1}{100}\,\mathbb{E}[|E(\mathsf{NF}_\lambda)|]\right] \leq \frac{1}{\tau^2}.$$

Choosing $\tau = 10$, we have with probability 0.99 that $|E(\mathsf{NF}_\lambda)| > 0.9\,\mathbb{E}[|E(\mathsf{NF}_\lambda)|]$, and consequently

$$\begin{aligned}
|E(\mathsf{NF}_\lambda)| &> 0.9\,\mathbb{E}[|E(\mathsf{NF}_\lambda)|] \\
&\geq 0.9\frac{1}{4^2k^{2\alpha}}|E(\mathsf{NG}_\lambda)| \\
&\geq 0.9\frac{1}{4^2}k^{\gamma/2-2}k^{2-\beta}\cdot\mathsf{w}_{\mathsf{layers}}^2 \\
&= 0.9\frac{1}{4^2}k^{\gamma/2-\beta}\cdot\mathsf{w}_{\mathsf{layers}}^2 \\
&\geq 0.9\frac{1}{4^2}k^{\frac{\gamma}{4}(2-4\beta/\gamma)}\cdot\mathsf{w}_{\mathsf{layers}}^2 \\
&\geq \Omega(|V(\mathsf{NF}_\lambda)|^{2-4\beta/\gamma}\cdot\mathsf{w}_{\mathsf{layers}}^2).
\end{aligned}$$

Thus, we complete the proof. □

Next, we show

Claim 30.3.4 (NF_λ has no reconstructive tuple)**.** *Let* NF_λ *be defined as Definition 30.3.5. With probability at least* 0.99*, there is no reconstructive tuple in* NF_λ.

Proof. By Lemma 30.3.1 and the definition of NG_λ, there is no pair of windows (w_i, w_j) in NG_λ such that pair (i, j) is well-connected. Formally speaking, for all pairs of windows (w_i, w_j), there are at most $k^{2-\gamma}$ pairs of windows (w_a, w_b) such that $\langle w_i, w_a, w_b, w_j\rangle$ is (ϵ, λ)-constructive.

For a fixed (i, a, b, j), the probability that we keep it in NF_λ is $(k^{-\alpha}/4)^4$. We can compute the expected number of constructive tuples in NF_λ,

$$\mathbb{E}[\#\text{constructive tuple}] \le k^2 k^{2-\gamma} \cdot (k^{-\alpha}/4)^4 = k^{4-\gamma-4\alpha}/256 = 1/256$$

where the last step follows from $\alpha = 1 - \gamma/4$.

By Markov's inequality, we have

$$\Pr[\#\text{constructive tuple} \ge 1/2] \le \frac{\mathbb{E}[\#\text{constructive tuple}]}{1/2} \le 1/100.$$

Thus, with probability 0.99, there is no reconstructive tuple in NF_λ. □

Finally, we use Lemma 30.3.2 to prove an upper bound on the number of edges in NG_λ.

Lemma 30.3.5. *Let* NG_λ *be as defined in Definition 30.3.3. Then with probability at least* 0.9*,* NG_λ *has at most* $k^{2-\Omega(\gamma\lambda\epsilon/\mathsf{w}_{\mathsf{gap}})} \cdot \mathsf{w}_{\mathsf{layers}}^2$ *edges.*

Proof. We start with assuming

$$|E(\mathsf{NG}_\lambda)| \geq k^{2-\beta} \cdot \mathsf{w}_{\mathsf{layers}}^2,$$

and will get the contradiction at the end.

We construct a graph NF_λ based on Definition 30.3.5. Using Claim 30.3.4, we know that NF_λ has no reconstructive tuple. Next, we are going to show two facts, and they are contradicting to each other by some choice of β.

(Fact A). Note that NF_λ does not satisfy the assumption in Lemma 30.3.2, thus we cannot apply Lemma 30.3.2 to NF_λ directly. Let $\mathsf{w}_{\mathsf{layers}}$ denote the number of different window sizes in total, we can decompose NF_λ into $\mathsf{w}_{\mathsf{layers}}^2$ graphs such that $\mathsf{NF}_\lambda^{i,j}$ only involves size i and j, $\forall i, j \in [\mathsf{w}_{\mathsf{layers}}] \times [\mathsf{w}_{\mathsf{layers}}]$. Now, applying Lemma 30.3.2, for every i, j, we imply that if $\mathsf{NF}_\lambda^{i,j}$ has no reconstructive tuple, then $\mathsf{NF}_\lambda^{i,j}$ has no bipartite graph of certain size, i.e., $K_{\Omega(\mathsf{w}_{\mathsf{gap}}/(\epsilon\lambda^3)),\Omega(\mathsf{w}_{\mathsf{gap}}/(\epsilon\lambda^3))}$.

(Fact B). Using the Turán's Theorem (Lemma 30.5.4), we know for any integer $s \geq 2$, a graph G with n vertices and $\Omega(n^{2-1/s})$ edges has at least one $K_{s,s}$ subgraph. We consider graph NF_λ. Since

$$|E(\mathsf{NF}_\lambda)| = \Omega(|V(\mathsf{NF}_\lambda)|^{2-4\beta/\gamma} \cdot \mathsf{w}_{\mathsf{layers}}^2),$$

by Claim 30.3.3, $|E(\mathsf{NF}_\lambda)| = \Omega(|V(\mathsf{NF}_\lambda)|^{2-4\beta/\gamma} \cdot \mathsf{w}_{\mathsf{layers}}^2)$. By the pigeonhole principle, there exist i and j such that $|E(\mathsf{NF}_\lambda^{i,j})| = \Omega(|V(\mathsf{NF}_\lambda)|^{2-4\beta/\gamma})$. By Turán's Theorem, such an $\mathsf{NF}_\lambda^{i,j}$ contains a large complete bipartite subgraph $K_{\gamma/(4\beta),\gamma/(4\beta)}$.

In order to get a contraction, we just need

$$\gamma/(4\beta) \geq c \cdot \mathsf{w}_{\mathsf{gap}}/(\epsilon\lambda^3).$$

for some sufficiently large constant $c > 1$ related to Lemma 30.3.2. Thus, we have

$$\beta \le \gamma\epsilon\lambda^3/(4c \cdot \mathsf{w}_{\mathsf{gap}}).$$

□

Theorem 30.3.6 (quadratic sparsification for constant λ)**.** *Given k windows $w_1, \cdots, w_k$. Let $\mathsf{w}_{\mathsf{max}} = \max_{i\in[k]} |w_i|$, $\mathsf{w}_{\mathsf{min}} = \min_{i\in[k]} |w_i|$ and $\mathsf{w}_{\mathsf{gap}} = \mathsf{w}_{\mathsf{max}}/\mathsf{w}_{\mathsf{min}}$. Let the number of different window sizes be $\mathsf{w}_{\mathsf{layers}}$. For any $\lambda \in (0,1)$ and $\epsilon \in (0, 1/10)$, there is a randomized algorithm (Algorithm 30.1) that runs in time*

$$O(\mathsf{w}_{\mathsf{max}}^2 k^{1.1} \log k + \mathsf{w}_{\mathsf{max}} k^{2.2} \log^2 k),$$

outputs a table $\widehat{\mathsf{O}}_{\lambda^2} \in \{0,1\}^{k\times k}$ such that

$$\|\mathsf{lcs}(w_i, w_j)\| \ge (1-\epsilon)\lambda^3, \text{ if } \widehat{\mathsf{O}}_{\lambda^2}[i][j] = 1$$

and

$$\left|\left\{(i,j) \;\middle|\; \|\mathsf{lcs}(w_i,w_j)\| \ge \lambda, \quad \text{and} \quad \widehat{\mathsf{O}}_{\lambda^2}[i][j] = 0\right\}\right| = O\left(k^{2-\Omega(\lambda\epsilon/\mathsf{w}_{\mathsf{gap}})} \cdot \mathsf{w}_{\mathsf{layers}}^2\right).$$

The algorithm has success probability $1/\operatorname{poly}(k)$.

Proof. The overall running time is

$$\begin{aligned}
&O(k|S|\mathsf{w}_{\mathsf{max}}^2 + k^2|S|^2\mathsf{w}_{\mathsf{max}}) \\
&= O(k \cdot k^\gamma \log k \cdot \mathsf{w}_{\mathsf{max}}^2 + k^2 \cdot k^{2\gamma} \log^2 k \cdot \mathsf{w}_{\mathsf{max}}) \\
&= O(k^{1.1} \log k \cdot \mathsf{w}_{\mathsf{max}}^2 + k^{2.2} \log^2 k \cdot \mathsf{w}_{\mathsf{max}})
\end{aligned}$$

where the first step follows from $|S| = O(k^\gamma \log k)$, the second step follows from $\gamma = 0.1$.

The guarantee of table $\widehat{\mathsf{O}}_{\lambda^2}$ follows from properties of graph NG_λ (Algorithm 30.1 provides the first property of table in Theorem statement, Lemma 30.3.5 provides the second property of table in Theorem statement). □

30.3.2 Sparsification for O_{λ^3}

One shortcoming of Lemma 30.3.5 is that the number of remaining edges is only truly subquadratic if λ is constant. As we discuss in Section 30.3.1, the overall running time of the algorithm depends on the number of edges in the remaining graph and in order for the running time to be truly subquadratic, we need to reduce the number of edges to truly subquadratic. In this section, we show how one can obtain this bound even when λ is super-constant. However, instead of losing a factor λ^2 in the approximation, our technique loses a factor of $O(\lambda^3)$.

Definition 30.3.6 ($\mathsf{lcs}_{w_a}(w_i, w_j)$)**.** For two windows w_i and w_j, and a window w_a, define $\mathsf{lcs}_{w_a}(w_i, w_j)$ as the length of LCS of $\mathsf{opt}_{i,a}$ and w_j, where $\mathsf{opt}_{i,a}$ denotes the LCS of w_i and w_a.

Notice that unlike lcs, this new definition is not symmetric. Similar to lcs, we also normalize the size of lcs_s by the geometric mean of the lengths of the two windows. That is, we divide the size of the common string by $\sqrt{|w_i||w_j|}$. In what follows, we first give an algorithm for detecting close pairs of windows, and then prove that the number of remaining pairs whose lcs is at least λ is truly subquadratic.

Our algorithm is based on a data structure which we call $\mathsf{lcs\text{-}cmp}(w_a, S, \widetilde{\lambda})$. Roughly

speaking, we will choose $\widetilde{\lambda} = \lambda^2$ when we use it. Let us fix a threshold $\widetilde{\lambda}$ and a window w_a. $\mathsf{lcs\text{-}cmp}(w_a, S, \widetilde{\lambda})$ receives a set S of windows as input such that the size of each window of S is at least $|w_a|$. Upon receiving the windows, it makes a preprocess in time $O(\widetilde{\lambda}^{-2}|w_a|\sum_{w_i \in S}|w_i|)$. Next, $\mathsf{lcs\text{-}cmp}(w_a, S, \widetilde{\lambda})$ would be able to answer each query of the following form in almost linear time:

- For two windows $w_i, w_j \in S$, either certify that $\|\mathsf{lcs}_{w_a}(w_i, w_j)\| < \widetilde{\lambda}/2$ or find a solution for $\|\mathsf{lcs}_{w_a}(w_i, w_j)\|$ with a size of at least $\widetilde{\lambda}^2/8$.

We first show in Section 30.3.2.1, how lcs-cmp gives us a sparsification in truly sublinear time and then discuss the algorithm for lcs-cmp in Section 30.3.2.2.

30.3.2.1 λ^3 Sparsification using lcs-cmp

Similar to what we did in Section 30.3.1, we again use a parameter γ in our algorithm and in the end, we adjust γ to minimize the total running time. In our algorithm, we repeat the following procedure $k^\gamma \log k$ times: sample a window w_a uniformly at random and let S be the set of all the other windows whose length is not smaller than $|w_a|$. Next, we obtain $\mathsf{lcs\text{-}cmp}(w_a, S, \widetilde{\lambda})$ via running the preprocessing step. Finally, for each pair of windows $w_i, w_j \in S$, we make a query to lcs-cmp to verify one of the following two possibilities:

- $\|\mathsf{lcs}_{w_a}(w_i, w_j)\| < \lambda^2/2$;
- $\|\mathsf{lcs}_{w_a}(w_i, w_j)\| \geq \lambda^4/8$.

If the latter is verified we set $\widehat{\mathsf{O}}_{\lambda^3/8}[i][j]$ to 1 otherwise we take no action. In what follows, we prove that after the above sparsification, the number of edges in the remaining graph is

truly sublinear.

Algorithm 30.2 Sparsification for O_{λ^3}

procedure CUBICSPARSIFICATION($w_1, w_2, \ldots, w_k, \lambda$) ▷ Theorem 30.3.10
  $\gamma \leftarrow 0.1$
  $\widetilde{\lambda} \leftarrow \lambda^2$
  for counter $= 1 \to k^\gamma \log k$ **do**
    Sample $a \sim [k]$ uniformly at random
    $S \leftarrow \emptyset$
    for $i = 1 \to k$ **do**
      if $i \neq a$ and $|w_i| \geq |w_a|$ **then**
        $S \leftarrow S \cup \{w_i\}$
      end if
    end for
    $\mathsf{lcs\text{-}cmp}.\textsc{Initial}(w_a, S, \widetilde{\lambda})$ ▷ Algorithm 30.3, Lemma 30.3.11
    for $w_i \in S$ **do**
      for $w_j \in S$ **do**
        if $\mathsf{lcs\text{-}cmp}.\textsc{Query}(w_i, w_j)$ outputs accept **then** ▷ Algorithm 30.3, Lemma 30.3.12
          $\widehat{\mathsf{O}}_{\lambda^3/8}[i][j] \leftarrow 1$ ▷ $\|\mathsf{lcs}_{w_a}(w_i, w_j)\| \geq \widetilde{\lambda}^2/8$
        end if
      end for
    end for
  end for
  return $\widehat{\mathsf{O}}_{\lambda^3/8}$
end procedure

In the rest of this section, we bound $|\{(i,j) \mid \|\mathsf{lcs}(w_i, w_j)\| \geq \lambda \text{ and } \widehat{\mathsf{O}}_{\lambda^3/8}[i][j] = 0\}|$. In other words, we prove an upper bound on the number of edges that remain in the graph. Define a graph NG_λ such that each vertex of NG_λ corresponds to a window and an edge (i,j) means that $\|\mathsf{lcs}(w_i, w_j)\| \geq \lambda$ but $\widehat{\mathsf{O}}_{\lambda^3/8}[i][j] = 0$. The goal is to prove an upper bound on the number of edges of NG_λ. We formally prove this in Lemma 30.3.9.

We define a notation called "close" which is similar to "well-connected" in Section 30.3.1.

In order to avoid confusion, we use "close" instead of well-connected.

Definition 30.3.7 (close)**.** Let $\gamma \in (0,1)$ and $\lambda \in (0,1)$. We say a pair (w_i, w_j) of windows is *close*, if there are at least $k^{1-\gamma}$ windows w_a such that $|w_a| \leq \min\{|w_i|, |w_j|\}$ and $\|\mathsf{lcs}_{w_a}(w_i, w_j)\| \geq \lambda^2$.

Our first observation is that Algorithm 30.2 detects all the close pairs with high probability.

Lemma 30.3.7. *Let $\widehat{\mathsf{O}}_{\lambda^3/8} \in \{0,1\}^{k \times k}$ be the output of Algorithm 30.2. For each $(i,j) \in [k] \times [k]$, if (w_i, w_j) is close (Definition 30.3.7) and $\|\mathsf{lcs}(w_i, w_j)\| \geq \lambda$, then $\widehat{\mathsf{O}}_{\lambda^3/8}[i][j] = 1$ holds with probability at least $1 - 1/\operatorname{poly}(k)$.*

Proof. We consider a fixed (i,j) such that (w_i, w_j) is close. By Definition 30.3.7, there are at least $k^{1-\gamma}$ w_a such that $|w_a| \leq \min\{|w_i|, |w_j|\}$ and $\|\mathsf{lcs}_{w_a}(w_i, w_j)\| \geq \lambda^2$. Then the probability that none of these windows is sampled is at most

$$\left(1 - \frac{k^{1-\gamma}}{k}\right)^t = \left(1 - \frac{1}{k^\gamma}\right)^t = \left(1 - \frac{1}{k^\gamma}\right)^{10k^\gamma \log k} \leq 1/\operatorname{poly}(k).$$

Taking a union over at most k^2 pairs completes the proof. □

Before we proceed to Lemma 30.3.9, we bring Lemma 30.3.8 as an auxiliary observation.

Lemma 30.3.8 (existence of a correlated pair)**.** *Let $\epsilon \in (0,1)$, $\lambda \in (0,1)$. Given a window w_a and a set of windows T. Let $\mathsf{w}_{\mathsf{gap}}$ denote the maximum size of windows divided by the minimum size of windows. Let T contain at most $\mathsf{w}_{\mathsf{layers}}$ different sizes and $|T| = \Omega(\mathsf{w}_{\mathsf{layers}} \cdot \mathsf{w}_{\mathsf{gap}}/(\epsilon\lambda))$. If for each $w_i \in T$ we have $\|\mathsf{lcs}(w_a, w_i)\| \geq \lambda$, then there exist two windows $w_i, w_j \in T$ such that $\|\mathsf{lcs}_{w_a}(w_i, w_j)\| \geq (1-\epsilon)\lambda^2$.*

Proof. Since T has size $\Omega(\mathsf{w}_{\mathsf{layers}} \cdot \mathsf{w}_{\mathsf{gap}}/(\epsilon\lambda))$ and the windows of T have at most $\mathsf{w}_{\mathsf{layers}}$ different sizes, there must exists a subset $T' \subseteq T$ such that $|T'| = \Omega(\mathsf{w}_{\mathsf{gap}}/(\epsilon\lambda))$ and all the windows in T' have the same size. In the rest of the proof, we will focus on w_a and T'.

Let t_0 be the size of w_a and t be the size of all the windows in T'. We consider a window-based bipartite graph, on one side it has one node w_a, and on the other side it has $|T'|$ nodes. Then we can expand the window-based bipartite graph into a character-based bipartite graph, on one side it has t_0 nodes, and on the other side it has $t|T'|$ nodes. Two nodes x, y in the character-based graph are adjacent iff $(x, y) \in \mathsf{opt}_{a,i}$ where w_a is the window containing character x and w_i is the window containing character y. Since $\|\mathsf{lcs}(w_a, w_i)\| \geq \lambda$ for every $w_i \in T'$, the number of edges in character-based bipartite graph is at least $\lambda|T'|\sqrt{t_0 t}$.

For i-th character in window w_a, we use D_i to denote the degree of the corresponding node in the character-based bipartite graph. The number of 2-walks is at least

$$\begin{aligned}
\sum_{i=1}^{t_0} D_i(D_i - 1) &= \left(\sum_{i=1}^{t_0} D_i^2 - \sum_{i=1}^{t_0} D_i\right) \\
&\geq \left(\frac{1}{t_0}(\sum_{i=1}^{t_0} D_i)^2 - \sum_{i=1}^{t_0} D_i\right) && \text{by Cauchy-Schwarz inequality} \\
&\geq \left(\frac{1}{t_0}(\lambda|T'|\sqrt{t_0 t})^2 - (\lambda|T'|\sqrt{t_0 t})\right) && \text{by Eq. (30.1)} \\
&= \left(\lambda^2 t|T'|^2 - \lambda|T'|\sqrt{t_0 t}\right) \\
&\geq (1-\epsilon)\lambda^2 t|T'|^2 && \text{by } |T'| = \Omega(\mathsf{w}_{\mathsf{gap}}/(\epsilon\lambda))
\end{aligned}$$

It remains to show Eq. (30.1). Since the number of edges in the character-based bipartite graph is at least $\lambda|T'|\sqrt{t_0 t}$, we have

$$\sum_{i=1}^{t_0} D_i \geq \lambda|T'|\sqrt{t_0 t}.$$

By $|T'| = \Omega(\mathsf{w}_{\mathsf{gap}}/(\epsilon\lambda))$, we have

$$\lambda|T'|\sqrt{t_0 t} \geq t_0.$$

Due to a simple fact : $x(x-z) > y(y-z)$ as long as $x > y > z > 0$, we have

$$\frac{1}{t_0}(\sum_{i=1}^{t_0} D_i)^2 - \sum_{i=1}^{t_0} D_i \geq \frac{1}{t_0}(\lambda|T'|\sqrt{t_0 t})^2 - (\lambda|T'|\sqrt{t_0 t}). \tag{30.1}$$

The number of 3-tuple (w_i, w_a, w_j) is at most $|T'|^2$. Thus, there must exists a pair such that

$$\|\mathsf{lcs}_{w_a}(w_i, w_j)\| \geq (1-\epsilon)\lambda^2$$

□

We define graph NG_λ and NF_λ as follows:

Definition 30.3.8 (NG_λ)**.** We assume that the edges of NG_λ are directed in the following way: For an edge (i, j) if $|w_i| \neq |w_j|$ the starting point of the edge would be the vertex corresponding to the longer window and the ending point of the edge would be the one corresponding to the shorter window. If both corresponding windows have the same lengths, then we use an arbitrary direction for (i, j).

Definition 30.3.9 (NF_λ)**.** We construct graph NF_λ in the following way : let a denote the node in $V(\mathsf{NG}_\lambda)$ that has the highest outgoing degree, let $V(\mathsf{NF}_\lambda) = N(a)$. We add an edge between vertices (i, j) in NF_λ if $\|\mathsf{lcs}_{w_a}(w_i, w_j)\| \geq (1-\epsilon)\lambda^2$. Give directions to the edges of NF_λ the same way we did it for NG_λ.

Now, we are ready to prove the main observation of this section.

Lemma 30.3.9 (upper bound on $E(\mathsf{NG}_\lambda)$)**.** *Let* NG_λ *be defined as Definition 30.3.3. Then*

$$|E(\mathsf{NG}_\lambda)| = O(k^{2-\gamma} \cdot \mathsf{w}_{\mathsf{layers}} \cdot \mathsf{w}_{\mathsf{gap}}/(\epsilon\lambda)).$$

holds with probability $1 - 1/\operatorname{poly}(k)$.

Proof. We start with assuming

$$|E(\mathsf{NG}_\lambda)| \geq \Omega(k^{2-\beta} \cdot \mathsf{w}_{\mathsf{layers}} \cdot \mathsf{w}_{\mathsf{gap}}/(\epsilon\lambda)).$$

We construct a graph NF_λ based on Definition 30.3.9. By Definition 30.3.9, we know that $|V(\mathsf{NF}_\lambda)| \geq \Omega(k^{1-\beta} \cdot \mathsf{w}_{\mathsf{layers}} \cdot \mathsf{w}_{\mathsf{gap}}/(\epsilon\lambda))$. By choosing some $\beta = \gamma$ we will make a contradiction.

Using Lemma 30.3.8, we have for each set $T \subseteq V(\mathsf{NF}_\lambda)$ with $|T| = \Omega(\mathsf{w}_{\mathsf{layers}} \cdot \mathsf{w}_{\mathsf{gap}}/(\epsilon\lambda))$, there exist two nodes u and v in T such that edge $(u, v) \in \mathsf{NF}_\lambda$.

If we look at the complement of graph NF_λ, we know there is no clique K_{r+1} where $r = O(\mathsf{w}_{\mathsf{layers}} \cdot \mathsf{w}_{\mathsf{gap}}/(\epsilon\lambda))$. Using Turan's theorem (Lemma 30.5.5) we know that complement of graph NF_λ has at most $(1 - \frac{1}{r})\frac{q^2}{2}$ edges, where $q = |V(\mathsf{NF}_\lambda)|$. Then we have

$$|E(\mathsf{NF}_\lambda)| \geq \frac{q(q-1)}{2} - (1 - \frac{1}{r})\frac{q^2}{2} = \frac{q^2}{2r} - \frac{q}{2}.$$

This implies that there is one vertex j whose outgoing degree in NF_λ is at least at least

$$\frac{|V(\mathsf{NF}_\lambda)|}{2r} - \frac{1}{2} \geq |V(\mathsf{NF}_\lambda)| \cdot \Omega\left(\frac{\epsilon\lambda}{\mathsf{w}_{\mathsf{layers}} \cdot \mathsf{w}_{\mathsf{gap}}}\right) \geq k^{1-\beta} \geq k^{1-\gamma}.$$

Thus, there is a node in NG_λ that has a lot of outgoing edges, which means NG_λ has a close pair.

On the other hand, Using definition of NG_λ and Lemma 30.3.7, we know that NG_λ should not contain any close pair. Thus, we get a contradiction. □

Theorem 30.3.10 (cubic sparsification for arbitrary λ)**.** *Given k windows $w_1, \cdots, w_k$. Let $\mathsf{w}_{\mathsf{layers}}$ denote the number of different sizes for windows. Let $\mathsf{w}_{\mathsf{max}} = \max_{i\in[k]} |w_i|$, $\mathsf{w}_{\mathsf{min}} = \min_{i\in[k]} |w_i|$ and $\mathsf{w}_{\mathsf{gap}} = \mathsf{w}_{\mathsf{max}}/\mathsf{w}_{\mathsf{min}}$. For any $\lambda \in (0,1), \gamma \in (0,1)$, there is a randomized algorithm (Algorithm 30.2) that runs in time*

$$O(\lambda^{-4} k^{1+\gamma} \mathsf{w}_{\mathsf{max}}^2 \log k + k^{2+\gamma} \mathsf{w}_{\mathsf{max}} \log k)$$

and outputs a table $\widehat{\mathsf{O}}_{\lambda^3} \in \{0,1\}^{k\times k}$ such that

$$\|\mathsf{lcs}(w_i, w_j)\| \geq \lambda^4/8, \text{ if } \widehat{\mathsf{O}}_{\lambda^3}[i][j] = 1$$

and

$$\left|\left\{(i,j) \mid \|\mathsf{lcs}(w_i,w_j)\| \geq \lambda, \quad \text{and} \quad \widehat{\mathsf{O}}_{\lambda^3}[i][j] = 0\right\}\right| = O(k^{2-\gamma} \mathsf{w}_{\mathsf{layers}} \mathsf{w}_{\mathsf{gap}}/\lambda).$$

The algorithm has success probability $1 - 1/\operatorname{poly}(k)$.

Proof. The running time of each round of Algorithm 30.2 is

$$\begin{aligned}
&= O(\text{ constructing } S \text{ time} + \mathsf{lcs\text{-}cmp}.\textsc{Initial} \text{ time} + |S|^2 \cdot (\mathsf{lcs\text{-}cmp}.\textsc{Query} \text{ time})) \\
&= O(k + \widetilde{\lambda}^{-2}|S|\mathsf{w}_{\mathsf{max}}^2 + |S|^2 \mathsf{w}_{\mathsf{max}}) \\
&= O(\lambda^{-4} k \mathsf{w}_{\mathsf{max}}^2 + k^2 \mathsf{w}_{\mathsf{max}})
\end{aligned}$$

where the last step follows by $\widetilde{\lambda} = \lambda^2$.

Since it repeats $O(k^\gamma \log k)$ rounds, thus the overall running time is

$$O(k^\gamma \log k) \cdot O(\lambda^{-4} k \mathsf{w}_{\mathsf{max}}^2 + k^2 \mathsf{w}_{\mathsf{max}}) = O(\lambda^{-4} k^{1+\gamma} \mathsf{w}_{\mathsf{max}}^2 \log k + k^{2+\gamma} \mathsf{w}_{\mathsf{max}} \log k).$$

□

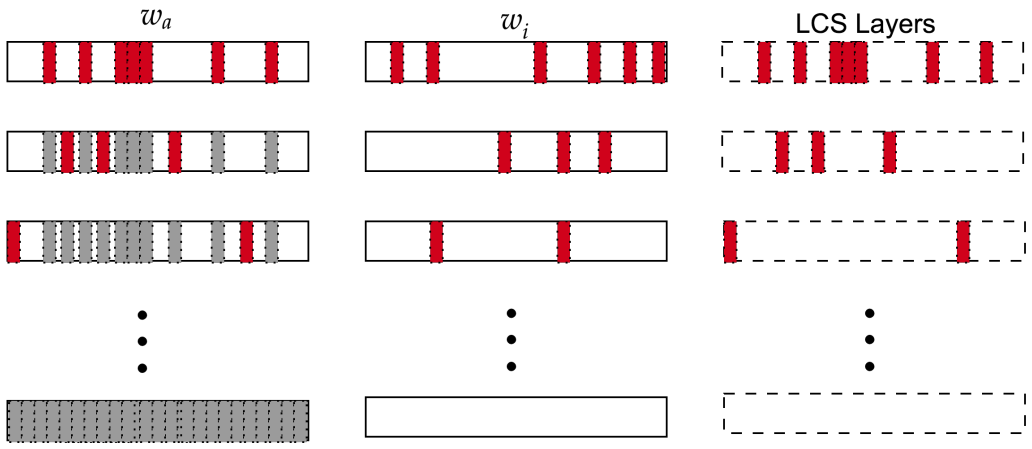


Figure 30.5: computing a set of common subsequences between w_a and w_i for each i such that every character of w_a appears at most once among the sequences for a fixed i.

30.3.2.2 Implementation of lcs-cmp

Given $\widetilde{\lambda}$, w_a and $w_1, w_2, \ldots, w_s$, we present an $O(\widetilde{\lambda}^{-1}|w_a|\sum_{i=1}^{s}|w_i|)$ time preprocessing algorithm and $O(|w_a|)$ time query algorithm for lcs-cmp such that for any $i, j \in [s]$

1. if $\|\mathsf{lcs}_{w_a}(w_i, w_j)\| > \widetilde{\lambda}/2$, then the query algorithm outputs accept;
2. if $\|\mathsf{lcs}_{w_a}(w_i, w_j)\| \leq \widetilde{\lambda}^2/8$, then the query algorithm outputs reject.

Algorithm 30.3 is our lcs-cmp preprocessing and query algorithm. The high level idea is to compute $\mathsf{opt}_{a,i}$ for every $i \in [s]$ and a set of at most $2/\lambda$ common subsequences between w_a and w_i for each i such that every character of w_a appears at most once among the sequences for a fixed i. $\mathsf{lcs}_{w_a}(w_i, w_j)$ is approximated by taking the longest intersection of $\mathsf{opt}_{a,i}$ and some sequence in the set of window w_j. Also see Figure 30.5 for the intuition.

Now we prove that Algorithm 30.3 implements lcs-cmp.

Lemma 30.3.11 (INITIAL)**.** *Given parameter $\widetilde{\lambda}$, a window w_a and a set of windows $w_1, \cdots, w_s$. The* INITIAL *of data structure* lcs-cmp *(Algorithm 30.3) takes $O(\widetilde{\lambda}^{-1}|w_a| \sum_{i=1}^{s} |w_i|)$ time, and outputs $\{X_{a,i}\}_{i\in[s]}$ and $\{Y_{a,i}\}_{i\in[s]}$ such that*

1. *$X_{a,i}$ corresponds to indices of a longest common subsequence between w_a and w_i for every $i \in [s]$.*
2. *$Y_{a,i}$ corresponds to indices of at most $2/\widetilde{\lambda}$ common subsequence between w_a and w_i for every $i \in [s]$.*
3. *If none of the element indices of a common subsequence between w_a and w_i is in $Y_{a,i}$, then the length of this common subsequence is less than $\widetilde{\lambda}|w_a|$.*

Proof. The first property follows from the definition of the algorithm.

For the second and third property, if an $\mathsf{opt}'_{a,i}$ obtained in Line 9 has length less than $\widetilde{\lambda}|w_a|/2$, then the element indices of $\mathsf{opt}'_{a,i}$ are not in $Y_{a,i}$. Hence, the third property holds. Also, it means that every time that Line 13 is executed, the size of $Y_{a,i}$ increases by at least $\widetilde{\lambda}|w_a|/2$. For a fixed $i \in [s]$, Line 13 is executed at most $2/\widetilde{\lambda}$ times. Thus, the second property holds.

The running time is also obtained by the fact that Line 13 is executed at most $2/\widetilde{\lambda}$ times for every fixed $i \in [s]$. □

Lemma 30.3.12 (QUERY)**.** *For any $(i, j) \in [s] \times [s]$, the* QUERY *of data structure* lcs-cmp *(Algorithm 30.3) runs in time $O(|w_a|)$ with the following properties:*

1. *If* $\|\mathsf{lcs}_{w_a}(w_i, w_j)\| > \widetilde{\lambda}/2$*, then it outputs accept.*

2. *If* $\|\mathsf{lcs}_{w_a}(w_i, w_j)\| \leq \widetilde{\lambda}^2/8$*, then it outputs reject.*

Proof. Let $\mathsf{opt}_{a,i,j}$ be the LCS between $\mathsf{opt}_{a,i}$ and w_j. By the definition, we have

$$\|\mathsf{lcs}_{w_a}(w_i, w_j)\| = \frac{|\mathsf{opt}_{a,i,j}|}{\sqrt{|w_i| \cdot |w_j|}}$$

Consider the case of $\|\mathsf{lcs}_{w_a}(w_i, w_j)\| \geq \widetilde{\lambda}/2$. By the third property of Lemma 30.3.11, there are less than $\widetilde{\lambda}|w_a|/4$ elements of $\mathsf{opt}_{a,i,j}$ with indices not in $Y_{a,j}$. So we have

$$|X_{a,i} \cap Y_{a,j}| > \frac{1}{2}\widetilde{\lambda}\sqrt{|w_i||w_j|} - \frac{1}{4}\widetilde{\lambda}|w_a| \geq \frac{1}{4}\widetilde{\lambda}\sqrt{|w_i||w_j|}$$

where the last inequality follows from the fact that $|w_a|$ is smaller than or equal to $|w_i|$ for every $i \in [s]$. Hence the algorithm accepts.

Consider the case of $\|\mathsf{lcs}_{w_a}(w_i, w_j)\| \leq \widetilde{\lambda}^2/8$. By the second property of Lemma 30.3.11, $Y_{a,j}$ corresponds to at most $2/\widetilde{\lambda}$ mutually non-overlapping common subsequence between w_a and w_j. Since $\|\mathsf{lcs}_{w_a}(w_i, w_j)\| \leq \widetilde{\lambda}^2/8$, the set of indices of every such a common subsequence has an intersection with $X_{a,i}$ of size at most $\widetilde{\lambda}^2\sqrt{|w_i||w_j|}/8$. Thus, we have

$$|X_{a,i} \cap Y_{a,j}| \leq \frac{1}{8}\widetilde{\lambda}^2\sqrt{|w_i||w_j|} \cdot \frac{2}{\widetilde{\lambda}} \leq \frac{1}{4}\widetilde{\lambda}\sqrt{|w_i||w_j|}.$$

Hence the algorithm rejects for this case. □

30.4 Longest Increasing Subsequence

We outlined the algorithm in Section 30.1.3.3. Here, we bring the details for each step of the algorithm. In Section 30.4.1 we discuss the solution domains and show how we construct them. Next, in Section 30.4.2 we discuss the details of constructing pseudo-solutions and finally in Section 30.4.3 we show how we can obtain an approximate solution from the pseudo-solutions. Also, in Section 30.4.4 we bring an improvement to the running time at the expense of having a larger approximation factor for the algorithm.

30.4.1 Solution Domains

We assume from now on that $\mathsf{lis}(A) > n\lambda$ holds. As mentioned earlier, we divide the input array into $\sqrt{n}$ subarrays of size $\sqrt{n}$ and denote them by $\mathsf{sa}_1, \mathsf{sa}_2, \ldots, \mathsf{sa}_{\sqrt{n}}$. For a fixed optimal solution opt, our goal is to approximate the smallest and the largest number of each subarray that contributes to opt. Let us refer to these numbers as the *domain* of each subarray. Let $\epsilon = 1/1000$ be accuracy. For a subarray sa_i, we sample k (will be decided later) different elements and refer to them by $a_{j_1}, a_{j_2}, \ldots, a_{j_k}$.

We first prove that,

Lemma 30.4.1 (constructing candidate domains)**.** *Let $\lambda \in (0,1)$, $\epsilon \in (0,1/2)$ and $\delta \in (0,1/10)$. Let sa_i be a length-$\sqrt{n}$ subarray whose contribution to the optimal solution is at least $\epsilon\sqrt{n}\lambda$, i.e., $\mathsf{lis}^{[\mathsf{sm}(\mathsf{sa}_i),\mathsf{lg}(\mathsf{sa}_i)]}(\mathsf{sa}_i) \geq \epsilon\sqrt{n}\lambda$. If we uniformly sample $k = 20\log(1/\delta)/(\lambda\epsilon^2)$ elements $a_{j_1}, a_{j_2}, \ldots, a_{j_k}$ from sa_i, then with probability at least $1-\delta$, there exists a pair $(\alpha, \beta) \in [k] \times [k]$ such that the following two conditions hold*

1. $\mathsf{sm}(\mathsf{sa}_i) \leq a_{j_\alpha} \leq a_{j_\beta} \leq \mathsf{lg}(\mathsf{sa}_i)$,

2. $\mathsf{lis}^{[a_{j_\alpha}, a_{j_\beta}]}(\mathsf{sa}_i) \geq (1-\epsilon)\mathsf{lis}^{[\mathsf{sm}(\mathsf{sa}_i), \mathsf{lg}(\mathsf{sa}_i)]}(\mathsf{sa}_i)$.

Proof. At least $\epsilon\sqrt{n}\lambda$ elements of sa_i appear in opt. Let us put all these elements in an array b in the same order that they appear in sa_i. Then it is obvious that b has at least $\epsilon\sqrt{n}\lambda$ elements. To prove the lemma, we bound the probability that none of the first $\epsilon/2$ fraction of the elements of b is sampled in our algorithm.

$$
\begin{aligned}
& \Pr[\text{none of the elements in the first } \epsilon/2 \text{ fraction of } \mathsf{b} \text{ is sampled}] \\
\leq\ & \left(1 - \frac{\epsilon}{2} \cdot \epsilon\sqrt{n}\lambda \cdot \frac{1}{\sqrt{n}}\right)^k \\
=\ & \left(1 - \frac{\epsilon^2\lambda}{2}\right)^{\frac{2}{\epsilon^2\lambda} \cdot 10\log(1/\delta)} \\
\leq\ & e^{-10\log(1/\delta)} \\
\leq\ & \delta/2,
\end{aligned}
$$

where the first step follows from the fact that b contains at least $\epsilon\lambda\sqrt{n}$ elements, the second step follows from $k = 20\log(1/\delta)/(\lambda\epsilon^2)$, and the third step follows from the fact that the $(1-1/x)^x \leq 1/e$ for $\forall x \geq 4$. Hence, with probability at least $1-\delta/2$, at least one of the elements in the first $\epsilon/2$ fraction of b are sampled.

With the same analysis, one can prove that with probability at least $1-\delta/2$ at least one of the elements in the last $\epsilon/2$ fraction of b are also sampled.

Taking a union bound of two events, with probability at least $1-\delta$, at least one of the elements in the first and at least one of the elements in the last $\epsilon/2$ fraction of b are sampled.

Therefore, the lis of sa_i subject to this interval is at least a $1-\epsilon$ fraction of $\mathsf{lis}^{[\mathsf{sm}(\mathsf{sa}_i),\mathsf{lg}(\mathsf{sa}_i)]}(\mathsf{sa}_i)$.

□

Notice that the average contribution of each subarray to opt is $\sqrt{n}\lambda$ and Lemma 30.4.1 applies to a subarray if its contribution to opt is at least an ϵ fraction of this value. Therefore Lemma 30.4.1 implies that a considerable fraction of the solution is covered by the candidate domains.

Corollary 30.4.2 (existence of a desirable solution)**.** *Let $\lambda \in (0,1)$ such that $\mathsf{lis}(A) \geq n\lambda$ and $\epsilon \in (0, 1/4)$. If we run Algorithm 30.4 with parameter $\delta = \epsilon$ on every subarray independently, then with probability at least $1 - \exp(-\Omega(\epsilon^2\sqrt{n}\lambda))$, there exist a set $T \subseteq [\sqrt{n}]$ and elements α_i and β_i sampled from sa_i for each $i \in T$ such that the following conditions hold:*

1. *For any $i \in T$, $\alpha_i \leq \beta_i$.*
2. *For any $i, j \in T$ satisfying $i < j$, $\beta_i < \alpha_j$.*
3. *$\sum_{i\in T} \mathsf{lis}^{[\alpha_i,\beta_i]}(\mathsf{sa}_i) \geq (1-4\epsilon)\mathsf{lis}(A)$.*

Proof. Lemma 30.4.1 holds for all subarrays whose contribution to opt is at least $\epsilon\sqrt{n}\lambda$. Let $S \subseteq [\sqrt{n}]$ denote the set of coordinates such that for each $i \in S$

$$\mathsf{lis}^{[\mathsf{sm}(\mathsf{sa}_i),\mathsf{lg}(\mathsf{sa}_i)]}(\mathsf{sa}_i) \geq \epsilon\sqrt{n}\lambda.$$

Since $\sum_{i=1}^{\sqrt{n}} \mathsf{lis}^{[\mathsf{sm}(\mathsf{sa}_i),\mathsf{lg}(\mathsf{sa}_i)]}(\mathsf{sa}_i) \geq n\lambda$ and $\mathsf{lis}^{[\mathsf{sm}(\mathsf{sa}_i),\mathsf{lg}(\mathsf{sa}_i)]}(\mathsf{sa}_i) \leq \sqrt{n}$, we have

$$\sum_{i\in S} \mathsf{lis}^{[\mathsf{sm}(\mathsf{sa}_i),\mathsf{lg}(\mathsf{sa}_i)]}(\mathsf{sa}_i) \geq \sum_{i=1}^{\sqrt{n}} \mathsf{lis}^{[\mathsf{sm}(\mathsf{sa}_i),\mathsf{lg}(\mathsf{sa}_i)]}(\mathsf{sa}_i) - \sqrt{n}\cdot\epsilon\sqrt{n}\lambda \geq \mathsf{lis}(A) - \epsilon n\lambda. \tag{30.2}$$

Let $T \subseteq S$ denote the set of coordinates such that for each $i \in T$,

$$\mathsf{lis}^{[\alpha_i,\beta_i]}(\mathsf{sa}_i) \geq (1-\epsilon)\mathsf{lis}^{[\mathsf{sm}(\mathsf{sa}_i),\mathsf{lg}(\mathsf{sa}_i)]}(\mathsf{sa}_i), \quad \text{and} \quad \mathsf{lis}^{[\mathsf{sm}(\mathsf{sa}_i),\mathsf{lg}(\mathsf{sa}_i)]}(\mathsf{sa}_i) \geq \epsilon\sqrt{n}\lambda.$$

Now we show that with probability at least $1 - \exp(-\Omega(\epsilon^2\sqrt{n}\lambda))$,

$$\sum_{i\in T} \mathsf{lis}^{[\mathsf{sm}(\mathsf{sa}_i),\mathsf{lg}(\mathsf{sa}_i)]}(\mathsf{sa}_i) \geq (1-2\epsilon)\sum_{i\in S} \mathsf{lis}^{[\mathsf{sm}(\mathsf{sa}_i),\mathsf{lg}(\mathsf{sa}_i)]}(\mathsf{sa}_i). \tag{30.3}$$

For each $i \in S$, let X_i denote a random variable such that

$$X_i = \begin{cases} \mathsf{lis}^{[\mathsf{sm}(\mathsf{sa}_i),\mathsf{lg}(\mathsf{sa}_i)]}(\mathsf{sa}_i), & \text{with probability of } i \in T; \\ 0, & \text{with probability of } i \notin T, \end{cases}$$

and $X = \sum_{i\in S} X_i$. By Lemma 30.4.1 (with $\delta = \epsilon$), We have $\mathbb{E}[X] \geq (1-\epsilon)\sum_{i\in S} \mathsf{lis}^{[\mathsf{sm}(\mathsf{sa}_i),\mathsf{lg}(\mathsf{sa}_i)]}(\mathsf{sa}_i)$. By Hoeffding bound (Theorem 30.5.2),

$$\begin{aligned}
\Pr[X - \mathbb{E}[X] \geq \epsilon\,\mathbb{E}[X]] &\leq 2\exp\left(-\frac{2\epsilon^2(\mathbb{E}[X])^2}{\sum_{i\in S}\left(\mathsf{lis}^{[\mathsf{sm}(\mathsf{sa}_i),\mathsf{lg}(\mathsf{sa}_i)]}(\mathsf{sa}_i)\right)^2}\right) \\
&\leq 2\exp\left(-\frac{2\epsilon^2(1-\epsilon)^2\left(\sum_{i\in S}\mathsf{lis}^{[\mathsf{sm}(\mathsf{sa}_i),\mathsf{lg}(\mathsf{sa}_i)]}(\mathsf{sa}_i)\right)^2}{n^{3/2}}\right) \\
&\leq 2\exp(-2\epsilon^2(1-\epsilon)^4\sqrt{n}\lambda) \\
&\leq \exp(-\Omega(\epsilon^2\sqrt{n}\lambda)).
\end{aligned}$$

Hence, Equation (30.3) holds with probability at least $1 - \exp(-\Omega(\epsilon^2\sqrt{n}\lambda))$.

Conditioned on Equation (30.3), we have

$$\begin{aligned}
&\sum_{i\in T} \mathsf{lis}^{[\alpha_i, a_{\beta_i}]}(\mathsf{sa}_i) \\
&\geq \sum_{i\in T} (1-\epsilon)\mathsf{lis}^{[\mathsf{sm}(\mathsf{sa}_i),\mathsf{lg}(\mathsf{sa}_i)]}(\mathsf{sa}_i) \\
&\geq (1-\epsilon)(1-2\epsilon)\sum_{i\in S} \mathsf{lis}^{[\mathsf{sm}(\mathsf{sa}_i),\mathsf{lg}(\mathsf{sa}_i)]}(\mathsf{sa}_i) \\
&\geq (1-\epsilon)(1-2\epsilon)(\mathsf{lis}(A) - \epsilon n\lambda) \\
&\geq (1-4\epsilon)\mathsf{lis}(A)
\end{aligned} \tag{30.4}$$

where the first inequality follows from the definition of T, the second inequality follows from Equation (30.3), the third inequality follows from Equation (30.2) and the last inequality follows from $\mathsf{lis}(A) \geq n\lambda$.

Finally, by Equation (30.3) and (30.4), we have

$$\Pr\left[\sum_{i\in T} \mathsf{lis}^{[\alpha_i, a_{\beta_i}]}(\mathsf{sa}_i) \geq (1-4\epsilon)\mathsf{lis}(A)\right] \geq 1 - \exp(-\Omega(\epsilon^2\sqrt{n}\lambda)).$$

□

30.4.2 Constructing Approximately Optimal Pseudo-solutions

We call a sequence of $\sqrt{n}$ intervals $[\ell_1, r_1], [\ell_2, r_2], \ldots, [\ell_{\sqrt{n}}, r_{\sqrt{n}}]$ a pseudo-solution if all of the intervals are monotone. That is $\ell_1 \leq r_1 < \ell_2 \leq r_2 < \ell_3 \leq r_3 \leq \ldots \leq \ell_{\sqrt{n}} \leq r_{\sqrt{n}}$. These intervals denote solution-domains for the subarrays. We also may decide not assign any solution domain to a subarray in which case we show the corresponding interval by $\emptyset$. Indeed, $\emptyset$ does not break the monotonicity of a pseudo-solution. The quality of a pseudo-solution is defined as $\sum_i \mathsf{lis}^{[\ell_i, r_i]}(\mathsf{sa}_i)$. We denote the quality of a pseudo-solution ps by $\mathsf{q}(\mathsf{ps})$.

Another way to interpret Corollary 30.4.2 is that one can construct a pseudo-solution using the sampled elements whose quality is at least a $1-\epsilon$ fraction of $\mathsf{lis}(A)$. In this section, we present an algorithm to construct a small set of pseudo-solutions with the promise that at least one of them has a quality of at least $\mathsf{lis}(A)/t$, where t is the number of pseudo-solutions. Finally, in Section 30.4.3, we present a method to approximate the size of the optimal solution using pseudo-solutions.

We construct the pseudo-solutions via Algorithm 30.5. The input of Algorithm 30.5 is the set of candidate domain intervals obtained by Algorithm 30.4 on every subarray. We first find an assignment of candidate solution domains to the subarrays which is monotone and has the largest number of candidate domain intervals. (This step can be implemented by dynamic programming.) We make a pseudo-solution out of this assignment and update the set of candidate intervals by removing the ones which are used in our pseudo-solution. We then repeat the same procedure to construct the second pseudo-solution and update the candidate solution domains accordingly. We continue on, until the number of solution domains used in our pseudo-solution drops below $\epsilon\sqrt{n}$ in which case we stop.

We first prove in Lemma 30.4.3 that the number of pseudo-solutions constructed in Algorithm 30.5 is bounded by $O(k^2/(\lambda\epsilon))$. Next, we show in Lemma 30.4.4 that at least one of the pseudo-solutions constructed by Algorithm 30.5 has a quality of at least $\Omega(\mathsf{lis}(A)/t)$ where t is the number of pseudo-solutions. Finally we prove in Lemma 30.4.5 that the running time of Algorithm 30.5 is $O(tk^2\sqrt{n}\log n)$.

Lemma 30.4.3 (number of pseudo-solutions)**.** *For each $i \in [\sqrt{n}]$, let cdi_i be a set of at most k^2 candidate domain intervals. Let t denote the number of pseudo-solutions constructed in Algorithm 30.5. Then, we have $t \leq k^2/(\lambda\epsilon)$.*

Proof. Note that for each subarray we have at most k^2 candidate domain intervals. Since there are $\sqrt{n}$ subarrays, then in total we have $\sqrt{n}k^2$ candidate domain intervals. Each time we construct a pseudo-solution, the total number of the candidate domain intervals is decreased by at least $\epsilon\sqrt{n}\lambda$. Thus, the total number of pseudo-solutions t can be upper bounded,

$$t \leq \frac{\sqrt{n}k^2}{\epsilon\sqrt{n}\lambda} \leq \frac{k^2}{\lambda\epsilon}.$$

□

Lemma 30.4.4 (quality of pseudo-solutions)**.** *Let* $\mathsf{ps}_1, \mathsf{ps}_2, \ldots, \mathsf{ps}_t$ *be the pseudo-solutions constructed by Algorithm 30.5. If* $\mathsf{lis}(A) \geq n\lambda$ *holds, then with probability at least* $1 - \exp(-\Omega(\sqrt{n}\lambda))$*, there exists an* $i \in [t]$ *such that*

$$\mathsf{q}(\mathsf{ps}_i) \geq \frac{\mathsf{lis}(A)}{2t}.$$

Proof. Let us focus again on the actual solution domains

$$[\mathsf{sm}(\mathsf{sa}_1), \mathsf{lg}(\mathsf{sa}_1)], [\mathsf{sm}(\mathsf{sa}_2), \mathsf{lg}(\mathsf{sa}_2)], \ldots, [\mathsf{sm}(\mathsf{sa}_{\sqrt{n}}), \mathsf{lg}(\mathsf{sa}_{\sqrt{n}})].$$

We define set $S \subseteq [\sqrt{n}]$ such that

$$\mathsf{lis}^{[\mathsf{sm}(\mathsf{sa}_i), \mathsf{lg}(\mathsf{sa}_i)]} \geq \epsilon\sqrt{n}\lambda, \forall i \in S.$$

Using Corollary 30.4.2 with $\epsilon = 1/10$, we know that there is a monotone pseudo-solution $[\alpha_i, \beta_i]_{i \in T}$ ($T \subseteq S$) such that $[\alpha_i, \beta_i]$ are candidate domain intervals and

$$\sum_{i \in T} \mathsf{lis}^{[\alpha_i, \beta_i]}(\mathsf{sa}_i) \geq (1 - 4\epsilon)\mathsf{lis}(A).$$

Denote this pseudo-solution as sol.

At the time we terminate Algorithm 30.5, there are at most $\epsilon\sqrt{n}\lambda$ candidate domain intervals of sol does not belongs to any pseudo-solution of the pseudo-solution set. Also, since each candidate domain interval contributes at most $\sqrt{n}$ to the quality of the pseudo-solution containing the interval, we have

$$\sum_{i=1}^{t} \mathsf{q}(\mathsf{ps}_i) \geq (1-4\epsilon)\mathsf{lis}(A) - \epsilon\sqrt{n}\lambda \cdot \sqrt{n} \geq (1-4\epsilon)\mathsf{lis}(A) - \epsilon\mathsf{lis}(A) = (1-5\epsilon)\mathsf{lis}(A).$$

Thus, there exists an $i \in [t]$ such that

$$\mathsf{q}(\mathsf{ps}_i) \geq (1-5\epsilon)\mathsf{lis}(A)/t = \mathsf{lis}(A)/(2t).$$

□

Lemma 30.4.5 (running time)**.** *For each $i \in [\sqrt{n}]$, let cdi_i be a set of at most k^2 candidate domain intervals. Let t denote the number of pseudo-solutions. The running time of Algorithm 30.5 is bounded by $O(tk^2\sqrt{n}\log n)$.*

Proof. Lemma 30.4.3 states that Algorithm 30.5 terminates after constructing t pseudo-solutions.

Now we show that constructing each pseudo-solution takes time $O(k^2\sqrt{n}\log n)$. Our solution is based on a dynamic programming technique. Let $D : [\sqrt{n}] \times [k^2] \to \mathbb{N}$ be an array such that $D[i][j]$ stores the size of the largest monotone pseudo-solution for the first i subarrays which ends with the j'th candidate domain interval of sa_i. Using classic segment-tree data structure (this data structure can be found in many textbooks, e.g. [CLRS09]), one can compute the value of $D[i][j]$ in time $O(\log n)$ from the previously computed elements of the array.

Thus, the total running is bounded by $O(tk^2\sqrt{n}\log n)$. □

30.4.3 Evaluating the Pseudo-solutions

We finally use a concentration bound to show that the quality of a pseudo-solution can be approximated well by sampling a small number of subarrays. Since a pseudo-solution specifies the range of the numbers used in every subarray, the quality of a pseudo-solution, or in other words, the size of the corresponding increasing subsequence of a pseudo-solution can be formulated as

$$\mathsf{q}(\mathsf{ps}) = \sum_{i=1}^{\sqrt{n}} \mathsf{lis}^{[\ell_i, r_i]}(\mathsf{sa}_i)$$

where $[\ell_i, r_i]$ denotes the corresponding solution domain of ps for sa_i.

In Lemma 30.4.6, we prove that by sampling $O(\log^{O(1)} n/\lambda^4)$ many subarrays and computing $\mathsf{lis}^{[\ell_i, r_i]}(\mathsf{sa}_i)$ for them, one can approximate the quality of a pseudo-solution pretty accurately.

Lemma 30.4.6 (the quality of pseudo-solution)**.** *Let $\lambda \in (0,1)$ and ϵ be a constant in $(0, 1/100)$. Let $\mathsf{ps}_1, \mathsf{ps}_2, \cdots, \mathsf{ps}_t$ be a set of t pseudo-solutions. With probability at least $1 - \exp(-\Omega(\log^2 n))$, Algorithm 30.6 runs in time $O(t^2\sqrt{n}\log^{O(1)} n/\lambda)$ such that ,*

1. *If there exists an $i \in [t]$, $\mathsf{q}(\mathsf{ps}_i) \geq \frac{\lambda n}{2t}$, then the algorithm outputs an estimation at least $\frac{\lambda n}{4t}$.*
2. *If $\mathsf{q}(\mathsf{ps}_i) < \frac{\lambda n}{8t}$ for al the $i \in [t]$, then the algorithm outputs an estimation smaller than $\frac{\lambda n}{4t}$.*

Proof. By Chernoff bound, with probability $1 - \exp(-\Omega(\log^3 n))$ most $2p\sqrt{n} = \frac{2000t\log^4 n}{\epsilon^4\lambda}$ subarrays are sampled. For each pseudo-solution, Algorithm 30.6 runs the algorithm for longest increasing subsequence once for every sampled subarray. Hence, the running time of the algorithm is $O(t^2\sqrt{n}\log^{O(1)} n/\lambda)$.

Consider an arbitrary pseudo-solution $\mathsf{ps} \in \{\mathsf{ps}_1, \ldots, \mathsf{ps}_t\}$ for now. We show that

$$\Pr\left[\left(1-\frac{\epsilon}{4}\right)^2\left(\mathsf{q}(\mathsf{ps}) - \frac{\epsilon\lambda^4 n}{100t}\right)\widetilde{\mathsf{q}}(\mathsf{ps}) \le \left(1+\frac{\epsilon}{4}\right)^2\mathsf{q}(\mathsf{ps}) + \frac{\epsilon\lambda^4 n}{100t}\right] \le 1 - \exp(-\Omega(\log^3 n)).$$

Then the lemma holds by a union bound on all the pseudo-solutions.

Let T be the set of subarray indices such that $i \in T$ iff there is a non-empty interval $[\ell_i, r_i]$ of ps corresponding to subarray sa_i. Let $p = \frac{1000t\log^4 n}{\epsilon^4\lambda\sqrt{n}}$ be the probability of sampling a subarray. For each $i \in T$, let X_i denote a random variable such that

$$X_i = \begin{cases} 1, & \text{with prob. } p; \\ 0, & \text{with prob. } 1-p. \end{cases}$$

and

$$X = \sum_{i\in T}\frac{1}{p}\mathsf{lis}^{[\ell_i,r_i]}(\mathsf{sa}_i)X_i.$$

We have

$$\mathbb{E}[X] = \mathbb{E}\left[\sum_{i\in T}\frac{1}{p}\mathsf{lis}^{[\ell_i,r_i]}(\mathsf{sa}_i)X_i\right] = \sum_{i\in T}\mathsf{lis}^{[\ell_i,r_i]}(\mathsf{sa}_i) = \mathsf{q}(\mathsf{ps}).$$

Let

$$T_j = \left\{i \in T : (1+\epsilon/4)^{j-1} \le \mathsf{lis}^{[\ell_i,r_i]}(\mathsf{sa}_i) < (1+\epsilon/4)^j\right\}$$

for integer $1 \le j \in \left\lceil \log_{1+\epsilon}\sqrt{n}\right\rceil$.

Let $\Delta = \frac{\epsilon^2 \lambda \sqrt{n}}{1000t \log n}$. Consider T_j's such that $|T_j| \geq \Delta$. The contribution of $\mathsf{q}(\mathsf{ps})$ mostly comes from T_j in the following sense

$$\sum_{j:|T_j|\geq\Delta} \sum_{i\in T_j} \mathsf{lis}^{[\ell_i,r_i]}(\mathsf{sa}_i) \leq \mathsf{q}(\mathsf{ps}) \tag{30.5}$$

and

$$\begin{aligned}\sum_{j:|T_j|\geq\Delta} \sum_{i\in T_j} \mathsf{lis}^{[\ell_i,r_i]}(\mathsf{sa}_i) =& \mathsf{q}(\mathsf{ps}) - \sum_{j:|T_j|<\Delta} \sum_{i\in T_j} \mathsf{lis}^{[\ell_i,r_i]}(\mathsf{sa}_i) \\ \geq& \mathsf{q}(\mathsf{ps}) - \Delta\sqrt{n} \cdot \left\lceil \log_{1+\epsilon} \sqrt{n} \right\rceil \\ \geq& \mathsf{q}(\mathsf{ps}) - \frac{\epsilon\lambda n}{100t}.\end{aligned} \tag{30.6}$$

Now we bound the random variable $\sum_{j:|T_j|\geq\Delta} \sum_{i\in T_j} \mathsf{lis}^{[\ell_i,r_i]}(\mathsf{sa}_i) X_i/p$. By Chernoff bound, for each j such that $|T_j| \geq \Delta$, we have

$$\Pr\left[\left(1-\frac{\epsilon}{4}\right) p|T_j| \leq \sum_{i\in T_j} X_i \leq \left(1+\frac{\epsilon}{4}\right) p|T_j|\right] \geq 1-\exp(-\Omega(\epsilon^2 p|T_j|)) = 1-\exp(-\Omega(\log^3 n)). \tag{30.7}$$

Notice that for set T_j,

$$\frac{(1+\epsilon/4)^{j-1}}{p} \sum_{i\in T_j} X_i \leq \sum_{i\in T_j} \frac{1}{p} \mathsf{lis}^{[\ell_i,r_i]}(\mathsf{sa}_i) X_i \leq \frac{(1+\epsilon/4)^j}{p} \sum_{i\in T_j} X_i. \tag{30.8}$$

We have

$$
\begin{aligned}
&\Pr\left[\left(1-\frac{\epsilon}{4}\right)^2\sum_{i\in T_j}\mathsf{lis}^{[\ell_i,r_i]}(\mathsf{sa}_i)\le\sum_{i\in T_j}\frac{1}{p}\mathsf{lis}^{[\ell_i,r_i]}(\mathsf{sa}_i)X_i\le\left(1+\frac{\epsilon}{4}\right)^2\sum_{i\in T_j}\mathsf{lis}^{[\ell_i,r_i]}(\mathsf{sa}_i)\right]\\
\ge&\Pr\left[\left(1-\frac{\epsilon}{4}\right)^2\sum_{i\in T_j}\mathsf{lis}^{[\ell_i,r_i]}(\mathsf{sa}_i)\le\frac{(1+\epsilon/4)^{j-1}}{p}\sum_{i\in T_j}X_i\right.\\
&\qquad\left.\bigwedge\frac{(1+\epsilon/4)^{j}}{p}\sum_{i\in T_j}X_i\le\left(1+\frac{\epsilon}{4}\right)^2\sum_{i\in T_j}\mathsf{lis}^{[\ell_i,r_i]}(\mathsf{sa}_i)\right]\\
\ge&\Pr\left[\left(1-\frac{\epsilon}{4}\right)^2\left(1+\frac{\epsilon}{4}\right)^j|T_j|\le\frac{(1+\epsilon/4)^{j-1}}{p}\sum_{i\in T_j}X_i\right.\\
&\qquad\left.\bigwedge\ \frac{(1+\epsilon/4)^{j}}{p}\sum_{i\in T_j}X_i\le\left(1+\frac{\epsilon}{4}\right)^{j+1}|T_j|\right]\\
\ge&\Pr\left[\left(1-\frac{\epsilon}{4}\right)p|T_j|\le\sum_{i\in T_j}X_i\ \bigwedge\ \sum_{i\in T_j}X_i\le\left(1+\frac{\epsilon}{4}\right)p|T_j|\right]\\
=&1-\exp(-\Omega(\log^3 n)),
\end{aligned}
\tag{30.9}
$$

where the first inequality follows from Equation (30.8), the second inequality follows from the definition of T_j and the last inequality follows from Equation (30.7). By Equation (30.5),

(30.6), (30.10) and union bound, we have

$$\begin{aligned}
&\Pr\left[\left(1-\frac{\epsilon}{4}\right)^2\left(\mathsf{q}(\mathsf{ps})-\frac{\epsilon\lambda n}{100t}\right)\le\sum_{j:|T_j|\ge\Delta}\sum_{i\in T_j}\frac{1}{p}\mathsf{lis}^{[\ell_i,r_i]}(\mathsf{sa}_i)X_i\le\left(1+\frac{\epsilon}{4}\right)^2\mathsf{q}(\mathsf{ps})\right]\\
\ge&\Pr\left[\left(1-\frac{\epsilon}{4}\right)^2\sum_{j:|T_j|\ge\Delta}\sum_{i\in T_j}\mathsf{lis}^{[\ell_i,r_i]}(\mathsf{sa}_i)\le\sum_{j:|T_j|\ge\Delta}\sum_{i\in T_j}\frac{1}{p}\mathsf{lis}^{[\ell_i,r_i]}(\mathsf{sa}_i)X_i\right.\\
&\qquad\left.\le\sum_{j:|T_j|\ge\Delta}\left(1+\frac{\epsilon}{4}\right)^2\sum_{i\in T_j}\mathsf{lis}^{[\ell_i,r_i]}(\mathsf{sa}_i)\right]\\
\ge&1-O\left(\frac{\log n}{\epsilon}\right)\exp(-\Omega(\log^3 n))\\
=&1-\exp(-\Omega(\log^3 n)).
\end{aligned}\tag{30.10}$$

Now we bound random variable $\sum_{j:|T_j|<\Delta}\sum_{i\in T_j}\mathsf{lis}^{[\ell_i,r_i]}(\mathsf{sa}_i)X_i/p$. If $|T_j|<\Delta$, then by Chernoff bound we have

$$\Pr\left[\sum_{i\in T_j}X_i\le\frac{2\log^3 n}{\epsilon^2}\right]\ge 1-\exp(-\Omega(\log^3 n)).$$

By union bound we have

$$\Pr\left[\sum_{j:|T_j|<\Delta}\sum_{i\in T_j}\frac{1}{p}\mathsf{lis}^{[\ell_i,r_i]}(\mathsf{sa}_i)X_i\le\frac{2\log^3 n}{\epsilon^2}\cdot\frac{\sqrt{n}}{p}\cdot\left\lceil\log_{1+\epsilon}\sqrt{n}\right\rceil<\frac{\epsilon\lambda n}{100t}\right]\ge 1-\exp(-\Omega(\log^3 n))).\tag{30.11}$$

If $\mathsf{q}(\mathsf{ps})\ge\frac{\lambda n}{2t}$, then then by Equation (30.10)

$$\begin{aligned}
&\Pr\left[X>\frac{\lambda n}{4t}\right]\\
\ge&\Pr\left[\left(1-\frac{\epsilon}{4}\right)^2\left(\mathsf{q}(\mathsf{ps})-\frac{\epsilon\lambda n}{100t}\right)\le\sum_{j:|T_j|\ge\Delta}\sum_{i\in T_j}\frac{1}{p}\mathsf{lis}^{[\ell_i,r_i]}(\mathsf{sa}_i)X_i\right]\\
\ge&1-\exp(-\Omega(\log^3 n)).
\end{aligned}$$

If $\mathsf{q}(\mathsf{ps}) < \frac{\lambda n}{8t}$, then by Equation (30.10) and Equation (30.11),

$$\begin{aligned}
&\Pr\left[X < \frac{\lambda n}{4t}\right] \\
\geq &\Pr\left[\sum_{j:|T_j|\geq\Delta}\sum_{i\in T_j}\frac{1}{p}\mathsf{lis}^{[\ell_i,r_i]}(\mathsf{sa}_i)X_i \leq \left(1+\frac{\epsilon}{4}\right)^2\mathsf{q}(\mathsf{ps}) \bigwedge \sum_{j:|T_j|<\Delta}\sum_{i\in T_j}\frac{1}{p}\mathsf{lis}^{[\ell_i,r_i]}(\mathsf{sa}_i)X_i \leq \frac{\epsilon\lambda n}{100t}\right] \\
\geq &1-\exp(-\Omega(\log^3 n)).
\end{aligned}$$

□

Putting all the previous Lemmas together gives the following result:

Corollary 30.4.7 (algorithm for LIS decision problem)**.** *Given a length-n sequence A, let $\lambda \in [1/n, 1]$. There is a randomized algorithm that runs in time $O(\lambda^{-7}\sqrt{n}\log^{O(1)} n)$ such that with probability $1 - 1/\operatorname{poly}(n)$*

- *The algorithm accepts if $\mathsf{lis}(A) > n\lambda$.*
- *The algorithm rejects if $\mathsf{lis}(A) = O(n\lambda^4)$.*

Proof. The correctness follows from Lemma 30.4.6, Lemma 30.4.4, and Lemma 30.4.5.

Running time: The running time is

$$\begin{aligned}
\text{time} &= \underbrace{O(tk^2\sqrt{n}\log^{O(1)} n)}_{\text{Lemma 30.4.5}} + \underbrace{O(t^2\lambda^{-1}\sqrt{n}\log^{O(1)} n)}_{\text{Lemma 30.4.6}} \\
&= O(t^2\lambda^{-1}\sqrt{n}\log^{O(1)} n) && \text{by Lemma 30.4.3} \\
&= O(k^4\lambda^{-3}\sqrt{n}\log^{O(1)} n) && \text{by Lemma 30.4.3} \\
&= O(\lambda^{-7}\sqrt{n}\log^{O(1)} n)
\end{aligned}$$

Thus, we complete the proof. □

Finally, by starting with $\lambda = 1$ and iteratively multiplying λ by a $1/(1+\epsilon)$ factor until a solution is found, we can approximate $\mathsf{lis}(A)$ within an approximation factor of $O(\lambda^3)$.

Theorem 30.4.8 (polynomial approximation for LIS)**.** *Given a length-n sequence A such that $\mathsf{lis}(A) = n\lambda$ where $\lambda \in [1/n, 1]$ is unknown to the algorithm. There is an algorithm that runs in time $\widetilde{O}(\lambda^{-7}\sqrt{n})$ and outputs a number est such that*

$$\Omega(\mathsf{lis}(A)\lambda^3) \leq \mathsf{est} \leq O(\mathsf{lis}(A)).$$

with probability at least $1 - 1/\operatorname{poly}(n)$.

We remark that one can turn Theorem 30.4.8 into an algorithm with running time $\widetilde{O}(n^{17/20})$ by considering two cases separately. If $\lambda < n^{-1/20}$ we sample the array with a rate of $n^{-3/20}$ and compute the LIS for the sampled array. Otherwise, the running time of the algorithm is already bounded by $\widetilde{O}(n^{17/20})$.

30.4.4 An $O(n^\kappa)$ Time Algorithm via Bootstrapping

Let us move a step backward and analyze the previous algorithm for obtaining an $O(\lambda^3)$ approximate solution. We first divide the input array into $\sqrt{n}$ subarrays of size $\sqrt{n}$ and after constructing the pseudo-solutions, we sample $O(\lambda^4)$ subarrays to estimate the size of the solution for pseudo-solutions. The reason we set the size of the subarrays to $\sqrt{n}$ is that there is a trade-off between the first and the last steps of the algorithm. More precisely, if we have more than $\sqrt{n}$ subarrays then the number of samples we draw in the beginning would exceed $O_\lambda(\sqrt{n})$. On the other hand, having fewer than $\sqrt{n}$ subarrays results in larger subarrays which would be costly in the last step.

If one favors the running time over the approximation factor, the following improvement can be applied to the algorithm: In the last step of the algorithm, instead of sampling the entire subarrays and computing lis for every pseudo-solution, we recursively call the same procedure to approximate the size of the solution for each subarray. This way, having large subarrays would no longer be an issue and therefore we can have fewer subarrays to improve the number of samples we draw in the first step of the algorithm.

More formally, in order to obtain a running time of $O(\text{poly}(\lambda)n^{\kappa})$ we set the size of each subarray to $n^{1-\kappa}$ and therefore after constructing the pseudo-solutions, the problem boils down to approximating the solution for $\text{poly}(\lambda)$ many subarrays of length $n^{1-\kappa}$. By running the same algorithm, we would have n^{κ} subarrays of length $n^{1-2\kappa}$ in the second iteration. After $1/\kappa - 1$ iterations, the subarrays are small enough and we can access all their elements in time $O(\text{poly}(\lambda)n^{\kappa})$. Of course, this imposes a factor of $\text{poly}(\lambda)$ to the approximation.

By generalizing the ideas from previous subsections, we show that if there is an algorithm for LIS with approximation factor $f(\lambda)$, then we can get a $\left(f\left(\frac{\lambda^4}{2^{32}}\right) \cdot \frac{\lambda^4}{2^{33}}\right)$-approximate LIS algorithm with better running time using the $f(\lambda)$-approximate algorithm as a subroutine.

Lemma 30.4.9. *Assume we partition the sequence into ζ subarrays, where ζ is polynomially related to the length of the input sequence. For parameter $\lambda \in (0,1)$, let* Oracle *be a $f(\lambda)$-approximate LIS algorithm (with respect to a domain interval) with running time $g(n,\lambda)$ and success probability $1 - \exp(-\Omega(\log^2 n))$ where n is the length of the input sequence. Then Algorithm 30.7 using* Oracle *as a subroutine is a $\left(f\left(\frac{\lambda^4}{2^{32}}\right) \cdot \frac{\lambda^4}{2^{33}}\right)$-approximate LIS algorithm*

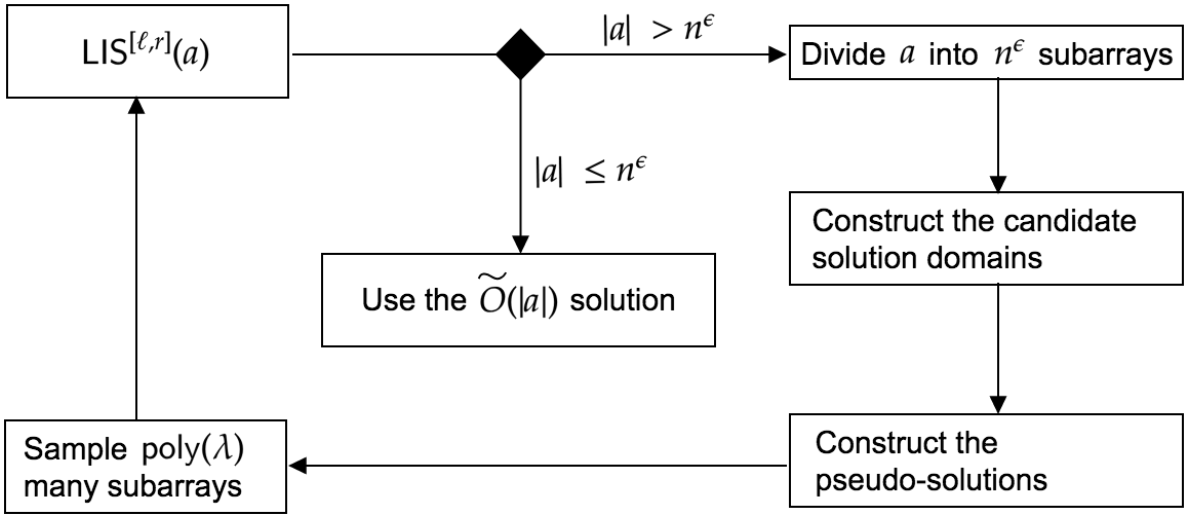


Figure 30.6: The flowchart of the $O_\lambda(n^\epsilon)$ time algorithm is shown.

with

$$O\left(\lambda^{-4} g\left(\frac{n}{\zeta}, \frac{\lambda^4}{2^{32}}\right) \log^{O(1)} n + \lambda^{-7}\zeta \log^{O(1)} n\right)$$

running time and success probability $1-\exp(-\Omega(\log^2 n))$*, where* ζ *is the number of subarrays.*

Proof. We first prove the correctness of the algorithm. Let A be a sequence of length n, and $\mathsf{sa}_1, \ldots, \mathsf{sa}_\zeta$ be the subarrays.

Consider the case of $\mathsf{lis}^{[\ell,r]}(A) \geq \lambda n$. By Corollary 30.4.2 and Lemma 30.4.4 with $\epsilon = \delta = 1/10$, with probability $1 - \exp(-\Omega(\zeta\lambda))$, there exists a pseudo-solution ps_j within interval $[\ell, r]$ satisfying

$$\mathsf{q}(\mathsf{ps}_j) \geq \frac{\mathsf{lis}^{[\ell,r]}(A)}{2t} \geq \frac{\mathsf{lis}^{[\ell,r]}(A)\lambda\epsilon}{2k^2} \geq \frac{\mathsf{lis}^{[\ell,r]}(A)\lambda\epsilon}{2 \cdot 20^2 \log^2(1/\delta)/(\lambda^2\epsilon^4)} \geq \frac{\epsilon^5\lambda^4}{800\log^2(1/\delta)} n \geq \frac{\lambda^4}{2^{31}} n.$$

Let α denote the number of subarrays sa_i such that $\mathsf{lis}^{[\ell_i,r_i]}(\mathsf{sa}_i) \geq \lambda_0 n/\zeta$ where $[\ell_i, r_i]$ is the

interval for subarray sa_i in ps_j. We have

$$\alpha \geq \frac{\mathsf{q}(\mathsf{ps}_j) - \lambda^4 n/2^{32}}{n/\zeta} \geq \frac{\lambda^4 \zeta}{2^{32}}.$$

By Chernoff bound, Step 14 to Step 22 of Algorithm 30.7 accepts on ps_j with probability at least $1 - \exp(-\Omega(\log^2 n))$.

Consider the case of

$$\mathsf{lis}^{[\ell,r]}(A) \leq f\left(\frac{\lambda^4}{2^{32}}\right)\frac{\lambda^4}{2^{33}}n.$$

Then for any pseudo-solution ps_j, we have

$$\mathsf{q}(\mathsf{ps}_j) \leq f\left(\frac{\lambda^4}{2^{32}}\right)\frac{\lambda^4}{2^{33}}n.$$

Let β denote the number of subarrays sa_i such that $\mathsf{lis}^{[\ell_i,r_i]}(\mathsf{sa}_i) \geq f(\lambda^4/2^{32})n/\zeta$ where $[\ell_i, r_i]$ is the interval for subarray sa_i in ps_j. We have

$$\beta \leq \frac{\mathsf{q}(\mathsf{ps}_j)}{f(\lambda^4/2^{32})n/\zeta} \leq \frac{\lambda^4}{2^{33}}\zeta.$$

By Chernoff bound, Step 14 to Step 22 of Algorithm 30.7 do not accept on ps_j with probability at least $1 - \exp(-\Omega(\log^2 n))$. By union bound, Algorithm 30.7 rejects with probability at least $1 - \exp(-\Omega(\log^2 n))$.

Hence, Algorithm 30.7 is a $\left(f\left(\frac{\lambda^4}{2^{32}}\right) \cdot \frac{\lambda^4}{2^{33}}\right)$-approximate algorithm for LIS of length n with success probability at least $1 - \exp(-\Omega(\log^2 n))$.

Finally, we discuss the running time of Algorithm 30.7. By the definition of procedures ConstructCandidateDomains and ConstructPseudoSolutions, the running time of Step 5 to Step 9 of Algorithm 30.7 is $O(\lambda^{-9}\zeta \log^{O(1)} n)$. By Lemma 30.4.3, $t = O(\lambda^{-3})$, and by Chernoff bound with probability $1 - \exp(-\Omega(\log^2 n))$ the size of Q is

at most $O(\lambda^{-4}\log^4 n)$. Hence, the running time of Step 12 to Step 24 of Algorithm 30.7 is $O(\lambda^{-7}g(n/\zeta, \lambda^4/2^{32})\log^4 n)$. □

By definition, we have the following basic fact about approximate ratio.

Fact 30.4.10. *Let f and f' be two functions mapping $(0,1)$ to $(0,1)$ such that $f(\lambda) \geq f'(\lambda)$ for any $\lambda \in (0,1)$. If there is a $f(\lambda)$-approximate LIS algorithm, then the algorithm is also $f'(\lambda)$-approximate.*

Now we present algorithm to approximate LIS using $\widetilde{O}(n^\kappa \operatorname{poly}(\lambda^{-1}))$ space by applying the pseudo-solution construction-evaluation framework recursively. In particular, we use the same algorithm on subarrays as an oracle and apply Lemma 30.4.9 recursively to approximate the entire sequence with slightly worse approximation ratio (compared with approximation ratio of the oracle).

Lemma 30.4.11. *Let κ be a constant of $(0,1)$ and $\lambda \in (0,1)$. Algorithm 30.8 approximates LIS with approximation ratio*

$$\frac{\lambda^{2\cdot 4^{(\lceil 1/\kappa \rceil - 1)}}}{256^{3\cdot 4^{(\lceil 1/\kappa \rceil - 1)}}}$$

and running time $O(n^\kappa \cdot \lambda^{-4^{O(1/\kappa)}} \log^{O(1)} n)$ and success probability $1 - \exp(-\Omega(\log^3 n))$.

Proof. We first prove the correctness of the algorithm by induction. Without loss of generality, we assume $1/\kappa$ is an integer.

For $i \in \{2, 3, \ldots, 1/\kappa\}$, denote

$$h_i(\lambda) = \frac{\lambda^{2\cdot 4^{(i-1)}-4}}{256^{2\cdot 4^{(i-1)}+3\cdot 4^{(i-2)}-7}}.$$

We show that if the length of the input sequence is $n^{i\cdot\kappa}$ then Algorithm 30.8 is $h_i(\lambda)$-approximate.

If the length of input sequence is $n^{2\kappa}$, then $h_2(\lambda) = \frac{\lambda^4}{2^{32}}$. By Corollary 30.4.2, Lemma 30.4.4, and Lemma 30.4.6, Algorithm 30.8 is $h_2(\lambda)$-approximate.

In the induction step, for an integer $2 \leq i < 1/\kappa$, we assume Algorithm 30.8 is $h_i(\lambda)$-approximate for input instance of length $n^{i\cdot\kappa}$. By Lemma 30.4.9, Algorithm 30.8 is $\left(h_i\left(\frac{\lambda^4}{2^{32}}\right)\frac{\lambda^4}{2^{33}}\right)$-approximate for input instance of length $n^{(i+1)\cdot\kappa}$. Since

$$\begin{aligned}
h_i\left(\frac{\lambda^4}{2^{32}}\right)\frac{\lambda^4}{2^{33}} =& \frac{\lambda^{4\cdot(2\cdot 4^{(i-1)}-4)}\cdot\lambda^4}{256^{4\cdot(2\cdot 4^{(i-1)}-4)}\cdot 256^{2\cdot 4^{(i-1)}+3\cdot 4^{(i-2)}-7}\cdot 2^{33}} \\
=& \frac{\lambda^{2\cdot 4^i-12}}{256^{2\cdot 4^i+2\cdot 4^{(i-1)}+3\cdot 4^{(i-2)}-18.875}} \\
>& \frac{\lambda^{2\cdot 4^i-4}}{256^{2\cdot 4^i+3\cdot 4^{(i-1)}-7}} \\
=& h_{i+1}(\lambda),
\end{aligned}$$

by Fact 30.4.10, Algorithm 30.8 is $h_{i+1}(\lambda)$-approximate for input instance of length $n^{(i+1)\cdot\kappa}$. Since

$$h_{1/\kappa}(\lambda) > \frac{\lambda^{2\cdot 4^{((1/\kappa)-1)}}}{256^{3\cdot 4^{((1/\kappa)-1)}}},$$

by Fact 30.4.10, Algorithm 30.8 is $\left(\frac{\lambda^{2\cdot 4^{((1/\kappa)-1)}}}{256^{3\cdot 4^{((1/\kappa)-1)}}}\right)$-approximate for input instance of length n.

By Corollary 30.4.2, Lemma 30.4.4, Lemma 30.4.6 and Lemma 30.4.9, we have the desired running time. The success probability is obtained by same corollaries/lemmas and union bound. □

Finally, by starting with $\lambda = 1$ and iteratively multiplying λ by a $1/(1+\epsilon)$ factor until a solution is found, we can approximate $\mathsf{lis}(A)$ within an approximation factor of $\lambda^{O(4^{1/\kappa})}$.

Theorem 30.4.12. *Let κ be a constant of $(0,1)$ and $\lambda \in (0,1)$. There exists a $\widetilde{O}(n^{\kappa} \cdot \lambda^{-4^{O(1/\kappa)}})$ time algorithm for lis with approximation factor $\lambda^{4^{O(1/\kappa)}}$ and success probability $1 - \exp(-\Omega(\log^3 n))$.*

30.5 Probability and Graph Tools

In this section, we restate probability and graph tools that we use throughout this paper. All these theorems are proven in previous work.

Theorem 30.5.1 (Chernoff Bounds)**.** *Let $X = \sum_{i=1}^{n} X_i$, where $X_i = 1$ with probability p_i and $X_i = 0$ with probability $1 - p_i$, and all X_i are independent. Let $\mu = \mathbb{E}[X] = \sum_{i=1}^{n} p_i$. Then*

1. $\Pr[X \geq (1+\delta)\mu] \leq \exp(-\delta^2\mu/3)$, $\forall \delta > 0$;

2. $\Pr[X \leq (1-\delta)\mu] \leq \exp(-\delta^2\mu/2)$, $\forall 0 < \delta < 1$.

Theorem 30.5.2 (Hoeffding bound)**.** *Let $X_1, \cdots, X_n$ denote n independent bounded variables in $[a_i, b_i]$. Let $X = \sum_{i=1}^{n} X_i$, then we have*

$$\Pr[|X - \mathbb{E}[X]| \geq t] \leq 2\exp\left(-\frac{2t^2}{\sum_{i=1}^{n}(b_i - a_i)^2}\right)$$

Theorem 30.5.3 (Blakley-Roy inequality, [BR65], see also Proposition 3.1 in [KSV13])**.** *Let G denote a graph that has n vertices and average degree d. The number of walks of length k in graph G is at least nd^k.*

Theorem 30.5.4 (Turán theorem for bipartite graphs, [KST54], see also [BBK13])**.** *For a graph G the Turán number $\mathrm{ex}(G, n)$ is the maximum number of edges that a graph on n vertices can have without containing a copy of G. For any $s \leq t$, $\mathrm{ex}(K_{s,t}, n) \leq \frac{1}{2}(t-1)^{1/s}n^{2-1/s} + o(n^{2-1/s})$*

Theorem 30.5.5 (Turán theorem for cliques [Tur41])**.** *Let G be any graph with n vertices, such that G is K_{r+1}-free. Then the number of edges in G is at most*

$$(1 - \frac{1}{r}) \cdot \frac{n^2}{2}.$$

Corollary 30.5.6 (of Theorem 30.5.5)**.** *Let G be any graph with n vertices, such that G has no independent set of size $r+1$. Then the number of edges in G is at least*

$$\binom{n}{2} - (1 - \frac{1}{r}) \cdot \frac{n^2}{2} = \frac{nr + n^2}{2r}.$$

30.6 LCS Step 0: Window-compatible Solutions

The goal of this section is to show window-compatible solutions provide very accurate estimates for LCS. Roughly speaking, in this section, we construct a set of windows for A and a set of windows for B and we show that it suffices to only compute the LCS of the windows. If that information is available, then we can estimate the LCS of two strings very accurately.

We use the same definition of window-compatible transformation as [BEG+18], but we will use a richer set of windows. Every window is essentially a substring of each string. Notice that windows may not have the same length. We use W_A to denote the set of windows constructed for A and W_B for the set of the windows that we construct for B.

Let us first state our definition of window-compatible solutions for LCS.

Definition 30.6.1 (Window-compatible LCS)**.** Let $S = \langle w_1, w_2, \cdots, w_\alpha \rangle$ and $S' = \langle w'_1, w'_2, \cdots w'_\alpha \rangle$ be two sequences of non-overlapping windows from W_1 and W_2, respectively.

- We call a *common substring* of (A, B) *window-compatible* with respect to S and S', if it is a union of k common substrings of $(w_1, w'_1), \ldots, (w_\alpha, w'_\alpha)$

We call a common substring window-compatible, if it is window-compatible with respect to at least a pair of sequence of non-overlapping windows from W_1 and W_2, respectively.

30.6.1 Construction of the Windows

This section presents a definition (see Definition 30.6.2) of a multiple layers window set for strings A and B. The construction is the same for both strings, therefore, we only

state it for A. As mentioned earlier, each window is a substring of the strings. For clarity, we use $A^{(j,l)}$ to denote a length-l substring of A which starts at index j and ends at index $j+l-1$. Our construction has multiple layers, each of which contains windows with equal lengths.

Given two length n strings A, B, and two parameters $1 \leq d \leq n$ and $\epsilon_0 \in (0,1)$, we define window set W_A to be a set of continuous subsequence of string A. The sizes of the windows depend on parameter d and the accuracy of our construction depends on parameter ϵ. Setting $d = n^x$ leads to a truly subquadratic time solution for any $0 < x < 1$, however, one can play with the value of d to optimize the running time.

Our construction has multiple layers, where in each layer the lengths of the windows are the same. Let $f = \Theta(\frac{1}{\epsilon_0} \log \frac{1}{\epsilon_0})$ and $f+1$ denote the number of different layers of windows in W_A.

For each $i \in \{0, 1, \cdots, f\}$, we define $d_i = d(1+\epsilon_0)^i$, $g_i = \epsilon_0 d_i$, and $t_i = n/d_i$. For each i, let W_A^i denote the set of all the windows in this layer. We consider window set W_A^i, in i-th layer, all the window have the same size which is d_i; g_i is the shift size, it means for each window (except the leftmost and rightmost one), if we shift either to the left or to the right, it becomes another window in W_A^i.

Formally speaking, we define W_A^i as follows

Definition 30.6.2. Let W_A be the set of windows we construct for string A. Then we have

$$W_A^i = \left\{ A^{(\text{left},\text{len})} \;\middle|\; \text{left} = x \cdot g_i + y \cdot d_i, \text{len} = d_i, \forall x \in \{0, 1, \cdots, d_i/g_i - 1\}, \forall y \in \{0, 1, \cdots, t_i - 1\} \right\}$$

where we use $A^{(\text{left},\text{len})}$ denotes a continuous subsequence of string A that starts at left and

has length len. Finally W_A is $\cup_{i\in\{0,1,\cdots,f\}} W_A^i$. Similarly, for each $i \in \{0, 1, \cdots, f\}$, we can obtain W_B^i and W_B.

Indeed, the running time of constructing the windows is truly subquadratic.

Lemma 30.6.1 (Generating (d, ϵ_0)-multiple layers window set)**.** *Given two strings A, B of n symbols over Σ symbols, and parameters $d \geq 1, \epsilon_0 \in (0, 1)$. Let set W_A and W_B be generated by procedure* GenerateMultipleLevels *(Algorithm 30.9). The algorithm takes $n \cdot \widetilde{O}(1/\epsilon_0^2)$ time and $|W_A| = |W_B| = O(n/(d\epsilon_0^2))$.*

Proof. Given two sets of blocks W_A and W_B, for each block $A^{(j_1,l_1)} \in W_A$, $A^{(j_1,l_1)}$ starts at j_1 and has length l_1. Similarly, for each block $B^{(j_2,l_2)} \in W_B$, $B^{(j_2,l_2)}$ starts at j_2 and has length l_2.

We can compute

$$\sum_{i=0}^{f} t_i = n \sum_{i=0}^{f} 1/d_i = (n/d) \sum_{i=0}^{f} (1+\epsilon_0)^{-i} \leq 2n/(d\epsilon_0). \tag{30.12}$$

Then we calculate the size of window set W_A,

$$\begin{aligned}
|W_A| &= \sum_{i=0}^{f} (\#\text{ shifts}) \cdot (\#\text{ windows per shift}) \\
&= \sum_{i=0}^{f} (d_i/g_i) \cdot t_i \\
&= \sum_{i=0}^{f} \frac{1}{\epsilon_0} \cdot t_i \\
&= \frac{1}{\epsilon_0} \sum_{i=0}^{f} t_i \\
&= 2n/(d\epsilon_0^2),
\end{aligned}$$

where the second step follows from definition of window set in each layer i, the third step follows from $d_i/g_i = 1/\epsilon_0$, and last step follows from Eq. (30.12).

The running time is

$$\sum_{i=0}^{f} (d_i/g_i) \cdot t_i \cdot d_i = n \sum_{i=0}^{f} (d_i/g_i) = n \sum_{i=0}^{f} 1/\epsilon_0 = \widetilde{O}(n/\epsilon_0^2).$$

□

Remark 30.6.1. In order to make sure the loss in the approximation is negligible, we use $\epsilon_0 = \epsilon\lambda$ and therefore we have

- The total number of windows is equal to $k = |W_A| + |W_B| = O((1/\lambda)^2 n/d)$.
- The maximum size of the windows is equal to $\mathsf{w}_{\mathsf{max}} = O(d/\lambda)$.
- The minimum size of the windows is equal to $\mathsf{w}_{\mathsf{min}} = d$.
- The ratio of the maximum window size over the minimum window size is bounded by $\mathsf{w}_{\mathsf{gap}} = \mathsf{w}_{\mathsf{max}}/\mathsf{w}_{\mathsf{min}} \leq O(1/\lambda)$.
- The number of different layers which is equal to the number of different window sizes is equal to $\mathsf{w}_{\mathsf{layers}} = O((1/\lambda)\log(1/\lambda))$.

30.6.2 Optimality of the Windows

We present a structural lemma for the windows in the following. This essentially, shows that the problem of computing LCS between the two strings reduces to the problem of computing the LCS's between the windows.

Lemma 30.6.2 (Improved Structural Lemma)**.** *For any two strings A, B of length n:*

- *If $\|\operatorname{LCS}(A, B)\| \geq \lambda$, there exists a window-compatible common substring of A, B of length at least $\lambda n - 8\epsilon_0 n$ via (d, ϵ_0)-Multiple layers window (Definition 30.6.2).*

Proof. We fix d and ϵ_0.

We pick the first block $\widetilde{A}_1$ to be length d, and consider the optimal matching. Suppose the optimal matching block for $\widetilde{A}_1$ is $\widetilde{B}_1$, there are three possibilities.

First, $\widetilde{B}_1$ has length between $[\epsilon_0 d, d/\epsilon_0]$. That's good and we keep this pair.

Second, $\widetilde{B}_1$ has length at least d/ϵ_0, then we throw out this pair. The characters we throw out from $2n$ is at least d/ϵ_0, and we decrease the entire LCS by at most d.

Third, the length of $\widetilde{B}_1$ has length at most $d\epsilon_0$, then we also throw out this pair. The characters we throw out from $2n$ is at least d, and we decrease the entire LCS by at most $\epsilon_0 d$.

The above procedure can be recursively applied until we found all the matching blocks. Let's say there are k matching pairs, $(\widetilde{A}_1, \widetilde{B}_1), (\widetilde{A}_2, \widetilde{B}_2), \cdots (\widetilde{A}_\alpha, \widetilde{B}_\alpha)$. Note that $\widetilde{A}_1, \widetilde{A}_2, \cdots, \widetilde{A}_\alpha$ is a partition of A. Similarly, $\widetilde{B}_1, \widetilde{B}_2, \cdots, \widetilde{B}_\alpha$ is a partition of B. Let z_i denote the variable such that $z_i = 0$ indicates we throw out the pair, and $z_i = 1$ otherwise. Since whenever we throw out a pair, the amount of LCS got decreased is always at most ϵ_0 fraction of the characters we throw out. Since there are $2n$ characters in total. Then we LCS we decrease is at most $2\epsilon_0 n$, i.e.,

$$\sum_{i \in Z} \operatorname{LCS}(\widetilde{A}_i, \widetilde{B}_i) \geq \operatorname{LCS}(A, B) - 2\epsilon_0 n, \tag{30.13}$$

where $Z = \{i \mid z_i = 1, i \in [k]\}$.

We consider a fixed $i \in Z$. For each pair $\widetilde{A}_i, \widetilde{B}_i$, let p and q denote the integer that

$$|\widetilde{A}_i| \in ((1+\epsilon_0)^p, (1+\epsilon_0)^{p+1}], \text{ and } |\widetilde{B}_i| \in ((1+\epsilon_0)^q, (1+\epsilon_0)^{q+1}].$$

Let $\widehat{A}_i$ and $\widehat{B}_i$ denote the two pairs from W_A and W_B that has the largest LCS and satisfies the following

$$|\widehat{A}_i| = (1+\epsilon_0)^{p-1},\ \widehat{A}_i \subseteq \widetilde{A}_i,\ |\widehat{B}_i| = (1+\epsilon_0)^{q-1},\ \widehat{B}_i \subseteq \widetilde{B}_i.$$

Since $|\widetilde{A}_i| - |\widehat{A}_i| \geq (1+\epsilon_0)^p - (1+\epsilon_0)^{p-1} = \epsilon_0(1+\epsilon_0)^{p-1}$. Recall the construction of W_A, we places a length $(1+\epsilon_0)^{p-1}$ block for every $\epsilon_0(1+\epsilon_0)^{p-1}$ shifts. Thus, there must exist such block $\widehat{A}_i$ from W_A and $\widehat{A}_i \subseteq \widetilde{A}_i$. Similarly, the same argument holds for $\widehat{B}_i$.

According to the definition of W_A and W_B, we know that

$$\begin{aligned}
|\widetilde{A}_i \backslash \widehat{A}_i| &\leq (1+\epsilon_0)^{p+1} - (1+\epsilon_0)^{p-1} \\
&\leq (1+\epsilon_0)^{p-1} 3\epsilon_0 \\
&\leq 3\epsilon_0 |\widetilde{A}_i|.
\end{aligned}$$

Similarly, $|\widetilde{B}_i \backslash \widehat{B}_i| \leq 3\epsilon_0 |\widetilde{B}_i|$. The characters we ignore is at most $3\epsilon_0 \cdot \max(|\widetilde{A}_i|, |\widetilde{B}_i|)$. Let $\tau_i = 3\epsilon_0 \cdot \max(|\widetilde{A}_i|, |\widetilde{B}_i|)$, then $\tau_i \leq 3\epsilon_0(|\widetilde{A}_i| + |\widetilde{B}_i|)$.

If we consider $\sum_{i \in Z} |\operatorname{LCS}(\widehat{A}_i, \widehat{B}_i)|$,

$$\begin{aligned}
\sum_{i \in Z} |\operatorname{LCS}(\widehat{A}_i, \widehat{B}_i)| &\geq \sum_{i \in Z} (|\operatorname{LCS}(\widetilde{A}_i, \widetilde{B}_i)| - \tau_i) \\
&\geq \sum_{i \in Z} |\operatorname{LCS}(\widetilde{A}_i, \widetilde{B}_i)| - 6\epsilon_0 n \\
&\geq |\operatorname{LCS}(A, B)| - 8\epsilon_0 n,
\end{aligned}$$

where the second step follows from $\sum_{i \in Z} \tau_i \leq 6\epsilon_0 n$, the third step follows from Eq. (30.13).

□

30.6.3 Dynamic Programming for Block-based LCS

We showed in Section 30.6.2 that there always exists a window-compatible solution whose size is very close to that of the optimal solution. In this section, we show how one can find such a solution via dynamic programming. Therefore, we assume that a matrix M is provided as input that stores the LCS of every pair of windows.

Lemma 30.6.3 (Dynamic Programming for Longest Common Sequence)**.** *Given two strings A, B of length n. Let $\epsilon \in (0,1)$, $d \geq 1$. Let W_A, W_B be generated by procedure* CreateMultipleLevels *$(A, B, n, d, \Theta(\epsilon\lambda))$ (Algorithm 30.9). Let table M be generated by procedure* NonMetricEstimation *(W_A, W_B).*

Then there is an algorithm (Algorithm 30.10) that runs in $(n/d)^2 \cdot \mathrm{poly}(\lambda, 1/\epsilon)$ time to output a monotone matching between two sets W_A and W_B which gives a string C such that

$$|C| \geq (1 - o(1)) \cdot (1 - \epsilon) \cdot |\operatorname{LCS}(A, B)|.$$

Proof. We choose $\epsilon_0 = \Theta(\epsilon/\lambda)$.

Using Lemma 30.6.2, we know there exits a batch of window-pairs $(\widehat{A}_{i_1}, \widehat{B}_{j_1}), (\widehat{A}_{i_2}, \widehat{B}_{j_2}), \cdots$ from W_A, W_B such that

$$\sum_l |\operatorname{LCS}(\widehat{A}_{i_l}, \widehat{B}_{i_l})| \geq |\operatorname{LCS}(A, B)| - 8\epsilon_0 n. \tag{30.14}$$

By assumption on M, we know that M gives a good estimation for every pair in W_A, W_B :

$$\delta_{i_l,j_l} d_{i_l,j_l} \geq M_{i_l,j_l} \geq (1 - o(1))\delta_{i_l,j_l}^3 \cdot d_{i_l,j_l}.$$

where $d_{i_l,j_l} = \max\{|\widehat{A}_{i_l}|, |\widehat{B}_{j_l}|\}$, $\delta_{i_l,j_l} = |\operatorname{LCS}(\widehat{A}_{i_l}, \widehat{B}_{j_l})|/d_{i_l,j_l}$. It implies that

$$|\operatorname{LCS}(\widehat{A}_{i_l}, \widehat{B}_{j_l})| \geq M_{i_l,j_l} \geq (1 - o(1))\delta_{i_l,j_l}^2 |\operatorname{LCS}(\widehat{A}_{i_l}, \widehat{B}_{j_l})|.$$

Now, if we consider this cost $\sum_l M_{i_l,j_l}$,

$$\begin{aligned}
\sum_l M_{i_l,j_l} &\geq (1 - o(1)) \cdot \sum_l \delta_{i_l,j_l}^2 |\operatorname{LCS}(\widehat{A}_{i_l}, \widehat{B}_{j_l})| \\
&\geq (1 - o(1)) \cdot \lambda^{-2} \sum_l |\operatorname{LCS}(\widehat{A}_{i_l}, \widehat{B}_{j_l})| \\
&\geq (1 - o(1)) \cdot (|\operatorname{LCS}(A, B)| - 8\epsilon_0 n) \\
&= (1 - o(1)) \cdot (|\operatorname{LCS}(A, B)| - \epsilon n/\lambda) \\
&\geq (1 - o(1))(1 - \epsilon)|\operatorname{LCS}(A, B)|
\end{aligned}$$

where the second step follows from $\delta_{i_l,j_l} \geq \lambda^{-1}$, the third step follows from Eq. (30.14), the fourth step follows from $\epsilon_0 = \Theta(\epsilon/\lambda)$, the last step follows from $\operatorname{LCS}(A, B) \geq \lambda n$.

Next, we prove the correctness of running time. Using Lemma 30.6.1, we know that $|W_A| = |W_B| = O(n/(d\epsilon_0^2))$. Then $|S_1| = |S_2| = O(n/(d\epsilon_0^2))$. The running time of sorting takes $O(|S_1| \log |S_1|)$.

The dominated part of the running time is three for-loops. For the third one, if we consider a fixed i_1 and i_2, there are $O((\frac{1}{\epsilon_0} \log(1/\epsilon_0))^2)$ pairs could have right index at i_1, i_2. Thus the running time of dynamic programming takes

$$(n/(d\epsilon_0^2)) \cdot (n/(d\epsilon_0^2)) \cdot O\left(((1/\epsilon_0) \log(1/\epsilon_0))^2\right) = (n/d)^2 \cdot \widetilde{O}(1/\epsilon_0^6).$$

For the correctness of dynamic programming, it follows from three cases cover all the possibilities. For each i_1, i_2, we define $C[i_1][i_2]$ to be optimal solution that only uses blocks from W_A with ending index no later than i_1 and blocks from W_B with ending index no later than i_2. Suppose we are at i_1, i_2. There are three cases, the first case is, the last pair of the optimal solution for $C[i_1][i_2]$ is ending at i_1, i_2. The second case is, the last pair of the optimal solution for $C[i_1][i_2]$ is ending at $\text{prev}(i_1), i_2$. The third case is, the last pair of the optimal solution for $C[i_1][i_2]$ is ending at $i_1, \text{prev}(i_2)$. □

Remark 30.6.2. If in the above algorithm, M instead of the correct LCS's has a $(1-\epsilon)\lambda^2$ or $O(\lambda^3)$ approximation of the solutions, then the overall approximation factor of the algorithm would be $(1-\epsilon)\lambda^2$ or $O(\lambda^3)$, respectively.

30.7 LCS Step 2: Discovering the Sparse Graph

The arguments of this section follow from the work of [CDG+18]. However, for the sake of completeness, we restate the overall ideas here. This section is dedicated to presenting a method NonMetricEstimation(W_A, W_B) to be used for estimating the longest common subsequence. The output of NonMetricEstimation(W_A, W_B) is a matrix $M : [|W_A|] \times [|W_B|] \to \mathbb{Z}^+$ where $M[i][j]$ estimates the LCS of windows $w_i \in W_A$ and $w_j \in W_B$. We allow for errors in a few elements of M but we prove at the end of this section that with high probability, the error does not affect the solution significantly.

Similar to Section 30.3, we define $W = W_A \cup W_B$ and denote the windows inside W by $w_1, w_2, \ldots, w_k$. For a threshold λ the edges of graph G_λ reflect pairs of windows (w_i, w_j) such that $\|\mathsf{lcs}(w_i, w_j)\| \geq \lambda$. In order to estimate M, we need to discover the edges of G_λ for $\lambda \in \{\epsilon\lambda_0, \epsilon\lambda_0(1+\epsilon), \epsilon\lambda_0(1+\epsilon)^2, \ldots, 1\}$ and build M based on that. In other words, for each w_i and w_j, we find the largest λ such that G_λ contains edge (i, j) and set $M[i][j] = \lambda$.

Let us for simplicity fix the approximation factor to be $(1-\epsilon)\lambda^2$. As mentioned earlier, we represent the edges of G_λ by a matrix $\mathsf{O} : [k] \times [k] \to \{0, 1\}$ where $\mathsf{O}[i][j] = 1$ implies that edge (i, j) exists in G_λ. The $(1-\epsilon)\lambda^2$ sparsification gives us an output $\widehat{\mathsf{O}}_{(1-\epsilon)\lambda^2}$ where except for a few pairs the rest of the edges of G_λ are detected within an approximation factor $(1-\epsilon)\lambda^2$. If all the edges were detected with this algorithm, this would provide us with a matrix M such that $(1-\epsilon)\|\mathsf{lcs}(w_i, w_j)\|^3 \leq M[i][j] \leq \|\mathsf{lcs}(w_i, w_j)\|$ holds if $\|\mathsf{lcs}(w_i, w_j)\| \geq \epsilon\lambda$. If such a matrix is given to Algorithm 30.11, it would result in a solution with approximation factor $(1-\epsilon)\lambda^2$.

However, not all the edges of G_λ are detected in Step 1. Therefore, if we use the

discovered edges to construct a matrix $\widehat{M}$ for the LCS of the windows, $\widehat{M}$ may not meet the guarantee of M for at most $\widetilde{O}(k^{2-\Omega(1)})$ pairs. In what follows, we present an algorithm that resolves this issue.

To simplify the notation, define a graph NG in which every window is a vertex and there is an edge between pair (i, j) if the LCS of windows w_i and w_j is not estimated correctly in the sparsification step. Our algorithm is not aware of the edges of NG but since the number of remaining edges is truly subquadratic, we have $|E(\mathsf{NG})| = \widetilde{O}(k^{2-\Omega(1)})$.

Let us assume that NG contains at most $\widetilde{O}(k^{2-\eta})$ edges. This implies that the average degrees of the vertices of NG is upper bounded by $\widetilde{O}(k^{1-\eta})$. Although we are not given the edges of NG explicitly, we can verify for each pair (i, j) whether there is an edge between them in NG by computing the LCS of w_i and w_j and comparing it to the value that has been obtained in $\widehat{M}$.

Let $\mathsf{w_{max}} = \max_i |w_i|$ be the maximum length of the windows. Define two windows (w_i, w_j) of W_A or two windows (w_i, w_j) of W_B to be *nearby* if their distance is bounded by at most $\mathsf{w_{max}}k^{\eta/2}$. That is w_i covers an index a and w_j covers an index b such that $|a-b| \leq \mathsf{w_{max}}k^{\eta/2}$. We subsample a set S of W_A with a rate of $\epsilon^{-1}k^{-\eta/2}$. That is, each window of W_A appears in S independently, with probability $\epsilon^{-1}k^{-\eta/2}$. Next, for each window $w_i \in S$, we discover all of its incident edges in NG by computing the LCS of w_i and every window $w_j \in W_B$. Every time we discover one edge of NG we add this edge to NG'. Notice that NG' contains a subset of edges of NG however, the edges of NG' are known to the algorithm and are available. Finally, we update M by computing the LCS of pairs of windows (w_i, w_j) such that there exist windows $w_{i'} \in W_A$ and $W_{j'} \in W_B$ such that 1) w_i and $w_{i'}$ are nearby, 2) w_j and $w_{j'}$ are nearby, and 3) vertices i and j are neighbors in NG'. This is shown in Algorithm

30.12.

It follows from the analysis of [CDG+18] that if we use M as an estimation of the windows, then Algorithm 30.11 finds the desired solution even though M does not correctly estimate all the values.

Lemma 30.7.1 ([CDG+18])**.** *Let $\widehat{M}$ be the output of the sparsification algorithm and M be constructed by Algorithm 30.12 from $\widehat{M}$. If NG has at most $\widetilde{O}(k^{2-\eta})$ edges, then the runtime of Algorithm 30.12 is bounded by $\widetilde{O}\left(k^{2-\eta/2}\mathsf{w}_{\mathsf{max}}^2/\lambda^4\right)$. Also using M as an estimate for the LCS of the windows hurts the solution by a factor of at most $1+\epsilon$.*

Proof. The proof follow from the arguments of [CDG+18]. Here we just mention the intuition and the overall ideas.

Runtime: Since every window of W_A is sampled with probability $k^{-\eta/2}$ the size of S is bounded by $O(k^{1-\eta/2})$ with high probability. In order to discover the incident edges of set S in NG, we compute the LCS of every element in S and all other windows. Therefore, the total running time at this point is $\widetilde{O}(k^{2-\eta/2}\mathsf{w}_{\mathsf{max}}^2)$.

NG contains at most $k^{2-\eta}$ edges and therefore the expected number of edges that we put in NG' is $\widetilde{O}(k^{2-3\eta/2})$. For every edge that we discover in NG' we compute the LCS of $\widetilde{O}((2k^{\eta/2}/\lambda^2)^2) = \widetilde{O}(k^{\eta}/\lambda^4)$ pairs of windows. Since computing the LCS of every pair of windows takes time $O(\mathsf{w}_{\mathsf{max}}^2)$ then the expected overall running time is

$$\widetilde{O}\left(k^{2-\eta/2}\mathsf{w}_{\mathsf{max}}^2 + k^{2-3\eta/2}\cdot k^{\eta}/\lambda^4 \cdot \mathsf{w}_{\mathsf{max}}^2\right) = \widetilde{O}\left(k^{2-\eta/2}\mathsf{w}_{\mathsf{max}}^2/\lambda^4\right).$$

Notice that $k^2\mathsf{w}_{\mathsf{max}}^2 \simeq n^2$ and thus the above running time is truly subquadratic.

Sketch of the correctness: We refer the reader to [CDG+18] for a complete proof. The high-level idea is based on the following: Originally, the window size is defined based on a parameter d. Now, suppose we define some larger windows by setting $d' = dk^{\eta/2}$. Let W'_A and W'_B be the two sets of windows we obtain by using d' as the base parameter for constructing windows. Our window-compatibility lemma holds for any window size and therefore there is also a desired window-compatible solution when using W'_A and W'_B.

The intuition behind the proof of [CDG+18] is the following: Suppose we know a window w'_x of W'_A is mapped to a window w'_y of W'_B in the optimal window-compatible solution for W'_A and W'_B. There are at least $k^{\eta/2}$ smaller windows of W_A that lie inside w'_x and similarly w'_y contains at least $k^{\eta/2}$ windows of W_B. Let the two sets be X and Y. There is indeed a window-compatible solution for the windows of X and Y. Now, the question that we need to answer, is how well such a window-compatible solution between w'_x and w'_y is approximated via $\widehat{M}$. If fewer than $\epsilon|X|$ pairs of this mapping are among the edges of NG then we can be sure that the approximation factor does not hurt by much. On the other hand, if more than $\epsilon|X|$ pairs are among the edges of NG then we can imply that one of such edges is discovered via Algorithm 30.12 in NG' and thus M would have the correct value of LCS between all windows of X and all windows of Y. Thus, when we run Algorithm 30.10 on M, the output has an error of at most ϵ even though M does not have the correct estimates for all pairs. □

Algorithm 30.3 lcs-cmp data structure

data structure lcs-cmp

members
  $X_{a,1}, \cdots, X_{a,s}$
  $Y_{a,1}, \cdots, Y_{a,s}$
  $\widetilde{\lambda} \in (0,1)$
end members

procedure INITIAL($w_a, \{w_i\}_{i\in[s]}, \widetilde{\lambda}$) ▷ Lemma 30.3.11
  for $i \in [s]$ **do**
    compute $\mathsf{opt}_{a,i}$, a LCS between w_a and w_i
    let $X_{a,i}$ be the set of indices of all the element of $\mathsf{opt}_{a,i}$ with respect to w_a
  end for
  for $i \in [s]$ **do**
    let $Y_{a,i}$ be an empty set
    while true **do**
      compute $\mathsf{opt}'_{a,i}$, a LCS between w_a removing the elements with index in $Y_{a,i}$ and w_i
      if the length of $\mathsf{opt}'_{a,i}$ is less than $\widetilde{\lambda}|w_a|/2$ **then**
        break
      else
        put the indices of all the elements of $\mathsf{opt}'_{a,i}$ with respect to w_a into $Y_{a,i}$
      end if
    end while
  end for
  return $\{X_{a,i}\}_{i\in[s]}$, $\{Y_{a,i}\}_{i\in[s]}$
end procedure

procedure QUERY(w_i, w_j) ▷ Lemma 30.3.12
  if $|X_{a,i} \cap Y_{a,j}| > \widetilde{\lambda}\sqrt{|w_i| \cdot |w_j|}/4$ **then**
    return accept
  else
    return reject
  end if
end procedure

end data structure

Algorithm 30.4 Constructing the candidate domains

procedure ConstructCandidateDomains(sa_i) ▷ Lemma 30.4.1
▷ Given random access to a subarray sa_i
  $k \leftarrow 20\log(1/\delta)/(\lambda\epsilon^2)$
  Sample k elements from sa_i, and denote the sampled elements by $a_{j_1}, a_{j_2}, \ldots, a_{j_k}$
  for α in $[k]$ **do**
    for β in $[k]$ **do**
      If $a_{j_\alpha} \leq a_{j_\beta}$, then construct a candidate domain $[a_{j_\alpha}, a_{j_\beta}]$
    end for
  end for
  return all the constructed candidate domains
end procedure

Algorithm 30.5 Constructing the pseudo solutions

procedure ConstructPseudoSolutions($\mathsf{cdi}_1, \ldots, \mathsf{cdi}_{\sqrt{n}}$) ▷ Lemma 30.4.3,30.4.4,30.4.5
▷ $\{\mathsf{cdi}_i\}_{i\in[\sqrt{n}]}$ is $\sqrt{n}$ sets of candidate domain intervals
  pseudo-solutions $\leftarrow \emptyset$
  while true do
    assg $\leftarrow$ largest assigment of candidate domain intervals to subarrays which is monotone
    if assg contains less than $\epsilon\sqrt{n}\lambda$ non-empty candidate domain intervals **then**
      break
    else
      Add assg to pseudo-solutions
      for $i \leftarrow 1$ to $\sqrt{n}$ **do**
        if assg contains a candidate domain interval for subarray sa_i **then**
          remove the corresponding candidate domain interval from cdi_i
        end if
      end for
    end if
  end while
  return pseudo-solutions $\mathsf{ps}_1, \mathsf{ps}_2, \ldots, \mathsf{ps}_t$
end procedure

Algorithm 30.6 Evaluate the pseudo solutions

procedure EVALUATEPSEUDOSOLUTIONS($\mathsf{ps}_1, \ldots, \mathsf{ps}_t$) ▷ Lemma 30.4.6
▷ $\{\mathsf{ps}_i\}_{i\in[t]}$ is a set of pseudo solutions
    $p \leftarrow \frac{1000t\log^4 n}{\epsilon^4\lambda\sqrt{n}}$
    Randomly sample each $i \in [\sqrt{n}]$ with probability p, and put all the samples in a set W
    for each ps_j **do**
        $\widetilde{\mathsf{q}}(\mathsf{ps}_j) \leftarrow 0$
        for each interval $[\ell_i, r_i]$ in ps_j **do**
            if $i \in W$ **then**
                $\widetilde{\mathsf{q}}(\mathsf{ps}_j) \leftarrow \widetilde{\mathsf{q}}(\mathsf{ps}_j) + \mathsf{lis}^{[\ell_i, r_i]}(\mathsf{sa}_i)/p$
            end if
        end for
    end for
    return largest $\widetilde{\mathsf{q}}(\mathsf{ps}_j)$ for all $j \in [t]$
end procedure

Algorithm 30.7 Recursive Estimate LIS with Oracle

procedure RecursiveEstimationWithOracle(Oracle, A, λ, ℓ, r) ▷ Lemma 30.4.9
  ▷ input: sequence A, parameter λ, domain interval $[\ell, r]$
  ▷ assume $\mathsf{sa}_1, \mathsf{sa}_2, \ldots, \mathsf{sa}_{n^\kappa}$ are subarrays of A
  ▷ subroutine Oracle approximate LIS for subarrays with approximation factor $f(\lambda)$
  for $i \in [\zeta]$ **do**
    $\mathsf{cdi}_i \leftarrow$ ConstructCandidateDomains(sa_i)
    discard all the intervals which are not in $[\ell, r]$ from cdi_i
  end for
  $\{\mathsf{ps}_1, \ldots, \mathsf{ps}_t\} \leftarrow$ ConstructPseudoSolutions($\mathsf{cdi}_1, \ldots, \mathsf{cdi}_\zeta$)
  $\lambda_0 \leftarrow \left(\frac{\lambda}{2^8}\right)^4$
  $p \leftarrow \frac{20 \log^4 n}{\lambda_0 \zeta}$
  randomly sample each $i \in [\zeta]$ with probability p, and put all the samples in a set Q
  for $j \in [t]$ **do**
    $\mathsf{c} \leftarrow 0$
    for $i \in W$ **do**
      if $\exists [\ell_i, r_i] \in \mathsf{ps}_j$ and Oracle($\mathsf{sa}_i, \lambda_0, \ell_i, r_i$) accepts **then**
        $\mathsf{c} \leftarrow \mathsf{c} + 1$
      end if
    end for
    if $\mathsf{c} \geq 3\lambda_0 p \zeta / 4$ **then**
      return accept
    end if
  end for
  return reject
end procedure

Algorithm 30.8 Recursive Estimate LIS

procedure RecursiveLIS(A, λ, ℓ, r) ▷ Lemma 30.4.11
    ▷ input: sequence A, parameter λ, domain interval $[\ell, r]$
    ▷ assume $\mathsf{sa}_1, \mathsf{sa}_2, \ldots, \mathsf{sa}_{n^\kappa}$ are subarrays of A
    if the length of A is greater than $n^{2\kappa}$ **then**
        return RecursiveLISWithOracle(RecursiveLIS, A, λ, ℓ, r) with $\zeta = n^\kappa$
    else
        for $i \in [n^\kappa]$ **do**
            $\mathsf{cdi}_i \leftarrow$ ConstructCandidateDomains(sa_i)
            discard all the intervals which are not in $[\ell, r]$ from cdi_i
        end for
        $\{\mathsf{ps}_1, \ldots, \mathsf{ps}_t\} \leftarrow$ ConstructPseudoSolutions($\mathsf{cdi}_1, \ldots, \mathsf{cdi}_{n^\kappa}$)
        if EvaluatePseudoSolutions($\mathsf{ps}_1, \ldots, \mathsf{ps}_t$) $\geq \lambda|A|$ **then**
            return accept
        else
            return reject
        end if
    end if
end procedure

Algorithm 30.9 An algorithm that is able to generate multiple layers

procedure GENERATEMULTIPLELEVELS(A, B, n, d, ϵ_0) ▷ Lemma 30.6.1
  $f \leftarrow \Theta(\frac{1}{\epsilon_0} \log \frac{1}{\epsilon_0})$ ▷ $d \cdot (1+\epsilon_0)^f = d/\epsilon_0$
  $W_A \leftarrow \emptyset$, $W_B \leftarrow \emptyset$
  for $i = 0 \to f$ **do**
    $W_A^i \leftarrow \emptyset$, $W_B^i \leftarrow \emptyset$
    $d_i \leftarrow d(1+\epsilon_0)^i$, $t_i \leftarrow n/d_i$, $g_i \leftarrow \epsilon_0 d_i$ ▷ d_i is the length, g_i is the shift
    for $x = 0 \to d_i/g_i - 1$ **do**
      for $y = 0 \to t_i - 1$ **do**
        left $\leftarrow x \cdot g_i + y \cdot d_i$
        len $\leftarrow d_i$
        $W_A^i \leftarrow W_A^i \cup A^{(\text{left,len})}$, $W_B^i \leftarrow W_B^i \cup B^{(\text{left,len})}$
      end for
    end for
    $W_A \leftarrow W_A \cup W_A^i$, $W_B \leftarrow W_B \cup W_B^i$
  end for
  return W_A, W_B, f
end procedure

Algorithm 30.10 Dynamic programming algorithm for block-based LCS problem

procedure DP-LCS(W_A, W_B, M) ▷ Lemma 30.6.3
    Note that M is table that for each A_{k_1} and B_{k_2}, $M(A_{k_1}, B_{k_2})$ gives a cost
    Note that W_A and W_B are sets produce by Algorithm 30.9
    Let S_1 denote the sorted list of the set(without duplicate) of right indices of each A_{k_1}
    Let S_2 denote the sorted list of the set(without duplicate) of right indices of each B_{k_2}
    for $i_1 \in S_1$ **do**
        for $i_2 \in S_2$ **do**
            $C[i_1][i_2] \leftarrow 0$
            $x_1 \leftarrow 0$
            for A_{k_1}, B_{k_2} have right index i_1, i_2 **do** ▷ Case 1
                $\text{left}_1 \leftarrow$ left index of A_{k_1}
                $\text{left}_2 \leftarrow$ left index of B_{k_2}
                $\text{tmp} \leftarrow C[\text{left}_1][\text{left}_2] + M(A_{k_1}, B_{k_2})$
                if $\text{tmp} > x_1$ **then**
                    $x_1 \leftarrow \text{tmp}$
                end if
            end for
            Let $\text{prev}(i_1)$ denote the first index $\in S_1$ that is early than i_1
            Let $\text{prev}(i_2)$ denote the first index $\in S_2$ that is early than i_2
            $x_2 \leftarrow C[\text{prev}(i_1)][i_2]$ ▷ Case 2
            $x_3 \leftarrow C[i_1][\text{prev}(i_2)]$ ▷ Case 3
            $C[i_1][i_2] \leftarrow \max(x_1, x_2, x_3)$
        end for
    end for
end procedure

Algorithm 30.11

procedure ApproxLCS(A, B, n, λ)
  $d \leftarrow n^{0.2}$, $\beta \leftarrow \epsilon \leftarrow \Theta(1)$, $\epsilon_0 \leftarrow \Theta(\epsilon/\lambda)$
  $W_A, W_B, L \leftarrow$ CreateMultipleLevels(A, B, n, d, ϵ_0) ▷ Lemma 30.6.1, Algorithm 30.9
  $M \leftarrow$ NonMetricEstimation(W_A, W_B)
  $C \leftarrow$ DPLCS(W_A, W_B, M) ▷ Lemma 30.6.3, Algorithm 30.10
  return C
end procedure
procedure NonMetricEstimation(W_A, W_B) ▷ Section 30.3 and Section 30.7
  for $i = 1 \to L$ **do**
    for $j = 1 \to L$ **do**
      Make strings in W_A^i and W_B^j have the same length by appending $\perp$ to the short one
      W_A^i has t_i blocks and each of it has length d_i
      W_B^j has t_j blocks and each of it has length d_j
      $d \leftarrow \max(d_i, d_j)$, $t \leftarrow \max(t_i, t_j)$
      Create $S^{(i,j)}$ by Step 1 (Section 30.3) and Step 2 (Section 30.7)
    end for
  end for
  Form cost table M by putting all $S^{(i,j)}$ together
  return M
end procedure

Algorithm 30.12 Constructing M based on $\widehat{M}$

procedure NonMetricEstimation(W_A, W_B) ▷ Lemma 30.7.1
  $S \leftarrow \emptyset$
  for $w_i \in W_A$ **do**
    Add w_i to S with probability $\epsilon^{-1}k^{-\eta/2}$
  end for
  for $w_i \in S$ **do**
    for $w_j \in W_B$ **do**
      Compute $\mathsf{lcs}(w_i, w_j)$ and put an edge in NG' if $\widehat{M}[i][j]$ does not correctly estimate $\mathsf{lcs}(w_i, w_j)$.
    end for
  end for
  $M \leftarrow \widehat{M}$
  for $w_i \in W_a$ **do**
    for $w_j \in W_B$ **do**
      if there exist nearby windows $w_{i'}, w_{j'}$ such that $(i', j') \in E(\mathsf{NG}')$ **then**
        $M[i][j] \leftarrow \mathsf{lcs}(w_i, w_j)$
      end if
    end for
  end for
  return M
end procedure

Chapter 31

Fourier Transform

31.1 One dimensional Fourier transform

The result in this section is based on the unpublished manuscript by Jerry Li, Ruoqi Shen and Zhao Song [LSS19].

Lemma 31.1.1 (Claim 5.2 in [CKPS16])**.** *For any k, there exists $m = O(k^2 \log k)$ such that for any k-Fourier-sparse signal $x(t)$, any $t_0 \geq 0$ and $\tau > 0$, there always exists $C_1, ..., C_m \in \mathbb{C}$ such that $|C_j| \leq 11$ for all $j \in [m]$ and*

$$x(t_0) = \sum_{j \in [m]} C_j \cdot x(t_0 + j \cdot \tau).$$

Lemma 31.1.2 (Bounding absolute by discrete sum on grids)**.** *For any k-Fourier-sparse signal $x(t) = \sum_{j=1}^k v_j e^{2\pi \mathbf{i} f_j t}$, we have*

$$\max_{t \in [0,T]} |x(t)|^2 \leq O(k^6 \log^3 k) \cdot \frac{1}{T} \sum_{t=0}^{T} |x(t)|^2.$$

Proof. Without loss of generality, we fix $T = 1$ and assume $|x(0)|^2 = \max_{t \in [0,T]} |x(t)|^2$. If $t^* = \arg\max_{t \in [0,T]} |x(t)|^2$ is not 0 or $T = 1$, we can rescale the two intervals $[0, t^*]$ and $[t^*, T]$ to $[0, 1]$. By Lemma 31.1.1, we choose $t_0 = 0$ such that $\forall \tau > 0$, there exists $C_1, ..., C_m \in \mathbb{C}$ and

$$x(0) = \sum_{j \in [m]} C_j \cdot x(j \cdot \tau).$$

By the Cauchy-Schwarz inequality, it implies that for any τ,

$$\begin{aligned}|x(0)|^2 &\le m \sum_{j\in[m]} |C_j|^2 |x(j\cdot\tau)|^2 \\ &\lesssim m \sum_{j\in[m]} |x(j\cdot\tau)|^2.\end{aligned}$$

Then,

$$\begin{aligned}|x(0)|^2 &= \frac{1}{\lfloor T/m \rfloor} \sum_{\tau=0}^{\lfloor T/m \rfloor} |x(0)|^2 \\ &\lesssim \frac{m}{T} \sum_{\tau=0}^{\lfloor T/m \rfloor} \left(m \sum_{j\in[m]} |x(j\cdot\tau)|^2 \right) \\ &= \frac{m^2}{T} \sum_{j\in[m]} \sum_{\tau=0}^{\lfloor T/m \rfloor} |x(j\cdot\tau)|^2 \\ &\le \frac{m^3}{T} \sum_{t=0}^{T} |x(t)|^2.\end{aligned}$$

From $m = O(k^2 \log k)$, we obtain $|x(0)|^2 = O(k^6 \log^3 k \|x\|_{[T]})$. □

31.2 High dimensional Fourier transform

The result in this section is based on the unpublished result by Zhao Song [Son17].

Lemma 31.2.1 (d dimensional version of Lemma 5.1 in [CKPS16])**.** *For any d-dimensional k-Fourier-sparse signal $x(t) : \mathbb{R}^d \to \mathbb{C}$ and any duration T, we have*

$$\max_{t\in[0,T]^d} |x(t)|^2 \le k^{O(d)} \|x\|_T^2,$$

where $\|x\|_T^2 = \frac{1}{T^d} \int_{[0,T]^d} |x(t)|^2 \mathrm{d}t$.

Proof. Without loss of generality, we fix $T = 1$. Then $\|x\|_T^2 = \int_{[0,1]^d} |x(t)| \mathrm{d}t$. Because $\|x\|_T^2$ is the average over the interval $[0, T]^d$, if $t^* = \arg\max_{t \in [0,T]^d} |x(t)|^2$ is not 0^d or $T = 1$, we can rescale the two intervals $[0^d, t^*]$ and $[t^*, T^d]$ to $[0, 1]$ and prove the desired property separately. Hence we assume that $|x(0)|^2 = \max_{t \in [0,T]} |x(t)|^2$ in the proof.

The proof of the following Claim can be found in Section 31.3 (is similarly to the proof of Claim 5.2 in [CKPS16])

Claim 31.2.2 (d-variables version of Claim 5.2 in [CKPS16])**.** *For any k and d, there exists $m = O(k^2 \log k)$ such that for any d-dimensional k-Fourier-sparse signal $x(t)$, any $t_0 \in \mathbb{R}^d_{\geq 0}$ and $\tau \in \mathbb{R}^d_{>0}$, there always exist $C_1, C_2, \cdots, C_m \in \mathbb{C}$ such that the following properties hold,*

$$\begin{aligned} &\text{Property I} \quad |C_j| \leq 11 \quad \textit{for all } j \in [m], \\ &\text{Property II} \quad x(t_0) = \sum_{j \in [m]} C_j \cdot x(t_0 + j \cdot \tau). \end{aligned}$$

In the next a few paragraph, we show how to use the above Claim to prove Lemma 31.2.1.

We use 0^d to denote a length-d vector with 0 everywhere. We choose $t_0 = 0^d$ such that $\forall \tau \in \mathbb{R}^d_{>0}$ there always exist $C_1, \cdots, C_m \in \mathbb{C}$, and

$$x(0^d) = \sum_{j \in [m]} C_j \cdot x(j \cdot \tau).$$

By the Cauchy-Schwarz inequality, it implies that for any τ,

$$|x(0^d)|^2 \leq m \sum_{j \in [m]} |C_j|^2 |x(j \cdot \tau)|^2$$

At last, we obtain

$$
\begin{aligned}
|x(0^d)|^2 &= m^d \int_{[0,1/m]^d} |x(0^d)|^2 \mathrm{d}\tau \\
&\lesssim m^d \cdot \int_{[0,1/m]^d} \left(m \sum_{j=1}^{m} |x(j \cdot \tau)|^2 \right) \mathrm{d}\tau \\
&= m^{d+1} \cdot \sum_{j=1}^{m} \int_{[0,1/m]^d} |x(j \cdot \tau)|^2 \mathrm{d}\tau \\
&= m^{d+1} \cdot \sum_{j=1}^{m} \frac{1}{j^d} \int_{[0,j/m]^d} |x(\tau)|^2 \mathrm{d}\tau \\
&= m^{d+1} \cdot \sum_{j=1}^{m} \frac{1}{j^d} \int_{[0,1]^d} |x(\tau)|^2 \mathrm{d}\tau \\
&\lesssim m^{d+1} \log m \cdot \|x\|_T^2 \\
&\leq k^{O(m)} \|x\|_T^2, \qquad (31.1)
\end{aligned}
$$

where the third step follows by moving m outside of the integral and swapping the integration and the summation, the fourth step follows by replacing $j\tau$ by τ, the fifth step follows by $j/m \leq 1$, the sixth step follows by $\sum_{j=1}^{m} 1/j = O(\log m)$ and the definition of $\|x\|_T^2$, and the last step follows by $m = \mathrm{poly}(k)$.

Thus, we have the desired bound. □

31.3 Proof of Claim 31.2.2

For $x(t) = \sum_{i=1}^{k} v_i e^{2\pi \mathbf{i} f_i^\top t}$, where $f_i \in \mathbb{R}^d$ and $t \in \mathbb{R}^d$. We fix $t_0 \in \mathbb{R}^d$ and $\tau \in \mathbb{R}^d$ then rewrite $x(t_0 + j \cdot \tau)$ as a polynomial of $b_i = v_i \cdot e^{2\pi \mathbf{i} f_i^\top t_0}$ and $z_i = e^{2\pi \mathbf{i} f_i^\top \tau}$ for each $i \in [k]$.

$$\begin{aligned} x(t_0 + j \cdot \tau) &= \sum_{i=1}^{k} v_i e^{2\pi \mathbf{i} f_i^\top (t_0 + j\tau)} \\ &= \sum_{i=1}^{k} v_i e^{2\pi \mathbf{i} f_i^\top t_0} e^{2\pi \mathbf{i} f_i^\top j\tau} \\ &= \sum_{i=1}^{k} b_i \cdot z_i^j. \end{aligned}$$

Given k and $z_1, \cdots, z_k$, let $P(z) = \sum_{j=0}^{m} c_j z^j$ be the degree m polynomial in Lemma 5.4 ([CKPS16]).

$$\begin{aligned} \sum_{j=0}^{m} c_j x(t_0 + j\tau) &= \sum_{j=0}^{m} c_j \sum_{i=1}^{k} b_i \cdot z_i^j \\ &= \sum_{i=1}^{k} b_i \sum_{j=0}^{m} c_j \cdot z_i^j \\ &= \sum_{i=1}^{k} b_i P(z_i) \\ &= 0, \end{aligned}$$

where the last step follows by Property I of P(z) in Lemma 5.4 ([CKPS16]). From the Property II and III of $P(z)$ (Lemma 5.4 [CKPS16]), we obtain $x(t_0) = -\sum_{j=1}^{m} c_j x(t_0 + j\tau)$.

Chapter 32

Map-Reduce

Many modern parallel systems, such as MapReduce, Hadoop and Spark, can be modeled well by the MPC model. The MPC model captures well coarse-grained computation on large data — data is distributed to processors, each of which has a sublinear (in the input data) amount of memory and we alternate between rounds of computation and rounds of communication, where each machine can communicate an amount of data as large as the size of its memory. This model is stronger than the classical PRAM model, and it is an intriguing question to design algorithms whose running time is smaller than in the PRAM model.

One fundamental graph problem is connectivity. On an undirected graph with n nodes and m edges, $O(\log n)$ round connectivity algorithms have been known for over 35 years. However, no algorithms with better complexity bounds were known. In this work, we give **fully scalable, faster algorithms for the connectivity problem**, by parameterizing the time complexity as a function of the *diameter* of the graph. Our main result is a $O(\log D \log \log_{m/n} n)$ time connectivity algorithm for diameter-D graphs, using $\Theta(m)$ total memory. If our algorithm can use more memory, it can terminate in fewer rounds, and there is no lower bound on the memory per processor.

We extend our results to related graph problems such as spanning forest, finding a

DFS sequence, exact/approximate minimum spanning forest, and bottleneck spanning forest. We also show that achieving similar bounds for reachability in *directed graphs* would imply faster boolean matrix multiplication algorithms.

We introduce several new algorithmic ideas. We describe a general technique called *double exponential speed problem size reduction* which roughly means that if we can use total memory N to reduce a problem from size n to n/k, for $k = (N/n)^{\Theta(1)}$ in one phase, then we can solve the problem in $O(\log \log_{N/n} n)$ phases. In order to achieve this fast reduction for graph connectivity, we use a multistep algorithm. One key step is a carefully constructed truncated broadcasting scheme where each node broadcasts neighbor sets to its neighbors in a way that limits the size of the resulting neighbor sets. Another key step is *random leader contraction*, where we choose a smaller set of leaders than many previous works do.

This part is based upon the following previous publication

- Alexandr Andoni, Zhao Song, Clifford Stein, Zhengyu Wang, Peilin Zhong
  Parallel graph connectivity in log diameter rounds.
  FOCS 2018 [ASS+18]

32.1 Introduction

Recently, several parallel systems, including MapReduce [DG04, DG08], Hadoop [Whi12], Dryad [IBY+07], Spark [ZCF+10], and others, have become successful in practice. This success has sparked a renewed interest in algorithmic ideas for these parallel systems.

One important theoretical direction has been to develop good models of these modern systems and to relate them to classic models such as PRAM. The work of [FMS+10, KSV10, GSZ11, BKS13, ANOY14] have led to the model of *Massive Parallel Computing* (MPC) that balances accurate modeling with theoretical elegance. MPC is a variant of the Bulk Synchronous Parallel (BSP) model [Val90]. In particular, MPC allows N^δ space per machine (processor), where $\delta \in (0, 1)$ and N is the input size, with alternating rounds of unlimited local computation, and communication of up to N^δ data per processor. An MPC algorithm can equivalently be seen as a small circuit, with arbitrary, N^δ-fan-in gates; the depth of the circuit is the parallel time. Any PRAM algorithm can be simulated on MPC in the same parallel time [KSV10, GSZ11]. However, MPC is in fact more powerful than the PRAM: even computing the XOR of N bits requires near-logarithmic parallel-time on the most powerful CRCW PRAMs [BH89], whereas it takes constant, $O(1/\delta)$, parallel time on the MPC model.

The main algorithmic question of this area is then: for which problems can we design MPC algorithms that are *faster* than the best PRAM algorithms? Indeed, this question has been the focus of several recent papers, see, e.g., [KSV10, LMSV11, EIM11, ANOY14, AG18, AK17, IMS17b, CLM+18]. Graph problems have been particularly well studied and one fundamental problem is *connectivity in a graph*. While this problem has a standard logarithmic time PRAM algorithm [SV82], we do not know whether we can solve it faster in the MPC model.

While we would like *fully scalable* algorithms—which work for any value of $\delta > 0$—there have been graph algorithms that use space close to the number of vertices n of the graph. In particular, the result of [LMSV11] showed a faster algorithm for the setting when the space per machine is polynomially larger than the number of vertices, i.e., $s \geq n^{1+\Omega(1)}$, and hence the number of edges is necessarily $m \geq n^{1+\Omega(1)}$. In fact, similar space restrictions are pervasive for all known sub-logarithmic time graph algorithms, which require $s = \Omega(\frac{n}{\log^{O(1)} n})$ [LMSV11, AG18, AK17, CLM+18] (the only exception is [ANOY14] who consider geometric graphs). We highlight the work of [CLM+18], who manage to obtain *slightly sublinear* space of $n/\log^{\Omega(1)} n$ in $\log^{O(1)} \log n$ parallel time, for the approximate matching problem and [ABB+17] who obtain *slightly sublinear* space of $n/\log^{\Omega(1)} n$ in $O(\log \log n)$ parallel time. We note that the space of $\sim n$ also coincides with the space barrier of the semi-streaming model: essentially no graph problems are solvable in less than n space in the streaming model, unless we have many more passes; see e.g. the survey [McG09].

It remains a major open question whether there exist fully scalable connectivity MPC algorithms with sub-logarithmic time (e.g., for sparse graphs). There are strong indications that such algorithms do *not* exist: [BKS13] show logarithmic lower bounds for restricted algorithms. Alas, showing an unconditional lower bound may be hard to prove, as that would imply circuit lower bounds [RVW16].

In this work, we show **faster, fully scalable algorithms for the connectivity problem**, by parameterizing the time complexity as a function of the *diameter* of the graph. The diameter of the graph is the largest diameter of its connected components. Our main result is an $O(\log D \log \log_{m/n} n)$ time connectivity algorithm for diameter-D graphs with m edges. Parameterizing as a function of D is standard, say, in the distributed computing

literature [PRS16, HHW18]. In fact, some previous MPC algorithms for connectivity in the applied communities have been *conjectured* to obtain $O(\log D)$ time [RMCS13]; alas, we show in Section 32.11 the algorithm of [RMCS13] has a lower bound of $\Omega(\log n)$ time.

Our algorithms exhibit a tradeoff between the total amount of memory available and the number of rounds of computation needed. For example, if the total space is $\Omega(n^{1+\gamma'})$ for some constant $\gamma' > 0$, then our algorithms run in $O(\log D)$ rounds only.

32.1.1 The MPC model

Before stating our full results, we briefly recall the MPC model [BKS13]. A detailed discussion appears in Section 32.7, along with some core primitives implementable in the MPC model.

Definition 32.1.1 ($(\gamma, \delta)-$MPC model)**.** Fix parameters $\gamma, \delta > 0$, and suppose $N \geq 1$ is the input size. There are $p \geq 1$ machines (processors) each with local memory size $s = \Theta(N^{\delta})$, such that $p \cdot s = O(N^{1+\gamma})$. The space size is measured by words, each of $\Theta(\log(s \cdot p))$ bits. The input is distributed on the local memory of $\Theta(N/s)$ input machines. The computation proceeds in rounds. In each round, each machine performs computation on the data in its local memory, and sends messages to other machines at the end of the round. The total size of messages sent or received by a machine in a round is bounded by s. In the next round, each machine only holds the received messages in its local memory. At the end of the computation, the output is distributed on the output machines. Input/output machines and other machines are identical except that input/output machine can hold a part of the input/output. The parallel time of an algorithm is the number of rounds needed to finish the computation.

In this model, the space per machine is sublinear in N, and the total space is only an $O(N^{\gamma})$ factor more than the input size N. In this paper, we consider the case when δ is an arbitrary constant in $(0, 1)$. Our results are for both the most restrictive case of $\gamma = 0$ (total space is linear in the input size), as well as $\gamma > 0$ (for which our algorithms are a bit faster). The model from Definition 32.1.1 matches the model MPC(ϵ) from [BKS13] with $\epsilon = \gamma/(1 + \gamma - \delta)$ and the number of machines $p = O(N^{1+\gamma-\delta})$.

32.1.2 Our Results

While our main result is a $\sim \log D$ time connectivity MPC algorithm, our techniques extend to related graph problems, such as spanning forest, finding a DFS sequence, and exact/approximate minimum spanning forest. We also prove a lower bound showing that, achieving similar bounds for reachability in *directed graphs* would imply faster boolean matrix multiplication algorithms.

We now state our results formally. For all results below, consider an input graph $G = (V, E)$, with $n = |V|$, $N = |V| + |E|$, and D being the upper bound on the diameter of any connected component of G.

Connectivity: In the connectivity problem, the goal is to output the connected components of an input graph G, i.e. at the end of the computation, $\forall v \in V$, there is a unique tuple (x, y) with $x = v$ stored on an output machine, where y is called the color of v. Any two vertices u, v have the same color if and only if they are in the same connected component.

Theorem 32.1.1 (Connectivity in MPC, restatement of Theorem 32.8.4)**.** *For any $\gamma \in [0, 2]$ and any constant $\delta \in (0, 1)$, there is a randomized $(\gamma, \delta) -$ MPC algorithm (see Algorithm 32.3) which outputs the connected components of the graph G in $O(\min(\log D \cdot \log \frac{\log n}{\log(N^{1+\gamma}/n)}, \log n))$ parallel time. The success probability is at least 0.98. In addition, if the algorithm fails, then it returns* FAIL.

Notice that in the most restrictive case of $\gamma = 0$ and $m = n$, we obtain $O(\min(\log D \cdot \log \log n, \log n))$ time. When the total space is slightly larger, or the graph is slightly denser—i.e. $\gamma > c$ or $\log_n m > 1 + c$, where $c > 0$ is an arbitrarily small constant—then we obtain $O(\log D)$ time.

Remark 32.1.1. We note the concurrent and independent work of [ASW18], who also give a connectivity algorithm in the MPC model but with different guarantees. In particular, their runtime is parameterized as a function of λ, which is a lower bound on the spectral gap[1] of the connected components of G. For a graph G with n vertices and $m = \widetilde{O}(n)$ edges, their algorithm runs in $O(\log\log n + \log(1/\lambda))$ parallel time and uses $\widetilde{O}(n/\lambda^2)$ total space. In contrast, our algorithm has a runtime of $O(\log D \cdot \log\log_{N/n} n)$, where D is the largest diameter of a connected component of G, and $N = \Omega(m)$ is the total space available. To compare the two runtimes, we note that: 1) $D \leq O(\frac{\log n}{\lambda})$ for any undirected graph G; and 2) there exist sparse graphs G[2] with n vertices and $O(n)$ edges such that $\frac{1}{\lambda} \geq D \cdot n^{\Omega(1)}$ and $D \leq O(\log n)$. Thus, our results subsume [ASW18] in the case when total space is $N = n^{1+\Omega(1)}$, but are incomparable otherwise.

Spanning forest problem: In the spanning forest problem, the goal is to output a subset of edges of an input graph G such that the output edges together with the vertices of G form a spanning forest of the graph G. In the rooted spanning forest problem, in addition to the edges of the spanning forest, we are also required to orient the edge from child to parent, so that the parent-child pairs form a rooted spanning forest of the input graph G.

Theorem 32.1.2 (Spanning Forest, restatement of Theorem 32.8.14)**.** *For any $\gamma \in [0, 2]$ and any constant $\delta \in (0, 1)$, there is a randomized $(\gamma, \delta) -$ MPC algorithm (see Algorithm 32.11 and Algorithm 32.12) which outputs the rooted spanning forest of the graph G in $O(\min(\log D \cdot \log \frac{\log n}{\log(N^{1+\gamma}/n)}, \log n))$ parallel time. The success probability is at least 0.98. In addition, if*

[1]The spectral gap of a graph G is the second smallest eigenvalue of the normalized Laplacian of G.

[2]We can construct G as the following: a bridge connects two 3-regular expanders where each expander has $n/2$ vertices.

the algorithm fails, then it returns FAIL.

Our spanning forest algorithm can also output an approximation to the diameter, as follows.

Theorem 32.1.3 (Diameter Estimator, restatement of Theorem 32.8.15)**.** *For any $\gamma \in [0,2]$ and any constant $\delta \in (0,1)$, there is a randomized $(\gamma,\delta)-$*MPC *algorithm which outputs a diameter estimator D' of the input graph G in $O(\min(\log D \cdot \log \frac{\log n}{\log(N^{1+\gamma}/n)}, \log n))$ parallel time such that $D \leq D' \leq D^{O(\log(1/\gamma'))}$, where $\gamma' = \frac{\log(N^{1+\gamma}/n)}{\log n}$. The success probability is at least* 0.98*. In addition, if the algorithm fails, then it returns* FAIL*.*

Depth-First-Search sequence: If the input graph G is a tree, then we are able to output a Depth-First-Search sequence of that tree in $O(\log D)+T$ parallel time, where T is parallel time to compute a rooted tree (see Theorem 32.1.2 for our upper bound of T) for G. (See Section 32.7.2 for a discussion how to represent a sequence in the MPC model.)

Theorem 32.1.4 (DFS Sequence of a Tree, restatement of Theorem 32.8.21)**.** *Suppose the graph G is a tree. For any $\gamma \in [\beta,2]$ and any constant $\delta \in (0,1)$, there is a randomized $(\gamma,\delta)-$*MPC *algorithm (Algorithm 32.17) that outputs a Depth-First-Search sequence for the input graph G in $O(\min(\log D \cdot \log(1/\gamma), \log n))$ parallel time, where $\beta = \Theta(\log\log n/\log n)$. The success probability is at least* 0.98*. In addition, if the algorithm fails, then it returns* FAIL*.*

Applications of DFS sequence of a tree include lowest common ancestor, tree distance oracle, the size of every subtree, and others. See Section 32.6.4 for a more detailed discussion of the DFS sequence of a tree.

Minimum Spanning Forest: In the minimum spanning forest problem, the goal is to compute the minimum spanning forest of a weighted graph G.

Theorem 32.1.5 (Minimum Spanning Forest, restatement of Theorem 32.9.3)**.** *Consider a weighted graph G with weights $w : E \to \mathbb{Z}$ such that $\forall e \in E, |w(e)| \leq \text{poly}(n)$. For any $\gamma \in [0, 2]$ and any constant $\delta \in (0, 1)$, there is a randomized $(\gamma, \delta) - \text{MPC}$ algorithm which outputs a minimum spanning forest of G in $O(\min(\log D_{MSF} \cdot \log(\frac{\log n}{1+\gamma \log n}), \log n) \cdot \frac{\log n}{1+\gamma \log n})$ parallel time, where D_{MSF} is the diameter (with respect to the number of edges/hops) of a minimum spanning forest of G. The success probability is at least 0.98. In addition, if the algorithm fails, then it returns* FAIL*.*

We note that we require the bounded weights condition merely to ensure that each weight is described by one word.

Theorem 32.1.6 (Approximate Minimum Spanning Forest, restatement of Theorem 32.9.4)**.** *Consider a weighted graph G with weights $w : E \to \mathbb{Z}_{\geq 0}$ such that $\forall e \in E, |w(e)| \leq \text{poly}(n)$. For any $\epsilon \in (0, 1)$, $\gamma \in [\beta, 2]$ and any constant $\delta \in (0, 1)$, there is a randomized $(\gamma, \delta) - \text{MPC}$ algorithm which can output a $(1 + \epsilon)$ approximate minimum spanning forest for G in $O(\min(\log D_{MSF} \cdot \log(\frac{\log n}{\log(N^{1+\gamma}/(\epsilon^{-1} n \log n))}), \log n))$ parallel time, where $\beta = \Theta(\log(\epsilon^{-1} \log n)/\log n)$, and D_{MSF} is the diameter (with respect to the number of edges/hops) of a minimum spanning forest of G. The success probability is at least 0.98. In addition, if the algorithm fails, then it returns* FAIL*.*

Theorem 32.1.7 (Bottleneck Spanning Forest, restatement of Theorem 32.9.5)**.** *Consider a weighted graph G with weights $w : E \to \mathbb{Z}$ such that $\forall e \in E, |w(e)| \leq \text{poly}(n)$. For any $\gamma \in$*

$[0,2]$ *and any constant* $\delta \in (0,1)$*, there is a randomized* $(\gamma,\delta)-$MPC *algorithm which can output a bottleneck spanning forest for* G *in* $O(\min(\log D_{MSF} \cdot \log(\frac{\log n}{1+\gamma \log n}), \log n) \cdot \log(\frac{\log n}{1+\gamma \log n}))$ *parallel time, where* D_{MSF} *is the diameter (with respect to the number of edges/hops) of a minimum spanning forest of* G*. The success probability is at least* 0.98*. In addition, if the algorithm fails, then it returns* FAIL*.*

Conditional hardness for directed reachability. We also consider the reachability question in the directed graphs, for which we show similar to the above results are unlikely. In particular, we show that if there is a fully scalable multi-query directed reachability $(0,\delta)-$ MPC algorithm with $n^{o(1)}$ parallel time and polynomial local running time, then we can compute the Boolean Matrix Multiplication in $n^{2+\epsilon+o(1)}$ time for arbitrarily small constant $\epsilon > 0$. We note that the equivalent problem for *undirected graphs* can be solved in $O(\log D \log\log n)$ parallel time via Theorem 32.1.1.

Theorem 32.1.8 (Directed Reachability vs. Boolean Matrix Multiplication, restatement of Theorem 32.10.1)**.** *Consider a directed graph* $G = (V,E)$*. If there is a polynomial local running time, fully scalable* $(\gamma,\delta)-$MPC *algorithm that can answer* $|V|+|E|$ *pairs of reachability queries simultaneously for* G *in* $O(|V|^{\alpha})$ *parallel time, then there is a sequential algorithm which can compute the multiplication of two* $n \times n$ *boolean matrices in* $O(n^2 \cdot n^{2\gamma+\alpha+\epsilon})$ *time, where* $\epsilon > 0$ *is a constant which can be arbitrarily small.*

Finally, in Section 32.11 we show hard instances for the algorithm [RMCS13].

32.1.3 Our Techniques

In this section, we give an overview of the various techniques that we use in our algorithms. More details, as well as some of the low level details of the implementation in the MPC model, are defered to later sections.

Before getting into our techniques, we mention two standard tools to help us build our MPC subroutines. The first one is sorting: while in the PRAM model it takes $\sim \log N$ parallel time, sorting takes only constant parallel time in the MPC model [Goo99, GSZ11]. The second tool is indexing/predecessor search [GSZ11], which also has a constant parallel time in MPC. Furthermore, these two tools are fully scalable, and hence all the subroutines built on these two tools are also fully scalable. See Section 32.7 for how to use these two tools to implement the MPC operations needed for our algorithms.

Graph Connectivity: A natural approach to the graph connectivity problem is via the classic primitive of contracting to leaders: select a number of leader verteces, and contract every vertex (or most vertices) to a leader from its connected component (this is usually implemented by labeling the vertex by the corresponding leader). Indeed, many previous works (see e.g. [KSV10, RMCS13, KLM+14b]) are based on this approach. There are two general questions to address in this approach: 1) how to choose leader vertices, and 2) how to label each vertex by its leader. For example, the algorithm in [KSV10] randomly chooses half of the vertices as leaders, and then contracts each non-leader vertex to one of its neighbor leader vertex. Thus, in each round of their algorithm, the number of vertices drops by a constant fraction. At the same time, half of the vertices are leaders, and hence their algorithm still needs at least $\Omega(\log n)$ rounds to contract all the vertices to one leader. Note

that a constant fraction of leaders is needed to ensure that there is a constant fraction of non-leader vertices who are adjacent to at least one leader vertex and hence are contracted. This leader selection method appears optimal for some graphs, e.g. path graphs.

To improve the runtime to $\ll \log n$, one would have to choose a much smaller fraction of the vertices to be leaders. Indeed, for a graph where every vertex has a large degree, say at least $d \gg \log n$, we can choose fewer leaders: namely, we can choose each vertex to be a leader with probability $p = \Theta((\log n)/d)$. Then the number of leaders will be about $\widetilde{O}(n/d)$, while each non-leader vertex has at least one leader neighbor with high probability. After contracting non-leader vertices to leader vertices, the number of remaining vertices is only a $1/d$ fraction of original number of vertices.

By the above discussion, the goal would now be to modify our input graph G so that every vertex has a uniformly large degree, without affecting the connectivity of the graph. An obvious such modification is to add edges between pairs of vertices that are already in the same connected component. In particular, if a vertex v learns of a large number of vertices which are in the same connected component as v, then we can add edges between v and those vertices to increase the degree of v. A naïve way to implement the latter is via broadcasting: each vertex v first initializes a set S_v which contains all the neighbors of v, and then, in each *round*, every vertex v updates the set S_v by adding the union of the sets S_u over all neighbors u of v (old and new). This approach takes log-diameter number of rounds, and each vertex learns all vertices which are in the same connected component at the end of the procedure. However, in a single round, the total communication needed may be as huge as $\Omega(n^3)$ since each of n vertices may have $\Omega(n)$ neighbors, each with a set of size $\Omega(n)$.

Since our goal of each vertex v is to learn only d vertices in the same component (not

necessarily the entire component), we can therefore implement a "truncated" version of the above broadcasting procedure:

1. If S_v already had size d, then we do not need any further operation for S_v.
2. If u is in S_v, and S_u already has d vertices, then we can just put all the elements from S_u into S_v and thus S_v becomes of size d.
3. If $|S_v| < d$, and for every $u \in S_v$, the set S_u is also smaller than d, then we can implement one step of the broadcasting — add the union of S_u's, for all neighbors $u \in S_v$, to S_v.

In the above procedure, if the number of vertices in S_v is smaller than d after the i^{th} round, then we expect S_v to contain all the vertices whose distance to v is at most 2^i. Thus, the above procedure also takes at most log-diameter rounds. Furthermore, the total communication needed is at most $O(n \cdot d^2)$.

Our full graph connectivity algorithm implements the above "truncated broadcasting" procedure iteratively, for values d that follow a certain "schedule", depending on the available space. At the beginning of the algorithm, we have an n vertex graph G with diameter D, and a total of $\Omega(m)$ space. The algorithm proceeds in phases, where each phase takes $O(\log D)$ rounds of communication. In the first phase, the starting number of vertices is $n_1 = n$. We implement a truncated broadcasting procedure where the target degree d is $d_1 = (m/n_1)^{1/2}$, using $O(\log D)$ rounds and $O(m)$ total space. Then we can randomly select $\widetilde{O}(n_1/d_1)$ leaders, and contract all the non-leader vertices to leader vertices. At the end of the first phase, the total number of remaining vertices is at most $n_2 = \widetilde{O}(n_1/d_1) = \widetilde{O}(n_1^{1.5}/m^{0.5})$. In general,

suppose, at the beginning of the i^{th} phase, the number of remaining vertices is n_i. Then we use the truncated broadcasting procedure for value d set to $d_i = (m/n_i)^{1/2}$, thus making each vertex have degree at least $d_i = (m/n_i)^{1/2}$ in $O(\log D)$ number of communication rounds and $O(m)$ total space. Then we choose $\widetilde{O}(n_i/d_i)$ leaders, and, after contracting non-leaders, the number n_{i+1} of remaining vertices is at most $\widetilde{O}(n_i^{1.5}/m^{0.5})$. Let us look at the progress of the value d_i. We have that $d_{i+1} = \widetilde{\Omega}((m/n_{i+1})^{1/2}) = \widetilde{\Omega}((m^{1.5}/n_i^{1.5})^{1/2}) = \widetilde{\Omega}(d_i^{1.5})$. Thus, we are making double exponential progress on d_i, which implies that the total number of phases needed is at most $O(\log \log_{m/n} n)$, and the total parallel time is thus $O(\log D \cdot \log \log_{m/n} n)$.

This technique of double-exponential progress is more general and extends to other problems beyond connectivity. In particular, for a problem, suppose its size is characterized by a parameter n (not necessarily the input size—e.g. in connectivity problem, n is the number of vertices). When n is a constant, the problem can be solved in $O(1)$ parallel time. If there is a procedure that uses total space $\Theta(m)$ to reduce the problem size to at most n/k for $k = (m/n)^c, c = \Omega(1)$, then we can repeat the procedure $O(\log \log_{m/n} n)$ times to solve the overall problem. In particular, after repeating the procedure i times, the problem size is $n_i \leq n_{i-1}/(m/n_{i-1})^c \leq n \cdot (n/m)^{(1+c)^i - 1}$. We call this technique *double-exponential speed problem size reduction*.

Remark 32.1.2. For any problem characterized by a size parameter n, if we can use parallel time T and total space $\Theta(m)$ to reduce the problem size such that the reduced problem size is n/k for $k = (m/n)^{\Omega(1)}$, then we can solve the problem in $O(m)$ total space and $O(T \cdot \log \log_{m/n} n)$ parallel time.

Spanning Forest and Diameter Estimator: Extending a connectivity algorithm to a spanning forest algorithm is usually straightforward. For example, in [KSV10], they only contract a non-leader vertex to an adjacent leader vertex, thus their algorithm can also give a spanning forest, using the contracted edges. Here however, extending our connectivity algorithm to a spanning forest algorithm requires several new ideas. In our connectivity algorithm, because of the added edges, we only ensure that when a vertex u is contracted to a vertex v, u and v must be in the same connected component; but u and v may not be adjacent in the original graph. Thus, we need to record more information to help us build a spanning forest.

We can represent a forest as a collection of parent pointers $\mathrm{part}(v)$, one for each vertex $v \in V$. If v is a root in the forest, then we let $\mathrm{part}(v) = v$. We use $\mathrm{dep}_{\mathrm{par}}(v)$ to denote the depth of v in the forest, i.e. $\mathrm{dep}_{\mathrm{par}}(v)$ is the distance from v to its root. Let $\mathrm{dist}_G(u, v)$ denote the distance between two vertices u and v in a graph G.

Our connectivity algorithm uses the "neighbor increment" procedure described above. We observed that if the set S_v has fewer than d vertices after the i^{th} round, then S_v should contain all the vertices with distance at most 2^i to v. This motivates us to maintain a shortest path tree for S_v, with root v. In the i^{th} round, if we need to update S_v to be $\bigcup_{u \in S_v} S_u$, then we can update the shortest path tree of S_v in the following way:

1. For each $x \in S_u$ for some $u \in S_v$, we can create a tuple (x, u).

2. Then, for each $x \in \left(\bigcup_{u \in S_v} S_u\right) \setminus S_v$, we can sort all the tuples $(x, u_1), (x, u_2), \cdots, (x, u_k)$ such that u_1 minimizes $\min_{u \in S_v} \mathrm{dist}_G(v, u) + \mathrm{dist}_G(u, x)$. Since u is in S_v, x is in S_u, it is easy to get the value of $\mathrm{dist}_G(v, u), \mathrm{dist}_G(u, x)$ by the information of shortest path

tree for S_v and S_u. Then we set the new parent of x in the shortest path tree for S_v to be the parent of x in the shortest path tree for S_{u_1}.

Since S_v before the update contains all the vertices which have distance to v at most 2^{i-1}, the union of the shortest path from x to u_1 and the shortest path from u_1 to v must be the shortest path from x to v. Then by induction, we can show that the parent of x in the shortest path tree for S_{u_1} is also the parent of x in the shortest path tree for updated S_v. Thus, this modified "neighbor increment" procedure can find n local shortest path trees where there is a tree with root v for each vertex v. Furthermore, the procedure still takes $O(\log D)$ rounds. And we can still use $O(nd^2)$ total space to make each shortest path tree have size at least d. Next, we show how to use these n local shortest path trees to construct a forest with the roots in the forest being the leaders.

As discussed in the connectivity algorithm, if every local shortest path tree has size at least d, we can choose each vertex as a leader with probability $p = \Theta((\log n)/d)$ and then every tree will contain at least one leader with high probability. Let L be the set of sampled leaders, and let $\text{dist}_G(v, L)$ be defined as $\min_{u \in L} \text{dist}_G(v, u)$. Let v be a non-leader vertex, i.e. $v \in V \setminus L$. According to the shortest path tree for S_v [3], since $L \cap S_v \neq \emptyset$, we can find a child u of the root v such that $\text{dist}_G(v, L) > \text{dist}_G(u, L)$; in this case we set $\text{part}(v) = u$. For vertex $v \in L$, we can set $\text{part}(v) = v$. We can see now that par denotes a rooted forest where the roots are sampled leaders. Furthermore, since $\forall v \notin L$, $(v, \text{part}(v))$ is from the shortest path tree for S_v, we know that v and $\text{part}(v)$ are adjacent in the original graph G. After doing

[3]The construction of S_v for spanning forest algorithm is slightly different from that described in the connectivity algorithm. S_v in spanning forest algorithm has a stronger property: $\forall u \in V \setminus S_v$, $\text{dist}_G(u, v)$ must be at least $\text{dist}_G(u', v)$ for any $u' \in S_v$.

the above for all nodes v, the forest denoted by the resulting vector par must be a subgraph of the spanning forest of G. We then apply the standard doubling algorithm to contract all the vertices to their leaders (roots), in $O(\log D)$ rounds. Therefore, the problem is reduced to finding a spanning forest in the contracted graph. The number of vertices remaining in the contracted graph is at most $\widetilde{O}(n/d)$, where $d = (m/n)^{\Theta(1)}$. By Remark 32.1.2, we can output a spanning forest in $O(\log D \cdot \log\log_{m/n} n)$ parallel time.

Although the above algorithm can output the edges of a spanning forest, it cannot output a rooted spanning forest. To output a rooted spanning forest, we follow a top-down construction. Suppose now we have a rooted spanning forest of the contracted graph. Since we have all the information of how vertices were contracted, we know the contraction trees in the original graph. To merge these contraction trees into the rooted spanning forest of the contracted graph, we only need to change the root of each contraction tree to a proper vertex in that tree. This changing root operation can be implemented by the doubling algorithm via a divide-and-conquer approach.

Since the spanning forest algorithm needs $O(\log\log_{m/n} n)$ phases to contract all vertices to a single vertex, the total parallel time to compute a rooted spanning forest is $O(\log D \cdot \log\log_{m/n} n)$. Furthermore, the depth of the rooted spanning forest will be at most $O(D^{O(\log\log_{m/n} n)})$. Thus, we can use the doubling algorithm to calculate the depth of the tree, and output this depth as an estimator of the diameter of the input graph.

Depth-First-Search Sequence: Here, when the input graph G is a tree, our goal is to output a DFS sequence for this tree. Once we have this sequence, it is easy to output a rooted tree. Thus, computing a DFS sequence is at least as hard as computing a rooted tree,

and all the previous algorithms need $\Omega(\log n)$ parallel time to do so.

First of all, we use our spanning forest algorithm to compute a rooted tree, reducing the problem to computing a DFS sequence for a rooted tree. The idea is motivated by TeraSort [O'M08]. If the size of the tree is small enough such that it can be handled by a single machine, then we can just use a single machine to generate its DFS sequence. Otherwise, our algorithm can be roughly described as follows. (Recall that δ is the parameter such that each machine has $\Theta(n^{\delta})$ local memory.)

1. Sample $n^{\delta/2}$ leaves $l_1, l_2, \cdots, l_s$.
2. Determine the order of sampled leaves in the DFS sequence.
3. Compute the DFS sequence $\widetilde{A}$ of the tree which only consists of sampled leaves and their ancestors.
4. Compute the DFS sequence A_v of every root-v subtree which does not contain any sampled leaf.
5. Merge $\widetilde{A}$ and all the A_v.

The first and second steps go as follows. Since we only sample $n^{\delta/2}$ leaves, we can send them to a single machine. We generate queries for every pair of sampled leaves where each query (l_i, l_j) queries the lowest common ancestor of (l_i, l_j). We have n^{δ} such queries in total. Since the input tree is rooted, we can use a doubling algorithm to preprocess a data structure in $O(\log D)$ parallel time and answer all the queries simultaneously in $O(\log D)$ parallel time. Thus, we know the lowest common ancestor of any pair of sampled leaves,

and we can store this all on a single machine. Based on the information of lowest common ancestors of each pair of sampled leaves, we are able to determine the order of the leaves.

For the third step, suppose the sampled leaves have order $l_1, l_2, \cdots, l_s$. Let v be the root of the tree. Then the DFS sequence $\widetilde{A}$ should be: the path from v to l_1, the path from l_1 to the lowest common ancestor of (l_1, l_2), the path from the lowest common ancestor of (l_1, l_2) to l_2, the path from l_2 to the lowest common ancestor of (l_2, l_3), ..., the path from l_s to v. We can find these paths simultaneously by a doubling algorithm together with a divide-and-conquer algorithm in $O(\log D)$ parallel time.

In the fourth step, we apply the procedure recursively. Suppose the total number of leaves in the tree is $q \leq n$. Since we randomly sampled $n^{\delta/2}$ number of leaves, with high probability, each subtree which does not contain a sampled leaf will have at most $O(q/n^{\delta/2})$ number of leaves. Thus, the depth of the recursion will be at most a constant, $O(1/\delta)$.

Minimum Spanning Forest and Bottleneck Spanning Forest. Recall that the input is a graph $G = (V, E = (e_1, e_2, \cdots, e_m))$ together with a weight function w on E. Without loss of generality, we only consider the case when all the weights of edges are different, i.e. $w(e_1) < w(e_2) < \cdots < w(e_m)$. Since the weights of edges are different, the minimum spanning forest of the graph is unique. By Kruskal's algorithm, the diameter of the graph induced by the first i edges for any $i \in [m]$ is at most the depth of the minimum spanning forest. Now, let us use D to denote the depth of the minimum spanning forest.

We first discuss the minimum spanning forest algorithm. A crucial observation of Kruskal's algorithm is: if we want to determine which edges in $e_i, e_{i+1}, \cdots, e_j$ are in the minimum spanning forest, we can always contract the first $i-1$ edges to obtain a graph G',

run a minimum spanning forest algorithm on the contracted graph G', and observe whether an edge is included in the spanning forest of G'. Thus, if the total space is $\Theta(m^{1+\gamma})$, we can have m^{γ} copies of the graph, where the i^{th} copy contracts the first $(i-1)\cdot m^{1-\gamma}$ edges. Thus, we are able to divide the edges into m^{γ} groups where each group has $m^{1-\gamma}$ number of edges. We only need to solve the minimum spanning forest problem for each group. Then in the second phase, we can divide the edges into $m^{2\gamma}$ groups where each group has $m^{1-2\gamma}$ number of edges. Thus, the total number of phases needed is at most $O(1/\gamma)$. In each phase, we just need to run our connectivity algorithm to contract the graph.

For the approximate minimum spanning forest algorithm, we use a similar idea. If we want a $(1+\epsilon)$ approximation, then we round each weight to the closest value $(1+\epsilon)^i$ for some integer i. After rounding, there are only $O(1/\epsilon \cdot \log n)$ edge groups. Since our total space is at least $\Omega(m \log(n)/\epsilon)$, we can make $O(1/\epsilon \cdot \log n)$ copies of the graph. The i^{th} copy of the graph contracts all the edges in group $1, 2, \cdots, i-1$. Then, we only need to run our spanning forest algorithm on each copy to determine which edges should be chosen in each group.

Another application of our *double exponential speed problem size reduction* technique is bottleneck spanning forest. For the bottleneck spanning forest, suppose we have $\Theta(km)$ total space. We can have k copies of the graph where the i^{th} copy contracts the first $(i-1)\cdot m/k$ number of edges. We can determine the group of $O(m/k)$ edges which contains the bottleneck edge. Thus, we reduce the problem to $O(m/k)$. According to Remark 32.1.2, the number of phases is at most $O(\log \log_k m)$, and each phase needs T parallel time, where T is the parallel time for spanning forest.

Directed Reachability vs. Boolean Matrix Multiplication If there is a fully scalable multi-query directed reachability MPC algorithm with almost linear total space, we can simulate the algorithm in sequential model. Thus, it will imply a good sequential multi-query directed reachability algorithm which implies a good sequential Boolean Matrix Multiplication algorithm.

32.1.4 Roadmap

The rest of the paper contains the technical details of our algorithms. In Section 32.2, we described a simplified connectivity algorithm. In Section 32.3, we described the notations. In Sections 32.4, 32.5, and 32.6, we give the details of our main algorithms for connectivity, spanning forest and depth first search sequence. In these sections, we focus on the design of the algorithms and the analysis of the number of rounds. In Section 32.7, we describe the MPC model in detail and discuss some known primitives in that model. In Section 32.8, we discuss how to implement the details of our algorithms in the MPC model to achieve the bounds claimed in the previous sections. In Section 32.9, we show how to apply our connectivity and spanning forest algorithm in minimum spanning forest and bottleneck spanning forest problems. In Section 32.11, we show hard instances for the algorithm [RMCS13]. In Section 32.12, we show an alternative approach for random leader selection.

32.2 A Simplified Batch Algorithm for Connectivity

In this section, we show a simplified version of our connectivity algorithm.

Firstly, let us describe the simplified version of truncated broadcasting procedure in the following. Since G' is obtained by adding edges between the vertices in the same component of G. G' will preserve the connectivity of G. The parallel time needed is at most $O(\log D)$ where D is the diameter of G. The procedure takes at most $O(nd^2+m)$ total space.

Truncated Broadcasting for Neighbor Increment:

- **Input**:
  - A graph $G = (V, E)$ with $n = |V|$ vertices and $m = |E|$ number of edges.
  - A parameter d.
- **Output**:
  - A graph $G' = (V, E')$ such that $\forall v \in V$, $|\Gamma(v)| \geq d$. $\triangleright$ $\Gamma(v)$ denotes the neighbors of v.
- **While** $\exists x \in V$ such that $|\Gamma(x)| < d$:
  - For each $v \in V$ with $|\Gamma(v)| < d$:
    - If $\exists u \in \Gamma(v)$ which has $|\Gamma(u)| \geq d$, then $\Gamma(v) \leftarrow \Gamma(v) \cup \Gamma(u)$.
    - Otherwise, $\Gamma(v) \leftarrow \Gamma(v) \cup \bigcup_{u \in \Gamma(v)} \Gamma(u)$.

We can apply the above procedure to make each vertex have a large degree. Next, let us briefly describe how to choose the leaders and implement the contraction operation for the graph where each vertex has a large degree. The following procedure just needs $O(n+m)$ total space and $O(1)$ parallel time. If every vertex has degree at least d, then in the following

procedure we can reduce the number of vertices to $\widetilde{O}(n/d)$ by contracting all the vertices to $\widetilde{O}(n/d)$ number of leaders.

Random Leader Contraction:

- **Input**:
  - A graph $G = (V, E)$ with $n = |V|$ vertices where each vertex has degree at least d.
- **Output**:
  - A graph $G' = (V', E')$ with $\widetilde{O}(n/d)$ vertices.
  - A mapping par : $V \to V'$, such that part(v) is the vertex that v contracts to.
- **Leader Selection:**
  - Let L denote the set of leaders.
  - For each $v \in V$, with probability at least $\widetilde{\Omega}(1/d)$, choose v as a leader, i.e. $L \leftarrow L \cup \{v\}$.
- **Contraction:**
  - For each $v \in L$, let part$(v) = v$, and put v into V'.
  - For each $v \in V \setminus L$, choose $u \in \Gamma(v) \cap L$, and set part$(u) = v$.
  - For each $(u, v) \in E$, if part$(u) \neq$ part(v), put the edge (part(u), part(v)) into E'.

Finally, we describe the simplified version of our connectivity algorithm in the following.

Connectivity:

- **Input**:
  - A graph $G = (V, E)$ with $n = |V|$ vertices and $m = |E|$ edges.
  - Total space N which is $\Theta(m)$.
- **Output**:
  - A mapping $\text{col} : V \to V$ satisfies $\forall u, v \in V, \text{col}(u) = \text{col}(v)$ if and only if u and v are connected.
- **Initialization:**
  - Let $G_0 \leftarrow G, n_0 \leftarrow n$.
- **In phase i:**
  - Compute G'_{i-1} : Increase the degree of every vertex in G_{i-1} to at least $d_i = \Theta\left((N/n_{i-1})^{1/2}\right)$.
  - Compute G_i : Select $n_i = \widetilde{O}(n_{i-1}/d_i)$ leaders in G'_{i-1} and contract all the vertices to the leaders.
  - If v is contracted to u, record $\text{part}(v) = u$.
  - If G_i does not have any edges, then for every vertex v in G_i, set $\text{part}(v) = v$, and exit the loop.
- **Finding the root leader:**
  - For each $v \in V$, find the root of v in par, i.e. find $u = \text{part}(\text{part}(\cdots \text{part}(v)))$ such that $\text{part}(u) = u$.
  - Set $\text{col}(v) = u$.

After phase i, the number of vertices survived is at most $\widetilde{O}(n_{i-1}/(N/n_{i-1})^{1/2})$. By Remark 32.1.2, there will be at most $O(\log \log_{N/n} n)$ phases. For phase i, we need $O(\log D)$ parallel time to increase the neighbors of every vertex in G_{i-1}. The total parallel time is thus

$O(\log D \cdot \log \log_{N/n} n)$. The total space used in phase i is at most $O(m + (N/n_{i-1})^{1/2} \cdot n_{i-1}) = O(N)$.

32.3 Notations

$[n]$ denotes the set $\{1, 2, \cdots, n\}$. Let G be an undirected graph with vertex set V and edge set E. For $v \in V$, $\Gamma_G(v)$ denotes the set of neighbors of v in G, i.e. $\Gamma_G(v) = \{u \in V \mid (v, u) \in E\}$. For any $u, v \in V$, $\mathrm{dist}_G(u, v)$ denotes the distance between u, v in graph G. If u, v are not in the same connected component, then $\mathrm{dist}_G(u, v) = \infty$. If u, v are in the same connected component, then $\mathrm{dist}_G(u, v) < \infty$. For $v \in V$, $\{u \in V \mid \mathrm{dist}_G(u, v) < \infty\}$ is the set of all the vertices in the same connected component as v. The diameter $\mathrm{diam}(G)$ of G is the largest diameter of its components, i.e. $\mathrm{diam}(G) = \max_{u,v \in V : \mathrm{dist}_G(u,v) < \infty} \mathrm{dist}_G(u, v)$.

32.4 Graph Connectivity

32.4.1 Neighbor Increment Operation

In this section, we describe a procedure which can increase the number of neighbors of every vertex and preserve the connectivity at the same time. The input of the procedure is an undirected graph $G = (V, E)$ and a parameter m which is larger than $|V|$. The output is a graph $G' = (V, E')$ such that for each vertex v, either the connected component which contains v is a clique or v has at least $\lceil (m/|V|)^{1/2} \rceil - 1$ neighbors. Furthermore, $|E'| \leq |E| + m$. We use $\Gamma_G(v)$ to denote the neighbors of v in graph G, i.e. $\Gamma_G(v) = \{u \in V \mid (u, v) \in E\}$. Similarly, we let $\Gamma_{G'}(v)$ be the neighbors of v in G', i.e. $\Gamma_{G'}(v) = \{u \in V \mid (u, v) \in E'\}$.

Lemma 32.4.1. *Let $G = (V, E)$ be an undirected graph, $m \in \mathbb{Z}_{\geq 0}$ which has $m \geq 4|V|$. Let $n = |V|$. Let r be the value at the end of the procedure* NeighborIncrement(m, G) *(Algorithm 32.1.) Then $\forall i \in \{0, 1, \cdots, r\}, v \in V$, $S_v^{(i)}$ satisfies the following properties:*

1. $v \in S_v^{(i)}$.
2. $\forall u \in S_v^{(i)}$, $\mathrm{dist}_G(u, v) < \infty$.
3. $|S_v^{(i)}| < \lceil (m/n)^{1/2} \rceil \Rightarrow S_v^{(i)} = \{u \in V \mid \mathrm{dist}_G(u, v) \leq 2^i\}$.
4. $|S_v^{(i)}| \leq m/n$.

Proof. For property 1, we can prove it by induction. When $i = 0$, due to line 3, we know $v \in S_v^{(0)}$. Suppose property 1 holds for $S_v^{(i-1)}$ for all $v \in V$. If $S_v^{(i)}$ is updated by line 17, there are two cases: 1. if $u = v$, then $v \in S_u^{(i-1)}$, and the condition of line 17 does not hold, thus $v \in S_v^{(i)}$; 2. if $u \neq v$, then after implementing line 17, v will not be removed, thus $v \in S_v^{(i)}$.

If $S_v^{(i)}$ is updated by line 20, then since $v \in S_v^{(i-1)}$, v is also in the set $S_v^{(i)}$. Thus, property 1 holds for every $S_v^{(i)}$.

For property 2, we can prove it by induction. When $i = 0$, it is easy to see $S_v^{(0)} \subseteq \Gamma_G(v) \cup \{v\}$, thus property 2 holds for it. Suppose property 2 holds for $S_v^{(i-1)}$ for all $v \in V$. If $S_v^{(i)}$ is updated by line 17, then since $u \in S_v^{(i-1)}$ and $S_v^{(i)} \subseteq S_u^{(i-1)} \cup \{v\}$, all the vertices from $S_v^{(i)}$ are in the same connected component as u and u is in the same connected component as v. Thus, property 2 holds in this case. If $S_v^{(i)}$ is updated by the line 20, then $\forall p \in S_v^{(i)}$, there exists $u \in S_v^{(i-1)}$ such that $p \in S_u^{(i-1)}$. We have p is in the same connected component as u, and u is in the same connected component as v. Thus, property 2 also holds in this case.

For property 3, we can prove it by induction. When $i = 0$, due to line 7, we have $|S_v^{(0)}| < \lceil (m/n)^{1/2} \rceil \rightarrow S_v^{(0)} = \Gamma_G(v) \cup \{v\} = \{u \in V \mid \text{dist}_G(u, v) \leq 1\}$. Suppose property 3 holds for $S_v^{(i-1)}$ for all $v \in V$. Since if $|S_v^{(i)}| < \lceil (m/n)^{1/2} \rceil$, then $S_v^{(i)}$ can only be updated by line 20, and $\forall u \in S_v^{(i-1)}$, it has $|S_u^{(i-1)}| < \lceil (m/n)^{1/2} \rceil$. Thus, $S_v^{(i)} = \bigcup_{u \in S_v^{(i-1)}} S_u^{(i-1)} = \bigcup_{u \in V, \text{dist}_G(u,v) \leq 2^{i-1}} \{p \in V \mid \text{dist}_G(p, u) \leq 2^{i-1}\} = \{u \in V \mid \text{dist}_G(u, v) \leq 2^i\}$. Thus, property 3 holds.

For property 4, we can prove it by induction. When $i = 0$, due to line 7, $\forall v \in V$, we have $|S_v^{(0)}| \leq \lceil (m/n)^{1/2} \rceil \leq m/n$, where the last inequality follows by $m/n \geq 4$. Now suppose property 4 holds for $S_v^{(i-1)}$ for all $v \in V$. If $S_v^{(i)}$ is updated by line 17, then $|S_v^{(i)}| = |S_u^{(i-1)}| \leq m/n$. If $S_v^{(i)}$ is updated by line 20, we know $\forall u \in S_v^{(i-1)}, |S_u^{(i-1)}| < \lceil (m/n)^{1/2} \rceil$. Notice that by property 1, $v \in S_v^{(i-1)}$, so $|S_v^{(i-1)}| < \lceil (m/n)^{1/2} \rceil$. Thus, $|S_v^{(i)}| = |\bigcup_{u \in S_v^{(i-1)}} S_u^{(i-1)}| \leq (m/n)^{1/2} \cdot (m/n)^{1/2} \leq m/n$. □

Algorithm 32.1 Neighbor Increment Operation

procedure NeighborIncrement($m, G = (V, E)$) ▷ Lemma 32.4.1, Lemma 32.4.2
▷ Output: $G' = (V, E')$
  Initially, $n = |V|, E' = \emptyset$ and let $S_v^{(0)} = \{v\}$ for all $v \in V$.
  for $v \in V$ **do** ▷ Initially, let $S_v^{(0)}$ be the set (or subset) of direct neighbors
    for $u \in \Gamma_G(v)$ **do**
      if $|S_v^{(0)}| < \lceil (m/n)^{1/2} \rceil$ **then**
        $S_v^{(0)} \leftarrow S_v^{(0)} \cup \{u\}$.
      end if
    end for
  end for
  $r \leftarrow 1$.
  for true do
    for $v \in V$ **do**
      if $\exists u \in S_v^{(r-1)}, |S_u^{(r-1)}| \geq \lceil (m/n)^{1/2} \rceil$ **then** ▷ neighbor u has many neighbors
        $S_v^{(r)} = S_u^{(r-1)} \cup \{v\}$.
        if $|S_v^{(r)}| > |S_u^{(r-1)}|$ **then**
          $S_v^{(r)} \leftarrow S_v^{(r)} \setminus \{u\}$.
        end if
      else
        $S_v^{(r)} = \bigcup_{u \in S_v^{(r-1)}} S_u^{(r-1)}$. ▷ neighbors of v's neighbors are v's new neighbors.
      end if
    end for
    ▷ $S_v^{(r)}$ is large or is a component
    if $\forall v \in V$, either $|S_v^{(r)}| \geq \lceil (m/n)^{1/2} \rceil$ or $|S_v^{(r)}| = |S_v^{(r-1)}|$ **then**
      Let $E' = E \cup \bigcup_{v \in V} \{(v, u) \mid v \in S_u^{(r)}$ or $u \in S_v^{(r)}, u \neq v\}$.
      return $G' = (V, E')$
    else
      $r \leftarrow r + 1$.
    end if
  end for
end procedure

The following definition defines the number of iterations of Algorithm 32.1.

Definition 32.4.1. Given an undirected graph $G = (V, E)$ and a parameter $m \in \mathbb{Z}_{\geq 0}, m \geq 4|V|$, the number of iterations of NeighborIncrement(m, G) (Algorithm 32.1) is the value of r at the end of the procedure.

In the following lemma, we characterize the properties of Algorithm 32.1.

Lemma 32.4.2. *Let $G = (V, E)$ be an undirected graph, $m \in \mathbb{Z}_{\geq 0}$ which has $m \geq 4|V|$. Let $G' = (V, E')$ be the output of* NeighborIncrement(m, G). *We have:*

1. *The number of iterations (Definition 32.4.1), $r \leq \min(\lceil \log(\mathrm{diam}(G)) \rceil, \lceil \log(m/n) \rceil) + 1$.*

2. *For all $u, v \in V$, $\mathrm{dist}_G(u, v) < \infty \Leftrightarrow \mathrm{dist}_{G'}(u, v) < \infty$.*

3. *$\forall v \in V$, if $|\Gamma_{G'}(v)| < \lceil (m/n)^{1/2} \rceil - 1$, then the connected component in G' which contains v is a clique. It also implies that $\forall u, v \in V$, if $|\Gamma_{G'}(v)| < \lceil (m/n)^{1/2} \rceil - 1$ and $|\Gamma_{G'}(u)| \geq \lceil (m/n)^{1/2} \rceil - 1$, then $\mathrm{dist}_{G'}(u, v) = \infty$.*

4. *$E \subseteq E', |E'| \leq |E| + m$.*

Proof. For property 1, if $r > \lceil \log(\mathrm{diam}(G)) \rceil + 1$, then let $i = \lceil \log(\mathrm{diam}(G)) \rceil + 1$. Let $v \in V$. By property 3 of Lemma 32.4.1, if $|S_v^{(i)}| < \lceil (m/n)^{1/2} \rceil$, then $S_v^{(i)} = \{u \in V \mid \mathrm{dist}_G(u, v) \leq 2^i\} = \{u \in V \mid \mathrm{dist}_G(u, v) \leq 2 \cdot 2^{\lceil \log(\mathrm{diam}(G)) \rceil}\} = \{u \in V \mid \mathrm{dist}_G(u, v) < \infty\}$. Furthermore, if $|S_v^{(i)}| < \lceil (m/n)^{1/2} \rceil$, then $|S_v^{(i-1)}| < \lceil (m/n)^{1/2} \rceil$, which means that $S_v^{(i-1)} = \{u \in V \mid \mathrm{dist}_G(u, v) \leq 2^{i-1}\} = \{u \in V \mid \mathrm{dist}_G(u, v) \leq 2^{\lceil \log(\mathrm{diam}(G)) \rceil}\} = \{u \in V \mid \mathrm{dist}_G(u, v) <$

$\infty\} = S_v^{(i)}$. Then due to the condition in line 24, it will end the procedure in this round, which contradicts to $r > \lceil \log(\text{diam}(G)) \rceil + 1 = i$. Similarly, if $r > \lceil \log(m/n) \rceil + 1$, then let $i = \lceil \log(m/n) \rceil + 1$. Furthermore, if $|S_v^{(i)}| < \lceil (m/n)^{1/2} \rceil$, then $|S_v^{(i-1)}| < \lceil (m/n)^{1/2} \rceil$, which means that $S_v^{(i-1)} = \{u \in V \mid \text{dist}_G(u, v) \leq 2^{i-1}\} \neq \{u \in V \mid \text{dist}_G(u, v) \leq 2^i\} = S_v^{(i)}$. Thus, there exists $u \in S_v^{(i)}$ such that $|S_v^{(i)}| > \text{dist}_G(u, v) > 2^{i-1} > m/n \geq \lceil (m/n)^{1/2} \rceil$ which leads to a contradiction.

For property 2, if u, v are in the same connected component in G, then since $E \subseteq E'$, u, v are in the same connected component in G'. If u, v are in the same connected component in G', then there should be a path $u = u_1 \to u_2 \to \cdots \to u_p = v$ in G', i.e. $\forall j \in [p-1], (u_j, u_{j+1}) \in E'$. $(u_j, u_{j+1}) \in E'$ implies that either $(u_j, u_{j+1}) \in E$ or $u_j \in S_{u_{j+1}}^{(r)}$ or $u_{j+1} \in S_{u_j}^{(r)}$. By property 2 of Lemma 32.4.1, we know that $\forall j \in [p-1]$, u_j and u_{j+1} are in the same connected component in G. Thus, u and v are in the same connected component in G.

For property 3, due to line 25, if $|\Gamma_{G'}(v)| < \lceil (m/n)^{1/2} \rceil - 1$, then we have $|S_v^{(r)}| < \lceil (m/n)^{1/2} \rceil$. By property 3 of Lemma 32.4.1, and the condition in line 24, we know $\{u \in V \mid \text{dist}_G(u, v) \leq 2^r\} = S_v^{(r)} = S_v^{(r-1)} = \{u \in V \mid \text{dist}_G(u, v) \leq 2^{r-1}\}$. Thus, $S_v^{(r)} = \{u \in V \mid \text{dist}_G(u, v) < \infty\}$. Due to property 2, we have $\Gamma_{G'}(v) \cup \{v\} \subseteq \{u \in V \mid \text{dist}_G(u, v) < \infty\}$. Notice that $S_v^{(r)} \subseteq \Gamma_{G'}(v) \cup \{v\}$, thus, we have $\Gamma_{G'}(v) \cup \{v\} = \{u \in V \mid \text{dist}_G(u, v) < \infty\}$. Let $v' \in \{u \in V \mid \text{dist}_G(u, v) < \infty\}$, then due to property 2, $\Gamma_{G'}(v') \cup \{v'\} \subseteq \{u \in V \mid \text{dist}_G(u, v') < \infty\} = \{u \in V \mid \text{dist}_G(u, v) < \infty\}$, then we have $|\Gamma_{G'}(v') \cup \{v'\}| < \lceil (m/n)^{1/2} \rceil$. Thus, $|S_{v'}^{(r)}| < \lceil (m/n)^{1/2} \rceil$. By property 3 of Lemma 32.4.1, and the condition in line 24, we know $\{u \in V \mid \text{dist}_G(u, v') \leq 2^r\} = S_{v'}^{(r)} = S_{v'}^{(r-1)} = \{u \in V \mid \text{dist}_G(u, v') \leq 2^{r-1}\}$. Thus, $S_{v'}^{(r)} = \{u \in V \mid \text{dist}_G(u, v') < \infty\}$. Thus, $\Gamma_{G'}(v') \cup \{v'\} = \{u \in V \mid \text{dist}_G(u, v') < \infty\} =$

$\{u \in V \mid \text{dist}_G(u, v) < \infty\}$. Thus, $\forall p, q \in \{u \in V \mid \text{dist}_G(u, v) < \infty\}$, we have $(p, q) \in E'$, which means that $\{u \in V \mid \text{dist}_G(u, v) < \infty\}$ is a clique in G'.

Now consider two vertices $u, v \in V$. Suppose $|\Gamma_{G'}(v)| < \lceil (m/n)^{1/2} \rceil - 1$, then we have that $\{p \in V \mid \text{dist}_G(p, v) < \infty\}$ is a clique in G'. Thus, $\forall q \in \{p \in V \mid \text{dist}_G(p, v) < \infty\}$, we have $|\Gamma_{G'}(q)| = |\Gamma_{G'}(v)| < \lceil (m/n)^{1/2} \rceil - 1$. If $|\Gamma_{G'}(u)| \geq \lceil (m/n)^{1/2} \rceil - 1$, then $\text{dist}_{G'}(u, v) = \infty$.

For property 4, by line 25, we have $E \subseteq E'$ and $|E'| \leq |E| + \sum_{v \in V} |S_v^{(r)}| \leq |E| + n \cdot m/n = |E| + m$ where the last inequality follows by the property 4 of Lemma 32.4.1. $\square$

32.4.2 Random Leader Selection

Given an undirected graph $G = (V, E)$, to design a connected component algorithm, a natural way is constantly contracting the vertices in the same component. One way to do the contraction is that we randomly choose some vertices as leaders, then contract non-leader vertices to the neighbor leader vertices.

In this section, we show that if $\forall v \in V$, the number of neighbors of v is large enough, then we can just sample a small number of leaders such that for each non-leader vertex $v \in V$, there is at least one neighbor of v which is chosen as a leader. A more generalized statement is stated in the following lemma.

Lemma 32.4.3. *Let V be a vertex set with n vertices. Let $0 < \gamma \leq n, \delta \in (0, 1)$. For each $v \in V$, let S_v be a subset of $V \setminus \{v\}$ with size at least $\gamma - 1$. Let $l : V \to \{0, 1\}$ be a random hash function such that $\forall v \in V, l(v)$ are i.i.d. Bernoulli random variables, i.e.*

$$l(v) = \begin{cases} 1 & \text{with probability } p; \\ 0 & \text{otherwise.} \end{cases}$$

If $p \geq \min((10 \log(2n/\delta))/\gamma, 1)$, then, with probability at least $1 - \delta$,

1. $\sum_{v \in V} l(v) \leq \frac{3}{2} pn$;
2. $\forall v \in V, \exists u \in S_v \cup \{v\}$ *such that* $l(u) = 1$.

Proof. For a fixed vertex $v \in V$, we have

$$\begin{aligned}
&\Pr\left(\sum_{u\in S_v\cup\{v\}}(\mathbb{E}(l(u))-l(u)) > \frac{1}{2}\sum_{u\in S_v\cup\{v\}}\mathbb{E}(l(u))\right)\\
\leq\ &\exp\left(-\frac{\frac{1}{2}\left(\frac{1}{2}\sum_{u\in S_v\cup\{v\}}\mathbb{E}(l(u))\right)^2}{\sum_{u\in S_v\cup\{v\}}\mathbb{V}(l(u))+\frac{1}{3}\cdot 1\cdot\frac{1}{2}\sum_{u\in S_v\cup\{v\}}\mathbb{E}(l(u))}\right)\\
\leq\ &\exp\left(-\frac{\frac{1}{2}\left(\frac{1}{2}\sum_{u\in S_v\cup\{v\}}\mathbb{E}(l(u))\right)^2}{\sum_{u\in S_v\cup\{v\}}\mathbb{E}(l(u))+\frac{1}{3}\cdot 1\cdot\frac{1}{2}\sum_{u\in S_v\cup\{v\}}\mathbb{E}(l(u))}\right)\\
=\ &\exp\left(-\frac{3}{28}\cdot\sum_{u\in S_v\cup\{v\}}\mathbb{E}(l(u))\right) = \exp\left(-\frac{3}{28}\cdot p\cdot|S_v\cup\{v\}|\right)\leq\frac{\delta}{2n},
\end{aligned}$$

where the first inequality follows by Bernstein inequality and $|l(u) - E(l(u))| \leq 1$, the second inequality follows by $\mathbb{V}(l(u)) \leq \mathbb{E}(l^2(u)) = \mathbb{E}(l(u))$. The last inequality follows by $|S_v \cup \{v\}| \geq \gamma$, and $p \geq \min((10\log(2n/\delta))/\gamma, 1)$. Since $\frac{1}{2}\sum_{u\in S_v\cup\{v\}}\mathbb{E}(l(u)) \geq 1$, with probability at least $1-\delta/(2n)$, $\sum_{u\in S_v\cup\{v\}} l(v) \geq 1$. By taking union bound over all S_v, with probability at least $1-\delta/2$, $\forall v \in V, \exists u \in S_v \cup \{v\}, l(u) = 1$.

Similarly, we have

$$\begin{aligned}
&\Pr\left(\sum_{u\in V}(l(u)-\mathbb{E}(l(u)))>\frac{1}{2}\sum_{u\in V}\mathbb{E}(l(u))\right)\\
\leq\ &\exp\left(-\frac{\frac{1}{2}\left(\frac{1}{2}\sum_{u\in V}\mathbb{E}(l(u))\right)^2}{\sum_{u\in V}\mathbb{V}(l(u))+\frac{1}{3}\cdot 1\cdot\frac{1}{2}\sum_{u\in V}\mathbb{E}(l(u))}\right)\\
\leq\ &\exp\left(-\frac{\frac{1}{2}\left(\frac{1}{2}\sum_{u\in V}\mathbb{E}(l(u))\right)^2}{\sum_{u\in V}\mathbb{E}(l(u))+\frac{1}{3}\cdot 1\cdot\frac{1}{2}\sum_{u\in V}\mathbb{E}(l(u))}\right)\\
=\ &\exp\left(-\frac{3}{28}\cdot\sum_{u\in V}\mathbb{E}(l(u))\right)\\
=\ &\exp\left(-\frac{3}{28}\cdot p\cdot|V|\right)\leq\frac{\delta}{2n}\leq\frac{\delta}{2}.
\end{aligned}$$

Since $\sum_{u\in V}\mathbb{E}(l(u))=p\cdot n$, with probability at least $1-\delta/2$, $\sum_{u\in V}l(u)\leq 1.5pn$.

By taking union bound, with probability at least $1-\delta, \sum_{u\in V}l(u)\leq 1.5pn$ and $\forall v\in V, \exists u\in S_v\cup\{v\}, l(u)=1$. □

If the number of neighbors of each vertex is not large, then we can still have a constant fraction of vertices which can contract to a leader.

Lemma 32.4.4. *Let V be a vertex set with n vertices. Let S_v be a subset of $V\setminus\{v\}$ with size at least 1. Let $l: V\to\{0,1\}$ be a random hash function such that $\forall v\in V, l(v)$ are i.i.d. Bernoulli random variables, i.e.*

$$l(v)=\begin{cases}1 & \text{with probability } \frac{1}{2};\\ 0 & \text{otherwise.}\end{cases}$$

Let $L=\{v\in V\mid l(v)=1\}\cup\{v\in V\mid\forall u\in S_v\cup\{v\}, l(u)=0\}$. $\mathbb{E}(L)\leq 0.75n$.

Proof. For $v\in V$, $\Pr(l(v)=1)=\frac{1}{2}$. Let $u\in S_v$. Then $\Pr(\forall x\in S_v\cup\{v\}, l(x)=0)\leq\Pr(l(v)=0, l(u)=0)=0.25$. $\mathbb{E}(|L|)=\sum_{v\in V}\Pr(v\in L)\leq 0.75n$. □

32.4.3 Tree Contraction Operation

In this section, we introduce the contraction operation. Firstly, let us introduce the concept of the parent pointers which can define a rooted forest.

Definition 32.4.2. Given a set of vertices V, let $\mathrm{par} : V \to V$ satisfy that $\forall v \in V, \exists i > 0$ such that $\mathrm{par}^{(i)}(v) = \mathrm{par}^{(i+1)}(v)$, where $\forall v \in V, j > 0, \mathrm{par}^{(j)}(v)$ is defined as $\mathrm{part}(\mathrm{par}^{(j-1)}(v))$, and $\mathrm{par}^{(0)}(v) = v$. Then, we call such par a set of parent pointers on V. For $v \in V$, if $\mathrm{part}(v) = v$, then we say v is a root of par . par can have more than one root. The depth of $v \in V$, $\mathrm{dep}_{\mathrm{par}}(v)$ is the smallest $i \in \mathbb{Z}_{\geq 0}$ such that $\mathrm{par}^{(i)}(v) = \mathrm{par}^{(i+1)}(v)$. The root of $v \in V$, $\mathrm{par}^{(\infty)}(v)$ is defined as $\mathrm{par}^{(\mathrm{dep}_{\mathrm{par}}(v))}(v)$. The depth of par, $\mathrm{dep}(\mathrm{par})$ is defined as $\max_{v \in V} \mathrm{dep}_{\mathrm{par}}(v)$.

It is easy to see that a set of parent pointers par on V formed a rooted forest on V. For a vertex $v \in V$, if $\mathrm{part}(v) = v$, then v is a root in the forest. Otherwise $\mathrm{part}(v)$ is the parent of v in the forest.

In the following, we define the union operation of several sets of parent pointers.

Definition 32.4.3. Let $\mathrm{par}_1 : V_1 \to V_1, \mathrm{par}_2 : V_2 \to V_2, \cdots, \mathrm{par}_k : V_k \to V_k$ be k sets of parent pointers on vertex sets $V_1, V_2, \cdots, V_k$ respectively, where $\forall i \neq j \in [k], V_i \cap V_j = \emptyset$. Then $\mathrm{par} = \mathrm{par}_1 \cup \mathrm{par}_2 \cup \cdots \cup \mathrm{par}_k$ is a set of parent pointers on the vertex set $V_1 \cup V_2 \cup \cdots \cup V_k$ such that $\forall i \in [k], v \in V_i, \mathrm{part}(v) = \mathrm{par}_i(v)$.

Now we focus on the parent pointers which can preserve the connectivity of the graph.

Definition 32.4.4. Given a graph $G = (V, E)$ and a set of parent pointers par on V, if $\forall v \in V$, we have $\mathrm{dist}_G(v, \mathrm{part}(v)) < \infty$, then par is *compatible* with G.

It is easy to show the following fact:

Fact 32.4.5. *Given a graph $G = (V, E)$ and a set of parent pointers* par *which is compatible with G, then $\forall u, v \in V$ with $\mathrm{par}^{(\infty)}(u) = \mathrm{par}^{(\infty)}(v)$, we have $\mathrm{dist}_G(u, v) < \infty$.*

Proof. By the definition of compatible, $\forall v \in V, \mathrm{dist}_G(v, \mathrm{part}(v)) < \infty$. By induction, $\forall l \in \mathbb{Z}_{>0}, v \in V$, we have $\mathrm{dist}_G(v, \mathrm{par}^{(l)}(v)) \leq \mathrm{dist}_G(v, \mathrm{par}^{(l-1)}(v)) + \mathrm{dist}_G(\mathrm{par}^{(l-1)}(v), \mathrm{par}^{(l)}(v)) < \infty$. Thus, for any pair of vertices $u, v \in V$, if $\mathrm{par}^{(\infty)}(u) = \mathrm{par}^{(\infty)}(v)$, then $\mathrm{dist}_G(u, v) \leq \mathrm{dist}_G(u, \mathrm{par}^{(\infty)}(u)) + \mathrm{dist}_G(\mathrm{par}^{(\infty)}(v), v) < \infty$. □

In this section, we describe a procedure which can be used to reduce the number of vertices. The input of the procedure is an undirected graph $G = (V, E)$ and a set of parent pointers $\mathrm{par} : V \to V$, where par is compatible with G. The output of the procedure will be the root of each vertex in V and an undirected graph $G' = (V', E')$ which satisfies $V' = \{v \in V \mid \mathrm{part}(v) = v\}, E' = \{(u, v) \in V' \times V' \mid u \neq v, \exists (p, q) \in E, \mathrm{par}^{(\infty)}(p) = u, \mathrm{par}^{(\infty)}(q) = v\}$. Notice that V' only contains all the roots in the forest induced by par, and $|E'| \leq |E|$.

Lemma 32.4.6. *Let $G = (V, E)$ be an undirected graph, $\mathrm{par} : V \to V$ be a set of parent pointers (See Definition 32.4.2). Then* TreeContraction(G, par) *(See Algorithm 32.2) will output $(G', g^{(r)})$ with $r \leq \lceil \log \mathrm{dep}(\mathrm{par}) \rceil$ satisfies the following properties:*

1. $\forall v \in V$, $g^{(r)}(v) = \mathrm{par}^{(\infty)}(v)$.
2. $V' = \{v \in V \mid \mathrm{part}(v) = v\}$.
3. $E' = \{(u, v) \in V' \times V' \mid u \neq v, \exists (p, q) \in E, \mathrm{par}^{(\infty)}(p) = u, \mathrm{par}^{(\infty)}(q) = v\}$.

Proof. One crucial observation is the following claim.

Algorithm 32.2 Tree Contraction Operation

1: **procedure** TREECONTRACTION($G = (V, E)$, par : $V \to V$) ▷ Lemma 32.4.6, Corollary 32.4.8
2: ▷ Output: $G' = (V', E')$, $\text{par}^{(\infty)}(v)$ for all $v \in V$
3: Initially, for each $v \in V$ let $g^{(0)}(v) \leftarrow \text{part}(v)$. Let $V' = \emptyset, E' = \emptyset$.
4: $l \leftarrow 0$.
5: **for** $\exists v \in V, \text{part}(g^{(l)}(v)) \neq g^{(l)}(v)$ **do**
6: $l \leftarrow l + 1$.
7: For each $v \in V$, compute $g^{(l)}(v) = g^{(l-1)}(g^{(l-1)}(v))$. ▷ $g^{(l)}$ is $\text{par}^{(2^l)}$
8: **end for**
9: $r \leftarrow l$. ▷ r is the number of iterations, and is used in the analysis.
10: For $v \in V$, if $\text{part}(v) = v$, let $V' \leftarrow V' \cup \{v\}$.
11: For $(u, v) \in E$, if $g^{(r)}(u) \neq g^{(r)}(v)$, let $E' \leftarrow E' \cup \{(g^{(r)}(u), g^{(r)}(v))\}$.
▷ $\forall v \in V$, contract v to $\text{par}^{(\infty)}(v)$
12: **return** $g^{(r)}(v)$ as $\text{par}^{(\infty)}(v)$ for all $v \in V$, and $G' = (V', E')$
13: **end procedure**

Claim 32.4.7. *$\forall l \in \{0, 1, \cdots, r\}, v \in V$, we have $g^{(l)}(v) = \text{par}^{(2^l)}(v)$.*

Proof. The proof is by induction. When $l = 0$, $\forall v \in V, g^{(0)}(v) = \text{part}(v) = \text{par}^{(1)}(v)$, the claim is true. Suppose for $l-1$, we have $\forall v \in V, g^{(l-1)}(v) = \text{par}^{(2^{l-1})}(v)$, then $\forall v \in V, g^{(l)}(v) = g^{(l-1)}(g^{(l-1)}(v)) = \text{par}^{(2^{l-1})}(\text{par}^{(2^{l-1})}(v)) = \text{par}^{(2^l)}(v)$. So the claim is true. □

If $r > \lceil \log \text{dep}(\text{par}) \rceil$, then $r - 1 \geq \lceil \log \text{dep}(\text{par}) \rceil$. Due to claim 32.4.7, we have $\forall v \in V, g^{(r-1)}(v) = \text{par}^{(2^{r-1})}(v) = \text{par}^{(\infty)}(v)$. Due to the condition in line 5, the loop will stop when $l \leq r - 1$ which leads to a contradiction to line 9. Thus, at the end of the algorithm, r should be at most $\lceil \log \text{dep}(\text{par}) \rceil$.

Since we have $\forall v \in V, \text{part}(g^{(r)}(v)) = g^{(r)}(v)$ at the end of the Algorithm 32.2, $\forall v \in V, g^{(r)}(v)$ must be $\text{par}^{(\infty)}(v)$. Then due to line 10 and line 11, we have $V' = \{v \in V \mid \text{part}(v) = v\}, E' = \{(u, v) \in V' \times V' \mid u \neq v, \exists (p, q) \in E, \text{par}^{(\infty)}(p) = u, \text{par}^{(\infty)}(q) = v\}$. □

Definition 32.4.5. Let $G = (V, E)$ be an undirected graph, $\mathrm{par} : V \to V$ be a set of parent pointers (See Definition 32.4.2). Then the number of iteration of TreeContraction(G, par) is defined as the value of r at the end of the procedure.

Corollary 32.4.8 (Preserved connectivity and diameter)**.** *Let $G = (V, E)$ be an undirected graph, $\mathrm{par} : V \to V$ be a set of parent pointers (See Definition 32.4.2) which is compatible (See Definition 32.4.4) with G. Then at the end of the Algorithm 32.2, $r \le \lceil \log \mathrm{dep}(\mathrm{par}) \rceil$ and the output $(G', g^{(r)})$ will satisfy the following properties:*

1. $\mathrm{diam}(G') \le \mathrm{diam}(G)$.
2. $\forall u, v \in V, \mathrm{dist}_G(u, v) < \infty \Rightarrow \mathrm{dist}_{G'}(\mathrm{par}^{(\infty)}(u), \mathrm{par}^{(\infty)}(v)) < \infty$.
3. $\forall u, v \in V, \mathrm{dist}_G(u, v) < \infty \Leftarrow \mathrm{dist}_{G'}(\mathrm{par}^{(\infty)}(u), \mathrm{par}^{(\infty)}(v)) < \infty$.

Proof. By Lemma 32.4.6, we have $r \le \lceil \log \mathrm{dep}(\mathrm{par}) \rceil$, $V' = \{v \in V \mid \mathrm{part}(v) = v\}$ and $E' = \{(u, v) \in V' \times V' \mid u \ne v, \exists (p, q) \in E, \mathrm{par}^{(\infty)}(p) = u, \mathrm{par}^{(\infty)}(q) = v\}$.

For any two vertices $u, v \in V$ which are in the same connected component in G, then there should be a path $u = u_1 \to u_2 \to \cdots \to u_p = v$ in graph G. So $\forall i \in [p-1], (u_i, u_{i+1}) \in E$ which means that either $\mathrm{par}^{(\infty)}(u_i) = \mathrm{par}^{(\infty)}(u_{i+1})$ or $(\mathrm{par}^{(\infty)}(u_i), \mathrm{par}^{(\infty)}(u_{i+1})) \in E'$. Thus, $\mathrm{par}^{(\infty)}(u_1) \to \mathrm{par}^{(\infty)}(u_2) \to \cdots \to \mathrm{par}^{(\infty)}(u_p)$ is a valid path in G', and the length of this path in G' is at most p. Thus, the properties 1 and 2 are true.

For any two vertices $u, v \in V$ which are not in the same connected component in G, but there is a path $\mathrm{par}^{(\infty)}(u) = u'_1 \to u'_2 \to \cdots \to u'_p = \mathrm{par}^{(\infty)}(v)$ in G', then it means that there exists vertices $u_{1,1}, u_{1,2}, u_{2,1}, u_{2,2}, \cdots, u_{p,1}, u_{p,2} \in V$ which satisfies

(a) $\forall i \in [p-1], (u_{i,2}, u_{i+1,1}) \in E, \mathrm{par}^{(\infty)}(u_{i,2}) = u'_i, \mathrm{par}^{(\infty)}(u_{i+1,1}) = u'_{i+1}$.

(b) $u_{1,1} = u, u_{p,2} = v$.

(c) $\forall i \in [p], \mathrm{par}^{(\infty)}(u_{i,1}) = \mathrm{par}^{(\infty)}(u_{i,2})$. By Fact 32.4.5, we have $\mathrm{dist}_G(u_{i,1}, u_{i,2}) < \infty$.

Thus, there exists a path from u to v. This contradicts to that u, v are not in the same connected component. Therefore, property 3 is also true. □

32.4.4 Connectivity Algorithm

In this section, we described a batch algorithm for graph connectivity/connected components problem. The input is an undirected graph $G = (V, E)$, a space/rounds trade-off parameter m, and the rounds parameter $r \leq |V|$. The output is a function $\text{col} : V \to V$ such that $\forall u, v \in V, \text{dist}_G(u, v) < \infty \Leftrightarrow \text{col}(u) = \text{col}(v)$.

The algorithm is described in Algorithm 32.3. The following theorem shows the correctness of Algorithm 32.3.

Theorem 32.4.9 (Correctness of Algorithm 32.3)**.** *Let $G = (V, E)$ be an undirected graph, $m \geq 4|V|$, and $r \leq |V|$ be the rounds parameter. If $\textsc{Connectivity}(G, m, r)$ (Algorithm 32.3) does not output FAIL, then $\forall u, v \in V$, we have $\text{dist}_G(u, v) < \infty \Leftrightarrow \text{col}(u) = \text{col}(v)$.*

Proof. Firstly, we show that the input of line 18 is valid.

Claim 32.4.10. *$\forall i \in [r]$, par_i is a set of parent pointers on V_i'', (See Definition 32.4.2) and is compatible (See Definition 32.4.4) with G_i''.*

Proof. $\forall v \in V_i''$, if $v \in L_i$, then $\text{par}_i(v) = v$. For $v \in V_i'' \setminus L_i$, due to property 3 of Lemma 32.4.2, we have $\text{par}_i(v) \in V_i''$. Since $\text{par}_i(v) \in L_i$, we have $\text{par}_i(\text{par}_i(v)) = \text{par}_i(v)$. Thus, $\text{par}_i : V_i'' \to V_i''$ is a set of parent pointers on V_i''. Due to property 2 of Lemma 32.4.2 and $\text{dist}_{G_i'}(\text{par}_i(v), v) < \infty$, we know that $\text{dist}_{G_{i-1}}(\text{par}_i(v), v) < \infty$. Thus, $\text{dist}_{G_i''}(\text{par}_i(v), v) < \infty$. It implies that par_i is compatible with G_i''. □

The following claim shows that the number of the remaining vertices cannot increase after each round.

Claim 32.4.11. *If* CONNECTIVITY(G, m, r) *does not output FAIL, then* $\forall i \in [r], V_i \subseteq V_i'' \subseteq V_i' = V_{i-1}$.

Proof. Let $i \in [r]$. Due to Claim 32.4.10, the input of line 18 is valid. Then, we can apply property 2 of Lemma 32.4.6 to get $V_i \subseteq V_i''$. By the construction of V_i'' we have $V_i'' \subseteq V_i'$. Since the procedure NEIGHBORINCREMENT(m, G_{i-1}) (Algorithm 32.1) does not change the vertex set, we have $V_i' = V_{i-1}$. □

Now, we show that $\forall u, v \in V_i, \text{dist}_{G_i}(u, v) < \infty \Leftrightarrow \text{dist}_G(u, v) < \infty$.

Claim 32.4.12. *If* CONNECTIVITY(G, m, r) *does not output* FAIL*, then* $\forall i \in [r], \forall u, v \in V_i$, *we have* $\text{dist}_{G_i}(u, v) < \infty \Leftrightarrow \text{dist}_G(u, v) < \infty$.

Proof. The proof is by induction. Suppose $\forall u, v \in V_{i-1}, \text{dist}_{G_{i-1}}(u, v) < \infty \Leftrightarrow \text{dist}_G(u, v) < \infty$. $\forall w, z \in V_i$, according to Claim 32.4.11, $w, z \in V_i''$. By property 2,3 of Lemma 32.4.8, and property 2 of Lemma 32.4.6, $\text{dist}_{G_i}(w, z) < \infty \Leftrightarrow \text{dist}_{G_i''}(w, z) < \infty$. Due to property 2,3 of Lemma 32.4.2, there is no edge in E_{i-1} between V_i'' and $V_i' \setminus V_i''$. According to Claim 32.4.11, $w, z \in V_{i-1}$. Thus, $\text{dist}_{G_i''}(w, z) < \infty \Leftrightarrow \text{dist}_{G_{i-1}}(w, z) < \infty$. By induction hypothesis, we have $\forall w, z \in V_i, \text{dist}_{G_i}(w, z) < \infty \Leftrightarrow \text{dist}_G(w, z)$. □

The following claim states that once a vertex $v \in V$ is contracted to an another vertex, it will never be operated.

Claim 32.4.13. *Suppose* CONNECTIVITY(G, m, r) *does not output* FAIL. $\forall i \in \{0, 1, \cdots, r\}$, $v \in V$, *we have* $h_i(v) = \text{null} \Leftrightarrow v \in V_i$. *Furthermore,* $\forall v \in V, \exists j \in [r]$ *such that* $h_0(v) = h_1(v) = \cdots = h_{j-1}(v) = \text{null}$ *and* $h_j(v) = h_{j+1}(v) = \cdots = h_r(v) \neq \text{null}, \text{dist}_G(v, h_r(v)) < \infty$.

Proof. When $i = 0$, $\forall v \in V$, $h_0(v) = \text{null}, v \in V_0 = V$. Suppose it is true that $\forall v \in V, h_{i-1}(v) = \text{null} \Leftrightarrow v \in V_{i-1}$. If $v \notin V_i$, according to Claim 32.4.11, there are three cases: $v \in V_i'' \setminus V_i, v \in V_i' \setminus V_i'', v \notin V_{i-1}$. In the first case, due to line 22, $h_i(v) \neq \text{null}$. In the second case, due to line 21, $h_i(v) \neq \text{null}$, In the third case, due to line 23, $h_i(v) \neq \text{null}$. If $h_i(v) = \text{null}$, then $h_i(v)$ cannot be updated by line 21, line 22 or line 23 which implies that $v \in V_{i-1}, v \notin V_i' \setminus V_i'', v \notin V_i'' \setminus V_i$. Thus, $v \in V_i$.

Since the procedure does not FAIL, we have $n_r = 0$ which means that $\forall v \in V, h_r(v) \neq \text{null}$. Notice that by line 23, if $h_{i-1}(v) \neq \text{null}$, then $h_i(v) = h_{i-1}(v)$. Thus, $\forall v \in V, \exists j \in [r]$ such that $h_0(v) = h_1(v) = \cdots = h_{j-1}(v) = \text{null}$ and $h_j(v) = h_{j+1}(v) = \cdots = h_r(v) \neq \text{null}$.

For $v \in V$, if $h_j(v) \neq \text{null}$ and $h_{j-1}(v) = \text{null}$, then $h_j(v)$ can only be updated by 21 or line 22. In both cases, $\text{dist}_{G_{j-1}}(v, h_j(v)) < \infty$. By Claim 32.4.12, we have that $\text{dist}_G(v, h_j(v)) < \infty$. □

In the following, we show that h_r is a rooted tree such that $\text{dist}_G(u, v) < \infty \Leftrightarrow u, v$ have the same root. Due to Claim 32.4.13, if $\textsc{Connectivity}(G, m, r)$ does not output FAIL, then $n_r = 0$ which implies that $\forall v \in V, h_r(v) \neq \text{null}$. Thus, we can define $h_r^{(k)}(v)$ for $k \in \mathbb{Z}_{>0}$ as applying h_r on v k times. $\forall v \in V$, by Claim 32.4.13, let $j \in [r]$ satisfy that $h_j(v) \neq \text{null}$ and $h_{j-1}(v) = \text{null}$. If $h_j(v)$ is updated by line 22, then $h_j(h_j(v)) = \text{null}$. If $h_j(v)$ is updated by line 21, then $h_j(h_j(v)) = h_j(v)$. In both cases, h_j cannot create a cycle. Thus, we can define $h_r^{(\infty)}(v) = h_r^{(k)}(v)$ for some k which satisfies $h_r(h_r^{(k)}(v)) = h_r^{(k)}(v)$.

Claim 32.4.14. *Suppose* $\textsc{Connectivity}(G, m, r)$ *does not output* FAIL. *Then* $\forall u, v \in V$, *we have* $\text{dist}_G(u, v) < \infty \Leftrightarrow h_r^{(\infty)}(u) = h_r^{(\infty)}(v)$.

Proof. Let $u, v \in V$. By Claim 32.4.13, if $h_r^{\infty}(u) = h_r^{\infty}(v)$ we have $\text{dist}_G(u, v) < \infty$.

If $\mathrm{dist}_G(u,v) < \infty$, then let $u' = h_r^{(\infty)}(u), v' = h_r^{(\infty)}(v)$. By Claim 32.4.13, $\mathrm{dist}_G(u',v') \leq \mathrm{dist}_G(u,u') + \mathrm{dist}_G(u,v) + \mathrm{dist}_G(v,v') < \infty$, and we can find $j \in [r]$ such that $h_j(u') \neq \text{null}, h_{j-1}(u') = \text{null}$. Without loss of generality, we can assume $h_{j-1}(v') = \text{null}$ (otherwise we can swap u' and v'). Due to Claim 32.4.13, $u', v' \in V_{i-1}$. Since $h_j(u') = h_r(u') = u'$, $h_j(u')$ can be only updated by line 21, and $u' \in V'_j \setminus V''_j$. Then due to property 3 of Lemma 32.4.6, v' should be in $\Gamma_{G'_i}(u) \cup \{u\}$. Since $h_j(v') = h_r(v') = v'$, we can conclude that $u' = v'$. □

If Connectivity(G,m,r) does not output FAIL, then in line 26, col is exactly $h_r^{(\infty)}$. By Claim 32.4.14, we have $\forall u,v \in V, \mathrm{dist}_G(u,v) < \infty \Leftrightarrow \mathrm{col}(u) = \mathrm{col}(v)$.

□

Now let us consider the number of iterations of Algorithm 32.3 and the success probability.

Definition 32.4.6 (Total iterations)**.** Let $G = (V,E)$ be an undirected graph, $\mathrm{poly}(n) \geq m > 4n$, and $r \leq n$ be the rounds parameter where n is the number of vertices in G. The total number of iterations of Connectivity(G,m,r) (Algorithm 32.3) is defined as $\sum_{i=1}^{r}(k_i + r'_i)$, where k_i denotes the number of iterations (See Definition 32.4.1) of NeighborIncrement(m, G_{i-1}) (see line 9), and r'_i denotes the number of iterations (See Definition 32.4.5) of TreeContraction(G''_i, par_i) (see line 18).

Theorem 32.4.15 (Success probability and total iterations)**.** *Let $G = (V,E)$ be an undirected graph, $\mathrm{poly}(n) \geq m > 4n$, and $r \leq n$ be the rounds parameter where $n = |V|$. Let $c > 0$ be a sufficiently large constant. If $r \geq c \log\log_{m/n}(n)$, then with probability at least 0.98,* Connectivity(G,m,r) *(Algorithm 32.3) will not return* FAIL*. If* Connectivity(G,m,r)

succeeds, let k_i denote the number of iterations (See Definition 32.4.1) of NeighborIncrement(m, G_{i-1}) *(see line 9), and let r'_i denote the number of iterations of (See Definition 32.4.5) of* TreeContraction$(G''_i$ *(see line 18), then*

1. $\forall i \in [r]$, $r'_i = 0$.
2. $\forall i \in [r]$, k_i *is at most* $\lceil \log(\mathrm{diam}(G)) \rceil + 1$.
3. *The number of iterations of line 26 is at most* $\lceil \log r \rceil$.
4. $\sum_{i=1}^{r} k_i \leq O(r \log(\mathrm{diam}(G)))$.

Let $c_1 > 0$ be a sufficiently large constant. If $m \geq c_1 n \log^4 n$, then with probability at least 0.99, $\sum_{i=1}^{r} k_i \leq O(\log(\mathrm{diam}(G)) \log \log_{\mathrm{diam}(G)}(n))$. If $m < c_1 n \log^4 n$, then with probability at least 0.98, $\sum_{i=1}^{r} k_i \leq O(\log(\mathrm{diam}(G)) \log \log_{\mathrm{diam}(G)}(n) + (\log \log(n))^2)$.

Proof. Suppose Connectivity(G, m, r) succeeds. Property 1 follows by $\forall v \in V''_i, \mathrm{par}_i(\mathrm{par}_i(v)) = \mathrm{par}_i(v)$ and Lemma 32.4.6. Property 2 follows by $\mathrm{diam}(G_r) \leq \mathrm{diam}(G''_r) \leq \mathrm{diam}(G'_r) \leq \mathrm{diam}(G_{r-1}) \leq \mathrm{diam}(G''_{r-1}) \leq \mathrm{diam}(G'_{r-1}) \leq \cdots \leq \mathrm{diam}(G_0) = \mathrm{diam}(G)$ and property 1 of Lemma 32.4.2. Property 3 follows by the depth of h_r is at most r and Lemma 32.4.6. Property 4 follows by property 2.

Now let us prove the success probability. Let $i \in [r]$. If $p_i < 0.5$, then we can apply Lemma 32.4.3 on vertex set V''_i, parameter γ_i, and hash function l_i. Notice that the set S_v in the statement of Lemma 32.4.3 is $\Gamma_{G'_i}(v)$ in the algorithm. Notice that $|V''_i| \leq n$. Then in the i^{th} round, if $p_i < 0.5$, then with probability at most $1/(100n^2)$, L_i will be $\{v \in V''_i \mid l_i(v) = 1\}$, and $n_i = |L_i| \leq 1.5 p_i n_{i-1}$. By taking union bound over all $i \in [r]$,

we have that with probability at least 0.99, event $\mathcal{E}$ happens: for all $i \in [r]$, if $p_i < 0.5$, then $n_i \le 1.5 p_i n_{i-1} \le 0.75 n_{i-1}$. Suppose $\mathcal{E}$ happens. For $i \in [r], p_i = 0.5$, if we apply Lemma 32.4.4, then condition on n_{i-1}, we have $\mathbb{E}(n_i) \le 0.75 n_{i-1}$. Thus, we know $\forall i \in [r], \mathbb{E}(n_i) \le 0.75\,\mathbb{E}(n_{i-1}) \le 0.75^i n$.

Next, we discuss the case for $p_0 = 0.5$ and the case for $p_0 < 0.5$ separately.

If $p_0 = 0.5$, then $m \le n \cdot (600 \log n)^4$. By Markov's inequality, when $i^* \ge 4 \log_{4/3}(6000 \log n)$, with probability at least 0.99, $n_{i^*} \le n/(600 \log n)^4$ and thus $p_{i^*} < 0.5$. Condition on this event and $\mathcal{E}$, we have

$$
\begin{aligned}
n_r &\le \frac{\left(\frac{\left(\frac{n_{i^*}^{1.5}}{m^{0.5}}(45\log n + 150)\right)^{1.5}}{m^{0.5}}(45\log n + 150)\right)^{\cdots}}{\cdots} && \text{(Apply } r' = r - i^* \text{ times)} \\
&= \frac{n_{i^*}^{1.5^{r'}}}{m^{1.5^{r'}-1}}(45\log n + 150)^{2\cdot(1.5^{r'}-1)} \\
&= n_{i^*}/(m/n_{i^*})^{1.5^{r'}-1} \cdot (45\log n + 150)^{2\cdot(1.5^{r'}-1)} \\
&\le n/\left(m/\left(n_{i^*}(45\log n + 150)^2\right)\right)^{1.5^{r'}-1} \\
&\le n/\left(m/\left(n_{i^*}(45\log n + 150)^2\right)\right)^{1.5^{r'/2}} \\
&\le n/\left(m/n\right)^{1.5^{r'/2}} \le \frac{1}{2},
\end{aligned}
$$

where the second inequality follows by $n_{i^*} \le n$, the third inequality follows by $r' \ge 5$, the forth inequality follows by $n_{i^*} \le n/(600 \log n)^4$, and the last inequality follows by $r' \ge \frac{2}{\log 1.5} \log\log_{m/n}(2n)$. Since $4n \le m \le n \cdot (600 \log n)^4, \log\log_{m/n} n = \Theta(\log\log n)$. Let $c > 0$ be a sufficiently large constant. Thus, when $r \ge c \log\log_{m/n} n \ge i^* + r' = 4\log(6000\log n)/\log(4/3) + \frac{2}{\log 1.5}\log\log_{m/n}(2n)$, with probability at least 0.98, CONNECTIVITY(G, m, r) will not fail.

Since property 1 of Lemma 32.4.2, we have $k_i \leq O(\log(\min(m/n_{i-1}, \mathrm{diam}(G))))$. Thus,

$$\begin{aligned}
&\sum_{i=1}^{r} k_i = \sum_{i=1}^{i^*} k_i + \sum_{i=i^*+1}^{r} k_i \leq O\left((\log\log n)^2\right) + \sum_{i=i^*+1}^{r} k_i \\
&\leq O\left((\log\log n)^2\right) + \sum_{i:i\geq i^*+1, m/n_{i-1}\leq \mathrm{diam}(G)} k_i + \sum_{i:i\leq r, m/n_{i-1}>\mathrm{diam}(G)} k_i \\
&\leq O\left((\log\log n)^2\right) + O\left(\sum_{i=0}^{\lceil \log_{1.25}\log_2(\mathrm{diam}(G))\rceil} \log(2^{1.25^i})\right) + O\left(\sum_{i=0}^{\lceil \log_{1.25}\log_{\mathrm{diam}(G)}(m)\rceil} \log(\mathrm{diam}(G))\right) \\
&\leq O\left((\log\log n)^2\right) + O(\log(\mathrm{diam}(G))) + O(\log(\mathrm{diam}(G))\log\log_{\mathrm{diam}(G)}(n)) \\
&\leq O(\log(\mathrm{diam}(G))\log\log_{\mathrm{diam}(G)}(n) + (\log\log(n))^2),
\end{aligned}$$

where the first inequality follows by $i^* = O(\log\log n)$ and $\forall i \leq [i^*], m/n_{i-1} \leq \mathrm{poly}(\log n)$, the third inequality follows by $m/n_{i+1} \geq (m/n_i)^{1.5}/(45\log n + 150) \geq (m/n_i)^{1.25}$.

If $m > n \cdot (600\log n)^4$, then $\forall i \in \{0\} \cup [r-1]$, we have $p_i < 0.5$. Since $\mathcal{E}$ happens. We have:

$$\begin{aligned}
n_r &\leq \frac{\left(\frac{\left(\frac{n^{1.5}}{m^{0.5}}(45\log n+150)\right)^{1.5}}{m^{0.5}}(45\log n+150)\right)^{\cdots}}{\cdots} \qquad \text{(Apply } r \text{ times)} \\
&= \frac{n^{1.5^r}}{m^{1.5^r-1}}(45\log n+150)^{2\cdot(1.5^r-1)} \\
&= n/(m/n)^{1.5^r-1}\cdot(45\log n+150)^{2\cdot(1.5^r-1)} \\
&= n/\left(m/\left(n(45\log n+150)^2\right)\right)^{1.5^r-1} \\
&\leq n/\left(m/\left(n(45\log n+150)^2\right)\right)^{1.5^{r/2}} \\
&\leq n/\left(m/\left(n(200\log n)^2\right)\right)^{1.5^{r/2}} \\
&\leq \frac{1}{2},
\end{aligned}$$

where the second inequality follows by $r \geq 5$, the third inequality follows by $45 \log n + 150 \leq 200 \log n$, and the last inequality follows by

$$r \geq c \log \log_{m/n} n \geq 2 \log_{1.5} \log_{(m/n)^{1/2}} 2n \geq 2 \log_{1.5} \log_{m/(n(200 \log n)^2)} 2n$$

for a sufficiently large constant $c > 0$.

By property 1 of Lemma 32.4.2, we have $k_i \leq O(\log(\min(m/n_{i-1}, \operatorname{diam}(G))))$. Thus,

$$\begin{aligned}
\sum_{i=1}^{r} k_i &\leq \sum_{m/n_{i-1} \leq \operatorname{diam}(G)} k_i + \sum_{m/n_{i-1} > \operatorname{diam}(G)} k_i \\
&\leq O\left(\sum_{i=0}^{\lceil \log_{1.25} \log_2(\operatorname{diam}(G)) \rceil} \log(2^{1.25^i}) \right) + O\left(\sum_{i=0}^{\lceil \log_{1.25} \log_{\operatorname{diam}(G)}(m) \rceil} \log(\operatorname{diam}(G)) \right) \\
&\leq O(\log(\operatorname{diam}(G))) + O(\log(\operatorname{diam}(G)) \log \log_{\operatorname{diam}(G)}(n)),
\end{aligned}$$

where the first inequality follows by $m/n_{i+1} \geq (m/n_i)^{1.5}/(45 \log n + 150) \geq (m/n_i)^{1.25}$.

Since n_r is an integer, n_r must be 0 when $n_r \leq 1/2$. Let $c > 0$ be a sufficiently large constant. For all $m \geq 4n$, if $r \geq c \log \log_{m/n} n$ then CONNECTIVITY(G, m, r) will succeed with probability at least 0.98.

□

32.5 Spanning Forest

32.5.1 Local Shortest Path Tree

In this section, we introduce an important procedure which will be used in the spanning tree algorithm. Roughly speaking, our procedure can merge several local shortest path trees into a larger local shortest path tree. Before we describe the details of the procedure, let us look at some concepts.

Definition 32.5.1 (Local shortest path tree (LSPT))**.** Let V' be a set of vertices, v be a vertex in V', and $\mathrm{par} : V' \to V'$ be a set of parent pointers (See Definition 32.4.2) on V' which satisfies that v is the only root of par . Let $T = (V', \mathrm{par})$. Given an undirected graph $G = (V, E)$, if $V' \subseteq V$ and $\forall u \in V' \setminus \{v\}, (u, \mathrm{part}(u)) \in E, \mathrm{dep}_{\mathrm{par}}(u) = \mathrm{dist}_G(u, v)$, then we say T is a local shortest path tree (LSPT) in G, and T has root v. The vertex set (V' in the above) in T is denoted as V_T. The set of parent pointers (par in the above) in T is denoted as par_T. For short, $\mathrm{dep}_{\mathrm{par}_T}$ is denoted as dep_T, and $\mathrm{dep}(\mathrm{part}(T))$ is denoted as $\mathrm{dep}(T)$.

Definition 32.5.2. Given an undirected graph $G = (V, E)$, a vertex $v \in V$, and $s \in \mathbb{Z}_{\geq 0}$, we define the ball centered at v with radius s as the set $B_{G,s}(v) = \{u \in V \mid \mathrm{dist}_G(u, v) \leq s\}$.

If in the context graph G is clear, then we use $B_s(v)$ to denote $B_{G,s}(v)$.

Definition 32.5.3 (Local complete shortest path tree (LCSPT))**.** Given an undirected graph $G = (V, E)$, $s \in \mathbb{Z}_{\geq 0}$ and a local shortest path tree $T = (V_T, \mathrm{par}_T)$ in G where T has root $v \in V$. If $V_T = B_{G,s}(v)$, then we call T a local complete shortest path tree (LCSPT) in G. The root of T is v. The radius of T is s.

Let $\widetilde{T} = (V_{\widetilde{T}}, \mathrm{par}_{\widetilde{T}})$ with radius $s_1 \in \mathbb{Z}_{\geq 0}$ and root v be a local complete shortest path tree in some graph $G = (V, E)$. For $s_2 \in \mathbb{Z}_{\geq 0}$, if for every $u \in V_{\widetilde{T}}$, we have a local complete

shortest path tree $T(u) = (V_{T(u)}, \mathrm{par}_{T(u)})$ with root u and radius s_2, then we can compute a larger local complete shortest path tree $\widehat{T}$ with root v and radius $s_1 + s_2$. The procedure is described in Algorithm 32.4.

Lemma 32.5.1. *Let $G = (V, E)$ be an undirected graph, $s_1, s_2 \in \mathbb{Z}_{\geq 0}$, and $v \in V$. Let $\widetilde{T} = (V_{\widetilde{T}}, \mathrm{par}_{\widetilde{T}})$ with root v and radius s_1 be a local complete shortest path tree in G, and $\mathrm{dep}_{\widetilde{T}} : V_{\widetilde{T}} \to \mathbb{Z}_{\geq 0}$ be the depth of every vertex in $\widetilde{T}$. $\forall u \in V_{\widetilde{T}}$, let $T(u)$ with root u and radius s_2 be a local complete shortest path tree in G, and $\mathrm{dep}_{T(u)} : V_{T(u)} \to \mathbb{Z}_{\geq 0}$ be the depth of every vertex in $T(u)$. Let $(\widehat{T} = (V_{\widehat{T}}, \mathrm{par}_{\widehat{T}}), \mathrm{dep}_{\widehat{T}}) = \textsc{TreeExpansion}(\widetilde{T}, \mathrm{dep}_{\widetilde{T}}, \{T(u) \mid u \in V_{\widetilde{T}}\}, \{\mathrm{dep}_{T(u)} \mid u \in V_{\widetilde{T}}\})$ (Algorithm 32.4), then $\widehat{T}$ is a local complete shortest path tree with root v and radius $s_1 + s_2$ in G. In addition, $\mathrm{dep}_{\widehat{T}}$ records the depth of every vertex in $\widehat{T}$.*

Proof. If $x \in B_{s_1+s_2}(v)$, then there must exist $u \in V$ such that $\mathrm{dist}_G(v, u) \leq s_1$ and $\mathrm{dist}_G(u, x) \leq s_2$. Thus, $V_{\widehat{T}} = \bigcup_{u \in \widetilde{T}} V_{T(u)} = \bigcup_{u \in B_{s_1}(v)} B_{s_2}(u) = B_{s_1+s_2}(v)$.

Now we want to prove that $\mathrm{par}_{\widehat{T}} : V_{\widehat{T}} \to V_{\widehat{T}}$ also satisfies the condition that $\widehat{T}$ is a local shortest path tree. We can prove it by induction. If $\mathrm{dist}_G(u, v) = 0$, then it means $u = v$. In this case, $\mathrm{par}_{\widehat{T}}(u) = \mathrm{par}_{\widetilde{T}}(u) = v$, and $h(u) = \mathrm{dep}_{\widetilde{T}}(u) = 0$. Let $s \in [s_1 + s_2]$. Suppose $\forall x \in B_{s-1}(v)$, we have $h(x) = \mathrm{dep}_{\widehat{T}}(x) = \mathrm{dist}_G(x, v)$. If $B_s(v) = B_{s-1}(v)$, then we are already done. Otherwise, let x be the vertex which has $\mathrm{dist}_G(x, v) = s$. If $x \in B_{s_1}(v)$, then $h(x) = \mathrm{dep}_{\widetilde{T}}(x) = \mathrm{dist}_G(x, v)$. Additionally, we have $\mathrm{par}_{\widehat{T}}(x) = \mathrm{par}_{\widetilde{T}}(x)$. Therefore, $\mathrm{dep}_{\widehat{T}}(x) = \mathrm{dep}_{\widehat{T}}(\mathrm{par}_{\widetilde{T}}(x)) + 1 = \mathrm{dist}_G(v, \mathrm{par}_{\widetilde{T}}(x)) + 1 = \mathrm{dist}_G(v, x)$. If $x \in B_{s_2}(v) \setminus B_{s_1}(v)$, then $h(x) = \min_{u:\mathrm{dist}_G(v,u) \leq s_1, \mathrm{dist}_G(u,x) \leq s_2} \mathrm{dep}_{\widetilde{T}}(u) + \mathrm{dep}_{T(u)}(x) = \min_{u:\mathrm{dist}_G(v,u) \leq s_1, \mathrm{dist}_G(u,x) \leq s_2} \mathrm{dist}_G(v, u) + \mathrm{dist}_G(u, x) = \mathrm{dist}_G(v, x) = s$. And we have $\mathrm{dist}_G(v, x) = \mathrm{dist}_G(v, u_x) + \mathrm{dist}_G(u_x, x)$. Notice that $\mathrm{dist}_G(v, \mathrm{par}_{T(u_x)}(x)) = \mathrm{dist}_G(v, u_x) + \mathrm{dist}_G(u_x, \mathrm{par}_{T(u_x)}(x)) = \mathrm{dist}_G(v, x) - 1 = s - 1$.

Thus,

$$\mathrm{dep}_{\widehat{T}}(x) = \mathrm{dep}_{\widehat{T}}(\mathrm{par}_{\widehat{T}}(x)) + 1 = \mathrm{dep}_{\widehat{T}}(\mathrm{par}_{T(u_x)}(x)) + 1 = s.$$

To conclude, $\widehat{T}$ is a local complete shortest path tree with root v and radius $s_1 + s_2$ in G. In addition, $\mathrm{dep}_{\widehat{T}}$ records the depth of every vertex in $\widehat{T}$. $\square$

32.5.2 Multiple Local Shortest Path Trees

In this section, we show a procedure which is a generalization of neighbor increment procedure shown in Section 32.4.1. The input of the procedure is an undirected graph $G = (V, E)$ and a parameter m which is larger than $|V| = n$. The output will be n local shortest path trees (See Definition 32.5.1) such that $\forall v \in V$, there is a shortest path tree with root v. Furthermore, the size of each shortest path tree is at least $\left\lceil (m/|V|)^{1/4} \right\rceil$ and at most $\left\lceil (m/|V|)^{1/2} \right\rceil$. The algorithm is described in Algorithm 32.6. The high level idea is that we firstly use doubling technique and the algorithm described in Section 32.5.1 to get local complete shortest path trees rooted at every vertex with multiple radius, and then use these LCSPTs to find large enough local shortest path trees rooted at every vertex. The doubling algorithm is described in Algorithm 32.5.

Definition 32.5.4. Given a graph $G = (V, E)$ and a parameter $m \in \mathbb{Z}_{\geq 0}, m \geq |V|$, the number of iterations of MultiRadiusLCSPT(G, m) (Algorithm 32.5) is the value of r at the end of the procedure.

Lemma 32.5.2. *Let $G = (V, E)$ be an undirected graph, and m be a parameter which is at least $|V|$. Let $(r, \{T_i(v) \mid i \in \{0\} \cup [r], v \in V\}, \{\text{dep}_{T_i(v)} \mid i \in \{0\} \cup [r], v \in V, T_i(v) \neq \text{null}\}) =$ MultiRadiusLCSPT(G, m) (Algorithm 32.5). We have following properties.*

1. *$\forall i \in \{0\} \cup [r], v \in V$, if $T_i(v) \neq \text{null}$, then $T_i(v)$ is a LCSPT (See Definition 32.5.3) with root v and radius 2^i in G. Furthermore, $\text{dep}_{T_i(v)}$ records the depth of every vertex in $T_i(v)$.*

2. *$\forall i \in \{0\} \cup [r], v \in V$, $|B_{G,2^i}(v)| \geq \lceil (m/n)^{1/4} \rceil \Leftrightarrow T_i(v) = \text{null}$.*

3. *For* $v \in V$, *if* $T_r(v) \neq \text{null}$, *then* $V_{T_r(v)} = \{u \in V \mid \text{dist}_G(u, v) < \infty\}$.

4. *The number of iterations (see Definition 32.5.4)* $r \leq \min(\lceil \log(\text{diam}(G)) \rceil, \lceil \log(m/n) \rceil) + 1$.

Proof. For property 1, we can prove it by induction. If $i = 0$, the property holds by line 4, line 5 and line 7. Now suppose $\forall v \in V$, if $T_{i-1}(v)$ is not null, then $T_{i-1}(v)$ is a LCSPT with root v and radius 2^{i-1} in G, and $\text{dep}_{T_i(v)}$ records the depth of every vertex in $T_i(v)$. For $v \in V$, notice that the only place that will make $T_i(v)$ not null is line 15, and if the procedure run line 15, any of $T_{i-1}(v)$ and $T_{i-1}(u)$ with $u \in V_{T_{i-1}(v)}$ cannot be null. By Lemma 32.5.1, since the radius of $T_{i-1}(v)$ is 2^{i-1}, and $\forall u \in V_{T_{i-1}(v)}, T_{i-1}(u)$ has radius 2^{i-1}, $T_i(v)$ is a LCSPT with root v and radius 2^i. Furthermore $\text{dep}_{T_i(v)}$ records the depth of every vertex in $T_i(v)$.

For property 2, if $i = 0$, then this property holds by line 4 to line 7. For $i \in [r]$, our proof is by induction. Suppose the property holds for $i - 1$. Now consider $T_i(v)$ for $v \in V$. The only way to make $T_i(v)$ not null is line 15. If the procedure invokes line 15, then any of $T_{i-1}(v)$ and $T_{i-1}(u)$ with $u \in V_{T_{i-1}(v)}$ cannot be null. By property 1 and Lemma 32.5.1, $T_i(v)$ will be a LCSPT with root v and radius 2^i in line 15. If $|B_{G,2^i}(v)| \geq \lceil (m/n)^{1/4} \rceil$, then $T_i(v)$ is set to be null in line 16. Thus, we already got $|B_{G,2^i}(v)| \geq \lceil (m/n)^{1/4} \rceil \Rightarrow T_i(v) = \text{null}$. Now we want to show $|B_{G,2^i}(v)| \geq \lceil (m/n)^{1/4} \rceil \Leftarrow T_i(v) = \text{null}$. If $T_i(v) = \text{null}$, then there are three cases. The first case is that $T_i(v)$ is set at line 12. In this case, $T_{i-1}(v) = \text{null}$ implies $|B_{G,2^i}(v)| \geq |B_{G,2^{i-1}}(v)| \geq \lceil (m/n)^{1/4} \rceil$. The second case is that $T_i(v)$ is set at line 13. In this case, $\exists u \in V_{T_{i-1}(v)} = B_{G,2^{i-1}}(v)$ such that $|B_{G,2^{i-1}}(u)| \geq \lceil (m/n)^{1/4} \rceil$ which implies $|B_{G,2^i}(v)| \geq \lceil (m/n)^{1/4} \rceil$. In the final case, $T_i(v)$ is set at line 16, and thus, $|B_{G,2^i}(v)| \geq \lceil (m/n)^{1/4} \rceil$.

For property 3, if $T_r(v) \neq \text{null}$, then by property 1, we know $V_{T_r(v)} = B_{G,2^r}(v)$. By the condition in line 19, we know $V_{T_r(v)} = V_{T_{r-1}(v)}$ which implies $B_{G,2^r}(v) = B_{G,2^{r-1}}(v)$. Thus, $V_{T_r(v)} = \{u \in V \mid \text{dist}_G(u,v) < \infty\}$.

For property 4, we can prove it by contradiction. If $r > \lceil \log(\text{diam}(G)) \rceil + 1$, then let $i = \lceil \log(\text{diam}(G)) \rceil + 1$. By the condition in line 19, we know there is a vertex $v \in V$ such that $T_i(v) \neq \text{null}$ and $V_{T_i(v)} \neq V_{T_{i-1}(v)}$. It means that $B_{G,2^i}(v) \neq B_{G,2^{i-1}}(v)$, i.e. $\exists u \in V, \text{dist}_G(v,u) > 2^{i-1}$. But this contradicts to $i = \lceil \log(\text{diam}(G)) \rceil + 1$. Similarly, if $r > \lceil \log(m/n) \rceil + 1$, then let $i = \lceil \log(m/n) \rceil + 1$. By the condition in line 19, we know there is a vertex $v \in V$ such that $T_i(v) \neq \text{null}$ and $V_{T_i(v)} \neq V_{T_{i-1}(v)}$. If $2^{i-1} \leq \text{diam}(G)$, then we have $V_{T_i(v)} \neq V_{T_{i-1}(v)}$ which leads to a contradiction. If $2^{i-1} \geq \text{diam}(G)$, then $|B_{G,2^{i-1}}(v)| \geq 2^{i-1} \geq m/n \geq \lceil (m/n)^{1/4} \rceil$ which contradicts to property 2. □

Next, we show how to use Algorithm 32.5 to design an algorithm which can output $|V|$ number of local shortest path trees rooted at every vertex in V. The details of the algorithm is described in Algorithm 32.6, and the guarantees of the algorithm is stated in the following lemma.

Lemma 32.5.3. *Let $G = (V, E)$ be an undirected graph, and m be a parameter which is at least $16|V|$. Let $\left(\{\widetilde{T}(v) \mid v \in V\}, \{\text{dep}_{\widetilde{T}(v)} \mid v \in V\}\right) = \text{MultipleLargeTrees}(G, m)$. (Algorithm 32.6) Then, the output satisfies the following properties.*

1. *$\forall v \in V$, $\widetilde{T}(v)$ is a LSPT (See Definition 32.5.1) with root v, and $\text{dep}_{\widetilde{T}(v)}$ records the depth of every vertex in $\widetilde{T}(v)$.*

2. *$\forall v \in V, u \in V_{\widetilde{T}(v)}, w \in V \setminus V_{\widetilde{T}(v)}$, it satisfies $\text{dist}_G(v,u) \leq \text{dist}_G(v,w)$.*

3. $\forall v \in V$, *either* $|V_{\widetilde{T}(v)}| \geq \lceil (m/n)^{1/4} \rceil$ *or* $V_{\widetilde{T}(v)} = \{u \in V \mid \mathrm{dist}_G(u, v) < \infty\}$.

4. $\forall v \in V$, $|V_{\widetilde{T}(v)}| \leq \lfloor (m/n)^{1/2} \rfloor$.

Proof. Before we prove above properties, we first show some crucial observations.

Claim 32.5.4. $\forall v \in V, i \in \{0\} \cup [r]$, *if* $T_i(v) \neq \text{null}$, *then* $T_i(v)$ *is a LCSPT with root* v *and radius* 2^i *in graph* G. *Furthermore,* $\mathrm{dep}_{T_i(v)} : V_{T_i(v)} \to \mathbb{Z}_{\geq 0}$ *records the depth of every vertex in* $T_i(v)$. *If* $T_i(v) = \text{null}$, *then* $|B_{G,2^i}(v)| \geq \lceil (m/n)^{1/4} \rceil$.

Proof. Follows by property 1 and property 2 of Lemma 32.5.2 directly. □

Claim 32.5.5. *Let* $v \in V$ *be a vertex with* $T_r(v) = \text{null}$. *Then,* $\forall i \in \{0\} \cup [r]$, $\widetilde{T}_i(v)$ *is a* LCSPT *(See Definition 32.5.3) with root* v *and radius* $s_i(v)$ *in* G, *and* $\mathrm{dep}_{\widetilde{T}_i(v)}$ *records the depth of every vertex in* $\widetilde{T}_i(v)$. *Furthermore, we have* $|B_{G,s_i(v)}(v)| < \lceil (m/n)^{1/4} \rceil, |B_{G,s_i(v)+2^{r-i}}(v)| \geq \lceil (m/n)^{1/4} \rceil$.

Proof. Let $v \in V$ be a vertex with $T_r(v) = \text{null}$. When $i = 0$, then due to line 5 and line 6, $\widetilde{T}_0(v)$ is a LCSPT (See Definition 32.5.3) with root v and radius $0 = s_0(v)$ in G. According to property 2 of Lemma 32.5.2, since $T_r(v) = \text{null}$, we know $|B_{G,0+2^r}(v)| \geq \lceil (m/n)^{1/4} \rceil$.

For $i \in [r]$, we prove it by induction. Suppose the claim is true for $i - 1$. By Claim 32.5.4, Lemma 32.5.1 and the condition in line 9, if the procedure executes line 10, then we know $\widetilde{T}_i(v)$ is a LCSPT with root v and radius $s_{i-1}(v) + 2^{r-i}$ in G at the end of the execution of line 10, and $\mathrm{dep}_{\widetilde{T}_i(v)}$ records the depth of every vertex in $\widetilde{T}_i(v)$. If $|V_{\widetilde{T}_i(v)}| < \lceil (m/n)^{1/4} \rceil$, then $|B_{G,s_{i-1}(v)+2^{r-i}}(v)| < \lceil (m/n)^{1/4} \rceil$. The procedure will execute line 11, and thus $s_i(v)$ is the radius of $\widetilde{T}_i(v)$. In addition, since $s_i(v) = s_{i-1}(v) + 2^{r-i}$ and

$|B_{G,s_{i-1}(v)+2^{r-i+1}}(v)| \geq \lceil (m/n)^{1/4} \rceil$, we have $|B_{G,s_i(v)+2^{r-i}}(v)| \geq \lceil (m/n)^{1/4} \rceil$. If at the end of line 10, $|V_{\widetilde{T}_i(v)}| \geq \lceil (m/n)^{1/4} \rceil$, then we know $|B_{G,s_{i-1}(v)+2^{r-i}}(v)| \geq \lceil (m/n)^{1/4} \rceil$. In this case, $\widetilde{T}_i(v), s_i(v)$ and $\text{dep}_{\widetilde{T}_i(v)}$ will be set to be $\widetilde{T}_{i-1}(v), s_{i-1}(v)$ and $\text{dep}_{\widetilde{T}_i(v)}$ respectively, and thus $|B_{G,s_i(v)}(v)| < \lceil (m/n)^{1/4} \rceil$. If the condition in line 9 does not hold, then we know $|B_{G,s_{i-1}(v)+2^{r-i}}(v)| \geq \lceil (m/n)^{1/4} \rceil$ by claim 32.5.4. In this case, $\widetilde{T}_i(v), s_i(v)$ and $\text{dep}_{\widetilde{T}_i(v)}$ will also be set to be $\widetilde{T}_{i-1}(v), s_{i-1}(v)$ and $\text{dep}_{\widetilde{T}_i(v)}$ respectively, and thus $|B_{G,s_i(v)}(v)| < \lceil (m/n)^{1/4} \rceil$. □

Claim 32.5.5 shows that for each vertex $v \in V$, we know $\widetilde{T}_r(v)$ is a LCSPT with root v and radius $s_r(v)$ in G such that $|B_{G,s_r(v)}(v)| < \lceil (m/n)^{1/4} \rceil$ and $|B_{G,s_r(v)+1}(v)| \geq \lceil (m/n)^{1/4} \rceil$.

Now, let us prove property 1 and property 2. For $v \in V$, if $T_r(v) \neq \text{null}$, then $\widetilde{T}(v), \text{dep}_{\widetilde{T}(v)}$ will be set to be $T_r(v), \text{dep}_{T_r(v)}$ respectively. By Claim 32.5.4, the properties holds. Let v be a vertex in V with $T_r(v) = \text{null}$. If $\widetilde{T}(v)$ is assigned at line 22, then by Lemma 32.5.1, we know $\widetilde{T}(v)$ is a LCSPT with root v, and $\text{dep}_{\widetilde{T}(v)}$ records the depth of every vertex in $\widetilde{T}(v)$. Thus, both properties hold. If $\widetilde{T}(v)$ is assigned at line 28, then there are two cases for the vertices in $V_{\widetilde{T}(v)}$:

1. If x is in $V_{\widetilde{T}_r(v)}$, then since Claim 32.5.5 shows $\widetilde{T}_r(v)$ is a LCSPT with root v, it is easy to show $\text{dep}_{\widetilde{T}(v)}(x) = \text{dep}_{\widetilde{T}_r(v)}(x) = \text{dist}_G(v, x)$, and $\text{par}_{\widetilde{T}(v)}(x) = \text{par}_{\widetilde{T}_r(v)}(x) \in E$.

2. if x is in $N(u_v)$ but not in $V_{\widetilde{T}_r(v)}$, then since $\widetilde{T}_r(v)$ is a LCSPT with root v and radius $s_r(v)$, $\text{dist}_G(v, x) \geq s_r(v)+1$. Also notice that $\text{dist}_G(v, x) \leq \text{dist}_G(v, u_v)+\text{dist}_G(u_v, x) \leq s_r(v)+1$. Therefore, $\text{dist}_G(v, x) = s_r(v)+1, \text{dep}_{\widetilde{T}(v)}(x) = \text{dep}_{\widetilde{T}(v)}(u_v)+1 = \text{dist}_G(v, x) = s_r(v) + 1$. Since $x \in N(u_v)$, $(\text{par}_{\widetilde{T}(v)}(x), x) = (u_v, x) \in E$.

Thus, $\widetilde{T}$ is a LSPT with root v, and it proves property 1. Due to above both cases, we know $B_{G,s_r(v)}(v) \subseteq V_{\widetilde{T}}$, and $\forall x \in V_{\widetilde{T}(v)}$, $\mathrm{dist}_G(v,x) \leq s_r(v)+1$. Thus, property 2 holds.

For property 3 and property 4, we have two cases. The first case is when v satisfies $T_r(v) \neq \text{null}$. In this case $\widetilde{T}(v) = T_r(v)$, due to property 3, 4 of Lemma 32.5.2, we have $V_{\widetilde{T}(v)} = \{u \in V \mid \mathrm{dist}_G(v,u) < \infty\}$, and $|V_{\widetilde{T}(v)}| < \lceil (m/n)^{1/4} \rceil \leq \lfloor (m/n)^{1/2} \rfloor$. The second case is $T_r(v) = \text{null}$. In this case, if $\widetilde{T}(v)$ is assigned at line 22, then $V_{\widetilde{T}(v)} = B_{G,s_r(v)+1}(v)$. Then by Claim 32.5.5, we can get $|V_{\widetilde{T}(v)}| \geq \lceil (m/n)^{1/4} \rceil$. Because $|V_{\widetilde{T}_r(v)}| < \lceil (m/n)^{1/4} \rceil$ and $\forall u \in V_{\widetilde{T}_r(v)}, |V_{T_0(u)}| < \lceil (m/n)^{1/4} \rceil$, we know $|V_{\widetilde{T}(v)}| \leq \lfloor (m/n)^{1/2} \rfloor$. If $\widetilde{T}(v)$ is assigned at line 28, then $|V_{\widetilde{T}(v)}| \geq |N(u_v)| \geq \lceil (m/n)^{1/4} \rceil$, and $|V_{\widetilde{T}(v)}| \leq |V_{\widetilde{T}_r(v)}| + |N(u_v)| < 2 \cdot \lceil (m/n)^{1/4} \rceil \leq \lfloor (m/n)^{1/2} \rfloor$. □

Definition 32.5.5. Let graph $G = (V, E)$, and let m be a parameter which is at least $16|V|$. The number of iterations of $\left(\{\widetilde{T}(v) \mid v \in V\}, \{\mathrm{dep}_{\widetilde{T}(v)} \mid v \in V\}\right) = \textsc{MultipleLargeTrees}(G, m)$ (Algorithm 32.6) is defined as the value of r in the procedure.

Lemma 32.5.6 (Number of iterations of Algorithm 32.6)**.** *Let $G = (V, E)$ be an undirected graph, and let m be a parameter which is at least $16|V|$. The number of iterations (see Definition 32.5.5) of $\left(\{\widetilde{T}(v) \mid v \in V\}, \{\mathrm{dep}_{\widetilde{T}(v)} \mid v \in V\}\right) = \textsc{MultipleLargeTrees}(G, m)$ (Algorithm 32.6) is at most $\min(\lceil \log(\mathrm{diam}(G)) \rceil, \lceil \log(m/n) \rceil) + 1$.*

Proof. It follows by property 4 of Lemma 32.5.2 directly. □

32.5.3 Path Generation and Root Changing

In this section, we show a procedure which can output a path from a certain vertex to the root in a rooted tree. Then we show how to use the procedure to change the root of a rooted tree to a certain vertex in the tree. To output the vertex-root path, we have two stages. The first stage is using doubling method to compute the depth and the 2^ith (for all $i \in \{0, 1, \cdots, \log(\text{dep})\}$) ancestor of each vertex. The second stage is using divide-and-conquer technique to split the path into segments, and recursively find the path for each segment. Once we have the procedure to find the vertex-root path, then we can use it to implement root-changing. The idea is very simple, if we want to change the root to a certain vertex, we just need to find the path from that vertex to the root, and reverse the parent pointers of every vertex on the path. The path finding procedure is described in Algorithm 32.8. The root changing procedure is described in Algorithm 32.9.

Definition 32.5.6. Let $\text{par} : V \to V$ be a set of parent pointers (See Definition 32.4.2) on a vertex set V. The number of iterations of FindAncestors(par) is defined as the value of r at the end of the procedure.

Lemma 32.5.7. *Let* $\text{par} : V \to V$ *be a set of parent pointers (See Definition 32.4.2) on a vertex set* V*. Let* $(r, \text{dep}_{\text{par}}, \{g_i \mid i \in \{0\} \cup [r]\}) = \text{FindAncestors}(\text{par})$ *(Algorithm 32.7). Then the number of iterations (see Definition 32.5.6)* r *should be at most* $\lceil \log(\text{dep}(\text{par}) + 1) \rceil$, $\text{dep}_{\text{par}} : V \to \mathbb{Z}_{\geq 0}$ *records the depth of every vertex in* V*, and* $\forall i \in \{0\} \cup [r], v \in V$ $g_i(v) = \text{par}^{(2^i)}(v)$.

Proof. h_l and g_l will satisfies the properties in the following claim.

Claim 32.5.8. $\forall i \in \{0\} \cup [r], v \in V$ $g_i(v) = \text{par}^{(2^i)}(v)$, *and if* $\text{dep}_{\text{par}}(v) \leq 2^i - 1$ *then* $h_i(v) = \text{dep}_{\text{par}}(v)$. *Otherwise* $\text{dep}_{\text{par}}(v) = \text{null}$.

Proof. The proof is by induction. The claim is obviously true when $i = 0$. Suppose the claim is true for $i - 1$. We have $g_i(v) = g_{i-1}(g_{i-1}(v)) = \text{par}^{(2^{i-1})}(\text{par}^{(2^{i-1})}(v)) = \text{par}^{(2^i)}(v)$. If $h_i(v) \neq \text{null}$, then there are two cases. In the first case, we have $h_i(v) = h_{i-1}(v)$. By induction we know $h_i(v) = \text{dep}_{\text{par}}(v)$. In the second case, we have $h_i(v) = h_{i-1}(g_{i-1}(v)) + 2^{i-1} = \text{dep}_{\text{par}}(\text{par}^{(2^{i-1})}(v)) + 2^{i-1}$. Notice that in this case $h_{i-1}(v) = \text{null}$, thus by the induction, $\text{dep}_{\text{par}}(v) \geq 2^{i-1}$. Therefore, $\text{dep}_{\text{par}}(v) = \text{dep}_{\text{par}}(\text{par}^{(2^{i-1})}(v)) + 2^{i-1} = h_i(v)$. If $h_i(v) = \text{null}$, then it means that $h_{i-1}(\text{par}^{(2^{i-1})}(v)) = \text{null}$ which implies that $\text{dep}_{\text{par}}(v) \geq 2^i$. □

Due to the above claim, we know that if $i \geq \lceil \log(\text{dep}_{\text{par}}(v) + 1) \rceil$ then $h_i(v) \neq \text{null}$. Thus, we have $r \leq \lceil \log(\text{dep}(\text{par}) + 1) \rceil$. Since the procedure returns h_r as dep_{par}, the returned dep_{par} is correct. □

Lemma 32.5.9. *Let* $\text{par} : V \to V$ *be a set of parent pointers (See Definition 32.4.2) on a vertex set* V. *Let* q *be a vertex in* V. *Let* $(\text{dep}_{\text{par}}, P, w) = \textsc{FindPath}(\text{par}, q)$ *(Algorithm 32.8). Then* $\text{dep}_{\text{par}} : V \to \mathbb{Z}_{\geq 0}$ *records the depth of every vertex in* V *and* $P \subseteq V$ *is the set of all vertices on the path from* q *to the root of* q, *i.e.* $P = \{v \in V \mid \exists k \geq 0, v = \text{par}^{(k)}(q)\}$. *If* $\text{dep}_{\text{par}}(q) \geq 1$, *then* $w = \text{par}^{(\text{dep}_{\text{par}}(q)-1)}(q)$. *Furthermore,* k *should be at most* $\lceil \log(\text{dep}(\text{par})) \rceil$.

Proof. By Lemma 32.5.7, since $(r, \text{dep}_{\text{par}}, \{g_i \mid i \in \{0\} \cup [r]\}) = \textsc{FindAncestors}(\text{par})$, we know r should be at most $\lceil \log(\text{dep}(\text{par}) + 1) \rceil$, $\text{dep}_{\text{par}} : V \to \mathbb{Z}_{\geq 0}$ records the depth of every vertex in V, and $\forall i \in \{0\} \cup [r], v \in V$ $g_i(v) = \text{par}^{(2^i)}(v)$. Thus $k = \lceil \log(\text{dep}_{\text{par}}(q)) \rceil \leq \lceil \log(\text{dep}(\text{par}) + 1) \rceil$

Now let us prove that P is the vertex set of all the vertices on the path from q to the root of q. We use divide-and-conquer to get P. The following claim shows that S_i is a set of segments which is a partition of the path, and each segment has length at most 2^{k-i}.

Claim 32.5.10. *$\forall i \in \{0\} \cup [k]$, S_i satisfies the following properties:*

1. *$\exists (x, y) \in S_i$ such that $x = q$.*
2. *$\exists (x, y) \in S_i$ such that $y = g_r(q)$.*
3. *$\forall (x, y) \in S_i$, $\mathrm{dep}_{\mathrm{par}}(y) - \mathrm{dep}_{\mathrm{par}}(x) \leq 2^{k-i}$.*
4. *$\forall (x, y) \in S_i$, if $y \neq g_r(q)$, then $\exists (x', y') \in S_i, x' = y$.*
5. *$\forall (x, y) \in S_i$, $\exists j \in \mathbb{Z}_{\geq 0}, \mathrm{par}^{(j)}(x) = y$.*

Proof. Our proof is by induction. According to line 4, all the properties hold when $i = 0$. Suppose all the properties hold for $i - 1$. For property 1, by induction we know there exists $(x, y) \in S_{i-1}$ such that $x = q$. Then by line 8 and line 9, there must be an (x, y') in S_i. For property 2, by induction we know there exists $(x, y) \in S_{i-1}$ such that $y = g_r(q)$. Thus, there must be an (x', y) in S_i. For property 3, if (x, y) is added into S_i by line 9, then $\mathrm{dep}_{\mathrm{par}}(x) - \mathrm{dep}_{\mathrm{par}}(y) \leq 2^{k-i}$. Otherwise, in line 8, we have $\mathrm{dep}_{\mathrm{par}}(x) - \mathrm{dep}_{\mathrm{par}}(g_{k-i}(x)) \leq 2^{k-i}, \mathrm{dep}_{\mathrm{par}}(g_{k-i}(x)) - \mathrm{dep}_{\mathrm{par}}(y) \leq 2^{k-i+1} - 2^{k-i} = 2^{k-i}$. For property 4, if (x, y) is added into S_i by line 9, then by induction there is $(y, y') \in S_{i-1}$, and thus by line 9 and line 8, there must be $(y, y'') \in S_i$. Otherwise, in line 8 will generate two pairs $(x, g_{k-i}(x)), (g_{k-i}(x), y)$. For $(x, g_{k-i}(x))$, the property holds. For $(g_{k-i}(x), y)$, there must be $(y, y') \in S_{i-1}$ and thus there should be $(y, y'') \in S_i$. For property 5, since $g_{k-i}(x) = \mathrm{par}^{(k-i)}(x)$, for all pairs generated by line 8 and line 9, the property holds. □

By Claim 32.5.10, we know

$$S_k = \{(q, \text{part}(q)), (\text{part}(q), \text{par}^{(2)}(q)), (\text{par}^{(2)}(q), \text{par}^{(3)}(q)), \cdots, (\text{par}^{(\text{dep}_{\text{par}}(q)-1)}(q), \text{par}^{(\text{dep}_{\text{par}}(q))}(q))\}.$$

Thus, P is the set of all the vertices on the path from q to the root of q. And $w = \text{par}^{(\text{dep}_{\text{par}}(q)-1)}(q)$ when $\text{dep}_{\text{par}}(q) \geq 1$. □

Lemma 32.5.11. *Let* $\text{par} : V \to V$ *be a set of parent pointers (See Definition 32.4.2) on a vertex set* V*. Let* q *be a vertex in* V*. Let* $\widehat{\text{par}} = \textsc{RootChange}(\text{par}, q)$ *(Algorithm 32.9). Then* $\widehat{\text{par}} : V \to V$ *is still a set of parent pointers (See Definition 32.4.2) on* V*.* $\forall v \in V$*, if* $\text{par}^{(\infty)}(v) = \text{par}^{(\infty)}(q)$ *then* $\widehat{\text{par}}^{(\infty)}(v) = q$*. Otherwise* $\widehat{\text{par}}^{(\infty)}(v) = \text{par}^{(\infty)}(v)$*.* $\forall u \neq v \in V, \text{part}(v) = u \Leftrightarrow$ *either* $\widehat{\text{par}}(v) = u$ *or* $\widehat{\text{par}}(u) = v$*. Furthermore,* $\text{dep}(\widehat{\text{par}}) \leq 2\text{dep}(\text{par})$.

Proof. For a vertex $v \in V$, if $\{u \mid i \in \mathbb{Z}_{\geq 0}, u = \text{par}^{(i)}(v)\} \cap P = \emptyset$, then we have $\forall i \in \mathbb{Z}_{\geq 0}, \text{par}^{(i)}(v) = \widehat{\text{par}}^{(i)}(v)$. According to Lemma 32.5.9, $P = \{u \in V \mid i \in \mathbb{Z}_{\geq 0}, \text{par}^{(i)}(q) = u\}$. Then for all $v \in P \setminus \{q\}$, if $\text{part}(u) = v$ then $\widehat{\text{par}}(v) = u$. Thus, $\forall u \in P$, $\widehat{\text{par}}^{(\infty)}(u) = q$. Let i^* be the smallest number such that $\text{par}^{(i^*)}(v) \in P$. Then $\widehat{\text{par}}^{(i^*)}(v) \in P$. Thus, $\widehat{\text{par}}^{(\infty)}(v) = \widehat{\text{par}}^{(\infty)}(\widehat{\text{par}}^{(i^*)}(v)) = q$. Furthermore, we have $\forall v \in V, \text{dep}(\widehat{\text{par}}) \leq \text{dep}(\text{par}) + \text{dep}_{\text{par}}(q) \leq 2\text{dep}(\text{par})$. □

32.5.4 Spanning Forest Expansion

In this section, we give the definition of spanning forest. If we are given a spanning forest of a contracted graph and spanning trees of each contracted component, then we show a procedure which can merge them to get a spanning forest of the original graph. Before go to the details, let us formally define the spanning forest.

Definition 32.5.7 (Rooted Spanning Forest)**.** Let $G = (V, E)$ be an undirected graph. Let $\mathrm{par} : V \to V$ be a set of parent pointers which is compatible (Definition 32.4.4) with G. If $\forall u, v \in V, \mathrm{dist}_G(u, v) < \infty \Rightarrow \mathrm{par}^{(\infty)}(u) = \mathrm{par}^{(\infty)}(v)$, and $\forall v \in V, \mathrm{part}(v) \neq v \Rightarrow (v, \mathrm{part}(v)) \in E$, then we call par a rooted spanning forest of G.

The Algorithm 32.10 shows how to combine the spanning forest in the contracted graph with local spanning trees to get a spanning forest in the graph before contraction. Figure 32.1 shows an example.

Lemma 32.5.12. *Let $G_2 = (V_2, E_2)$ be an undirected graph. Let $\widetilde{\mathrm{par}} : V_2 \to V_2$ be a set of parent pointers (See Definition 32.4.2) which satisfies that $\forall v \in V_2$ with $\widetilde{\mathrm{par}}(v) \neq v$, $(v, \widetilde{\mathrm{par}}(v))$ must be in E_2. Let $G_1 = (V_1, E_1)$ be an undirected graph satisfies $V_1 = \{v \in V_2 \mid \widetilde{\mathrm{par}}(v) = v\}, E_1 = \{(u, v) \in V_1 \times V_1 \mid u \neq v, \exists (x, y) \in E_2, \widetilde{\mathrm{par}}^{(\infty)}(x) = u, \widetilde{\mathrm{par}}^{(\infty)}(y) = v\}$. Let $\mathrm{par} : V_1 \to V_1$ be a rooted spanning forest (See Definition 32.5.7) of G_1. Let $f : V_1 \times V_1 \to \{\mathrm{null}\} \cup (V_2 \times V_2)$ satisfy the following property: for $u \neq v \in V_1$, if $\mathrm{part}(u) = v$, then $f(u, v) \in \{(x, y) \in E_2 \mid \widetilde{\mathrm{par}}^{(\infty)}(x) = u, \widetilde{\mathrm{par}}^{(\infty)}(y) = v\}$, and $f(v, u) \in \{(x, y) \in E_2 \mid \widetilde{\mathrm{par}}^{(\infty)}(x) = v, \widetilde{\mathrm{par}}^{(\infty)}(y) = u\}$. Let $\widehat{\mathrm{par}} = \textsc{ForestExpansion}(\mathrm{par}, \widetilde{\mathrm{par}}, f)$. Then $\widehat{\mathrm{par}} : V_2 \to V_2$ is a rooted spanning forest of G_2. In addition, $\mathrm{dep}(\widehat{\mathrm{par}}) \leq (2 \cdot \mathrm{dep}(\widetilde{\mathrm{par}}) + 1)(\mathrm{dep}(\mathrm{par}) + 1)$.*

Proof. Let $x, y \in V_2, u = \widetilde{\text{par}}^{(\infty)}(x), v = \widetilde{\text{par}}^{(\infty)}(y) \in V_1$ if $\text{dist}_{G_2}(x, y) < \infty$, then since $E_1 = \{(u', v') \in V_1 \times V_1 \mid u' \neq v', \exists (x', y') \in E_2, \widetilde{\text{par}}^{(\infty)}(x') = u', \widetilde{\text{par}}^{(\infty)}(y') = v'\}$, it must be true that $\text{dist}_{G_1}(u, v) < \infty$. Since par is a spanning forest of G_1, we have $\text{par}^{(\infty)}(u) = \text{par}^{(\infty)}(v)$. It suffices to say $\forall x \in V_2, \widehat{\text{par}}^{(\infty)}(x) = \text{par}^{(\infty)}(\widetilde{\text{par}}^{(\infty)}(x))$. We can prove it by induction on $\text{dep}_{\text{par}}(\widetilde{\text{par}}^{(\infty)}(x))$. Let $u = \widetilde{\text{par}}^{(\infty)}(x)$. If $\text{dep}_{\text{par}}(u) = 0$, then $\text{par}^{(\infty)}(u) = u$. In this case, we have $\widehat{\text{par}}^{(\infty)}(x) = \widetilde{\text{par}}^{(\infty)}(x) = u = \text{par}^{(\infty)}(u) = \text{par}^{(\infty)}(\widetilde{\text{par}}^{(\infty)}(x))$, and also we have $\text{dep}_{\widehat{\text{par}}}(x) = \text{dep}_{\widetilde{\text{par}}}(x)$. Now suppose for all $x \in V_2$ with $\text{dep}_{\text{par}}(\widetilde{\text{par}}^{(\infty)}(x)) \leq i - 1$, it has $\widehat{\text{par}}^{(\infty)}(x) = \text{par}^{(\infty)}(\widetilde{\text{par}}^{(\infty)}(x))$ and $\text{dep}_{\widehat{\text{par}}}(x) \leq i \cdot (2\text{dep}(\widetilde{\text{par}}) + 1)$. Let $y \in V_2$ satisfy $\text{dep}_{\text{par}}(\widetilde{\text{par}}^{(\infty)}(y)) = i$. Let $v = \widetilde{\text{par}}^{(\infty)}(y)$. By line 8 and the properties of f, we know $\widetilde{\text{par}}^{(\infty)}(x_v) = v$, and $\widetilde{\text{par}}^{(\infty)}(y_v) = \text{part}(v)$. By line 9, line 10 and Lemma 32.5.11, we have $\widehat{\text{par}}_v^{(\infty)}(y) = x_v, \widehat{\text{par}}(x_v) = y_v$. Thus, there must be $k \leq 2\text{dep}_{\widehat{\text{par}}}(y)$ such that $\widehat{\text{par}}^{(k)}(y) = x_v$. Since $\widehat{\text{par}}^{(\infty)}(y_v) = \text{par}^{(\infty)}(v)$ and $\text{dep}_{\widehat{\text{par}}}(y_v) \leq i \cdot (2\text{dep}(\widetilde{\text{par}}) + 1)$, we have $\widehat{\text{par}}^{(\infty)}(y) = \text{par}^{(\infty)}(v) = \text{par}^{(\infty)}(\widetilde{\text{par}}^{(\infty)}(y))$ and $\text{dep}_{\widehat{\text{par}}}(y) \leq (i + 1) \cdot (2\text{dep}(\widetilde{\text{par}}) + 1)$.

In addition, by the properties of f and Lemma 32.5.11, $\forall v \in V_2$ with $\widehat{\text{par}}(v) \neq v$, we have $(v, \widehat{\text{par}}(v)) \in E_2$. To conclude, $\widehat{\text{par}} : V_2 \to V_2$ is a spanning forest of G_2, and $\text{dep}(\widehat{\text{par}}) \leq (\text{dep}(\text{par}) + 1)(2\text{dep}(\widetilde{\text{par}}) + 1)$. □

32.5.5 Spanning Forest Algorithm

In this section, we show how to apply the ideas shown in connectivity algorithm to get an spanning forest algorithm. Algorithm 32.11 can output a spanning forest of a graph G, but the edges are not orientated. Then in the Algorithm 32.12, we assign each forest edge an direction thus it is a rooted spanning forest.

Before we prove the correctness of the algorithms, let us briefly introduce the meaning of each variables appeared in the algorithms.

In Algorithm 32.11, G_0 is the original input graph, for $i \in \{0\} \cup [r-1]$, G'_i is obtained by deleting all the small size connected components in G_i, and G_{i+1} is obtained by contracting some vertices of G'_i. For a vertex v in graph G_i, if $h_i(v) = \text{null}$, then it means that the connected component which contains v is deleted when obtaining G'_i. If $h_i(v) \neq \text{null}$, it means that the vertex v is contracted to the vertex $h_i(v)$ when obtaining G_{i+1}. par_i is a rooted forest (may not be spanning) in graph G_i, if a tree from the forest is spanning in G_i, then all the vertex in that tree will be deleted when obtaining G'_i. Otherwise all the vertices in that tree will be contracted to the root, and the root will be one of the vertex in G_{i+1}. Since each connected component in G_{i+1} is obtained by contraction of some vertices in a connected component in G_i, each edge in G_{i+1} must correspond to an edge in G_i where the end vertices of the edge are contracted to different vertices. Thus, each edge in G_i should correspond to an edge in G, and $g_i : E_i \to E$ records the such correspondence. D_i records the edges added to the spanning forest F in the i^{th} round. For each vertex v in graph G_i, $\widetilde{T}_i(v)$ is a local shortest path tree (See definition 32.5.1) which is either with a large size or is a spanning tree in the component of v. L_i is a set of random leaders in G'_i such that in each local shortest path tree $\widetilde{T}_i(v)$, there is at least one leader shown in the tree. The following

lemmas formally state the properties of the algorithm.

Lemma 32.5.13. *Let $G = (V, E)$ be an undirected graph, m be a parameter which is at least $16|V|$, and r be a rounds parameter. If* SpanningForest(G, m, r) *(Algorithm 32.11) does not return FAIL, then* $\mathrm{diam}(G) = \mathrm{diam}(G_0) \geq \mathrm{diam}(G'_0) \geq \mathrm{diam}(G_1) \geq \mathrm{diam}(G'_1) \geq \cdots \geq \mathrm{diam}(G_r)$.

Proof. By property 3 of Lemma 32.5.3, $\forall [i] \in \{0\} \cup [r-1]$, there is no edge between $V_i \setminus V'_i$ and V'_i. Thus, $\mathrm{diam}(G'_i) \leq \mathrm{diam}(G_i)$. Then due to property 1 of Corollary 32.4.8, we have $\mathrm{diam}(G_{i+1}) \leq \mathrm{diam}(G'_i)$. □

Lemma 32.5.14. *Let $G = (V, E)$ be an undirected graph, m be a parameter which is at least $16|V|$, and r be a rounds parameter. If* SpanningForest(G, m, r) *(Algorithm 32.11) does not return FAIL, then* $\forall i \in \{0\} \cup [r-1]$, $\mathrm{dep}(\mathrm{par}_i) \leq \min(\mathrm{diam}(G), \lfloor (m/n_i)^{1/2} \rfloor)$.

Proof. Let $v \in V_i$. If $v \in V_i \setminus V'_i$, then due to property 3 of Lemma 32.5.3, we have $V_{\widetilde{T}_i(v)} = V_{\widetilde{T}_i(u_v)}$. Due to Lemma 32.5.13 and Lemma 32.5.3, we have $\mathrm{dep}_{\mathrm{par}_i}(v) \leq \mathrm{dep}(\widetilde{T}_i(u_v)) \leq \min(\mathrm{diam}(G), \lfloor (m/n_i)^{1/2} \rfloor)$. For $v \in V_i$, we define $\mathrm{dist}_{G_i}(v, L_i) = \min_{u \in L_i} \mathrm{dist}_{G_i}(v, u)$. By Lemma 32.5.3, we know $\mathrm{dist}_{G_i}(v, L_i) = \mathrm{dist}_{G_i}(v, z_i(v))$. Since $\widetilde{T}_i(v)$ is a LSPT (See Definition 32.5.1), by applying Lemma 32.5.9, we know $\mathrm{dist}_{G_i}(v, L_i) = \mathrm{dist}_{G_i}(w_i(v), L_i) + 1$, and $(v, w_i(v)) \in E_i$. Thus, by induction on $\mathrm{dist}_{G_i}(v, L_i)$, we can get $\mathrm{dep}_{\mathrm{par}_i}(v) \leq \mathrm{dist}_{G_i}(v, L_i)$. By Lemma 32.5.13 and Lemma 32.5.3, we can conclude $\mathrm{dep}(\mathrm{par}_i)(v) \leq \min(\mathrm{diam}(G), \lfloor (m/n_i)^{1/2} \rfloor)$. □

Lemma 32.5.15. *Let $G = (V, E)$ be an undirected graph, m be a parameter which is at least $16|V|$, and r be a rounds parameter. If* SpanningForest(G, m, r) *(Algorithm 32.11) does*

not return FAIL, then $\forall i \in \{-1, 0\} \cup [r-1]$, *we can define*

$$\forall v \in V, h^{(i)}(v) = \begin{cases} v & i = -1; \\ h_i(h^{(i-1)}(v)) & h^{(i-1)}(v) \neq \text{null}; \\ \text{null} & \text{otherwise .} \end{cases}$$

Then we have following properties:

1. *If* $h^{(i)}(v) \neq \text{null}$, *then* $h^{(i)}(v) \in V_{i+1}$.
2. $\forall u, v \in V, h^{(i)}(u) \neq h^{(i)}(v), (u, v) \in E$, *we have* $(h^{(i)}(u), h^{(i)}(v)) \in E_{i+1}$.
3. $\forall (x, y) \in E_{i+1}, (u, v) = g_{i+1}(x, y)$, *we have* $(u, v) \in E, h^{(i)}(u) = x, h^{(i)}(v) = y$.

Proof. For property 1, we can prove it by induction. It is true for $i = -1$. If $h_0(v) \neq \text{null}$, we know $h_0(v)$ must be assigned at line 19. Due to property 2 of Lemma 32.4.6, $h_0(v) \in V_1$. Suppose $\forall v \in V, h^{(i-1)}(v) \neq \text{null}$, we have $h^{(i-1)}(v) \in V_i$. For a vertex v with $h^{(i)}(v) \neq \text{null}$, according to the definition of $h^{(i)}(v)$, we know $h^{(i-1)}(v) \neq \text{null}$. Let $u = h^{(i-1)}(v)$. u must be a vertex in G_i by the induction hypothesis. Since $h^{(i)}(v) \neq \text{null}$, we know $h_i(u) \neq \text{null}$. Thus, $h_i(u)$ must be assigned at line 19. Due to property 2 of Lemma 32.4.6, $h_i(u)$ must be in G_{i+1}, which implies $h^{(i)}(v) \in V_{i+1}$.

For property 2, we can also prove it by induction. It is true for $i = -1$. If $(u, v) \in E$, then due to property 3 of Lemma 32.5.3, either both u, v are in V_0' or both u, v are in $V_0 \setminus V_0'$. If both u, v are in $V_0 \setminus V_0'$, then $h_0(u) = h_0(v) = \text{null}$. Otherwise, if $h_0(u) \neq h_0(v)$, then due to property 3 of Lemma 32.4.6, $(h_0(u), h_0(v)) \in E_1$. Now suppose we have $\forall u, v \in V$, if $h^{(i-1)}(u) \neq h^{(i-1)}(v), (u, v) \in E$, then $(h^{(i-1)}(u), h^{(i-1)}(v)) \in E_i$. Let $(u, v) \in E, h^{(i)}(u) \neq h^{(i)}(v)$. Let $x = h^{(i-1)}(u), y = h^{(i-1)}(v)$. Due to property 3 of Lemma 32.5.3, either both x, y

are in V_i' or both are in $V_i \setminus V_i'$. If $x, y \in V_i \setminus V_i'$, then $h_i(x) = h_i(y) = \text{null}$ which contradicts to $h^{(i)}(u) \neq h^{(i)}(v)$. Thus, both of $x, y \in V_i'$. Then due to property 3 of Lemma 32.4.6, $(h_i(x), h_i(v)) \in E_{i+1}$. Thus, $(h^{(i)}(u), h^{(i)}(v)) \in E_{i+1}$.

For property 3, we can prove it by induction. It is true for $i = -1$. Let us consider the case when $i = 0$. Due to property 3 of Lemma 32.4.6 and the definition of g_0, g_1, we have $\forall (x, y) \in E_1$, $(u, v) = g_1(x, y)$, $h_0(u) = x, h_0(v) = y, (u, v) \in E$. Now suppose the property holds for $i - 1$. Let $(x, y) \in E_{i+1}$. Then $g_{i+1}(x, y) = g_i(x', y')$ for some $(x', y') \in E_i, h_i(x') = x, h_i(y') = y$. Let $(u, v) = g_i(x', y')$. By the induction hypothesis $(u, v) \in E, h^{(i-1)}(u) = x', h^{(i-1)}(v) = y'$. Thus, $h^{(i)}(u) = x, h^{(i)}(v) = y$. □

Lemma 32.5.16. *Let $G = (V, E)$ be an undirected graph, m be a parameter which is at least $16|V|$, and r be a rounds parameter. If* SpanningForest(G, m, r) *(Algorithm 32.11) does not return FAIL, then $\forall i \in \{-1, 0\} \cup [r - 1]$, we can define*

$$\forall v \in V, h^{(i)}(v) = \begin{cases} v & i = -1 \\ h_i(h^{(i-1)}(v)) & h^{(i-1)}(v) \neq \text{null} \\ \text{null} & \text{otherwise.} \end{cases}$$

Let $\forall i \in \{0\} \cup [r]$, $\widehat{G}_i = (V_i, \widehat{E}_i = \{(x, y) \mid (u, v) \in \bigcup_{j=i}^{r-1} D_j, h^{(j-1)}(u) = x, h^{(j-1)}(v) = y\})$. Then $\widehat{G}_i$ is a spanning forest of G_i.

Proof. The proof is by induction. When $i = r$, since $V_r = \emptyset$, $\widehat{G}_r = (\emptyset, \emptyset)$ is a spanning forest of G_r. Now suppose $\widehat{G}_{i+1}$ is a spanning forest of G_{i+1}. Let $u, v \in V_i$. By property 2, 3 of Lemma 32.5.15, we have $\widehat{E}_i \subseteq E_i$. Thus, if $\text{dist}_{G_i}(u, v) = \infty$, then $\text{dist}_{\widehat{G}_i}(u, v) = \infty$. If $\text{dist}_{G_i}(u, v) < \infty$, there are several cases:

1. If $h_i(u) = h_i(v) = \text{null}$, then due to line 10, we know $u_u = u_v$, and $\widetilde{T}_i(u_v)$ is a spanning tree of the component which contains u, v. Thus, $\widehat{G}_i$ has a spanning tree of the component which contains u, v.

2. If $h_i(u) = h_i(v) \neq \text{null}$, then $\text{par}_i : \{x \in V_i \mid h_i(x) = h_i(v)\} \to \{x \in V_i \mid h_i(x) = h_i(v)\}$ is a tree, and $\forall y \in \{x \in V_i \mid h_i(x) = h_i(v)\}$, if $\text{par}_i(y) \neq y$, then $(y, \text{par}_i(y)) \in \widehat{E}_i$. Since $\widehat{G}_{i+1}$ does not have any cycle, there is a unique path from u to v in $\widehat{G}_i$.

3. If $h_i(u) \neq h_i(v)$, then neither of them can be null. Since $\widehat{G}_{i+1}$ is a spanning forest on G_{i+1}, there must be a unique path from $h_i(u)$ to $h_i(v)$ in $\widehat{G}_{i+1}$. Suppose the path in $\widehat{G}_{i+1}$ is $h_i(u) = p_1 - p_2 - \cdots - p_t = h_i(v)$. Then there must be a sequence of vertices in G_i, $u = p_{1,1}, p_{1,2}, p_{2,1}, p_{2,2}, \cdots, p_{t,1}, p_{t,2} = v$ such that $h_i(p_{j,1}) = h_i(p_{j,2}) = p_j$ and $(p_{j-1,2}, p_{j,1}) \in \widehat{E}_i$. Thus, there is a unique path from u to v.

Thus, $\widehat{G}_i$ is a spanning forest of G_i. □

Corollary 32.5.17 (Correctness of Algorithm 32.11)**.** *Let $G = (V, E)$ be an undirected graph, m be a parameter which is at least $16|V|$, and r be a rounds parameter. If* SpanningForest(G, m, r) *(Algorithm 32.11) does not return FAIL, then $\widehat{G}_0 = (V, F)$ is a spanning forest of G.*

Proof. Just apply Lemma 32.5.16 for $i = 0$ case. □

Lemma 32.5.18. *Let $G = (V, E)$ be an undirected graph, m be a parameter which is at least $16|V|$, and r be a rounds parameter. If* SpanningForest(G, m, r) *(Algorithm 32.11) does*

not return FAIL, then $\forall i \in \{-1, 0\} \cup [r-1]$, *we can define*

$$\forall v \in V, h^{(i)}(v) = \begin{cases} v & i = -1 \\ h_i(h^{(i-1)}(v)) & h^{(i-1)}(v) \neq \text{null} \\ \text{null} & \text{otherwise.} \end{cases}$$

$\forall i \in \{0\} \cup [r-1], v \in V_i$ *with* $\text{par}_i(v) \neq v$, *there exists* $(x, y) \in F$ *such that* $h^{(i-1)}(x) = v, h^{(i-1)}(y) = \text{par}_i(v)$.

Proof. By line 11, line 19, $\forall i \in \{0\} \cup [r-1], v \in V_i, \text{par}_i(v) \neq v$, we have $g_i(v, \text{par}_i(v)), g_i(\text{par}_i(v), v) \in D_i \subseteq F$. Since $(\text{par}_i(v), v) \in E_i$, by property 3 of Lemma 32.5.15, $(x, y) = g_i(v, \text{par}_i(v))$ satisfies $h^{(i-1)}(x) = v, h^{(i-1)}(y) = \text{par}_i(v)$. □

Theorem 32.5.19 (Correctness of Algorithm 32.12)**.** *Let* $G = (V, E)$ *be an undirected graph,* m *be a parameter which is at least* $16|V|$, *and* r *be a rounds parameter. If* SpanningForest(G, m, r) *(Algorithm 32.11) does not return FAIL, then let the output be the input of* Orientate$(\cdot)$, *(Algorithm 32.12) and the output* $\text{par} : V \to V$ *of* Orientate$(\cdot)$ *will be a rooted spanning forest (See Definition 32.5.7) of* G. *Furthermore,* $\text{dep}(\text{par}) \leq O(\text{diam}(G))^r$.

Proof. The proof is by induction. We want to show $\widehat{\text{par}}_i$ is a rooted spanning forest of G_i. When $i = r$, since $V_r = \emptyset$, the claim is true. Now suppose we have $\widehat{\text{par}}_{i+1}$ is a spanning forest of G_{i+1}. Let $\widetilde{G}_{i+1} = (\widetilde{V}_{i+1}, E_{i+1})$. It is easy to see $\widetilde{\text{par}}_{i+1} : \widetilde{V}_{i+1} \to \widetilde{V}_{i+1}$ is a spanning forest of $\widetilde{G}_{i+1}$. An observation is $\widetilde{V}_{i+1} = \{v \in V_i \mid \text{par}_i(v) = v\}$. Thus, $\widetilde{\text{par}}_{i+1}, \text{par}_i$ satisfies the condition in Lemma 32.5.12 when invoking ForestExpansion$(\widetilde{\text{par}}_{i+1}, \text{par}_i, f_{i+1})$. By Lemma 32.5.18, we know f_{i+1} also satisfies the condition in Lemma 32.5.12 when we invoke ForestExpansion$(\widetilde{\text{par}}_{i+1}, \text{par}_i, f_{i+1})$. Thus, $\widehat{\text{par}}_i$ is a rooted spanning forest of G_i due to Lemma 32.5.12.

By Lemma 32.5.12, we have $\mathrm{dep}(\widehat{\mathrm{par}}_i) \leq 16\mathrm{dep}(\widehat{\mathrm{par}}_{i+1})\,\mathrm{diam}(G)$. By induction, we have $\mathrm{dep}(\mathrm{par}) \leq O(\mathrm{diam}(G))^r$. □

Lemma 32.5.20. *Let $G = (V, E)$ be an undirected graph, m be a parameter which is at least $16|V|$, and $r \leq n$ be a round parameter. If* SpanningForest(G, m, r) *(Algorithm 32.11) does not return FAIL, then with probability at least* 0.89, $\sum_{i=0}^{r} n_i \leq 40n$.

Proof. Since $V_r \subseteq V_{r-1} \subseteq \cdots \subseteq V_0 = V$, we have $n_r \leq n_{r-1} \leq n_{r-2} \leq \cdots \leq n$. Due to line 14, line 15 and line 17, we know $\forall i \in \{0\} \cup [r-1]$, $V_{i+1} = L_i$. If $p_i < 1/2$, we know $p_i = (30\log(n) + 100)/\gamma_i$. Since $|V_{\widetilde{T}_i}(v)| \geq \gamma_i$, we can apply Lemma 32.4.3 to get $\Pr(|L_i| \leq 1.5p_in_i) \geq \Pr(|L_i| \leq 0.75n_i) \geq 1 - 1/(100n)$. By taking union bound over all $i \in \{0\} \cup [r-1]$, with probability at least 0.99, if $p_i < 0.5$, then $n_{i+1} \leq 0.75n_i$. By applying Lemma 32.4.4, condition on n_i and $p_i = \frac{1}{2}$, we have $\mathbb{E}(n_{i+1}) \leq 0.75n_i$. By Markov's inequality, with probability at 0.89, we have $\sum_{i=0}^{r} n_i \leq 40n$. □

Now let us define the total iterations of Algorithm 32.11 as the following:

Definition 32.5.8 (Total iterations)**.** Let graph $G = (V, E)$, $m \leq \mathrm{poly}(|V|)$ be a parameter which is at least $16|V|$, and r be a rounds parameter. The total number of iterations of SpanningForest(G, m, r) (Algorithm 32.11) is defined as $\sum_{i=0}^{r-1}(k_i + k'_i)$, where $\forall i \in \{0\} \cup [r-1], k_i$ denotes the number of iterations (See Definition 32.5.5) of MultipleLargeTrees(G_i, m) (see line 8), and k'_i denotes the number of iterations (See Definition 32.4.5) of TreeContraction(G'_i, par_i) (see line 17).

Theorem 32.5.21 (Success probability of Algorithm 32.11)**.** *Let $G = (V, E)$ be an undirected graph. Let $m \leq \mathrm{poly}(n)$ and $m \geq 16|V|$. Let r be a rounds parameter. Let $c > 0$*

be a sufficiently large constant. If $r \geq c \log \log_{m/n} n$*, then with probability at least* 0.79, SpanningForest(G, m, r) *(Algorithm 32.11) does not return* FAIL*. Furthermore, let* $\forall i \in \{0\} \cup [r-1], k_i$ *be the number of iterations (See Definition 32.5.5) of* MultipleLargeTrees(G_i, m) *and* k'_i *be the number of iterations (See Definition 32.4.5) of* TreeContraction$(G'_i, \mathrm{par}_i : V'_i \to V'_i)$*. Let* $c_1 > 0$ *be a sufficiently large constant. If* $m \geq c_1 n \log^8 n$*, then with probability at least* 0.99, $\sum_{i=0}^{r-1} k_i + k'_i \leq O(\min(\log(\mathrm{diam}(G)) \log \log_{\mathrm{diam}(G)}(n),\ r \log(\mathrm{diam}(G))))$*. If* $m < c_1 n \log^8 n$*, then with probability at least* 0.98, $\sum_{i=0}^{r-1} k'_i + k_i \leq O(\min(\log(\mathrm{diam}(G)) \log \log_{\mathrm{diam}(G)}(n) + (\log \log n)^2, r \log(\mathrm{diam}(G))))$.

Proof. Due to Lemma 32.5.20, with probability at last 0.89, we have $\forall i \in [r], n'_i \leq 40n$. Thus, we can condition on that SpanningForest(G, m, r) will not fail on line 21.

Due to Lemma 32.5.6, $k_i \leq O(\log(\mathrm{diam}(G_i))) \leq O(\log(\mathrm{diam}(G)))$. Due to Corollary 32.4.8 and Lemma 32.5.14, $k'_i \leq O(\log(\mathrm{diam}(G)))$. Thus, $\sum_{i \in \{0\} \cup [r-1]} k'_i + k_i \leq O(r \log(\mathrm{diam}(G)))$.

Since $V_r \subseteq V_{r-1} \subseteq V_{r-2} \subseteq \cdots \subseteq V_0 = V$, we have $n_r \leq n_{r-1} \leq n_{r-2} \leq \cdots \leq n$. Due to line 14, line 15 and line 17, we know $\forall i \in \{0\} \cup [r-1]$, $V_{i+1} = L_i$. If $p_i < 1/2$, we know $p_i = (30 \log(n) + 100)/\gamma_i$. Since $|V_{\widetilde{T}_i}(v)| \geq \gamma_i$, we can apply Lemma 32.4.3 to get $\Pr(|L_i| \leq 1.5 p_i n_i) \geq 1 - 1/(100n)$. By taking union bound over all $i \in \{0\} \cup [r-1]$, with probability at least 0.99, if $p_i < 0.5$, then $n_{i+1} \leq 1.5 p_i n_i \leq 0.75 n_i$. Let $\mathcal{E}$ be the event that $\forall i \in \{0\} \cup [r-1]$, if $p_i < 0.5$, then $n_{i+1} \leq 1.5 p_i n_i$. Now, we suppose $\mathcal{E}$ happens.

If $p_0 = 0.5$, then $m \leq n \cdot (600 \log n)^8$. By applying Lemma 32.4.4, $\mathbb{E}(n_{i+1}) = \mathbb{E}(|L_i|) \leq 0.75\, \mathbb{E}(n_i) \leq \cdots \leq 0.75^{i+1} n$. By Markov's inequality, when $i^* \geq 8 \log(6000 \log n)/\log(4/3)$, with probability at least 0.99, $n_{i^*} \leq n/(600 \log n)^8$ and thus $p_{i^*} < 0.5$. Condition on this

event and $\mathcal{E}$, we have

$$
\begin{aligned}
n_r &\le \frac{\left(\frac{\left(\frac{n_{i^*}^{1.25}}{m^{0.25}}(45\log n+150)\right)^{1.25}}{m^{0.25}}(45\log n+150)\right)^{\cdots}}{\cdots} && \text{(Apply } r'=r-i^* \text{ times)}\\
&= n_{i^*}/(m/n_{i^*})^{1.25^{r'}-1}\cdot(45\log n+150)^{4\cdot(1.25^{r'}-1)}\\
&\le n/\left(m/\left(n_{i^*}(45\log n+150)^4\right)\right)^{1.25^{r'}-1}\\
&\le n/\left(m/\left(n_{i^*}(45\log n+150)^4\right)\right)^{1.25^{r'}/2} \le n/(m/n)^{1.25^{r'}/2} \le 1/2,
\end{aligned}
$$

where the second inequality follows by $n_{i^*} \le n$, the third inequality follows by $r' \ge 5$, the forth inequality follows by $n_{i^*} \le n/(600\log n)^8$, and the last inequality follows by $r' \ge \frac{2}{\log 1.25}\log\log_{m/n}(2n)$. Since $16n \le m \le n\cdot(600\log n)^8, \log\log_{m/n} n = \Theta(\log\log n)$. Let $c > 0$ be a sufficiently large constant. Thus, when $r \ge c\log\log_{m/n} n \ge i^* + r' = 8\log(6000\log n)/\log(4/3) + \frac{2}{\log 1.25}\log\log_{m/n}(2n)$, with probability at least 0.98, $n_r = 0$ implies that SpanningForest(G, m, r) will not fail. Due to Lemma 32.5.6, we have $k_i \le$

$O(\min(\log(m/n_i), \log(\mathrm{diam}(G))))$. Thus,

$$
\begin{aligned}
&\sum_{i=0}^{r-1} k_i \\
&= \sum_{i=0}^{i^*} k_i + \sum_{i=i^*+1}^{r-1} k_i \\
&\le O\left((\log\log n)^2\right) + \sum_{i=i^*+1}^{r-1} k_i \\
&\le O\left((\log\log n)^2\right) + \sum_{i:i\ge i^*+1, m/n_i \le \mathrm{diam}(G)} k_i + \sum_{i:i\le r, m/n_i > \mathrm{diam}(G)} k_i \\
&\le O\left((\log\log n)^2\right) + O\left(\sum_{i=0}^{\lceil \log_{1.25}\log_2(\mathrm{diam}(G))\rceil} \log(2^{1.25^i})\right) + O\left(\sum_{i=0}^{\lceil \log_{1.25}\log_{\mathrm{diam}(G)}(m)\rceil} \log(\mathrm{diam}(G))\right) \\
&\le O\left((\log\log n)^2\right) + O(\log(\mathrm{diam}(G))) + O(\log(\mathrm{diam}(G))\log\log_{\mathrm{diam}(G)}(n)) \\
&\le O(\log(\mathrm{diam}(G))\log\log_{\mathrm{diam}(G)}(n) + (\log\log(n))^2),
\end{aligned}
$$

where the first inequality follows by $i^* = O(\log\log n)$ and $\forall i \le [i^*], m/n_i \le \mathrm{poly}(\log n)$, the third inequality follows by $m/n_{i+1} \ge (m/n_i)^{1.25}/(45\log n + 100) \ge (m/n_i)^{1.125}$. Due to Corollary 32.4.8 and Lemma 32.5.14, we also have $k'_i \le O(\min(\log(m/n_i), \log(\mathrm{diam}(G))))$. Then, by the same argument, we have $\sum_{i=0}^{r-1} k'_i = O(\log(\mathrm{diam}(G))\log\log_{\mathrm{diam}(G)}(n) + (\log\log(n))^2)$.

If $m > n \cdot (600\log n)^8$, then $\forall i \in \{0\} \cup [r-1]$, we have $p_i < 0.5$. Since $\mathcal{E}$ happens.

We have:

$$
\begin{aligned}
n_r &\le \frac{\left(\frac{\left(\frac{n^{1.25}}{m^{0.25}}(45\log n+150)\right)^{1.25}}{m^{0.25}}(45\log n+150)\right)^{\cdots}}{\cdots} && \text{(Apply } r \text{ times)} \\
&= \frac{n^{1.25^r}}{m^{1.25^r-1}}(45\log n+150)^{4\cdot(1.25^r-1)} \\
&= n/(m/n)^{1.25^r-1}\cdot(45\log n+150)^{4\cdot(1.25^r-1)} \\
&= n/\left(m/\left(n(45\log n+150)^4\right)\right)^{1.25^r-1} \\
&\le n/\left(m/\left(n(45\log n+150)^4\right)\right)^{1.25^{r/2}} \\
&\le n/\left(m/\left(n(200\log n)^4\right)\right)^{1.25^{r/2}} \\
&\le \frac{1}{2},
\end{aligned}
$$

where the second inequality follows by $r \ge 5$, the third inequality follows by $45\log n + 150 \le 200\log n$, and the last inequality follows by

$$
r \ge c\log\log_{m/n} n \ge 2\log_{1.25}\log_{(m/n)^{1/2}} 2n \ge 2\log_{1.25}\log_{m/(n(200\log n)^4)} 2n,
$$

for a sufficiently large constant $c > 0$. Since n_r is an integer, n_r must be 0 when $n_r \le 1/2$. SpanningForest(G, m, r) will succeed with probability at least 0.99. Due to Lemma 32.5.6, we have $k_i \le O(\min(\log(m/n_i), \log(\mathrm{diam}(G))))$. Thus,

$$
\begin{aligned}
\sum_{i=0}^{r-1} k_i &\le \sum_{m/n_i \le \mathrm{diam}(G)} k_i + \sum_{m/n_i > \mathrm{diam}(G)} k_i \\
&\le O\left(\sum_{i=0}^{\lceil\log_{1.25}\log_2(\mathrm{diam}(G))\rceil} \log(2^{1.25^i})\right) + O\left(\sum_{i=0}^{\lceil\log_{1.25}\log_{\mathrm{diam}(G)}(m)\rceil} \log(\mathrm{diam}(G))\right) \\
&\le O(\log(\mathrm{diam}(G))) + O(\log(\mathrm{diam}(G))\log\log_{\mathrm{diam}(G)}(n)),
\end{aligned}
$$

where the second inequality follows by $m/n_{i+1} \geq (m/n_i)^{1.25}/(45 \log n + 100) \geq (m/n_i)^{1.125}$.

Due to Corollary 32.4.8 and Lemma 32.5.14, we also have $k'_i \leq O(\min(\log(m/n_i), \log(\mathrm{diam}(G))))$.

Then, by the same argument, we have $\sum_{i=0}^{r-1} k'_i = O(\log(\mathrm{diam}(G))) + O(\log(\mathrm{diam}(G)) \log \log_{\mathrm{diam}(G)}(n))$.

□

32.6 Depth-First-Search Sequence for Tree and Applications

32.6.1 Lowest Common Ancestor and Multi-Paths Generation

Given a rooted forest induced by $\text{par} : V \to V$ which is a set of parent pointers (See Definition 32.4.2) on V, and a set of q queries $Q = \{(u_1, v_1), (u_2, v_2), \cdots, (u_q, v_q) \mid u_i, v_i \in V\}$, we show an algorithm which can return a mapping $\text{lca} : Q \to (V \cup \{\text{null}\}) \times (V \cup \{\text{null}\}) \times (V \cup \{\text{null}\})$ such that $\forall (u_i, v_i) \in Q, (p, p_{u_i}, p_{v_i}) = \text{lca}(u_i, v_i)$ satisfies the following properties:

1. If $\text{par}^{(\infty)}(u_i) = \text{par}^{(\infty)}(v_i)$, then p is the lowest ancestor of u_i and v_i. Otherwise $p = p_{u_i} = p_{v_i} = \text{null}$.
2. Suppose $p \neq \text{null}$. If $p \neq u_i$, then p_{u_i} is an ancestor of u_i and $\text{part}(p_{u_i}) = p$. Otherwise, $p_{u_i} = \text{null}$.
3. Suppose $p \neq \text{null}$. If $p \neq v_i$, then p_{v_i} is an ancestor of v_i and $\text{part}(p_{v_i}) = p$. Otherwise, $p_{v_i} = \text{null}$.

Before we describe the algorithms, let us formally define *ancestor* and the *lowest common ancestor*.

Definition 32.6.1 (Ancestor)**.** Let $\text{par} : V \to V$ be a set of parent pointers (See Definition 32.4.2) on a vertex set V. For $u, v \in V$, if $\exists k \in \mathbb{Z}_{\geq 0}$ such that $u = \text{par}^{(k)}(v)$, then u is an ancestor of v.

Definition 32.6.2 (Common ancestor and the lowest common ancestor)**.** $\text{par} : V \to V$ be a set of parent pointers (See Definition 32.4.2) on a vertex set V. For $u, v \in V$, if w is an ancestor of u and is also an ancestor of v, then w is a common ancestor of (u, v). If a common

ancestor w of (u, v) satisfies $\mathrm{dep}_{\mathrm{par}}(w) \geq \mathrm{dep}_{\mathrm{par}}(x)$ for any common ancestor x of (u, v), then w is the lowest common ancestor (LCA) of (u, v).

Definition 32.6.3 (Path between two vertices)**.** $\mathrm{par} : V \to V$ be a set of parent pointers (See Definition 32.4.2) on a vertex set V. For $u, v \in V$, if $\mathrm{par}^{(\infty)}(u) = \mathrm{par}^{(\infty)}(v)$, then the path from u to v is a sequence $(x_1, x_2, \cdots, x_j, x_{j+1}, \cdots x_k)$ such that $\forall i \neq i' \in [k], x_i \neq x_{i'}, x_1 = u, x_k = v, x_j$ is the lowest common ancestor of (u, v), $\forall i \in [j-1], \mathrm{part}(x_i) = x_{i+1}$, and $\forall i \in \{j+1, j+2, \cdots, k\}, \mathrm{part}(x_i) = x_{i-1}$.

The algorithm which can compute the lowest common ancestor is described in Algorithm 32.13.

Lemma 32.6.1. *Let* $\mathrm{par} : V \to V$ *be a set of parent pointers (See Definition 32.4.2) on a vertex set* V*. Let* $Q = \{(u_1, v_1), (u_2, v_2), \cdots, (u_q, v_q)\}$ *be a set of* q *pairs of vertices, and* $\forall i \in [q], u_i \neq v_i$*. Let* $\mathrm{lca} = \mathrm{LCA}(\mathrm{par}, Q)$ *(Algorithm 32.13). Then for any* $(u, v) \in Q$*,* $(p, p_u, p_v) = \mathrm{lca}(u, v)$ *satisfies the following properties:*

1. *If* $\mathrm{par}^{(\infty)}(u) \neq \mathrm{par}^{(\infty)}(v)$*, then* $p = p_u = p_v = \mathrm{null}$.
2. *If* u *(or* v*) is the lowest common ancestor of* (u, v)*, then* $p = u, p_u = \mathrm{null}, p_v \neq u$ *is an ancestor of* v *such that* $\mathrm{part}(p_v) = u$ *(or* $p = v, p_v = \mathrm{null}, p_u \neq v$ *is an ancestor of* u *such that* $\mathrm{part}(p_u) = v$*.)*
3. *If neither* u *nor* v *is the lowest common ancestor of* (u, v) *and* $\mathrm{par}^{(\infty)}(u) = \mathrm{par}^{(\infty)}(v)$*, then* p *is the lowest common ancestor of* (u, v)*,* $p_u \neq p$ *is an ancestor of* u*,* $p_v \neq p$ *is an ancestor of* v*, and* $\mathrm{part}(p_u) = \mathrm{part}(p_v) = p$.

Proof. According to Lemma 32.5.7, r should be at most $\lceil \log(\text{dep}(\text{par})+1) \rceil$, $\text{dep}_{\text{par}} : V \to \mathbb{Z}_{\geq 0}$ records the depth of every vertex in V, and $\forall i \in \{0\} \cup [r], v \in V$ $g_i(v) = \text{par}^{(2^i)}(v)$. Then property 1 follows by line 5 directly.

Then for all $(u, v) \in Q$ with $\text{par}^{(\infty)}(u) = \text{par}^{(\infty)}(v)$, either $(u, v) \in Q'$ or $(v, u) \in Q'$. For each $(u, v) \in Q'$, we have $\text{dep}_{\text{par}}(u) \geq \text{dep}_{\text{par}}(v)$. For all $(u, v) \in Q'$, with $\text{dep}_{\text{par}}(u) > \text{dep}_{\text{par}}(v)$, by induction we can prove that $\forall i \in \{0\} \cup [r-1], (x, y) = h_i(u, v)$ satisfies that x is an ancestor of u, $y = v$, $\text{dep}_{\text{par}}(x) > \text{dep}_{\text{par}}(v)$ and $\text{par}^{(2^i)}(x)$ is an ancestor of v. Thus, for $(p, p_u, p_v) = \text{lca}(u, v)$, if v is the lowest common ancestor of u, then we have $p = v, p_u = h_0(u), p_v = \text{null}$. In this case, $h_0(u)$ is an ancestor of u, and $\text{dep}_{\text{par}}(u) = \text{dep}_{\text{par}}(v) + 1, \text{part}(h_0(u)) = v$. Thus, property 2 holds.

For all $(u, v) \in Q$ with $\text{par}^{(\infty)}(u) = \text{par}^{(\infty)}(v)$, if neither u nor v is the lowest common ancestor of (u, v), then we know either (u, v) or (v, u) is in Q''. Now let $(u, v) \in Q''$. We have $\text{dep}_{\text{par}}(h'_r(u)) = \text{dep}_{\text{par}}(h'_r(v)), h'_r(u) \neq h'_r(v)$, and $h'_r(u), h'_r(v)$ are ancestors of u, v respectively. We can prove by induction to get $\forall i \in \{0\} \cup [r], h'_i(u) \neq h'_i(v)$ and $\text{par}^{(2^i)}(h'_i(u)) = \text{par}^{(2^i)}(h'_i(v))$ is a common ancestor of (u, v). Thus, $p = \text{part}(h'_0(u)) = \text{part}(h'_0(v))$ is the lowest common ancestor of (u, v), and $\text{dep}_{\text{par}}(h'_0(u)) = \text{dep}_{\text{par}}(h'_0(v)) = \text{dep}_{\text{par}}(p) + 1$. Since $p_u = h'_0(u), p_v = h'_0(v)$, property 3 holds. □

In Algorithm 32.14, we show a generalization of Algorithm 32.8 such that we can find multiple vertex-to-ancestor paths simultaneously.

The following lemma claims the properties of the outputs of Algorithm 32.14. And the proof is similar to the proof of Lemma 32.5.9.

Lemma 32.6.2. *Let* $\text{par} : V \to V$ *be a set of parent pointers (See Definition 32.4.2) on a vertex set* V*. Let* $Q = \{(u_1, v_1), (u_2, v_2), \cdots, (u_q, v_q)\} \subseteq V \times V$ *satisfy* $\forall j \in [q]$, v_j *is an ancestor (See Definition 32.6.1) of* u_j *in* par*. Let* $(\text{dep}_{\text{par}}, \{P_j \mid j \in [q]\}) = \textsc{MultiPath}(\text{par}, Q)$ *(Algorithm 32.14). Then* $\text{dep}_{\text{par}} : V \to \mathbb{Z}_{\geq 0}$ *records the depth of every vertex in* V *and* $\forall j \in [q], P_j \subseteq V$ *is the set of all vertices on the path from* u_j *to* v_j*, i.e.* $P_j = \{v \in V \mid \exists k_1, k_2 \in \mathbb{Z}_{\geq 0}, v = \text{par}^{(k_1)}(u_j), v_j = \text{par}^{(k_2)}(v)\}$*. Furthermore,* r *should be at most* $\lceil \log(\text{dep}(\text{par}) + 1) \rceil$*.*

Proof. By Lemma 32.5.7, since $(r, \text{dep}_{\text{par}}, \{g_i \mid i \in \{0\} \cup [r]\}) = \textsc{FindAncestors}(\text{par})$, we know r should be at most $\lceil \log(\text{dep}(\text{par}) + 1) \rceil$, $\text{dep}_{\text{par}} : V \to \mathbb{Z}_{\geq 0}$ records the depth of every vertex in V, and $\forall i \in \{0\} \cup [r], v \in V$ $g_i(v) = \text{par}^{(2^i)}(v)$.

For $j \in [q]$, let us prove that P_j is the vertex set of all the vertices on the path from u_j to its ancestor v_j. We use divide-and-conquer to get P_j. The following claim shows that $S_j^{(i)}$ is a set of segments which is a partition of the path from u_j to v_j, and each segment has length at most 2^{r-i}.

Claim 32.6.3. $\forall i \in \{0\} \cup [r], j \in [q]$ $S_j^{(i)}$ *satisfies the following properties:*

1. $\exists (x, y) \in S_j^{(i)}$ *such that* $x = u_j$*.*
2. $\exists (x, y) \in S_j^{(i)}$ *such that* $y = v_j$*.*
3. $\forall (x, y) \in S_j^{(i)}$, $\text{dep}_{\text{par}}(y) - \text{dep}_{\text{par}}(x) \leq 2^{r-i}$*.*
4. $\forall (x, y) \in S_j^{(i)}$*, if* $y \neq v_j$*, then* $\exists (x', y') \in S_j^{(i)}, x' = y$*.*
5. $\forall (x, y) \in S_j^{(i)}$, $\exists k \in \mathbb{Z}_{\geq 0}, \text{par}^{(k)}(x) = y$*.*

Proof. We fix a $j \in [q]$. Our proof is by induction. According to line 4, all the properties hold when $i = 0$. Suppose all the properties hold for $i - 1$. For property 1, by induction we know there exists $(x, y) \in S_j^{(i-1)}$ such that $x = u_j$. Then by line 9 and line 10, there must be an (x, y') in $S_j^{(i)}$. For property 2, by induction we know there exists $(x, y) \in S_j^{(i-1)}$ such that $y = v_j$. Thus, there must be an (x', y) in $S_j^{(i)}$. For property 3, if (x, y) is added into $S_j^{(i)}$ by line 10, then $\mathrm{dep}_{\mathrm{par}}(x) - \mathrm{dep}_{\mathrm{par}}(y) \le 2^{r-i}$. Otherwise, in line 9, we have $\mathrm{dep}_{\mathrm{par}}(x) - \mathrm{dep}_{\mathrm{par}}(g_{r-i}(x)) \le 2^{r-i}, \mathrm{dep}_{\mathrm{par}}(g_{r-i}(x)) - \mathrm{dep}_{\mathrm{par}}(y) \le 2^{r-i+1} - 2^{r-i} = 2^{r-i}$. For property 4, if (x, y) is added into $S_j^{(i)}$ by line 10, then by induction there is $(y, y') \in S_j^{(i-1)}$, and thus by line 10 and line 9, there must be $(y, y'') \in S_j^{(i)}$. Otherwise, in line 9 will generate two pairs $(x, g_{r-i}(x)), (g_{r-i}(x), y)$. For $(x, g_{r-i}(x))$, the property holds. For $(g_{r-i}(x), y)$, there must be $(y, y') \in S_{i-1}$ and thus there should be $(y, y'') \in S^{(i)}$. For property 5, since $g_{r-i}(x) = \mathrm{par}^{(r-i)}(x)$, for all pairs generated by line 9 and line 10, the property holds. □

By Claim 32.6.3, we know

$$
\begin{aligned}
S_j^{(r)} = \{ & \\
& (u_j, \mathrm{part}(u_j)), \\
& \left(\mathrm{part}(u_j), \mathrm{par}^{(2)}(u_j)\right), \\
& \left(\mathrm{par}^{(2)}(u_j), \mathrm{par}^{(3)}(u_j)\right), \\
& \cdots, \\
& \left(\mathrm{par}^{(\mathrm{dep}_{\mathrm{par}}(u_j) - \mathrm{dep}_{\mathrm{par}}(v_j) - 1)}(u_j), \mathrm{par}^{(\mathrm{dep}_{\mathrm{par}}(u_j) - \mathrm{dep}_{\mathrm{par}}(v_j))}(u_j)\right) \\
\} & .
\end{aligned}
$$

Thus, P_j is the set of all the vertices on the path from u_j to an ancestor v_j. □

32.6.2 Depth-First-Search Sequence for a Tree

Since we can use our spanning tree algorithm to get a rooted tree, in this section, we only consider how to get a Depth-First-Search (DFS) sequence for a rooted tree. Before we go to the details, let us firstly give formal definitions of some useful concepts.

Definition 32.6.4 (Children in the forest)**.** Given a set of parent pointers (See Definition 32.4.2) $\text{par} : V \to V$ on a vertex set V. $\forall u, v \in V, u \neq v$ if $\text{part}(u) = v$, then we say u is a child of v. $\forall v \in V$, we can define $\text{child}_{\text{par}}(v)$ as the set of all children of v, i.e. $\text{child}_{\text{par}}(v) = \{u \in V \mid u \neq v, \text{part}(u) = v\}$. Furthermore, if u is the k^{th} smallest vertex in the children set $\text{child}_{\text{par}}(v)$, then we say $\text{rank}_{\text{par}}(u) = k$, or u is the k^{th} child of v. If $\text{part}(v) = v$, then $\text{rank}_{\text{par}}(v) = 1$. We use $\text{child}_{\text{par}}(v, k)$ to denote the k^{th} child of v.

For simplicity of the notation, if $\text{par} : V \to V$ is clear in the context, we just use $\text{child}(v), \text{rank}(v)$ and $\text{child}(v, k)$ to denote $\text{child}_{\text{par}}(v), \text{rank}_{\text{par}}(v)$ and $\text{child}_{\text{par}}(v, k)$ respectively.

Definition 32.6.5 (Leaves in the forest)**.** Given a set of parent pointers (See Definition 32.4.2) $\text{par} : V \to V$ on a vertex set V. If $\text{child}_{\text{par}}(v) = \emptyset$, then v is called a leaf. The set of all the leaves of par is defined as $\text{leaves}(\text{par}) = \{v \mid \text{child}_{\text{par}}(v) = \emptyset\}$.

Definition 32.6.6 (Subtree)**.** Let $\text{par} : V \to V$ be a set of parent pointers (See Definition 32.4.2) on a vertex set V. Let $v \in V, V' = \{u \in V \mid v$ is an ancestor (Definition 32.6.1) of $u\}$. Let $\text{par}' : V' \to V'$ be a set of parent pointers on V'. If $\forall u \in V' \setminus \{v\}, \text{par}'(u) = \text{part}(u)$, and $\text{par}'(v) = v$, then we say par' is the subtree of v in par. For $u \in V'$, we say u is in the subtree of v.

Definition 32.6.7 (Depth-First-Search (DFS) sequence)**.** Let par : $V \to V$ be a set of parent pointers (See Definition 32.4.2) on a vertex set V. Let v be a vertex in V. If v is a leaf (See Definition 32.6.5) in par, then the DFS sequence of the subtree (See Definition 32.6.6) of v is (v). Otherwise the DFS sequence of the subtree of v in par is recursively defined as

$$(v, a_{1,1}, a_{1,2}, \cdots, a_{1,n_1}, v, a_{2,1}, a_{2,2}, \cdots, a_{2,n_2}, v, \cdots, a_{k,1}, a_{k,2}, \cdots, a_{k,n_k}, v),$$

where $k = |\operatorname{child}(v)|$ is the number of children (See Definition 32.6.4) of v, and $\forall i \in [k], (a_{i,1}, \cdots, a_{i,n_i})$ is the DFS sequence of the subtree of $\operatorname{child}(v, i)$, i.e. the i^{th} child of v.

If $\forall u \in V, \text{par}^{(\infty)}(u) = v$, then the subtree of v is exactly par, and thus the DFS sequence of the subtree of v is also called the DFS sequence of par.

Here are some useful facts of the above defined DFS sequence.

Fact 32.6.4. *Let* par : $V \to V$ *be a set of parent pointers (See Definition 32.4.2) on a vertex set* V*, and* par *has a unique root. Let* $A = (a_1, a_2, \cdots, a_m)$ *be the DFS sequence (See Definition 32.6.7) of* par*. Then,* A *satisfies the following properties:*

1. $\forall v \in V$, v *appears exactly* $|\operatorname{child}(v)| + 1$ *times in* A*.*
2. *If* a_i *is the* k^{th} *time that* v *appears, and* a_j *is the* $(k+1)^{th}$ *time that* v *appears. Then* $(a_{i+1}, a_{i+2}, \cdots, a_{j-1})$ *is the DFS sequence of the subtree of* $\operatorname{child}(v, k)$ *(See Definition 32.6.4), the* k^{th} *child of* v*. Furthermore,* a_{i+1} *is the first time that* $\operatorname{child}(v, k)$ *appears, and* a_{j-1} *is the last time of* $\operatorname{child}(v, k)$ *appears.*
3. *If* a_i *is the first time that* v *appears, and* a_j *is the last time that* v *appears. Then* $(a_i, a_{i+1}, \cdots, a_j)$ *is the DFS sequence of the subtree of* v*.*

4. $m = 2|V| - 1$.

Proof. The property 1, 2, 3 directly follows by Definition 32.6.7.

For property 4, notice that $\forall u \in V, \text{part}(u) \neq u$, u can only be a child of $\text{part}(u)$. Thus, $\sum_{v \in V}(|\text{child}(v)| + 1) = |V| - 1 + |V| = 2|V| - 1$. □

Due to the above fact, if v is a leaf in par, then it will only once in the DFS sequence. Thus, we are able to determine the order of all the leaves.

Definition 32.6.8 (The order of the leaves)**.** Let $\text{par} : V \to V$ be a set of parent pointers (See Definition 32.4.2) on a vertex set V, and par has a unique root. Let $A = (a_1, a_2, \cdots, a_m)$ be the DFS sequence (See Definition 32.6.7) of par . Let u, v be two leaves (See definition 32.6.5) of par . If u appears before v in A, then we say $u <_{\text{par}} v$.

32.6.2.1 Leaf Sampling

Given a set of rooted trees, our goal is to sample a set of leaves for each tree, and to give an order of those sampled leaves. The algorithm is shown in Algorithm 32.15.

Lemma 32.6.5. *Let* $\text{par} : V \to V$ *be a set of parent pointers (See Definition 32.4.2) on a vertex set* V*, and* par *has a unique root. Let* $m > 0, \delta \in (0, 1)$ *be parameters, and let* $|V| \leq m^{1/\delta}$*. Let* $(a_1, a_2, \cdots, a_s) = \textsc{LeafSampling}(\text{par}, m, \delta)$ *(Algorithm 32.15). Then it has following properties:*

1. $a_1 <_{\text{par}} a_2 <_{\text{par}} \cdots <_{\text{par}} a_s$.
2. *If* $|V| \leq m$ *or* $|\text{leaves}(\text{par})| \leq 8\lceil m^{1/3} \rceil$*, then* $\{a_1, a_2, \cdots, a_s\} = \text{leaves}(\text{par})$*. Otherwise, with probability at least* $1 - 1/(100m^{5/\delta})$*,* $\forall v \in \text{leaves}(\text{par}) \setminus \{a_1\}$*, there is a vertex*

$w \in \{a_1, a_2, \cdots, a_s\}$ *such that* $w <_{\text{par}} v$ *and the number of leaves between* w *and* v *is at most* $|\operatorname{leaves}(\text{par})|/\lceil m^{1/3} \rceil$, *i.e.* $|\{u \in \operatorname{leaves}(\text{par}) \mid w <_{\text{par}} u <_{\text{par}} v\}| \leq |\operatorname{leaves}(\text{par})|/\lceil m^{1/3} \rceil$.

3. *If* $|V| > m$ *and* $|\operatorname{leaves}(\text{par})| > 8\lceil m^{1/3} \rceil$, *then with probability at least* $1 - 1/(100m^{5/\delta})$, $s = |S| = |\{a_1, a_2, \cdots, a_s\}| \leq 960\lceil m^{1/3} \rceil (1 + \log(m)/\delta)$.

Proof. Firstly, let us focus on property 1. According to line 11 to line 13 and Lemma 32.5.7, we know $\forall k \in \mathbb{Z}_{\geq 0} \operatorname{rank}_{\text{par}}(\text{par}^{(k)}(a_1)) = 1$, and $\text{par}'(a_1) = a_1$ which implies that a_1 is a leaf. Due to the definition of 32.6.7, we know that a_1 must be the first leaf appeared in the DFS sequence of par . We can prove the property by induction. Suppose we already have $a_1 <_{\text{par}} a_2 <_{\text{par}} \cdots <_{\text{par}} a_{i-1}$. According to line 19 and Lemma 32.6.1, p_{a_{i-1},a_i} is the LCA of (a_{i-1}, a_i). $p_{a_{i-1}a_i,a_{i-1}}$ is a child of p_{a_{i-1},a_i} and is an ancestor of a_{i-1}. $p_{a_{i-1}a_i,a_i}$ is a child of p_{a_{i-1},a_i} and is an ancestor of a_i. By Fact 32.6.4, since $\operatorname{rank}(p_{a_{i-1}a_i,a_{i-1}}) < \operatorname{rank}(p_{a_{i-1}a_i,a_i})$, we have $a_{i-1} <_{\text{par}} a_i$. To conclude, we have $a_1 <_{\text{par}} a_2 <_{\text{par}} \cdots <_{\text{par}} a_s$.

For property 2, if $|V| \leq m$ or $|\operatorname{leaves}(\text{par})| \leq 8\lceil m^{1/3} \rceil$, then by line 6 and line 7, we know $\{a_1, a_2, \cdots, a_s\} = \operatorname{leaves}(\text{par})$.

Now consider the case when $|V| > m$ and $|\operatorname{leaves}(\text{par})| > 8\lceil m^{1/3} \rceil$. Let $t = \lceil m^{1/3} \rceil$. Let $\operatorname{leaves}(\text{par}) = \{u_1, u_2, \cdots, u_q\}$, and let $u_1 <_{\text{par}} u_2 <_{\text{par}} \cdots <_{\text{par}} u_q$. Let us partition $u_1, \cdots, u_q$ into $4 \cdot t$ groups $G_1 = \{u_1, u_2, \cdots, u_{\lfloor q/(4t) \rfloor}\}, G_2 = \{u_{\lfloor q/(4t) \rfloor + 1}, u_{\lfloor q/(4t) \rfloor + 2}, \cdots, u_{2 \cdot \lfloor q/(4t) \rfloor}\}, \cdots, G_{4t} = \{u_{(4t-1)\lfloor q/(4t) \rfloor + 1}, u_{(4t-1)\lfloor q/(4t) \rfloor + 2}, \cdots, u_q\}$. Then each group has size at least $q/(8t)$ and at most

$q/(2t)$. For a certain G_i, by Chernoff bound, we have

$$\begin{aligned}
&\Pr\left(|G_i \cap S| \leq \frac{1}{2} \cdot \frac{q}{8t} \cdot p\right) \\
\leq\ &\exp\left(-\frac{1}{8} \cdot \frac{q}{8t} \cdot p\right) \\
\leq\ &1/(100m^{10/\delta})
\end{aligned}$$

where the last inequality follows by $p = \min(1, (10 + 10\log(m)/\delta) \cdot 64t/q)$. Notice that $q \leq |V| \leq m^{1/\delta}$. We can take union bound over all G_i. Then with probability at least $1 - 1/(100m^{5/\delta})$, $\forall i \in [4t], G_i \cap S \neq \emptyset$. Thus, $\forall v \in \text{leaves(par)}$, there is a vertex $w \in \{a_1, a_2, \cdots, a_s\}$ such that $w <_{\text{par}} v$ and the number of leaves between w and v is at most $|\text{leaves(par)}|/\lceil m^{1/3} \rceil$, i.e. $|\{u \in \text{leaves(par)} \mid w <_{\text{par}} u <_{\text{par}} v\}| \leq |\text{leaves(par)}|/\lceil m^{1/3} \rceil$.

For property 3, by applying Chernoff bound, we have

$$\begin{aligned}
&\Pr\left(|S \cap \text{leaves(par)}| \geq \frac{3}{2}|\text{leaves(par)}| \cdot p\right) \\
\leq\ &\exp\left(-\frac{1}{12} \cdot |\text{leaves(par)}| \cdot p\right) \\
\leq\ &1/(100m^{10/\delta})
\end{aligned}$$

where the last inequality follows by $p = \min(1, (10 + 10\log(m)/\delta) \cdot 64t/|\text{leaves(par)}|)$.

Since $\frac{3}{2}|\text{leaves(par)}| \cdot p \leq 960\lceil m^{1/3} \rceil (1 + \log(m)/\delta)$, we complete the proof. □

32.6.2.2 DFS Subsequence

Let $\text{par} : V \to V$ be a set of parent pointers on a vertex set V, and par has a unique root v. Let $\{u_1, u_2, \cdots, u_q\} = \text{leaves(par)}$, and $u_1 <_{\text{par}} u_2 <_{\text{par}} \cdots <_{\text{par}} u_q$. One observation is that the DFS sequence of par can be generated in the following way:

1. The first part of the DFS sequence is the path from v to u_1.

2. Then it follows by the path from part(u_1) to the LCA of (u_1, u_2), the path from one of the child of the LCA of (u_1, u_2) to u_2, the path from part(u_2) to the LCA of (u_2, u_3), the path from one of the child of the LCA of (u_2, u_3) to $u_3, \cdots$, the path from one of the child of the LCA of (u_{q-1}, u_q) to u_q.

3. The last part of the DFS sequence is a path from part(u_q) to v.

Fact 32.6.6. *Let* par $: V \to V$ *be a set of parent pointers (See Definition 32.4.2) on a vertex set* V*, and* par *has a unique root* v*. Let* $\{u_1, u_2, \cdots, u_q\} =$ leaves(par) *(See Definition 32.6.5), and* $u_1 <_{\text{par}} u_2 <_{\text{par}} \cdots <_{\text{par}} u_q$*. Let* $A = (a_1, a_2, \cdots, a_m)$ *be the DFS sequence (See Definition 32.6.7) of* par*. Then,*

1. *If* u_1 *appears at* a_i*, then* $(a_1, a_2, \cdots, a_i)$ *is the path from* v *to* u_1*.*

2. $\forall i \in [q-1]$*, if* u_i *appears at* a_j*, and* u_{i+1} *appears at* a_k*, then* $\exists j < t < k$ *such that* a_t *is the LCA of* (u_i, u_{i+1})*. In addition,* $(a_j, a_{j+1}, \cdots, a_t)$ *is the path from* a_j *to* a_t*, and* $(a_t, a_{t+1}, \cdots, a_k)$ *is the path from* a_t *to* a_k*.*

3. *If* u_q *appears at* a_i*, then* $(a_i, a_{i+1}, \cdots, a_m)$ *is the path from* u_q *to* v*.*

Proof. Property 1, 3 follows by the definition of DFS sequence (See Definition 32.6.7) and a simple induction.

Now consider property 2. Since A is a DFS sequence, $\forall l \in \{j, j+1, \cdots, k-1\}$, either part$(a_l) = a_{l+1}$ or part$(a_{l+1}) = a_l$. Thus, the path between u_i and u_{i+1} is a subsequence of

$(a_j, a_{j+1}, \cdots, a_k)$. If $\text{part}(a_{l+1}) = a_l$ but a_{l+1} is not on the path between u_i and u_{i+1}, then there must be a leaf x in the subtree of a_{l+1} which implies $u_i <_{\text{par}} x <_{\text{par}} u_{i+1}$, and thus leads to a contradiction. If $\text{part}(a_l) = a_{l+1}$ but a_{l+1} is not on the path between u_i and u_{i+1}, then both u_i and u_{i+1} should be in the subtree of a_l, and both of u_i and u_{i+1} should be in the DFS sequence of the subtree of a_l. But we know a_{l+1} cannot be in the DFS sequence of the subtree of a_l. This leads to a contradiction. □

Lemma 32.6.7. *Let* $\text{par} : V \to V$ *be a set of parent pointers (See Definition 32.4.2) on a vertex set* V*, with a unique root. Let* $v \in V$*. Let* $V' = V \backslash \{u \in V \mid v$ *is an ancestor (See Definition 32.6.1) of* u*. Let* $\text{par}' : V' \to V'$ *satisfy* $\forall u \in V', \text{par}'(u) = \text{part}(u)$*. Then the DFS sequence (See Definition 32.6.7) of* par′ *is a subsequence of the DFS sequence of* par*.*

Proof. The proof follows by the property 3 of Fact 32.6.4 directly. □

Corollary 32.6.8. *Let* $\text{par} : V \to V$ *be a set of parent pointers (See Definition 32.4.2) on a vertex set* V*, and* par *has a unique root. Let* $v_1, v_2, \cdots, v_t$ *be* t *vertices in* V*. Let* $V' = V \backslash \{u \in V \mid \exists v \in \{v_1, \cdots, v_t\}, v$ *is an ancestor (See Definition 32.6.1) of* $u\}$*. Let* $\text{par}' : V' \to V'$ *satisfy* $\forall u \in V', \text{par}'(u) = \text{part}(u)$*. Then the DFS sequence (See Definition 32.6.7) of* par′ *is a subsequence of the DFS sequence of* par*.*

Proof. The proof is by induction on t. When $t = 1$, then the statement is true by Lemma 32.6.7. Suppose the statement is true for $t-1$. Let $V'' = V \backslash \{u \in V \mid \exists v \in \{v_1, \cdots, v_{t-1}\}, v$ is an ancestor of $u\}$, and let $\text{par}'' : V'' \to V''$ satisfy $\forall v \in V'', \text{par}''(v) = \text{part}(v)$. By induction hypothesis, the DFS sequence of par″ is a subsequence of the DFS sequence of par. If one of the $v_1, \cdots, v_{t-1}$ is an ancestor of v_t, then par′ = par″, thus, the DFS sequence of par′ is a subsequence of

the DFS sequence of par. Otherwise, we have $V' = V'' \setminus \{u \in V'' \mid v_t \text{ is an ancestor of } u\}$. By Lemma 32.6.7, the DFS sequence of par$'$ is a subsequence of the DFS sequence of par$''$. Thus, the DFS sequence of par$'$ is a subsequence of the DFS sequence of par. □

Lemma 32.6.9 (Removing several subtrees)**.** *Let* par $: V \to V$ *be a set of parent pointers (See Definition 32.4.2) on a vertex set* V*, and* par *has a unique root. Let* $m > 0, \delta \in (0, 1)$ *be parameters, and let* $|V| \leq m^{1/\delta}$*. Let* $(V', A) = \textsc{SubDFS}(\text{par}, m, \delta)$ *(Algorithm 32.16). Then* $\forall u \in V'$*, we have* $\text{part}(u) \in V'$*. Furthermore, with probability at least* $1 - 1/(100m^{5/\delta})$*,* $\forall u \in V \setminus V'$*, the number of leaves (See Definition 32.6.5) in the subtree (See Definition 32.6.6) of* u *is at most* $\lfloor |\text{leaves}(\text{par})|/\lceil m^{1/3} \rceil \rfloor$.

Proof. By Lemma 32.6.5, we know $L \subseteq \text{leaves}(\text{par})$, and $l_1 <_{\text{par}} l_2 <_{\text{par}} \cdots <_{\text{par}} l_t$.

We first prove $\forall u \in V', \text{part}(u) \in V'$. Our proof is by induction on the leaf l_i. By Lemma 32.6.1, we have that $\forall i \in [t-1], p_{l_i,l_{i+1}}$ is the LCA of (l_i, l_{i+1}), $p_{i,l_{i+1}}$ is an ancestor of l_{i+1}, and $p_{i,l_{i+1}} \neq p_{l_i,l_{i+1}}, \text{part}(p_{i,l_{i+1}}) = p_{l_i,l_{i+1}}$. By Lemma 32.14, P_1 contains all the vertices on the path from l_1 to the root v. P_1 is the set of all the ancestors of l_1. Thus, every ancestor u of l_1 is in P_1 and satisfies $\text{part}(u) \in V'$. P_2 contains all the vertices on the path from l_1 to an ancestor of l_1. Thus, $P_2 \subseteq P_1$. Suppose now $P_1 \cup P_2 \cup \cdots \cup P_{2i-2} = \{u \in V \mid \exists j \in [i-1], u \text{ is an ancestor of } l_j\}$. Notice that P_{2i-1} contains all the vertices on the path from l_i to the ancestor p_{i-1,l_i}. Since $\text{part}(p_{i-1,l_i}) = p_{l_{i-1},l_i}$ is also an ancestor of l_{i-1}, we have $P_1 \cup P_2 \cup \cdots \cup P_{2i-2} \cup P_{2i-1} = \{u \in V \mid \exists j \in [i], u \text{ is an ancestor of } l_j\}$. Since P_{2i} contains all the vertices on the path from l_i to an ancestor of l_i, we have $P_{2i} \subseteq P_1 \cup P_2 \cup \cdots \cup P_{2i-2} \cup P_{2i-1}$. To conclude, we have $V' = P_1 \cup P_2 \cup \cdots \cup P_{2t} = \{u \in V \mid \exists j \in [t], u \text{ is an ancestor of } l_j\}$. Thus, $\forall u \in V'$, we have $\text{part}(u) \in V'$.

By Lemma 32.6.5, with probability at least $1 - 1/(100m^{5/\delta})$, $\forall u \in \text{leaves}(\text{par}) \setminus L$, there exists $w \in L, w <_{\text{par}} u$ such that $|\{x \in \text{leaves}(\text{par}) \mid w <_{\text{par}} x <_{\text{par}} u\}| \leq \lfloor |\text{leaves}(\text{par})|/\lceil m^{1/3} \rceil \rfloor$. In the following, we condition on the above event happens. Let $u \in V \setminus V'$. Due to Fact 32.6.4, the DFS sequence of the subtree of u in par must be a consecutive subsequence of the DFS sequence of par. Thus, $\exists x, y \in \text{leaves}(\text{par})$, the leaves in the subtree of u in par is the set $\{z \in \text{leaves}(\text{par}) \mid x <_{\text{par}} z <_{\text{par}} y\} \cup \{x\} \cup \{y\}$. If the number of leaves in the subtree of u is more than $\lfloor |\text{leaves}(\text{par})|/\lceil m^{1/3} \rceil \rfloor$, then $\exists l_i \in L$, u is an ancestor of leaf l_i. But $l_i \in V'$ contradicts to $u \notin V'$. Thus, the number of leaves in the subtree of u is at most $\lfloor |\text{leaves}(\text{par})|/\lceil m^{1/3} \rceil \rfloor$. □

Lemma 32.6.10 (A is a subsequence)**.** *Let* $\text{par} : V \to V$ *be a set of parent pointers (See Definition 32.4.2) on a vertex set* V*, and* par *has a unique root. Let* $m > 0, \delta \in (0, 1)$ *be parameters, and let* $|V| \leq m^{1/\delta}$*. Let* $(V', A) = \textsc{SubDFS}(\text{par}, m, \delta)$ *(Algorithm 32.16). Then* A *is a subsequence of the DFS sequence of* par*. Furthermore,* $\forall u \in V'$*,* u *appears in* A *exactly* $|\text{child}_{\text{par}}(u) + 1|$ *times, and* $\forall u \notin V'$*,* u *does not appear in* A*.*

Proof. We first show that A' is the DFS sequence of par'.

Claim 32.6.11. A' *is the DFS sequence of* $\text{par}' : V' \to V'$*.*

Proof. By Lemma 32.6.5, we know $\{l_1, l_2, \cdots, l_t\} = L \subseteq \text{leaves}(\text{par})$, and $l_1 <_{\text{par}} l_2 <_{\text{par}} \cdots <_{\text{par}} l_t$. By Lemma 32.6.1, we have that $\forall i \in [t-1], p_{l_i,l_{i+1}}$ is the LCA of (l_i, l_{i+1}), $p_{i,l_{i+1}}$ is an ancestor of l_{i+1}, and $p_{i,l_{i+1}} \neq p_{l_i,l_{i+1}}, \text{part}(p_{i,l_{i+1}}) = p_{l_i,l_{i+1}}$. By Lemma 32.14, $\forall i \in [t]$, P_{2i-1} and P_{2i} only contains some ancestors of l_i. Thus, $\text{leaves}(\text{par}') = L$.

According to Lemma 32.6.9 and Corollary 32.6.8, the DFS sequence of par' is a subsequence of the DFS sequence of par. Thus, we still have $l_1 <_{\text{par}'} l_2 <_{\text{par}'} <_{\text{par}'} \cdots <_{\text{par}'} l_t$.

Due to Lemma 32.14, P_1 contains all the vertices on the path from l_1 to the root v, P_{2t} contains all the vertices on the path from l_t to the root v, $\forall i \in [t-1], P_{2i}$ contains all the vertices on the path from $\text{par}'(l_i)$ to the LCA of (l_i, l_{i+1}), and P_{2i+1} contains all the vertices on the path from l_{i+1} to $p_{i,l_{i+1}}$. Thus, A'_1 is the path from the root v to leaf l_1, A'_{2t} is the path from l_t to the root v, $\forall i \in [t-1]$, $A'_{2i}A'_{2i+1}$ is the path from $\text{par}'(l_i)$ to l_{i+1}. Due to Fact 32.6.6, $A' = A'_1A'_2 \cdots A'_{2t}$ is the DFS sequence of par'. □

Let us define some notations. Let $\widetilde{A} = \{\widetilde{a}_1, \widetilde{a}_2, \cdots, \widetilde{a}_{\widetilde{s}}\}$ be the DFS sequence of par. $\forall u \in V$, let $\text{st}_{\widetilde{A}}(u) = j$ such that $\widetilde{a}_j$ is the first time that u appears in $\widetilde{A}$. We define $\text{ed}_{\widetilde{A}}(u)$ be the position such that $\widetilde{a}_{\text{ed}_{\widetilde{A}}(u)}$ is the last time that u appears in $\widetilde{A}$. Similarly, $\forall u \in V'$, we can define $\text{st}_{A'}(u), \text{st}_A(u), \text{ed}_{A'}(u), \text{ed}_A(u)$ to be the positions of the first time u appears in A', the first time u appears in A, the last time u appears in A', and the last time u appears in A respectively.

Since v is the root (in both par and par'), it suffices to prove that $(a_{\text{st}_A(v)}, a_{\text{st}_A(v)+1}, \cdots, a_{\text{ed}_A(v)})$ is a subsequence of $(\widetilde{a}_{\text{st}_{\widetilde{A}}(v)}, \widetilde{a}_{\text{st}_{\widetilde{A}}(v)+1}, \cdots, \widetilde{a}_{\text{ed}_{\widetilde{A}}(v)})$. Our proof is by induction on $\text{dep}_{\text{par}}(u)$ for $u \in V'$. If $\text{dep}_{\text{par}}(u) = \text{dep}(\text{par})$, then u must be a leaf in par' (or par, since par' and par are the same on V'). In this case, $\text{st}_A(u) = \text{ed}_A(u), \text{st}_{\widetilde{A}}(u) = \text{ed}_{\widetilde{A}}(u)$, and $(a_{\text{st}_A(u)}) = (\widetilde{a}_{\text{st}_{\widetilde{A}}(u)}) = (u)$. Suppose for all $u \in V'$ with $\text{dep}_{\text{par}}(u) > d$, we have that $(a_{\text{st}_A(u)}, \cdots, a_{\text{ed}_A(u)})$ is a subsequence of $(\widetilde{a}_{\text{st}_{\widetilde{A}}(u)}, \cdots, \widetilde{a}_{\text{ed}_{\widetilde{A}}(u)})$. Let u be a vertex in V' with $\text{dep}_{\text{par}}(u) = d$. If u is a leaf, then it is the same as the previous argument. Now let us consider the case when u is not a leaf. According to Claim 32.6.11, A' is the DFS sequence of par'. Due to line 30, A is obtained by duplicating each element of A' several times. Let $w_1, w_2, \cdots, w_k$ be the children of u in par', and $\text{rank}_{\text{par}'}(w_1) = 1, \text{rank}_{\text{par}'}(w_2) = 2, \cdots, \text{rank}_{\text{par}'}(w_k) = |\text{child}_{\text{par}'}(u)|$. Then, according to

Fact 32.6.4, $(a_{\mathrm{st}_A(u)}, \cdots, a_{\mathrm{ed}_A(u)})$ should look like:

$$(u, \cdots, u, a_{\mathrm{st}_A(w_1)}, \cdots, a_{\mathrm{ed}_A(w_1)}, u, \cdots, u, a_{\mathrm{st}_A(w_2)}, \cdots, a_{\mathrm{ed}_A(w_2)}, \cdots, a_{\mathrm{st}_A(w_k)}, \cdots, a_{\mathrm{ed}_A(w_k)}, u, \cdots, u)$$

where the number of u before $a_{\mathrm{st}_A(w_1)}$ is $\mathrm{rank}_{\mathrm{par}}(w_1)$ (see line 26), the number of u before $a_{\mathrm{st}_A(w_i)}$ for $i \in [k] \setminus \{1\}$ is $\mathrm{rank}_{\mathrm{par}}(w_i) - \mathrm{rank}_{\mathrm{par}}(w_{i-1})$ (see line 28), and the number of u after $a_{\mathrm{ed}_A(w_k)}$ is $|\operatorname{child}_{\mathrm{par}}(u)| - \mathrm{rank}_{\mathrm{par}}(w_k) + 1$ (see line 27). Since $\widetilde{A}$ is the DFS sequence of par, according to Fact 32.6.4, the number of u in $\widetilde{A}$ before $\widetilde{a}_{\mathrm{st}_{\widetilde{A}}(w_1)}$ is $\mathrm{rank}_{\mathrm{par}}(w_1)$. By our induction hypothesis, $(a_{\mathrm{st}_A(w_1)}, \cdots, a_{\mathrm{ed}_A(w_1)})$ is a subsequence of $(\widetilde{a}_{\mathrm{st}_{\widetilde{A}}(w_1)}, \cdots, \widetilde{a}_{\mathrm{ed}_{\widetilde{A}}(w_1)})$. Thus, $(a_{\mathrm{st}_A(u)}, \cdots, a_{\mathrm{ed}_A(w_1)})$ is a subsequence of $(\widetilde{a}_{\mathrm{st}_{\widetilde{A}}(u)}, \cdots, \widetilde{a}_{\mathrm{ed}_{\widetilde{A}(w_1)}})$. According to Fact 32.6.4, $\forall i \in [k] \setminus \{1\}$, the number of u in $\widetilde{A}$ between $\widetilde{a}_{\mathrm{ed}_{\widetilde{A}}(w_{i-1})}$ and $\widetilde{a}_{\mathrm{st}_{\widetilde{A}}(w_i)}$ is $\mathrm{rank}_{\mathrm{par}}(w_i) - \mathrm{rank}_{\mathrm{par}}(w_{i-1})$. By our induction hypothesis, for all $i \in [k] \setminus \{1\}$, $(a_{\mathrm{st}_A(w_i)}, \cdots, a_{\mathrm{ed}_A(w_i)})$ is a subsequence of $(\widetilde{a}_{\mathrm{st}_{\widetilde{A}}(w_i)}, \cdots, \widetilde{a}_{\mathrm{ed}_{\widetilde{A}}(w_i)})$. Thus, $(a_{\mathrm{st}_A(u)}, \cdots, a_{\mathrm{ed}_A(w_k)})$ is a subsequence of $(\widetilde{a}_{\mathrm{st}_{\widetilde{A}}(u)}, \cdots, \widetilde{a}_{\mathrm{ed}_{\widetilde{A}(w_k)}})$. According to Fact 32.6.4, the number of u in $\widetilde{A}$ after $\widetilde{a}_{\mathrm{ed}_{\widetilde{A}}(w_k)}$ is $|\operatorname{child}_{\mathrm{par}}(u)| - \mathrm{rank}_{\mathrm{par}}(w_k) + 1$. Thus, $(a_{\mathrm{st}_A(u)}, \cdots, a_{\mathrm{ed}_A(u)})$ is a subsequence of $(\widetilde{a}_{\mathrm{st}_{\widetilde{A}}(u)}, \cdots, \widetilde{a}_{\mathrm{ed}_{\widetilde{A}(u)}})$. Furthermore, the number of u appears in A is $|\operatorname{child}_{\mathrm{par}}(u)| - \mathrm{rank}_{\mathrm{par}}(w_k) + 1 + \mathrm{rank}_{\mathrm{par}}(w_1) + \sum_{i=2}^{k} \mathrm{rank}_{\mathrm{par}}(w_i) - \mathrm{rank}_{\mathrm{par}}(w_{i-1}) = |\operatorname{child}_{\mathrm{par}}(u)| + 1$.

Since A' is the DFS sequence of par', $\forall u \notin V'$, u does not appear in A'. Thus, $\forall u \notin V'$, u does not appear in A.

□

32.6.2.3 DFS Sequence

In this section, we show how to use Algorithm 32.16 as a subroutine to output a DFS sequence. The high level idea is that we use Algorithm 32.16 to generate subsequences of

the DFS sequence in each iteration, and we ensure that the miss part of the DFS sequence must be the DFS sequences of many subtrees. After the i^{th} iteration, we should ensure that the number of leaves of each subtree which has missing DFS sequence is at most n/m^i, where m is some parameter depends on some computational resources (e.g. memory size of a machine). The description of the algorithm is shown in Algorithm 32.17. Figure 32.2 shows one step in our algorithm.

Theorem 32.6.12 (Correctness of DFS sequence)**.** *Let* par $: V \to V$ *be a set of parent pointers (See Definition 32.4.2) on a vertex set* V*, and* par *has a unique root. Let* $n = |V|, m = n^\delta$ *for some constant* $\delta \in (0, 1)$*. If* $A = \mathrm{DFS}(\mathrm{par}, m)$ *(Algorithm 32.17) does not output* FAIL*, then* A *is the DFS sequence of* par *.*

Proof. It suffices to prove the following claim.

Claim 32.6.13. *Let* $i \in \{0\} \cup [r]$*.* A_i *is a subsequence of the DFS sequence of* par *.* $\forall v \in V_i$, $\mathrm{part}(v) \in V_i$*. Furthermore,* $\forall v \in V_i$*,* v *appears in* A_i *exactly* $|\mathrm{child}_{\mathrm{par}}(v)| + 1$ *times, and* $\forall v \notin V_i$*,* v *does not appear in* A_i*.*

Proof. Our proof is by induction on i. If $i = 0$, then by Lemma 32.6.10, A_0 is a subsequence of the DFS sequence of par, $\forall v \in V_0$, v appears in A_0 exactly $|\mathrm{child}_{\mathrm{par}}(v)| + 1$ times, and $\forall v \notin V_0$, v does not appear in A_0. By Lemma 32.6.9, we have $\forall v \in V_0, \mathrm{part}(v) \in V_0$.

Suppose the claim is true for $i - 1$. Let $u \in V_i$.

If $u \in V_{i-1}$, then since $V_{i-1} \subseteq V_i$, $\mathrm{part}(u) \in V_i$. Otherwise $u \in V_{i,v}$ for some v with $\mathrm{par}_i(v) = v$. If $u = v$, then $\mathrm{part}(v) \in V_{i-1} \subseteq V_i$. Otherwise, by Lemma 32.6.9, $\mathrm{part}(u) \in V_{i,v} \subseteq V_i$.

Now consider the property of A_i. If $u \in V_{i-1}$, then since A_{i-1} is a subsequence of A_i, and by Lemma 32.6.10 u cannot appear in any $A_{i,v}$, u must appear in A_i exactly $|\operatorname{child}_{\text{par}}(u)| + 1$ times. Otherwise $u \in V_{i,v}$ for some v with $\text{par}_i(v) = v$. By Lemma 32.6.10, u must appear in $A_{i,v}$ $|\operatorname{child}_{\text{par}_{i,v}}(u)| + 1 = |\operatorname{child}_{\text{par}}(u)| + 1$ times. Since u cannot appear in A_{i-1}, u must appear in A_i exactly $|\operatorname{child}_{\text{par}}(u)| + 1$ times. For $v \in V_i'$ with $\text{par}_i(v) = v$, according to Fact 32.6.4 and $\forall w \in \{x \in V \mid v \text{ is an ancestor of } x\}$, w cannot be in V_{i-1}, the $\text{rank}_{\text{par}}(v)^{\text{th}}$ time appearance of v and the $(\text{rank}_{\text{par}}(v)+1)^{\text{th}}$ time appearance of v should be adjacent in A_{i-1}. Due to Lemma 32.6.10, $A_{i,v}$ is a subsequence of the DFS sequence of the subtree of v in par . Due to Fact 32.6.4, A_i is still a subsequence of the DFS sequence of par after insertion of the sequence $A_{i,v}$.

For any $x \notin V_i$, by Lemma 32.6.10, x cannot be in any $A_{i,v}$. By our induction hypothesis, x cannot be in A_{i-1}. Thus, x cannot be in A_i. □

If the procedure does not output FAIL, then according to the above Claim 32.6.13, $\forall v \in V_r = V$, v appears in $A_r = A$ exactly $|\operatorname{child}_{\text{par}}(v)|+1$ times, and $A_r = A$ is a subsequence of the DFS sequence of par. Due to Fact 32.6.4, $A = A_r$ is the DFS sequence of par. □

The following lemma claims the success probability of Algorithm 32.17.

Theorem 32.6.14 (Success probability)**.** *Let* $\text{par} : V \to V$ *be a set of parent pointers (See Definition 32.4.2) on a vertex set* V*, and* par *has a unique root. Let* $n = |V|, m = n^{\delta}$ *for some constant* $\delta \in (0, 1)$*. With probability at least* $1 - 1/(100n^4)$*,* $A = \text{DFS}(\text{par}, m)$ *(Algorithm 32.17) does not output* FAIL*.*

Proof. $\forall i \in [r], v \in V'_i$ with $\mathrm{par}_i(v) = v$, let $\mathcal{E}_{i,v}$ be the event that $\forall u \in V'_i(v) \setminus V_{i,v}$, the number of leaves in the subtree of u in par is at most $|\operatorname{leaves}(\mathrm{par}_{i,v})|/n^{\delta/3}$. Notice that due to Lemma 32.6.9, if $\mathrm{par}_i(v) = v$, then v will be in V_i. Thus, we use $\mathcal{E}_v$ to denote the event $\mathcal{E}_{i,v}$. By Lemma 32.6.9, $\mathcal{E}_v$ happens with probability at least $1 - 1/(100n^5)$. By taking union bound over all v, with probability at least $1 - 1/(100n^4)$, all the events $\mathcal{E}_v$ will happen.

Claim 32.6.15. *Condition on all the events $\mathcal{E}_v$ happen. $\forall i \in [r], v \in V'_i$ with $\mathrm{par}_i(v) = v$, we have $|\operatorname{leaves}(\mathrm{par}_{i,v})| \le n/n^{(i-1)\delta/3}$.*

Proof. When $i = 1$, the claim is obviously true, since $|\operatorname{leaves}(\mathrm{par}_{i,v})| \le n$. Suppose the claim holds for $i - 1$. Let $v \in V'_i$ with $\mathrm{par}_i(v) = v$. There must be $v' \in V'_{i-1}$ with $\mathrm{par}_{i-1}(v') = v'$, and $v \in V'_{i-1}(v') \setminus V_{i-1,v'}$. Since $\mathcal{E}_{v'}$ happens, the number of leaves in the subtree of v in par is at most $|\operatorname{leaves}(\mathrm{par}_{i-1,v'})|/n^{\delta/3} \le n/n^{(i-1)\delta/3}$. □

If $V'_r \neq \emptyset$, then $\exists v \in V'_r$ with $\mathrm{par}_i(v) = v$ and $|\operatorname{leaves}(\mathrm{par}_{i,v})| \ge 1$. If all the events $\mathcal{E}_v$ happens, it will contradict to Claim 32.6.15. Thus, if all the events $\mathcal{E}_v$ happens, V'_r must be $\emptyset$, and thus $V_r = V$ which implies that the procedure will not fail. □

32.6.3 Range Minimum Query

Range Minimum Query (RMQ) problem is defined as following: given a sequence of n numbers $a_1, a_2, \cdots, a_n$, the goal is to preprocess the sequence a to get a data structure such that for any query $(p, q), (p < q)$ we can efficiently find the element which is the minimum in $a_p, a_{p+1}, \cdots, a_q$. A classic method is to preprocess a sparse table f in $\log(n)$ number of iterations such that $\forall i \in [n], j \in [\lceil \log n \rceil] \cup \{0\}$, $f_{i,j} = \arg\min_{i \le i' \le \min(n, i+2^j-1)} a_i$. To answer query for (p, q), it just needs to return $\arg\min_{i \in \{f_{p,j^*}, f_{q-2^{j^*}+1,j^*}\}} a_i$ for $j^* = \lfloor \log(q - p + 1) \rfloor$. In this section, we firstly show a modified data structure. We will compute $\widehat{f}_{i,j} = \arg\min_{i \le i' \le \min(n, i+\lceil n^\delta \rceil^j - 1)} a_{i'}$ The Algorithm is shown in Algorithm 32.18. Then we show how to use $\widehat{f}$ to compute f in Algorithm 32.19.

Lemma 32.6.16. *Let $a_1, a_2, \cdots, a_n$ be a sequence of numbers, and $\delta \in (0, 1)$. Let $\{\widehat{f}_{p,q}\}$ be the output of* SparseTable$^+(a_1, a_2, \cdots, a_n, \delta)$ *(Algorithm 32.18). Then $\forall p \in [n], q \in \{0\} \cup [\lceil 1/\delta \rceil]$, $\widehat{f}_{p,q} = \arg\min_{p \le p' \le \min(n, i+\lceil n^\delta \rceil^q - 1)} a_{p'}$.*

Proof. The proof is by induction on q. When $q = 0$, the statement obviously holds for all $\widehat{f}_{p,0}$. Suppose all $p \in [n], \widehat{f}_{p,0}, \widehat{f}_{p,1}, \cdots, \widehat{f}_{p,q-1}$ satisfy the property. The first observation is that the value of $\widehat{f}_{p,q}$ will be assigned in the procedure when $l = q, j = \lfloor (p - 1)/m \rfloor, i' = (p - 1) \mod m$. Then by line 8, $z^*_{j,l}$ will be the position of the minimum value in $a_{j \cdot m+m}, a_{j \cdot m+m+2}, \cdots, a_{j \cdot m+m^l-1}$ by our induction hypothesis. Then by line 11, $\widehat{f}_{i+i',l}$ will be the position of the minimum value in $a_{j \cdot m+i'+1}, a_{j \cdot m+i'+2}, \cdots, a_{j \cdot m+i'+m^l}$. Thus, Since $j \cdot m + i' + 1 = p$, $\widehat{f}_{p,q}$ satisfies the property. □

Lemma 32.6.17. *Let $a_1, a_2, \cdots, a_n$ be a sequence of numbers, and $\delta \in (0,1)$. Let $\{f_{p,q}\}$ be the output of* SparseTable$(a_1, a_2, \cdots, a_n, \delta)$ *(Algorithm 32.19). Then $\forall p \in [n], q \in \{0\} \cup [\lceil \log n \rceil]$, $f_{p,q} = \arg\min_{p \le p' \le \min(n, i+2^q-1)} a_{p'}$.*

Proof. Let $m = \lceil n^\delta \rceil$. By Lemma 32.6.16, $\forall x \in [n], y \in \{0\} \cup [\lceil 1/\delta \rceil]$, $\widehat{f}_{x,y} = \arg\min_{x \le x' \le \min(n, i+m^y-1)} a_{x'}$. Thus, by the definition of S_t, we know $z^*_{j,t} = \arg\min_{j \cdot m + m + 1 \le z \le j \cdot m + 2^t} a_z$. An observation is that the value of $f_{p,q}$ will be assigned in the procedure when $t = q, j = \lfloor (p-1)/m \rfloor, i' = (p-1) \bmod m$. By line 23, we know

$$f_{p,q} = f_{i+i',t} = \arg \min_{z: i+i'+1 \le z \le i+i'+2^t} a_z = \arg \min_{p \le p' \le \min(n, i+2^q-1)} a_{p'}.$$

□

32.6.4 Applications of DFS Sequence

In this section, we briefly discuss some applications of the DFS sequence of a tree.

Since the DFS sequence of a subtree should be a continuous subsequence of the DFS sequence of the tree, one direct application of the DFS sequence is to compute the size of each subtree, i.e. for each subtree with root v, we can find the first place v appeared and the last place v appeared, and then calculate the vertices between those two appearances.

Another application of the DFS sequence and the range minimum query is to output a data structure which can answer any LCA query in $O(1)$ time (for both sequential and parallel). This is better than the data structure provided by Section 32.6.1 which needs $O(\log D)$ time (for both sequential and parallel) to answer the query.

Since it is easy to output a data structure which can answer the depth of each vertex in $O(1)$ time (in both sequential and parallel), together with the lowest common ancestor data structure, we can answer the query of the tree distance between any two vertices in $O(1)$ time (for both sequential and parallel).

32.7 The MPC Model

In this section, let us introduce the computational model studied in this paper. Suppose we have p machines indexed from 1 to p each with memory size s words, where n is the number of words of the input and $p \cdot s = O(n^{1+\gamma}), s = \Theta(n^{\delta})$. Here $\delta \in (0, 1)$ is a constant, $\gamma \in \mathbb{R}_{\geq 0}$, and a word has $\Theta(\log(s \cdot p))$ bits. Thus, the total space in the system is only $O(n^{\gamma})$ factor more than the input size n, and each machine has local memory size sublinear in n. When $0 \leq \gamma \leq O(1/\log n)$, the total space is just linear in the input size. The computation proceeds in rounds. At the beginning of the computation, the input is distributed on the local memory of $\Theta(n/s)$ input machines. Input machines and other machines are identical except that input machine can hold a part of the input in its local memory at the beginning of the computation while each of other machines initially holds nothing. In each round, each machine performs computation on the data in its local memory, and sends messages to other machines (including the sender itself when it wants to keep the data) at the end of the round. Although any two machines can communicate directly in any round, the total size of messages (including the self-sent messages) sent or received of a machine in a round is bounded by s, its local memory size. In the next round, each machine only holds the received messages in its local memory. At the end of the computation, the output is distributed on the output machines. Output machines and other machines are identical except that output machine can hold a part of the output in its local memory at the end of the computation while each of other machines holds nothing. We call the above model $(\gamma, \delta) - \text{MPC}$ model. The model is exactly the same as the model MPC(ϵ) defined by [BKS13] with $\epsilon = \gamma/(1 + \gamma - \delta)$ and the number of machines $p = O(n^{1+\gamma-\delta})$. Since we care more about the total space used by the algorithm, we use (γ, δ) to characterize the model, while in [BKS13] they use parameter

ϵ to characterize the repetition of the data. The main complexity measure is the number of rounds R required to solve the problem.

32.7.1 Basic MPC Algorithms

Sorting One of the most important algorithms in MPC model is sorting. The following theorem shows that there is an efficient sorting algorithm.

Theorem 32.7.1 ([GSZ11, Goo99]). *Sorting can be solved in c/δ rounds in $(0,\delta)-$MPC model for any constant $\delta \in (0,1)$, where $c \geq 0$ is a universal constant. Precisely, there is an algorithm $\mathcal{A}$ in $(0,\delta)-$MPC model such that for any set S of n comparable items stored $O(n^\delta)$ per machine on input machines, $\mathcal{A}$ can run in c/δ rounds and leave the n items sorted on the output machines, i.e. the ouput machine with smaller index holds a smaller part of $O(n^\delta)$ items.*

Notice that for any $\delta' \geq \delta$, $O(1)$ number of machines with $\Theta(n^{\delta'})$ memory can always simulate the computation of $O(n^{\delta'-\delta})$ number of machines with $\Theta(n^\delta)$ memory. Thus, if an algorithm $\mathcal{A}$ can solve a problem in $(\gamma,\delta)-$MPC model in $R(n)$ rounds, then $\mathcal{A}$ can be simulated in $(\gamma',\delta')-$MPC model still in $R(n)$ rounds with all $\gamma' \geq \gamma, \delta' \geq \delta$.

Indexing In the indexing problem, a set $S = \{x_1, x_2, \cdots, x_n\}$ of n items are stored $O(n^\delta)$ per machine on input machines. The output is

$$S' = \{(x,y) \mid x \in S, y-1 \text{ is the number of items before } x\}$$

of n pairs stored $O(n^\delta)$ per machine on output machines. Here, "an item $x' \in S$ is before $x \in S$" means that x' is held by a input machine with a smaller index, or x', x are stored in the same input machine but x' has a smaller local memory address.

Prefix sum In the prefix sum problem, a set $S = \{(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)\}$ of n (item, number) pairs are stored $O(n^\delta)$ per machine on input machines. The output is

$$S' = \left\{ (x, y') \;\middle|\; (x, y) \in S, y' - y = \sum_{(\widetilde{x},\widetilde{y}) \text{ is before } (x,y)} \widetilde{y} \right\}$$

of n pairs stored $O(n^\delta)$ per machine on output machines. Here, "an pair $(\widetilde{x}, \widetilde{y}) \in S$ is before $(x, y) \in S$" means that $(\widetilde{x}, \widetilde{y})$ is held by a input machine with a smaller index, or $(\widetilde{x}, \widetilde{y}), (x, y)$ are stored in the same input machine but $(\widetilde{x}, \widetilde{y})$ has a smaller local memory address. Notice that indexing problem is a special case of prefix sum problem.

Theorem 32.7.2 ([GSZ11]). *Indexing/prefix sum problem can be solved in c/δ rounds in $(0, \delta) - \text{MPC}$ model for any constant $\delta \in (0, 1)$, where $c \geq 0$ is a universal constant.*

Once each item has an index, it is able to reallocate them onto the machines.

Load balance Sometimes, local computations of a machine may generate new data. When some machines are not able to keep the new data generated, we need to do loading balance. Fortunately, this operation can be done in constant number of rounds of computations.

For arbitrary constant $\delta \in (0, 1)$, we are able to spend constant number of rounds to reallocate the data in $(0, \delta) - \text{MPC}$ model such that if a machine is not empty, the size of its local data is at least n^δ/k and is at most $2n^\delta/k$ where $k > 1$ is an arbitrary constant. The method is very simple, we can use the algorithm mentioned in Theorem 32.7.2 to index each data item, and then send them to the corresponding machine.

Predecessor In the predecessor problem, a set $S = \{(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)\}$ of n (item, 0/1) pairs are stored $O(n^\delta)$ per machine on input machines. The output machines are all input machines. If an input (also output) machine holds a tuple $(x_i, y_i) \in S$ at the beginning of the computation, then at the end of the computation, that machine should still hold the tuple (x_i, y_i). In addition, if an input (also output) machine holds a tuple $(x, 0) \in S$ at the beginning of the computation, then at the end of the computation, that machine should hold a tuple (x, x') such that $(x', 1) \in S$, and $(x', 1)$ is the last tuple occurred before $(x, 0)$. Here, "$(x', 1)$ is before $(x, 0)$" means that $(x', 1)$ is held by a input machine with a smaller index, or $(x', 1), (x, 0)$ are stored in the same input machine but $(x', 1)$ has a smaller local memory address.

Theorem 32.7.3 ([GSZ11])**.** *Predecessor problem can be solved in c/δ rounds in $(0, \delta)-$*MPC *model for any constant $\delta \in (0, 1)$, where $c \geq 0$ is a universal constant.*

Roughly speaking the algorithm is as the following: firstly, build a $\Theta(n^\delta)$ branching tree on the machines, then follows by bottom-up stages to collect the last $(x_l, 1)$ tuple in each large interval and then follows by top-down stages to compute the predecessors of every prefix. For completeness, we describe the algorithm for predecessor problem in the following:

Predecessor Algorithm:

- **Setups:**
  - There are $2p = \Theta(n^\delta)$ machines indexed from 1 to $2p$ each with local memory size $s = \Theta(n^\delta)$. The machine with index from $p+1$ to $2p$ are input/output machines.
  - $(x_1, y_1), \cdots (x_n, y_n)$ are stored on input/output machine $p+1$ to $2p$, where $\forall i \in [n], y_i \in \{0, 1\}$.
  - The goal: If an input machine holds a tuple (x, y) with $y = 0$, then it will create a tuple (x, x') at the end of the computation, where (x', y') is the last tuple with $y' = 1$ stored before (x, y).
- **Bottom-up stage ($O(1/\delta)$ constant rounds):**
  - Let $d = s/10$ be the branching factor.
  - In the i^{th} round, each machine j with j in the range $\lfloor p/d^{i-1} \rfloor + 1$ to $\lfloor (2p-1)/d^{i-1} \rfloor + 1$ sends the last (x_l, y_l) tuple with $y_l = 1$ in its local memory to machine $\lfloor (j-1)/d \rfloor + 1$. If machine j does not have any tuple with $y_l = 1$, it just sends an arbitrary tuple to machine $\lfloor (j-1)/d \rfloor + 1$.
  - Until the end of the computation, machine j sends itself messages to keep the data. The stage ends when machine 1 receives messages.
- **Top-down stage ($O(1/\delta)$ constant rounds):**
  - Let $d = s/10$ be the branching factor.
  - In the i^{th} round, each machine j with j in the range $\lfloor d^{i-2} \rfloor + 1$ to $\min(d^{i-1}, p)$ sends to each machine h in the range $(j-1)d+1$ to $\min(j \cdot d, 2p)$ a tuple (x_l, y_l) which is the last tuple with $y_l = 1$ appeared before machine h.
  - The stage ends when machine $2p$ receives messages.
- **The last round:**
  - Machine $i \in \{p+1, \cdots, 2p\}$ scans its local memory, for each tuple (x, y) with $y = 0$, create a tuple (x, x') where (x', y') is the last tuple stored before (x, y) with $y' = 1$.

32.7.2 Data Organization

In this section, we introduce the method to organize the data in the system.

Set Let $S = \{x_1, x_2, \cdots, x_m\}$ be a set of m items, and each item x_i can be described by $O(1)$ number of words. If $x \in S$ is equivalent to that there is a unique machine which holds a pair ("S", x) in its local memory, then we say that S is stored in the system. Here "S" is the name of the set S and can be described by $O(1)$ number of words.

Let $\mathcal{S} = \{S_1, S_2, \cdots, S_m\}$ be a set of m sets, where $\forall i \in [m]$, S_i is stored in the system, and the name "S_i" of each set S_i can be described by $O(1)$ number of words. If $S \in \mathcal{S}$ is equivalent to that there is a unique machine which holds a pair ("$\mathcal{S}$", "S") in its local memory, then we say $\mathcal{S}$ is stored in the system. Here "$\mathcal{S}$" is the name of $\mathcal{S}$ and can be described by $O(1)$ number of words.

Let S be a set stored in the system. If machine i has a pair ("S", x), then we say that the element x of S is held by the machine i. If every element of S is held by a machine with index in $\{i, i+1, \cdots, j\}$, then we say S is stored on the machine i to the machine j.

The total space needed to store S is $\Theta(m)$.

Mapping Let $f : U \to H$ be a mapping from a finite set U to a set H. In the following, we show how to use a set to represent a mapping.

Definition 32.7.1 (Set representation of a mapping)**.** Let $f : U \to H$ be a mapping from a finite set U to a set H. Let $S = \{(x, y) \mid x \in U, y = f(x)\}$. then the set S is a set representation of the mapping f.

Let U be a finite set where each element of U can be described by $O(1)$ number of words. Let S be a set representation of the mapping $f : U \to H$. If S is stored in the system, then we say f is stored in the system. If S is stored on the machine i to the machine j, then f is stored on the machine i to the machine j. At any time of the system, there can be at most one set representation S of f stored in the system. Furthermore, the name of S is "f" which is the same as the name of mapping f, and can be described by $O(1)$ number of words.

The total space needed to store f is the total space needed to store S, and thus is $\Theta(|U|)$.

Sequence Let $A = (a_1, a_2, \cdots, a_m)$ be a sequence of m elements. In the following, we show how to use a set to represent a sequence.

Definition 32.7.2 (Set representation of a sequence)**.** Let $A = (a_1, a_2, \cdots, a_m)$ be a sequence of n elements. If a set $S = \{(x_1, y_1), (x_2, y_2), \cdots, (x_m, y_m)\} \subseteq \mathbb{R} \times \{a_1, a_2, \cdots, a_m\}$ satisfies $x_1 < x_2 < \cdots < x_m, y_1 = a_1, y_2 = a_2, \cdots, y_m = a_m$, then the set S is a set representation of the sequence A. Furthermore, if $x_1 = 1, x_2 = 2, \cdots, x_m = m$, then S is a standard set representation of A.

Let A be a sequence of elements where each element can be described by $O(1)$ number of words. Let S be a set representation of the sequence A. If S is stored in the system, then we say A is stored in the system. If S is stored on the machine i to the machine j, then A is stored on the machine i to the machine j. At any time of the system, there can be at most one set representation S of A stored in the system. Furthermore, the name of S is

"A" which is the same as the name of sequence A, and can be described by $O(1)$ number of words.

The total space needed to store A is the total space needed to store S, and thus is $\Theta(m)$.

32.7.3 Set Operations

In this section, we introduce some MPC model operations for sets.

Duplicates removing There are n tuples stored in the machines. But there are some duplicates of them. The goal is to remove all the duplicates. To achieve this, we can just sort all the tuples. After sorting, if a tuple is different from its previous tuple, then we keep it. Otherwise, we remove the tuple.

Sizes of sets Suppose we have k sets $S_1, S_2, \cdots, S_k$ stored in the system. Our goal is to get the sizes of all the sets. We can firstly sort all the tuples such that the tuples from the same set are consecutive. Then we can calculate the index of each tuple. Every machine can scan all the tuples in its local memory, if x is an element of set S_i and has the smallest/largest index y, then create a pair ("boundary of 'S_i' ", y). Then we sort all the created pairs, then for each set S_i, there are two pairs ("boundary of 'S_i' ", y_1), ("boundary of 'S_i' ", y_2) stored on the same machine. Each machine can store its local memory. For each pair of tuples ("boundary of 'S_i' ", y_1), ("boundary of 'S_i' ", y_2) with $y_1 < y_2$, the machine can generate a new tuple ("f", ("S_i", $y_2 - y_1 + 1$)). Finally, there will be a mapping f stored in the system, where $f(S_i) = |S_i|$. Thus, the total number of rounds is a constant.

Copies of sets Suppose we have k sets $S_1, S_2, \cdots, S_k$ stored in the system. Let $s_1, s_2, \cdots, s_k \in \mathbb{Z}_{\geq 1}$. If a machine holds an element $x \in S_i$, then the machine knows the value of s_i. Our goal is to create sets $S_{1,1}, S_{1,2}, \cdots, S_{1,s_1}, S_{2,1}, S_{2,2}, \cdots, S_{2,s_2}, \cdots, S_{k,s_k}$ and make them stored in the system, where $S_{i,j}$ is a copy of S_i.

The idea is very simple: for an element $x \in S_i$, we need to make s_i copies $(\text{“}S_{i,1}\text{”}, x), (\text{“}S_{i,2}\text{”}, x), \cdots, (\text{“}S_{i,s_i}\text{”}, x)$ of tuple $(\text{“}S_i\text{”}, x)$. But the issue is that s_i may be very large such that it is not able to generate all the copies of a tuple on a single machine. For the above reason, we implement it in three steps: firstly we compute the new "position" of each original tuple among all the copies, then send the original tuples to their new "positions", and finally filling the gap by generating copies between any two adjacent original tuples. Precisely, each machine can scan its local memory, and assign each tuple $(\text{“}S_i\text{”}, x)$ a weight s_i. Then we can use prefix sum algorithm (See Theorem 32.7.2) to compute the prefix sum of each tuple $(\text{“}S_i\text{”}, x)$. The prefix sum $\text{pos}(\text{“}S_i\text{”}, x)$ of a tuple $(\text{“}S_i\text{”}, x)$ denotes the new "position" of the last copy of this tuple when all the copies are generated. Let $n = \sum_{i=1}^{k} s_i \cdot |S_i|$. Let machine 1 to t be t empty machines each maintains $s/10$ "positions", i.e. machine 1 has "positions" 1 to $s/10$, machine 2 has "positions" $s/10 + 1$ to $2s/10$, and so on. Let $t \cdot s/10 = \Theta(n)$. The machine which holds tuple $(\text{“}S_i\text{”}, x)$ sends the tuple $(\text{“}S_i\text{”}, x)$ to the "position" $\text{pos}(\text{“}S_i\text{”}, x) - s_i + 1$, and sends the tuple $(\text{“}S_{i,s_i}\text{”}, x)$ to the "position" $\text{pos}(\text{“}S_i\text{”}, x)$. Then each machine $i \in [t]$ scans its "positions". If a "position" received a tuple, the machine marks that "position" as "1". Otherwise, the machine marks that position as "0". Now we can apply the predecessor algorithm (See Theorem 32.7.3) such that each empty "position" learns its predecessor tuple. If the predecessor tuple of an empty "position" l is $(\text{“}S_i\text{”}, x)$, and the predecessor tuple is at "position" l', then create a tuple $(\text{“}S_{i,l-l'}\text{”}, x)$ at this empty position. Thus, at the end of all the computations, $S_{1,1}, S_{1,2}, \cdots, S_{1,s_1}, S_{2,1}, S_{2,2}, \cdots, S_{2,s_2}, \cdots, S_{k,s_k}$ are stored on the system.

Indexing elements in sets Suppose we have k sets $S_1, S_2, \cdots, S_k$ stored in the system. The goal is to compute a mapping f such that $\forall i \in [k], x \in S_i$, x is the $f(S_i, x)^{\text{th}}$ element of

S_i.

To achieve this goal, we can sort (See Theorem 32.7.1) all the tuples such that the elements from the same set are stored consecutively on several machines. Then we can run indexing algorithm (See Theorem 32.7.2) to compute the global index of each tuple. In the next, each machine scans its local data. If $(\text{“}S_i\text{”}, x)$ is in the local memory, and x is the first element of S_i, then the machine marks this tuple as “1”. For other tuples in the local memory, the machine marks them as “0”. Then we can invoke predecessor algorithm (See Theorem 32.7.3) on all the tuples. At the end of the computation, each machine scans its all tuples. For a tuple $(\text{“}S_i\text{”}, x)$ with global index l, the machine determine the index of x in S_i based on the global index l' of its predecessor $(\text{“}S_i\text{”}, x)$. Precisely, the machine creates a tuple $(\text{“}f\text{”}, ((\text{“}S_i\text{”}, x), l - l' + 1))$ stored in the memory. Thus at the end of the computation, the desired mapping f is stored in the system.

Set merging Suppose we need to merge several sets $S_1, S_2, \cdots, S_k$ stored on the system, i.e. create a new set $S = \bigcup_{i=1}^{k} S_i$. To implement this operation, each machine scans its local memory. If there is a tuple $(\text{“}S_i\text{”}, x)$ in its memory, then it creates a tuple $(\text{“}S\text{”}, x)$. Finally, we just need to remove all the duplicates.

Set membership Suppose we have k sets $S_1, S_2, \cdots, S_k$ stored in the system. There is an another set $Q = \{(x_1, y_1), \cdots, (x_q, y_q)\}$ also stored in the system where x_i is the name of a set S, and y_i is an item. The goal is to answer whether y_i is in S.

To achieve this, we can firstly sort all the tuples. For tuple with form $(\text{“}S_i\text{”}, x)$, the first key is S_i, the second key is x, and the third key is $-\infty$ which has the highest priority.

For tuple with form $(\text{“}Q\text{”}, (x, y))$, the first key is x, the second key is y, and the third key is ∞ which has the lowest priority. The comparison in the sorting procedure firstly compare the first key, then the second key, and finally the third key. After sorting, for each tuple with form $(\text{“}S_i\text{”}, x)$, we mark it as “1”. For each tuple with form $(\text{“}Q\text{”}, (x, y))$, we mark it as “0”. Now we can apply the predecessor algorithm (See Theorem 32.7.3). For each tuple $(\text{“}Q\text{”}, (x, y))$, if its predecessor is $(\text{“}S\text{”}, y)$ where x is the name of “S”, then we create a tuple $(\text{“}f\text{”}, ((x, y), 1))$; Otherwise, we create a tuple $(\text{“}f\text{”}, ((x, y), 0))$. Thus, at the end of the computation, there is a mapping f stored on the system such that for each $(x, y) \in Q$, if x is the name of some set S_i, and $y \in S_i$, then $f(x, y) = 1$; Otherwise $f(x, y) = 0$.

32.7.4 Mapping Operations

In this section, we introduce some MPC model operations for mapping. The most important operation is called Multiple queries.

Multiple queries We have k sets $S_1, S_2, \cdots, S_k$ stored in the system. Without loss of generality, $S_1, S_2, \cdots, S_t$ ($t \leq k$) are sets representations of mappings (See Definition 32.7.1) $f_1 : U_1 \to H_1, f_2 : U_2 \to H_2, \cdots, f_t : U_t \to H_t$ respectively. When a machine does local computation, it may need to query some values which are in the form $f_i(u)$ for some $u \in U_i$. The following lemma shows that we can answer all the such queries simultaneously in constant number of rounds in $(0, \delta) - \text{MPC}$ model for all constant $\delta \in (0, 1)$. It means that we can use constant number of rounds to simulate concurrent read operations on a shared memory where $S_1, \cdots, S_k$ are stored in the shared memory.

Lemma 32.7.4 (Multiple queries)**.** *Let $\delta \in (0, 1)$ be an arbitrary constant. There is a constant number of rounds algorithm $\mathcal{A}$ in $(0, \delta) - \text{MPC}$ model which satisfies the following properties. The input of $\mathcal{A}$ contains two parts. The first part are k sets $S_1, S_2, \cdots, S_k$ stored (See Section 32.7.2 for data organization of sets) on the input machines, where $S_1, S_2, \cdots, S_t$ ($t \leq k$) are sets representations of mappings (See Definition 32.7.1) $f_1 : U_1 \to H_1, f_2 : U_2 \to H_2, \cdots, f_t : U_t \to H_t$ respectively. The second part is a set*

$$Q = \{(x_1, y_1, z_1), (x_2, y_2, z_2), \cdots, (x_q, y_q, z_q)\}$$

stored on the input machines, where $\forall (x, y, z) \in Q$, x is the name "f_i" of the mapping f_i for some $i \in [t]$, y is an element in U_i, and z is the index of the input machine which holds the element (x, y, z) of Q. The total input size $n = |Q| + \sum_{i=1}^{k} |S_i|$. The output machines

are all the input machines. $\forall i \in [k], x \in S_i$*, if the element* x *of* S_i *is held by the input (also output) machine* j*, then at the end of the computation, the element* x *of* S_i *should still be held by the output (also input) machine* j*. Let* Q' *be the set* $\{(x, y, z, w) \mid \exists (x, y, z) \in Q, w = f_i(y)$*, where* x *is the name of* $f_i\}$*. At the end of the computation,* Q' *is stored on the output (also input) machines such that* $\forall (x, y, z, w) \in Q'$*, the element* (x, y, z, w) *of* Q' *is held by the machine* z*.*

Proof. The idea is that we can firstly use sorting (See Theorem 32.7.1) to make queries and the corresponding values be stored consecutively in several machines. The issue remaining is that there may be many queries queried the same position such that some queries may not be stored in the machine which holds the corresponding value. In this case, we need to find the predecessor by invoking the algorithm shown in Theorem 32.7.3. □

The Multiple queries algorithm is shown as the following:

Multiple Queries Algorithm:

- **Setups:**
  - There are $3p = \Theta(n^\delta)$ machines indexed from 1 to $3p$ each with local memory size $s = \Theta(n^\delta)$.
  - The machine with index from $2p+1$ to $3p$ are input/output machines.
  - Sets $S_1, S_2, \cdots, S_k, Q$ are stored on machine $2p+1$ to $3p$. ▷ Corresponding to Lemma 32.7.4
- **The first round:**
  - Machine $i \in \{2p+1, \cdots, 3p\}$ scans its local memory, and send all the tuples with form $(\text{“}f_j\text{”}, (x, y))$ or $(\text{“}Q\text{”}, (x, y, z))$ to machine $i - p$, where “f_j” is the name of f_j (also S_j) for $j \in [t]$. Until the end of the computation, machine i sends itself messages to keep its local data.
- **Using constant number ($O(1/\delta)$) of rounds to sort:**
  - Use machine 1 to $2p$ to sort all the tuples stored on machine $p+1$ to $2p$, and thus at the end of this stage, machine $p+1$ to $2p$ holds sorted tuples. For tuple with the form $(\text{“}f_j\text{”}, (x, y))$, the first key value is “f_j”, the second key value is x and the third key value is $-\infty$ which is the highest priority. For tuple with form $(\text{“}Q\text{”}, (x, y, z))$, the first key value is x, the second key value is y, and the third key value is ∞ which is the lowest priority. The comparison in the sorting is: Firstly compare the first key. If they are the same, then compare the second key. If they are still the same, compare the third key.
- **Using constant number ($O(1/\delta)$) of rounds to find predecessors:**
  - Machine $p+1$ to $2p$ scans its local memory. For a tuple in the form $(\text{“}f_j\text{”}, (x, y))$, the machine marked it as “1”. For a tuple in the form $(\text{“}Q\text{”}, (x, y, z))$, the machine marked it as “0”.
  - Machine 1 to $2p$ together invoke the Predecessor algorithm (Theorem 32.7.3), where the input is on machine $p+1$ to machine $2p$.
- **The last round:**
  - Machine $p+1$ to $2p$ scans its local memory. For each tuple with form $(\text{“}Q\text{”}, (x, y, z))$, it sends machine z a tuple $(\text{“}Q'\text{”}, (x, y, z, w))$, where x is the name of f_j, and $w = f_j(y)$.

32.7.5 Sequence Operations

In this section, we introduce some MPC model operations for sequence.

Sequence standardizing Suppose there is a sequence A, and one of its set representation (see Definition 32.7.2) S is stored in the system. The goal is to modify the set S such that S is a standard set representation of A.

We can compute the index (see **Indexing elements in sets** in Section 32.7.3) of elements in S. Then for each element $(x, y) \in S$, we can query (see **Multiple queries** in Section 32.7.4) the index of (x, y) in S. Suppose the index is i, we modify the tuple $(\text{“}S\text{”}, (x, y))$ to $(\text{“}S\text{”}, (i, y))$.

Sequence duplicating Suppose there is a sequence $A = (a_1, a_2, \cdots, a_s)$, and one of its set representation (see Definition 32.7.2) S is stored in the system. Furthermore, there is a mapping $f : [s] \to \mathbb{Z}_{\geq 0}$ which is also stored in the system. The goal is to get a set S' stored in the system such that S' is a set representation of the sequence:

$$(\underbrace{a_1, a_1, \cdots, a_1}_{f(1) \text{ times}}, \underbrace{a_2, a_2, \cdots, a_2}_{f(2) \text{ times}}, \cdots, \underbrace{a_s, a_s, \cdots, a_s}_{f(s) \text{ times}}).$$

Firstly, we can standardize (see the above paragraph **Sequence standardizing**) the set S. Then for each tuple $(\text{“}S\text{”}, (i, a_i))$, we create a tuple $(\text{“}S_i\text{”}, a_i)$, and we can query (see **Multiple queries** in Section 32.7.4) the value of $f(i)$. Then we can copy (see **Copies of sets** in Section 32.7.3) set S_i $f(i)$ times. For each tuple $(\text{“}S_{i,j}\text{”}, a_i)$, we create a tuple $(\text{“}S'\text{”}, ((i, j), a_i))$. Then we can compute the index (see **Indexing elements in sets** in Section 32.7.3) of each element in S'. For each tuple $(\text{“}S'\text{”}, ((i, j), a_i))$, we can query

(see **Multiple queries** in Section 32.7.4) the index i' of it, and then modify the tuple as $(\text{“}S\text{”}, (i', a_i))$.

Sequence insertion Suppose there are $k+1$ sequences $A = (a_1, a_2, \cdots, a_s), A_1, \cdots, A_k$ which have sets representations (see Definition 32.7.2) $S, S_1, \cdots, S_k$ respectively and stored on the system. There is also a mapping $f : [k] \to \{0\} \cup [s]$ stored on the system where $\forall i \neq j \in [k], f(i) \neq f(j)$. The goal is to insert each sequence A_i into the sequence A, and A_i should be between the element $a_{f(i)}$ and $a_{f(i)+1}$.

Firstly, we can standardize (see **Sequence standardizing** in Section 32.7.3) S. Then we can compute the total size (see **Sizes of sets** in Section 32.7.3) $N = |S|+|S_1|+\cdots+|S_k|+1$. For each tuple $(\text{“}S\text{”}, (i, a_i))$, we can modify it as $(\text{“}S\text{”}, (i \cdot N, a_i))$. For each tuple $(\text{“}S_i\text{”}, (j, a_{ij}))$, we query (see **Multiple queries** in Section 32.7.4) the value of $f(i)$, then create a tuple $(\text{“}S\text{”}, (f(i) \cdot N + j, a_{ij}))$.

32.7.6 Multiple Tasks

In this section, we show that if the entire computational tasks consist of some independent small computational tasks, then we are able to schedule the machines such that the small computational tasks can be computed simultaneously.

Task and multiple tasks problem A computational task here is running a specific algorithm on specific input data.

There are k sets $S_1, S_2, \cdots, S_k$ stored in the system. Let $n = \sum_{i=1}^{k} |S_i|$ be the total input size. There are h independent computational tasks $T_1, T_2, \cdots, T_h$. Each task T_i needs to take some sets $\mathcal{S}_i \subseteq \{S_1, S_2, \cdots, S_k\}$ as its input, and is running a $(\gamma_i, \delta_i) - \text{MPC}$ algorithm in r_i rounds where $\gamma_i \in \mathbb{R}_{\geq 0}$, constant $\delta_i \in (0, 1)$. $\forall i \in [h]$, let $n_i = \sum_{S \in \mathcal{S}_i} |S|$ be the input size of task T_i. Without loss of generality, we can assume that the input of different tasks are disjoint. Otherwise we can use sets copying technique (See Section 32.7.3) to generate different copies of input sets for the tasks shared the same input set. The goal here is to use the small number of rounds to finish all the tasks. Since we can always use sorting and indexing to extract the desired input data. The most naive way is to compute the tasks one-by-one. This can be trivially done in $r = O(\sum_{i=1}^{h} r_i)$ rounds in $(\gamma, \delta) - \text{MPC}$ model for $\gamma = \log_n(h) + \max_{i \in [h]} \gamma_i, \delta = \max_{i \in [h]} \delta_i$. Here we show how to compute all the tasks simultaneously in $r = O(\max_{i \in [h]} r_i)$ rounds in $(\gamma, \delta) - \text{MPC}$ model for $\gamma = \log_n(m) - 1, \delta = \max_{i \in [h]} \delta_i$, where $m = \Theta(n + \sum_{i=1}^{h} n_i^{1+\gamma_i})$.

Each machine scans its local memory. If the machine holds a tuple $(\text{“}S_i\text{”}, x)$, and S_i is a part of input of task T_j, then it creates a tuple $(\text{“}W_j\text{”}, (\text{“}S_i\text{”}, x))$. Thus, at the end of this step, there are additional h sets $W_1, W_2, \cdots, W_h$ stored in the system. Here $W_i, i \in [h]$

contains all the information of input data of task T_i. Then we can compute a mapping f such that $\forall i \in [h], f(W_i) = |W_i|$ (see Section 32.7.3). Thus, we know the input size of each task. Then each machine scans its local memory. If the machine holds a tuple $(\text{“}f\text{”}, (\text{“}W_i\text{”}, |W_i|))$, then it creates a tuple $(\text{“}H_i\text{”}, |W_i|)$, i.e. a set $H_i = \{|W_i|\}$. Then for each set $H_i = \{|W_i|\}, i \in [h]$, we can copy (see Section 32.7.3) it $s_i = c \cdot |W_i|^{1+\gamma_i}$ times for a sufficiently large c to get sets $H_{i,1} = H_{i,2} = \cdots = H_{i,s_i} = |W_i|$. Each set $H_{i,j}$ is just a placeholder of one unit working space of the task T_i. Thus, the number of copies of the set H_i is the total space needed for the task T_i. We can sort all the tuples $(\text{“}H_{i,j}\text{”}, |W_i|)$ on machines with index in $I = \{2, 5, 8, 11, \cdots, 3p-1\}$, where local memory $s = \Theta(n^\delta)$, total required memory $m = \Theta(n + \sum_{i=1}^{h} n_i^{1+\gamma_i})$, and $p = \Theta(m/s)$ For each machine with index $q \in I$, the tuples on that machine must be in the following form

$$(\text{“}H_{i,j}\text{”}, |W_i|), (\text{“}H_{i,j+1}\text{”}, |W_i|), \cdots, (\text{“}H_{i,s_i}\text{”}, |W_i|), (\text{“}H_{i+1,1}\text{”}, |W_{i+1}|), \cdots, (\text{“}H_{i+1,s_{i+1}}\text{”}, |W_{i+1}|),$$
$$(\text{“}H_{i+2,1}\text{”}, |W_{i+2}|), \cdots, (\text{“}H_{i+2,s_{i+2}}\text{”}, |W_{i+2}|), \cdots, (\text{“}H_{i',1}\text{”}, |W_{i'}|), \cdots (\text{“}H_{i',j'}\text{”}, |W_{i'}|).$$

Then machine q just sends all the tuples $(\text{“}H_{i,j}\text{”}, |W_i|), (\text{“}H_{i,j+1}\text{”}, |W_i|), \cdots, (\text{“}H_{i,s_i}\text{”}, |W_i|)$ to machine $q-1$, and sends all the tuples $(\text{“}H_{i',1}\text{”}, |W_{i'}|), (\text{“}H_{i',2}\text{”}, |W_{i'}|), \cdots (\text{“}H_{i',j'}\text{”}, |W_{i'}|)$ to machine $q+1$. Thus, $\forall i \in [h]$,

1. either all the $H_{i,1}, H_{i,2}, \cdots, H_{i,s_i}$ are stored on consecutive machines, machine q to machine q', and any of machine q to machine q' does not hold other tuples,
2. or there is a unique machine q which holds all the sets $H_{i,1}, H_{i,2}, \cdots, H_{i,s_i}$.

For each machine $q \in [3p]$, if $H_{i,1}$ is held by machine q, then it creates a tuple $(\text{“st”}, (\text{“}T_i\text{”}, q))$. If H_{i,s_i} is held by machine q, then it creates a tuple $(\text{“ed”}, (\text{“}T_i\text{”}, q))$. The mapping st, ed

then are stored in the system, where $\mathrm{st}(T_i)$ is the index of the first machine assigned to task T_i, and $\mathrm{ed}(T_i)$ is the index of the last machine assigned to task T_i. Recall that W_i contains all the information of the input data to task T_i. The remaining task is to move the input data of task T_i to the machines with index from $\mathrm{st}(T_i)$ to $\mathrm{ed}(T_i)$. According to Section 32.7.3, we can compute a mapping f', such that $f'(W_i, x)$ records the index of $x \in W_i$ in set W_i. Now, each machine scans its local memory. For each tuple $(\text{“}W_j\text{”}, (\text{“}S_i\text{”}, x))$, the machine needs to query the value of $f'(W_j, (\text{“}S_i\text{”}, x))$, the value of $\mathrm{st}(T_j)$ and the value of $\mathrm{ed}(T_j)$. By Lemma 32.7.4, these queries can be handled simultaneously in constant number of rounds. Then the machine can send the tuple $(\text{“}S_i\text{”}, x)$ to the corresponding machine based on the value of $f'(W_j, (\text{“}S_i\text{”}, x))$, $\mathrm{st}(T_j)$, and $\mathrm{ed}(T_j)$. Finally, $\forall i \in [h]$, since $\delta \geq \delta_i$ and $(\mathrm{ed}(T_i) - \mathrm{st}(T_i) + 1) \cdot s = \Theta(n_i^{1+\gamma_i})$, the machines with index from $\mathrm{st}(T_i)$ to $\mathrm{ed}(T_i)$ can simulate task T_i in r_i number of rounds.

32.8 Implementations in MPC Model

In this section, we show how to implement all the previous batch algorithms in MPC model.

32.8.1 Neighbor Increment Operation

Lemma 32.8.1. *Let graph $G = (V, E), n = |V|, N = |V| + |E|$ and $m = \Theta(N^\gamma)$ for some arbitrary $\gamma \in [0, 2]$.* NeighborIncrement(m, G) *(Algorithm 32.1) can be implemented in $(\gamma, \delta) -$ MPC model for any constant $\delta \in (0, 1)$. Furthermore, the parallel running time is $O(r)$, where r is the number of iterations (see Definition 32.4.1) of* NeighborIncrement(m, G).

Proof. To implement line 7, we can create a tuple $(\text{“}S_v^{(0)}\text{”}, u)$ for each tuple $(\text{“}E\text{”}, (v, u))$. Then for each $(\text{“}S_v^{(0)}\text{”}, u)$ we can compute the index (see **Indexing elements in sets** and **Multiple queries**) of u in set $S_v^{(0)}$. If the index of u in set $S_v^{(0)}$ is at least $\lceil (m/n)^{1/2} \rceil$, then delete u from $S_v^{(0)}$, i.e. delete the tuple $(\text{“}S_v^{(0)}\text{”}, u)$.

Now let us discuss how to implement line 14 and line 17 in the i^{th} iteration. Firstly, we can compute the size of every set stored in the system (see **Sizes of sets**). Then for each tuple $(\text{“}S_v^{(i-1)}\text{”}, u)$, the corresponding machine queries (see **Multiple queries**) the size of $S_u^{(i-1)}$. If $|S_u^{(i-1)}| \geq \lceil (m/n)^{1/2} \rceil$, then create a tuple $(\text{“temp}_v^i\text{”}, u)$. We can index (see **Indexing elements in sets**) all the elements in set temp_v^i, and only keep the element with index 1. Thus, temp_v^i has a only element u, and we need to create a set $S_v^{(i)} = S_u^{(i-1)}$. Notice that there may be many $v \in V$ which needs need to implement $S_v^{(i)} = S_u^{(i-1)}$. Thus, for each tuple $(\text{“temp}_v^i\text{”}, u)$, we create a tuple $(\text{“target}_u^i\text{”}, v)$. $v \in \text{target}_u^i$ means that $S_v^{(i)}$ needs a copy of $S_u^{(i-1)}$. Thus, $|\text{target}_u^i|$ means that $S_u^{(i-1)}$ needs to copy $|\text{target}_u^i|$ times. For each tuple

(“$S_u^{(i-1)}$”, x), the machine queries (see **Multiple queries**) the size of target_u^i. Then each set $S_u^{(i-1)}$ can be copied (see **Copies of sets**) $|\text{target}_u^i|$ times. For each tuple (“target_u^i”, v), we query (see **Multiple queries**) the index (see **Indexing elements in sets**) of v in set target_u^i, and then create a tuple (“f^i”, ((“target_u^i”, x), v)), where x is the index of v in target_u^i. Thus f^i is a mapping such that $f^i(\text{target}_u^i, x)$ is the x^{th} element in target_u^i. For each tuple (“$S_{u,j}^{(i-1)}$”, x), we query (see **Multiple queries**) the value $v = f^i(\text{target}_u^i, j)$, and then create a tuple (“$S_v^{(i)}$”, x), and a tuple (“$S_v^{(i)}$”, v). We then remove the duplicates (see **Duplicates removing**) of elements of for every set $S_v^{(i)}$. For each tuple (“temp_v^i”, u), query (see **Multiple queries**) the size (see **Sizes of sets**) of $S_v^{(i)}$ and $S_u^{(i-1)}$. If $|S_v^{(i)}| > |S_u^{(i-1)}|$, then we create a tuple (“g^i”, (v, (u, “delete”))); Otherwise, create a tuple (“g^i”, (v, (u, “keep”))). Finally, for each tuple ($S_v^{(i)}, x$), we query (see **Multiple queries**) $(u, o) = g^i(v)$, if $u = x$ and $o =$ “delete", the machine deletes the tuple ($S_v^{(i)}, x$).

Next, let us discuss how to implement line 20. Similar as before, we can compute the size of every set stored in the system (see **Sizes of sets**). Then for each tuple (“$S_v^{(i-1)}$”, u), the corresponding machine queries (see **Multiple queries**) the size of $S_u^{(i-1)}$. If $|S_u^{(i-1)}| \geq \lceil (m/n)^{1/2} \rceil$, then create a tuple (“$\text{temp}_v^i$”, u). For each tuple (“V”, v), we can create a tuple (“temp_v^i”, null). Then for each tuple (“V”, v) we can query (see **Multiple queries**) the size (see **Sizes of sets**) of temp_v^i. If $|\text{temp}_v^i| = 1$, then we create a tuple (“f'^i”, 1); Otherwise, we create a tuple (“f'^i”, 0). Thus, mapping f'^i is stored in the system, and $f'^i(v) = 1$ if and only if $\forall u \in S_v^{(i-1)}, |S_u^{(i-1)}| < \lceil (m/n)^{1/2} \rceil$. For each tuple (“$S_v^{(i-1)}$”, u), we query (see **Multiple queries**) the value $f'^i(v)$. If $f'^i(v) = 1$, we create a tuple (“target_u^i”, v). Thus, $v \in \text{target}_u^i$ means that $S_u^{(i-1)}$ should be a part of $S_v^{(i)}$. $|\text{target}_u^i|$ means that $S_u^{(i-1)}$ needs to copy $|\text{target}_u^i|$ times. For each tuple (“$S_u^{(i-1)}$”, v), we query (see **Multiple queries**) the size (see **Sizes of**

sets) of target_u^i. Then we can copy (see **Copies of sets**) each set $S_u^{(i-1)}$ $|\text{target}_u^i|$ times. Then for each tuple $(\text{“target}_u^i\text{”}, v)$, we can query (see **Multiple queries**) the index x (see **Indexing elements in sets**) of v in set target_u^i, and then create a tuple $(\text{“}f^i\text{”}, ((\text{“target}_u^i\text{”}, x), v))$ which means that the x^{th} element of target_u^i is $f^i(\text{target}_u^i, x) = v$. For each tuple $(\text{“}S_{u,j}^{(i-1)}\text{”}, x)$, we query (see **Multiple queries**) the value $v = f^i(\text{target}_u^i, j)$, and then create a tuple $(\text{“}S_v^{(i)}\text{”}, x)$. We then remove the duplicates (see **Duplicates removing**) of elements of for every set $S_v^{(i)}$.

Finally, let us consider how to implement line 24. It is very simple, we only need to query the sizes of sets. For each tuple $(\text{“}V\text{”}, v)$, query (see **Multiple queries**) the size (see **Sizes of sets**) of $S_v^{(i)}$ and $S_v^{(i-1)}$, if v satisfies the condition, create a tuple $(\text{“}Done\text{”}, v)$. Every machine queries (see **Multiple queries**) the size (see **Sizes of sets**) of Done. If it is $|V|$, then all the machines know that they finish the loop. In the end, for each tuple $(S_v^{(r)}, u)$ we create tuples $(\text{“}E'\text{”}, (u, v)), (\text{“}E'\text{”}, (v, u))$, and for each tuple $(\text{“}E\text{”}, (u, v))$ we create tuple $(\text{“}E'\text{”}, (u, v))$. Then we then remove the duplicates (see **Duplicates removing**) of elements of E.

In the i^{th} iteration, we only need to maintain sets $V, E, S_v^{(i-1)}$. Since all the copy operation will create at most $n \cdot (m/n)^{1/2} \cdot (m/n)^{1/2} = m$ tuples, the total space needed is $\Theta(m)$ plus the space needed to maintain $V, E, S_v^{(i)}$. By Property 4 of Lemma 32.4.1, $|S_v^{(i)}| \leq m/n$. Thus, the total space is $\Theta(m) + |V| + |E| + \sum_{v \in V} |S_v^{(i)}| = \Theta(m) + N = \Theta(m)$.

The above implementation shows that the parallel time is $O(r)$, where r is the number of iterations (see Definition 32.4.1). □

32.8.2 Tree Contraction Operation

In this section, we show how to implement Algorithm 32.2 in MPC model.

Lemma 32.8.2. *Let graph $G = (V, E)$ and* par $: V \to V$ *be a set of parent points (see Definition 32.4.2) on the vertex set V.* TreeContraction(G, par) *(Algorithm 32.2) can be implemented in $(0, \delta) -$* MPC *model for any constant $\delta \in (0, 1)$. Furthermore, the parallel running time is $O(r)$, where r is the number of iterations (see Definition 32.4.5) of* TreeContraction(G, par).

Proof. Let $N = |V| + |E|$. Then the total space is $\Theta(N)$.

Initially, each machine scans its local memory. If there is a tuple $(\text{“}V\text{”}, v)$, then it queries the value of part(v). It needs $O(1)$ parallel time to answer all the queries (see Multiple queries in Lemma 32.7.4). Then the machine creates a tuple $(\text{“}g^{(0)}\text{”}, (v, \text{part}(v)))$. Thus, in the initialization stage, mapping $g^{(0)}$, par, set V, E are stored in the system.

In the l^{th} iteration, Each machine scans its local memory. If there is a tuple $(\text{“}V\text{”}, v)$, then it queries the value of $g^{(l-1)}(v)$. This can be done by Multiple queries. Then it queries the value of part$(g^{(l-1)}(v))$. This can also be done by Multiple queries. If $\text{part}(g^{(l-1)}(v)) = g^{(l-1)}(v)$, it creates a tuple $(\text{“Done”}, v)$. Then the machines can compute the sizes (see Section 32.7.3) of V and Done. Each machine queries the size of V and Done. This can be done by Multiple queries. Then if $|V| = |\text{Done}|$, every machine knows that the iterations are finished. Otherwise, the machine which holds $(\text{“}V\text{”}, v)$ queries the value of $g^{(l-1)}(g^{(l-1)})(v)$. This can be done by Multiple queries. And then it creates a tuple $(\text{“}g^{(l)}\text{”}, (v, g^{(l-1)}(g^{(l-1)})(v)))$.

At the end, if a machine holds a tuple $(\text{“}V\text{”}, v)$, then the queries part(v). If $v =$

part(v), it creates a tuple $(\text{“}V'\text{”}, v)$. If a machine holds a tuple $(\text{“}E\text{”}, (u, v))$, then it queries $g^{(r)}(u), g^{(r)}(v)$, and creates a tuple $(\text{“}E'\text{”}, (g^{(r)}(u), g^{(r)}(v)))$.

Since at the end of each iteration l, the system only stores mappings par : $V \to V, g^{(r)} : V \to V$, and sets V, E, the total space used is at most $O(N)$. Thus, we can implement the algorithm in $(0, \delta) -$ MPC model.

The total parallel time is $O(r)$. By Corollary 32.4.8, $r = O(\text{dep}(\text{par}))$. Thus, the total parallel time is $O(\text{dep}(\text{par}))$. □

32.8.3 Graph Connectivity

Theorem 32.8.3. *Let graph $G = (V, E), n = |V|, N = |V| + |E|$ and $m = \Theta(N^\gamma)$ for some arbitrary $\gamma \in [0, 2]$. Let $r > 0$ be a round parameter.* Connectivity(G, m, r) *(Algorithm 32.3) can be implemented in $(\gamma, \delta) -$ MPC model for any constant $\delta \in (0, 1)$. Furthermore, the parallel running time is $O(R)$, where R is the total number of iterations (see Definition 32.4.6) of* Connectivity(G, m, r).

Proof. Initially, we store sets V_0, E_0, V, E and mapping h_0 in the system. Now consider the i^{th} round. Due to Lemma 32.8.1, line 9 can be implemented in total space $\Theta(m)$ and with $O(k_i)$ parallel time, where k_i is the number of iterations (See Definition 32.4.1) of NeighborIncrement(m, G_{i-1}). To store V_i' and E_i', we need total space $\Theta(m)$. Line 10 can be implemented by operations described in **Sizes of sets** and **Multiple queries** (see Section 32.7). Line 11 can be implemented by the operations described in **Set membership** and **Multiple queries**. To implement line 14, for each tuple $(\text{“}V_i''\text{”}, v)$, we can create a tuple $(\text{“}l_i\text{”}, (v, x))$ where $x = 1$ with probability p_i, $x = 0$ with probability $1 - p_i$. To calculate p_i, the machine only needs to know n_{i-1}. This can be done by the operations described in **Sizes of sets** and **Multiple queries**. Line 15 and line 16 can be implemented by operations described in **Set membership** and **Multiple queries**. For line 17, set $L_i \cap (\Gamma_{G_i'}(v) \cup \{v\})$ can be computed by operations described in **Set membership** and **Multiple queries**. Then, by operations in **Indexing elements in sets** and **Multiple queries**, we can get $\min_{u \in L_i \cap (\Gamma_{G_i'}(v) \cup \{v\})} u$. Finally, by operation described in **Multiple queries**, $\forall v \in V_i''$ with $v \notin L_i$, the tuple $(\text{“par}_i\text{”}, (v, x))$ can be created, where $x = \min_{u \in L_i \cap (\Gamma_{G_i'}(v) \cup \{v\})} u$. Due to Lemma 32.8.2, line 18 can be implemented in total $\Theta(m)$ space and $O(r_i')$ parallel running

time, where r'_i is the number of iterations (see Definition 32.4.5) of TreeContraction(G''_i, par_i). Line 21 can be implemented by operations in **Set membership**, **Indexing elements in sets** and **Multiple queries**. Line 22 can be implemented by operations in **Set membership** and **Multiple queries**. Line 23 can be implemented by **Multiple queries**. For other $v \in V$ with $h_i(v) = \text{null}$ assigned by line 8, we can use the operations in **Set membership** and **Multiple queries** to find those v, and create a tuple $(\text{“}h_i\text{”}, v, \text{null})$.

Thus, in the i^{th} round, the parallel time needed is $O(k_i + r'_i)$. At the end of the i^{th} round, we only need to keep sets V_i, E_i, V, E and mapping h_i in the system. It will take total space at most $O(m)$.

Due to Lemma 32.8.2, line 26 can be implemented in at most $O(m)$ total space and $O(\log r)$ parallel time.

Thus, the total parallel time is $O(\log r + \sum_{i=1}^{r}(k_i + r'_i)) = O(\sum_{i=1}^{r}(k_i + r'_i))$. By definition 32.4.6, the total parallel time is $O(R)$, where R is the total number of iterations of CONNECTIVITY(G, m, r). The total space in the computation is always at most $\Theta(m)$. □

Here, we are able to conclude the following theorem for graph connectivity problem.

Theorem 32.8.4. *For any $\gamma \in [0, 2]$ and any constant $\delta \in (0, 1)$, there is a randomized $(\gamma, \delta) -$ MPC algorithm (see Algorithm 32.3) which can output the connected components for any graph $G = (V, E)$ in $O(\min(\log D \cdot \log(1/\gamma'), \log n))$ parallel time, where D is the diameter of G, $n = |V|$, $N = |V| + |E|$ and $\gamma' = (1 + \gamma) \log_n \frac{2N}{n^{1/(1+\gamma)}}$. The success probability is at least 0.98. In addition, if the algorithm fails, then it will return* FAIL.

Proof. The implementation of Algorithm 32.3 in MPC model is shown by Theorem 32.8.3.

The correctness of Algorithm 32.3 is proved by Theorem 32.4.9. The total parallel time of Algorithm 32.3 is proved by Theorem 32.4.15. □

32.8.4 Algorithms for Local Shortest Path Trees

In this section, we mainly explained how to implement local shortest path tree algorithms described in Section 32.5.1 and Section 32.5.2.

Lemma 32.8.5. *Let $G = (V, E)$ be an undirected graph, $s_1, s_2 \in \mathbb{Z}_{\geq 0}$, and $v \in V$. Let $\widetilde{T} = (V_{\widetilde{T}}, \mathrm{par}_{\widetilde{T}})$ with root v and radius s_1 be a local complete shortest path tree (see Definition 32.5.3) in G, and $\mathrm{dep}_{\widetilde{T}} : V_{\widetilde{T}} \to \mathbb{Z}_{\geq 0}$ be the depth of every vertex in $\widetilde{T}$. $\forall u \in V_{\widetilde{T}}$, let $T(u)$ with root u and radius s_2 be a local complete shortest path tree in G, and $\mathrm{dep}_{T(u)} : V_{T(u)} \to \mathbb{Z}_{\geq 0}$ be the depth of every vertex in $T(u)$. Then* TreeExpansion$(\widetilde{T}, \mathrm{dep}_{\widetilde{T}}, \{T(u) \mid u \in V_{\widetilde{T}}\}, \{\mathrm{dep}_{T(u)} \mid u \in V_{\widetilde{T}}\})$ *(Algorithm 32.4) can be implemented in $(0, \delta) -$ MPC model for any constant $\delta \in (0, 1)$ in $O(1)$ parallel time.*

Proof. For line 3, we apply operation shown in **Copies of sets** to copy each $V_{T(u)}$, then we can merge (see **Set merging**) all the copies to get $V_{\widehat{T}}$. To implement line 4 and line 5, we only need to apply the operation shown in **Multiple queries**. To implement line 6, for each tuple $(\text{“}V_{T(u)}\text{”}, x)$, we can firstly check whether $x \in V_{\widehat{T}} \setminus V_{\widetilde{T}}$ by operations described in **Set membership** and **Multiple queries**. If $x \in V_{\widehat{T}} \setminus V_{\widetilde{T}}$, then we can query the values of $\mathrm{dep}_{\widetilde{T}}(u)$ and $\mathrm{dep}_{T(u)}(x)$ by operations shown in **Multiple queries**. Then we create a tuple $(\text{“temp}_x\text{”}, (\mathrm{dep}_{\widetilde{T}}(u) + \mathrm{dep}_{T(u)}(x), u))$. By **Indexing elements in sets** and **Multiple queries**, we can find the element with the smallest index in set temp_x, and thus that element is $(\mathrm{dep}_{\widetilde{T}}(u_x) + \mathrm{dep}_{T(u_x)}(x), u_x)$. Finally, the remaining things in line 6 and line 7 can be done by the operations described by **Multiple queries**.

For all the operations, the total space is always linear. The parallel time needed for the above operations is also a constant. □

Lemma 32.8.6. *Let graph $G = (V, E), n = |V|, N = |V| + |E|$ and $m = \Theta(N^\gamma)$ for some arbitrary $\gamma \in [0, 2]$.* MultiRadiusLCSPT(G, m) *(Algorithm 32.5) can be implemented in $(\gamma, \delta) -$* MPC *model for any constant $\delta \in (0, 1)$. Furthermore, the parallel running time is $O(r)$, where r is the number of iterations (see Definition 32.5.4) of* MultiRadiusLCSPT(G, m).

Proof. To implement line 4 to line 6, we can scan all the tuples $(\text{“}E\text{”}, (u, v))$, then query the size of $\{v\} \cup \Gamma_G(v)$ and the size of $\{u\} \cup \Gamma_G(u)$, where these operations are described in **Sizes of sets** and **Multiple queries.** Then based on the sizes, we decide whether we need to create the corresponding tuples for $V_{T_0(v)}, V_{T_0(u)}, \text{par}_{T_0(u)}, \text{part}(T_0(v))$.

Now consider the main loop. We focus on the i^{th} round. Line 12 can be implemented by the operation described in **Multiple queries**. To implement line 13, for each tuple $(\text{“}V_{T_{i-1}(v)}\text{”}, u)$, we can query (see **Multiple queries**) whether $T_{i-1}(u)$ is null. If $T_{i-1}(u)$ is null, then we create a tuple $(\text{“temp}_v\text{”}, u)$. Then for each tuple $(\text{“}V\text{”}, v)$, we can query the size of temp_v by operations described in **Sizes of sets** and **Multiple queries**. If the size is not 0, then $T_i(v)$ must be null. Line 15 can be implemented by coping input for different tasks and running tasks in parallel, where it only needs operations shown in **Copies of sets**, **Multiple queries** and **Multiple Tasks** (see Section 32.7.6). According to Lemma 32.8.5, it only needs $O(1)$ parallel time. Line 16 and line 19 only need the operation shown in **Multiple queries**.

Thus, the total parallel time is $O(r)$ where r is the number of iterations (see Definition 32.5.4) of MultiRadiusLCSPT(G, m). For the total space, we stored the sets $V_{T_i(v)}$ for all $i \in [r], v \in V$ and mappings $\text{par}_{T_i(v)}, \text{dep}_{T_i(v)}$ for all $i \in [r], v \in V$. By Lemma 32.5.2, the total space to store all of them is at most $O(r \cdot n \cdot (m/n)^{1/4}) = O(m)$. In the i^{th} round of

the main loop, line 15 may make copies of the set. By Lemma 32.5.2, the input size of each task will be at most $O((m/n)^{1/4} \cdot (m/n)^{1/4})$. Since the there are at most n tasks, the total space needed is at most $O(m)$. □

Lemma 32.8.7. *Let graph $G = (V, E), n = |V|, N = |V| + |E|$ and $m = \Theta(N^\gamma)$ for some arbitrary $\gamma \in [0, 2]$.* MultipleLargeTrees(G, m) *(Algorithm 32.6) can be implemented in $(\gamma, \delta) -$* MPC *model for any constant $\delta \in (0, 1)$. Furthermore, the parallel running time is $O(r)$, where r is the number of iterations (see Definition 32.5.5) of* MultipleLargeTrees(G, m).

Proof. By Lemma 32.8.6, line 3 can be implemented in total space m and $O(r)$ parallel time where r is the number of iterations (see Definition 32.5.4) of MultiRadiusLCSPT(G, m). Line 4 to line 6 can be implemented by the operation described by **Multiple queries**. The implementation of line 7 to line 17 is similar as the implementation of the main loop of Algorithm 32.5 (See Lemma 32.8.6 for details of the implementation). The implementation of line 18 and line 19 only needs the operation described in **Indexing elements in sets** and **Multiple queries**. Line 22 can be implemented by copying input sets for different tasks and running multiple tasks in parallel, where the operations needed are described in **Copies of sets**, **Multiple queries** and **Multiple Tasks** (see Section 32.7.6). Line 24 to line 28 can be implemented by the operations described in **Copies of sets**, **Set membership**, **Indexing elements in sets**, and **Multiple queries**.

The total parallel time of the first loop is $O(r)$ since it has r rounds. The second loop can be done in one round. Thus the parallel time of the second loop is $O(1)$. Then the total parallel time is $O(r)$. Due to Lemma 32.8.6 and Lemma 32.5.6, r is the number of iterations (see Definition 32.5.5) of MultipleLargeTrees(G, m).

We stored all the $V_{T_i(v)}, V_{\widetilde{T}_i(v)}, \mathrm{par}_{T_i(v)}, \mathrm{par}_{\widetilde{T}_i(v)}, \mathrm{dep}_{T_i(v)}, \mathrm{dep}_{\widetilde{T}_i(v)}$ in the system. By Lemma 32.5.3 and Lemma 32.5.2, the total space needed to store them is at most $O(m)$. Furthermore, at any round, the size of all the input copies for multiple tasks is at most $n \cdot (m/n)^{1/4} \cdot (m/n)^{1/4} = O(m)$. Thus, the total space needed is $O(m)$. □

32.8.5 Path Generation and Root Changing

Lemma 32.8.8. *Let* par : $V \to V$ *be a set of parent pointers (See Definition 32.4.2) on a vertex set* V*. Let* $n = |V|$*.* FindAncestors(par) *(Algorithm 32.7) can be implemented in* $(\gamma, \delta)-$MPC *model for any* $\gamma \geq \frac{\log n}{\log \log n}$ *and any constant* $\delta \in (0, 1)$*. The parallel running time is* $O(r)$*, where* r *is the number of iterations (see Definition 32.5.6) of* FindAncestors(par)*.*

Proof. The structure of the whole algorithm is the same as the Algorithm 32.2 (see Lemma 32.8.2). All the steps can be done by operation described in **Multiple queries**.

Since the number of rounds needed is r, the parallel time is $O(r)$. For the total space, we need to store all the mappings $g_1, \cdots, g_r$. At the end of the i^{th} round, we need to store mapping h_i. According to Lemma 32.5.7, $r = O(\log n)$ Thus, the total space is $O(rn) = O(n \log n)$. □

Lemma 32.8.9. *Let* par : $V \to V$ *be a set of parent pointers (See Definition 32.4.2) on a vertex set* V*. Let* q *be a vertex in* V*, and* $n = |V|$*.* FindPath(par, q) *(Algorithm 32.8) can be implemented in* $(\gamma, \delta) -$ MPC *model for any* $\gamma \geq \frac{\log n}{\log \log n}$ *and any constant* $\delta \in (0, 1)$*. The parallel running time is* $O(r)$*, where* r *is the number of iterations (see Definition 32.5.6) of* FindAncestors(par) *(Algorithm 32.7).*

Proof. By Lemma 32.8.8, FindAncestors(par) can be implemented in $(\gamma, \delta)-$MPC model for $\gamma \geq \frac{\log n}{\log \log n}$ and any constant $\delta \in (0, 1)$. All the other other steps in the algorithm can be done by operation described in **Multiple queries**. Notice that, after each round, we need to do load balancing which can be done by operation described in **Load balance**.

The number of rounds must be smaller than $O(r)$, where r should be the number of iterations of FindAncestors(par) according to Lemma 32.8.8.

We store all the mappings $g_i, \text{dep}_{\text{par}}$ in the system. They need $O(n \log n)$ total space. In the i^{th} round, we only need to additionally store set S_i which has size at most $O(n)$. Thus, the total space needed is at most $O(n \log n)$. □

Lemma 32.8.10. *Let* par : $V \to V$ *be a set of parent pointers (See Definition 32.4.2) on a vertex set* V*. Let* q *be a vertex in* V*.* RootChange(par, q) *(Algorithm 32.9) can be implemented in* $(\gamma, \delta) -$ MPC *model for any* $\gamma \geq \frac{\log n}{\log \log n}$ *and any constant* $\delta \in (0, 1)$*. The parallel running time is* $O(r)$*, where* r *is the number of iterations (see Definition 32.5.6) of* FindAncestors(par) *(Algorithm 32.7).*

Proof. By Lemma 32.8.9, FindPath(par, q) can be implemented in $(\gamma, \delta) -$ MPC model. The remaining steps in the procedure can be implemented by the operation described by **Multiple queries**, and has $O(1)$ parallel running time.

The total space needed is the total space needed for FindPath(par, q) plus the space needed to store mapping $h, \widehat{\text{par}}$. Thus the total space needed is $O(n \log n) + O(n) = O(n \log n)$.

The parallel running time is linear in the parallel running time of FindPath(par, q). Then, by Lemma 32.8.9, the parallel running time is $O(r)$ where r is the number of iterations (see Definition 32.5.6) of FindAncestors(par). □

32.8.6 Spanning Forest Algorithm

Lemma 32.8.11. *Let $G_2 = (V_2, E_2)$ be an undirected graph. Let $\widetilde{\text{par}} : V_2 \to V_2$ be a set of parent pointers (See Definition 32.4.2) which satisfies that $\forall v \in V_2$ with $\widetilde{\text{par}}(v) \neq v$, $(v, \widetilde{\text{par}}(v))$ must be in E_2. Let $G_1 = (V_1, E_1)$ be an undirected graph satisfies $V_1 = \{v \in V_2 \mid \widetilde{\text{par}}(v) = v\}, E_1 = \{(u, v) \in V_1 \times V_1 \mid u \neq v, \exists (x, y) \in E_2, \widetilde{\text{par}}^{(\infty)}(x) = u, \widetilde{\text{par}}^{(\infty)}(y) = v\}$. Let $\text{par} : V_1 \to V_1$ be a rooted spanning forest (See Definition 32.5.7) of G_1. Let $f : V_1 \times V_1 \to \{\text{null}\} \cup (V_2 \times V_2)$ satisfy the following property: for $u \neq v \in V_1$, if $\text{part}(u) = v$, then $f(u, v) \in \{(x, y) \in E_2 \mid \widetilde{\text{par}}^{(\infty)}(x) = u, \widetilde{\text{par}}^{(\infty)}(y) = v\}$, and $f(v, u) \in \{(x, y) \in E_2 \mid \widetilde{\text{par}}^{(\infty)}(x) = v, \widetilde{\text{par}}^{(\infty)}(y) = u\}$. Let $n = |V_2|$. Then* ForestExpansion$(\text{par}, \widetilde{\text{par}}, f)$ *(Algorithm 32.10) can be implemented in $(\gamma, \delta) -$ MPC model for any $\gamma \geq \log n / \log \log n$ and any constant $\delta \in (0, 1)$ in parallel running time $O(R)$, where $R = \log(\text{dep}(\widetilde{\text{par}}))$.*

Proof. Due to Lemma 32.8.2, line 3 can be done in $O(R)$ parallel time for $R = \log(\text{dep}(\widetilde{\text{par}}))$. Line 9 corresponds to multiple tasks, we can implement them parallelly by operations described in **Multiple queries**, and **Multiple Tasks** (see Section 32.7.6). By Lemma 32.8.10, the total space needed is at most $O(n \log n)$ and the parallel running time is at most $O(R)$ where $R = \log(\text{dep}(\widetilde{\text{par}}))$. □

Theorem 32.8.12. *Let graph $G = (V, E), n = |V|, N = |V| + |E|$ and $m = \Theta(N^\gamma)$ for some arbitrary $\gamma \in [0, 2]$. Let $r > 0$ be a round parameter.* SpanningForest(G, m, r) *(Algorithm 32.11) can be implemented in $(\gamma, \delta) -$ MPC model for any constant $\delta \in (0, 1)$. Furthermore, the parallel running time is $O(R)$, where R is the total number of iterations (see Definition 32.5.8) of* SpanningForest(G, m, r).

Proof. At the beginning of the algorithm, we just store sets V, E, V_0, E_0 and mapping g_0 in the system.

Consider the i^{th} round of the loop. By Lemma 32.8.7, line 8 can be implemented in total space $\Theta(m)$ and in parallel running time $O(k_i)$ where k_i is the number of iterations (see Definition 32.5.5) of MULTIPLELARGETREES(G_i, m). Line 9 can be implemented by operations described in **Sizes of sets**, **Set membership**, and **Multiple queries**. Line 10 can be implemented by operations described in **Indexing elements in sets**, **Set membership**, and **Multiple queries**. In line 12, to calculate γ_i, we need to query n_i, this can be done by operations described in **Sizes of sets** and **Multiple queries**. In line 14, to compute L_i, we only need operations described in **Set membership** and **Multiple queries**. Line 15 can be implemented by operations shown in **Set membership**, **Indexing elements in sets** and **Multiple queries**. By Lemma 32.8.9, for line 16, there are multiple tasks each can be implemented in $O(|V_{\widetilde{T}_i(v)}| \log |V_{\widetilde{T}_i(v)}|)$ total space, and $O(k_i)$ parallel time. We can schedule these multiple tasks (see Section 32.7.6) such that we can finish them in parallel in $O(k_i)$ parallel time. According to Lemma 32.8.2, for line 17, we can implement it in $O(n_i) = O(n)$ total space, and in $O(k'_i)$ parallel time, where k'_i is the number of iterations (see Definition 32.4.5) of TREECONTRACTION(G'_i, par_i). Line 19 can be done by the operation described in **Multiple queries**. Line 20 can be done by the operation described in **Indexing elements in sets** and **Multiple queries**.

Thus, the parallel time is $O(R)$, where $R = \sum_{i=0}^{r-1}(k_i + k'_i)$. By definition of the total number of iterations (see Definition 32.5.8) of SPANNINGFOREST(G, m, r). R is the total number of iterations of SPANNINGFOREST(G, m, r).

For the space, we store all the sets V, E, V_i, D_i and mappings par_i, h_i in all the rounds.

Notice that $\sum_{i=0}^{r} |V_i| \leq 40|V|$. Thus this part takes only $O(N)$ space. In the i^{th} round, we additionally store all the sets $V_{\widetilde{T}_i(v)}, V'_i, E'_i, L_i$ and all the mappings $\text{par}_{\widetilde{T}_i(v)}, \text{dep}_{\widetilde{T}_i(v)}, l_i, z_i$. The total space for this part is at most $O(m)$. For line 16, it creates multiple tasks. The input of each task is at most $|V_{\widetilde{T}_i(v)}| \leq (m/n_i)^{1/2}$. There are at most n_i tasks, and by Lemma 32.8.9, each task will need space at most $O(|V_{\widetilde{T}_i(v)}| \log |V_{\widetilde{T}_i(v)}|)$. Thus, the space for this part is at most $O(m)$. To conclude, the total space needed is at most $O(m)$.

□

Theorem 32.8.13. *Let graph $G = (V, E), n = |V|, N = |V| + |E|$ and $m = \Theta(N^\gamma)$ for some arbitrary $\gamma \in [0, 2]$. Let $r > 0$ be a round parameter. If* SpanningForest(G, m, r) *(Algorithm 32.11) does not return* FAIL*, then let the output be the input of* Orientate$(\cdot)$ *(Algorithm 32.12), and* Orientate$(\cdot)$ *can be implemented in $(\gamma, \delta) -$* MPC *model for any constant $\delta \in (0, 1)$. Furthermore, the parallel running time is $O(R)$, where R is the total number of iterations (see Definition 32.5.8) of* SpanningForest(G, m, r).

Proof. Line 4 to line 7 can be implemented by operations described in **Multiple queries**. Notice that there is a trick here, if $f_i(u, v) =$ null, we do not need to store the tuple $(\text{“}f_i\text{”}, ((u, v), \text{null}))$ in the system. The total space needed to store all the mappings f_i and all the sets F_i for $i \in \{0\} \cup [r]$ is at most $\sum_{i=0}^{r} |V_i| = O(m)$.

Line 10 and line 11 can be implemented by operations described in **Set membership** and **Multiple queries**.

We now look at the second loop, and focus on round i. Line 12 can be implemented by Lemma 32.8.11. The total space needed is at most $O(|V_i| \cdot (m/|V_i|)^{1/2} \cdot \log(m/|V_i|)) = O(m)$. The parallel running time needed is at most $O(k_i)$, where k_i is the number of iterations (see

Definition 32.5.5) of MULTIPLELARGETREES(G_i, m), G_i is the intermediate graph in the procedure SPANNINGFOREST(G, m, r).

Thus, the parallel running time is $O(R)$, where R is the total number of iterations (see Definition 32.5.8) of SPANNINGFOREST(G, m, r). The total space needed is $O(m)$.

□

Now, we are able to conclude the following theorem for spanning forest problem.

Theorem 32.8.14. *For any $\gamma \in [0, 2]$ and any constant $\delta \in (0, 1)$, there is a randomized $(\gamma, \delta) -$ MPC algorithm (see Algorithm 32.11 and Algorithm 32.12) which can output the rooted spanning forest for any graph $G = (V, E)$ in $O(\min(\log D \cdot \log \frac{1}{\gamma'}, \log n))$ parallel time, where D is the diameter of G, $n = |V|$, $N = |V| + |E|$ and $\gamma' = (1 + \gamma) \log_n \frac{2N}{n^{1/(1+\gamma)}}$. The success probability is at least 0.98. In addition, if the algorithm fails, then it will return* FAIL.

Proof. Algorithm 32.11 outputs all the edges in the spanning forest and all the contraction information. Algorithm 32.12 takes the output of Algorithm 32.11 as its input, and outputs a rooted spanning forest.

The implementation of Algorithm 32.11 and Algorithm 32.12 in MPC model is shown by Theorem 32.8.12 and Theorem 32.8.13 respectively. The correctness of Algorithm 32.11 and Algorithm 32.12 is proved by Corollary 32.5.17 and Theorem 32.5.19 respectively. The parallel time of Algorithm 32.11 and Algorithm 32.12 is proved by Theorem 32.5.21.

□

A byproduct of our spanning forest algorithm is an estimator of the diameter of the graph.

Theorem 32.8.15. *For any $\gamma \in [0,2]$ and any constant $\delta \in (0,1)$, there is a randomized $(\gamma,\delta)-$*MPC *algorithm which can output an diameter estimator D' for any graph $G=(V,E)$ in $O(\min(\log D \cdot \log(1/\gamma'), \log n))$ parallel time such that $D \leq D' \leq D^{O(\log(1/\gamma'))}$, where D is the diameter of G, $n=|V|$, $N=|V|+|E|$ and $\gamma' = (1+\gamma)\log_n \frac{2N}{n^{1/(1+\gamma)}}$. The success probability is at least* 0.98. *In addition, if the algorithm fails, then it will return* FAIL.

Proof. By Theorem 32.8.14, we can find a rooted spanning forest. By Theorem 32.5.19, the depth of that rooted spanning forest is at most $D^{O(\log(1/\gamma'))}$. Then we can implement a doubling algorithm (e.g. Modified Lemma 32.8.8, Algorithm 32.7 without maintaining useless g_l) with log in depth parallel time to output the depth of that spanning forest. □

32.8.7 Lowest Common Ancestor and Multi-Paths Generation

Lemma 32.8.16. *Let* $\text{par} : V \to V$ *be a set of parent pointers (See Definition 32.4.2) on a vertex set* V*. Let* $Q = \{(u_1, v_1), (u_2, v_2), \cdots, (u_q, v_q)\}$ *be a set of* q *pairs of vertices, and* $\forall i \in [q], u_i \neq v_i$*. Let* $n = |V|, N = n + q$*.* $\text{LCA}(\text{par}, Q)$ *(Algorithm 32.13) can be implemented in* $(\gamma, \delta) - \text{MPC}$ *model for any* $\gamma \geq \log\log N / \log N$ *and any constant* $\delta \in (0, 1)$ *in* $O(\log(\text{dep}(\text{par})))$ *parallel running time.*

Proof. By Lemma 32.8.8, line 3 can be implemented in space $O(N \log N)$ and $O(\log(\text{dep}(\text{par})))$ parallel running time. It is easy to see that all the other steps in the procedure can be done by the operations shown in **Multiple queries**.

Thus, the total space needed is $O(N \log N)$ and the parallel running time is $O(\log(\text{dep}(\text{par})))$.

□

Lemma 32.8.17. *Let* $\text{par} : V \to V$ *be a set of parent pointers (See Definition 32.4.2) on a vertex set* V*. Let* $Q = \{(u_1, v_1), (u_2, v_2), \cdots, (u_q, v_q)\} \subseteq V \times V$ *satisfy* $\forall j \in [q]$*,* v_j *is an ancestor (See Definition 32.6.1) of* u_j *in* par*. Let* $n = |V|, N = n + q$*.* $\textsc{MultiPath}(\text{par}, Q)$ *(Algorithm 32.14) can be implemented in* $(\gamma, \delta) - \text{MPC}$ *model for any* γ *with* $N \log N + \sum_{i=1}^{q}(\text{dep}_{\text{par}}(u_i) - \text{dep}_{\text{par}}(v_i) + 1) = O(N^{\gamma})$ *and any constant* $\delta \in (0, 1)$ *in* $O(\text{dep}(\text{par}))$ *parallel running time.*

Proof. By Lemma 32.8.8, line 3 can be implemented in space $O(N \log N)$ and $O(\log(\text{dep}(\text{par})))$ parallel running time. It is easy to see that all the other steps in the procedure can be done by the operations shown in **Multiple queries**. Notice that after each round, we need to do load balancing (see **Load balance**) to make each machine have large enough available local

memory. The total space needed is to store all the pathes and the output of line 3. Notice that in round i, we do not need to keep $S_j^{(i')}$ for $i' < i-1$, thus, the space to keep $S_j^{(i)}$ for all $j \in [q]$ only needs $O(\sum_{j=1}^{q}(\mathrm{dep}_{\mathrm{par}}(u_j) - \mathrm{dep}_{\mathrm{par}}(v_j) + 1))$ space.

Thus, the total space needed is at most $O(N \log N + \sum_{i=1}^{q}(\mathrm{dep}_{\mathrm{par}}(u_i) - \mathrm{dep}_{\mathrm{par}}(v_i) + 1)) = O(N^{\gamma})$. The parallel running time is then $O(\mathrm{dep}(\mathrm{par}))$. □

32.8.8 Leaf Sampling

Lemma 32.8.18. *Let* $\mathrm{par} : V \to V$ *be a set of parent pointers (See Definition 32.4.2) on a vertex set* V*, and* par *has a unique root. Let* $n = |V|$*. Let* δ *be an arbitrary constant in* $(0, 1)$*, and let* $m = \lceil n^\delta \rceil$*. Then* LeafSampling$(\mathrm{par}, m, \delta)$ *(Algorithm 32.15) can be implemented in* $(\gamma, \delta) - \mathrm{MPC}$ *model for any* $\gamma \geq \log\log n / \log n$*. Furthermore, with probability at least* $1 - 1/(100m^{5/\delta})$*, the parallel running time is at most* $O(\log \mathrm{dep}(\mathrm{par}))$*.*

Proof. To implement line 4, for each $v \in V$, we can add $\mathrm{part}(v)$ to a temporary set X. Then each v can check whether v is a leaf by checking whether v is in X, and this can be done by the operations shown in **Set membership** and **Multiple queries**.

To implement line 5, for each $v \in V$, we can add v to the set $\mathrm{child}_{\mathrm{par}}(\mathrm{part}(v))$. Then rank can be computed by the operations shown in **Indexing elements in sets** and **Multiple queries**. For line 6, we can implement it on a single machine, since a single machine has local memory $\Theta(m)$. For line 7 to line 9, for each $x \in L$, we add x into S with probability p, where p can be computed by querying the size of L (see **Sizes of sets** and **Multiple queries**). Line 10 can be implemented by operation described in **Indexing elements in sets**, **Set membership**, and **Multiple queries**. By Lemma 32.8.2, line 11 can be implemented in total space $O(N \log N)$ and $O(\log \mathrm{dep}(\mathrm{par}))$ parallel time. By Property 3 of Lemma 32.6.5, with probability at least $1 - 1/(100m^{5/\delta})$, $|S|^2 = O(m)$. Thus, Q can be stored on a single machine. By Lemma 32.8.16, line 15 can be implemented in total space $O(n \log n + |Q|) = O(n \log n)$ and in $O(\log \mathrm{dep}(\mathrm{par}))$ parallel time. By Lemma 32.8.8, line 17 can be implemented in total space $O(n \log n)$ and in $O(\log \mathrm{dep}(\mathrm{par}))$ parallel time. Then line 18 to line 22 can be implemented on a single machine.

Thus, the total space needed is at most $O(n \log n)$. The parallel time is at most $O(\log \mathrm{dep}(\mathrm{par}))$ $\square$

32.8.9 DFS Sequence

Lemma 32.8.19. *Let* $\mathrm{par} : V \to V$ *be a set of parent pointers (See Definition 32.4.2) on a vertex set* V*, and* par *has a unique root. Let* $n = |V|$*. Let* δ *be an arbitrary constant in* $(0, 1)$*, and let* $m = \lceil n^{\delta} \rceil$*.* SubDFS$(\mathrm{par}, m, \delta)$ *(Algorithm 32.16) can be implemented in* $(\gamma, \delta)-$MPC *model for any* $\gamma \geq \log\log n / \log n$*. Furthermore, with probability at least* $1 - 1/(100m^{5/\delta})$*, the parallel running time is at most* $O(\log \mathrm{dep}(\mathrm{par}))$*.*

Proof. By Lemma 32.8.18, line 5 can be implemented in total space $O(n \log n)$ and with probability at least $1-1/(100m^{5/\delta})$ has parallel running time $O(\log \mathrm{dep}(\mathrm{par}))$. By Lemma 32.8.16, line 7 can be implemented in total space $O(n \log n)$ and in parallel running time $O(\log \mathrm{dep}(\mathrm{par}))$. Line 9 can be implemented by operation shown in **Multiple queries**. By Lemma 32.8.17, since all the pathes are disjoint (except the first path and the last path intersecting on the root) and V has n vertices, line 10 can be implemented in $O(n \log n)$ total space and in $O(\log \mathrm{dep}(\mathrm{par}))$ parallel running time. Loop in line 13 and Loop in line 16 can be implemented in parallel, and can be implemented by operations shown in **Indexing elements in sets** and **Multiple queries**. Line 20 can be implemented by operations shown in **Indexing elements in sets** and **Multiple queries**. Now we describe the implementation of line 21. Firstly, we can standardize (see **Sequence standardizing**) the sequence A'. For each tuple $(\text{“}A'\text{”}, (j, u))$, create a tuple $(\text{“}temp_u\text{”}, j)$. Thus, “$temp_u$” is a set which contains all the positions that u appeared. For each tuple $(\text{“}temp_u\text{”}, j)$, we query (see **Multiple queries**) the index i (see **Indexing elements in sets**) of j in set $(\text{“}temp_u\text{”}, j)$, and create a tuple $(\text{“pos”}, ((u, i), j))$. Thus, the desired mapping pos is stored in the system. The loop in line 24 is implemented in parallel. Line 25 can be implemented by the operations shown

in **Set membership** and **Multiple queries**. Line 26 to line 28 can be implemented by the operation shown in **Multiple queries**. Finally, line 30 can be implemented by **Multiple queries** and **Sequence duplicating**.

The total space used in the procedure is at most $O(n \log n)$. The parallel running time is $O(\log \operatorname{dep}(\text{par}))$. □

Theorem 32.8.20. *Let* $\text{par} : V \to V$ *be a set of parent pointers (See Definition 32.4.2) on a vertex set* V*, and* par *has a unique root. Let* $n = |V|, m = n^{\delta}$ *for some arbitrary constant* $\delta \in (0, 1)$*.* $\text{DFS}(\text{par}, m)$ *(Algorithm 32.17) can be implemented in* $(\gamma, \delta) - \text{MPC}$ *model for any* $\gamma \geq \log \log n / \log n$*. With probability at least* 0.99*, the parallel running time is* $O(\log(\operatorname{dep}(\text{par})))$.

Proof. By Lemma 32.8.19, line 5 can be implemented in total space $O(n \log n)$. With probability at least $1 - 1/(100n^5)$, the parallel running time is $O(\log(\operatorname{dep}(\text{par})))$. Line 8 to line 10 can be implemented by operations shown in **Set membership** and **Multiple queries**. By Lemma 32.8.2, line 11 can be implemented in $O(n)$ total space, and $O(\log \operatorname{dep}(\text{par}))$ parallel running time. The loop in line 14 contains multiple tasks (see Section 32.7.6 **Multiple Tasks**), thus we can implement those tasks in parallel. By Lemma 32.8.19, line 17 can be implemented in total space $O(|V_i'(v)| \log |V_i'(v)|)$. Furthermore, with probability at least $1 - 1/(100n^5)$, the parallel running time is $O(\log(\operatorname{dep}(\text{par})))$. Thus, the total space needed for those tasks is at most $O(n \log n)$. Line 19 can be implemented by operations shown in **Indexing elements in sets**, **Sequence insertion** and **Multiple queries**.

Thus, the total space needed is $O(n \log n)$. By taking union bound over all the task SubDFS, with probability at least 0.99, the parallel running time is $O(\log \operatorname{dep}(\text{par}))$. □

Now we are able to conclude the following theorem.

Theorem 32.8.21. *For any $\gamma \in [\beta, 2]$ and any constant $\delta \in (0, 1)$, there is a randomized $(\gamma, \delta)-$*MPC *algorithm (Algorithm 32.17) which can output a Depth-First-Search sequence for any tree graph $G = (V, E)$ in $O(\min(\log D \cdot \log(1/\gamma'), \log n))$ parallel time, where $n = |V|$, $\beta = \Theta(\log \log n / \log n)$, D is the diameter of G, and $\gamma' = \gamma + \Theta(1/\log n)$. The success probability is at least* 0.98. *In addition, if the algorithm fails, then it will return* FAIL.

Proof. Firstly, by Theorem 32.8.14, we can find a rooted tree. Algorithm 32.17 can output the DFS sequence for a rooted tree.

The implementation and parallel time of Algorithm 32.17 is shown by Theorem 32.8.20. The correctness of Algorithm 32.17 is proved by Theorem 32.6.12. The success probability of Algorithm 32.17 is proved by Theorem 32.6.14. □

32.8.10 Range Minimum Query

Lemma 32.8.22. *Let $A = (a_1, a_2, \cdots, a_n)$ be a sequence of numbers. Let δ be an arbitrary constant in $(0, 1)$.* SparseTable$^+(a_1, a_2, \cdots, a_n, \delta)$ *(Algorithm 32.18) can be implemented in $(0, \delta)-$*MPC *model with $O(1)$ parallel running time.*

Proof. Let A be the sequence $(a_1, a_2, \cdots, a_n)$. The algorithm takes $O(1/\delta)$ rounds. m is the local space of a machine. There are $\Theta(n/m)$ machines each holds a consecutive $\Theta(m)$ elements of sequence A. Now consider the round l. Machine $j \in \{0\} \cup [\lceil n/m \rceil]$ needs to compute $\widehat{f}_{j\cdot m+1,l}, \widehat{f}_{j\cdot m+2,l}, \cdots, \widehat{f}_{j\cdot m+m-1,l}$. The number of queries machine j made in line 8 and line 11 is at most $\sum_{t=1}^{\lceil 1/\delta \rceil} |S_t| + 2m \leq O(m/\delta) = O(m)$. Thus, there are total $O(n)$ queries. These queries can be answered simultaneously by operation shown in **Multiple queries**.

Thus, the total space needed is $O(n)$, and the parallel running time is $O(1)$. □

Lemma 32.8.23. *Let $a_1, a_2, \cdots, a_n$ be a sequence of numbers. Let δ be an arbitrary constant in $(0, 1)$.* SparseTable$(a_1, a_2, \cdots, a_n, \delta)$ *(Algorithm 32.19) can be implemented in $(\gamma, \delta)-$*MPC *model for any $\gamma \geq \log\log n / \log n$ in $O(1)$ parallel time.*

Proof. By Lemma 32.8.22, line 4 can be implemented in $O(n)$ total space and $O(1)$ parallel time. The loop in line 15 is similar to Algorithm 32.18. Each machine j needs to compute $f_{j\cdot m+1,t}, \cdots, f_{j\cdot m+m-1,t}$ for all $t \in [\lceil \log n \rceil] \cup \{0\}$. The difference from Algorithm 32.18 is that, it can compute for all t at the same time since it only depends on the value of $\widehat{f}$. The number of queries made by each machine is $O(m \log n)$. Thus, the total number of queries is at most $O(n \log n)$. These queries can be answered simultaneously by operation shown in **Multiple queries**.

Thus, the total space needed is $O(n \log n)$, and the parallel running time is $O(1)$. $\square$

32.9 Minimum Spanning Forest

In this section, we discuss how to apply our connectivity/spanning forest algorithm to the Minimum Spanning Forest (MSF) and Bottleneck Spanning Forest (BSF) problem.

The input of MSF/BSF problem is an undirected graph $G = (V, E)$ together with a weight function $w : E \to \mathbb{Z}$, where E contains m edges $e_1, e_2, \cdots, e_m$ with $w(e_1) \leq w(e_2) \leq \cdots \leq w(e_m)$. The goal of MSF is to output a spanning forest such that the sum of weights of the edges in the forest is minimized. The goal of BSF is to output a spanning forest such that the maximum weight of the edges in the forest is minimized. D is the diameter of the minimum spanning forest. If there are multiple choices of the minimum spanning forest, then let D be the minimum diameter among all the minimum spanning forests.

For simplicity, in all of our proofs, we only discuss the case when all the edges have different weights, i.e. $w(e_1) < w(e_2) < \cdots < w(e_m)$. In this case, the minimum spanning forest is unique. It is easy to extend our algorithms to the case when there are edges with the same weight. We omit the proof for this fact.

Firstly, we show that D is an upper bound of the diameter of G' where the vertex set of G' is the vertex set of G, and the edge set of G' is $\{e_1, e_2, \cdots, e_i\}$ for some arbitrary $i \in [m]$.

Lemma 32.9.1. *Given a graph $G = (V, E)$ for $E = \{e_1, e_2, \cdots, e_m\}$ together with a weight function w which satisfies $w(e_1) < w(e_2) < \cdots < w(e_m)$, then the diameter of $G' = (V, E')$ is at most D, where D is the diameter of the minimum spanning forest of G, and E' only contains the first i edges of E, i.e. $e_1, e_2, \cdots, e_i$ for some arbitrary $i \in [m]$.*

Proof. The proof follows by Kruskal's algorithm directly. □

Our algorithms is based on the following simple but useful Lemma.

Lemma 32.9.2. *Given a graph $G = (V, E)$ for $E = \{e_1, e_2, \cdots, e_m\}$ together with a weight function w which satisfies $w(e_1) \leq w(e_2) \leq \cdots \leq w(e_m)$, $\forall 1 \leq i < j \leq m$, an edge e from $\{e_i, e_{i+1}, \cdots, e_j\}$ is in the minimum spanning forest of G if and only if e' from $\{e'_i, e'_{i+1}, \cdots, e'_j\}$ is in the minimum spanning forest of G', where the vertices of G' is obtained by contracting all the edges $e_1, e_2, \cdots, e_{i-1}$ of G, and $e', e'_i, e'_{i+1}, \cdots, e'_j$ are the edges (or vertices) in G' which corresponds to the edges $e, e_i, e_{i+1}, \cdots, e_j$ before contraction.*

Proof. The proof follows by Kruskal's algorithm directly. □

A natural way to apply Lemma 32.9.2 to parallel minimum spanning forest algorithm is that we can divide the edges into several groups, and recursively solve the minimum spanning forest for each group of edges. More precisely, suppose we have total space $\Theta(km)$, we can divide E into k groups $E_1, E_2, \cdots, E_k$, where $E_i = \{e_{(i-1)\cdot m/k+1}, e_{(i-1)\cdot m/k+2}, \cdots, e_{i\cdot m/k}\}$. We can compute graph $G_1, G_2, \cdots, G_k$ where the vertices of G_i is obtained by contracting all the edges from e_1 to $e_{(i-1)\cdot m/k}$, the edges of G_i are corresponding to the edges in E_i. Then by Lemma 32.9.2, we can obtain the whole minimum spanning forest by solving these k size $O(m/k)$ minimum spanning forest problems. For each sub-problem, we can assign it $\Theta(m)$ working space, thus each sub-problem still has $\Theta(k)$ factor more total space. Therefore, we can recursively apply the above argument.

Theorem 32.9.3. *For any $\gamma \in [0, 2]$ and any constant $\delta \in (0, 1)$, there is a randomized $(\gamma, \delta) - \mathrm{MPC}$ algorithm which can output a minimum spanning forest for any weighted graph $G = (V, E)$ with weights $w : E \to \mathbb{Z}$ in $O(\min(\log D \cdot \log(1/\gamma'), \log n) \cdot 1/\gamma')$ parallel time,*

where $n = |V|$, $\forall e \in E, |w(e)| \leq \text{poly}(n)$, D *is the diameter of a minimum spanning forest of* G, *and* $\gamma' = \gamma/2 + \Theta(1/\log n)$. *The success probability is at least* 0.98. *In addition, if the algorithm fails, then it will return* FAIL.

Proof. Let $n = |V|, m = |E|$. Let $E = \{e_1, \cdots, e_m\}$ with $w(e_1) \leq w(e_2) \leq \cdots \leq w(e_m)$. The total space in the system is $\Theta(m^{1+\gamma})$. Let $k = \Theta(m^{\gamma/2})$. By our previous discussion, we can divide E into k groups $E_1, E_2, \cdots, E_k$, where $E_i = \{e_{(i-1)\cdot m/k+1}, e_{(i-1)\cdot m/k+2}, \cdots, e_{i\cdot m/k}\}$. By Lemma 32.9.1 and Theorem 32.8.4, we can use $O(\min(\log D \cdot \log(1/\gamma'), \log n))$ parallel time and $\Theta(km^{1+\gamma/2})$ total space to compute graph $G_1, G_2, \cdots, G_k$ where the vertices of G_i is obtained by contracting all the edges from e_1 to $e_{(i-1)\cdot m/k}$, the edges of G_i are corresponding to the edges in E_i after contraction.

By Lemma 32.9.2, it suffices to recursively solve the minimum spanning forest problem for each group G_i. Since each time, we split the edges into k groups, the recursion will have at most $O(1/\gamma')$ levels. At the end of the recursion, we are able to determine for every edge e whether e is in the minimum spanning forest.

Now let us consider the success probability. Although Theorem 32.8.4 is a randomized algorithm, the parallel time is always bounded by $\min(\log D \cdot \log(1/\gamma'), \log n)$. If we repeat the algorithm until it succeeds, the expectation of number of trials is a constant. Furthermore, for each level of the recursion, we can regard the graphs in all the tasks composed one large graph. Thus, in real implementation, in each level of the recursion, we will only invoke one connectivity procedure. Thus in expectation, the total parallel time is $O(\min(\log D \cdot \log(1/\gamma'), \log n) \cdot 1/\gamma')$. By applying Markov's inequality, we complete the proof. □

In the following theorem, we show that Lemma 32.9.2 can also be applied in approximate minimum spanning forest problem.

Theorem 32.9.4. *For any $\gamma \in [\beta, 2]$ and any constant $\delta \in (0, 1)$, there is a randomized $(\gamma, \delta) -$ MPC algorithm which can output a $(1+\epsilon)$ approximate minimum spanning forest for any weighted graph $G = (V, E)$ with weights $w : E \to \mathbb{Z}_{\geq 0}$ in $O(\min(\log D \cdot \log(1/\gamma'), \log n))$ parallel time, where $n = |V|$, $N = |V| + |E|$, $\beta = \Theta(\log(\epsilon^{-1} \log n)/\log n)$, $\forall e \in E, |w(e)| \leq \text{poly}(n)$, D is the diameter of a minimum spanning forest of G, and $\gamma' = (1 + \gamma - \beta) \log_n \frac{2N}{n^{1/(1+\gamma-\beta)}}$. The success probability is at least 0.98. In addition, if the algorithm fails, then it will return* FAIL.

Proof. For each edge $e \in E$, we can round $w(e)$ to $w'(e)$ such that $w'(e) = 0$ when $w(e) = 0$, and $w'(e) = (1+\epsilon)^i$ when $w(e) \neq 0$, and i is the smallest integer such that $w(e) \leq (1+\epsilon)^i$.

Since $|w(e)| \leq \text{poly}(n)$ for all $e \in E$, there are only $k = O(\log(n)/\epsilon)$ different values of $w'(e)$. We can divide E into k groups, where the i^{th} group E_i contains all edges with the i^{th} largest weight in w'. By Lemma 32.9.1 and Theorem 32.8.4, we can use $O(\min(\log D \cdot \log(1/\gamma'), \log n))$ parallel time and $\Theta(kN^{1+\gamma-\beta}) = \Theta(N^{1+\gamma})$ total space to compute graph $G_1, G_2, \cdots, G_k$ where the vertices of G_i is obtained by contracting all the edges from E_1 to E_{i-1}, the edges of G_i are corresponding to the edges in E_i after contraction.

Then, for each G_i, since all the edges have the same w' weight, any spanning forest of G_i is a minimum spanning forest of G_i. By Theorem 32.8.14, we can use $O(\min(\log D \cdot \log(1/\gamma'), \log n))$ parallel time and $\Theta(kN^{1+\gamma-\beta}) = \Theta(N^{1+\gamma})$ total space to compute the spanning forest for each graph $G_1, G_2, \cdot, G_k$. By Lemma 32.9.2, the union of all the minimum spanning forest with respect to w' must be the minimum spanning forest of G with respect

to w'. Since all the weights w are nonnegative integers, w' is a $(1+\epsilon)$ approximation to w. Therefore, our output minimum spanning forest with respect to w' is a $(1+\epsilon)$ approximation to the minimum spanning forest with respect to w.

For the success probability, we can apply the similar argument made in the proof of Theorem 32.9.3 to prove that the success probability is at least 0.98. □

In the following, we show that if we only need to find the largest edge in the minimum spanning tree, then we are able to get a better parallel time. It is an another application of our

Theorem 32.9.5. *For any $\gamma \in [0,2]$ and any constant $\delta \in (0,1)$, there is a randomized $(\gamma,\delta)-$*MPC *algorithm which can output a bottleneck spanning forest for any weighted graph $G=(V,E)$ with weights $w : E \to \mathbb{Z}$ in $O(\min(\log D \cdot \log(1/\gamma'), \log n) \cdot \log(1/\gamma'))$ parallel time, where $n = |V|$, $\forall e \in E, |w(e)| \leq \text{poly}(n)$, D is the diameter of a minimum spanning forest of G, and $\gamma' = \gamma/2 + \Theta(1/\log n)$. The success probability is at least 0.98. In addition, if the algorithm fails, then it will return* FAIL.

Proof. Let $n = |V|, m = |E|$. Let $E = \{e_1, \cdots, e_m\}$ with $w(e_1) \leq w(e_2) \leq \cdots \leq w(e_m)$. The total space in the system is $\Theta(m^{1+\gamma})$. Let $k = \Theta(m^{\gamma/2})$. By our previous discussion, we can divide E into k groups $E_1, E_2, \cdots, E_k$, where $E_i = \{e_{(i-1)\cdot m/k+1}, e_{(i-1)\cdot m/k+2}, \cdots, e_{i\cdot m/k}\}$. By Lemma 32.9.1 and Theorem 32.8.4, we can use $O(\min(\log D \cdot \log(1/\gamma'), \log n))$ parallel time and $\Theta(km^{1+\gamma/2})$ total space to compute graph $G_1, G_2, \cdots, G_k$ where the vertices of G_i is obtained by contracting all the edges from e_1 to $e_{(i-1)\cdot m/k}$, the edges of G_i are corresponding to the edges in E_i after contraction.

By Lemma 32.9.2, the edge with largest weight must be in the group E_i for some i with $G_{i+1} = G_{i+2}$. Thus, we reduce the problem size to m/k. By Remark 32.1.2, we can finish the recursion in $O(\log(1/\gamma'))$ phases.

Suppose the bottleneck is e_i, then by Theorem 32.8.14, we can find a spanning forest by only using edges from $\{e_1, \cdots, e_i\}$ in $O(\min(\log D \cdot \log(1/\gamma'), \log n))$ parallel time and in $\Theta(m^{1+\gamma/2})$ total space. Thus, the resulting spanning forest is a bottleneck spanning forest.

For the success probability, we can apply the similar argument made in the proof of Theorem 32.9.3 to prove that the success probability is at least 0.98. □

32.10 Directed Reachability vs. Boolean Matrix Multiplication

In this section, we discuss the directed graph reachability problem which is a directed graph problem highly related to the undirected graph connectivity. In the all-pair directed graph reachability problem, we are given a directed graph $G = (V, E)$, the goal is to answer for every pair $(u, v) \in V \times V$ whether there is a directed path from u to v. There is a simple standard way to reduce Boolean Matrix Multiplication to all-pair directed graph reachability problem. In the Boolean Matrix Multiplication problem, we are given two boolean matrices $A, B \in \{0,1\}^{n \times n}$, the goal is to compute $C = A \cdot B$, where $\forall i, j \in [n]$, $C_{i,j} = \bigvee_{k \in [n]} A_{i,k} \wedge B_{k,j}$. The reduction is as the following. We create $3n$ vertices $u_1, u_2, \cdots, u_n, v_1, v_2, \cdots, v_n, w_1, w_2, \cdots, w_n$. For every $i, j \in [n]$, if $A_{i,j} = 1$, then we add an edge from u_i to v_j, and if $B_{i,j} = 1$, then we add an edge from v_i to w_j. Thus, $C_{i,j} = 1$ is equivalent to there is a path from u_i to w_j. Thus, if we can solve all-pair directed graph reachability problem in $O(T)$ sequential time, then we can solve Boolean Matrix Multiplication in $O(T)$ time. For the current status of sequential running time of Boolean Matrix Multiplication problem, we refer readers to [LG14] and the references therein.

Now, consider the multi-query directed graph reachability problem. In this problem, we are given a directed graph $G = (V, E)$ together with $|V| + |E|$ queries where each query queries the reachability from vertex u to vertex v. The goal is to answer all these queries. A similar problem in the undirected graph is called multi-query undirected graph connectivity problem. In this problem, we are given an undirected graph $G = (V, E)$ together with $|V| + |E|$ queries where each query queries the connectivity between vertex u and vertex v.

According to Theorem 32.8.4 and Lemma 32.7.4, there is a polynomial local running time fully scalable $\sim \log D$ parallel time $(0, \delta) -$ MPC algorithm for multi-query undirected

graph connectivity problem. Here polynomial local running time means that there is a constant $c > 0$ (independent from δ) such that every machine in one round can only have $O((n^{\delta})^{c})$ local computation.

For multi-query directed graph reachability problem, we show that if there is a polynomial local running time fully scalable $(\gamma, \delta) - \text{MPC}$ algorithm which can solve multi-query reachability problem in $O(n^{\alpha})$ parallel time, then we can solve all-pair directed graph reachability problem in $O(n^2 \cdot n^{2\gamma+\alpha+\epsilon})$ sequential running time for any arbitrarily small constant $\epsilon > 0$. Especially, if the algorithm is in $(0, \delta) - \text{MPC}$ model, and the parallel time is $n^{o(1)}$, then we will have an $O(n^{2+\epsilon+o(1)})$ sequential running time algorithm for Boolean Matrix Multiplication which implies a break through in this field.

Suppose we have a such MPC algorithm. Let the input size be $\Theta(m)$, i.e. the number of edges is $\Theta(m)$, and the number of queries is also $\Theta(m)$. Then the total space is $\Theta(m^{1+\gamma})$. Let $\delta = \epsilon/(c-2)$. Then the number of machines is $\Theta(m^{1+\gamma-\delta})$. Now we just simulate this $(\gamma, \delta) - \text{MPC}$ algorithm sequentially, the total running time is $O(m^{1+\gamma-\delta} \cdot m^{c\delta} \cdot n^{\alpha}) = O(m \cdot n^{2\gamma+\epsilon+\alpha})$. To answer reachability for all pairs, we need total $O(n^2 \cdot m \cdot n^{2\gamma+\epsilon+\alpha}/m) = O(n^2 \cdot n^{2\gamma+\alpha+\epsilon})$ time. Therefore, we can use this algorithm to solve Boolean Matrix Multiplication in $O(n^2 \cdot n^{2\gamma+\alpha+\epsilon})$ time.

Theorem 32.10.1. *If there is a polynomial local running time fully scalable $(\gamma, \delta) - \text{MPC}$ algorithm which can answer $|V| + |E|$ pairs of reachability queries simultaneously for any directed graph $G = (V, E)$ in $O(|V|^{\alpha})$ parallel time, then there is a sequential algorithm which can compute the multiplication of two $n \times n$ boolean matrices in $O(n^2 \cdot n^{2\gamma+\alpha+\epsilon})$ time, where $\epsilon > 0$ is a constant which can be arbitrarily small.*

Proof. See above discussions. □

32.11 Discussion on a Previous Conjectured Fast Algorithm

In this section, we discuss the hard example for the algorithm described by [RMCS13]. In [RMCS13], they conjectured that their Hash-to-Min connectivity algorithm can finish in $O(\log D)$ rounds. The description of their algorithm is as the following:

1. The input graph is $G = (V, E)$.
2. For each vertex $v \in V$, initialize a set $S_v^{(0)} = v$.
3. in round i:
   (a) Each vertex v find $u \in S_v^{(i-1)}$ which has the minimum label, i.e. $u = \min_{x \in S_v^{(i-1)}} x$.
   (b) v sends u the all the vertices in $S_v^{(i-1)}$.
   (c) v sends every $x \in S_v^{((i-1))} \setminus \{u\}$ the vertex u.
   (d) Let $S_v^{(i)}$ be $\{v\}$ union the set of all the vertices received.
   (e) If for all v, $S_v^{(i)}$ is the same as $S_v^{(i-1)}$, then finish the procedure.

The above procedure can be seen as the modification of the graph: in each round, all the vertices together create a new graph. For each vertex v, let u be the neighbor of v with the minimum label, and if x is a neighbor of v, then add an edge between x and u in the new graph. So in each round, each vertex just communicates with its neighbors to update the new minimum neighbor it learned. At the end of the algorithm, it is obvious that the minimum vertex in each component will have all the other vertices in that component, and for each non minimum vertex, it will have the minimum vertex in the same component.

A hard example for this algorithm is shown by Figure 32.3. The example is a thin and tall grid graph with a vertex connected to all the vertices in the first column. The total number of vertices is n. The grid graph has $D = \frac{1}{2}\log n$ columns and n/D rows. We index each column from left to right by 1 to D. We index each row from top to down by 1 to n/D. The single large degree vertex has label 0. The i^{th} row has the vertices with label $(i-1)\cdot D+1$ to $i\cdot D$ from the first column to the D^{th} column. We claim that if vertex v is the i^{th} row and j^{th} column, then before round k for $2^k < i, k < j$, the neighbors of v will only in column $j-1$, column j and column $j+1$. Furthermore, the minimum neighbor of v in column $j-1$ will be $v-(2^{k-1}-1)\cdot D-1$. The minimum neighbor of v in column j will be $v-2^{k-1}\cdot D$. The minimum neighbor of v in column $j+1$ will be $v-D\cdot(2^{k-1}-1)+1$. This claim is true when $k=1$. Then by induction, we can prove the claim. Thus, it will take at least $\Theta(D)$ rounds to finish the procedure where $D=\Theta(\log n)$.

If we randomly label the vertices at the beginning, then consider the case we copy that hard structure at least n^{n+2} times, then with high probability, there is a component which has the labels with the order as the same as described above. In this case, the procedure needs $\Omega(\log\log N)$ rounds, where $N=n^{n+3}$ is the total number of the vertices.

Also notice that, even we give more total space to this algorithm, this algorithm will not preform better. In our connectivity algorithm, if we have $\Omega(n^{1+\epsilon})$ total space for some arbitrary constant $\epsilon > 0$, then our parallel running time is $O(\log D)$.

32.12 Alternative Approach for Leader Selection

In this section, we show that there is a different way to select leaders (see Section 32.4.2). The number of leaders selected by this approach will depend on the sum of inverse degrees of all the vertices. Let us first introduce the concept of *Min Parent Forest*.

32.12.1 Min Parent Forest

Let $G = (V, E)$ be an undirected graph where V denotes the vertex set of G, and E denotes the edge set of G. Each vertex $v \in V$ has a weight $w(v) \in \mathbb{R}$, and it also has a unique label from $\mathbb{Z}$. For convenience, for each vertex $v \in V$, we also use v to denote its label. Let $\Gamma_G(v)$ denote the set of neighbors of v, i.e. $\Gamma_G(v) = \{u \in V \mid (u, v) \in E\}$. If G is clear in the context, we just use $\Gamma(v)$ to denote $\Gamma_G(v)$. The size of $\Gamma(v)$, $|\Gamma(v)|$, is called the degree of v. Let $f_{G,w} : V \to V$ be the "min-weight-parent" function defined as the following:

1. If $w(v) = \min_{u \in \Gamma(v) \cup \{v\}} w(u)$, then $f_{G,w}(v) = v$.
2. Otherwise, let $u^* \in \Gamma(v)$ be the vertex which has the smallest weight, i.e. $w(u^*) = \min_{u \in \Gamma(v)} w(u)$. If there is more than one choice of u^*, let u^* be the one with the smallest label. And $f_{G,w}(v)$ is defined to be u^*.

We call $(V, f_{G,w})$ the *min-parent-forest* of graph G with vertex weights w. We can then define i-step "min-weight-parent" function. For $v \in V$, we define $f^{(0)}_{G,w}(v) = v$. For $i \in \mathbb{Z}_{>0}$, we can define $f^{(i)}_{G,w}$ as the following:

$$\forall v \in V, f^{(i)}_{G,w}(v) = f_{G,w}(f^{(i-1)}_{G,w}(v)).$$

In the following, we define the concept of roots in the *min-parent-forest*.

Definition 32.12.1 (Roots in the forest)**.** Let $v \in V$, and let $(V, f_{G,w})$ be the *min-parent-forest* of graph $G = (V, E)$ with vertex weights w. If $f_{G,w}(v) = v$, then v is a root in the forest $(V, f_{G,w})$.

The depth of a vertex v is defined as the distance on the tree between v and the corresponding root in the forest.

Definition 32.12.2 (The depth of v)**.** Let $v \in V$, and let $(V, f_{G,w})$ be the *min-parent-forest* of graph $G = (V, E)$ with vertex weights w. The depth of v in the forest $(V, f_{G,w})$ is the smallest $i \in \mathbb{Z}_{\geq 0}$ such that $f_{G,w}^{(i)}(v) = f_{G,w}^{(i+1)}(v)$. We use $\mathrm{dep}_{G,w}(v)$ to denote the depth of v in $(V, f_{G,w})$. We call $f_{G,w}^{(\mathrm{dep}_{G,w}(v))}(v)$ the root of v. For the simplicity of the notation, we also use $f_{G,w}^{(\infty)}(v)$ to denote the root of v.

The above definition is well defined since if $f_{G,w}^{(i+1)}(v) \neq f_{G,w}^{(i)}(v)$ then $w(f_{G,w}^{(i+1)}(v))$ should be strictly smaller than $w(f_{G,w}^{(i)}(v))$ by the definition of $f_{G,w}$ and $f_{G,w}^{(j)}$ for all $j \in \mathbb{Z}_{\geq 0}$. Therefore, there must exist i such that $f_{G,w}^{(i)}(v) = f_{G,w}^{(i+1)}(v)$.

The depth of the forest is the largest depth among all the vertices.

Definition 32.12.3 (The depth of the *min-parent-forest*)**.** The depth $\mathrm{dep}(G, w)$ of the forest $(V, f_{G,w})$ is defined as:

$$\mathrm{dep}(G, w) = \max_{v \in V} \mathrm{dep}_{G,w}(v).$$

If the weights w of vertices of G are some i.i.d. random variables, then with high probability, the depth of $(V, f_{G,w})$ is only $O(\log |V|)$. Precisely, we have the following Lemma.

Lemma 32.12.1 (The depth of the random *min-parent-forest*)**.** *Let $G = (V, E)$ be an undirected graph with n vertices where $V = \{v_1, v_2, \cdots, v_n\}$, and the labels satisfies $v_1 < v_2 < \cdots < v_n$. Let $w(v_1), w(v_2), \cdots, w(v_n)$ be n i.i.d. random variables drawn uniformly from $[N]$. If $N > n^2/\delta$ for some $\delta \in (0, 1)$, then for any $t \geq 60 \log n$,*

$$\Pr_{w \sim [N]^n} (\mathrm{dep}(G, w) \leq t) \geq 1 - \delta - e^{-\frac{1}{2}t}.$$

Proof. Let $w(v_1), w(v_2), \cdots, w(v_n)$ be n i.i.d. random variables drawn uniformly from $[N]$. Let $(V, f_{G,w})$ be the *min-parent-forest* of (G, w). For a fixed $s \in V$, we create a set of random variables $z_1, z_2, \cdots, z_n$ by the following deterministic procedure:

1. Let $z_1 = w(s), k = 0, S_k = \{s\}, u_k = s, i = 2, \mathrm{pos}(s) \leftarrow 1$.
2. Let $S_{k+1} = S_k$.
3. For $j = 1 \to n$,

   if $v_j \in \Gamma(u_k)$ and $v_j \notin S_k$ then let $\mathrm{pos}(v_j) \leftarrow i, S_{k+1} \leftarrow S_{k+1} \cup \{v_j\}, z_i = w(v_j)$, $i \leftarrow i + 1$.
4. If $f_{G,w}(u_k) \neq u_k$, then let $u_{k+1} = f_{G,w}(u_k)$, $k \leftarrow k + 1$ and go to step 2.
5. Otherwise, for $j = 1 \to n$,

   if $v_j \notin S_{k+1}$ then let $\mathrm{pos}(v_j) \leftarrow i, z_i = w(v_j), i \leftarrow i + 1$.

It is easy to observe that k is exactly $\mathrm{dep}_{G,w}(s)$ at the end of the above procedure. The reason is that $u_0 = s = f^{(0)}_{G,w}(s)$, $\forall j \in [k], u_j = f_{G,w}(u_{j-1}) = f^{(j)}_{G,w}(s)$ and $f_{G,w}(u_k) = u_k$.

Fact 32.12.2. *$\forall v \in V$, $w(v) = z_{\mathrm{pos}(v)}$, where $\mathrm{pos} : [V] \to [n]$ and $\mathrm{pos}^{-1} : [n] \to [V]$.*

Claim 32.12.3. $\forall j \in \{0, 1, \cdots, k+1\}, S_j = \{u_0\} \cup \bigcup_{p=0}^{j-1} \Gamma(u_p)$.

Proof. We can prove this by induction. The statement is obviously true for S_0 since $S_0 = \{u_0\}$. Now suppose the claim is true for S_{j-1}. Then according to the step 3 of the procedure $S_j = S_{j-1} \cup (\Gamma(u_{j-1}) \setminus S_{j-1}) = S_{j-1} \cup \Gamma(u_{j-1}) = \{u_0\} \cup \bigcup_{p=0}^{j-1} \Gamma(u_p)$. □

Claim 32.12.4. $\forall j \in \{0, 1, \cdots, k\}$, $w(u_j) = \min_{v \in S_j} w(v)$.

Proof. Since $\forall j \in [k], u_j = f_{G,w}(u_{j-1})$, we have $w(u_j) = \min_{v \in \Gamma(u_{j-1}) \cup \{u_{j-1}\}} w(v)$. Then we have $w(u_j) = \min_{v \in \{u_0\} \cup \bigcup_{p=0}^{j-1} \Gamma(u_p)} w(v) = \min_{v \in S_j} w(v)$, where the last equality follows by Claim 32.12.3. □

We use $\text{pos}^{-1}(i)$ to denote vertex v which satisfies $\text{pos}(v) = i$. According to the step 3, it is easy to see $\forall j \in \{0, 1, \cdots, k+1\}$, we have $\{\text{pos}^{-1}(i) \mid i \in [|S_j|]\} = S_j$.

Claim 32.12.5. $\forall j \in \{0, 1, \cdots, k\}, z_{\text{pos}(u_j)} = \min_{p \in [\text{pos}(u_j)]} z_p$.

Proof. $z_{\text{pos}(u_j)} = w(u_j) = \min_{v \in S_j} w(v) = \min_{v \in S_j} z_{\text{pos}(v)} = \min_{p \in [|S_j|]} z_p = \min_{p \in [\text{pos}(u_j)]} z_p$, where the second equality follows by Claim 32.12.4, and the last equality follows by $u_j \in S_j$, so $\text{pos}(u_j) \leq |S_j|$. □

Now we define an another set of random variables $y_1, y_2, \cdots, y_n$, where $\forall i \in [n], y_i \in \{0, 1\}$ and $y_i = 1$ if and only if $z_i = \min_{j \in [i]} z_j$. According to Claim 32.12.5, we have that $\forall i \in \{0, 1, \cdots, k\}, y_{\text{pos}(u_i)} = 1$. Thus, $\text{dep}_{G,w}(s) = k \leq \sum_{i=1}^{n} y_i$. To upper bound $\text{dep}_{G,w}(s)$, it suffices to upper bound $\sum_{i=1}^{n} y_i$.

Before we look at $y_1, \cdots, y_n$, we firstly focus on the properties of $z_1, \cdots, z_n$:

Claim 32.12.6. *$z_1, z_2, \cdots, z_n$ are n i.i.d random variables drawn uniformly from $[N]$.*

Proof. A key observation is that if $z_1, z_2, \cdots, z_n$ are given, then we can recover $w(v_1), w(v_2), \cdots, w(v_n)$ exactly by the following deterministic procedure:

1. Let $w(s) = z_1, k = 0, S_k = \{s\}, u_k = s, i = 2$.
2. Let $S_{k+1} = S_k$.
3. For $j = 1 \to n$,

   if $v_j \in \Gamma(u_k)$ and $v_j \notin S_k$ then let $S_{k+1} \leftarrow S_{k+1} \cup \{v_j\}, w(v_j) = z_i$, $i \leftarrow i + 1$.
4. If $f_{G,w}(u_k) \neq u_k$, then let $u_{k+1} = f_{G,w}(u_k)$, $k \leftarrow k + 1$ and go to step 2.
5. Otherwise, for $j = 1 \to n$,

   if $v_j \notin S_{k+1}$ then let $\mathrm{pos}(v_j) \leftarrow i, w(v_j) = z_i, i \leftarrow i + 1$.

Notice that after step 3, $\forall v \in \Gamma(u_k) \cup \{u_k\}$, $w(v)$ is already recovered, thus we can implement step 4. Thus, the above procedure is a valid procedure. Since $z_1, \cdots, z_n$ are generated by $w(v_1), \cdots, w(v_n)$, we can also know $z_1, \cdots, z_n$ by given $w(v_1), \cdots, w(v_n)$. This means that

$$H(z_1, z_2, \cdots, z_n \mid w(v_1), w(v_2), \cdots, w(v_n)) = H(w(v_1), w(v_2), \cdots, w(v_n) \mid z_1, z_2, \cdots, z_n) = 0,$$

where $H(\cdot)$ is the information entropy. Notice that

$$\begin{aligned}
&I(z_1, z_2, \cdots, z_n; w(v_1), w(v_2), \cdots, w(v_n)) \\
&= H(z_1, z_2, \cdots, z_n) - H(z_1, z_2, \cdots, z_n \mid w(v_1), w(v_2), \cdots, w(v_n)) \\
&= H(w(v_1), w(v_2), \cdots, w(v_n)) - H(w(v_1), w(v_2), \cdots, w(v_n) \mid z_1, z_2, \cdots, z_n),
\end{aligned}$$

where $I(\cdot)$ is the mutual information. Thus, $H(z_1, z_2, \cdots, z_n) = H(w(v_1), w(v_2), \cdots, w(v_n)) = n \log N$. For $i \in [n]$, since the size of the support of z_i is at most N, $H(z_i) \leq \log N$ where the equality holds if and only if z_i is uniformly distributed on $[N]$. Also notice that $H(z_1, z_2, \cdots, z_n) \leq \sum_{i=1}^{n} H(z_i)$, where the equality holds if and only if z_i are independent. Since $\sum_{i=1}^{n} H(z_i) \leq n \log N$, we have $H(z_1, z_2, \cdots, z_n) = \sum_{i=1}^{n} H(z_i)$, and for each $i \in [n]$, $H(z_i) = \log N$. Thus, $z_1, z_2, \cdots, z_n$ are i.i.d. random variables drawn uniformly from $[N]$. □

Claim 32.12.7. *If $N > n^2/\delta$ for some $\delta \in (0, 1)$, then with probability at least $1 - \delta$, $\forall i \neq j \in [n]$, we have $w(v_i) \neq w(v_j)$.*

Proof. Recall that $w(v_1), w(v_2), \cdots, w(v_n)$ are n i.i.d. random variables drawn uniformly from N. For any $i \neq j \in [n]$, the $\Pr(w(v_i) \neq w(v_j)) = 1/N$, thus $\mathbb{E}(|\{(i, j) \in [n] \times [n] \mid i \neq j, w(v_i) \neq w(v_j)\}|) \leq n^2/N$. By Markov's inequality,

$$\Pr(|\{(i, j) \in [n] \times [n] \mid i \neq j, w(v_i) \neq w(v_j)\}| \geq 1) \leq n^2/N \leq \delta.$$

Thus,

$$\Pr(\forall i \neq j \in [n], z_i \neq z_j) \geq 1 - \delta.$$

□

Claim 32.12.8. *Let $\mathcal{E}$ be the event that $\forall i \neq j \in [n], w(v_i) \neq w(v_j)$. Then, for any $t \geq 3 \sum_{i=1}^{n} \frac{1}{i}$, we have*

$$\Pr_{w \sim [N]^n} \left(\sum_{i=1}^{n} y_i \geq t + \sum_{i=1}^{n} \frac{1}{i} \;\middle|\; \mathcal{E} \right) \leq e^{-\frac{3}{4}t}.$$

Proof. Note that $\mathcal{E}$ happened if and only if we have $\forall i \neq j \in [n], z_i \neq z_j$. Due to Claim 32.12.6, $z_1, z_2, \cdots, z_n$ are i.i.d. random variables drawn uniformly from $[N]$, then conditioned on $\mathcal{E}$, $y_1, y_2, \cdots, y_n$ are independent, and the probability that $y_i = 1$ is $1/i$. Thus, we have:

$$\begin{aligned}
& \Pr\left(\sum_{i=1}^{n} y_i \geq \sum_{i=1}^{n} \frac{1}{i} + t \;\middle|\; \mathcal{E}\right) \\
= \; & \Pr\left(\sum_{i=1}^{n} (y_i - \mathbb{E}\left(y_i \mid \mathcal{E}\right)) \geq t \;\middle|\; \mathcal{E}\right) \\
\leq \; & \exp\left(-\frac{\frac{1}{2}t^2}{\sum_{i=1}^{n} \mathbb{V}(y_i \mid \mathcal{E}) + \frac{1}{3}t}\right) \\
\leq \; & \exp\left(-\frac{\frac{1}{2}t^2}{\sum_{i=1}^{n} \frac{1}{i} + \frac{1}{3}t}\right) \\
\leq \; & \exp\left(-\frac{\frac{1}{2}t^2}{\frac{2}{3}t}\right) \\
= \; & \exp\left(-\frac{3}{4}t\right),
\end{aligned}$$

where the first equality follows by $\mathbb{E}(y_i|\mathcal{E}) = 1/i$. The first inequality follows by Berinstein inequality. The second inequality follows by

$$\sum_{i=1}^{n} \mathbb{V}(y_i \mid \mathcal{E}) \leq \sum_{i=1}^{n} \mathbb{E}(y_i^2 \mid \mathcal{E}) = \sum_{i=1}^{n} \mathbb{E}(y_i \mid \mathcal{E}) = \sum_{i=1}^{n} \frac{1}{i}.$$

The third inequality follows by $\sum_{i=1}^{n} \frac{1}{i} \leq \frac{1}{3}t$. □

For a fixed vertex $s \in V$, due to Claim 32.12.8, for any $t \geq 3\sum_{i=1}^{n} 1/i$, we have

$$\Pr\left(\mathrm{dep}_{G,w}(s) \geq \sum_{i=1}^{n} 1/i + t \;\middle|\; \mathcal{E}\right) \leq e^{-\frac{3}{4}t}. \tag{32.1}$$

Thus, for any $t \geq 60 \log n$,

$$
\begin{aligned}
& \Pr_{w \sim [N]^n} \left(\exists s \in V \text{ s.t. } \mathrm{dep}_{G,w}(s) \geq t \right) \\
\leq\ & \Pr \left(\exists s \in V \text{ s.t. } \mathrm{dep}_{G,w}(s) \geq 5t/6 + \sum_{i=1}^{n} 1/i \right) \\
=\ & \Pr \left(\exists s \in V \text{ s.t. } \mathrm{dep}_{G,w}(s) \geq 5t/6 + \sum_{i=1}^{n} 1/i \;\middle|\; \mathcal{E} \right) \Pr(\mathcal{E}) \\
& + \Pr \left(\exists s \in V \text{ s.t. } \mathrm{dep}_{G,w}(s) \geq 5t/6 + \sum_{i=1}^{n} 1/i \;\middle|\; \neg\mathcal{E} \right) \Pr(\neg\mathcal{E}) \\
\leq\ & \Pr \left(\exists s \in V \text{ s.t. } \mathrm{dep}_{G,w}(s) \geq 5t/6 + \sum_{i=1}^{n} 1/i \;\middle|\; \mathcal{E} \right) + \Pr(\neg\mathcal{E}) \\
\leq\ & \Pr \left(\exists s \in V \text{ s.t. } \mathrm{dep}_{G,w}(s) \geq 5t/6 + \sum_{i=1}^{n} 1/i \;\middle|\; \mathcal{E} \right) + \delta \\
\leq\ & \sum_{s \in V} \Pr \left(\mathrm{dep}_{G,w}(s) \geq 5t/6 + \sum_{i=1}^{n} 1/i \;\middle|\; \mathcal{E} \right) + \delta \\
\leq\ & n e^{-\frac{5}{8}t} + \delta \\
\leq\ & e^{-\frac{1}{2}t} + \delta
\end{aligned}
$$

where the first inequality follows by $\frac{1}{6}t \geq 10 \log n \geq \sum_{i=1}^{n} 1/i$. The third inequality follows by Claim 32.12.7. The forth inequality follows by union bound. The fifth inequality follows by Equation (32.1). The sixth inequality follows by $e^{-\frac{1}{8}t} \leq \frac{1}{n}$.

Thus, we can conclude that for any $t \geq 60 \log n$, we have $\Pr(\mathrm{dep}(G, w) \leq t) \geq 1 - \delta - e^{-\frac{1}{2}t}$. □

Lemma 32.12.9 (The number of roots of the random *min-parent-forest*)**.** *Let $G = (V, E)$ be an undirected graph with n vertices where $V = \{v_1, v_2, \cdots, v_n\}$, and the labels satisfies*

$v_1 < v_2 < \cdots < v_n$. *Let* $w(v_1), w(v_2), \cdots, w(v_n)$ *be* n *i.i.d. random variables drawn uniformly from* $[N]$. *Let* $\delta \in (0,1)$. *If* $N > n^3$, *then*

$$\Pr_{w \sim [N]^n} \left(|\{v \in V \mid f_{G,w}(v) = v\}| \geq \frac{2}{\delta} \sum_{v \in V} \frac{1}{|\Gamma(v)| + 1} \right) \leq \delta.$$

Proof. Let $w(v_1), w(v_2), \cdots, w(v_n)$ be n i.i.d. random variables drawn uniformly from $[N]$. Let $\mathcal{E}$ be the event that $\forall i \neq j \in [n], w(v_i) \neq w(v_j)$. Notice that for $i \neq j$, the probability that $w(v_i) = w(v_j)$ is $1/N$. Thus, $\mathbb{E}(|\{(i,j) \in [n] \times [n] \mid i \neq j, w(v_i) = w(v_j)\}|) \leq n^2/N$. Thus, if $N > n^2$, then $\Pr(\neg\mathcal{E}) = \Pr\left(|\{(i,j) \in [n] \times [n] \mid i \neq j, w(v_i) = w(v_j)\}| \geq 1\right) \leq n^2/N \leq \frac{1}{n}$. Now, we fix a vertex $v \in V$,

$$\begin{aligned}
& \Pr\left(f_{G,w}(v) = v\right) \\
= \; & \Pr\left(f_{G,w}(v) = v \mid \mathcal{E}\right) \Pr(\mathcal{E}) + \Pr\left(f_{G,w}(v) = v \mid \neg\mathcal{E}\right) \Pr(\neg\mathcal{E}) \\
\leq \; & \Pr\left(f_{G,w}(v) = v \mid \mathcal{E}\right) + \Pr(\neg\mathcal{E}) \\
\leq \; & \Pr\left(w(v) = \min_{u \in \{v\} \cup \Gamma(v)} w(u) \mid \mathcal{E}\right) + \frac{1}{n} \\
\leq \; & \frac{1}{|\Gamma(v)| + 1} + \frac{1}{n} \\
\leq \; & \frac{2}{|\Gamma(v)| + 1}
\end{aligned}$$

where the third inequality follows by the symmetry of all the variables $w(u)$ for $u \in \{v\} \cup \Gamma(v)$ so condition on all the w are different, with probability $\frac{1}{1+|\Gamma(v)|}$, $w(v)$ is the smallest one. The last inequality follows by $|\Gamma(v)| + 1 \leq |V| = n$.

Thus, $\mathbb{E}(|\{v \in V \mid f_{G,w}(v) = v\}|) \leq \sum_{v \in V} \frac{2}{|\Gamma(v)|+1}$. Let $\delta \in (0,1)$, then by Markov's inequality,

$$\Pr\left(|\{v \in V \mid f_{G,w}(v) = v\}| \geq \frac{2}{\delta} \sum_{v \in V} \frac{1}{|\Gamma(v)| + 1} \right) \leq \delta.$$

□

32.12.2 Leader Selection via Min Parent Forest

Given a graph, we can randomly assign each vertex a weight, thus we have a *min-parent-forest*, then we select those roots in the *min-parent-forest* as leaders, and try to contract all the vertices to the leaders. If we replace line 13 to line 17 of Algorithm 32.3 by Algorithm 32.20. We can get a new algorithm with the following guarantees.

Theorem 32.12.10. *Suppose we replace line 13 to line 17 of Algorithm 32.3 by Algorithm 32.20.*

Let $G = (V, E)$ be an undirected graph, $m = \Omega(n)$, and $r \leq n$ be the rounds parameter where n is the number of vertices in G. Let $c > 0$ be a sufficiently large constant. If $r \geq c \log \log_{m/n}(n)$, then with probability at least $2/3$, the modified CONNECTIVITY(G, m, r) *(Algorithm 32.3) will not return FAIL, and the total number of iterations (see Definition 32.4.6) of the modified* CONNECTIVITY(G, m, r) *is at most $O(r \cdot (\log D + \log \log n))$, where $D = \operatorname{diam}(G)$.*

Proof. Let k_i denote the number of iterations (see Definition 32.4.1) of NEIGHBORINCREMENT(m, G_{i-1}). By Lemma 32.4.2, we have $k_i \leq O(\log D)$. Thus, $\sum_{i=1}^{r} k_i = O(r \cdot \log D)$.

According to Lemma 32.12.1, with probability at least $1 - \frac{2}{100r}$, $\operatorname{dep}(G_i'', w_i) \leq O(\log n)$. By Lemma 32.4.6, with probability at least $1 - \frac{2}{100r}$, the number of iteration of TREECONTRACTION$(G_i'', \mathrm{par}_i$ (see Definition 32.4.5) $r_i' \leq O(\log \log n)$ By taking union bound over all $i \in [r]$, then with probability at least $1 - \frac{1}{50}$, $\sum_{i=1}^{r} r_i' \leq O(r \cdot \log \log n)$.

Due to the Property 3 of Lemma 32.4.2, $\forall i \in [r]$, $\forall v \in V_i'', u \in \Gamma_{G_i'}(v)$, we have $u \in V_i''$ which means that $u \in \Gamma_{G_i''}(v)$. Thus, $|\Gamma_{G_i''}(v)| \geq \lceil (m/n_{i-1})^{1/2} \rceil - 1$. Then due

to Lemma 32.12.9, we have that with probability at most $\frac{1}{8}$, $n_i \geq 16n_{i-1}^{3/2}/m^{1/2}$. Since $m/n \geq m/n_i \geq 1024$, we have that with probability at most $\frac{1}{8}$, $n_i \geq n_{i-1}^{11/10}/m^{1/10}$. Let $y_1, y_2, \cdots, y_r$ be r random variables. If $n_i \geq n_{i-1}^{11/10}/m^{1/10}$, then $y_i = 1$, otherwise $y_i = 0$. We have $\mathbb{E}(\sum_{i=1}^{r} y_i) \leq \frac{r}{8}$. By Markov's inequality, we have $\Pr(\sum_{i=1}^{r} y_i \geq \frac{r}{2}) \leq \frac{1}{4}$. Thus, with probability at least $\frac{3}{4}$, $\sum_{i=1}^{r} y_i \leq \frac{r}{2}$. Notice that when $y_i = 0$, then $n_i \leq n_{i-1}^{11/10}/m^{1/10}$, and when $y_i = 1$, we have $n_i \leq n_{i-1}$. So if there are at least $\frac{r}{2}$ number of y_is which are 0, then

$$
\begin{aligned}
n_r &\leq \frac{\left(\frac{\left(\frac{n^{1.1}}{m^{0.1}}\right)^{1.1}}{m^{0.1}}\right)^{\cdots}}{\cdots} && \text{(Apply } r/2 \text{ times)} \\
&= \frac{n^{1.1^{r/2}}}{m^{1.1^{r/2-1}}} \\
&= n/(m/n)^{1.1^{r/2-1}} \\
&\leq n/(m/n)^{1.1^{r/4}} \\
&\leq \frac{1}{2}
\end{aligned}
$$

where the last inequality follows by $r \geq \frac{4}{\log 1.1}(\log\log_{m/n}(2n))$. Since n_r is an integer, when $n_r \leq \frac{1}{2}$, $n_r = 0$. Thus, we can conclude that if $r \geq c \cdot \log\log_{m/n} n$ for a sufficiently large constant $c > 0$, then with probability at least $\frac{3}{4} - \frac{1}{50} \geq \frac{2}{3}$, the modified CONNECTIVITY(G, m, r) will not output FAIL. □

Notice that though the theoretical guarantees of the *min-parent-forest* leader selection method is worse than the random leader sampling, the merit of *min-parent-forest* leader selection method is that it can have an "early start".

Consider the case when the total space size m is $\Theta(n)$. In this case, random leader sampling will always sample a half of the vertices as the leaders until the total space m is

poly($\log n$) larger than the number of vertices. However, *min-parent-forest* leader selection method can make a large progress at the beginning, it will choose the number of leaders to be about the sum of inverse degrees. Furthermore, the depth of the *min-parent-forest* may not always have $\log n$ depth. Thus, it is an interesting question which leader selection approach has better performance in practice.

Algorithm 32.3 Graph Connectivity

procedure CONNECTIVITY($G = (V, E), m, r$) ▷ Theorem 32.4.9, Theorem 32.4.15
  Output: FAIL or col : $V \to V$.
  $n \leftarrow |V|$
  $\forall v \in V$, $h_0(v) \leftarrow$ null.
  $G_0 = (V_0, E_0) = G$, i.e. $V_0 = V, E_0 = E$.
  $n_0 = n$.
  for $i = 1 \to r$ **do**
    $\forall v \in V$, $h_i(v) \leftarrow$ null. ▷ $h_i(v)$ is the vertex that v contracts to
    $G'_i = (V'_i, E'_i) =$ NEIGHBORINCREMENT(m, G_{i-1}). ▷ Algorithm 32.1
    Compute $V''_i = \{v \in V'_i \mid |\Gamma_{G'_i}(v)| \geq \lceil (m/n_{i-1})^{1/2} \rceil - 1\}$.
    Compute $E''_i = \{(u, v) \in E_{i-1} \mid u \in V''_i, v \in V''_i\}$.
    $G''_i = (V''_i, E''_i)$. ▷ G''_i is obtained by removing all the small components of G_i
    Let $\gamma_i = \lceil (m/n_{i-1})^{1/2} \rceil, p_i = \min((30 \log(n) + 100)/\gamma_i, 1/2)$.
    Let $l_i : V''_i \to \{0, 1\}$ be a random hash function such that $\forall v \in V''_i, l_i(v)$ are i.i.d. Bernoulli random variables, and $\Pr(l_i(v) = 1) = p_i$.
    Let $L_i = \{v \in V''_i \mid l_i(v) = 1\} \cup \{v \in V''_i \mid \forall u \in \Gamma_{G'_i}(v) \cup \{v\}, l_i(u) = 0\}$. ▷ L_i are leaders
    $\forall v \in V''_i$ with $v \in L_i$, let $\mathrm{par}_i(v) = v$.
    $\forall v \in V''_i$ with $v \notin L_i$, let $\mathrm{par}_i(v) = \min_{u \in L_i \cap (\Gamma_{G'_i}(v) \cup \{v\})} u$. ▷ Non-leader finds a leader.
    Let $((V_i, E_i), g_i^{(r'_i)}) =$ TREECONTRACTION(G''_i, par_i). ▷ Algorithm 32.2
    $G_i = (V_i, E_i)$.
    $n_i = |V_i|$.
    For each $v \in V'_i \setminus V''_i$, let $h_i(v) \leftarrow \min_{u \in \Gamma_{G'_i}(v) \cup \{v\}} u$. ▷ Contract small component to one vertex
    For each $v \in V''_i \setminus V_i$, let $h_i(v) \leftarrow g_i^{(r'_i)}(v)$. ▷ Contract non-leader to leader
    For each $v \in V$, if $h_{i-1}(v) \neq$ null, then let $h_i(v) = h_{i-1}(v)$.
  end for
  If $n_r \neq 0$, return FAIL.
  $((\widehat{V}, \widehat{E}), \mathrm{col}) =$ TREECONTRACTION(G, h_r). ▷ Algorithm 32.2
  return col .
end procedure

Algorithm 32.4 Local Complete Shortest Path Tree Expansion

procedure TREEEXPANSION($\widetilde{T}, \mathrm{dep}_{\widetilde{T}}, \{T(u) \mid u \in V_{\widetilde{T}}\}, \{\mathrm{dep}_{T(u)} \mid u \in V_{\widetilde{T}}\}$) ▷ Lemma 32.5.1

▷ $\widetilde{T} = (V_{\widetilde{T}}, \mathrm{par}_{\widetilde{T}})$ with root v and radius s_1 is a LCSPT in graph $G = (V, E)$.

▷ $\mathrm{dep}_{\widetilde{T}} : V_{\widetilde{T}} \to \mathbb{Z}_{\geq 0}$ records the depth of every vertex in $\widetilde{T}$.

▷ $\forall u \in V_{\widetilde{T}}, T(u) = (V_{T(u)}, \mathrm{par}_{T(u)})$ with root u and radius s_2 is a LCSPT in G.

▷ $\forall u \in V_{\widetilde{T}}, \mathrm{dep}_{T(u)} : V_{T(u)} \to \mathbb{Z}_{\geq 0}$ records the depth of every vertex in $T(u)$.

    Output: $\widehat{T} = (V_{\widehat{T}}, \mathrm{par}_{\widehat{T}}), \mathrm{dep}_{\widehat{T}}$.

    Let $V_{\widehat{T}} = \bigcup_{u \in \widetilde{T}} V_{T(u)}$.

    $\forall x \in V_{\widetilde{T}}, \mathrm{par}_{\widehat{T}}(x) \leftarrow \mathrm{par}_{\widetilde{T}}(x)$.

    $\forall x \in V_{\widetilde{T}}, h(x) \leftarrow \mathrm{dep}_{\widetilde{T}}(x)$.

    $\forall x \in V_{\widehat{T}} \setminus V_{\widetilde{T}}, u_x \leftarrow \arg\min_{u : u \in V_{\widetilde{T}}, x \in V_{T(u)}} \mathrm{dep}_{\widetilde{T}}(u) + \mathrm{dep}_{T(u)}(x), \mathrm{par}_{\widehat{T}}(x) \leftarrow \mathrm{par}_{T(u_x)}(x)$.

▷ u_x is on the shortest path from x to v.

    $\forall x \in V_{\widehat{T}} \setminus V_{\widetilde{T}}, h(x) \leftarrow \mathrm{dep}_{\widetilde{T}}(u_x) + \mathrm{dep}_{T(u_x)}(x)$.

    return $\widehat{T} = (V_{\widehat{T}}, \mathrm{par}_{\widehat{T}})$, and return $h : V_{\widehat{T}} :\to \mathbb{Z}_{\geq 0}$ as $\mathrm{dep}_{\widehat{T}}$.

end procedure

Algorithm 32.5 Doubling Algorithm for Local Complete Shortest Path Trees

procedure MULTIRADIUSLCSPT($G = (V, E), m$) ▷ Lemma 32.5.2
  Output: $r, \{T_i(v) \mid i \in \{0\} \cup [r], v \in V\}, \{\mathrm{dep}_{T_i(v)} \mid i \in \{0\} \cup [r], v \in V, T_i(v) \neq \text{null}\}$
  Initialization:
  $\forall v \in V$, if $|\{v\} \cup \Gamma_G(v)| < \lceil (m/n)^{1/4} \rceil$, then let $T_0(v) \leftarrow (\{v\} \cup \Gamma_G(v), \mathrm{par}_{T_0(v)})$,
  where $\mathrm{par}_{T_0(v)} : \{v\} \cup \Gamma_G(v) \to \{v\} \cup \Gamma_G(v)$, and $\forall u \in \{v\} \cup \Gamma_G(v), \mathrm{par}_{T_0(v)}(u) = v$.
  $\forall v \in V$, if $|\{v\} \cup \Gamma_G(v)| \geq \lceil (m/n)^{1/4} \rceil$, then let $T_0(v) \leftarrow$ null.
  $\forall v \in V$, if $T_0(v) \neq$ null, let $\mathrm{dep}_{T_0(v)} : V_{T_0(v)} \to \mathbb{Z}_{\geq 0}$ s.t. $\mathrm{dep}_{T_0(v)}(v) = 0$, $\forall u \in \Gamma_G(v), \mathrm{dep}_{T_0(v)}(u) = 1$.
  $r = 1$.
  Main Loop:
  for true do
    for $v \in V$ **do** ▷ If $T_r(v) \neq$ null, $T_r(v)$ is a local complete shortest path tree with radius 2^r.
      if $T_{r-1}(v)$ is null **then** $T_r(v) \leftarrow$ null.
      else if $\exists u \in V_{T_{r-1}(v)}$, $T_{r-1}(u)$ is null **then** $T_r(v) \leftarrow$ null.
      else
        $\left(T_r(v), \mathrm{dep}_{T_r(v)}\right) = \textsc{TreeExpansion}\left(T_{r-1}(v), \mathrm{dep}_{T_{r-1}(v)}, \bigcup_{u \in V_{T_{r-1}(v)}} \{T_{r-1}(u)\}, \bigcup_{u \in V_{T_{r-1}(v)}} \left\{\mathrm{dep}_{T_{r-1}(u)}\right\}\right)$.
        ▷ Algorithm 32.4
        If $|V_{T_r(v)}| \geq \lceil (m/n)^{1/4} \rceil$, let $T_r(v) \leftarrow$ null.
      end if
    end for
    if $\forall v \in V$ either $T_r(v) =$ null or $|V_{T_r(v)}| = |V_{T_{r-1}(v)}|$ **then**
      return $r, \{T_i(v) \mid i \in \{0\} \cup [r], v \in V\}, \{\mathrm{dep}_{T_i(v)} \mid i \in \{0\} \cup [r], v \in V, T_i(v) \neq \text{null}\}$
    end if
    $r \leftarrow r + 1$.
  end for
end procedure

Algorithm 32.6 Large Local Shortest Path Trees

procedure MULTIPLELARGETREES($G = (V, E), m$) ▷ Lemma 32.5.3, Lemma 32.5.6
  Output: $\{\widetilde{T}(v) \mid v \in V\}, \{\mathrm{dep}_{\widetilde{T}(v)} \mid v \in V\}$.
  $\left(r, \{T_i(v) \mid i \in \{0\} \cup [r], v \in V\}, \{\mathrm{dep}_{T_i(v)} \mid i \in \{0\} \cup [r], v \in V, T_i(v) \neq \text{null}\}\right) = \text{MULTIRADIUSLCSPT}(G, m)$. ▷ Algorithm 32.5
  $\forall v \in V$ with $T_r(v) \neq \text{null}$ let $\widetilde{T}(v) = T_r(v)$ and $\mathrm{dep}_{\widetilde{T}(v)} \leftarrow \mathrm{dep}_{T_r(v)}$.
  $\forall v \in V$ with $T_r(v) = \text{null}$, let $\widetilde{T}_0(v) = (\{v\}, \mathrm{par}_{\widetilde{T}_0(v)})$, $s_0(v) = 0$, and $\mathrm{dep}_{\widetilde{T}_0(v)} : \{v\} \to \mathbb{Z}_{\geq 0}$,
  where $\mathrm{par}_{\widetilde{T}_0(v)} : \{v\} \to \{v\}$ satisfies $\mathrm{par}_{\widetilde{T}_0(v)}(v) = v$, and $\mathrm{dep}_{\widetilde{T}_0(v)}(v) = 0$.
  for $i = 1 \to r$ **do**
    for $v \in \{u \in V \mid T_r(u) = \text{null}\}$ **do**
      if $\forall u \in V_{\widetilde{T}_{i-1}(v)}, T_{r-i}(u) \neq \text{null}$ **then**
        $\left(\widetilde{T}_i(v), \mathrm{dep}_{\widetilde{T}_i(v)}\right) = \text{TREEEXPANSION}\left(\widetilde{T}_{i-1}(v), \mathrm{dep}_{\widetilde{T}_{i-1}(v)}, \bigcup_{u \in V_{\widetilde{T}_{i-1}(v)}} \{T_{r-i}(u)\}, \bigcup_{u \in V_{\widetilde{T}_{i-1}(v)}} \left\{\mathrm{dep}_{T_{r-i}(u)}\right\}\right)$. ▷ Algorithm 32.4
        If $|V_{\widetilde{T}_i(v)}| < \lceil (m/n)^{1/4} \rceil$, then let $s_i(v) = s_{i-1}(v) + 2^{r-i}$.
        Otherwise, let $s_i(v) = s_{i-1}(v), \widetilde{T}_i(v) \leftarrow \widetilde{T}_{i-1}(v), \mathrm{dep}_{\widetilde{T}_i(v)} \leftarrow \mathrm{dep}_{\widetilde{T}_{i-1}(v)}$.
      else
        Let $s_i(v) = s_{i-1}(v), \widetilde{T}_i(v) = \widetilde{T}_{i-1}(v), \mathrm{dep}_{\widetilde{T}_i(v)} \leftarrow \mathrm{dep}_{\widetilde{T}_{i-1}(v)}$.
      end if
    end for
  end for ▷ $\widetilde{T}_r(v)$ is a LCSPT with root v and the largest radius s.t. $|V_{\widetilde{T}_r(v)}| < \lceil (m/n)^{1/4} \rceil$.
  $\forall v \in V$, if $|\Gamma_G(v) \cup \{v\}| \leq \lceil (m/n)^{1/4} \rceil$, then let $N(v) = \Gamma_G(v) \cup \{v\}$.
  Otherwise arbitrarily choose $N(v) \subseteq \Gamma_G(v) \cup \{v\}$ with $|N(v)| = \lceil (m/n)^{1/4} \rceil$.
  for $v \in \{u \in V \mid T_r(u) = \text{null}\}$ **do** ▷ Expand $\widetilde{T}_r(v)$ a little bit to get large enough $\widetilde{T}$.
    if $\forall u \in V_{\widetilde{T}_r(v)}, T_0(u) \neq \text{null}$ **then**
      $\left(\widetilde{T}(v), \mathrm{dep}_{\widetilde{T}(v)}\right) = \text{TREEEXPANSION}\left(\widetilde{T}_r(v), \mathrm{dep}_{\widetilde{T}_r(v)}, \bigcup_{u \in V_{\widetilde{T}_r(v)}} \{T_0(u)\}, \bigcup_{u \in V_{\widetilde{T}_r(v)}} \left\{\mathrm{dep}_{T_0(u)}\right\}\right)$. ▷ Algorithm 32.4
      else
        Select an arbitrary $u_v \in V_{\widetilde{T}_r(v)}$ with $T_0(u_v) = \text{null}$.
        Let $V_{\widetilde{T}(v)} = N(u_v) \cup V_{\widetilde{T}_r(v)}$.
        $\forall x \in V_{\widetilde{T}_r(v)}$, let $\mathrm{par}_{\widetilde{T}(v)}(x) = \mathrm{par}_{\widetilde{T}_r(v)}(x), \mathrm{dep}_{\widetilde{T}(v)}(x) = \mathrm{dep}_{\widetilde{T}_r(v)}(x)$.
        $\forall x \in N(u_v), x \notin V_{\widetilde{T}_r(v)}$, let $\mathrm{par}_{\widetilde{T}(v)}(x) = u_v, \mathrm{dep}_{\widetilde{T}(v)}(x) = \mathrm{dep}_{\widetilde{T}_r(v)}(u_v) + 1$.
        Let $\widetilde{T}(v) = (V_{\widetilde{T}(v)}, \mathrm{par}_{\widetilde{T}(v)})$.
      end if
    end for
    return $\{\widetilde{T}(v) \mid v \in V\}, \{\mathrm{dep}_{\widetilde{T}(v)} \mid v \in V\}$.
end procedure

Algorithm 32.7 Depth and Ancestors of Every Vertex

procedure FINDANCESTORS ($\mathrm{par} : V \to V$) ▷ Lemma 32.5.7
  For $v \in V$ let $g_0(v) = \mathrm{part}(v)$. If $\mathrm{part}(v) = v$, let $h_0(v) = 0$. Otherwise, let $h_0(v) =$ null.
  Let $l = 0$.
  for $\exists v \in V, h_l(v) = \text{null}$ **do**
    $l \leftarrow l + 1$.
    for $v \in V$ **do**
      Let $g_l(v) = g_{l-1}(g_{l-1}(v))$. ▷ g_l is $\mathrm{par}^{(2^l)}$.
      if $h_{l-1}(v) \neq \text{null}$ **then** $h_l(v) = h_{l-1}(v)$.
      else if $h_{l-1}(g_{l-1}(v)) \neq \text{null}$ **then** $h_l(v) = h_{l-1}(g_{l-1}(v)) + 2^{l-1}$.
      else $h_l(v) = \text{null}$.
      end if
    end for
  end for
  Let $r = l, \mathrm{dep}_{\mathrm{par}} \leftarrow h_r$.
  return $r, \mathrm{dep}_{\mathrm{par}}, \{g_i : V \to V \mid i \in \{0\} \cup [r]\}$. ▷ $\mathrm{dep}_{\mathrm{par}} : V \to \mathbb{Z}_{\geq 0}$
end procedure

Algorithm 32.8 Path in a Tree

procedure FINDPATH ($\mathrm{par} : V \to V, q \in V$) ▷ Lemma 32.5.9
  Output: $\mathrm{dep}_{\mathrm{par}} : V \to \mathbb{Z}_{\geq 0}, P \subseteq V, w \in V \cup \{\mathrm{null}\}$.
  $(r, \mathrm{dep}_{\mathrm{par}}, \{g_i \mid i \in \{0\} \cup [r]\}) = \text{FINDANCESTORS}(\mathrm{par})$ ▷ Algorithm 32.7
  Let $S_0 = \{(q, g_r(q))\}, k = \lceil \log(\mathrm{dep}_{\mathrm{par}}(q)) \rceil$. ▷ S_0 contains $(q,$ the root of $q)$
  for $i = 1 \to k$ **do** ▷ S_i is a set of segments partitioned the path from q to the root of q
    Let $S_i \leftarrow \emptyset$.
    for $(x, y) \in S_{i-1}$ **do**
      if $\mathrm{dep}_{\mathrm{par}}(x) - \mathrm{dep}_{\mathrm{par}}(y) > 2^{k-i}$ **then** $S_i \leftarrow S_i \cup \{(x, g_{k-i}(x)), (g_{k-i}(x), y)\}$.
      else $S_i \leftarrow S_i \cup \{(x, y)\}$.
      end if
    end for
  end for ▷ S_k only contains segments with length at most 1
  Let $P \leftarrow \{q\}$
  for $(x, y) \in S_k$ **do**
    Let $P \leftarrow P \cup \{y\}$
  end for
  Find $w \in P$ with $\mathrm{dep}_{\mathrm{par}}(w) = 1$. If w does not exist, let $w \leftarrow \mathrm{null}$.
  return $(\mathrm{dep}_{\mathrm{par}}, P, w)$
end procedure

Algorithm 32.9 Root Changing

procedure ROOTCHANGE($\mathrm{par} : V \to V, q \in V$) ▷ Lemma 32.5.11
  Output: $\widehat{\mathrm{par}} : V \to V$.
  $(\mathrm{dep}_{\mathrm{par}}, P, w) = \text{FINDPATH}(\mathrm{par}, q)$. ▷ Algorithm 32.8
  $\forall v \in V \setminus P$, let $\widehat{\mathrm{par}}(v) = \mathrm{part}(v)$.
  Let $\widehat{\mathrm{par}}(q) = q$.
  Let $h : \{0\} \cup [\mathrm{dep}_{\mathrm{par}}(q)] \to P$ such that $\forall i \in \{0\} \cup [\mathrm{dep}_{\mathrm{par}}(q)], h(i) = x$ where $\mathrm{dep}_{\mathrm{par}}(x) = i$.
  for $v \in P \setminus \{q\}$ **do** ▷ Reverse par of all the vertices on the path from q to the root of q.
    Let $\widehat{\mathrm{par}}(v) = h(\mathrm{dep}_{\mathrm{par}}(v) + 1)$.
  end for
  return $\widehat{\mathrm{par}}$.
end procedure

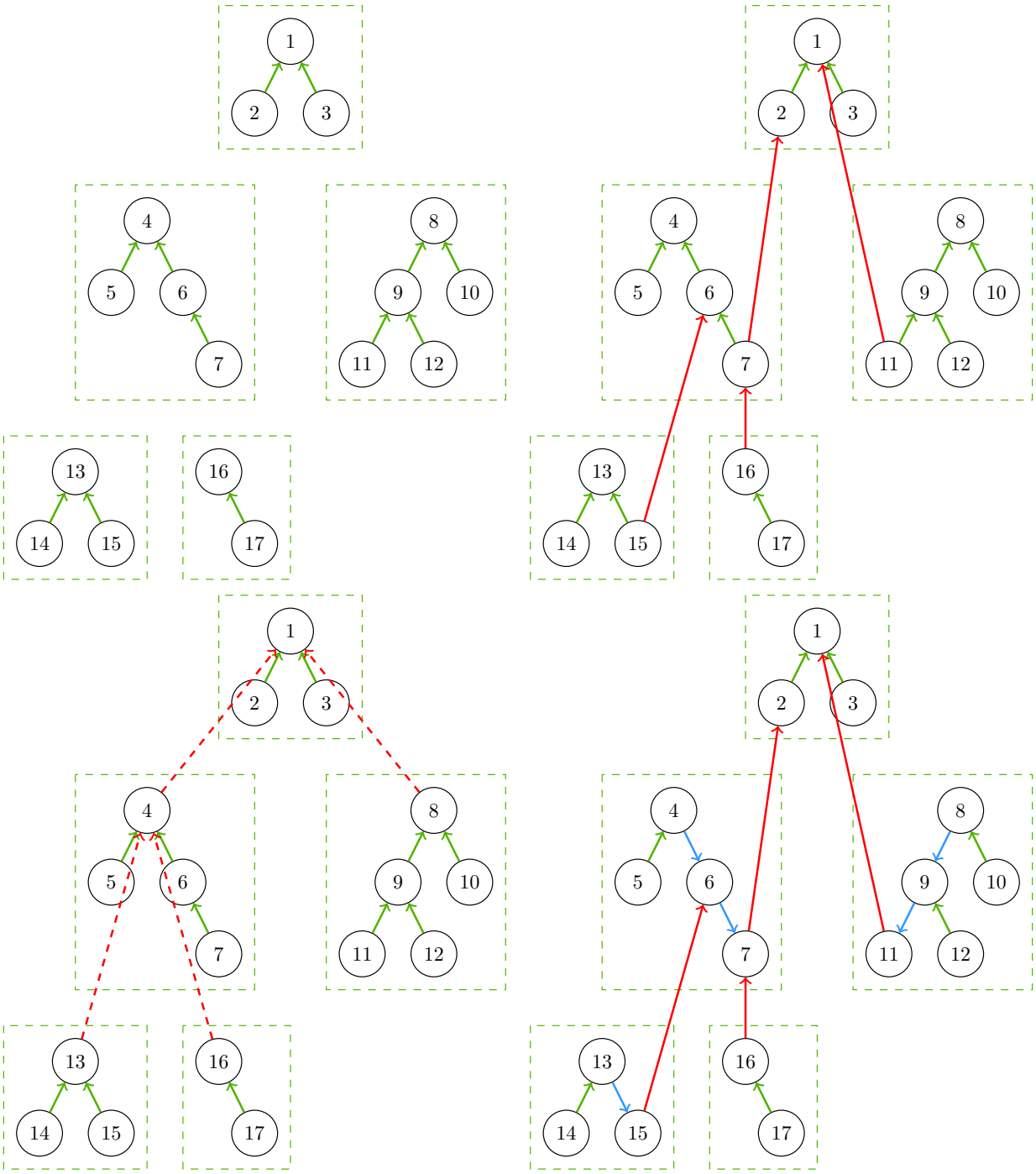


Figure 32.1: Each tree with green edges on the top-left is a rooted tree of each contracted component. For example, there are five components $\{1, 2, 3\}, \{4, 5, 6, 7\}$, $\{8, 9, 10, 11, 12\}, \{13, 14, 15\}, \{16, 17\}$. The dashed edges in the bottom-left figure is a root spanning tree of five components. The red edges in the top-right figure correspond to the dashed edges in the bottom-left figure before contraction. In bottom-right figure, by changing (see blue edges) the root of each contracted tree, we get a rooted spanning tree in the original graph

Algorithm 32.10 Spanning Forest Expansion

procedure ForestExpansion($\text{par} : V_1 \to V_1, \widetilde{\text{par}} : V_2 \to V_2, f : V_1 \times V_1 \to \{\text{null}\} \cup (V_2 \times V_2)$)
▷ Lemma 32.5.12
    Output: $\widehat{\text{par}} : V_2 \to V_2$.
    $((V_2', \emptyset), \widetilde{\text{par}}^{(\infty)}) = \text{TreeContraction}((V_2, \emptyset), \widetilde{\text{par}})$. ▷ Algorithm 32.2
    for $v \in V_1$ **do**
        Let $V_2(v) = \{u \in V_2 \mid \widetilde{\text{par}}^{(\infty)}(u) = v\}$.
        Let $\widetilde{\text{par}}_v : V_2(v) \to V_2(v)$ such that $\forall u \in V_2(v), \widetilde{\text{par}}_v(u) = \widetilde{\text{par}}(u)$.
        if $\text{part}(v) \neq v$ **then**
            Let $(x_v, y_v) = f(v, \text{part}(v))$.
            $\widehat{\text{par}}_v = \text{RootChange}(\widetilde{\text{par}}_v, x_v)$. ▷ Algorithm 32.9
            Let $\widehat{\text{par}}(x_v) = y_v$, and $\forall u \in V_2(v) \setminus \{x_v\}$, $\widehat{\text{par}}(u) = \widehat{\text{par}}_v(u)$.
        else $\forall u \in V_2(v)$ let $\widehat{\text{par}}(u) = \widetilde{\text{par}}_v(u)$.
        end if
    end for
    return $\widehat{\text{par}}$.
end procedure

Algorithm 32.11 Undirected Graph Spanning Forest

procedure SPANNINGFOREST($G = (V, E), m, r$) ▷ Corollary 32.5.17, Theorem 32.5.21
  Output: FAIL or $\{V_i \subseteq V \mid i \in \{0\} \cup [r]\}, \{\mathrm{par}_i : V_i \to V_i \mid i \in \{0\} \cup [r-1]\}, \{h_i : V_i \to V_{i+1} \cup \{\mathrm{null}\} \mid i \in \{0\} \cup [r-1]\}, F \subseteq E$.
  $n_0 = n = |V|, G_0 = (V_0, E_0) = (V, E)$.
  Let $g_0 : E_0 \to E$ be an identity map.
  Let $n'_0 = n_0$.
  for $i = 0 \to r-1$ **do**
    $D_i \leftarrow \emptyset$.
    $\left(\{\widetilde{T}_i(v) \mid v \in V_i\}, \{\mathrm{dep}_{\widetilde{T}_i(v)} \mid v \in V_i\}\right) = \textsc{MultipleLargeTrees}(G_i, m)$. ▷ Algorithm 32.6
    Let $V'_i = \{v \in V_i \mid |V_{\widetilde{T}_i(v)}| \geq \lceil (m/n_i)^{1/4} \rceil\}, E'_i = \{(u, v) \in E_i \mid u, v \in V'_i\}, G'_i = (V'_i, E'_i)$.
    $\forall v \in V_i \setminus V'_i$, let $h_i(v) = \mathrm{null}, u_v = \min_{u \in V_{\widetilde{T}_i(v)}} u$. Let $\mathrm{par}_i(v) = \mathrm{par}_{\widetilde{T}_i(u_v)}(v)$.
    $\forall v \in V_i \setminus V'_i$, if $\mathrm{par}_i(v) \neq v$, then $D_i \leftarrow D_i \cup \{g_i(\mathrm{par}_i(v), v), g_i(v, \mathrm{par}_i(v))\}$.
    Let $\gamma_i = \lceil (m/n_i)^{1/4} \rceil, p_i = \min((30 \log(n) + 100)/\gamma_i, 1/2)$.
    Let $l_i : V'_i \to \{0, 1\}$ be chosen randomly s.t. $\forall v \in V'_i, l_i(v)$ are i.i.d. Bernoulli random variables with $\Pr(l_i(v) = 1) = p_i$.
    Let $L_i = \{v \in V'_i \mid l_i(v) = 1\} \cup \{v \in V'_i \mid \forall u \in V_{\widetilde{T}_i(v)}, l_i(u) = 0\}$.
    For $v \in V'_i$, let $z_i(v) = \arg\min_{u \in L_i \cap V_{\widetilde{T}_i(v)}} \mathrm{dep}_{\widetilde{T}_i(v)}(u)$. If $z_i(v) = v$, let $\mathrm{par}_i(v) = v$.
    Otherwise, $(\mathrm{dep}_{\widetilde{T}_i(v)}, P_i(v), w_i(v)) = \textsc{FindPath}(\mathrm{par}_{\widetilde{T}_i(v)}, z_i(v))$, and let $\mathrm{par}_i(v) = w_i(v)$.
    ▷ Algorithm 32.8
    Let $((V_{i+1}, E_{i+1}), \mathrm{par}_i^{(\infty)}) = \textsc{TreeContraction}(G'_i, \mathrm{par}_i : V'_i \to V'_i)$. ▷ Algorithm 32.2
    $G_{i+1} = (V_{i+1}, E_{i+1}), n_{i+1} = |V_{i+1}|$.
    $\forall v \in V'_i$, $h_i(v) = \mathrm{par}_i^{(\infty)}(v)$. If $\mathrm{par}_i(v) \neq v$, then $D_i \leftarrow D_i \cup \{g_i(\mathrm{par}_i(v), v), g_i(v, \mathrm{par}_i(v))\}$.
    Let $g_{i+1} : E_{i+1} \to E$ satisfy $g_{i+1}(u, v) = \min_{(x,y) \in E_i, h_i(x) = u, h_i(y) = v} g_i(x, y)$.
    Let $n'_{i+1} = n'_i + n_{i+1}$. If $n'_{i+1} > 40n$, then return FAIL.
  end for
  If $n_r \neq 0$, return FAIL.
  Let $F = \bigcup_{i \in \{0\} \cup [r-1]} D_i$.
  return $\{V_i \mid i \in \{0\} \cup [r]\}, \{\mathrm{par}_i \mid i \in \{0\} \cup [r-1]\}, \{h_i \mid i \in \{0\} \cup [r-1]\}, F$.
end procedure

Algorithm 32.12 Rooted Spanning Forest

procedure ORIENTATE($\{V_i \mid i \in \{0\} \cup [r]\}, \{\mathrm{par}_i \mid i \in \{0\} \cup [r-1]\}, \{h_i \mid i \in \{0\} \cup [r-1]\}, F$)
▷ Takes the output of Algorithm 32.11 as input.
▷ Theorem 32.5.19
    Output: $\mathrm{par} : V_0 \to V_0$.
    Let $F_0 = F$.
    for $i = 0 \to r-1$ **do**
        Initialize $F_{i+1} \leftarrow \emptyset$, $f_{i+1} : V_{i+1} \times V_{i+1} \to \{\mathrm{null}\}$.
        $\forall (u,v) \in F_i, h_i(u) \neq h_i(v)$, let $F_{i+1} \leftarrow F_{i+1} \cup \{(h_i(u), h_i(v))\}$, $f_{i+1}(h_i(u), h_i(v)) \leftarrow (u,v)$.
    end for
    $\widehat{\mathrm{par}}_r : \emptyset \to \emptyset$.
    for $i = r \to 1$ **do** ▷ $\widehat{\mathrm{par}}_i$ is the spanning forest of G_i.
        Let $\widetilde{V}_i = V_i \cup \{v \in V_{i-1} \mid h_{i-1}(v) = \mathrm{null}, \mathrm{par}_{i-1}(v) = v\}$.
        Let $\widetilde{\mathrm{par}}_i : \widetilde{V}_i \to \widetilde{V}_i$ satisfy $\forall v \in V_i$, $\widetilde{\mathrm{par}}_i(v) = \widehat{\mathrm{par}}_i(v)$, and $\forall v \in \widetilde{V}_i \setminus V_i, \widetilde{\mathrm{par}}_i(v) = v$.
        Let $\widehat{\mathrm{par}}_{i-1} = \textsc{ForestExpansion}(\widetilde{\mathrm{par}}_i, \mathrm{par}_{i-1}, f_i)$. ▷ Algorithm 32.10
    end for
    Return $\widehat{\mathrm{par}}_0$ as par .
end procedure

Algorithm 32.13 Lowest Common Ancestor

procedure LCA(par : $V \to V, Q = \{(u_1, v_1), (u_2, v_2), \cdots, (u_q, v_q)\}$) ▷ Lemma 32.6.1
  Output: lca : $Q \to (V \cup \{\text{null}\}) \times (V \cup \{\text{null}\}) \times (V \cup \{\text{null}\})$
  $(r, \text{dep}_{\text{par}}, \{g_i \mid i \in \{0\} \cup [r]\}) = \text{FINDANCESTORS}(\text{par})$. ▷ Algorithm 32.7
  $\forall (u, v) \in Q$, if $u = v$ then let $\text{lca}(u, v) = (u, \text{null}, \text{null}), Q \leftarrow Q \setminus \{(u, v)\}$.
  $\forall (u, v) \in Q, g_r(u) \neq g_r(v)$, let $\text{lca}(u, v) = (\text{null}, \text{null}, \text{null})$.
  Let $Q' = \emptyset$.
  $\forall (u, v) \in Q, g_r(u) = g_r(v)$, if $\text{dep}_{\text{par}}(u) \geq \text{dep}_{\text{par}}(v)$, then let $Q' \leftarrow Q' \cup \{(u, v)\}$; Otherwise let $Q' \leftarrow Q' \cup \{(v, u)\}$.
  Let $h_r : Q' \to Q'$ be an identity mapping.
  for $i = r - 1 \to 0$ **do** ▷ Move u to the almost same depth as v.
    For each $(u, v) \in Q'$, let $(x, v) = h_{i+1}(u, v)$. If $\text{dep}_{\text{par}}(x) - 2^i > \text{dep}_{\text{par}}(y)$, then let $h_i(u, v) = (g_i(x), v)$; Otherwise let $h_i(u, v) = (x, v)$.
  end for
  For each $(u, v) \in Q'$ with $\text{part}(h_0(u)) = v$, if $(u, v) \in Q$, then let $\text{lca}(u, v) = (v, h_0(u), \text{null})$; Otherwise $\text{lca}(v, u) = (v, \text{null}, h_0(u))$.
  Let $Q'' = \emptyset$.
  For each $(u, v) \in Q'$ with $\text{part}(h_0(u)) \neq v, \text{dep}_{\text{par}}(h_0(u)) > \text{dep}_{\text{par}}(v)$ let $Q'' \leftarrow Q'' \cup \{(u, v)\}$, $h'_r(u, v) \leftarrow (\text{part}(h_0(u)), v)$.
  For each $(u, v) \in Q'$ with $\text{dep}_{\text{par}}(u) = \text{dep}_{\text{par}}(v)$ let $Q'' \leftarrow Q'' \cup \{(u, v)\}$, $h'_r(u, v) \leftarrow (u, v)$.
  for $i = r - 1 \to 0$ **do** ▷ Move u, v to the lowest common ancestor.
    For each $(u, v) \in Q''$, let $(x, y) = h'_{i+1}(u, v)$. If $g_i(x) \neq g_i(y)$, then let $h'_i(u, v) = (g_i(x), g_i(y))$; Otherwise let $h'_i(u, v) = (x, y)$.
  end for
  For each $(u, v) \in Q''$, if $(u, v) \in Q$, then let $\text{lca}(u, v) = (\text{part}(h'_0(u)), h'_0(u), h'_0(v))$; Otherwise $\text{lca}(v, u) = (\text{part}(h'_0(v)), h'_0(v), h'_0(u))$.
  return lca .
end procedure

Algorithm 32.14 Multiple Paths

procedure MULTIPATH(par : $V \to V, Q = \{(u_1, v_1), (u_2, v_2), \cdots, (u_q, v_q)\}$) ▷ Lemma 32.6.2
    Output: $\mathrm{dep}_{\mathrm{par}} : V \to \mathbb{Z}_{\geq 0}, \{P_i \subseteq V \mid i \in [q]\}$.
    $(r, \mathrm{dep}_{\mathrm{par}}, \{g_i \mid i \in \{0\} \cup [r]\}) = $ FINDANCESTORS(par). ▷ Algorithm 32.7
    $\forall j \in [q]$, let $S_j^{(0)} = \{(u_j, v_j) \mid (u_j, v_j) \in Q\}$.
    for $i = 1 \to r$ **do**
        for $j = 1 \to q$ **do** ▷ $S_j^{(i)}$ is a set of segments partitioned the path from u_j to v_j.
            Let $S_j^{(i)} \leftarrow \emptyset$.
            for $(x, y) \in S_j^{(i-1)}$ **do**
                if $\mathrm{dep}_{\mathrm{par}}(x) - \mathrm{dep}_{\mathrm{par}}(y) > 2^{r-i}$ **then** $S_j^{(i)} \leftarrow S_j^{(i)} \cup \{(x, g_{r-i}(x)), (g_{r-i}(x), y)\}$.
                else $S_j^{(i)} \leftarrow S_j^{(i)} \cup \{(x, y)\}$.
                end if
            end for
        end for
    end for ▷ $S_j^{(r)}$ only contains segments with length 1
    Let $\forall j \in [q], P_j \leftarrow \{u_j\}$.
    for $j = 1 \to q$ **do**
        for $(x, y) \in S_j^{(r)}$ **do**
            Let $P_j \leftarrow P_j \cup \{y\}$.
        end for
    end for
end procedure

Algorithm 32.15 Leaf Sampling

procedure LeafSampling(par : $V \to V, m, \delta$) ▷ Lemma 32.6.5
  Output: $A = (a_1, a_2, \cdots, a_s)$.
  Let $t = \lceil m^{1/3} \rceil$.
  Compute $L = \text{leaves}(\text{par})$.
  Compute rank : $V \to \mathbb{Z}_{\geq 0}$ such that $\forall v \in V, \text{rank}(v) = \text{rank}_{\text{par}}(v)$. ▷ Definition 32.6.4
  If $|V| \leq m$, let $\{a_1, a_2, \cdots, a_s\} = L$, and return $A = (a_1, a_2, \cdots, a_s)$ which satisfies $a_1 <_{\text{par}} a_2 <_{\text{par}} \cdots <_{\text{par}} a_s$. ▷ $<_{\text{par}}$ follows Definition 32.6.8
  If $|L| \leq 8t$, let $S = L$.
  Let $p = \min(1, 640(1 + \log(m)/\delta)t/|L|)$.
  If $|L| > t$, sample each $v \in L$ with probability p independently. let S be the set of samples.
  Compute $\text{par}' : V \to V$ such that $\forall v \in V$, if $\text{child}_{\text{par}}(v) \neq \emptyset$, then $\text{par}'(v) = \text{child}_{\text{par}}(v, 1)$; Otherwise let $\text{par}'(v) = v$. ▷ $\text{par}'(v)$ points to v's first child in par.
  $(r', \text{dep}_{\text{par}'} : V \to \mathbb{Z}_{\geq 0}, \{g'_i : V \to V \mid i \in \{0\} \cup [r']\}) = \text{FindAncestors}(\text{par}')$.
  Find $w \in V$ with $\text{part}(w) = w$. ▷ Find the root.
  Let $a_1 = g'_{r'}(w), S \leftarrow S \cup \{a_1\}$. ▷ Find the first leaf.
  Let $Q = \{(u, v) \mid (u, v) \in S \times S, u \neq v\}$.
  Let $\text{lca} = \text{LCA}(\text{par}, Q)$. ▷ Algorithm 32.13
  Let $s = |S|$.
  $(r, \text{dep}_{\text{par}} : V \to \mathbb{Z}_{\geq 0}, \{g_i : V \to V \mid i \in \{0\} \cup [r]\}) = \text{FindAncestors}(\text{par})$.
  for $i = 2 \to s$ **do** ▷ Determine the order of sampled leaves.
    For all $x, y \in S \setminus \{a_1, a_2, \cdots, a_{i-1}\}$, let $(p_{x,y}, p_{xy,x}, p_{xy,y}) = \text{lca}(x, y)$.
    Find $x^* \in S \setminus \{a_1, a_2, \cdots, a_{i-1}\}$ s.t. $\forall y \in S \setminus \{a_1, a_2, \cdots, a_{i-1}, x^*\}$, $\text{rank}(p_{x^*y,x^*}) < \text{rank}(p_{x^*y,y})$.
    Let $a_i = x^*$.
  end for
  return $A = (a_1, a_2, \cdots, a_s)$.
end procedure

Algorithm 32.16 DFS subsequence

procedure SUBDFS(par : $V \to V, m, \delta$) ▷ Lemma 32.6.9, Lemma 32.6.10
  Output: $V' \subseteq V, A = (a_1, a_2, \cdots, a_s)$.
  If $V = \{v\}$, return $V' = V, A = (v)$.
  Let v be the root in par, i.e. $\text{part}(v) = v$.
  $L = (l_1, l_2, \cdots, l_t) = \text{LEAFSAMPLING}(\text{par}, m, \delta)$. ▷ Algorithm 32.15
  $Q = \{(l_i, l_{i+1}) \mid i \in [t-1]\}$.
  $\text{lca} = \text{LCA}(\text{par}, Q)$. ▷ Algorithm 32.13
  $\forall i \in [t-1], (p_{l_i,l_{i+1}}, p_{i,l_i}, p_{i,l_{i+1}}) = \text{lca}(l_i, l_{i+1})$.
  $Q' = \{(l_1, v), (\text{part}(l_1), p_{l_1,l_2}), (l_2, p_{1,l_2}), (\text{part}(l_2), p_{l_2,l_3}), (l_3, p_{2,l_3}), \cdots, (l_t, p_{t-1,l_t}), (\text{part}(l_t), v)\}$.
  $(\text{dep}_{\text{par}}, \{P_i \mid i \in [2t]\}) = \text{MULTIPATH}(\text{par}, Q')$. ▷ Algorithm 32.14
  $V' = \bigcup_{i=1}^{2t} P_i$.
  Let $\text{par}' : V' \to V'$ satisfy $\forall v \in V', \text{par}'(v) = \text{part}(v)$.
  for $i \in \{1, 3, 5, \cdots, 2t-1\}$ **do**
    Compute $A'_i = (u_1, u_2, \cdots, u_{|P_i|})$ such that $\{u_1, u_2, \cdots, u_{|P_i|}\} = P_i$ and $\text{dep}_{\text{par}}(u_1) < \text{dep}_{\text{par}}(u_2) < \cdots < \text{dep}_{\text{par}}(u_{|P_i|})$
  end for
  for $i \in \{2, 4, 6, \cdots, 2t\}$ **do**
    Compute $A'_i = (u_1, u_2, \cdots, u_{|P_i|})$ such that $\{u_1, u_2, \cdots, u_{|P_i|}\} = P_i$ and $\text{dep}_{\text{par}}(u_1) > \text{dep}_{\text{par}}(u_2) > \cdots > \text{dep}_{\text{par}}(u_{|P_i|})$
  end for
  Let $A' = A'_1 A'_2 \cdots A'_{2t}$. ▷ A' is the concatenation of $A'_1, A'_2, \cdots, A'_{2t}$.
  $\forall u \in V'$, compute $\text{rank}_{\text{par}}(u)$ and $\text{rank}_{\text{par}'}(u)$.
  $\forall u \in V', i \in [|\text{child}_{\text{par}'}| + 1]$ compute $\text{pos}(u, i) = j$ such that the j^{th} element in A' is the i^{th} time that u appears.
  Let b be the length of A'.
  Initialize $c : [b] \to \mathbb{Z}_{\geq 0}$. ▷ c determine the number of copies needed for each element in A'
  for $u \in V' \setminus \{v\}$ **do**
    If $u \in \text{leaves}(\text{par}')$, let $c(\text{pos}(u, 1)) = 1$. ▷ A leaf should only have one copy.
    If $\text{rank}_{\text{par}'}(u) = 1$, let $c(\text{pos}(\text{par}'(u), 1)) = \text{rank}_{\text{par}}(u)$.
    If $\text{rank}_{\text{par}'}(u) = |\text{child}_{\text{par}'}(\text{par}'(u))|$, let $c(\text{pos}(\text{par}'(u), \text{rank}_{\text{par}'}(u) + 1)) = |\text{child}_{\text{par}}(\text{part}(u))| + 1 - \text{rank}_{\text{par}}(u)$.
    If $1 \leq \text{rank}_{\text{par}'}(u) < |\text{child}_{\text{par}'}(\text{par}'(u))|$, let $c(\text{pos}(\text{par}'(u), \text{rank}_{\text{par}'}(u) + 1)) = \text{rank}_{\text{par}}(\text{child}_{\text{par}'}(\text{par}'(u), \text{rank}_{\text{par}'}(u) + 1)) - \text{rank}_{\text{par}}(u)$.
  end for
  For each $j \in [b]$, duplicate the j^{th} element of A' $c(j)$ times. Let A be the obtained sequence.
  return V', A.
end procedure

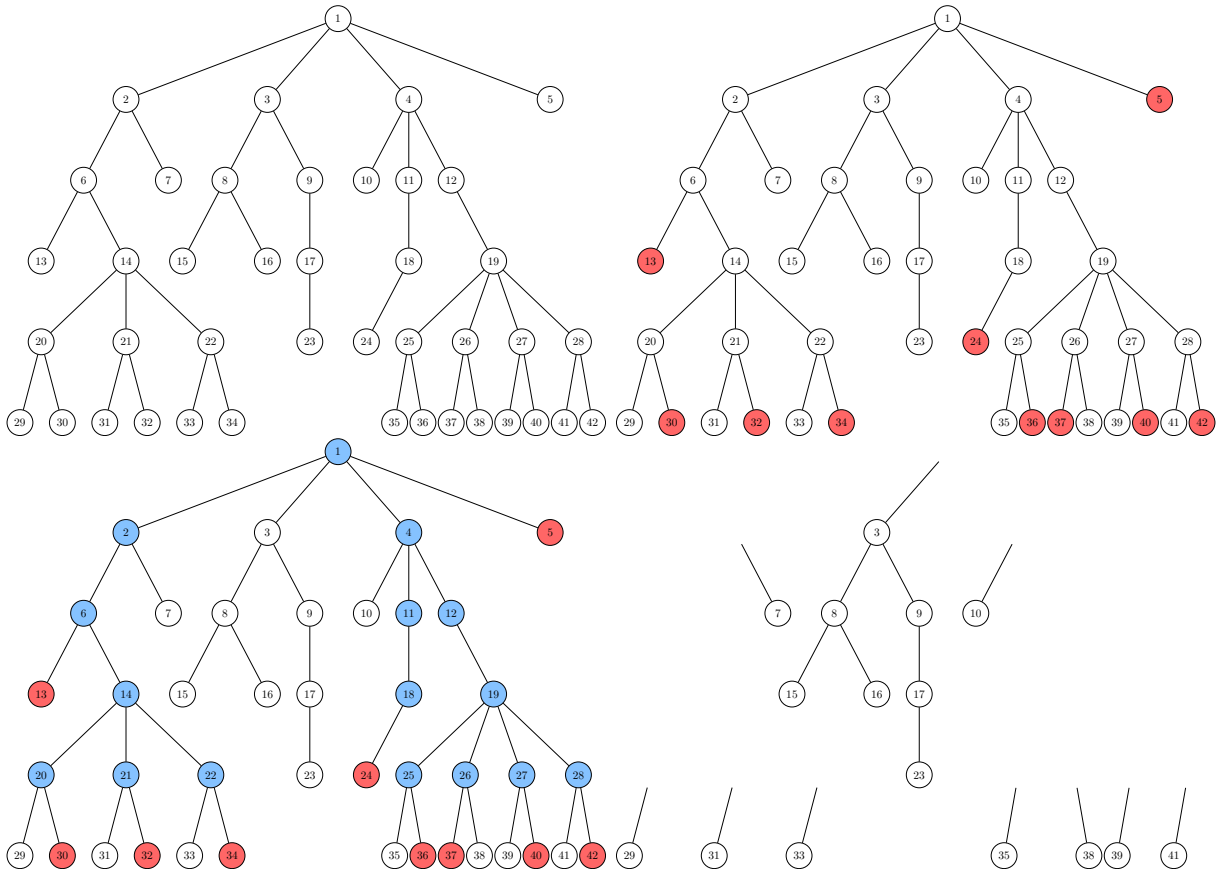


Figure 32.2: Given a tree that has 42 vertices (top-left), we label all the vertices from 1 to 42. Firstly, we sample some leaves (red vertices, i.e. $\{5, 13, 24, 30, 32, 34, 36, 37, 40, 42\}$) in the tree (top-right tree). Then we find a DFS sequence of the tree (the tree formed by all the blue and red vertices in the bottom-left tree) which only contains all the sampled leaves and their ancestors. Finally, we recursively find the DFS sequences of remaining subtrees(bottom-right).

Algorithm 32.17 DFS sequence

procedure DFS($\mathrm{par} : V \to V, m$) ▷ Theorem 32.6.12, Theorem 32.6.14
  Output: FAIL or $A = (a_1, a_2, \cdots, a_{2|V|-1})$.
  $n = |V|, \delta = 1/\log_m n$.
  Let $\mathrm{par}_0 = \mathrm{par}$.
  $(V_0, A_0) = \textsc{SubDFS}(\mathrm{par}_0, m, \delta)$. ▷ Algorithm 32.16
  Let $r = \lceil 3/\delta \rceil + 2$.
  for $i = 1 \to r$ **do** ▷ $v \in V_i \Leftrightarrow v$ appears in A_i
    ▷ If $v \in V_i$, then v appears $|\mathrm{child}_{\mathrm{par}}(v)| + 1$ times in A_i
    Let $V_i' = V \setminus V_{i-1}$.
    Initialize $\mathrm{par}_i : V_i' \to V_i'$.
    For $v \in V_i'$, if $\mathrm{part}(v) \in V_{i-1}$, let $\mathrm{par}_i(v) = v$; Otherwise, let $\mathrm{par}_i(v) = \mathrm{part}(v)$.
    $((V_i'', \emptyset), \mathrm{par}_i^{(\infty)}) = \textsc{TreeContraction}((V_i', \emptyset), \mathrm{par}_i)$. ▷ Algorithm 32.2
    $V_i \leftarrow V_{i-1}$.
    $A_i \leftarrow A_{i-1}$.
    for $v \in V_i', \mathrm{par}_i(v) = v$ **do** ▷ The DFS sequence of the subtree of v in par is missing.
      Let $V_i'(v) = \{u \in V_i' \mid \mathrm{par}_i^{(\infty)}(u) = v\}$.
      Let $\mathrm{par}_{i,v} : V_i'(v) \to V_i'(v)$ satisfy $\forall u \in V_i'(v), \mathrm{par}_{i,v}(u) = \mathrm{par}_i(u)$.
      Let $(V_{i,v}, A_{i,v}) = \textsc{SubDFS}(\mathrm{par}_{i,v}, m, \delta)$. ▷ Algorithm 32.16
      $V_i \leftarrow V_i \cup V_{i,v}$.
      Insert $A_{i,v}$ after the $\mathrm{rank}_{\mathrm{par}}(v)^{\mathrm{th}}$ time appearance of v in A_i.
    end for
  end for
  If $V_r = V$, return A_r as A. Otherwise, return FAIL.
end procedure

Algorithm 32.18 A Sparser Table for RMQ

procedure SPARSETABLE$^+(a_1, a_2, \cdots, a_n, \delta)$ ▷ Lemma 32.6.16

▷ Output: $\widehat{f}_{i,j}$ for $i \in [n], j \in \{0\} \cup [\lceil 1/\delta \rceil]$

Initially, for all $i \in [n]$ let $\widehat{f}_{i,0} = i$. $\forall i > n, j \in \mathbb{Z}$, let $\widehat{f}_{i,j} = 0$, and let $a_0 = \infty$. Let $m = \lceil n^\delta \rceil$.

For $t \in [\lceil 1/\delta \rceil]$, let $S_t = \{x \mid \exists y \in [m-1], x = y \cdot m^t\}$.

for $l = 1 \to \lceil 1/\delta \rceil$ **do**

for $j = 0 \to \lceil n/m \rceil$ **do**

$i \leftarrow j \cdot m + 1$.

$z^*_{j,l} \leftarrow \arg\min_{z:t\in[l-1],x\in S_t,z=\widehat{f}_{j\cdot m+1+x,t}} a_z$.

for $i' = 0 \to \min(m-1, n-i)$ **do**

$T \leftarrow \{x \in \mathbb{Z} \mid i + i' \le x \le i + m - 1\} \cup \{x \in \mathbb{Z} \mid i + m^l \le x \le i + m^l + i' - 1\} \cup \{z^*_{j,l}\}$

$\widehat{f}_{i+i',l} = \arg\min_{z\in T} a_z$.

end for

end for

$l \leftarrow l + 1$.

end for

return $\widehat{f}_{i,j}$ for $i \in [n], j \in \{0\} \cup [\lceil 1/\delta \rceil]$.

end procedure

Algorithm 32.19 A Sparse Table for RMQ

procedure SPARSETABLE($a_1, a_2, \cdots, a_n, \delta$) ▷ Lemma 32.6.17
▷ Output: $f_{i,j}$ for $i \in [n], j \in \{0\} \cup [\lceil \log n \rceil]$.
Initially, for all $i \in [n]$ let $f_{i,0} = i$. $\forall i > n, j \in \mathbb{Z}$, let $f_{i,j} = 0$, and let $a_0 = \infty$. Let $m = \lceil n^\delta \rceil$.
Let $\{\widehat{f}_{p,q} \mid p \in [n], q \in \{0\} \cup \lceil 1/\delta \rceil\} = \text{SPARSETABLE}^+(a_1, a_2, \cdots, a_n, \delta)$. ▷ Algorithm 32.18
Let all undefined $\widehat{f}_{p,q}$ be 0.
for $t \in [\lceil \log n \rceil]$ **do**
if $2^t \leq m$ **then**
$k_t \leftarrow -1$
$S_t \leftarrow \emptyset$
else
$k_t \leftarrow \lfloor \log_m(2^t - m) \rfloor$
$S_t \leftarrow \{x \mid x \in [2^t - m - m^{k_t} + 1] \text{ s.t. } x \equiv 1 \pmod{m^{k_t}} \text{ or } (2^t - m - x) \equiv -1 \pmod{m^{k_t}}\}$
end if
end for
for $j = 0 \to \lceil n/m \rceil$ **do**
for $t = 0 \to \lceil \log n \rceil$ **do**
$i \leftarrow j \cdot m + 1$.
$z^*_{j,t} \leftarrow \arg\min_{z: x \in S_t, z = \widehat{f}_{j \cdot m + m + x, k_t}} a_z$.
for $i' = 0 \to \min(m - 1, n - i)$ **do**
$T_1 \leftarrow \{x \in \mathbb{Z} \mid i + i' \leq x \leq \min(i + m - 1, i + i' + 2^t - 1)\}$
$T_2 \leftarrow \{x \in \mathbb{Z} \mid \max(i + 2^t, i + i') \leq x \leq i + 2^t + i' - 1\}$
$T \leftarrow T_1 \cup T_2 \cup \{z^*_{j,t}\}$
$f_{i+i',t} = \arg\min_{z \in T} a_z$.
end for
end for
end for
return $f_{i,j}$ for $i \in [n], j \in \{0\} \cup [\lceil \log n \rceil]$.
end procedure

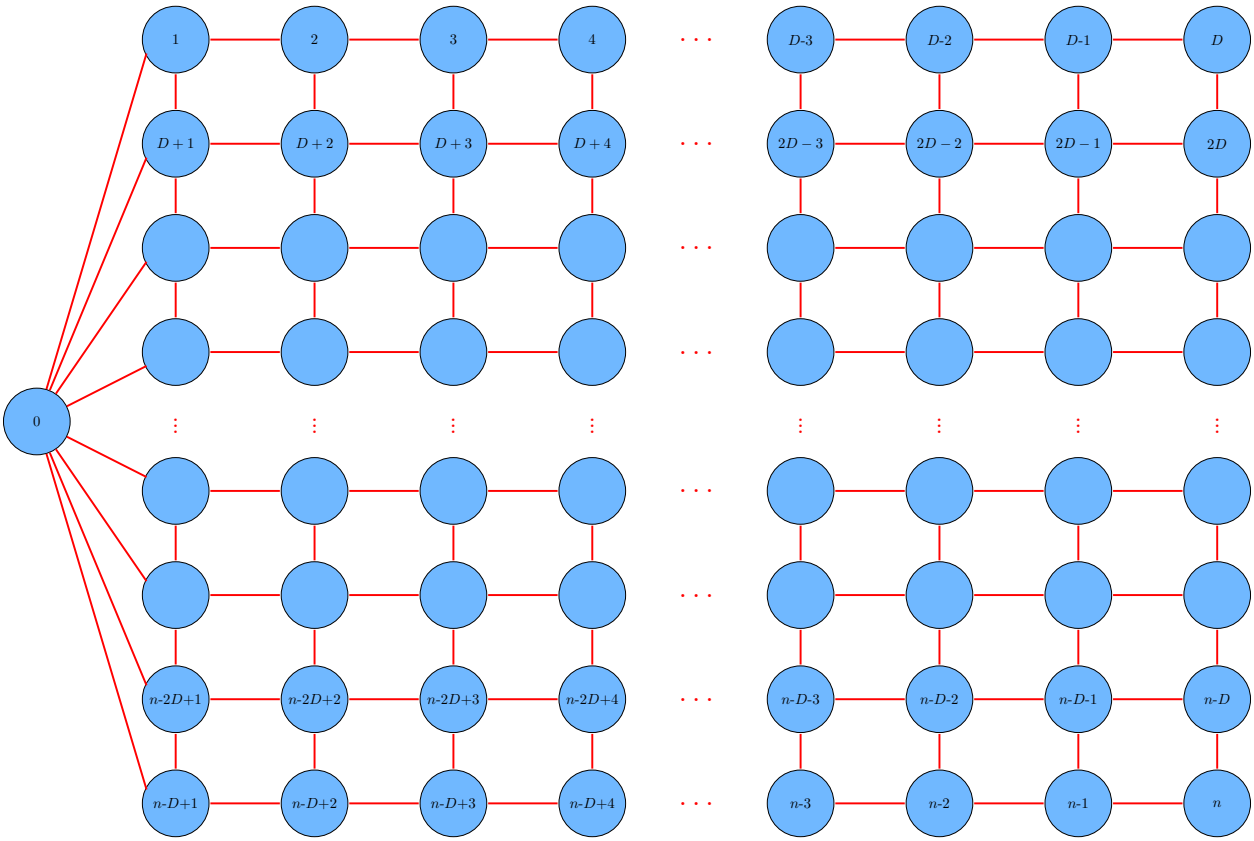


Figure 32.3: A hard example for [RMCS13]. For each $i \in \{2, 3, \cdots, n/D - 1\}$ and $j \in \{1, 2, \cdots, D-1\}$, node $(i-1) \cdot D + j$ has degree 4. For node D and n, they have degree 2. Node 0 has degree D. All the other nodes have degree 3.

Algorithm 32.20 Leader Selection via Min Parent Forest

1: Let $N = 100rn^{10}$.
2: $\forall v \in V_i'$, let $w_i(v)$ be i.i.d. random variables drawn uniformly from $[N]$.
3: $\forall v \in V_i''$, let $\text{par}_i(v) = f_{G_i', w_i}(v)$. ▷ (V_i', f_{G_i', w_i}) is a *min-parent-forest* of G_i' with w_i.

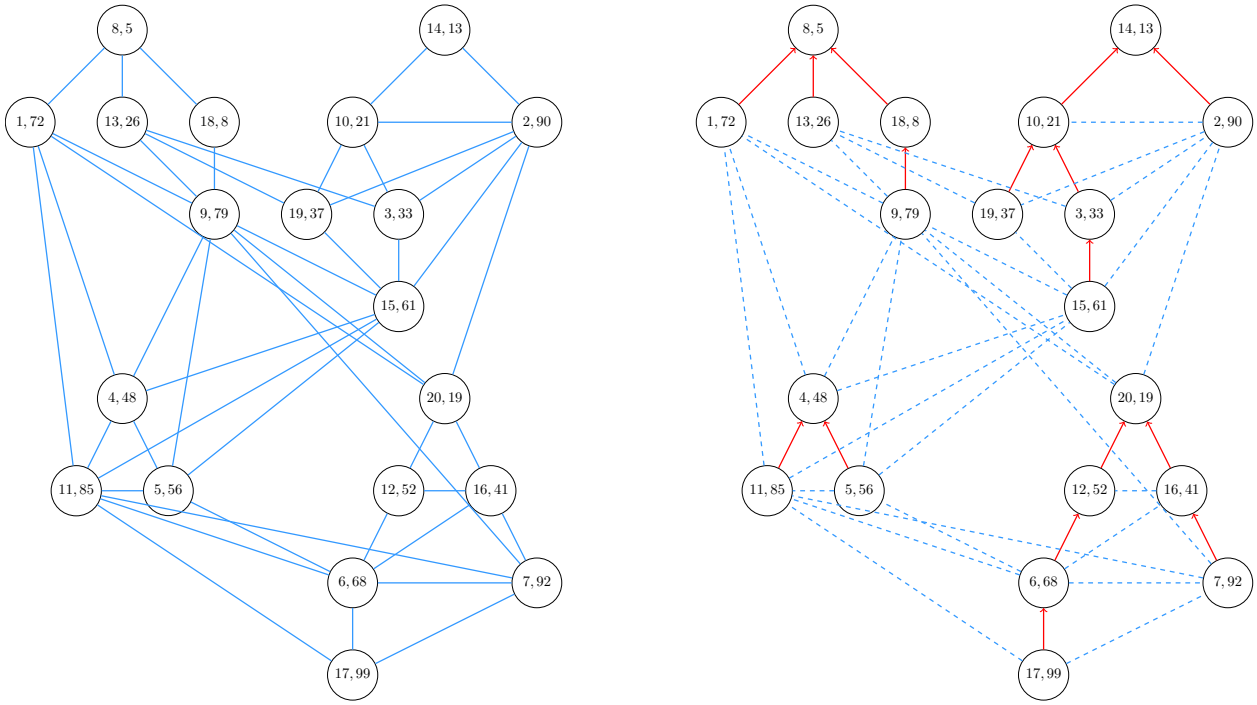


Figure 32.4: An example where #roots $\approx \sum_{i=1}^{20} 1/(d(v_i)+1)$. For each node, it has two numbers, the first number is the ID, and the second number is weight. $\sum_{i=1}^{20} 1/(d(v_i)+1) = 1/4+1/3+1/3+1/6+1/5+1/5+1/5+1/5+1/5+1/6+1/4+1/6+1/8+1/7+1/6+1/9+1/8+1/7+1/6+1/4 \approx 3.89$ and #roots$= 4$.

Appendices

Appendix A

Probability and Inequality Tools

A.1 Scalar version probability tools

Lemma A.1.1 (Chernoff bounds [Che52]). *Let $x_1, \cdots, x_n$ be independent random variables. Assume that $x_i \in [0,1]$ always, for each $i \in [n]$. Let $x = \sum_{i=1}^n$ and $\mu = \mathbb{E}[x] = \sum_{i=1}^n \mathbb{E}[x_i]$. Then for any $\epsilon > 0$*

$$\Pr[x \geq (1+\epsilon)\mu] \leq \exp\left(-\frac{\epsilon^2}{2+\epsilon}\mu\right), \quad \text{and} \quad \Pr[x \leq (1-\epsilon)\mu] \leq \exp\left(-\frac{\epsilon^2}{2}\mu\right).$$

Lemma A.1.2 (Bernstein inequality [Ber24]). *Let $x_1, \cdots, x_n$ be independent zero-mean variables. Suppose $|x_i| \leq M$ almost surely, for all $i \in [n]$. Then for all positive t,*

$$\Pr\left[\sum_{i=1}^n x_i > t\right] \leq \exp\left(-\frac{t^2/2}{\sum_{i=1}^n \mathbb{E}[x_i^2] + Mt/3}\right).$$

Lemma A.1.3 (Hanson-Wright inequality [HW71, Wri73]). *Let $x = (x_1, \cdots, x_n) \in \mathbb{R}^n$ be a random vector with independent components x_i which satisfy $\mathbb{E}[x_i] = 0$ and $\|x_i\|_{\psi_2} \leq K$ (where $\|x_i\|_{\psi_2} := \sup_{p \geq 1} p^{-1/2}(\mathbb{E}[|x_i|^p])^{1/p}$). Let A be an $n \times n$ matrix. Then, for every $t \geq 0$,*

$$\Pr\left[\left|x^\top A x - \mathbb{E}[x^\top A x]\right| > t\right] \leq 2\exp\left(-c \cdot \min\left(\frac{t^2}{K^4\|A\|_F^2}, \frac{t}{K^2\|A\|}\right)\right).$$

Lemma A.1.4 (Khintchine's inequality [Haa81]). *Let $\sigma_1, \cdots, \sigma_n$ be i.i.d. sign random variables, and let $z_1, \cdots, z_n$ be real numbers. Then there are constants $C, C' > 0$ so that*

$$\Pr\left[\left|\sum_{i=1}^n z_i \sigma_i\right| \geq Ct\|z\|_2\right] \leq \exp(-C't^2).$$

Lemma A.1.5 (Hoeffding inequality [Hoe63])**.** *Let $x_i \in [a_i, b_i]$ denote independent random variables, let $x = \frac{1}{n}\sum_{i=1}^{n} x_i$ then*

$$\Pr[|x - \mathbb{E}[x]| \geq t] \leq 2\exp\left(-\frac{2n^2t^2}{\sum_{i=1}^{n}(b_i - a_i)^2}\right).$$

Lemma A.1.6 (Lemma 1 on page 1325 of [LM00])**.** *Let $x \sim \mathcal{X}_k^2$ be a chi-squared distributed random variable with k degrees of freedom. Each one has zero mean and σ^2 variance. Then*

$$\Pr[x - k\sigma^2 \geq (2\sqrt{kt} + 2t)\sigma^2] \leq \exp(-t)$$
$$\Pr[k\sigma^2 - x \geq 2\sqrt{kt}\sigma^2] \leq \exp(-t)$$

A.1.1 Reverse Chernoff bound

In this Section, we prove that the classical Chernoff bound is tight in some regimes. There are several different proofs, e.g. [Mou10, Slu77, AB09, YZD12]. For completeness, we provide a proof from [YZD12].

Fact A.1.7. *If $1 \leq l \leq k-1$, then*

$$\binom{k}{l} \geq \frac{1}{e\sqrt{2\pi l}}(\frac{k}{l})^l(\frac{k}{k-l})^{k-l}.$$

Proof. By Stirling's approximation, $i! = \sqrt{2\pi i}(i/e)^i e^\lambda$ for some $\lambda \in [1/(12i+1), 1/(12i)]$.

Thus,

$$\begin{aligned}\binom{k}{l} &= \frac{k!}{l!(k-l)!} \\ &\geq \frac{\sqrt{2\pi k}(\frac{k}{e})^k}{\sqrt{2\pi l}(\frac{l}{e})^l \sqrt{2\pi(k-l)}(\frac{k-l}{e})^{k-l}} \exp(\frac{1}{12k+1} - \frac{1}{12l} - \frac{1}{12(k-l)}) \\ &\geq \frac{\sqrt{2\pi k}(\frac{k}{e})^k}{\sqrt{2\pi l}(\frac{l}{e})^l \sqrt{2\pi(k-l)}(\frac{k-l}{e})^{k-l}} e^{-1} \\ &\geq \frac{1}{\sqrt{2\pi l}}(\frac{k}{l})^l (\frac{k}{k-l})^{k-l} e^{-1}.\end{aligned}$$

where the first step follows by definition, the second step follows by Stirling's approximation, the third step follows by $\frac{1}{1+a+b} + 1 \geq \frac{1}{a} + \frac{1}{b}$ for $a \geq 12, b \geq 12$.

□

Now, we are ready to prove the following result

Lemma A.1.8 ([Mou10, Slu77, AB09, YZD12]). *Let X be the average of k independent, Bernoulli random variables with mean p. For any $\epsilon \in (0, 1/2]$ and $p \in (0, 1/2]$, assuming $\epsilon^2 pk \geq 3$, we have*

$$\Pr[X \leq (1-\epsilon)p] \geq \exp(-9\epsilon^2 pk),$$

and

$$\Pr[X \geq (1+\epsilon)p] \geq \exp(-9\epsilon^2 pk).$$

Proof. Note that $\Pr[X \leq (1-\epsilon)p]$ equals the sum $\sum_{i=0}^{\lfloor (1-\epsilon)pk \rfloor} \Pr[X = i/k]$, and $\Pr[X = i/k] = \binom{k}{i} p^i (1-p)^{k-i}$.

Fix $l = \lfloor (1-2\epsilon)pk \rfloor + 1$. The terms in the sum are increasing, so the terms with index $i \geq \ell$ each have value at least $\Pr[X = l/k]$, so their sum has total value at least $(\epsilon pk - 2)\Pr[X = l/k]$. To complete the proof, we show that

$$(\epsilon pk - 2)\Pr[X = l/k] \geq \exp(-9\epsilon^2 pk).$$

The assumptions $\epsilon^2 pk \geq 3$ and $\epsilon \leq 1/2$ give $\epsilon pk \geq 6$, so we have

$$\begin{aligned}(\epsilon pk - 2)\Pr[X = l/k] &\geq \frac{2}{3}\epsilon pk \binom{k}{l} p^l (1-p)^{k-l} \\ &\geq \underbrace{\frac{2}{3}\epsilon pk \frac{1}{\sqrt{2\pi l}}}_{A} \cdot \underbrace{(\frac{k}{l})^l (\frac{k}{k-l})^{k-l} p^l (1-p)^{k-l}}_{B}\end{aligned}$$

Below, we will show that $A \geq \exp(-\epsilon^2 pk)$ and $B \geq \exp(-8\epsilon^2 pk)$.

Claim A.1.9. $A \geq \exp(-\epsilon^2 pk)$.

Proof. The assumptions $\epsilon^2 pk \geq 3$ and $\epsilon \leq 1/2$ imply that $pk \geq 12$.

By $l \leq pk + 1$ (from definition), and $pk \geq 12$, thus $l \leq 1.1pk$.

Therefore, we have

$$\begin{aligned}A &\geq \frac{2}{3e}\epsilon\sqrt{pk/(2.2\pi)} \\ &\geq \frac{2}{3e}\sqrt{3/(2.2\pi)} && \text{by } \epsilon\sqrt{pk} \geq \sqrt{3} \\ &\geq 0.1 \\ &\geq \exp(-3) \\ &\geq \exp(-\epsilon^2 pk).\end{aligned}$$

This completes the proof of the claim. □

Claim A.1.10. $B \geq \exp(-8\epsilon^2 pk)$.

Proof. Fix δ such that $l = (1-\delta)pk$. The choice of l implies $\delta \leq 2\epsilon$, so the claim will hold as long as $B \geq \exp(-2\delta^2 pk)$. Manipulting this latter inequality, we get

$$\begin{aligned} & B^{-1/l} \leq \exp(\frac{2\delta^2 pk}{l}) \\ \iff & \frac{l}{pk}(\frac{k-l}{(1-p)k})^{k/l-1} \leq \exp(\frac{2\delta^2 pk}{l}). \end{aligned}$$

Substituting $l = (1-\delta)pk$ and simplifying, it is equivalent to

$$(1-\delta)(1+\frac{\delta p}{1-p})^{\frac{1}{(1-\delta)p}-1} \leq \exp(\frac{2\delta^2}{1-\delta}).$$

Taking the logarithm of both sides, we have

$$\ln(1-\delta) + (\frac{1}{(1-\delta)p} - 1)(\ln(1+\frac{\delta p}{1-p})) \leq \frac{2\delta^2}{1-\delta}$$

Since $\ln(1+z) \leq z$, it suffices to prove

$$\begin{aligned} & -\delta + (\frac{1}{(1-\delta)p} - 1)(\frac{\delta p}{1-p}) \leq \frac{2\delta^2}{1-\delta} \\ \iff & \frac{\delta^2}{(1-p)(1-\delta)} \leq \frac{2\delta^2}{1-\delta}. \end{aligned}$$

Since $p \leq 1/2$, this finishes the proof of the claim. □

Combining the above two claims, we obtain the desired probability lower bound for the $(1-\epsilon)$ side. We can prove it for the $(1+\epsilon)$ side similarly. □

Lemma A.1.11 (Anti-concentration of Gaussian distribution)**.** *Let $X \sim N(0, \sigma^2)$, that is, the probability density function of X is given by $\phi(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}}$. Then*

$$\Pr[|X| \leq t] \in \left(\frac{2}{3}\frac{t}{\sigma}, \frac{4}{5}\frac{t}{\sigma}\right).$$

A.2 Matrix version probability tools

We state a version from [Tro11c], and there is also another version can be found in [OT09].

Lemma A.2.1 (Matrix Freedman). *Consider a matrix martingale* $\{Y_i : i = 0, 1, 2, \cdots\}$ *whose values are self-adjoint matrices with dimension* n*, and let* $\{X_i : i = 1, 2, 3, \cdots\}$ *be the difference sequence. Assume that the difference sequence is uniformly bounded in the sense that*

$$\lambda_{\max}(X_i) \leq R, \text{ almost surely } \text{ for } i = 1, 2, 3, \cdots.$$

Define the predictable quadratic variation process of the martingale :

$$W_i = \sum_{j=1}^{i} \mathbb{E}_{j-1}[X_j^2], \text{ for } i = 1, 2, 3, \cdots.$$

Then, for all $t \geq 0$ *and* $\sigma^2 > 0$*,*

$$\Pr\left[\exists i \geq 0 : \lambda_{\max}(Y_i) \geq t \text{ and } \|W_i\| \leq \sigma^2\right] \leq n \cdot \exp\left(-\frac{t^2/2}{\sigma^2 + Rt/3}\right).$$

Lemma A.2.2 (Matrix Bernstein, Theorem 6.1.1 in [Tro15]). *Consider a finite sequence* $\{X_1, \cdots, X_m\} \subset \mathbb{R}^{n_1 \times n_2}$ *of independent, random matrices with common dimension* $n_1 \times n_2$*. Assume that*

$$\mathbb{E}[X_i] = 0, \forall i \in [m] \quad \text{and} \quad \|X_i\| \leq M, \forall i \in [m].$$

Let $Z = \sum_{i=1}^{m} X_i$*. Let* $\mathrm{Var}[Z]$ *be the matrix variance statistic of sum:*

$$\mathrm{Var}[Z] = \max\left\{\left\|\sum_{i=1}^{m} \mathbb{E}[X_i X_i^\top]\right\|, \left\|\sum_{i=1}^{m} \mathbb{E}[X_i^\top X_i]\right\|\right\}.$$

Then

$$\mathbb{E}[\|Z\|] \leq (2\mathrm{Var}[Z] \cdot \log(n_1 + n_2))^{1/2} + M \cdot \log(n_1 + n_2)/3.$$

Furthermore, for all $t \geq 0$*,*

$$\Pr[\|Z\| \geq t] \leq (n_1 + n_2) \cdot \exp\left(-\frac{t^2/2}{\mathrm{Var}[Z] + Mt/3}\right).$$

A.3 Inequalities

We state the Hölder's inequality for complex numbers. We will use the corresponding version $p = q = 2$ of Cauchy-Schwarz inequality for complex numbers.

Lemma A.3.1 (Hölder's inequality)**.** *If* S *is a measurable subset of* $\mathbb{R}^n$ *with the Lebesgue measure, and* f *and* g *are measurable complex-valued functions on* S*, then*

$$\int_S |f(x)g(x)|\mathrm{d}x \leq \left(\int_S |f(x)|^p \mathrm{d}x\right)^{\frac{1}{p}} \left(\int_S |g(x)|^q \mathrm{d}x\right)^{\frac{1}{q}}$$

Fact A.3.2. *For any vector* $x \in \mathbb{R}^n$*, we have*

$$\|x\|_2 \leq \|x\|_1 \leq \sqrt{n}\|x\|_2.$$

For any matrix $A \in \mathbb{R}^{n \times d}$*, we have*

$$\|A\|_F \leq \|A\|_1 \leq \sqrt{nd}\|A\|_F,$$

and

$$\|A\| \leq \|A\|_F \leq \sqrt{d} \cdot \|A\|.$$

Appendix B

Coauthors Index

B.1 Coauthors Index

We provide a summary of all the coauthors related to this thesis. I would like to thank all of my coauthors x times, where x is the number of pages of this thesis. The purpose of writing my Ph.D. thesis is to thank and memorize helps from all the coauthors.

- Part I
  - Zeyuan Allen-Zhu
    - [AZLS18, AZLS19]
  - Michael B. Cohen
    - [CLS19]
  - Yin Tat Lee
    - [CLS19, LSZ19, LSV18]
  - Yuanzhi Li
    - [AZLS18, AZLS19]
  - Santosh S. Vempala
    - [LSV18]

  - Qiuyi Zhang
    - [LSZ19]

- Part II
  - Ankit Garg
    - [GLSS18]
  - Rasmus Kyng
    - [KS18, KLS19]
  - Yin Tat Lee
    - [GLSS18]
  - Kyle Luh
    - [KLS19]
  - Nikhil Srivastava
    - [GLSS18]

- Part III
  - Xue Chen
    - [CKPS16]
  - Danial Kane
    - [CKPS16]
  - Eric Price

    * [PS15, CKPS16]
  - Vasileios Nakos
    * [NS19, NSW18, NSW19a, NSW19b]
  - Zhengyu Wang
    * [NSW18, NSW19a, NSW19b]
- Part IV
  - Ilya Razenshteyn
    * [RSW16]
  - David P. Woodruff
    * [RSW16, SWZ16, SWZ17, SWZ18, SWZ19b, SWZ19a]
  - Huan Zhang
    * [SWZ16]
  - Peilin Zhong
    * [SWZ17, SWZ18, SWZ19b, SWZ19a]
- Part VI
  - Alexandr Andoni
    * [ASS+18]
  - Huaian Diao
    * [DJS+19]

- Rajesh Jayaram
  - [DJS+19]
- Jerry Li
  - [LSS19]
- Eric Price
  - [PSW17]
- Aviad Rubinstein
  - [RS19b, RSSS19]
- Saeed Seddighin
  - [RSSS19]
- Ruoqi Shen
  - [LSS19]
- Clifford Stein
  - [ASS+18]
- Wen Sun
  - [DJS+19]
- Xiaorui Sun
  - [RSSS19]
- Ruosong Wang
  - [SWY+19]

- Zhengyu Wang
  - [ASS+18]
- David P. Woodruff
  - [PSW17, DJS+19]
- Lin F. Yang
  - [SWY+19]
- Xin Yang
  - [SY19]
- Hongyang Zhang
  - [SWY+19]
- Peilin Zhong
  - [ASS+18, SWY+19]

Bibliography

[AA91] Charles A. Akemann and Joel Anderson. Lyapunov theorems for operator algebras. *Mem. Amer. Math. Soc.*, 94(458):iv+88, 1991. 593

[AAA+16] Dario Amodei, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Qiang Cheng, Guoliang Chen, et al. Deep speech 2: End-to-end speech recognition in English and Mandarin. In *International Conference on Machine Learning (ICML)*, pages 173–182, 2016. 20, 250, 351

[AAB+07] Evrim Acar, Canan Aykut-Bingöl, Haluk Bingol, Rasmus Bro, and Bülent Yener. Multiway analysis of epilepsy tensors. In *Proceedings 15th International Conference on Intelligent Systems for Molecular Biology (ISMB) & 6th European Conference on Computational Biology (ECCB), Vienna, Austria, July 21-25, 2007*, pages 10–18, 2007. 57, 1469

[AAMM18] Atiye Alaeddini, Siavash Alemzadeh, Afshin Mesbahi, and Mehran Mesbahi. Linear model regression on time-series data: Non-asymptotic error bounds and applications. *arXiv preprint arXiv:1807.06611*, 2018. 257

[AB09] Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009. 1273, 2506, 2507

[AB17] Amir Abboud and Arturs Backurs. Towards hardness of approximation for polynomial time problems. In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 67. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017. 2191, 2214, 2216, 2218

[ABB+17] Sepehr Assadi, MohammadHossein Bateni, Aaron Bernstein, Vahab S. Mirrokni, and Cliff Stein. Coresets meet EDCS: algorithms for matching and vertex cover on massive graphs. In *arXiv preprint*. http://arxiv.org/pdf/1711.03076, 2017. 2316

[ABEZ16] Amir Adler, David Boublil, Michael Elad, and Michael Zibulevsky. A deep learning approach to block-based compressed sensing of images. *arXiv preprint arXiv:1606.01519*, 2016. 988

[ABF+16] Jason Altschuler, Aditya Bhaskara, Gang Fu, Vahab Mirrokni, Afshin Rostamizadeh, and Morteza Zadimoghaddam. Greedy column subset selection: New bounds and distributed algorithms. In *International Conference on Machine Learning (ICML)*. https://arxiv.org/pdf/1605.08795, 2016. 1476

[ABGM14] Sanjeev Arora, Aditya Bhaskara, Rong Ge, and Tengyu Ma. Provable bounds for learning some deep representations. In *International Conference on Machine Learning (ICML)*, pages 584–592, 2014. 358

[ABIW09] Alexandr Andoni, Khanh Do Ba, Piotr Indyk, and David P Woodruff. Efficient sketches for earth-mover distance, with applications. In *50th Annual IEEE Symposium on Foundations of Computer Science*, pages 324–330, 2009. 1262

[ABSV14] Pranjal Awasthi, Avrim Blum, Or Sheffet, and Aravindan Vijayaraghavan. Learning mixtures of ranking models. In *Advances in Neural Information Processing Systems (NIPS)*. https://arxiv.org/pdf/1410.8750, 2014. 57, 1469

[ABW15] Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. Tight hardness results for LCS and other sequence similarity measures. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 59–78, 2015. 1914, 2191, 2214

[ABW17] Pranjal Awasthi, Maria-Florina Balcan, and Colin White. General and robust communication-efficient algorithms for distributed clustering. In *arXiv preprint*. https://arxiv.org/pdf/1703.00830, 2017. 1784, 1788

[AC06] Nir Ailon and Bernard Chazelle. Approximate nearest neighbors and the fast johnson-lindenstrauss transform. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing (STOC)*, pages 557–563. ACM, 2006. 1907, 1922, 1966

[ACCL07] Nir Ailon, Bernard Chazelle, Seshadhri Comandur, and Ding Liu. Estimating the distance to a monotone function. *Random Structures & Algorithms*, 31(3):371–383, 2007. 2218

[ACD+16] Ittai Abraham, Shiri Chechik, Daniel Delling, Andrew V Goldberg, and Renato F Werneck. On dynamic approximate shortest paths for planar graphs

with worst-case costs. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 740–753. Society for Industrial and Applied Mathematics, 2016. 2094

[ACGH18] Sanjeev Arora, Nadav Cohen, Noah Golowich, and Wei Hu. A convergence analysis of gradient descent for deep linear neural networks. *arXiv preprint arXiv:1810.02281*, 2018. 21, 251, 257, 351

[AÇKY05] Evrim Acar, Seyit A Çamtepe, Mukkai S Krishnamoorthy, and Bülent Yener. Modeling and multiway analysis of chatroom tensors. In *International Conference on Intelligence and Security Informatics*, pages 256–268. Springer, 2005. 57, 1469

[ACW17] Haim Avron, Kenneth L Clarkson, and David P Woodruff. Sharper bounds for regression and low-rank approximation with regularization. ., 2017. manuscript. 1335, 1338, 1342, 1343, 1344

[ACY06] Evrim Acar, Seyit A Camtepe, and Bülent Yener. Collective sampling and analysis of high order tensors for chatroom communications. In *International Conference on Intelligence and Security Informatics*, pages 213–224. Springer, 2006. 57, 1469

[ADGM16] Anima Anandkumar, Yuan Deng, Rong Ge, and Hossein Mobahi. Homotopy analysis for tensor pca. In *arXiv preprint*. https://arxiv.org/pdf/1610.09322, 2016. 58, 1470

[ADH+19] Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, Ruslan Salakhutdinov, and Ruosong Wang. On exact computation with an infinitely wide neural net. *arXiv preprint arXiv:1904.11955*, 2019. 1908

[ADHP09] Daniel Aloise, Amit Deshpande, Pierre Hansen, and Preyas Popat. Np-hardness of euclidean sum-of-squares clustering. *Machine learning*, 75(2):245–248, 2009. 2093, 2097

[ADK+16] Ittai Abraham, David Durfee, Ioannis Koutis, Sebastian Krinninger, and Richard Peng. On fully dynamic graph sparsifiers. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 335–344. IEEE, 2016. 2094

[ADL+12] Patrick Amestoy, Iain S. Duff, Jean-Yves L'Excellent, Yves Robert, François-Henry Rouet, and Bora Uçar. On computing inverse entries of a sparse matrix in an out-of-core environment. *SIAM J. Scientific Computing*, 34(4), 2012. 1925

[AFdLGTL09] Santiago Aja-Fernández, Rodrigo de Luis Garcia, Dacheng Tao, and Xuelong Li. *Tensors in image processing and computer vision*. Springer Science & Business Media, 2009. 57, 1469

[AFH+12] Anima Anandkumar, Dean P Foster, Daniel J Hsu, Sham M Kakade, and Yi-Kai Liu. A spectral algorithm for latent dirichlet allocation. In *Advances in Neural Information Processing Systems(NIPS)*, pages 917–925. https://arxiv.org/pdf/1204.6703, 2012. 57, 1469, 1803, 1814

[AFK+01] Yossi Azar, Amos Fiat, Anna R. Karlin, Frank McSherry, and Jared Saia. Spectral analysis of data. In *STOC*, 2001. 48, 1036

[AFKM01] Dimitris Achlioptas, Amos Fiat, Anna R. Karlin, and Frank McSherry. Web search via hub synthesis. In *FOCS*, 2001. 48, 1036

[AFLG15] Andris Ambainis, Yuval Filmus, and François Le Gall. Fast matrix multiplication: limitations of the coppersmith-winograd method. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing(STOC)*, pages 585–593. ACM, 2015. 12, 4

[AFS12] Andreas Argyriou, Rina Foygel, and Nathan Srebro. Sparse prediction with the k-support norm. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1457–1465, 2012. 1968, 1971, 1984

[AG09] Kook Jin Ahn and Sudipto Guha. Graph sparsification in the semi-streaming model. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 328–338. Springer, 2009. 26, 549

[AG14] Nima Anari and Shayan Oveis Gharan. The kadison-singer problem for strongly rayleigh measures and applications to asymmetric tsp. *arXiv preprint arXiv:1412.1143*, 2014. 596, 597, 607, 612, 613

[AG15] Nima Anari and Shayan Oveis Gharan. Effective-resistance-reducing flows, spectrally thin trees, and asymmetric tsp. In *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*, pages 20–39. IEEE, 2015. 554, 562, 563

[AG18] Kook Jin Ahn and Sudipto Guha. Access to data and number of iterations: Dual primal algorithms for maximum matching under resource constraints. *ACM Transactions on Parallel Computing (TOPC)*, 4(4):17, 2018. 2315, 2316

[AGH+14] Animashree Anandkumar, Rong Ge, Daniel J. Hsu, Sham M. Kakade, and Matus Telgarsky. Tensor decompositions for learning latent variable models. In *Journal of Machine Learning Research*, volume 15(1), pages 2773–2832. https://arxiv.org/pdf/1210.7559, 2014. 57, 58, 1469, 1800, 1803, 1810, 1814, 1850

[AGHK14] Animashree Anandkumar, Rong Ge, Daniel J Hsu, and Sham M Kakade. A tensor approach to learning mixed membership community models. In *Journal of Machine Learning Research*, volume 15(1), pages 2239–2312. https://arxiv.org/pdf/1302.2684, 2014. 57, 1469

[AGKM12] Sanjeev Arora, Rong Ge, Ravindran Kannan, and Ankur Moitra. Computing a nonnegative matrix factorization - provably. In *Proceedings of the 44th Symposium on Theory of Computing Conference (STOC), New York, NY, USA, May 19 - 22, 2012*, pages 145–162. https://arxiv.org/pdf/1111.0952, 2012. 1044, 1046, 1050, 1088, 1131, 1506, 1899

[AGM+10] Arash Asadpour, Michel X. Goemans, Aleksander Mądry, Shayan Oveis Gharan, and Amin Saberi. An $O(\log n/\log\log n)$-approximation algorithm for the asymmetric traveling salesman problem. In *Proceedings of the Twenty-first Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages

379–389, Philadelphia, PA, USA, 2010. Society for Industrial and Applied Mathematics. 550, 555

[AGM12a] Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. Analyzing graph structure via linear measurements. In *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms*, pages 459–467. Society for Industrial and Applied Mathematics, 2012. 2094

[AGM12b] Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. Graph sketches: sparsification, spanners, and subgraphs. In *Proceedings of the 31st ACM SIGMOD-SIGACT-SIGAI symposium on Principles of Database Systems*, pages 5–14. ACM, 2012. 2094

[AGM13] Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. Spectral sparsification in dynamic graph streams. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 1–10. Springer, 2013. 2094

[AGMR16] Sanjeev Arora, Rong Ge, Tengyu Ma, and Andrej Risteski. Provable learning of noisy-or networks. In *Proceedings of the 49th Annual Symposium on the Theory of Computing (STOC)*. ACM, https://arxiv.org/pdf/1612.08795, 2016. 57, 1469

[AGR16] Nima Anari, Shayan Oveis Gharan, and Alireza Rezaei. Monte carlo markov chain algorithms for sampling strongly rayleigh distributions and determinantal point processes. In *Conference on Learning Theory (COLT)*, pages 103–115, 2016. 555, 622

[AGS03] Adi Akavia, Shafi Goldwasser, and Shmuel Safra. Proving hard-core predicates using list decoding. In *FOCS*, volume 44, pages 146–159, 2003. 38, 919

[AH16] Zeyuan Allen-Zhu and Elad Hazan. Variance Reduction for Faster Non-Convex Optimization. In *Proceedings of the 33rd International Conference on Machine Learning*, ICML '16, 2016. Full version available at http://arxiv.org/abs/1603.05643. 15, 77

[AHL+18] Sanjeev Arora, Elad Hazan, Holden Lee, Karan Singh, Cyril Zhang, and Yi Zhang. Towards provable control for unknown linear dynamical systems. In *ICLR workshop*, 2018. 257

[AHPV04] Pankaj K Agarwal, Sariel Har-Peled, and Kasturi R Varadarajan. Approximating extent measures of points. *Journal of the ACM (JACM)*, 51(4):606–635, 2004. 2094

[AHWW16] Amir Abboud, Thomas Dueholm Hansen, Virginia Vassilevska Williams, and Ryan Williams. Simulating branching programs with edit distance and friends: or: a polylog shaved is a lower bound made. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 375–388, 2016. 1914, 2191, 2214

[AIK08] Alexandr Andoni, Piotr Indyk, and Robert Krauthgamer. Earth mover distance over high-dimensional spaces. In *Proceedings of the nineteenth annual*

ACM-SIAM symposium on Discrete algorithms, pages 343–352. Society for Industrial and Applied Mathematics, 2008. 1262

[AK17] Sepehr Assadi and Sanjeev Khanna. Randomized composable coresets for matching and vertex cover. In *SPAA*. https://arxiv.org/pdf/1705.08242, 2017. 2315, 2316

[AKDM10] E. Acar, T. G. Kolda, D. M. Dunlavy, and M. Morup. Scalable Tensor Factorizations for Incomplete Data. In *arXiv preprint*. https://arxiv.org/pdf/1005.2197, 2010. 58, 1469

[AKK+17] Naman Agarwal, Sham Kakade, Rahul Kidambi, Yin Tat Lee, Praneeth Netrapalli, and Aaron Sidford. Leverage score sampling for faster accelerated regression and erm. *arXiv preprint arXiv:1711.08426*, 2017. 13, 74

[AKM+19] Haim Avron, Michael Kapralov, Cameron Musco, Christopher Musco, Ameya Velingker, and Amir Zandieh. A universal sampling method for reconstructing signals with simple fourier transforms. In *STOC*, 2019. 920

[AKO10] Alexandr Andoni, Robert Krauthgamer, and Krzysztof Onak. Polylogarithmic approximation for edit distance and the asymmetric query complexity. In *Proceedings of the Fifty-First IEEE Annual Symposium on Foundations of Computer Science*, 2010. 1914, 2029, 2191, 2214, 2215

[AKO11] Alexandr Andoni, Robert Krauthgamer, and Krzysztof Onak. Streaming algorithms via precision sampling. In *Foundations of Computer Science*

(FOCS), 2011 IEEE 52nd Annual Symposium on, pages 363–372. IEEE, https://arxiv.org/pdf/1011.1263, 2011. 1568, 1593

[AKPS19] Deeksha Adil, Rasmus Kyng, Richard Peng, and Sushant Sachdeva. Iterative refinement for ℓ_p-norm regression. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1405–1424. SIAM, 2019. 14, 16, 74, 78

[AL13] Nir Ailon and Edo Liberty. An almost optimal unrestricted fast johnson-lindenstrauss transform. *ACM Transactions on Algorithms (TALG)*, 9(3):21, 2013. 1922

[AL17] Zeyuan Allen-Zhu and Yuanzhi Li. Follow the Compressed Leader: Faster Online Learning of Eigenvectors and Faster MMWU. In *ICML*, 2017. Full version available at http://arxiv.org/abs/1701.01722. lxxi, 265

[AL18] Zeyuan Allen-Zhu and Yuanzhi Li. Neon2: Finding Local Minima via First-Order Oracles. In *NeurIPS*, 2018. Full version available at http://arxiv.org/abs/1711.06673. lxxi, 265

[AL19a] Zeyuan Allen-Zhu and Yuanzhi Li. Can SGD Learn Recurrent Neural Networks with Provable Generalization? *ArXiv e-prints*, abs/1902.01028, February 2019. Full version available at http://arxiv.org/abs/1902.01028. 357, 366

[AL19b] Zeyuan Allen-Zhu and Yuanzhi Li. What Can ResNet Learn Efficiently, Going Beyond Kernels? *ArXiv e-prints*, abs/1905.10337, May 2019. 256

[ALA16] Kamyar Azizzadenesheli, Alessandro Lazaric, and Animashree Anandkumar. Reinforcement learning of POMDPs using spectral methods. In *29th Annual Conference on Learning Theory (COLT)*, pages 193–256. https://arxiv.org/pdf/1602.07764, 2016. 57, 1469

[ALB13] Mohammad Gheshlaghi Azar, Alessandro Lazaric, and Emma Brunskill. Sequential transfer in multi-armed bandit with finite set of models. In *Advances in Neural Information Processing Systems(NIPS)*, pages 2220–2228. https://arxiv.org/pdf/1307.6887, 2013. 57, 1469

[Ald90] David Aldous. The random walk construction of uniform spanning trees and uniform labelled trees. In *SIAM Journal on Discrete Mathematics*, pages 450–465, 1990. 29, 550

[All12a] Genevera Allen. Sparse higher-order principal components analysis. In *AISTATS*, volume 15, 2012. 58, 1470

[All12b] Genevera I Allen. Regularized tensor factorizations and higher-order principal components analysis. In *arXiv preprint*. https://arxiv.org/pdf/1202.2476, 2012. 58, 1470

[All17a] Zeyuan Allen-Zhu. Katyusha: The first direct acceleration of stochastic gradient methods. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1200–1205. ACM, 2017. 15, 77

[All17b] Zeyuan Allen-Zhu. Natasha: Faster Non-Convex Stochastic Optimization via Strongly Non-Convex Parameter. In *Proceedings of the 34th International*

Conference on Machine Learning, ICML '17, 2017. Full version available at http://arxiv.org/abs/1702.00763. 15, 77

[All18a] Zeyuan Allen-Zhu. Katyusha X: Practical Momentum Method for Stochastic Sum-of-Nonconvex Optimization. In *Proceedings of the 35th International Conference on Machine Learning*, ICML '18, 2018. Full version available at http://arxiv.org/abs/1802.03866. 15, 77

[All18b] Zeyuan Allen-Zhu. Natasha 2: Faster Non-Convex Optimization Than SGD. In *Proceedings of the 32nd Conference on Neural Information Processing Systems*, NIPS '18, 2018. Full version available at http://arxiv.org/abs/1708.08694. 15, 77

[ALL18c] Zeyuan Allen-Zhu, Yuanzhi Li, and Yingyu Liang. Learning and Generalization in Overparameterized Neural Networks, Going Beyond Two Layers. *arXiv preprint arXiv:1811.04918*, November 2018. 256, 270, 357, 366

[Alm19] Josh Alman. Limits on the universal method for matrix multiplication. In *CCC (Best Student Paper)*. arXiv preprint arXiv:1812.08731, 2019. 16, 78

[ALS+18] A. Andoni, C. Lin, Y. Sheng, P. Zhong, and R. Zhong. Subspace embedding and linear regression with Orlicz norm. In *ICML*, pages 224–233, 2018. 1967, 1968, 1970, 1971, 1972, 1976

[AM05] Dimitris Achlioptas and Frank McSherry. On spectral learning of mixtures of distributions. In *COLT*, 2005. 48, 1036

[AM07] Dimitris Achlioptas and Frank McSherry. Fast computation of low-rank matrix approximations. *J. ACM*, 54(2):9, 2007. 1338, 1468

[AMR+12] Marcel R Ackermann, Marcus Märtens, Christoph Raupach, Kamil Swierkot, Christiane Lammersen, and Christian Sohler. Streamkm++: A clustering algorithm for data streams. *Journal of Experimental Algorithmics (JEA)*, 17:2–4, 2012. 2094

[AMS11] Christoph Ambühl, Monaldo Mastrolilli, and Ola Svensson. Inapproximability results for maximum edge biclique, minimum linear arrangement, and sparsest cut. *SIAM Journal on Computing*, 40(2):567–596, 2011. 1081

[AMT10] Haim Avron, Petar Maymounkov, and Sivan Toledo. Blendenpik: Supercharging lapack's least-squares solver. *SIAM J. Scientific Computing*, 32(3):1217–1236, 2010. 1927

[ANN+17] A. Andoni, H. L. Nguyen, A. Nikolov, I. Razenshteyn, and E. Waingarten. Approximate near neighbors for general symmetric norms. In *STOC*, pages 902–913, 2017. 1967

[ANN+18] A. Andoni, A. Naor, A. Nikolov, I. Razenshteyn, and E. Waingarten. Hölder homeomorphisms and approximate nearest neighbors. In *FOCS*, pages 159–169, 2018. 1967

[ANOY14] Alexandr Andoni, Aleksandar Nikolov, Krzysztof Onak, and Grigory Yaroslavtsev. Parallel algorithms for geometric graph problems. In *STOC*. http://arxiv.org/pdf/1401.0042, 2014. 2315, 2316

[ANW14] Haim Avron, Huy Nguyen, and David Woodruff. Subspace embeddings for the polynomial kernel. In *Advances in Neural Information Processing Systems(NIPS)*, pages 2258–2266, 2014. 1482, 1486, 1518, 1519

[AO09] Alexandr Andoni and Krzysztof Onak. Approximating edit distance in near-linear time. In *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*, 2009. 1914, 2214, 2215

[AO12] Alexandr Andoni and Krzysztof Onak. Approximating edit distance in near-linear time. *SIAM Journal on Computing*, 41(6):1635–1648, 2012. 2191

[AOGSS18] Nima Anari, Shayan Oveis Gharan, Amin Saberi, and Nikhil Srivastava. Approximating the largest root and applications to interlacing families. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1015–1028. SIAM, Philadelphia, PA, 2018. 598

[APY09] Noga Alon, Rina Panigrahy, and Sergey Yekhanin. Deterministic approximation algorithms for the nearest codeword problem. In *Algebraic Methods in Computational Complexity*, 2009. lxxxi, 1431, 1457

[AR18] Amir Abboud and Aviad Rubinstein. Fast and deterministic constant factor approximation algorithms for lcs imply new circuit lower bounds. In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 94. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018. 2191, 2214, 2216, 2218

[AS16] Noga Alon and Joel H. Spencer. *The probabilistic method*. Wiley Series in

Discrete Mathematics and Optimization. John Wiley & Sons, Inc., Hoboken, NJ, fourth edition, 2016. 30, 590

[ASS+18] Alexandr Andoni, Zhao Song, Clifford Stein, Zhengyu Wang, and Peilin Zhong. Parallel graph connectivity in log diameter rounds. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 674–685. IEEE, 2018. xxviii, 1916, 2314, 2514, 2515, 2516

[ASS19] Josh Alman, Aaron Schild, and Zhao Song. Fast sparsification and linear algebra for geometric graphs. *Manuscript*, 2019. xxix

[ASSN08] Abiodun M Aibinu, Momoh JE Salami, Amir A Shafie, and Athaur Rahman Najeeb. MRI reconstruction using discrete Fourier transform: a tutorial. *World Academy of Science, Engineering and Technology*, 42:179, 2008. 36, 917

[ASW18] Sepehr Assadi, Xiaorui Sun, and Omri Weinstein. Massively parallel algorithms for finding well-connected components in sparse graphs. In *ArXiv preprint*. https://arxiv.org/pdf/1805.02974, 2018. 2320

[Aue30] H. Auerbach. *On the area of convex curves with conjugate diameters*. PhD thesis, University of Lwów, 1930. 2019

[AW02] Rudolf Ahlswede and Andreas Winter. Strong converse for identification via quantum channels. *IEEE Transactions on Information Theory*, 48(3):569–579, 2002. 25, 26, 30, 503, 506, 548, 553, 590, 2141

[AW14] Charles Akemann and Nik Weaver. A Lyapunov-type theorem from Kadison-Singer. *Bull. Lond. Math. Soc.*, 46(3):517–524, 2014. 593

[AW18a] Josh Alman and Virginia Vassilevska Williams. Further limitations of the known approaches for matrix multiplication. In *ITCS*. arXiv preprint arXiv:1712.07246, 2018. 16, 78

[AW18b] Josh Alman and Virginia Vassilevska Williams. Limits on all known (and some unknown) approaches to matrix multiplication. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 2018. 12, 16, 4, 78

[AWW14] Amir Abboud, Virginia Vassilevska Williams, and Oren Weimann. Consequences of faster alignment of sequences. In *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part I*, pages 39–51, 2014. 1914, 2191

[AY16] Zeyuan Allen-Zhu and Yang Yuan. Improved SVRG for Non-Strongly-Convex or Sum-of-Non-Convex Objectives. In *Proceedings of the 33rd International Conference on Machine Learning*, ICML '16, 2016. Full version available at http://arxiv.org/abs/1506.01972. 15, 77

[AZLS18] Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. On the convergence rate of training recurrent neural networks. In *arXiv preprint*. https://arxiv.org/pdf/1810.12065, 2018. xxv, 8, 24, 255, 272, 294, 311, 1908, 1909, 1910, 2140, 2143, 2512

[AZLS19] Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. A convergence theory for deep learning via over-parameterization. In *ICML*. https://arxiv.org/pdf/1811.03962, 2019. xxv, 7, 24, 352, 355, 357, 363, 366, 395, 495, 1908, 1909, 2140, 2143, 2512

[Ban38] Stefan Banach. Über homogene polynome in (l^2). *Studia Mathematica*, 7(1):36–44, 1938. 1676, 1677

[Bas08] Surender Baswana. Streaming algorithm for graph spanners-single pass and constant processing time per edge. *Information Processing Letters*, 106(3):110–114, 2008. 2094

[Bas14] Saugata Basu. Algorithms in real algebraic geometry: a survey. *arXiv preprint arXiv:1409.1534*, 2014. 1048, 1052, 1131

[Baw75] Vijay S Bawa. Optimal rules for ordering uncertain prospects. *Journal of Financial Economics*, 2(1):95–121, 1975. 654

[BB08] Léon Bottou and Olivier Bousquet. The tradeoffs of large scale learning. In *Advances in neural information processing systems*, pages 161–168, 2008. 13, 74, 598

[BBB+19a] Frank Ban, Vijay Bhattiprolu, Karl Bringmann, Pavel Kolev, Euiwoong Lee, and David P. Woodruff. A PTAS for ℓ_p-low rank approximation. In *SODA*, 2019. 56, 1371, 1428

[BBB+19b] Frank Ban, Vijay Bhattiprolu, Karl Bringmann, Pavel Kolev, Euiwoong Lee, and David P Woodruff. A ptas for ℓ_p-low rank approximation. In *Proceed-*

ings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, pages 747–766. SIAM, 2019. 2027

[BBC+17] Jaroslaw Blasiok, Vladimir Braverman, Stephen R Chestnut, Robert Krauthgamer, and Lin F Yang. Streaming symmetric norms via measure concentration. In *Proceedings of the 49th Annual Symposium on the Theory of Computing(STOC)*. ACM, https://arxiv.org/pdf/1511.01111, 2017. 1717, 1971, 1976, 1983, 1984, 2000, 2001, 2018

[BBD+16] Aritra Banik, Binay K. Bhattacharya, Sandip Das, Tsunehiko Kameda, and Zhao Song. The p-center problem in tree networks revisited. In *SWAT*, pages 6:1–6:15, 2016. xxix

[BBD+18] Sergey Bereg, Binay Bhattacharya, Sandip Das, Tsunehiko Kameda, Priya Ranjan Sinha Mahapatra, and Zhao Song. Optimizing squares covering a set of points. *Theoretical Computer Science*, 729:68–83, 2018. xxix

[BBK13] Pavle VM Blagojević, Boris Bukh, and Roman Karasev. Turán numbers for $k_{s,t}$-free graphs: Topological obstructions and algebraic constructions. *Israel Journal of Mathematics*, 197(1):199–214, 2013. 2284

[BBL09] Julius Borcea, Petter Brändén, and Thomas Liggett. Negative dependence and the geometry of polynomials. *Journal of the American Mathematical Society*, 22(2):521–567, 2009. 28, 549, 557, 558, 566, 568, 579, 588

[BBLM14] MohammadHossein Bateni, Aditya Bhaskara, Silvio Lattanzi, and Vahab Mirrokni. Distributed balanced clustering via mapping coresets. In *Advances*

in Neural Information Processing Systems (NIPS), pages 2591–2599, 2014. 1784, 1788

[BBM05] Peter L Bartlett, Olivier Bousquet, and Shahar Mendelson. Local rademacher complexities. *The Annals of Statistics*, 33(4):1497–1537, 2005. 13, 74

[BCDH10] Richard G Baraniuk, Volkan Cevher, Marco F Duarte, and Chinmay Hegde. Model-based compressive sensing. *IEEE Transactions on Information Theory*, 56(4):1982–2001, 2010. 635

[BCG+12] Petros Boufounos, Volkan Cevher, Anna C Gilbert, Yi Li, and Martin J Strauss. What's the frequency, Kenneth?: Sublinear Fourier sampling off the grid. In *Algorithmica(A preliminary version of this paper appeared in the Proceedings of RANDOM/APPROX 2012, LNCS 7408, pp.61–72)*, pages 1–28. Springer, 2012. 40, 41, 622, 676, 686, 690, 691, 763, 920, 1334

[BCI+16] Vladimir Braverman, Stephen R Chestnut, Nikita Ivkin, Jelani Nelson, Zhengyu Wang, and David P Woodruff. Bptree: an ℓ_2 heavy hitters algorithm using constant memory. In *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS)*. https://arxiv.org/pdf/1603.00759, 2016. 1779

[BCIW16] Vladimir Braverman, Stephen R Chestnut, Nikita Ivkin, and David P Woodruff. Beating countsketch for heavy hitters in insertion streams. In *Proceedings of the 48th Annual Symposium on the Theory of Computing (STOC)*. https://arxiv.org/pdf/1511.00661, 2016. 1779

[BCKY16] Vladimir Braverman, Stephen R Chestnut, Robert Krauthgamer, and Lin F Yang. Sketches for matrix norms: Faster, smaller and more general. In *arXiv preprint*. https://arxiv.org/pdf/1609.05885, 2016. 1717

[BCL05] Zheng-Jian Bai, Raymond H Chan, and Franklin T Luk. Principal component analysis for distributed data sets with updating. In *Advanced Parallel Processing Technologies*, pages 471–483. Springer, 2005. 1318, 1784, 1788

[BCLL18] Sébastien Bubeck, Michael B Cohen, Yin Tat Lee, and Yuanzhi Li. An homotopy method for ℓ_p regression provably beyond self-concordance and in input-sparsity time. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 1130–1137. ACM, 2018. 11, 14, 16, 3, 74, 78, 2033, 2046, 2051, 2067

[BCMN14] Sayan Bhattacharya, Parinya Chalermsook, Kurt Mehlhorn, and Adrian Neumann. New approximability results for the robust k-median problem. In *Scandinavian Workshop on Algorithm Theory*, pages 50–61. Springer, 2014. 2094

[BCMV14] Aditya Bhaskara, Moses Charikar, Ankur Moitra, and Aravindan Vijayaraghavan. Smoothed analysis of tensor decompositions. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, pages 594–603. ACM, https://arxiv.org/pdf/1311.3651, 2014. 58, 1470

[BCR87] Jacek Bochnak, Michel Coste, and Marie-Françoise Roy. *Géométrie algébrique réelle*, volume 12. Springer Science & Business Media, 1987. 1051

[BCS97] Peter Bürgisser, Michael Clausen, and Amin Shokrollahi. *Algebraic complexity theory*, volume 315. Springer Science & Business Media, 1997. 59, 1471

[BCS18] Mitali Bafna, Chi-Ning Chou, and Zhao Song. An exposition of dinur-khot-kindler-minzer-safra's proof for the 2-to-2 games conjecture. In *Notes from Boaz Barak's Feb 23rd and March 2nd 2018 talks at CMSA*, 2018. xxix

[BCV14] Aditya Bhaskara, Moses Charikar, and Aravindan Vijayaraghavan. Uniqueness of tensor decompositions with applications to polynomial identifiability. In *27th Annual Conference on Learning Theory (COLT)*, pages 742–778. https://arxiv.org/pdf/1304.8087, 2014. liii, 57, 1469, 1470, 1471, 1474, 1491

[BCWY16] V. Braverman, S. R. Chestnut, David P. Woodruff, and Lin F. Yang. Streaming space complexity of nearly all functions of one variable on frequency vectors. In *PODS*, pages 261–276, 2016. 1967

[BD08] Thomas Blumensath and Mike E Davies. Iterative thresholding for sparse approximations. *Journal of Fourier analysis and Applications*, 14(5-6):629–654, 2008. lxvii, lxviii, 37, 38, 918, 919, 923

[BD09a] Thomas Blumensath and Mike E Davies. Iterative hard thresholding for compressed sensing. *Applied and computational harmonic analysis*, 27(3):265–274, 2009. 34, 38, 617, 918, 925

[BD09b] Thomas Blumensath and Mike E Davies. A simple, efficient and near optimal algorithm for compressed sensing. *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2009. 38, 918

[BD10] Thomas Blumensath and Mike E Davies. Normalized iterative hard thresholding: Guaranteed stability and performance. *IEEE Journal of selected topics in signal processing*, 4(2):298–309, 2010. 38, 918

[BD13] J. Paul Brooks and José H. Dulá. The ℓ_1-norm best-fit hyperplane problem. *Appl. Math. Lett.*, 26(1):51–55, 2013. 52, 53, 1109, 1110, 1370

[BDB13] J. Paul Brooks, José H. Dulá, and Edward L Boone. A pure ℓ_1-norm principal component analysis. *Computational statistics & data analysis*, 61:83–98, 2013. l, lxxv, 53, 1110, 1286, 1289, 1290, 1298, 1299, 1334, 1370

[BDE+19] Soheil Behnezhad, Laxman Dhulipala, Hossein Esfandiari, Jakub Lacki, and Vahab Mirrokni. Near-optimal massively parallel graph connectivity. In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 2019. 1916

[BDG16] Nikhil Bansal, Daniel Dadush, and Shashwat Garg. An algorithm for komlós conjecture matching banaszczyk's bound. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 788–799. IEEE, 2016. 596

[BDGL18] Nikhil Bansal, Daniel Dadush, Shashwat Garg, and Shachar Lovett. The

gram-schmidt walk: A cure for the banaszczyk blues. In *STOC*. https://arxiv.org/pdf/1708.01079, 2018. 596

[BDK+14] Binay K. Bhattacharya, Minati De, Tsunehiko Kameda, Sasanka Roy, Vladyslav Sokol, and Zhao Song. Back-up 2-center on a path/tree/cycle/unicycle. In *COCOON*, pages 417–428, 2014. xxix

[BDL16] Amitabh Basu, Michael Dinitz, and Xin Li. Computing approximate PSD factorizations. In *arXiv preprint*. https://arxiv.org/pdf/1602.07351, 2016. 1046, 1131, 1506

[BDM11] Christos Boutsidis, Petros Drineas, and Malik Magdon-Ismail. Near optimal column-based matrix reconstruction. In *IEEE 52nd Annual Symposium on Foundations of Computer Science (FOCS), 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 305–314. https://arxiv.org/pdf/1103.0995, 2011. 1114, 1433, 1476, 1604

[BDMO03] Brain Babcock, Mayur Datar, Rajeev Motwani, and Liadan O'Callaghan. Maintaining variance and k-medians over data stream windows. In *Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 234–243. ACM, 2003. 2094

[BDN15] Jean Bourgain, Sjoerd Dirksen, and Jelani Nelson. Toward a unified theory of sparse dimensionality reduction in euclidean space. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 499–508, 2015. 52, 1109, 1370, 1426, 1922

[BEG+18] Mahdi Boroujeni, Soheil Ehsani, Mohammad Ghodsi, MohammadTaghi HajiAghayi, and Saeed Seddighin. Approximating edit distance in truly subquadratic time: Quantum and mapreduce. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, 2018. 62, 1914, 2191, 2212, 2214, 2215, 2220, 2231, 2286

[Ber24] Sergei Bernstein. On a modification of chebyshev's inequality and of the error formula of laplace. *Ann. Sci. Inst. Sav. Ukraine, Sect. Math*, 1(4):38–49, 1924. 25, 548, 2141, 2505

[Ber44] Joseph Berkson. Application of the logistic function to bio-assay. *Journal of the American Statistical Association*, 39(227):357–365, 1944. 16, 171

[BES06] Tuğkan Batu, Funda Ergun, and Cenk Sahinalp. Oblivious string embeddings and edit distance approximations. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithm*, 2006. 1914, 2191, 2214, 2215

[BFFN17] Jack Baker, Paul Fearnhead, Emily B Fox, and Christopher Nemeth. Control variates for stochastic gradient mcmc. *arXiv preprint arXiv:1706.05439*, 2017. 181

[BFL16] Vladimir Braverman, Dan Feldman, and Harry Lang. New frameworks for offline and streaming coreset constructions. *arXiv preprint arXiv:1612.00889*, 2016. 246, 2090, 2094, 2098, 2099, 2136, 2137

[BFL+17] Vladimir Braverman, Gereon Frahling, Harry Lang, Christian Sohler, and Lin F Yang. Clustering high dimensional dynamic data streams. In *ICML*.

https://arxiv.org/pdf/1706.03887, 2017. lxxvii, 2089, 2090, 2094, 2095, 2105, 2137, 2138

[BFN+17] Richard G Baraniuk, Simon Foucart, Deanna Needell, Yaniv Plan, and Mary Wootters. Exponential decay of reconstruction error from binary measurements of sparse signals. *IEEE Transactions on Information Theory*, 63(6):3368–3385, 2017. 636

[BFR16] Joris Bierkens, Paul Fearnhead, and Gareth Roberts. The zig-zag process and super-efficient sampling for bayesian analysis of big data. *arXiv preprint arXiv:1607.03188*, 2016. 181

[BG17] Nikhil Bansal and Shashwat Garg. Algorithmic discrepancy beyond partial coloring. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 914–926. ACM, 2017. 21, 250, 257, 351, 596, 2140

[BGMSS18] Alon Brutzkus, Amir Globerson, Eran Malach, and Shai Shalev-Shwartz. SGD learns over-parameterized networks that provably generalize on linearly separable data. In *ICLR*, 2018. 351, 358

[BGS98] Mihir Bellare, Oded Goldreich, and Madhu Sudan. Free bits, pcps, and nonapproximability—towards tight results. *SIAM Journal on Computing*, 27(3):804–915, 1998. 1269

[BGS15] Surender Baswana, Manoj Gupta, and Sandeep Sen. Fully dynamic maximal

matching in o($\log n$) update time. *SIAM Journal on Computing*, 44(1):88–113, 2015. 2094

[BH89] Paul Beame and Johan Håstad. Optimal bounds for decision problems on the CRCW PRAM. *J. ACM*, 36(3):643–670, 1989. 2315

[Bha97] Rajendra Bhatia. *Matrix analysis*, volume 169. Springer Science & Business Media, 1997. 517

[BHL18] Peter Bartlett, Dave Helmbold, and Phil Long. Gradient descent with identity initialization efficiently learns positive definite linear transformations. In *International Conference on Machine Learning*, pages 520–529, 2018. 21, 251, 257, 351, 1451

[BHNT15] Sayan Bhattacharya, Monika Henzinger, Danupon Nanongkai, and Charalampos Tsourakakis. Space-and time-efficient algorithm for maintaining dense subgraphs on one-pass dynamic streams. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 173–182. ACM, 2015. 2094

[BI08] Radu Berinde and Piotr Indyk. Sparse recovery using sparse random matrices. *preprint*, 2008. 625

[BI15] Arturs Backurs and Piotr Indyk. Edit Distance Cannot Be Computed in Strongly Subquadratic Time (unless SETH is false). In *Proc. of the 47th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 51–58, 2015. 1914, 2191, 2214, 2215

[Bin80] Dario Bini. Border rank of a p × q × 2 tensor and the optimal approximation of a pair of bilinear forms. *Automata, languages and programming*, pages 98–108, 1980. 59, 1471

[Bin86] Dario Bini. Border rank of m × n × (mn-q) tensors. *Linear Algebra and Its Applications*, 79:45–51, 1986. 59, 1471

[BIR08a] Radu Berinde, Piotr Indyk, and Milan Ruzic. Practical near-optimal sparse recovery in the ℓ_1 norm. In *Communication, Control, and Computing, 2008 46th Annual Allerton Conference on*, pages 198–205. IEEE, 2008. 34, 617

[BIR08b] Radu Berinde, Piotr Indyk, and Milan Ruzic. Practical near-optimal sparse recovery in the l1 norm. In *Communication, Control, and Computing, 2008 46th Annual Allerton Conference on*, pages 198–205. IEEE, 2008. 989

[BIRW16] Arturs Backurs, Piotr Indyk, Ilya Razenshteyn, and David P Woodruff. Nearly-optimal bounds for sparse recovery in generic norms, with applications to k-median sketching. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 318–337. SIAM, 2016. 620, 1262, 2094

[BJ12] J. Paul Brooks and Sapan Jot. Pcal1: An implementation in r of three methods for ℓ_1-norm principal component analysis. *Optimization Online preprint*, 2012. 53, 1110, 1370

[BJPD17] Ashish Bora, Ajil Jalal, Eric Price, and Alexandros G Dimakis. Compressed

sensing using generative models. In *ICML*. https://arxiv.org/pdf/1703.03208, 2017. 988

[BJW19] Ainesh Bakshi, Rajesh Jayaram, and David P Woodruff. Learning two layer rectified neural networks in polynomial time. In *COLT*. http://arxiv.org/pdf/:1811.01885, 2019. 2140

[BK96] András A. Benczúr and David R. Karger. Approximating s-t minimum cuts in $\widetilde{O}(n^2)$ time. In *Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing (STOC)*, pages 47–55, New York, NY, USA, 1996. ACM. 26, 29, 549, 556

[BK02] Piotr Berman and Marek Karpinski. Approximating minimum unsatisfiability of linear equations. In *Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms, January 6-8, 2002, San Francisco, CA, USA.*, pages 514–516, 2002. 1431

[BK15] Karl Bringmann and Marvin Kunnemann. Quadratic conditional lower bounds for string problems and dynamic time warping. In *Proc. of the 56th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 79–97, 2015. 1914, 2191

[BKLW14] Maria-Florina Balcan, Vandana Kanchanapally, Yingyu Liang, and David Woodruff. Improved distributed principal component analysis. In *Advances in Neural Information Processing Systems (NIPS)*. https://arxiv.org/pdf/1408.5823, 2014. 1318, 1784, 1788

[BKS12a] Surender Baswana, Sumeet Khurana, and Soumojit Sarkar. Fully dynamic randomized algorithms for graph spanners. *ACM Transactions on Algorithms (TALG)*, 8(4):35, 2012. 2094

[BKS12b] Binay Bhattacharya, Tsunehiko Kameda, and Zhao Song. Computing minmax regret 1-median on a tree network with positive/negative vertex weights. In *The 23rd International Symposium on Algorithms and Computation (ISAAC 2012), December 19-21, 2012 National Taiwan University, Taipei, Taiwan*, 2012. xxix

[BKS12c] Binay Bhattacharya, Tsunehiko Kameda, and Zhao Song. Minmax regret 1-center on a path/cycle/tree. In *The Sixth International Conference on Advanced Engineering Computing and Applications in Sciences(ADVCOMP 2012), September 23-28, 2012 - Barcelona, Spain*, 2012. xxix

[BKS13] Paul Beame, Paraschos Koutris, and Dan Suciu. Communication steps for parallel query processing. In *Proceedings of the 32nd ACM SIGMOD-SIGACT-SIGAI symposium on Principles of database systems*, pages 273–284. ACM, 2013. 2315, 2316, 2318, 2411

[BKS14] Binay Bhattacharya, Tsunehiko Kameda, and Zhao Song. Improved minmax regret 1-center algorithms for cactus networks with c cycles. In *LATIN*, pages 330–341, 2014. xxix

[BKS15a] Boaz Barak, Jonathan A Kelner, and David Steurer. Dictionary learning and tensor decomposition via the sum-of-squares method. In *Proceedings*

of the Forty-Seventh Annual ACM on Symposium on Theory of Computing (STOC), pages 143–151. ACM, https://arxiv.org/pdf/1407.1543, 2015. 58, 1470

[BKS15b] Binay Bhattacharya, Tsunehiko Kameda, and Zhao Song. Minmax regret 1-center algorithms for path/tree/unicycle/cactus networks. In *Discrete Applied Mathematics*, pages 195: 18–30, 2015. xxix

[BKW17] Karl Bringmann, Pavel Kolev, and David P. Woodruff. Approximation algorithms for ℓ_0-low rank approximation. In *Advances in Neural Information Processing Systems (NIPS)*, pages 6651–6662, 2017. 56, 1428, 1431, 1436

[BLG+15] Aurélien Bellet, Yingyu Liang, Alireza Bagheri Garakani, Maria-Florina Balcan, and Fei Sha. A distributed frank-wolfe algorithm for communication-efficient sparse learning. In *Proceedings of the 2015 SIAM International Conference on Data Mining (ICDM)*, pages 478–486. SIAM, https://arxiv.org/pdf/1404.2644, 2015. 1784, 1788

[BLLM19] Jaroslaw Blasiok, Patrick Lopatto, Kyle Luh, and Jake Marcinek. Sparse reconstruction from hadamard matrices: A lower bound. In *arXiv preprint*. https://arxiv.org/pdf/1903.12135.pdf, 2019. 925

[BLO05] James V Burke, Adrian S Lewis, and Michael L Overton. A robust gradient sampling algorithm for nonsmooth, nonconvex optimization. *SIAM Journal on Optimization*, 15(3):751–779, 2005. 20, 250

[BLS+16a] Maria-Florina Balcan, Yingyu Liang, Le Song, David Woodruff, and Bo Xie. Communication efficient distributed kernel principal component analysis. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 725–734. ACM, https://arxiv.org/pdf/1503.06858, 2016. 1784, 1788

[BLS+16b] Maria-Florina Balcan, Yingyu Liang, Le Song, David Woodruff, and Bo Xie. Distributed kernel principal component analysis. *KDD*, 2016. 1318

[BLS+19] Maria-Florina Balcan, Yingyu Liang, Zhao Song, David P. Woodruff, and Hongyang Zhang. Non-convex matrix completion and related problems via strong duality. *JMLR*, 2019. xxix

[BM86] Y. Bresler and A. Macovski. Exact maximum likelihood parameter estimation of superimposed exponential signals in noise. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 34(5):1081–1089, Oct 1986. 762

[BM16] Boaz Barak and Ankur Moitra. Noisy tensor completion via the sum-of-squares hierarchy. In *Proceedings of the 29th Conference on Learning Theory, COLT 2016, New York, USA, June 23-26, 2016*, pages 417–445. https://arxiv.org/pdf/1501.06521, 2016. 58, 1470

[BMD09] Christos Boutsidis, Michael W Mahoney, and Petros Drineas. An improved approximation algorithm for the column subset selection problem. In *Proceedings of the twentieth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 968–977. Society for Industrial and Applied Mathematics, https://arxiv.org/pdf/0812.4293, 2009. 1114, 1433, 1476, 1604

[BNJ01] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. In *NIPS*, 2001. 49, 1036, 1814, 1887

[BNR+15] Guillaume Bouchard, Jason Naradowsky, Sebastian Riedel, Tim Rocktäschel, and Andreas Vlachos. Matrix and tensor factorization methods for natural language processing. In *ACL (Tutorial Abstracts)*, pages 16–18, 2015. 57, 1469

[BO10] V. Braverman and R. Ostrovsky. Zero-one frequency laws. In *STOC*, pages 281–290, 2010. 1967

[Bou10] Petros T Boufounos. Reconstruction of sparse signals from distorted randomized measurements. In *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*, pages 3998–4001. IEEE, 2010. 596

[Bou11] Christos Boutsidis. Topics in matrix sampling algorithms. In *Ph.D. Thesis. arXiv preprint*. https://arxiv.org/pdf/1105.0709, 2011. 1501

[Bou14] Jean Bourgain. An improved estimate in the restricted isometry problem. In *Geometric Aspects of Functional Analysis*, pages 65–70. Springer, 2014. lxvii, lxviii, 16, 37, 38, 171, 623, 762, 918, 919, 923, 925, 1922

[BP12] Karl Bringmann and Konstantinos Panagiotou. Efficient sampling methods for discrete distributions. In *International Colloquium on Automata, Languages, and Programming*, pages 133–144. Springer, 2012. 1807

[BPD18] Ashish Bora, Eric Price, and Alexandros G. Dimakis. AmbientGAN: Generative models from lossy measurements. In *International Conference on*

Learning Representations. https://openreview.net/forum?id=Hy7fDog0b, 2018. 988

[BPR96] Saugata Basu, Richard Pollack, and Marie-Françoise Roy. On the combinatorial and algebraic complexity of quantifier elimination. *J. ACM*, 43(6):1002–1045, 1996. 1046, 1052, 1053, 1131, 1339, 1483, 1506

[BPR05] Saugata Basu, Richard Pollack, and Marie-Francoise Roy. *Algorithms in real algebraic geometry*, volume 20033. Springer, 2005. 1051, 1052, 1065, 1066, 1131

[BR65] George R Blakley and Prabir Roy. A hölder type inequality for symmetric matrices with nonnegative entries. *Proceedings of the American Mathematical Society*, 16(6):1244–1245, 1965. 2284

[BR93] Avrim L Blum and Ronald L Rivest. Training a 3-node neural network is np-complete. In *Machine learning: From theory to applications (A preliminary version of this paper was appeared in NIPS 1989)*, pages 9–28. Springer, 1993. 258

[BR94] Mihir Bellare and John Rompel. Randomness-efficient oblivious sampling. In *Foundations of Computer Science, 1994 Proceedings., 35th Annual Symposium on*, pages 276–287. IEEE, 1994. 2119

[BR19] Joshua Brakensiek and Aviad Rubinstein. Constant-factor approximation of near-linear edit distance in near-linear time. Manuscript, 2019. 2193

[Brä18] Petter Brändén. Hyperbolic polynomials and the kadison-singer problem. *arXiv preprint arXiv:1809.03255*, 2018. 1912, 1913

[BRB08] Yann-Ael Le Borgne, Sylvain Raybaud, and Gianluca Bontempi. Distributed principal component analysis for wireless sensor networks. *Sensors*, 2008. 1318, 1784, 1788

[Bro89] Andrei Broder. Generating random spanning trees. In *Proceedings of the 30th annual Symposium on Foundations of Computer Science (FOCS)*, pages 442–447, 1989. 29, 550

[BS80a] Jon Louis Bentley and James B Saxe. Decomposable searching problems i. static-to-dynamic transformation. *Journal of Algorithms*, 1(4):301–358, 1980. 2094

[BS80b] Jon Louis Bentley and James B Saxe. Generating sorted lists of random numbers. *ACM Transactions on Mathematical Software (TOMS)*, 6(3):359–364, 1980. 1807, 1818, 1819, 1820

[BS12] Markus Blaser and Chandan Saha. Lecture 6: Multipoint evaluation of a polynomial. *Max-Planck-Institut für Informatik Class Notes, Computational Number Theory and Algebra*, pages 1–4, 2012. 882

[BS15] Srinadh Bhojanapalli and Sujay Sanghavi. A new sampling technique for tensors. In *arXiv preprint*. https://arxiv.org/pdf/1502.05023, 2015. 57, 1469, 1797, 1798, 1804

[BS16] Aaron Bernstein and Cliff Stein. Faster fully dynamic matchings with small approximation ratios. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 692–711. Society for Industrial and Applied Mathematics, 2016. 2094

[BSS12] Joshua Batson, Daniel A Spielman, and Nikhil Srivastava. Twice-ramanujan sparsifiers. In *SIAM Journal on Computing*, volume 41(6), pages 1704–1721. https://arxiv.org/pdf/0808.0163, 2012. 29, 551, 596, 607, 1560, 1913, 2031

[BSST13] Joshua Batson, Daniel A Spielman, Nikhil Srivastava, and Shang-Hua Teng. Spectral sparsification of graphs: theory and algorithms. *Communications of the ACM*, 56(8):87–94, 2013. 5, 552, 1911

[BW14] Christos Boutsidis and David P Woodruff. Optimal cur matrix decompositions. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing (STOC)*, pages 353–362. ACM, https://arxiv.org/pdf/1405.7910, 2014. 1114, 1115, 1433, 1476, 1477, 1484, 1489, 1520, 1550, 1552, 1561, 1604, 2027

[BW18] Ainesh Bakshi and David Woodruff. Sublinear time low-rank approximation of distance matrices. In *Advances in Neural Information Processing Systems*, pages 3782–3792, 2018. 2027

[BWZ16] Christos Boutsidis, David P Woodruff, and Peilin Zhong. Optimal principal component analysis in distributed and streaming models. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing*

(STOC), pages 236–249. ACM, https://arxiv.org/pdf/1504.06729, 2016. 1310, 1318, 1319, 1338, 1511, 1779, 1784, 1788, 2027, 2094

[BYJK+02] Ziv Bar-Yossef, TS Jayram, Ravi Kumar, D Sivakumar, and Luca Trevisan. Counting distinct elements in a data stream. In *International Workshop on Randomization and Approximation Techniques in Computer Science*, pages 1–10. Springer, 2002. 2094

[BYJKK04] Ziv Bar-Yossef, TS Jayram, Robert Krauthgamer, and Ravi Kumar. Approximating edit distance efficiently. In *Proceedings of the Forty-Fifth Annual IEEE Symposium on Foundations of Computer Science*, 2004. 1914, 2191, 2214, 2215

[BZI17] Sina Bittens, Ruochuan Zhang, and Mark A Iwen. A deterministic sparse FFT for functions with structured Fourier sparsity. *Advances in Computational Mathematics, to appear.*, 2017. 38, 919

[CAKM16] Vincent Cohen-Addad, Philip N Klein, and Claire Mathieu. Local search yields approximation schemes for k-means and k-median in euclidean and minor-free metrics. In *Foundations of Computer Science (FOCS), 2016 IEEE 57th Annual Symposium on*, pages 353–364. IEEE, 2016. 2094

[Cau42] Augustin Louis Cauchy. *Mémoire sur l'emploi du calcul des limites dans l'intégration des équations aux dérivées partielles.* ., 1842. 233

[CBJC14] Sheng Cai, Mayank Bakshi, Sidharth Jaggi, and Minghua Chen. Super:

Sparse signals with unknown phases efficiently recovered. *arXiv preprint arXiv:1401.4269*, 2014. 635

[CC70] J Douglas Carroll and Jih-Jie Chang. Anaylsis of individual differences in multidimensional scaling via an n-way generalization of eckart-young decomposition. *Psychometrika*, 35(3):283–319, 1970. 57, 1469

[CC10] Cesar F Caiafa and Andrzej Cichocki. Generalizing the column–row matrix decomposition to multi-way arrays. *Linear Algebra and its Applications*, 433(3):557–573, 2010. 58, 1470, 1476

[CCAY+18] Xiang Cheng, Niladri S Chatterji, Yasin Abbasi-Yadkori, Peter L Bartlett, and Michael I Jordan. Sharp convergence rates for langevin dynamics in the nonconvex setting. *arXiv preprint arXiv:1805.01648*, 2018. 17, 172

[CCBJ18] Xiang Cheng, Niladri S Chatterji, Peter L Bartlett, and Michael I Jordan. Underdamped langevin mcmc: A non-asymptotic analysis. In *COLT*. arXiv preprint arXiv:1707.03663, 2018. 17, 18, 172, 174, 176

[CCF02] Moses Charikar, Kevin Chen, and Martin Farach-Colton. Finding frequent items in data streams. In *Automata, Languages and Programming*, pages 693–703. Springer, 2002. 35, 619, 624, 635, 688, 924, 996, 1009

[CCFC04] Moses Charikar, Kevin Chen, and Martin Farach-Colton. Finding frequent items in data streams. *Theoretical Computer Science*, 312(1):3–15, 2004. 2052, 2053

[CCGG98] Moses Charikar, Chandra Chekuri, Ashish Goel, and Sudipto Guha. Rounding via trees: deterministic approximation algorithms for group steiner trees and k-median. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 114–123. ACM, 1998. 2094

[CD05] Zizhong Chen and Jack J. Dongarra. Condition numbers of gaussian random matrices. *SIAM Journal on Matrix Analysis and Applications*, 27(3):603–620, 2005. 1045, 1056

[CDD09] Albert Cohen, Wolfgang Dahmen, and Ronald DeVore. Compressed sensing and best k-term approximation. *Journal of the American mathematical society*, 22(1):211–231, 2009. lxvii, lxviii, 37, 619, 923

[CDG+18] Diptarka Charkraborty, Debarati Das, Elazar Goldenberg, Michal Koucky, and Michael Saks. Approximating edit distance within constant factor in truly sub-quadratic time. In *FOCS*. https://iuuk.mff.cuni.cz/~diptarka/publications/approxEdit.pdf, 2018. 63, 1915, 2191, 2192, 2212, 2214, 2215, 2220, 2221, 2223, 2227, 2231, 2295, 2297, 2298

[CDHB09] Volkan Cevher, Marco F Duarte, Chinmay Hegde, and Richard Baraniuk. Sparse signal recovery using markov random fields. In *Advances in Neural Information Processing Systems*, pages 257–264, 2009. 988

[CDL13] Albert Cohen, Mark A Davenport, and Dany Leviatan. On the stability and accuracy of least squares approximations. *Foundations of computational mathematics*, 13(5):819–834, 2013. 761, 764

[CDMI+13] Kenneth L Clarkson, Petros Drineas, Malik Magdon-Ismail, Michael W Mahoney, Xiangrui Meng, and David P Woodruff. The fast cauchy transform and faster robust linear regression. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 466–477. Society for Industrial and Applied Mathematics, https://arxiv.org/pdf/1207.4684, 2013. 52, 1109, 1117, 1170, 1186, 1203, 1336, 1717, 2030

[CEM+15] Michael B Cohen, Sam Elder, Cameron Musco, Christopher Musco, and Madalina Persu. Dimensionality reduction for k-means clustering and low rank approximation. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing (STOC)*, pages 163–172. ACM, https://arxiv.org/pdf/1410.6801, 2015. 48, 1036, 1561, 2080

[CEPR07] Raphaël Clifford, Klim Efremenko, Ely Porat, and Amir Rothschild. k-mismatch with don't cares. *Algorithms–ESA 2007*, pages 151–162, 2007. 33, 617

[CF14] Emmanuel J Candès and Carlos Fernandez-Granda. Towards a mathematical theory of super-resolution. *Communications on Pure and Applied Mathematics*, 67(6):906–956, 2014. 41, 676, 686, 763

[CFHW17] Marek Cygan, Fedor V Fomin, Danny Hermelin, and Magnus Wahlström. Randomization in parameterized complexity (dagstuhl seminar 17041). In *Dagstuhl Reports*, volume 7:1. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017. 1898

[CFM+18] Niladri S Chatterji, Nicolas Flammarion, Yi-An Ma, Peter L Bartlett, and Michael I Jordan. On the theory of variance reduction for stochastic gradient monte carlo. In *ICML*, 2018. 17, 172, 181

[CG19] Yuan Cao and Quanquan Gu. A Generalization Theory of Gradient Descent for Learning Over-parameterized Deep ReLU Networks. *arXiv preprint arXiv:1902.01384*, 2019. 357, 366

[CGCB14] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014. 22, 352

[CGK+17a] Flavio Chierichetti, Sreenivas Gollapudi, Ravi Kumar, Silvio Lattanzi, Rina Panigrahy, and David P Woodruff. Algorithms for ℓ_p low rank approximation. In *ICML*. arXiv preprint arXiv:1705.06730, 2017. 56, 1371, 1374, 1375, 1377, 1428, 1433, 1434, 1435

[CGK+17b] Flavio Chierichetti, Sreenivas Gollapudi, Ravi Kumar, Silvio Lattanzi, Rina Panigrahy, and David P Woodruff. Algorithms for ℓ_p low rank approximation. In *ICML*. https://arxiv.org/pdf/1705.06730, 2017. 1334

[CGK+17c] Flavio Chierichetti, Sreenivas Gollapudi, Ravi Kumar, Silvio Lattanzi, Rina Panigrahy, and David P Woodruff. Algorithms for ℓ_p low rank approximation. In *ICML*. arXiv preprint arXiv:1705.06730, 2017. 2027, 2082

[CGKK18] Moses Charikar, Ofir Geri, Michael P. Kim, and William Kuszmaul. On estimating edit distance: Alignment, dimension reduction, and embeddings. In

45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic, pages 34:1–34:14, 2018. 2192

[CGL+19] Lijie Chen, Shafi Goldwasser, Kaifeng Lyu, Guy N Rothblum, and Aviad Rubinstein. Fine-grained complexity meets IP=PSPACE. In *SODA*. arXiv preprint arXiv:1805.02351, 2019. 2191, 2214, 2216, 2218

[CGV13] Mahdi Cheraghchi, Venkatesan Guruswami, and Ameya Velingker. Restricted isometry of Fourier matrices and list decodability of random linear codes. *SIAM Journal on Computing*, 42(5):1888–1914, 2013. lxvii, lxviii, 37, 923, 925

[CH09] Graham Cormode and Marios Hadjieleftheriou. Finding the frequent items in streams of data. *Communications of the ACM*, 52(10):97–105, 2009. 33, 617, 631

[Cha00] Bernard Chazelle. *The discrepancy method*. Cambridge University Press, Cambridge, 2000. Randomness and complexity. 30, 590

[Cha12] Sourav Chatterjee. Matrix estimation by universal singular value thresholding. *pre-print*, 2012. http://arxiv.org/abs/1212.1247. 48, 1036, 1043

[Che52] Herman Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *The Annals of Mathematical Statistics*, pages 493–507, 1952. 25, 30, 502, 548, 590, 2141, 2505

[Che09] Ke Chen. On coresets for k-median and k-means clustering in metric and euclidean spaces and their applications. *SIAM Journal on Computing*, 39(3):923–947, 2009. 2089, 2094

[CHW12] Kenneth L. Clarkson, Elad Hazan, and David P. Woodruff. Sublinear optimization for machine learning. *J. ACM*, 59(5):23, 2012. 1799

[CI17] Mahdi Cheraghchi and Piotr Indyk. Nearly optimal deterministic algorithm for sparse walsh-hadamard transform. *ACM Transactions on Algorithms (TALG)*, 13(3):34, 2017. 38, 919, 1904

[CIHB09] Volkan Cevher, Piotr Indyk, Chinmay Hegde, and Richard G Baraniuk. Recovery of clustered sparse signals from compressive measurements. Technical report, RICE UNIV HOUSTON TX DEPT OF ELECTRICAL AND COMPUTER ENGINEERING, 2009. 635

[Cip00] Barry A Cipra. The best of the 20th century: Editors name top 10 algorithms. *SIAM news*, 33(4):1–2, 2000. 1

[CJK+04] Graham Cormode, Theodore Johnson, Flip Korn, Shan Muthukrishnan, Oliver Spatscheck, and Divesh Srivastava. Holistic udafs at streaming speeds. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 35–46. ACM, 2004. 33, 617

[CK11] Imre Csiszar and János Körner. *Information theory: coding theorems for discrete memoryless systems*. Cambridge University Press, 2011. 1219

[CKK+18] Michael B Cohen, Jonathan Kelner, Rasmus Kyng, John Peebles, Richard Peng, Anup B Rao, and Aaron Sidford. Solving directed laplacian systems in nearly-linear time through sparse LU factorizations. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 898–909. IEEE, 2018. 6

[CKM+11] Paul Christiano, Jonathan A Kelner, Aleksander Madry, Daniel A Spielman, and Shang-Hua Teng. Electrical flows, laplacian systems, and faster approximation of maximum flow in undirected graphs. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 273–282. ACM, 2011. 6

[CKM+14] Michael B Cohen, Rasmus Kyng, Gary L Miller, Jakub W Pachocki, Richard Peng, Anup B Rao, and Shen Chen Xu. Solving sdd linear systems in nearly m log 1/2 n time. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pages 343–352. ACM, 2014. 6

[CKP+17] Michael B Cohen, Jonathan Kelner, John Peebles, Richard Peng, Anup B Rao, Aaron Sidford, and Adrian Vladu. Almost-linear-time algorithms for markov chains and new spectral primitives for directed graphs. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 410–419. ACM, 2017. 25, 548

[CKPS16] Xue Chen, Daniel M Kane, Eric Price, and Zhao Song. Fourier-sparse interpolation without a frequency gap. In *Foundations of Computer Science (FOCS), 2016 IEEE 57th Annual Symposium on*, pages 741–750. IEEE, 2016.

xxvi, 10, 40, 47, 164, 622, 920, 965, 977, 996, 1016, 1474, 1901, 1905, 2308, 2309, 2310, 2312, 2513, 2514

[CKSZ17] Volkan Cevher, Michael Kapralov, Jonathan Scarlett, and Amir Zandieh. An adaptive sublinear-time block sparse Fourier transform. In *Proceedings of the 49th Annual Symposium on the Theory of Computing (STOC)*. ACM, https://arxiv.org/pdf/1702.01286, 2017. 623, 636, 637

[CKY97] Sunglok Choi, Taemin Kim, and Wonpil Yu. Performance evaluation of ransac family. *Journal of Computer Vision*, 24(3):271–300, 1997. 990

[CL06] Fan Chung and Linyuan Lu. Concentration inequalities and martingale inequalities: a survey. *Internet Mathematics*, 3(1):79–127, 2006. 509

[Cla95] Kenneth L Clarkson. Las vegas algorithms for linear and integer programming when the dimension is small. *Journal of the ACM (JACM)*, 42(2):488–499, 1995. 13, 5

[Cla05] Kenneth L Clarkson. Subgradient and sampling algorithms for ℓ_1 regression. In *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms (SODA)*, pages 257–266, 2005. 14, 52, 74, 1109, 1117, 1124, 1134, 1516, 2032

[CLK+15] Fengyu Cong, Qiu-Hua Lin, Li-Dan Kuang, Xiao-Feng Gong, Piia Astikainen, and Tapani Ristaniemi. Tensor decomposition of eeg signals: a brief review. *Journal of neuroscience methods*, 248:59–69, 2015. 57, 1469

[CLLM12] Kai-min Chung, Henry Lam, Zhenming Liu, and Michael Mitzenmacher. Chernoff-Hoeffding bounds for Markov chains: Generalized and simplified. *arXiv preprint arXiv:1201.0559*, 2012. 26, 502, 537

[CLM+15] Michael B Cohen, Yin Tat Lee, Cameron Musco, Christopher Musco, Richard Peng, and Aaron Sidford. Uniform sampling for matrix approximation. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science (ITCS)*, pages 181–190. ACM, https://arxiv.org/pdf/1408.5099, 2015. 1133, 1515

[CLM+16] Michael B Cohen, Yin Tat Lee, Gary Miller, Jakub Pachocki, and Aaron Sidford. Geometric median in nearly linear time. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing (STOC)*, pages 9–21. ACM, 2016. 11, 3

[CLM+18] Artur Czumaj, Jakub Lacki, Aleksander Madry, Slobodan Mitrovic, Krzysztof Onak, and Piotr Sankowski. Round compression for parallel matching algorithms. In *STOC*. https://arxiv.org/pdf/1707.03478, 2018. 2315, 2316

[CLMW11] Emmanuel J Candès, Xiaodong Li, Yi Ma, and John Wright. Robust principal component analysis? *Journal of the ACM (JACM)*, 58(3):11, 2011. l, 56, 1300, 1334, 1370, 1428

[CLRS09] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT press, 2009. 2270

[CLS19] Michael B Cohen, Yin Tat Lee, and Zhao Song. Solving linear programs in the current matrix multiplication time. In *Proceedings of the 51st Annual ACM Symposium on Theory of Computing (STOC)*. https://arxiv.org/pdf/1810.07896.pdf, 2019. xxiv, lxvi, lxviii, 3, 4, 13, 15, 24, 35, 77, 79, 87, 88, 89, 119, 123, 126, 127, 131, 133, 135, 137, 138, 139, 144, 624, 1895, 1896, 2046, 2050, 2067, 2512

[CLS20] Sitan Chen, Jerry Li, and Zhao Song. Tbd. *Manuscript*, 2020. xxix

[CLZ17] Longxi Chen, Yipeng Liu, and Ce Zhu. Iterative block tensor singular value thresholding for extraction of low rank component of image data. In *ICASSP 2017*, 2017. 57, 58, 1469, 1470

[CM04] Graham Cormode and Shan Muthukrishnan. An improved data stream summary: the count-min sketch and its applications. In *Proceedings of the Annual Conference on Foundations of Software Technology and Theoretical Computer Science*, 2004. 35, 624, 2029, 2053

[CM06] Graham Cormode and S Muthukrishnan. Combinatorial algorithms for compressed sensing. In *Information Sciences and Systems, 2006 40th Annual Conference on*, pages 198–201. IEEE, 2006. 35, 619, 624

[CMDL+15] Andrzej Cichocki, Danilo Mandic, Lieven De Lathauwer, Guoxu Zhou, Qibin Zhao, Cesar Caiafa, and Huy Anh Phan. Tensor decompositions for signal processing applications: From two-way to multiway component analysis. *IEEE Signal Processing Magazine*, 32(2):145–163, 2015. 57, 1469

[CMM11] Alexandra Carpentier, Odalric-Ambrym Maillard, and Rémi Munos. Sparse recovery with brownian sensing. In *Advances in Neural Information Processing Systems*, pages 1782–1790, 2011. 988

[CMM15] Michael B Cohen, Cameron Musco, and Christopher Musco. Ridge leverage scores for low-rank approximation. *arXiv preprint arXiv:1511.07263*, 2015. 1926

[CMN96] Charles J Colbourn, Wendy J Myrvold, and Eugene Neufeld. Two algorithms for unranking arborescences. *Journal of Algorithms*, 20(2):268–281, 1996. 29, 550

[CMP16] Michael B Cohen, Cameron Musco, and Jakub Pachocki. Online row sampling. In *International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*. arXiv preprint arXiv:1604.05448, 2016. 554

[CMS13] Michael S Crouch, Andrew McGregor, and Daniel Stubbs. Dynamic graphs in the sliding-window model. In *European Symposium on Algorithms*, pages 337–348. Springer, 2013. 2094

[CMSV17] Michael B Cohen, Aleksander Mądry, Piotr Sankowski, and Adrian Vladu. Negative-weight shortest paths and unit capacity minimum cost flow in $\widetilde{O}(m^{10/7} \log W)$ time. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms(SODA)*, pages 752–771. SIAM, 2017. 11, 3

[CMTV17] Michael B Cohen, Aleksander Madry, Dimitris Tsipras, and Adrian Vladu. Matrix scaling and balancing via box constrained newton's method and interior point methods. In *Foundations of Computer Science (FOCS), 2017 IEEE 58th Annual Symposium on*, pages 902–913. IEEE, 2017. 11, 3

[CNW15] Michael B Cohen, Jelani Nelson, and David P Woodruff. Optimal approximate matrix product in terms of stable rank. In *Proceedings of the 43rd International Colloquium on Automata, Languages and Programming (ICALP), Rome, Italy, July 12-15, 2016*. https://arxiv.org/pdf/1507.02268, 2015. 770, 1562, 1565

[CNW16] Michael B Cohen, Jelani Nelson, and David P. Woodruff. Optimal approximate matrix product in terms of stable rank. In *Proceedings of the 43rd International Colloquium on Automata, Languages and Programming (ICALP 2016), Rome, Italy, July 12-15, arXiv preprint arXiv:1507.02268*, 2016. 1922

[Coh16a] Michael B. Cohen. Nearly tight oblivious subspace embeddings by trace inequalities. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), Arlington, VA, USA, January 10-12, 2016*, pages 278–287, 2016. 25, 52, 548, 1109, 1370, 1426, 1922, 1966

[Coh16b] Michael B Cohen. Ramanujan graphs in polynomial time. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 276–281. IEEE, 2016. 595, 596

[Com09] P. Comon. Tensor Decompositions, State of the Art and Applications. *ArXiv e-prints*, 2009. 57, 1469

[Con78] John B. Conway. *Functions of one Complex Variable*, volume I of *Graduate Texts in Mathematics (Second Edition)*. Springer New York, Heidelberg, Berlin, 1978. 519

[COP03] Moses Charikar, Liadan O'Callaghan, and Rina Panigrahy. Better streaming algorithms for clustering problems. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 30–39. ACM, 2003. 2094

[Cox58] David R Cox. The regression analysis of binary sequences. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 215–242, 1958. 14, 74

[CP15] Michael B. Cohen and Richard Peng. ℓ_p row sampling by lewis weights. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing (STOC)*, STOC '15, pages 183–192, New York, NY, USA, 2015. https://arxiv.org/pdf/1412.0588. 52, 1109, 1116, 1133, 1185, 1200, 1253, 1338, 1341, 1488, 1515, 1717, 1966, 1971

[CPS18] Minmin Chen, Jeffrey Pennington, and Samuel S. Schoenholz. Dynamical isometry and a mean field theory of RNNs: Gating enables signal propagation in recurrent neural networks. *arXiv:1806.05394*, 2018. 358

[CR09] Emmanuel J. Candès and Benjamin Recht. Exact matrix completion via convex optimization. *Foundations of Computational Mathematics*, 9(6):717–772, 2009. 51, 1038

[CRT06a] Emmanuel J Candès, Justin Romberg, and Terence Tao. Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *IEEE Transactions on information theory*, 52(2):489–509, 2006. 623, 917

[CRT06b] Emmanuel J Candes, Justin K Romberg, and Terence Tao. Stable signal recovery from incomplete and inaccurate measurements. *Communications on pure and applied mathematics*, 59(8):1207–1223, 2006. lxvi, lxviii, 9, 33, 34, 35, 617, 619, 624, 762, 917, 989

[CS07] Maria Chudnovsky and Paul Seymour. The roots of the independence polynomial of a clawfree graph. *J. Combin. Theory Ser. B*, 97(3):350–357, 2007. 600

[Csi18] Dominik Csiba. Data sampling strategies in stochastic algorithms for empirical risk minimization. *arXiv preprint arXiv:1804.00437*, 2018. 13, 74

[CT65] James W Cooley and John W Tukey. An algorithm for the machine calculation of complex Fourier series. *Mathematics of computation*, 19(90):297–301, 1965. 37, 164, 676, 918

[CT05] Emmanuel Candes and Terence Tao. Decoding by linear programming. *arXiv preprint math/0502327*, 2005. 925

[CT06] Emmanuel J Candes and Terence Tao. Near-optimal signal recovery from random projections: Universal encoding strategies? *IEEE transactions on*

information theory, 52(12):5406–5425, 2006. lxvii, lxviii, 37, 38, 917, 918, 919, 923, 925

[CV95] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995. 14, 74

[CV14] Joon Hee Choi and S. Vishwanathan. Dfacto: Distributed factorization of tensors. In *NIPS*, pages 1296–1304, 2014. 1797

[CV15] Nicoló Colombo and Nikos Vlassis. Fastmotif: spectral sequence motif discovery. *Bioinformatics*, pages 2623–2631, 2015. 57, 1469

[CVMBB14] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014. 22, 352

[CVMG+14] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *EMNLP*, pages 1724–1734, 2014. 353

[CW87] Don Coppersmith and Shmuel Winograd. Matrix multiplication via arithmetic progressions. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing(STOC)*, pages 1–6. ACM, 1987. 12, 16, 4, 78, 761, 775, 792, 1133, 1426, 1468, 1515

[CW90] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. In *Journal of Symbolic Computation*, volume 9, pages 251–280, 1990. 1921, 1927

[CW09] Kenneth L. Clarkson and David P. Woodruff. Numerical linear algebra in the streaming model. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 205–214, 2009. 52, 1045, 1109, 1125, 1163, 1310, 1779, 2044

[CW13] Kenneth L. Clarkson and David P. Woodruff. Low rank approximation and regression in input sparsity time. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 81–90. https://arxiv.org/pdf/1207.6365, 2013. 49, 52, 13, 770, 1037, 1045, 1057, 1083, 1085, 1105, 1109, 1119, 1337, 1338, 1370, 1401, 1426, 1468, 1489, 1490, 1499, 1511, 1512, 1535, 1555, 1566, 1606, 1609, 1610, 1758, 1781, 1786, 1791, 1922, 1927, 1966, 1999, 2027, 2029, 2037, 2042, 2052, 2075, 2076, 2078, 2082

[CW15a] Kenneth L Clarkson and David P Woodruff. Input sparsity and hardness for robust subspace approximation. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 310–329. IEEE, https://arxiv.org/pdf/1510.06073, 2015. 54, 56, 246, 993, 1046, 1112, 1131, 1428, 1474, 1506, 1652, 1740, 1741, 1743, 1744, 1966, 1967, 1971, 2027

[CW15b] Kenneth L Clarkson and David P Woodruff. Sketching for m-estimators: A unified approach to robust regression. In *Proceedings of the Twenty-Sixth*

Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pages 921–939. SIAM, 2015. 52, 246, 993, 1109, 1338, 1455, 1456, 1740, 1747, 1748, 1749, 1966, 1967, 1971, 2027

[CWW19] Kenneth L. Clarkson, Ruosong Wang, and David P. Woodruff. Dimensionality reduction for tukey regression. In *ICML*, 2019. 1431

[CWZ+17] Changyou Chen, Wenlin Wang, Yizhe Zhang, Qinliang Su, and Lawrence Carin. A convergence analysis for a class of practical variance-reduction stochastic gradient mcmc. *arXiv preprint arXiv:1709.01180*, 2017. 181

[CYYM14] Kai-Wei Chang, Scott Wen-tau Yih, Bishan Yang, and Chris Meek. Typed tensor decomposition of knowledge bases for relation extraction. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1568–1579, 2014. 57, 1469

[Dal17] Arnak S Dalalyan. Theoretical guarantees for approximate sampling from smooth and log-concave densities. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 79(3):651–676, 2017. 16, 17, 18, 21, 171, 172, 174, 251, 253, 254, 262, 351, 356

[Dan16] Amit Daniely. Complexity theoretic limitations on learning halfspaces. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing (STOC)*, pages 105–117. ACM, 2016. 258

[DB13] Marco F Duarte and Richard G Baraniuk. Spectral compressive sensing.

Applied and Computational Harmonic Analysis, 35(1):111–129, 2013. 41, 676, 686, 763

[DB14] Alexandre Défossez and Francis Bach. Constant step size least-mean-square: Bias-variance trade-offs and optimal sampling distributions. *arXiv preprint arXiv:1412.0156*, 2014. 13, 74

[DB16] Aymeric Dieuleveut and Francis Bach. Nonparametric stochastic approximation with large step-sizes. *The Annals of Statistics*, 44(4):1363–1399, 2016. 13, 74

[DBLJ14] Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in neural information processing systems*, pages 1646–1654, 2014. 13, 74

[DCWY18] Raaz Dwivedi, Yuansi Chen, Martin J Wainwright, and Bin Yu. Log-concave sampling: Metropolis-hastings algorithms are fast! In *COLT*. arXiv preprint arXiv:1801.02309, 2018. 16, 18, 171, 174

[DDH+09] Anirban Dasgupta, Petros Drineas, Boulos Harb, Ravi Kumar, and Michael W Mahoney. Sampling algorithms and coresets for ℓ_p regression. *SIAM Journal on Computing*, 38(5):2060–2078, 2009. 14, 74, 1117, 1124, 1134, 1190, 1406, 1407, 1408, 1409, 1516, 1717, 1966, 1971, 1973, 2033, 2034, 2035, 2044, 2045, 2050, 2051, 2066, 2067

[DDT+08] Marco F Duarte, Mark A Davenport, Dharmpal Takhar, Jason N Laska, Ting Sun, Kevin F Kelly, and Richard G Baraniuk. Single-pixel imaging via compressive sampling. *IEEE signal processing magazine*, 25(2):83–91, 2008. 33, 617, 630

[DDTS06] David Leigh Donoho, Iddo Drori, Yaakov Tsaig, and Jean-Luc Starck. *Sparse solution of underdetermined linear equations by stagewise orthogonal matching pursuit*. Department of Statistics, Stanford University, 2006. 38, 918

[DFK+04] Petros Drineas, Alan M. Frieze, Ravi Kannan, Santosh Vempala, and V. Vinay. Clustering large graphs via the singular value decomposition. *Machine Learning*, 56(1-3):9–33, 2004. 48, 1036

[DFS16] Amit Daniely, Roy Frostig, and Yoram Singer. Toward deeper understanding of neural networks: The power of initialization and a dual view on expressivity. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2253–2261, 2016. 358

[DG94] Jean-Pierre Dedieu and R. J. Gregorac. Corrigendum: "Obreschkoff's theorem revisited: what convex sets are contained in the set of hyperbolic polynomials?" [J. Pure Appl. Algebra **81** (1992), no. 3, 269–278; MR1179101 (93g:12001)] by Dedieu. *J. Pure Appl. Algebra*, 93(1):111–112, 1994. 600

[DG04] Jeffrey Dean and Sanjay Ghemawat. MapReduce: Simplified data processing on large clusters. In *OSDI*, 2004. 2315

[DG08] Jeffrey Dean and Sanjay Ghemawat. MapReduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008. 2315

[DGL+99] Yevgeniy Dodis, Oded Goldreich, Eric Lehman, Sofya Raskhodnikova, Dana Ron, and Alex Samorodnitsky. Improved testing algorithms for monotonicity. In *Randomization, Approximation, and Combinatorial Optimization. Algorithms and Techniques*, pages 97–108. Springer, 1999. 2218

[DGR96] A Dutt, M Gu, and V Rokhlin. Fast algorithms for polynomial interpolation, integration, and differentiation. *SIAM Journal on Numerical Analysis*, 33(5):1689–1711, 1996. 198

[DGRS15] Alexandros G Dimakis, Anna Gál, Ankit Singh Rawat, and Zhao Song. Batch codes through dense graphs without short cycles. In *ISIT*, 2015. xxix

[DHKP97] Martin Dietzfelbinger, Torben Hagerup, Jyrki Katajainen, and Martti Penttonen. A reliable randomized algorithm for the closest-pair problem. *Journal of Algorithms*, 25(1):19–51, 1997. 1334

[DJS+19] Huaian Diao, Rajesh Jayaram, Zhao Song, Wen Sun, and David P. Woodruff. Optimal sketching for kronecker product regression and low rank approximation. *Manuscript*, 2019. xxix, 2024, 2514, 2515, 2516

[DK17] Arnak S Dalalyan and Avetik G Karagulyan. User-friendly guarantees for the langevin monte carlo with inaccurate gradient. *arXiv preprint arXiv:1710.00095*, 2017. 16, 17, 171, 172, 181

[DKM06] Petros Drineas, Ravi Kannan, and Michael W Mahoney. Fast monte carlo algorithms for matrices ii: Computing a low-rank approximation to a matrix. *SIAM Journal on computing*, 36(1):158–183, 2006. 1338

[DKP+17] David Durfee, Rasmus Kyng, John Peebles, Anup B. Rao, and Sushant Sachdeva. Sampling random spanning trees faster than matrix multiplication. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 730–742, 2017. 29, 550, 555, 567, 568

[DKR02] Petros Drineas, Iordanis Kerenidis, and Prabhakar Raghavan. Competitive recommendation systems. In *STOC*, 2002. 48, 1036

[DLDM98] Lieven De Lathauwer and Bart De Moor. From matrix to tensor: Multilinear algebra and signal processing. In *Institute of Mathematics and Its Applications Conference Series*, volume 67, pages 1–16. Citeseer, 1998. 57, 1469

[DLL+19] Simon S Du, Jason D Lee, Haochuan Li, Liwei Wang, and Xiyu Zhai. Gradient descent finds global minima of deep neural networks. In *ICML*. https://arxiv.org/pdf/1811.03804, 2019. 256, 1909, 2140

[DLT+18] Simon S. Du, Jason D. Lee, Yuandong Tian, Barnabás Póczos, and Aarti Singh. Gradient descent learns one-hidden-layer CNN: don't be afraid of spurious local minima. In *ICML*, 2018. 21, 250, 257, 351, 2140

[DLTY06] Ke-xue Dai, Guo-hui Li, Dan Tu, and Jian Yuan. Prospects and current studies on background subtraction techniques for moving objects detection from surveillance video. *Journal of Image and Graphics*, 7(002), 2006. 630

[DM08] Wei Dai and Olgica Milenkovic. Subspace pursuit for compressive sensing: Closing the gap between performance and complexity. Technical report, ILLINOIS UNIV AT URBANA-CHAMAPAIGN, 2008. 38, 918

[DM16] Alain Durmus and Eric Moulines. High-dimensional bayesian inference via the unadjusted langevin algorithm. *arXiv preprint arXiv:1605.01559*, 2016. 205, 209

[DMIMW12] Petros Drineas, Malik Magdon-Ismail, Michael W Mahoney, and David P Woodruff. Fast approximation of matrix coherence and statistical leverage. *Journal of Machine Learning Research*, 13(Dec):3475–3506, 2012. 1338, 1468, 1568, 1593, 1907, 1922, 1926

[DMM06a] Petros Drineas, Michael W Mahoney, and S Muthukrishnan. Sampling algorithms for ℓ_2 regression and applications. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pages 1127–1136. Society for Industrial and Applied Mathematics, 2006. 1907

[DMM06b] Petros Drineas, Michael W. Mahoney, and S. Muthukrishnan. Subspace sampling and relative-error matrix approximation: Column-based methods. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, 9th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, APPROX 2006 and 10th International Workshop on Randomization and Computation, RANDOM 2006, Barcelona, Spain, August 28-30 2006, Proceedings*, pages 316–326, 2006. 52, 1109, 1433, 1468

[DMM06c] Petros Drineas, Michael W. Mahoney, and S. Muthukrishnan. Subspace sampling and relative-error matrix approximation: Column-row-based methods. In *Algorithms - ESA 2006, 14th Annual European Symposium, Zurich, Switzerland, September 11-13, 2006, Proceedings*, pages 304–314, 2006. 52, 1109, 1433, 1468, 1922

[DMM08] Petros Drineas, Michael W. Mahoney, and S. Muthukrishnan. Relative-error CUR matrix decompositions. *SIAM J. Matrix Analysis Applications*, 30(2):844–881, 2008. 52, 1109, 1114, 1433, 1468, 1476, 1604

[DMM+17] Sarah Dean, Horia Mania, Nikolai Matni, Benjamin Recht, and Stephen Tu. On the sample complexity of the linear quadratic regulator. *arXiv preprint arXiv:1710.01688*, 2017. 257

[DMMS11] Petros Drineas, Michael W Mahoney, S Muthukrishnan, and Tamás Sarlós. Faster least squares approximation. *Numerische Mathematik*, 117(2):219–249, 2011. 1337, 1907, 1922, 1931

[Don06] David L. Donoho. Compressed sensing. *IEEE Trans. Information Theory*, 52(4):1289–1306, 2006. lxvi, lxviii, 9, 33, 34, 35, 617, 619, 624, 917

[DPFJ16] Kévin Degraux, Gabriel Peyré, Jalal Fadili, and Laurent Jacques. Sparse support recovery with non-smooth loss functions. In *Advances in Neural Information Processing Systems*, pages 4269–4277, 2016. 988

[DPPR17] David Durfee, John Peebles, Richard Peng, and Anup B. Rao. Determinant-preserving sparsification of SDDM matrices with applications to counting

and sampling spanning trees. In *Proceedings of the 2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, 2017. 29, 550, 555

[DR10] Amit Deshpande and Luis Rademacher. Efficient volume sampling for row/column subset selection. In *2010 51st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 329–338. IEEE, https://arxiv.org/pdf/1004.4057, 2010. 1114, 1476, 1604

[DRD18] Arnak S Dalalyan and Lionel Riou-Durand. On sampling from a log-concave density using kinetic langevin diffusions. *arXiv preprint arXiv:1807.09382*, 2018. 16, 17, 171, 172

[DRW+16] Kumar Avinava Dubey, Sashank J Reddi, Sinead A Williamson, Barnabas Poczos, Alexander J Smola, and Eric P Xing. Variance reduction in stochastic gradient langevin dynamics. In *Advances in neural information processing systems*, pages 1154–1162, 2016. 181

[DS13] Alexander Munro Davie and Andrew James Stothers. Improved bound for complexity of matrix multiplication. *Proceedings of the Royal Society of Edinburgh Section A: Mathematics*, 143(2):351–369, 2013. 12, 4

[DSL08] Vin De Silva and Lek-Heng Lim. Tensor rank and the ill-posedness of the best low-rank approximation problem. *SIAM Journal on Matrix Analysis and Applications*, 30(3):1084–1127, 2008. 59, 1471, 1497, 1652

[DSM+16] Alain Durmus, Umut Simsekli, Eric Moulines, Roland Badeau, and Gaël Richard. Stochastic gradient richardson-romberg markov chain monte carlo.

In *Advances in Neural Information Processing Systems*, pages 2047–2055, 2016. 181

[DSS16] Amit Daniely and Shai Shalev-Shwartz. Complexity theoretic limitations on learning dnf's. In *Conference on Learning Theory (COLT)*, pages 815–830, 2016. 258

[DSSW18] Huaian Diao, Zhao Song, Wen Sun, and David P. Woodruff. Sketching for kronecker product regression and p-splines. In *AISTATS*. https://arxiv.org/pdf/1712.09473, 2018. xxix, 2024, 2025, 2026, 2028, 2038, 2039, 2040, 2041, 2066

[DSWY19] Huaian Diao, Zhao Song, David P. Woodruff, and Xin Yang. Total least squares regression in input sparsity time. *Manuscript*, 2019. xxix

[DTMR18] Sarah Dean, Stephen Tu, Nikolai Matni, and Benjamin Recht. Safely learning to control the constrained linear quadratic regulator. *arXiv preprint arXiv:1809.10121*, 2018. 257

[Dun10] Mark Dunster. *Legendre and Related Functions*. Handbook of Mathematical Functions, Cambridge University Press, 2010. 785

[DV06] Amit Deshpande and Santosh Vempala. Adaptive sampling and fast low-rank matrix approximation. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 292–303. Springer, 2006. 1338

[DV07] Amit Deshpande and Kasturi R. Varadarajan. Sampling-based dimension reduction for subspace approximation. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007*, pages 641–650, 2007. 54, 1112, 1474

[DVTV09] Amit Deshpande, Kasturi R. Varadarajan, Madhur Tulsiani, and Nisheeth K. Vishnoi. Algorithms and hardness for subspace approximation. *CoRR*, abs/0912.1403, 2009. 56, 1428

[DW17] Huaian Diao and David P. Woodruff. Kronecker product and spline regression. *manuscript*, 2017. 1518

[DWG+13] Denisa Duma, Mary Wootters, Anna C Gilbert, Hung Q Ngo, Atri Rudra, Matthew Alpert, Timothy J Close, Gianfranco Ciardo, and Stefano Lonardi. Accurate decoding of pooled sequenced data using compressed sensing. In *International Workshop on Algorithms in Bioinformatics*, pages 70–84. Springer, 2013. 33, 617

[DZHZ06] Chris Ding, Ding Zhou, Xiaofeng He, and Hongyuan Zha. R1-pca: rotational invariant ℓ_1-norm principal component analysis for robust subspace factorization. In *Proceedings of the 23rd international conference on Machine learning*, pages 281–288. ACM, 2006. l, lxxv, 1286, 1287, 1288, 1298

[DZPS19] Simon S Du, Xiyu Zhai, Barnabas Poczos, and Aarti Singh. Gradient descent provably optimizes over-parameterized neural networks. In *ICLR*. https://arxiv.org/pdf/1810.02054, 2019. 257, 352, 1909, 2140, 2142, 2143, 2144, 2145, 2146, 2150, 2152, 2154, 2157, 2159, 2161, 2174

[ECG+09] Yaniv Erlich, Kenneth Chang, Assaf Gordon, Roy Ronen, Oron Navon, Michelle Rooks, and Gregory J Hannon. DNA sudoku-harnessing high-throughput sequencing for multiplexed specimen analysis. *Genome research*, 19(7):1243–1253, 2009. 33, 617

[EFW10] Alexandre V. Evfimievski, Ronald Fagin, and David P. Woodruff. Epistemic privacy. *J. ACM*, 58(1):2, 2010. 1046

[EIM11] Alina Ene, Sungjin Im, and Benjamin Moseley. Fast clustering using MapReduce. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 681–689. ACM, 2011. 2315

[EKK+00] Funda Ergün, Sampath Kannan, S Ravi Kumar, Ronitt Rubinfeld, and Mahesh Viswanathan. Spot-checkers. *Journal of Computer and System Sciences*, 60(3):717–751, 2000. 2218

[Elm90] Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990. 22, 352

[ELMM19] Yonina Eldar, Jerry Li, Cameron Musco, and Christopher Musco. The sample complexity of toeplitz covariance estimation. In *manuscript*, 2019. 1906

[EM06] Paul HC Eilers and Brian D Marx. Multidimensional density smoothing with p-splines. In *Proceedings of the 21st international workshop on statistical modelling*, 2006. 2024

[End10] Joachim HG Ender. On compressive sensing applied to radar. *Signal Processing*, 90(5):1402–1414, 2010. 622

[ES09] Lars Eldén and Berkant Savas. A newton-grassmann method for computing the best multilinear rank-(r1,r2,r3) approximation of a tensor. *SIAM J. Matrix Analysis Applications*, 31(2):248–271, 2009. 57, 1469

[ESAZ09] Yaniv Erlich, Noam Shental, Amnon Amir, and Or Zuk. Compressed sensing approach for high throughput carrier screen. In *Communication, Control, and Computing, 2009. Allerton 2009. 47th Annual Allerton Conference on*, pages 539–544. IEEE, 2009. 33, 617

[EV03] Cristian Estan and George Varghese. New directions in traffic measurement and accounting: Focusing on the elephants, ignoring the mice. *ACM Transactions on Computer Systems (TOCS)*, 21(3):270–313, 2003. 34, 618

[Exc13] Stack Exchange. Low-rank matrix approximation in terms of entry-wise ℓ_1 norm, 2013. 53, 1110

[FB17] C. Daniel Freeman and Joan Bruna. Topology and geometry of half-rectified network optimization. In *ICLR*, 2017. 351

[FEGK13] Ahmed K Farahat, Ahmed Elgohary, Ali Ghodsi, and Mohamed S Kamel. Distributed column subset selection on mapreduce. In *2013 IEEE 13th International Conference on Data Mining (ICDM)*, pages 171–180. IEEE, 2013. 1318, 1433, 1476, 1784, 1788

[Fei02] Uriel Feige. Relations between average case complexity and approximation complexity. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing(STOC)*, pages 534–543. ACM, 2002. 1041, 1081, 1671

[Fel80] H. J. Fell. On the zeros of convex combinations of polynomials. *Pacific J. Math.*, 89(1):43–50, 1980. 600

[FFSS07] Dan Feldman, Amos Fiat, Micha Sharir, and Danny Segev. Bi-criteria linear-time approximations for generalized k-mean/median/center. In *Proceedings of the 23rd ACM Symposium on Computational Geometry, Gyeongju, South Korea, June 6-8, 2007*, pages 19–26, 2007. 1474

[FGKS15] Roy Frostig, Rong Ge, Sham M Kakade, and Aaron Sidford. Competing with the empirical risk minimizer in a single pass. In *Conference on learning theory (COLT)*, pages 728–763, 2015. 13, 15, 74, 77

[FGRW12] Vitaly Feldman, Venkatesan Guruswami, Prasad Raghavendra, and Yi Wu. Agnostic learning of monomials by halfspaces is hard. *SIAM Journal on Computing*, 41(6):1558–1590, 2012. 13, 74

[FGV17] V. Feldman, C. Guzmán, and S. S. Vempala. Statistical query algorithms for mean vector estimation and stochastic convex optimization. In *SODA*, pages 1265–1277, 2017. 1967

[FHHP11] Wai Shing Fung, Ramesh Hariharan, Nicholas JA Harvey, and Debmalya Panigrahi. A general framework for graph sparsification. In *Proceedings of the forty-third annual ACM symposium on Theory of computing (STOC)*, pages 71–80. ACM, 2011. 551, 552, 556, 559, 561

[Fis04] Eldar Fischer. The art of uninformed decisions: A primer to property testing.

Current Trends in Theoretical Computer Science: The Challenge of the New Century, 1:229–264, 2004. 2218

[FIS05] Gereon Frahling, Piotr Indyk, and Christian Sohler. Sampling in dynamic data streams and applications. In *Symposium on Computational Geometry 2005*, pages 142–149, 2005. 2135

[FKG71] Cees M Fortuin, Pieter W Kasteleyn, and Jean Ginibre. Correlation inequalities on some partially ordered sets. *Communications in Mathematical Physics*, 22(2):89–103, 1971. 28, 549

[FKM+05] Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. On graph problems in a semi-streaming model. *Theoretical Computer Science*, 348(2-3):207–216, 2005. 2094

[FKV04] Alan M. Frieze, Ravi Kannan, and Santosh Vempala. Fast monte-carlo algorithms for finding low-rank approximations. *J. ACM*, 51(6):1025–1041, 2004. 1468

[FL11] Dan Feldman and Michael Langberg. A unified framework for approximating and clustering data. In *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San Jose, CA, USA, 6-8 June 2011*, pages 569–578, 2011. 54, 1112, 2089, 2090, 2094, 2098

[FL12] Albert Fannjiang and Wenjing Liao. Coherence pattern-guided compressive sensing with unresolved grids. *SIAM Journal on Imaging Sciences*, 5(1):179–202, 2012. 763

[FLP15] Dimitris Fotakis, Michael Lampis, and Vangelis Th Paschos. Sub-exponential approximation schemes for csps: From dense to almost sparse. *arXiv preprint arXiv:1507.04391*, 2015. 1269, 1274

[FM92] Tomás Feder and Milena Mihail. Balanced matroids. In *Proceedings of the twenty-fourth annual ACM symposium on Theory of computing (STOC)*, pages 26–38. ACM, 1992. 555

[FMMN11] Shmuel Friedland, V Mehrmann, A Miedlar, and M Nkengla. Fast low rank approximations of matrices and tensors. *Electron. J. Linear Algebra*, 22(10311048):462, 2011. 58, 1470, 1476

[FMPS13] Shmuel Friedland, Volker Mehrmann, Renato Pajarola, and Susanne K. Suter. On best rank one approximation of tensors. *Numerical Lin. Alg. with Applic.*, 20(6):942–955, 2013. 57, 1469

[FMS07] Dan Feldman, Morteza Monemizadeh, and Christian Sohler. A ptas for k-means clustering based on weak coresets. In *Proceedings of the twenty-third annual symposium on Computational geometry*, pages 11–18. ACM, 2007. 2094

[FMS+10] Jon Feldman, S. Muthukrishnan, Anastasios Sidiropoulos, Clifford Stein, and Zoya Svitkina. On distributing symmetric streaming computations. *ACM Transactions on Algorithms*, 6(4), 2010. Previously in SODA'08. 2315

[FMSW10] Dan Feldman, Morteza Monemizadeh, Christian Sohler, and David P. Woodruff. Coresets and sketches for high dimensional subspace approxima-

tion problems. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), 2010, Austin, Texas, USA, January 17-19, 2010*, pages 630–649, 2010. 54, 1112

[Fou11] Simon Foucart. Hard thresholding pursuit: an algorithm for compressive sensing. *SIAM Journal on Numerical Analysis*, 49(6):2543–2563, 2011. 38, 918

[Fre75] Michael L Fredman. On computing the length of longest increasing subsequences. *Discrete Mathematics*, 11(1):29–35, 1975. 2218

[FRS16] Zachary Friggstad, Mohsen Rezapour, and Mohammad R Salavatipour. Local search yields a ptas for k-means in doubling metrics. In *Foundations of Computer Science (FOCS), 2016 IEEE 57th Annual Symposium on*, pages 365–374. IEEE, 2016. 2094

[FS97] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997. 14, 74

[FS99] Roger Fischlin and Jean-Pierre Seifert. Tensor-based trapdoors for cvp and their application to public key cryptography. *Cryptography and Coding*, pages 801–801, 1999. 57, 1469

[FS05] Gereon Frahling and Christian Sohler. Coresets in dynamic geometric data streams. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing (STOC)*, pages 209–217. ACM, 2005. 2090, 2094, 2136

[FS12] Dan Feldman and Leonard J Schulman. Data reduction for weighted and outlier-resistant clustering. In *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms*, pages 1343–1354. Society for Industrial and Applied Mathematics, 2012. 2094, 2136, 2137

[FSGM+99] Min Fang, Narayanan Shivakumar, Hector Garcia-Molina, Rajeev Motwani, and Jeffrey D Ullman. Computing iceberg queries efficiently. In *Internaational Conference on Very Large Databases (VLDB'98), New York, August 1998*. Stanford InfoLab, 1999. 33, 617

[FSS13] Dan Feldman, Melanie Schmidt, and Christian Sohler. Turning big data into tiny data: Constant-size coresets for k-means, pca and projective clustering. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1434–1453. Society for Industrial and Applied Mathematics, 2013. 48, 1036

[FT07] Shmuel Friedland and Anatoli Torokhti. Generalized rank-constrained matrix approximations. *SIAM Journal on Matrix Analysis and Applications*, 29(2):656–659, 2007. 1135, 1508

[FT15] Shmuel Friedland and Venu Tammali. Low-rank approximation of tensors. In *Numerical Algebra, Matrix Theory, Differential-Algebraic Equations and Control Theory*, pages 377–411. Springer, 2015. 57, 58, 1469, 1470, 1476

[Gan05] Sumit Ganguly. Counting distinct items over update streams. In *International Symposium on Algorithms and Computation*, pages 505–514. Springer, 2005. 2092, 2117

[Gao08] Junbin Gao. Robust ℓ_1 principal component analysis and its bayesian variational inference. *Neural computation*, 20(2):555–572, 2008. 53, 516, 1111

[GBT84] Harold N Gabow, Jon Louis Bentley, and Robert E Tarjan. Scaling and related techniques for geometry problems. In *Proceedings of the sixteenth annual ACM symposium on Theory of computing (STOC)*, pages 135–143. ACM, 1984. 1334

[GG11] Nicolas Gillis and François Glineur. Low-rank matrix approximation with weights or missing data is np-hard. *SIAM Journal on Matrix Analysis and Applications*, 32(4):1149–1165, 2011. 51, 1038, 1081, 1334

[GGH14] Quanquan Gu, Huan Gui, and Jiawei Han. Robust tensor decomposition with gross corruption. In *Advances in Neural Information Processing Systems(NIPS)*, pages 1422–1430, 2014. 58, 1470

[GGI+02] Anna C Gilbert, Sudipto Guha, Piotr Indyk, S Muthukrishnan, and Martin Strauss. Near-optimal sparse Fourier representations via sampling. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 152–161. ACM, 2002. 38, 40, 46, 623, 636, 676, 682, 688, 759, 762, 919

[GHK15] Rong Ge, Qingqing Huang, and Sham M Kakade. Learning mixtures of gaussians in high dimensions. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing (STOC)*, pages 761–770. ACM, https://arxiv.org/pdf/1503.00424, 2015. 57, 1469

[GI10] Anna Gilbert and Piotr Indyk. Sparse recovery using sparse matrices. *Proceedings of the IEEE*, 98(6):937–947, 2010. 8, 34, 618

[Gil98] David Gillman. A chernoff bound for random walks on expander graphs. *SIAM Journal on Computing*, 27(4):1203–1220, 1998. 26, 27, 502, 503

[GK09] Rahul Garg and Rohit Khandekar. Gradient descent with sparsification: an iterative algorithm for sparse recovery with restricted isometry property. In *ICML*, volume 9, pages 337–344, 2009. 38, 918

[GKK12] Ashish Goel, Michael Kapralov, and Sanjeev Khanna. On the communication and streaming complexity of maximum bipartite matching. In *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms*, pages 468–485. SIAM, 2012. 2094

[GKKT17] Surbhi Goel, Varun Kanade, Adam Klivans, and Justin Thaler. Reliably learning the ReLU in polynomial time. In *Conference on Learning Theory (COLT)*, 2017. 258

[GKM18] Surbhi Goel, Adam Klivans, and Raghu Meka. Learning one convolutional layer with overlapping patches. In *ICML*. arXiv preprint arXiv:1802.02547, 2018. 257, 1451

[GKP12] Ashish Goel, Michael Kapralov, and Ian Post. Single pass sparsification in the streaming model with edge deletions. *arXiv preprint arXiv:1203.4900*, 2012. 2094

[GKU19] Mohsen Ghaffari, Fabian Kuhn, and Jara Uitto. Conditional hardness results for massively parallel computation from distributed lower bounds. In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 2019. 1917

[GL89] Oded Goldreich and Leonid A Levin. A hard-core predicate for all one-way functions. In *Proceedings of the twenty-first annual ACM symposium on Theory of computing*, pages 25–32. ACM, 1989. 38, 919

[GL04] Andreas Goerdt and André Lanka. An approximation hardness result for bipartite clique. In *Electronic Colloquium on Computational Complexity, Report*, volume 48. https://eccc.weizmann.ac.il/report/2004/048/, 2004. 1081, 1672

[GLM17] Rong Ge, Jason D. Lee, and Tengyu Ma. Learning one-hidden-layer neural networks with landscape design. In *ICLR*, 2017. 21, 251, 351

[GLM18] Rong Ge, Jason D. Lee, and Tengyu Ma. Learning one-hidden-layer neural networks with landscape design. In *ICLR*. http://arxiv.org/pdf/1711.00501, 2018. 2140

[GLPS10] Anna C Gilbert, Yi Li, Ely Porat, and Martin J Strauss. Approximate sparse recovery: optimizing time and measurements. *SIAM Journal on Computing 2012 (A preliminary version of this paper appears in STOC 2010)*, 41(2):436–453, 2010. xliii, 34, 35, 620, 623, 624, 625, 626, 627, 628, 632, 688, 690, 989

[GLPS17] Anna C Gilbert, Yi Li, Ely Porat, and Martin J Strauss. For-all sparse recovery in near-optimal time. *ACM Transactions on Algorithms (TALG)*, 13(3):32, 2017. 623

[GLSS18] Ankit Garg, Yin-Tat Lee, Zhao Song, and Nikhil Srivastava. A matrix expander chernoff bound. In *STOC*. https://arxiv.org/pdf/1704.03864, 2018. xxv, 5, 32, 553, 2141, 2513

[GM15] Rong Ge and Tengyu Ma. Decomposing overcomplete 3rd order tensors using sum-of-squares algorithms. In *The 18th. International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX'2015), and the 19th. International Workshop on Randomization and Computation (RANDOM'2015)*. https://arxiv.org/pdf/1504.05287, 2015. 58, 1470

[GMH13] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6645–6649. IEEE, 2013. 20, 250, 351

[GMMO00] Sudipto Guha, Nina Mishra, R. Motwani, and L. O'Callaghan. Clustering data streams. In *FOCS*, pages 359–366, 2000. 2094

[GMS05] Anna C Gilbert, S Muthukrishnan, and Martin Strauss. Improved time bounds for near-optimal sparse Fourier representations. In *Optics & Photonics 2005*, pages 59141A–59141A. International Society for Optics and Pho-

tonics, 2005. 37, 38, 40, 623, 636, 676, 682, 688, 689, 691, 762, 919, 923, 1904

[GNP+13] Anna C Gilbert, Hung Q Ngo, Ely Porat, Atri Rudra, and Martin J Strauss. ℓ_2/ℓ_2-foreach sparse recovery with low risk. In *International Colloquium on Automata, Languages, and Programming*, pages 461–472. Springer, 2013. 623

[GO16] Inc. Gurobi Optimization. Gurobi optimizer reference manual, 2016. 2041

[Gol65] Sidney Golden. Lower bounds for the Helmholtz function. *Physical Review*, 137(4B):B1127, 1965. 28, 506

[Goo99] Michael T Goodrich. Communication-efficient parallel sorting. *SIAM Journal on Computing*, 29(2):416–432, 1999. 2324, 2413

[Goo05] Joseph W Goodman. *Introduction to Fourier optics*. Roberts and Company Publishers, 2005. 36, 917

[GP14] Mina Ghashami and Jeff M Phillips. Relative errors for deterministic low-rank matrix approximations. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 707–717. Society for Industrial and Applied Mathematics, https://arxiv.org/pdf/1307.7454, 2014. 1310, 1779

[GQ14] Donald Goldfarb and Zhiwei Qin. Robust low-rank tensor recovery: Models and algorithms. *SIAM Journal on Matrix Analysis and Applications*, 35(1):225–253, 2014. 58, 1470

[GRV09] Navin Goyal, Luis Rademacher, and Santosh Vempala. Expanders via random spanning trees. In *Proceedings of the twentieth annual ACM-SIAM symposium on Discrete algorithms (SODA)*, pages 576–585. Society for Industrial and Applied Mathematics, 2009. 550, 552, 555

[GS+99] Antoine Guitton, William W Symes, et al. Robust and stable velocity analysis using the huber function. In *69th SEG meeting, Houston, USA, Expanded Abstracts*, pages 1166–1169, 1999. 990

[GS05] Alex Graves and Jürgen Schmidhuber. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, 18(5-6):602–610, 2005. 353

[GS12] Surya Ganguli and Haim Sompolinsky. Compressed sensing, sparsity, and dimensionality in neuronal information processing and data analysis. *Annual review of neuroscience*, 35:485–508, 2012. 33, 617

[GSS11] Shayan Oveis Gharan, Amin Saberi, and Mohit Singh. A randomized rounding approach to the traveling salesman problem. In *Proceedings of the 2011 IEEE 52nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 550–559, 2011. 550, 555

[GSS17] Alon Gonen and Shai Shalev-Shwartz. Fast rates for empirical risk minimization of strict saddle problems. *arXiv preprint arXiv:1701.04271*, 2017. 13, 74

[GSZ11] Michael T Goodrich, Nodari Sitchinava, and Qin Zhang. Sorting, searching, and simulation in the mapreduce framework. In *ISAAC*, volume 7074, pages 374–383. Springer, 2011. 2315, 2324, 2413, 2414, 2415

[Gue83] Alain Guenoche. Random spanning tree. *Journal of Algorithms*, 4(3):214–220, 1983. 29, 550

[GV15] Nicolas Gillis and Stephen A Vavasis. On the complexity of robust pca and ℓ_1-norm low-rank matrix approximation. *arXiv preprint arXiv:1509.09236*, 2015. 53, 1110, 1268, 1270, 1274, 1276, 1334, 1370

[GVL13] Gene H. Golub and Charles F. Van Loan. *Matrix computations*. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, Baltimore, MD, 2013. 2024

[GVS03] David Guillamet, Jordi Vitri, and Bernt Schiele. Introducing a weighted non-negative matrix factorization for image classification. *Pattern Recognition Letters*, 24(14):2447–2454, 2003. 1044

[GVS15] Ian J Goodfellow, Oriol Vinyals, and Andrew M Saxe. Qualitatively characterizing neural network optimization problems. In *ICLR*, 2015. 20, 250, 254, 351, 356, 364, 366

[GW95] Michel X Goemans and David P Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM (JACM)*, 42(6):1115–1145, 1995. 1269

[GWD14] Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *CoRR*, abs/1410.5401, 2014. 354

[GZ16] Venkatesan Guruswami and David Zuckerman. Robust Fourier and polynomial curve fitting. *Manuscript*, 2016. 765

[GZD10] Quanquan Gu, Jie Zhou, and Chris H. Q. Ding. Collaborative filtering: Weighted nonnegative matrix factorization incorporating user and item graphs. In *SDM*, 2010. 1044

[Haa81] Uffe Haagerup. The best constants in the khintchine inequality. *Studia Mathematica*, 70:231–283, 1981. 2505

[Hah98] Richard LT Hahnloser. On the piecewise analysis of networks of linear threshold neurons. *Neural Networks*, 11(4):691–697, 1998. 23, 353

[Har70] Richard A Harshman. Foundations of the parafac procedure: Models and conditions for an "explanatory" multi-modal factor analysis. ., 1970. 57, 1469, 1797

[Har14] Sariel Har-Peled. Low rank matrix approximation in linear time. *CoRR*, abs/1410.8802, 2014. 49, 1037

[Hås90] Johan Håstad. Tensor rank is np-complete. *Journal of Algorithms*, 11(4):644–654, 1990. 60, 1472, 1669, 1681, 1686, 1689

[Hås00] Johan Håstad. On bounded occurrence constraint satisfaction. *Information Processing Letters*, 74(1-2):1–6, 2000. 1670

[Hås01] Johan Håstad. Some optimal inapproximability results. *Journal of the ACM (JACM)*, 48(4):798–859, 2001. 1269, 1670, 1671

[Haz01] Michiel Hazewinkel. *Gram matrix*. Encyclopedia of Mathematics, Springer, 2001. 786

[HD08] Heng Huang and Chris Ding. Robust tensor factorization using r 1 norm. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8. IEEE, 2008. 57, 58, 1469, 1470

[Hea08] Alexander D Healy. Randomness-efficient sampling within nc. *Computational Complexity*, 17(1):3–37, 2008. 26, 501, 502, 509, 531, 537, 553

[HHW18] Bernhard Haeupler, D. Ellis Hershkowitz, and David Wajc. Round- and message-optimal distributed graph algorithms. In *arXiv preprint*. http://arxiv.org/pdf/1801.05127, 2018. 2317

[HIKP12a] Haitham Hassanieh, Piotr Indyk, Dina Katabi, and Eric Price. Nearly optimal sparse Fourier transform. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 563–578. ACM, 2012. lxvii, lxviii, 37, 38, 40, 46, 164, 623, 625, 636, 676, 682, 688, 689, 691, 694, 700, 701, 715, 719, 759, 762, 769, 803, 822, 919, 923, 924, 926, 965, 977, 978, 979, 996, 1015, 1016, 1017, 1018, 1904

[HIKP12b] Haitham Hassanieh, Piotr Indyk, Dina Katabi, and Eric Price. Simple and practical algorithm for sparse Fourier transform. In *Proceedings of the twenty-*

third annual ACM-SIAM symposium on Discrete Algorithms, pages 1183–1194. SIAM, 2012. 38, 164, 623, 636, 701, 762, 803, 919, 924, 1904

[HJLS13] David W Hosmer Jr, Stanley Lemeshow, and Rodney X Sturdivant. *Applied logistic regression*, volume 398. John Wiley & Sons, 2013. 14, 16, 17, 74, 171

[HK13] Daniel Hsu and Sham M Kakade. Learning mixtures of spherical gaussians: moment methods and spectral decompositions. In *Proceedings of the 4th conference on Innovations in Theoretical Computer Science(ITCS)*, pages 11–20. ACM, https://arxiv.org/pdf/1206.5766, 2013. 57, 1469

[HK15] Qingqing Huang and Sham M Kakade. Super-resolution off the grid. In *Advances in Neural Information Processing Systems*, pages 2647–2655, 2015. 763

[HKT16] Fumio Hiai, Robert Koenig, and Marco Tomamichel. Generalized log-majorization and multivariate trace inequalities. *arXiv preprint arXiv:1609.01999*, September 2016. 507

[HKZ11] Daniel Hsu, Sham M. Kakade, and Tong Zhang. Robust matrix decomposition with sparse corruptions. *ITIT*, 57(11):7221–7234, 2011. 1043

[HL13] Christopher J Hillar and Lek-Heng Lim. Most tensor problems are np-hard. In *Journal of the ACM (JACM)*, volume 60(6), page 45. https://arxiv.org/pdf/0911.1393, 2013. 60, 1472, 1673, 1676

[HLS+18] Elad Hazan, Holden Lee, Karan Singh, Cyril Zhang, and Yi Zhang. Spectral filtering for general linear dynamical systems. In *Advances in Neural Information Processing Systems (NIPS)*, 2018. 24, 257, 354

[HM17] Moritz Hardt and Tengyu Ma. Identity matters in deep learning. In *ICLR*, 2017. 21, 251, 257, 258, 277, 351

[HMR18] Moritz Hardt, Tengyu Ma, and Benjamin Recht. Gradient descent learns linear dynamical systems. *Journal of Machine Learning Research (JMLR)*, 19(29):1–44, 2018. 23, 257, 354

[HMRW14] Moritz Hardt, Raghu Meka, Prasad Raghavendra, and Benjamin Weitz. Computational limits for matrix completion. In *COLT*, 2014. 51, 1038

[HMT+13] Hong Huang, Satyajayant Misra, Wei Tang, Hajar Barani, and Hussein Al-Azzawi. Applications of compressed sensing in communications networks. *arXiv preprint arXiv:1305.3002*, 2013. 620

[HNH+13] Furong Huang, Niranjan U. N, Mohammad Umar Hakeem, Prateek Verma, and Animashree Anandkumar. Fast detection of overlapping communities via online tensor methods on gpus. *CoRR*, abs/1309.0787, 2013. 1797

[HO14] Nicholas JA Harvey and Neil Olver. Pipage rounding, pessimistic estimators and matrix concentration. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms (SODA)*, pages 926–945. Society for Industrial and Applied Mathematics, 2014. 553

[Hoe63] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963. 2506

[HPK05] Sariel Har-Peled and Akash Kushal. Smaller coresets for k-median and k-means clustering. In *Proceedings of the twenty-first annual symposium on Computational geometry*, pages 126–134. ACM, 2005. 2094

[HPM04] Sariel Har-Peled and Soham Mazumdar. On coresets for k-means and k-median clustering. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 291–300. ACM, 2004. 2094

[HPS05] Tamir Hazan, Simon Polak, and Amnon Shashua. Sparse image coding using a 3d non-negative tensor factorization. In *Tenth IEEE International Conference on Computer Vision(ICCV)*, volume 1, pages 50–57. IEEE, 2005. 57, 1469

[HR69] Josef Hadar and William R Russell. Rules for ordering uncertain prospects. *The American economic review*, 59(1):25–34, 1969. 654

[HR16] Ishay Haviv and Oded Regev. The restricted isometry property of subsampled Fourier matrices. In *SODA*, pages 288–297. arXiv preprint arXiv:1507.01768, 2016. lxvii, lxviii, 37, 38, 623, 762, 918, 919, 920, 923, 925, 1922

[HRRS11] Frank R Hampel, Elvezio M Ronchetti, Peter J Rousseeuw, and Werner A Stahel. *Robust statistics: the approach based on influence functions*, volume 196. John Wiley & Sons, 2011. 56, 1428

[HS90] Yingbo Hua and Tapan K Sarkar. Matrix pencil method for estimating parameters of exponentially damped/undamped sinusoids in noise. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 38(5):814–824, 1990. 687

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 23, 353

[HSE12] Bo Hu, Zhao Song, and Martin Ester. User features and social networks for topic modelling in online social media. In *The 2012 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, (ASONAM 2012), 26-29 August, 2012, Kadir Has University, Istanbul, Turkey.*, 2012. xxix

[HSS15] Samuel B Hopkins, Jonathan Shi, and David Steurer. Tensor principal component analysis via sum-of-square proofs. In *28th Annual Conference on Learning Theory (COLT)*, pages 956–1006. https://arxiv.org/pdf/1507.03269, 2015. 58, 1470

[HSS19] MohammadTaghi Hajiaghayi, Saeed Seddighin, and Xiaorui Sun. Massively parallel approximation algorithms for edit distance and longest common subsequence. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1654–1672. SIAM, 2019. 1917

[HSSS16] Samuel B Hopkins, Tselil Schramm, Jonathan Shi, and David Steurer. Fast spectral algorithms from sum-of-squares proofs: tensor decomposition and

planted sparse vectors. In *Proceedings of the 48th Annual Symposium on the Theory of Computing*. ACM, https://arxiv.org/pdf/1512.02337, 2016. 25, 58, 548, 1470

[HSSS19] MohammadTaghi Hajiaghayi, Masoud Seddighin, Saeed Seddighin, and Xiaorui Sun. Approximating lcs in linear time: Beating the barrier. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1181–1200. SIAM, 2019. 2192, 2214, 2216

[HSYZ18] Wei Hu, Zhao Song, Lin F. Yang, and Peilin Zhong. Nearly optimal dynamic k-means clustering for high-dimensional data. *arXiv preprint arXiv:1802.00459*, 2018. 2087

[HSZ17] Elad Hazan, Karan Singh, and Cyril Zhang. Learning linear dynamical systems via spectral filtering. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 6702–6712, 2017. 24, 257, 354

[HT16] Daniel Hsu and Matus Telgarsky. Greedy bi-criteria approximations for k-medians and k-means. *arXiv preprint arXiv:1607.06203*, 2016. 1474

[Hub64] Peter J. Huber. Robust estimation of a location parameter. *The Annals of Mathematical Statistics*, 35(1):73–101, 1964. 56, 246, 989, 992, 1007, 1370, 1428

[HW71] David Lee Hanson and Farroll Tim Wright. A bound on tail probabilities for quadratic forms in independent random variables. *The Annals of Mathematical Statistics*, 42(3):1079–1083, 1971. 2505

[HX16] Nicholas J. A. Harvey and Keyulu Xu. Generating random spanning trees via fast matrix multiplication. In *LATIN 2016: Theoretical Informatics*, volume 9644, pages 522–535, 2016. 29, 550

[HZRS16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 20, 22, 250, 252, 351

[IBY+07] Michael Isard, Mihai Budiu, Yuan Yu, Andrew Birrell, and Dennis Fetterly. Dryad: distributed data-parallel programs from sequential building blocks. *ACM SIGOPS Operating Systems Review*, 41(3):59–72, 2007. 2315

[IK14] Piotr Indyk and Michael Kapralov. Sample-optimal Fourier sampling in any constant dimension. In *Foundations of Computer Science (FOCS), 2014 IEEE 55th Annual Symposium on*, pages 514–523. IEEE, 2014. lxvii, lxviii, 37, 38, 40, 164, 623, 636, 676, 682, 683, 691, 762, 919, 920, 923, 924, 926, 927, 937, 1904

[IKP14] Piotr Indyk, Michael Kapralov, and Eric Price. (Nearly) Sample-optimal sparse Fourier transform. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 480–499. SIAM, 2014. 38, 46, 164, 623, 636, 682, 759, 762, 919, 1904

[IMS17a] Sungjin Im, Benjamin Moseley, and Xiaorui Sun. Efficient massively parallel methods for dynamic programming. In *Proceedings of the 49th Annual ACM*

SIGACT Symposium on Theory of Computing, pages 798–811. ACM, 2017. 1917

[IMS17b] Sungjin Im, Benjamin Moseley, and Xiaorui Sun. Efficient massively parallel methods for dynamic programming. In *STOC*, pages 798–811, 2017. 2315

[Ind04] Piotr Indyk. Algorithms for dynamic geometric problems over data streams. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 373–380. ACM, 2004. 2088, 2135

[Ind06] Piotr Indyk. Stable distributions, pseudorandom generators, embeddings, and data stream computation. *Journal of the ACM (JACM)*, 53(3):307–323, 2006. 666, 1118, 1170, 1186, 2030, 2051, 2070, 2074, 2086

[Ind07] Piotr Indyk. Sketching, streaming and sublinear-space algorithms. *Graduate course notes, available at*, 2007. 33, 617

[IP11] Piotr Indyk and Eric Price. K-median clustering, model-based compressive sensing, and sparse recovery for earth mover distance. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 627–636. ACM, 2011. 1262, 2094

[IPW11] Piotr Indyk, Eric Price, and David P Woodruff. On the power of adaptivity in sparse recovery. In *Foundations of Computer Science (FOCS), 2011 IEEE 52nd Annual Symposium on*, pages 285–294. IEEE, 2011. 623

[IPZ98] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? In *Proceedings. 39th Annual Sympo-*

sium on Foundations of Computer Science (FOCS), pages 653–662. IEEE, 1998. 1273, 1670

[IR08a] Piotr Indyk and Milan Ruzic. Near-optimal sparse recovery in the l1 norm. In *Foundations of Computer Science, 2008. FOCS'08. IEEE 49th Annual IEEE Symposium on*, pages 199–207. IEEE, 2008. 34, 617

[IR08b] Piotr Indyk and Milan Ruzic. Near-optimal sparse recovery in the l1 norm. In *Foundations of Computer Science, 2008. FOCS'08. IEEE 49th Annual IEEE Symposium on*, pages 199–207. IEEE, 2008. 989

[IR13] Piotr Indyk and Ilya Razenshteyn. On model-based RIP-1 matrices. In *International Colloquium on Automata, Languages, and Programming*, pages 564–575. Springer, 2013. 622

[Ise09] Arieh Iserles. *A first course in the numerical analysis of differential equations*. Number 44 in . Cambridge university press, 2009. 19, 182

[IT03] Piotr Indyk and Nitin Thaper. Fast image retrieval via embeddings. In *Workshop on Statistical and Computational Theories of Vision (at ICCV)*, 2003. 1262, 1264

[IVWW19] Piotr Indyk, Ali Vakilian, Tal Wagner, and David P. Woodruff. Sample-optimal low-rank approximation of distance matrices. In *COLT*, 2019. 2027

[IW97] Russell Impagliazzo and Avi Wigderson. P= BPP if E requires exponential circuits: Derandomizing the XOR lemma. In *Proceedings of the twenty-ninth*

annual ACM symposium on Theory of computing (STOC), pages 220–229. ACM, 1997. 1704

[IW05] P. Indyk and David P. Woodruff. Optimal approximations of the frequency moments of data streams. In *STOC*, pages 202–208, 2005. 1976

[Iwe08] Mark A Iwen. A deterministic sub-linear time sparse Fourier algorithm via non-adaptive compressed sensing methods. In *Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 20–29. Society for Industrial and Applied Mathematics, 2008. 38, 919, 1904

[Iwe10] Mark A Iwen. Combinatorial sublinear-time Fourier algorithms. *Foundations of Computational Mathematics*, 10(3):303–338, 2010. 38, 919, 920, 1904

[Iwe13] Mark A Iwen. Improved approximation guarantees for sublinear-time Fourier algorithms. *Applied And Computational Harmonic Analysis*, 34(1):57–82, 2013. 38, 623, 636, 762, 919, 920, 1904

[Jai10] Anil K Jain. Data clustering: 50 years beyond k-means. *Pattern recognition letters*, 31(8):651–666, 2010. 2088

[JGH18] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in neural information processing systems*, pages 8571–8580, 2018. 21, 251, 253, 254, 269, 328

[Jia06] Tiefeng Jiang. How many entries of a typical orthogonal matrix can be approximated by independent normals? *The Annals of Probability*, 34(4):1497–1529, 2006. 1929, 1941

[JLGJ18] Chi Jin, Lydia T Liu, Rong Ge, and Michael I Jordan. On the local minima of the empirical risk. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 4901–4910, 2018. 13, 74

[JMZ15] Bo Jiang, Shiqian Ma, and Shuzhong Zhang. Tensor principal component analysis via convex optimization. *Mathematical Programming*, 150(2):423–457, 2015. 58, 1470

[JO14a] Prateek Jain and Sewoong Oh. Learning mixtures of discrete product distributions using spectral decompositions. In *27th Annual Conference on Learning Theory (COLT)*, pages 824–856. https://arxiv.org/pdf/1311.2972, 2014. 57, 1469

[JO14b] Prateek Jain and Sewoong Oh. Provable tensor factorization with missing data. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1431–1439. https://arxiv.org/pdf/1406.2784, 2014. 58, 1470

[JPT13] Gabriela Jeronimo, Daniel Perrucci, and Elias Tsigaridas. On the minimum of a polynomial function on a basic closed semialgebraic set and applications. *SIAM Journal on Optimization*, 23(1):241–255, 2013. 1053, 1507

[JSA15] Majid Janzamin, Hanie Sedghi, and Anima Anandkumar. Beating the perils of non-convexity: Guaranteed training of neural networks using tensor meth-

ods. In *arXiv preprint*. https://arxiv.org/pdf/1506.08473, 2015. 57, 358, 1469

[JST11] Hossein Jowhari, Mert Sağlam, and Gábor Tardos. Tight bounds for lp samplers, finding duplicates in streams, and related problems. In *Proceedings of the thirtieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 49–58. ACM, 2011. 619

[JW18] Rajesh Jayaram and David P Woodruff. Perfect lp sampling in a data stream. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 544–555. IEEE, 2018. 2072, 2073

[JZ13] Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in neural information processing systems*, pages 315–323, 2013. 13, 15, 74, 77

[KABO10] Alexandros Karatzoglou, Xavier Amatriain, Linas Baltrunas, and Nuria Oliver. Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 79–86. ACM, 2010. 57, 1469

[KAC+08] Fredrik Kahl, Sameer Agarwal, Manmohan Krishna Chandraker, David Kriegman, and Serge Belongie. Practical global optimization for multiview geometry. *International Journal of Computer Vision*, 79(3):271–284, 2008. 53, 1111

[Kah97] Nabil Kahale. Large deviation bounds for Markov chains. *Combinatorics, Probability and Computing*, 6(04):465–474, 1997. 26, 502

[Kap16] Michael Kapralov. Sparse Fourier transform in any constant dimension with nearly-optimal sample complexity in sublinear time. In *Symposium on Theory of Computing Conference, STOC'16, Cambridge, MA, USA, June 19-21, 2016*, 2016. lxvii, lxviii, 37, 38, 164, 623, 636, 762, 919, 923, 925, 926, 1903, 1904

[Kap17] Michael Kapralov. Sample efficient estimation and recovery in sparse fft via isolation on average. In *Foundations of Computer Science, 2017. FOCS'17. IEEE 58th Annual IEEE Symposium on*. https://arxiv.org/pdf/1708.04544, 2017. lxvii, lxviii, 37, 38, 164, 623, 636, 637, 919, 923, 925, 926, 964, 965, 993, 1904

[Kar84] Narendra Karmarkar. A new polynomial-time algorithm for linear programming. In *Proceedings of the sixteenth annual ACM symposium on Theory of computing(STOC)*, pages 302–311. ACM, 1984. 10, 13, 3, 15

[Kar93] David R Karger. Global min-cuts in rnc, and other ramifications of a simple min-cut algorithm. In *SODA*, volume 93, pages 21–30, 1993. 555, 556

[Kaw10] Akitoshi Kawamura. Lipschitz continuous ordinary differential equations are polynomial-space complete. *Computational Complexity*, 19(2):305–332, 2010. 20, 177

[Kaw16] Kenji Kawaguchi. Deep learning without poor local minima. In *Advances in Neural Information Processing Systems*, pages 586–594, 2016. 21, 251, 257

[KB06] Tamara Kolda and Brett Bader. The tophits model for higher-order web link analysis. In *Workshop on link analysis, counterterrorism and security*, volume 7, pages 26–29, 2006. 57, 1469

[KB09] Tamara G. Kolda and Brett W. Bader. Tensor decompositions and applications. *SIAM Review*, 51(3):455–500, 2009. 57, 1469

[KB13] Nal Kalchbrenner and Phil Blunsom. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709, 2013. 22, 352

[KBG+10] Raghunandan M Kainkaryam, Angela Bruex, Anna C Gilbert, John Schiefelbein, and Peter J Woolf. poolmc: Smart pooling of mrna samples in microarray experiments. *BMC bioinformatics*, 11(1):299, 2010. 33, 617

[KBJ78] Roger Koenker and Gilbert Bassett Jr. Regression quantiles. *Econometrica: journal of the Econometric Society*, pages 33–50, 1978. 1430

[KC07] Yong-Deok Kim and Seungjin Choi. Nonnegative tucker decomposition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*, pages 1–8. IEEE, 2007. 1706

[KC12] Akitoshi Kawamura and Stephen Cook. Complexity theory for operators in analysis. *ACM Transactions on Computation Theory (TOCT) (A preliminary version of this paper appeared in STOC 2010)*, 4(2):5, 2012. 20, 177

[KDS08] Wim P Krijnen, Theo K Dijkstra, and Alwin Stegeman. On the non-existence of optimal solutions and the occurrence of "degeneracy" in the can-decomp/parafac model. *Psychometrika*, 73(3):431–439, 2008. 59, 1471

[KF82] Ker-I Ko and Harvey Friedman. Computational complexity of real functions. *Theoretical Computer Science*, 20(3):323–352, 1982. 20, 177

[KG12] Serap Kirbiz and Bilge Günsel. Perceptually weighted non-negative matrix factorization for blind single-channel music source separation. In *ICPR*, 2012. 1044

[KH01] Roger Koenker and Kevin F Hallock. Quantile regression. *Journal of economic perspectives*, 15(4):143–156, 2001. 14, 74

[KHL89] JB Kruskal, RA Harshman, and ME Lundy. How 3-mfa data can cause degenerate parafac solutions, among other relationships. *Multiway data analysis*, pages 115–121, 1989. 59, 1471

[Kir47] Gustav Kirchhoff. Uber die auflosung der gliechungen, auf welche man bei der undersuchung der linearen vertheilung galvanischer strome gefuhrt wird. *Poggendorgs Ann. Phys. Chem.*, pages 497–508, 1847. 29, 550

[KK03] Qifa Ke and Takeo Kanade. Robust subspace computation using ℓ_1 norm. *Technical Report CMU-CS-03-172, Carnegie Mellon University, Pittsburgh, PA.*, 2003. 53, 1110, 1370

[KK05] Qifa Ke and Takeo Kanade. Robust ℓ_1 norm factorization in the presence of outliers and missing data by alternative convex programming. In *2005 IEEE*

Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), volume 1, pages 739–746. IEEE, 2005. l, lxxv, 53, 1110, 1286, 1295, 1296, 1298, 1299, 1334, 1370

[KKMO07] Subhash Khot, Guy Kindler, Elchanan Mossel, and Ryan O'Donnell. Optimal inapproximability results for max-cut and other 2-variable csps? *SIAM Journal on Computing*, 37(1):319–357, 2007. 1269

[KL11] J. Kelner and A. Levin. Spectral sparsification in the semi-streaming setting. In *Symposium on Theoretical Aspects of Computer Science (STACS)*, 2011. 1310, 1779, 2094

[KLC+15] Eunwoo Kim, Minsik Lee, Chong-Ho Choi, Nojun Kwak, and Songhwai Oh. Efficient-norm-based low-rank matrix approximations for large-scale problems using alternating rectified gradient method. *IEEE transactions on neural networks and learning systems*, 26(2):237–251, 2015. 53, 1110, 1370

[Kle99] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *J. ACM*, 46(5):604–632, 1999. 48, 1036

[KLM+14a] Michael Kapralov, Yin Tat Lee, Cameron Musco, Christopher Musco, and Aaron Sidford. Single pass spectral sparsification in dynamic streams. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 561–570. IEEE, https://arxiv.org/pdf/1407.1289, 2014. 1310, 1779

[KLM+14b] Raimondas Kiveris, Silvio Lattanzi, Vahab Mirrokni, Vibhor Rastogi, and Sergei Vassilvitskii. Connected components in mapreduce and beyond. In *Proceedings of the ACM Symposium on Cloud Computing*, pages 1–13. ACM, 2014. 2324

[KLM+17] Michael Kapralov, Yin Tat Lee, CN Musco, CP Musco, and Aaron Sidford. Single pass spectral sparsification in dynamic streams. *SIAM Journal on Computing*, 46(1):456–477, 2017. 2094

[KLP+16] Rasmus Kyng, Yin Tat Lee, Richard Peng, Sushant Sachdeva, and Daniel A Spielman. Sparsified cholesky and multigrid solvers for connection laplacians. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 842–850. ACM, 2016. 6

[KLS19] Rasmus Kyng, Kyle Luh, and Zhao Song. Four deviations suffice for rank 1 matrices. In *Manuscript*. https://arxiv.org/pdf/1901.06731, 2019. xxvi, 33, 1913, 2513

[KLT+16] Kuldeep Kulkarni, Suhas Lohit, Pavan Turaga, Ronan Kerviche, and Amit Ashok. Reconnet: Non-iterative reconstruction of images from compressively sensed measurements. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 449–458, 2016. 988

[KM93] Eyal Kushilevitz and Yishay Mansour. Learning decision trees using the Fourier spectrum. *SIAM Journal on Computing*, 22(6):1331–1348, 1993. 38, 919

[KM09] Jonathan Kelner and Aleksander Madry. Faster generation of random spanning trees. In *Proceedings of the 50th annual Symposium on Foundations of Computer Science (FOCS)*, pages 13–21, 2009. Available at https://arxiv.org/abs/0908.1448. 29, 550

[KM11] Tamara G Kolda and Jackson R Mayo. Shifted power method for computing tensor eigenpairs. *SIAM Journal on Matrix Analysis and Applications*, 32(4):1095–1124, 2011. 58, 1469

[KMN+02] Tapas Kanungo, David M Mount, Nathan S Netanyahu, Christine D Piatko, Ruth Silverman, and Angela Y Wu. A local search approximation algorithm for k-means clustering. In *Proceedings of the eighteenth annual symposium on Computational geometry*, pages 10–18. ACM, 2002. 2093

[KMP10] Ioannis Koutis, Gary L Miller, and Richard Peng. Approaching optimality for solving sdd linear systems. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*. IEEE, 2010. 6

[KMP11] Ioannis Koutis, Gary L Miller, and Richard Peng. A nearly-m log n time solver for sdd linear systems. In *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, pages 590–598. IEEE, 2011. 6

[KN14] Daniel M Kane and Jelani Nelson. Sparser johnson-lindenstrauss transforms. In *Journal of the ACM (JACM)*, volume 61(1), page 4. https://arxiv.org/pdf/1012.1577, 2014. 1781, 1786, 1791, 2076, 2079

[KNO18] Valentin Khrulkov, Alexander Novikov, and Ivan Oseledets. Expressive power of recurrent neural networks. In *International Conference on Learning Representations*, 2018. 22, 352

[KNPW11] Daniel M Kane, Jelani Nelson, Ely Porat, and David P Woodruff. Fast moment estimation in data streams in optimal space. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 745–754. ACM, 2011. 2029

[Knu69] Donald E Knuth. The art of computer programming. vol. 2: Seminumerical algorithms. addisonwesley. *Reading, MA*, pages 229–279, 1969. 1807, 1819, 1820

[Knu72] D. F. Knuth. Selected combinatorial research problems. *Computer Science Department, Stanford University*, 1972. 1915, 2191

[Knu98] Donald E. Knuth. The art of computer programming, vol. 2 : seminumerical algorithms, 1998. 59, 1471

[KNW10a] Daniel M Kane, Jelani Nelson, and David P Woodruff. On the exact space complexity of sketching and streaming small norms. In *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*, pages 1161–1178. Society for Industrial and Applied Mathematics, 2010. 1303, 1304, 1305, 1307, 1308

[KNW10b] Daniel M Kane, Jelani Nelson, and David P Woodruff. On the exact space complexity of sketching and streaming small norms. In *Proceedings of the*

twenty-first annual ACM-SIAM symposium on Discrete Algorithms, pages 1161–1178. SIAM, 2010. 2030, 2051, 2069

[Ko83] Ker-I Ko. On the computational complexity of ordinary differential equations. *Information and control*, 58(1-3):157–194, 1983. 20, 177

[Ko10] Ker-I Ko. Polynomial-time computability in analysis: A survey. 2010. 20, 177

[Koe00] Roger Koenker. Galton, edgeworth, frisch, and prospects for quantile regression in econometrics. *Journal of Econometrics*, 95(2):347–374, 2000. 14, 74

[Koe05] Roger Koenker. *Quantile Regression*. Cambridge University Press, 2005. 14, 74

[KOSZ13] Jonathan A Kelner, Lorenzo Orecchia, Aaron Sidford, and Zeyuan Allen Zhu. A simple, combinatorial algorithm for solving sdd systems in nearly-linear time. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 911–920. ACM, 2013. 6

[Kow75] Sophie Kowalevsky. Zur theorie der partiellen differentialgleichungen. In *Journal für die reine und angewandte Mathematik*, 80, pages 1–32. (German spelling of her lastname used at that time), 1875. 233

[KPHF12] U. Kang, Evangelos E. Papalexakis, Abhay Harpale, and Christos Faloutsos. Gigatensor: scaling tensor analysis up by 100 times - algorithms and discoveries. In *KDD*, pages 316–324, 2012. 1797

[KPPS17] Rasmus Kyng, Jakub Pachocki, Richard Peng, and Sushant Sachdeva. A framework for analyzing resparsification algorithms. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2032–2043. SIAM, 2017. 26, 549, 554

[KPSZ18] Rasmus Kyng, Richard Peng, Robert Schwieterman, and Peng Zhang. Incomplete nested dissection. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 404–417. ACM, 2018. 6

[Kro83] Pieter M Kroonenberg. *Three-mode principal component analysis: Theory and applications*, volume 2. DSWO press, 1983. 57, 233, 1469

[KS96] David R Karger and Clifford Stein. A new approach to the minimum cut problem. *Journal of the ACM (JACM)*, 43(4):601–640, 1996. 555, 556

[KS08] Tamara G Kolda and Jimeng Sun. Scalable tensor decompositions for multi-aspect data mining. In *Eighth IEEE International Conference on Data Mining (ICDM)*, pages 363–372. IEEE, 2008. 57, 1469

[KS09] Adam R Klivans and Alexander A Sherstov. Cryptographic hardness for learning intersections of halfspaces. *Journal of Computer and System Sciences*, 75(1):2–12, 2009. 258

[KS16] Rasmus Kyng and Sushant Sachdeva. Approximate gaussian elimination for laplacians-fast, sparse, and simple. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 573–582. IEEE, 2016. 26, 6, 548, 554, 559

[KS18] Rasmus Kyng and Zhao Song. A matrix chernoff bound for strongly rayleigh distributions and spectral sparsifiers from a few random spanning trees. In *FOCS*. https://arxiv.org/pdf/1810.08345, 2018. xxvi, 5, 32, 1911, 2141, 2513

[KS19] Michal Koucky and Michael Saks. Constant factor approximations to edit distance on far input pairs in nearly linear time. Manuscript, 2019. 2193

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 20, 22, 250, 251, 351

[KSS10] Amit Kumar, Yogish Sabharwal, and Sandeep Sen. Linear-time approximation schemes for clustering problems in any dimensions. *Journal of the ACM (JACM)*, 57(2):5, 2010. 2094

[KST54] Tamás Kovári, Vera Sós, and Pál Turán. On a problem of k. zarankiewicz. In *Colloquium Mathematicum*, volume 1, pages 50–57, 1954. 2284

[KSV08] Ravindran Kannan, Hadi Salmasian, and Santosh Vempala. The spectral method for general mixture models. *SIAM J. Comput.*, 38(3):1141–1156, 2008. 48, 1036

[KSV10] Howard Karloff, Siddharth Suri, and Sergei Vassilvitskii. A model of computation for mapreduce. In *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*, pages 938–948. Society for Industrial and Applied Mathematics, 2010. 2315, 2324, 2328

[KSV13] Peter Keevash, Benny Sudakov, and Jacques Verstraëte. On a conjecture of erdős and simonovits: Even cycles. *Combinatorica*, 33(6):699–732, 2013. 2284

[KSW16] Karin Knudson, Rayan Saab, and Rachel Ward. One-bit compressive sensing with norm estimation. *IEEE Transactions on Information Theory*, 62(5):2748–2758, 2016. 636

[KSZC03] Balachander Krishnamurthy, Subhabrata Sen, Yin Zhang, and Yan Chen. Sketch-based change detection: methods, evaluation, and applications. In *Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*, pages 234–247. ACM, 2003. 33, 34, 617, 618

[Kul90] Vidyadhar G. Kulkarni. Generating random combinatorial objects. *Journal of Algorithms*, 11(2):185–207, 1990. 29, 550

[KV09] Ravi Kannan and Santosh Vempala. Spectral algorithms. *Foundations and Trends in Theoretical Computer Science*, 4(3-4):157–288, 2009. 53, 1110

[KV17] Ravindran Kannan and Santosh Vempala. Randomized algorithms in numerical linear algebra. *Acta Numerica*, 26:95–135, 2017. 2043

[KVW14] Ravindran Kannan, Santosh S Vempala, and David P Woodruff. Principal component analysis and higher correlations for distributed data. In *Proceedings of The 27th Conference on Learning Theory (COLT)*, pages 1040–1057, 2014. 1318, 1781, 1784, 1786, 1788, 1789, 1791

[KVZ19] Michael Kapralov, Ameya Velingker, and Amir Zandieh. Dimension-independent sparse Fourier transform. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2709–2728. SIAM, 2019. lxvii, lxviii, 37, 39, 623, 919, 923, 925, 1902

[KW11] Felix Krahmer and Rachel Ward. New and improved Johnson-Lindenstrauss embeddings via the restricted isometry property. *SIAM Journal on Mathematical Analysis*, 43(3):1269–1281, 2011. 1922

[Kwa08] Nojun Kwak. Principal component analysis based on ℓ_1-norm maximization. *IEEE transactions on pattern analysis and machine intelligence*, 30(9):1672–1680, 2008. l, lxxv, 53, 1110, 1286, 1291, 1298, 1299, 1370

[KYFD15] Liwei Kuang, Laurence Yang, Jun Feng, and Mianxiong Dong. Secure tensor decomposition using fully homomorphic encryption scheme. *IEEE Transactions on Cloud Computing*, 2015. 57, 1469

[LA03] Wu-Sheng Lu and Andreas Antoniou. New method for weighted low-rank approximation of complex-valued matrices and its application for the design of 2-d digital filters. In *ISCAS*, 2003. 51, 1038

[LAKD08] Song Li, Shaikh S. Ahmed, Gerhard Klimeck, and Eric Darve. Computing entries of the inverse of a sparse matrix using the FIND algorithm. *J. Comput. Physics*, 227(22):9408–9427, 2008. 1925

[Lan06] J Landsberg. The border rank of the multiplication of $2\times$ 2 matrices is seven.

In *Journal of the American Mathematical Society*, volume 19(2), pages 447–459, 2006. 59, 1471

[Lan12] Joseph M Landsberg. *Tensors: geometry and applications*, volume 128. American Mathematical Society Providence, RI, USA., http://www.math.tamu.edu/~joseph.landsberg/Tbookintro.pdf, 2012. 59, 1471

[Lat97] Rafal Latala. Estimation of moments of sums of independent real random variables. *The Annals of Probability*, pages 1502–1513, 1997. 1390

[LBKW14] Yingyu Liang, Maria-Florina Balcan, Vandana Kanchanapally, and David P. Woodruff. Improved distributed principal component analysis. In *NIPS*, 2014. 48, 1036

[LBN+17] Jaehoon Lee, Yasaman Bahri, Roman Novak, Samuel S. Schoenholz, Jeffrey Pennington, and Jascha Sohl-Dickstein. Deep neural networks as Gaussian processes. *arXiv:1711.00165*, 2017. 358

[LDFU13] Yichao Lu, Paramveer Dhillon, Dean P Foster, and Lyle Ungar. Faster ridge regression via the subsampled randomized hadamard transform. In *Advances in neural information processing systems*, pages 369–377, 2013. 164, 165, 168, 1907, 1922, 1937

[LDP07a] Michael Lustig, David Donoho, and John M Pauly. Sparse MRI: The application of compressed sensing for rapid MR imaging. *Magnetic resonance in medicine*, 58(6):1182–1195, 2007. 33, 617

[LDP07b] Michael Lustig, David Donoho, and John M Pauly. Sparse mri: The application of compressed sensing for rapid mr imaging. *Magnetic resonance in medicine*, 58(6):1182–1195, 2007. 988

[LDSP08] Michael Lustig, David L Donoho, Juan M Santos, and John M Pauly. Compressed sensing MRI. *IEEE signal processing magazine*, 25(2):72–82, 2008. 621

[LDW+18] Xiaotong Lu, Weisheng Dong, Peiyao Wang, Guangming Shi, and Xuemei Xie. Convcsnet: A convolutional compressive sensing framework based on deep learning. *arXiv preprint arXiv:1801.10342*, 2018. 988

[Lee12] Jeff Leek. Prediction: the lasso vs. just using the top 10 predictors. http://simplystatistics.org/2012/02/23/prediction-the-lasso-vs-just-using-the-top-10/, 2012. 1925

[Lee17] Yin Tat Lee. Uniform sampling and inverse maintenance. In *Talk at Michael Cohen Memorial Symposium*. Available at: https://simons.berkeley.edu/talks/welcome-andbirds-eye-view-michaels-work., 2017. 119

[Lee18] Yin Tat Lee. Personal communication. *UW*, 2018. 1909, 2140

[Lez98] Pascal Lezaud. Chernoff-type bound for finite Markov chains. *Annals of Applied Probability*, pages 849–867, 1998. 26, 502

[LFC+16] Canyi Lu, Jiashi Feng, Yudong Chen, Wei Liu, Zhouchen Lin, and Shuicheng Yan. Tensor robust principal component analysis: Exact recovery of corrupted low-rank tensors via convex optimization. In *Proceedings of the IEEE*

Conference on Computer Vision and Pattern Recognition (CVPR), pages 5249–5257, 2016. 57, 58, 1469, 1470

[LG14] François Le Gall. Powers of tensors and fast matrix multiplication. In *Proceedings of the 39th international symposium on symbolic and algebraic computation(ISSAC)*, pages 296–303. ACM, 2014. 12, 4, 75, 89, 158, 159, 1426, 1468, 1921, 2466

[LGU18] Francois Le Gall and Florent Urrutia. Improved rectangular matrix multiplication using powers of the coppersmith-winograd tensor. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms(SODA)*, pages 1029–1046. SIAM, 2018. 15, 67, 75, 89, 158, 159

[LHP+15] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015. 20, 250, 351

[LHW17] Xingguo Li, Jarvis Haupt, and David Woodruff. Near optimal sketching of low-rank tensor regression. In *Advances in Neural Information Processing Systems*, pages 3466–3476, 2017. 2025, 2027

[Lib13] Edo Liberty. Simple and deterministic matrix sketching. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD)*, pages 581–588. ACM, 2013. 1310, 1779

[Lin66] Joram Lindenstrauss. A short proof of Liapounoff's convexity theorem. *J. Math. Mech.*, 15:971–972, 1966. 593

[LJCJ17] Lihua Lei, Cheng Ju, Jianbo Chen, and Michael I Jordan. Non-convex finite-sum optimization via scsg methods. In *Advances in Neural Information Processing Systems*, pages 2348–2358, 2017. 15, 77

[LL18] Yuanzhi Li and Yingyu Liang. Learning overparameterized neural networks via stochastic gradient descent on structured data. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018. 21, 251, 257, 259, 273, 351, 1909, 2140

[Llo82] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982. 2093

[LLR16] Yuanzhi Li, Yingyu Liang, and Andrej Risteski. Recovery guarantee of weighted low-rank approximation via alternating minimization. *CoRR*, abs/1602.02262, 2016. 51, 1038

[LM00] Beatrice Laurent and Pascal Massart. Adaptive estimation of a quadratic functional by model selection. *Annals of Statistics*, pages 1302–1338, 2000. 390, 2506

[LM15] Shachar Lovett and Raghu Meka. Constructive discrepancy minimization by walking on the edges. In *SIAM Journal on Computing (A preliminary version of paper appeared in FOCS 2012)*, volume 44(5), pages 1573–1582. https://arxiv.org/pdf/1203.5747, 2015. 596

[LMH15] Hongzhou Lin, Julien Mairal, and Zaid Harchaoui. A universal catalyst for first-order optimization. In *Advances in Neural Information Processing Systems*, pages 3384–3392, 2015. 15, 77

[LMH17] Hongzhou Lin, Julien Mairal, and Zaid Harchaoui. Catalyst acceleration for first-order convex optimization: from theory to practice. *arXiv preprint arXiv:1712.05654*, 2017. 15, 77

[LMP13] Mu Li, Gary L. Miller, and Richard Peng. Iterative row sampling. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 127–136. https://arxiv.org/pdf/1211.2713, 2013. 52, 1109, 1926, 1966

[LMS98] Gad M Landau, Eugene W Myers, and Jeanette P Schmidt. Incremental string comparison. *SIAM Journal on Computing*, 27(2):557–582, 1998. 1914, 2214, 2215, 2221

[LMS11] Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Lower bounds based on the exponential time hypothesis. In *Bull. EATCS 105*, pages 41–72, 2011. 1671

[LMSV11] Silvio Lattanzi, Benjamin Moseley, Siddharth Suri, and Sergei Vassilvitskii. Filtering: a method for solving graph problems in MapReduce. In *Proceedings of the 23rd ACM symposium on Parallelism in algorithms and architectures*, pages 85–94. ACM, 2011. 2315, 2316

[LMV00a] Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle. A multilinear singular value decomposition. *SIAM J. Matrix Analysis Applications*, 21(4):1253–1278, 2000. 57, 1469

[LMV00b] Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle. On the best rank-1 and rank-$(R_1, R_2, \cdots, R_n)$ approximation of higher-order tensors. *SIAM J. Matrix Analysis Applications*, 21(4):1324–1342, 2000. 57, 1469

[LMWY13] Ji Liu, Przemyslaw Musialski, Peter Wonka, and Jieping Ye. Tensor completion for estimating missing values in visual data. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(1):208–220, 2013. 57, 58, 1469, 1470

[LN18] Yi Li and Vasileios Nakos. Sublinear-time algorithms for compressive phase retrieval. In *ISIT*. arXiv preprint arXiv:1709.02917, 2018. 635

[LN19] Yi Li and Vasileios Nakos. Deterministic sparse Fourier transform with an ℓ_∞ guarantee. *arXiv preprint arXiv:1903.00995*, 2019. lxvii, lxviii, 37, 38, 919, 923, 926, 1904

[LNN15] Kasper Green Larsen, Jelani Nelson, and Huy L Nguyên. Time lower bounds for nonadaptive turnstile streaming algorithms. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 803–812. ACM, 2015. 621

[LNNT16] Kasper Green Larsen, Jelani Nelson, Huy L Nguyên, and Mikkel Thorup. Heavy hitters via cluster-preserving clustering. In *Foundations of Computer*

Science (FOCS), 2016 IEEE 57th Annual Symposium on, pages 61–70. IEEE, https://arxiv.org/pdf/1604.01357, 2016. 618, 620, 631, 1779, 2029, 2053

[LNRW19] J. Li, A. Nikolov, I. Razenshteyn, and E. Waingarten. On mean estimation for general norms with statistical queries. In *COLT*, 2019. 1967

[LNW18] Yi Li, Vasileios Nakos, and David P. Woodruff. On low-risk heavy hitters and sparse recovery schemes. In *RANDOM/APPROX*. arXiv preprint arXiv:1709.02919, 2018. 623

[LOG16] Wuchen Li, Stanley Osher, and Wilfrid Gangbo. Fast algorithm for earth mover's distance based on optimal transport and ℓ_1 type regularization i. *arXiv preprint arXiv:1609.07092*, 2016. 1262

[Loj63] S Lojasiewicz. A topological property of real analytic subsets. *Coll. du CNRS, Les équations aux dérivées partielles*, 117:87–89, 1963. 364

[LP04] C. A. Leon and F. Perron. Optimal Hoeffding bounds for discrete reversible Markov chains. *Annals of Applied Probability*, 14(2), 2004. 26, 502

[LPS88] Alexander Lubotzky, Ralph Phillips, and Peter Sarnak. Ramanujan graphs. *Combinatorica*, 8(3):261–277, 1988. 27, 503, 595

[LPS19] Yin Tat Lee, Swati Padmanabhan, and Zhao Song. Tbd. *Manuscript*, 2019. xxix

[LPSY18] David Liau, Eric Price, Zhao Song, and Ger Yang. Stochastic multi-armed bandits in constant space. In *AISTATS*, 2018. xxix

[LPW97] W.-S Lu, S.-C Pei, and P.-H Wang. Weighted low-rank approximation of general complex matrices and its application in the design of 2-d digital filters. In *IEEE Transactions on Circuits and Systems*, volume 44, pages 650–655, 1997. 51, 1038

[LPW09] David Asher Levin, Yuval Peres, and Elizabeth Lee Wilmer. *Markov chains and mixing times*. American Mathematical Soc., 2009. 1219

[LRHG13] Ben London, Theodoros Rekatsinas, Bert Huang, and Lise Getoor. Multi-relational learning using weighted tensor decomposition with modular loss. In *arXiv preprint*. https://arxiv.org/abs/1303.1733, 2013. 58, 1470

[LRSB12] Nicolas Le Roux, Mark W Schmidt, and Francis R Bach. A stochastic gradient method with an exponential convergence rate for finite training sets. In *NIPS*, pages 2672–2680, 2012. 13, 74

[LS90] László Lovász and Miklós Simonovits. The mixing rate of markov chains, an isoperimetric inequality, and computing the volume. In *31st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 346–354. IEEE, 1990. 16, 171

[LS92] László Lovász and Miklós Simonovits. On the randomized complexity of volume and diameter. In *33rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 482–492. IEEE, 1992. 16, 171

[LS93] László Lovász and Miklós Simonovits. Random walks in a convex body and

an improved volume algorithm. *Random structures & algorithms*, 4(4):359–412, 1993. 16, 171

[LS00] Daniel D. Lee and H. Sebastian Seung. Algorithms for non-negative matrix factorization. In *NIPS*, 2000. 49, 1036

[LS13] Yin Tat Lee and Aaron Sidford. Path finding I: Solving linear programs with $\widetilde{O}(\sqrt{rank})$ linear system solves. *arXiv preprint arXiv:1312.6677*, 2013. 7

[LS14] Yin Tat Lee and Aaron Sidford. Path finding methods for linear programming: Solving linear programs in $O(\sqrt{rank})$ iterations and faster algorithms for maximum flow. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 424–433. IEEE, 2014. 10, 11, 13, 3, 5, 15, 21, 1895

[LS15] Yin Tat Lee and Aaron Sidford. Efficient inverse maintenance and faster algorithms for linear programming. In *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*, pages 230–249. IEEE, 2015. 10, 13, 3, 15, 1046, 1895

[LS17] Yin Tat Lee and He Sun. An sdp-based algorithm for linear-sized spectral sparsification. In *Proceedings of the 49th annual acm sigact symposium on theory of computing*, pages 678–687. ACM, 2017. 29, 596

[LS18] Yin Tat Lee and He Sun. Constructing linear-sized spectral sparsification in almost-linear time. *SIAM Journal on Computing (A Preliminary version of this paper appeared in FOCS 15)*, 47(6):2315–2336, 2018. 29, 596

[LSS19] Jerry Li, Ruoqi Shen, and Zhao Song. Tbd. *Manuscript*, 2019. xxix, 2308, 2515

[LSSS14] Roi Livni, Shai Shalev-Shwartz, and Ohad Shamir. On the computational efficiency of training neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 855–863, 2014. 258

[LSV18] Yin Tat Lee, Zhao Song, and Santosh S Vempala. Algorithmic theory of odes and sampling from well-conditioned logconcave densities. *arXiv preprint arXiv:1812.06243*, 2018. xxv, 18, 24, 174, 1897, 2512

[LSW15] Yin Tat Lee, Aaron Sidford, and Sam Chiu-wai Wong. A faster cutting plane method and its implications for combinatorial and convex optimization. In *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*, pages 1049–1065. IEEE, 2015. 21

[LSW+19] Yingyu Liang, Zhao Song, Mengdi Wang, Lin Yang, and Xin Yang. Sketching transformed matrices with applications to natural language processing. *Manuscript*, 2019. xxix

[LSY19] Yibo Lin, Zhao Song, and Lin F Yang. Towards a theoretical understanding of hashing-based neural nets. In *AISTATS*, 2019. xxix

[LSZ19] Yin Tat Lee, Zhao Song, and Qiuyi Zhang. Solving empirical risk minimization in the current matrix multiplication time. In *COLT*. https://arxiv.org/pdf/1905.04447.pdf, 2019. xxiv, 3, 8, 24, 1896, 2512, 2513

[LT13] Michel Ledoux and Michel Talagrand. *Probability in Banach Spaces: isoperimetry and processes*. Springer Science & Business Media, 2013. 511

[LV06a] László Lovász and Santosh Vempala. Fast algorithms for logconcave functions: Sampling, rounding, integration and optimization. In *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS).*, pages 57–68. IEEE, 2006. 16, 171

[LV06b] László Lovász and Santosh Vempala. Hit-and-run from a corner. *SIAM Journal on Computing*, 35(4):985–1005, 2006. 16, 17, 18, 171, 174

[LV13] Xiaodong Li and Vladislav Voroninski. Sparse signal recovery from quadratic measurements via convex programming. *SIAM Journal on Mathematical Analysis*, 45(5):3019–3033, 2013. 635

[LV17] Yin Tat Lee and Santosh S Vempala. Geodesic walks in polytopes. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 927–940. ACM, https://arxiv.org/pdf/1606.04696.pdf, 2017. 19, 177, 182, 197

[LV18] Yin Tat Lee and Santosh S Vempala. Convergence rate of riemannian hamiltonian monte carlo and faster polytope volume computation. In *STOC*. https//arxiv.org/pdf/1710.06261.pdf, 2018. 19, 173, 182

[LWC13] David Lawlor, Yang Wang, and Andrew Christlieb. Adaptive sub-linear time Fourier algorithms. *Advances in Adaptive Data Analysis*, 5(01):1350003, 2013. 38, 919

[LY17] Yuanzhi Li and Yang Yuan. Convergence analysis of two-layer neural networks with ReLU activation. In *Advances in Neural Information Processing Systems (NeurIPS)*. http://arxiv.org/abs/1705.09886, 2017. 21, 250, 257, 351, 2140

[Lya40] AA Lyapunov. On completely additive vector functions. *Izv. Akad. Nauk SSSR*, 4:465–478, 1940. 593

[LZBJ14] Tao Lei, Yuan Zhang, Regina Barzilay, and Tommi Jaakkola. Low-rank tensors for scoring dependency structures. In *Association for Computational Linguistics(ACL), Best student paper award*, 2014. 57, 1469

[LZMB15] Tao Lei, Yuan Zhang, Alessandro Moschitti, and Regina Barzilay. High-order low-rank tensors for semantic role labeling. In *In Proceedings of the 2015 Conference of the North America Chapter of the Association For Computational Linguistics–Human Language Technologies (NAACLHLT 2015*. Citeseer, 2015. 57, 1469

[LZWW16] Gui-Fu Lu, Jian Zou, Yong Wang, and Zhongqun Wang. L1-norm-based principal component analysis with adaptive regularization. *Pattern Recognition*, 60:901–907, 2016. 1334

[Mad13] Aleksander Madry. Navigating central path with electrical flows: From flows to matchings, and back. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 253–262. IEEE, 2013. 11, 12, 3, 4, 1088, 1089, 1094, 1095, 1100, 1131, 1506

[Mad16] Aleksander Madry. Computing maximum flow with augmenting electrical flows. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 593–602. IEEE, 2016. 11, 12, 3, 4

[Mah11a] M. W. Mahoney. Randomized algorithms for matrices and data. *Foundations and Trends® in Machine Learning*, 3(2):123–224, 2011. 1974

[Mah11b] Michael W. Mahoney. Randomized algorithms for matrices and data. *Foundations and Trends in Machine Learning*, 3(2):123–224, 2011. 2025

[Man92] Yishay Mansour. Randomized interpolation and approximation of sparse polynomials stpreliminary version. In *International Colloquium on Automata, Languages, and Programming*, pages 261–272. Springer, 1992. 38, 919

[Mar73] GA Margulis. Explicit construction of expanders. *Problemy Peredaci Informacii*, 9(4):71–80, 1973. 595

[Mas69] James L Massey. Shift-register synthesis and bch decoding. *Information Theory, IEEE Transactions on*, 15(1):122–127, 1969. 46, 759, 763

[Mau03] Andreas Maurer. A bound on the deviation probability for sums of non-negative random variables. *J. Inequalities in Pure and Applied Mathematics*, 4(1):15, 2003. 1389

[MB11] Eric Moulines and Francis R Bach. Non-asymptotic analysis of stochastic approximation algorithms for machine learning. In *Advances in Neural Information Processing Systems*, pages 451–459, 2011. 13, 74

[MBZ10] Sergio V Macua, Pavle Belanovic, and Santiago Zazo. Consensus-based distributed principal component analysis in wireless sensor networks. In *Signal Processing Advances in Wireless Communications (SPAWC), 2010 IEEE Eleventh International Workshop on*, pages 1–5. IEEE, 2010. 1318, 1784, 1788

[McG09] Andrew McGregor. Graph mining on streams. *Encyclopedia of Database Systems*, pages 1271–1275, 2009. 2316

[Meg12] Nimrod Megiddo. *Progress in Mathematical Programming: Interior-Point and Related Methods*. Springer Science & Business Media, 2012. 6

[Mek04] Raghu Meka. discrepancy-and-beating-the-union-bound/. 2004. 1912

[MES08] Julien Mairal, Michael Elad, and Guillermo Sapiro. Sparse representation for color image restoration. *Image Processing, IEEE Transactions on*, 17(1):53–69, 2008. 51, 1038

[MH09] Morten Mørup and Lars Kai Hansen. Sparse coding and automatic relevance determination for multi-way models. In *SPARS'09-Signal Processing with Adaptive Sparse Structured Representations*, 2009. 1706

[MHG15] Cun Mu, Daniel Hsu, and Donald Goldfarb. Successive rank-one approximations for nearly orthogonally decomposable symmetric tensors. *SIAM Journal on Matrix Analysis and Applications*, 36(4):1638–1659, 2015. 58, 1469

[MHWG14] Cun Mu, Bo Huang, John Wright, and Donald Goldfarb. Square deal: Lower bounds and improved relaxations for tensor recovery. In *The Thirty-first*

International Conference on Machine Learning (ICML), pages 73–81. https://arxiv.org/pdf/1307.5870, 2014. 58, 1470

[MKB+10] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Eleventh Annual Conference of the International Speech Communication Association*, 2010. 22, 352

[MKB+11] Tomáš Mikolov, Stefan Kombrink, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. Extensions of recurrent neural network language model. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 5528–5531. IEEE, 2011. 22, 352

[MKCP16] P. P. Markopoulos, S. Kundu, S. Chamadia, and D. A. Pados. Efficient ℓ_1-Norm Principal-Component Analysis via Bit Flipping. *ArXiv e-prints*, 2016. 53, 1110, 1370

[MKP13] Panos P. Markopoulos, George N. Karystinos, and Dimitrios A. Pados. Some options for ℓ_1-subspace signal processing. In *ISWCS 2013, The Tenth International Symposium on Wireless Communication Systems, Ilmenau, TU Ilmenau, Germany, August 27-30, 2013*, pages 1–5, 2013. 53, 1110, 1370

[MKP14] Panos P. Markopoulos, George N. Karystinos, and Dimitrios A. Pados. Optimal algorithms for ℓ_1-subspace signal processing. *IEEE Trans. Signal Processing*, 62(19):5046–5058, 2014. 53, 1110, 1370

[MLF15] Zhuang Ma, Yichao Lu, and Dean Foster. Finding linear structure in large datasets with scalable canonical correlation analysis. In *International Conference on Machine Learning*, pages 169–178, 2015. 15, 77

[MM13] Xiangrui Meng and Michael W Mahoney. Low-distortion subspace embeddings in input-sparsity time and applications to robust linear regression. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 91–100. ACM, https://arxiv.org/pdf/1210.3135, 2013. 49, 52, 1037, 1109, 1119, 1134, 1185, 1198, 1253, 1254, 1260, 1336, 1337, 1338, 1341, 1344, 1345, 1370, 1401, 1426, 1468, 1489, 1516, 1535, 1717, 1922, 1966, 1971, 2027, 2029, 2030, 2031, 2032

[MMD08] Michael W Mahoney, Mauro Maggioni, and Petros Drineas. Tensor-cur decompositions for tensor-based data. *SIAM Journal on Matrix Analysis and Applications*, 30(3):957–987, 2008. 58, 1469, 1470

[MMN18] Song Mei, Andrea Montanari, and Phan-Minh Nguyen. A mean field view of the landscape of two-layer neural networks. *Proceedings of the National Academy of Sciences*, 115(33):E7665–E7671, 2018. 358

[MMSW15] Konstantin Makarychev, Yury Makarychev, Maxim Sviridenko, and Justin Ward. A bi-criteria approximation algorithm for k means. *arXiv preprint arXiv:1507.04227*, 2015. 1474

[MNV09] Meena Mahajan, Prajakta Nimbhorkar, and Kasturi Varadarajan. The planar k-means problem is np-hard. In *International Workshop on Algorithms and Computation*, pages 274–285. Springer, 2009. 2093

[Moi13] Ankur Moitra. An almost optimal algorithm for computing nonnegative rank. In *SODA*, 2013. 1044, 1046, 1050, 1899

[Moi14a] Ankur Moitra. *Algorithmic Aspects of Machine Learning*. Cambridge University Press, 2014. 58, 1470, 2094

[Moi14b] Ankur Moitra. Tensor decompositions and their applications, 2014. 1797

[Moi15] Ankur Moitra. The threshold for super-resolution via extremal functions. In *STOC*, 2015. lxxi, 9, 40, 44, 46, 47, 622, 678, 683, 687, 759, 762, 764

[Mør11] Morten Mørup. Applications of tensor (multiway array) factorizations and decompositions in data mining. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1(1):24–40, 2011. 57, 1469

[Mou10] Nima Mousavi. How tight is chernoff bound, 2010. 2506, 2507

[MP80] William J. Masek and Mike Paterson. A faster algorithm computing string edit distances. *J. Comput. Syst. Sci.*, 20(1):18–31, 1980. 1915, 2191, 2214

[MP14] Gregory T Minton and Eric Price. Improved concentration bounds for count-sketch. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 669–686. Society for Industrial and Applied Mathematics, 2014. 793, 794

[MPB15] Ali Mousavi, Ankit B Patel, and Richard G Baraniuk. A deep learning approach to structured signal recovery. In *Communication, Control, and Computing (Allerton), 2015 53rd Annual Allerton Conference on*, pages 1336–1343. IEEE, 2015. 988

[MPS14] Andrew M McDonald, Massimiliano Pontil, and Dimitris Stamos. Spectral k-support norm regularization. In *Advances in Neural Information Processing Systems*, pages 3644–3652, 2014. 1968, 1971, 1984

[MR05] Elchanan Mossel and Sébastien Roch. Learning nonsingular phylogenies and hidden markov models. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing (STOC)*, pages 366–375. ACM, https://arxiv.org/pdf/cs/0502076, 2005. 57, 1469

[MR10] Dana Moshkovitz and Ran Raz. Two-query pcp with subconstant error. In *Journal of the ACM (JACM)*, volume 57(5), page 29. A preliminary version appeared in the Proceedings of The 49th Annual IEEE Symposium on Foundations of Computer Science (FOCS 08), FOCS 08 Best paper award, https://eccc.weizmann.ac.il/eccc-reports/2008/TR08-071/, 2010. 1670, 1671, 1702

[MR18] Pasin Manurangsi and Daniel Reichman. The computational complexity of training ReLU(s). *arXiv preprint arXiv:1810.04207*, 2018. 258

[MS17] Tomoya Murata and Taiji Suzuki. Doubly accelerated stochastic variance reduced dual averaging method for regularized empirical risk minimization. In *Advances in Neural Information Processing Systems*, pages 608–617, 2017. 13, 15, 18, 19, 74, 77, 173, 174

[MSS15a] Adam W. Marcus, Daniel A. Spielman, and Nikhil Srivastava. Interlacing families I: Bipartite Ramanujan graphs of all degrees. *Ann. of Math. (2)*, 182(1):307–325, 2015. 595, 597, 599, 600

[MSS15b] Adam W. Marcus, Daniel A. Spielman, and Nikhil Srivastava. Interlacing families II: Mixed characteristic polynomials and the Kadison-Singer problem. *Ann. of Math. (2)*, 182(1):327–350, 2015. 31, 554, 563, 591, 594, 597, 601, 607, 1912, 1913

[MSS16] Tengyu Ma, Jonathan Shi, and David Steurer. Polynomial-time tensor decompositions with sum-of-squares. In *Foundations of Computer Science (FOCS), 2016 IEEE 57th Annual Symposium on*, pages 438–446. IEEE, https://arxiv.org/pdf/1610.01980, 2016. 58, 1470

[MSS18] A. Marcus, D. Spielman, and N. Srivastava. Interlacing families IV: Bipartite ramanujan graphs of all sizes. *SIAM Journal on Computing*, 47(6):2488–2509, 2018. 595

[MST15] Aleksander Madry, Damian Straszak, and Jakub Tarnawski. Fast generation of random spanning trees and the effective resistance metric. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2019–2036, 2015. Available at http://arxiv.org/pdf/1501.00267v1.pdf. 29, 550

[MT18] Jakub Marecek and Tigran Tchrakian. Robust spectral filtering and anomaly detection. *arXiv preprint arXiv:1808.01181*, 2018. 257

[Mut05a] S. Muthukrishnan. Data Streams: Algorithms and Applications. *Foundations and Trends in Theoretical Computer Science*, 1(2):117–236, 2005. 33, 617

[Mut05b] Shanmugavelayutham Muthukrishnan. Data streams: Algorithms and applications. *Foundations and Trends® in Theoretical Computer Science*, 1(2):117–236, 2005. 2088

[MV18] Oren Mangoubi and Nisheeth K Vishnoi. Dimensionally tight running time bounds for second-order hamiltonian monte carlo. *arXiv preprint arXiv:1802.08898*, 2018. 18, 19, 173, 174

[MW10] Morteza Monemizadeh and David P Woodruff. 1-pass relative-error lp-sampling with applications. In *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*, pages 1143–1160. SIAM, 2010. 1568, 1593, 1799

[MW17] Cameron Musco and David P Woodruff. Sublinear time low-rank approximation of positive semidefinite matrices. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 672–683. IEEE, 2017. 2027

[MXZZ13] Deyu Meng, Zongben Xu, Lei Zhang, and Ji Zhao. A cyclic weighted median method for ℓ_1 low-rank matrix factorization with missing entries. In *AAAI*, volume 4, page 6, 2013. 53, 1110, 1370

[MZIC17] Sami Merhi, Ruochuan Zhang, Mark A Iwen, and Andrew Christlieb. A new class of fully discrete sparse Fourier transforms: Faster stable implementations with guarantees. *Journal of Fourier Analysis and Applications*, pages 1–34, 2017. 38, 919

[N+03] Yurii Nesterov et al. *Random walk in a simplex and quadratic optimization over convex polytopes*. CORE, 2003. 1673

[Nac10] Mergen Nachin. Lower bounds on the column sparsity of sparse recovery matrices. *UAP: MIT Undergraduate Thesis*, 2010. 622

[Nad64] Elizbar A Nadaraya. On estimating regression. *Theory of Probability & Its Applications*, 9(1):141–142, 1964. 14, 74

[Nak94] A.M. Nakhushev. Cauchy-kovaleskaya theorem. *Hazewinkel, Michiel, Encyclopedia of Mathematics*, 1994. 233

[Nak17a] Vasileios Nakos. Almost optimal phaseless compressed sensing with sublinear decoding time. In *2017 IEEE International Symposium on Information Theory, ISIT 2017, Aachen, Germany, June 25-30, 2017*, pages 1142–1146, 2017. 635

[Nak17b] Vasileios Nakos. On fast decoding of high-dimensional signals from one-bit measurements. In *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, pages 61:1–61:14, 2017. 636

[Nak19] Vasileios Nakos. One-bit expandersketch for one-bit compressed sensing. In *ISIT*. arXiv preprint arXiv:1711.04049, 2019. 636

[NDH+17] Tigran Nagapetyan, Andrew B Duncan, Leonard Hasenclever, Sebastian J Vollmer, Lukasz Szpruch, and Konstantinos Zygalakis. The true cost of

stochastic gradient langevin dynamics. *arXiv preprint arXiv:1706.02692*, 2017. 181

[NDTTJ18] Aleksandar Sasho Nikolov, Daniel Dadush, Kunal Talwar, and Nicole Tomczak-Jaegermann. Balancing vectors in any norm. FOCS, 2018. 596

[Nes83] Yurii E Nesterov. A method for solving the convex programming problem with convergence rate $O(1/k^2)$. In *Dokl. Akad. Nauk SSSR*, volume 269, pages 543–547, 1983. 13, 74

[Nes98] Yurii Nesterov. Introductory lectures on convex programming volume i: Basic course. *Lecture notes*, 1998. 15, 77, 81, 82, 160

[Nes04] Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2004. 13, 74, 267, 332

[NJ02] Andrew Y Ng and Michael I Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *Advances in neural information processing systems*, pages 841–848, 2002. 16, 171

[NJLS09] Arkadi Nemirovski, Anatoli Juditsky, Guanghui Lan, and Alexander Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on optimization*, 19(4):1574–1609, 2009. 13, 74

[NN89] Yu Nesterov and Arkadi Nemirovsky. Self-concordant functions and polynomial-time methods in convex programming. *Report, Central Economic and Mathematic Institute, USSR Acad. Sci*, 1989. 10, 3

[NN91] Yu Nesterov and Arkadi Nemirovsky. Acceleration and parallelization of the path-following interior point method for a linearly constrained convex quadratic problem. *SIAM Journal on Optimization*, 1(4):548–564, 1991. 15

[NN94] Yurii Nesterov and Arkadii Nemirovskii. *Interior-point polynomial algorithms in convex programming*, volume 13. Siam, 1994. 13, 15

[NN13a] Jelani Nelson and Huy L Nguyên. Osnap: Faster numerical linear algebra algorithms via sparser subspace embeddings. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 117–126. IEEE, https://arxiv.org/pdf/1211.1002, 2013. 49, 52, 13, 770, 1037, 1109, 1337, 1338, 1370, 1426, 1468, 1489, 1535, 1758, 1922, 1966, 2027, 2029

[NN13b] Jelani Nelson and Huy L Nguyên. Sparsity lower bounds for dimensionality reducing maps. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 101–110. ACM, 2013. 622

[NN14] Jelani Nelson and Huy L Nguyên. Lower bounds for oblivious subspace embeddings. In *International Colloquium on Automata, Languages, and Programming*, pages 883–894. Springer, 2014. 1242, 1922

[Nol07] John P Nolan. Stable distributions. 2007. 2030, 2054

[NPSS06] Assaf Naor, Yuval Peres, Oded Schramm, and Scott Sheffield. Markov chains in smooth banach spaces and gromov-hyperbolic metric spaces. *Duke Mathematical Journal*, 134(1):165–197, 2006. 509

[NS17] Yurii Nesterov and Sebastian U Stich. Efficiency of the accelerated coordinate descent method on structured optimization problems. *SIAM Journal on Optimization*, 27(1):110–123, 2017. 13, 74

[NS19] Vasileios Nakos and Zhao Song. Stronger L2/L2 compressed sensing; without iterating. In *Proceedings of the 51st Annual ACM Symposium on Theory of Computing (STOC)*, 2019. xxvi, 9, 35, 48, 2029, 2514

[NSW18] Vasileios Nakos, Zhao Song, and Zhengyu Wang. The power of (careful) iterative loop analysis in compressed sensing. In *manuscript*, 2018. xxvii, 48, 964, 1904, 2514

[NSW19a] Vasileios Nakos, Zhao Song, and Zhengyu Wang. (Nearly) sample-optimal sparse Fourier transform in any dimension; RIPless and Filterless. In *FOCS*, 2019. xxvii, 9, 37, 48, 164, 246, 1902, 1904, 2514

[NSW19b] Vasileios Nakos, Zhao Song, and Zhengyu Wang. Robust sparse recovery via m-estimators. *Manuscript*, 2019. 1967, 2514

[NSWZ18] Vasileios Nakos, Xiaofei Shi, David P. Woodruff, and Hongyang Zhang. Improved algorithms for adaptive compressed sensing. In *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, pages 90:1–90:14, 2018. 623

[NT09a] Deanna Needell and Joel A Tropp. Cosamp: Iterative signal recovery from incomplete and inaccurate samples. *Applied and computational harmonic analysis*, 26(3):301–321, 2009. lxvi, lxviii, 34, 35, 617, 624

[NT09b] Deanna Needell and Joel A Tropp. CoSaMP: Iterative signal recovery from incomplete and inaccurate samples. *Applied and computational harmonic analysis*, 26(3):301–321, 2009. 38, 918, 919, 925

[NV09] Deanna Needell and Roman Vershynin. Uniform uncertainty principle and signal recovery via regularized orthogonal matching pursuit. *Foundations of computational mathematics*, 9(3):317–334, 2009. 38, 918, 919

[NV10] D Needel and R Vershynin. Signal recovery from inaccurate and incomplete measurements via regularized orthogonal matching pursuit. *IEEE Journal of Selected Topics in Signal Processing*, pages 310–316, 2010. 38, 918

[NW14] Jelani Nelson and David P. Woodruff. Personal communication. ., 2014. 1563, 1566

[O'M08] Owen O'Malley. Terabyte sort on apache hadoop. *Yahoo Tech. Rep*, 2008. 2331

[OO18] Samet Oymak and Necmiye Ozay. Non-asymptotic identification of LTI systems from a single trajectory. *arXiv preprint arXiv:1806.05722*, 2018. 257

[OPS17] Junier B Oliva, Barnabás Póczos, and Jeff Schneider. The statistical recurrent unit. In *International Conference on Machine Learning (ICML)*, pages 2671–2680, 2017. 353

[OS14] Sewoong Oh and Devavrat Shah. Learning mixed multinomial logit model from ordinal data. In *Advances in Neural Information Processing Systems (NIPS)*, pages 595–603. https://arxiv.org/pdf/1411.0073, 2014. 57, 1469

[Ose11] Ivan V. Oseledets. Tensor-train decomposition. *SIAM J. Scientific Computing*, 33(5):2295–2317, 2011. 58, 1469, 1711

[OST08] Ivan V Oseledets, DV Savostianov, and Eugene E Tyrtyshnikov. Tucker dimensionality reduction of three-dimensional arrays in linear time. *SIAM Journal on Matrix Analysis and Applications*, 30(3):939–956, 2008. 58, 1470, 1476

[OT09] Ivan V Oseledets and Eugene E Tyrtyshnikov. Breaking the curse of dimensionality, or how to use svd in many dimensions. *SIAM Journal on Scientific Computing*, 31(5):3744–3759, 2009. 58, 1469, 2510

[OTZ11] Ivan Oseledets, Eugene Tyrtyshnikov, and Nickolai Zamarashkin. Tensor-train ranks for matrices and their inverses. *Computational Methods in Applied Mathematics Comput. Methods Appl. Math.*, 11(3):394–403, 2011. 1711

[OY05] S. Oh, S. Kwon and J. Yun. A method for structured linear total least norm on blind deconvolution problem. *Applied Mathematics and Computing*, 19:151–164, 2005. 2024

[OYDS11] Henrik Ohlsson, Allen Y Yang, Roy Dong, and S Shankar Sastry. Compressive phase retrieval from squared output measurements via semidefinite programming. *arXiv preprint arXiv:1111.6323*, pages 1–27, 2011. 635

[Oym18] Samet Oymak. Learning compact neural networks with regularization. *arXiv preprint arXiv:1802.01223*, 2018. 257

[Paa97] Pentti Paatero. A weighted non-negative least squares algorithm for three-way "parafac" factor analysis. *Chemometrics and Intelligent Laboratory Systems*, 38(2):223–242, 1997. 58, 1470

[Paa00] Pentti Paatero. Construction and analysis of degenerate parafac models. *Journal of chemometrics*, 14(3):285–299, 2000. 16, 57, 59, 171, 1469, 1471

[Pag13] Rasmus Pagh. Compressed matrix multiplication. *ACM Transactions on Computation Theory (TOCT)*, 5(3):9, 2013. 1482, 1486, 1518

[PB17] Jeffrey Pennington and Yasaman Bahri. Geometry of neural network loss surfaces via random matrix theory. In *International Conference on Machine Learning (ICML)*, International Convention Centre, Sydney, Australia, 2017. 358

[PBLJ15] Anastasia Podosinnikova, Francis Bach, and Simon Lacoste-Julien. Rethinking lda: moment matching for discrete ica. In *Advances in Neural Information Processing Systems(NIPS)*, pages 514–522. https://arxiv.org/pdf/1507.01784, 2015. 57, 1469

[PC08] Anh Phan and Andrzej Cichocki. Fast and efficient algorithms for nonnegative tucker decomposition. *Advances in Neural Networks-ISNN 2008*, pages 772–782, 2008. 1706

[Pee96] René Peeters. Orthogonal representations over finite fields and the chromatic number of graphs. *Combinatorica*, 16(3):417–431, 1996. 51, 1038

[Phi] Phillips. Compressed sense. https://www.philips.com/healthcare/resources/landing/compressed-sense. 36, 917

[Pin60] Mark S Pinsker. Information and information stability of random variables and processes. *San Francisco: Holden-Day, 1964, originally published in Russian in 1960*, 1960. 1219

[PJ92] Boris T Polyak and Anatoli B Juditsky. Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*, 30(4):838–855, 1992. 13, 74

[PK16] Young Woong Park and Diego Klabjan. Iteratively reweighted least squares algorithms for ℓ_1-norm principal component analysis. *arXiv preprint arXiv:1609.02997*, 2016. 53, 1110, 1370

[PKB14] Dimitris Papailiopoulos, Anastasios Kyrillidis, and Christos Boutsidis. Provable deterministic leverage score sampling. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 997–1006. ACM, 2014. 1926

[PLR+16] Ben Poole, Subhaneil Lahiri, Maithreyi Raghu, Jascha Sohl-Dickstein, and Surya Ganguli. Exponential expressivity in deep neural networks through transient chaos. In *Advances In Neural Information Processing Systems*, pages 3360–3368, 2016. 00047. 358

[PLY10] Yanwei Pang, Xuelong Li, and Yuan Yuan. Robust tensor analysis with l1-

norm. *IEEE Transactions on Circuits and Systems for Video Technology*, 20(2):172–178, 2010. 57, 58, 1469, 1470

[PMvdG+13] Jack Poulson, Bryan Marker, Robert A van de Geijn, Jeff R Hammond, and Nichols A Romero. Elemental: A new framework for distributed memory dense matrix computations. *ACM Transactions on Mathematical Software (TOMS)*, 39(2):13, 2013. 1318, 1784, 1788

[Pol63] Boris Teodorovich Polyak. Gradient methods for minimizing functionals. *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki*, 3(4):643–653, 1963. 364

[PP+08] Kaare Brandt Petersen, Michael Syskind Pedersen, et al. The matrix cookbook. *Technical University of Denmark*, 7:15, 2008. 1221, 1223

[PP13] Ninh Pham and Rasmus Pagh. Fast and scalable polynomial kernels via explicit feature maps. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining(KDD)*, pages 239–247. ACM, 2013. 1482, 1486, 1518, 1797

[PP14] Robin Pemantle and Yuval Peres. Concentration of lipschitz functionals of determinantal and other strong rayleigh measures. *Combinatorics, Probability and Computing*, 23(1):140–160, 2014. 28, 29, 549, 557, 566, 567, 569

[PR14] Sameer Pawar and Kannan Ramchandran. A robust R-FFAST framework for computing a k-sparse n-length DFT in O(k log n) sample complexity using

sparse-graph codes. In *Information Theory (ISIT), 2014 IEEE International Symposium on*, pages 1852–1856. IEEE, 2014. 38, 919

[Pri11] Eric Price. Efficient sketches for the set query problem. In *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms*, pages 41–56. Society for Industrial and Applied Mathematics, 2011. 633, 964, 965, 993

[Pri13] Eric C. Price. *Sparse recovery and Fourier sampling*. PhD thesis, Massachusetts Institute of Technology, 2013. 164

[PRS16] Gopal Pandurangan, Peter Robinson, and Michele Scquizzato. Fast distributed algorithms for connectivity and mst in large graphs. In *Proceedings of the 28th ACM Symposium on Parallelism in Algorithms and Architectures*, pages 429–438. ACM, 2016. 2317

[PRSZ18] Rina Panigrahy, Ali Rahimi, Sushant Sachdeva, and Qiuyi Zhang. Convergence results for neural networks via electrodynamics. In *ITCS*, 2018. 21, 251, 351

[PRT02] Jiming Peng, Cornelis Roos, and Tamás Terlaky. Self-regular functions and new search directions for linear and semidefinite optimization. *Mathematical Programming*, 93(1):129–171, 2002. 12, 4

[PRTV00] Christos H. Papadimitriou, Prabhakar Raghavan, Hisao Tamaki, and Santosh Vempala. Latent semantic indexing: A probabilistic analysis. *J. Comput. Syst. Sci.*, 61(2):217–235, 2000. 48, 1036

[Prü18] Heinz Prüfer. Neuer beweis eines satzes uber permutationen. *Arch. Math. Phys.*, 27:742–744, 1918. 580

[PS85] Franco P Preparata and Michael Ian Shamos. *Computational geometry: an introduction*. Springer Science & Business Media, 1985. 1334

[PS12] Ely Porat and Martin J Strauss. Sublinear time, measurement-optimal, sparse recovery for all. In *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms*, pages 1215–1227. Society for Industrial and Applied Mathematics, 2012. 623

[PS15] Eric Price and Zhao Song. A robust sparse Fourier transform in the continuous setting. In *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*, pages 583–600. IEEE, 2015. xxvi, lxxiii, 9, 10, 40, 46, 47, 164, 622, 759, 764, 765, 769, 803, 806, 822, 920, 965, 977, 996, 1016, 1901, 1904, 1905, 1908, 2514

[PS17] Aaron Potechin and David Steurer. Exact tensor completion with sum-of-squares. In *arXiv preprint*. https://arxiv.org/pdf/1702.06237, 2017. 58, 1470

[PSG17] Jeffrey Pennington, Samuel S. Schoenholz, and Surya Ganguli. Resurrecting the sigmoid in deep learning through dynamical isometry: theory and practice. *arXiv:1711.04735*, November 2017. 358

[PSL+12] Mark S Pearce, Jane A Salotti, Mark P Little, Kieran McHugh, Choonsik Lee, Kwang Pyo Kim, Nicola L Howe, Cecile M Ronckers, Preetha Rajaraman,

Alan W Craft, et al. Radiation exposure from ct scans in childhood and subsequent risk of leukaemia and brain tumours: a retrospective cohort study. *The Lancet*, 380(9840):499–505, 2012. 620

[PSW17] Eric Price, Zhao Song, and David P. Woodruff. Fast regression with an ℓ_∞ guarantee. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, 2017. xxix, 164, 1242, 1920, 2515, 2516

[PTBD16] Ho N Phien, Hoang D Tuan, Johann A Bengua, and Minh N Do. Efficient tensor completion: Low-rank tensor train. In *arXiv preprint*. https://arxiv.org/pdf/1601.01083, 2016. 1711

[PTC13] Anh Huy Phan, Petr Tichavský, and Andrzej Cichocki. Fast alternating LS algorithms for high order CANDECOMP/PARAFAC tensor factorizations. *IEEE Transactions on Signal Processing*, 61(19):4834–4846, 2013. 1797

[PW11] Eric Price and David P Woodruff. (1+ eps)-approximate sparse recovery. In *Foundations of Computer Science (FOCS), 2011 IEEE 52nd Annual Symposium on*, pages 295–304. IEEE, 2011. 620, 633, 634, 989, 991, 996, 1334

[PW16] Mert Pilanci and Martin J Wainwright. Iterative hessian sketch: Fast and accurate solution approximation for constrained least-squares. *The Journal of Machine Learning Research*, 17(1):1842–1879, 2016. 86

[PW17] Mert Pilanci and Martin J Wainwright. Newton sketch: A near linear-time optimization algorithm with linear-quadratic convergence. *SIAM Journal on Optimization*, 27(1):205–245, 2017. 86, 358

[PYLR17] Ramtin Pedarsani, Dong Yin, Kangwook Lee, and Kannan Ramchandran. Phasecode: Fast and efficient compressive phase retrieval based on sparse-graph codes. *IEEE Transactions on Information Theory*, 63(6):3663–3691, 2017. 635

[QBI+13] Saad Qaisar, Rana Muhammad Bilal, Wafa Iqbal, Muqaddas Naureen, and Sungyoung Lee. Compressive sensing: From theory to applications, a survey. *Journal of Communications and networks*, 15(5):443–456, 2013. 620

[QOSG02] Yongming Qu, George Ostrouchov, Nagiza Samatova, and Al Geist. Principal component analysis for dimension reduction in massive distributed data sets. In *Proceedings of IEEE International Conference on Data Mining (ICDM)*, 2002. 1318, 1784, 1788

[Rao19] Sharavas Rao. Improved lower bounds for the restricted isometry property of subsampled fourier matrices. In *arXiv preprint*. https://arxiv.org/pdf/1903.12146.pdf, 2019. 920, 925

[Ren88] James Renegar. A polynomial-time algorithm, based on newton's method, for linear programming. *Mathematical Programming*, 40(1-3):59–93, 1988. 10, 12, 13, 3, 4, 7

[Ren92a] James Renegar. On the computational complexity and geometry of the first-order theory of the reals, part I: introduction. preliminaries. the geometry of semi-algebraic sets. the decision problem for the existential theory of the reals. *J. Symb. Comput.*, 13(3):255–300, 1992. 1046, 1052, 1053, 1088, 1131, 1339, 1483, 1506

[Ren92b] James Renegar. On the computational complexity and geometry of the first-order theory of the reals, part II: the general decision problem. preliminaries for quantifier elimination. *J. Symb. Comput.*, 13(3):301–328, 1992. 1046, 1052, 1053, 1088, 1131, 1339, 1506

[Ren92c] James Renegar. On the computational complexity and geometry of the first-order theory of the reals. part iii: quantifier elimination. *Journal of Symbolic Computation*, 13(3):329–352, 1992. 1088

[Ren92d] James Renegar. On the computational complexity of approximating solutions for real algebraic formulae. *SIAM J. Comput.*, 21(6):1008–1025, 1992. 1046

[Ren01] James Renegar. *A mathematical view of interior-point methods in convex optimization*, volume 3. Siam, 2001. 6

[Rey89] George O Reynolds. *The New Physical Optics Notebook: Tutorials in Fourier Optics*. ERIC, 1989. 36, 917

[RHS+16] Sashank J Reddi, Ahmed Hefny, Suvrit Sra, Barnabas Poczos, and Alex Smola. Stochastic variance reduction for nonconvex optimization. In *International conference on machine learning*, pages 314–323, 2016. 15, 77

[Rip04] BD Ripley. Robust statistics. *Course notes*, 2004. 988

[RM14] Emile Richard and Andrea Montanari. A statistical model for tensor pca. In *Advances in Neural Information Processing Systems*, pages 2897–2905. https://arxiv.org/pdf/1411.1076, 2014. 58, 1470

[RMCS13] Vibhor Rastogi, Ashwin Machanavajjhala, Laukik Chitnis, and Anish Das Sarma. Finding connected components in map-reduce in logarithmic rounds. In *Data Engineering (ICDE), 2013 IEEE 29th International Conference on*, pages 50–61. IEEE, 2013. lxxviii, 2317, 2323, 2324, 2334, 2469, 2502

[RNSS16] Avik Ray, Joe Neeman, Sujay Sanghavi, and Sanjay Shakkottai. The search problem in mixture models. In *arXiv preprint*. https://arxiv.org/pdf/1610.00843, 2016. 57, 1469

[Rot13] Thomas Rothvoß. Approximating bin packing within O(logOPTloglogOPT) bins. In *Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on*, pages 20–29. IEEE, https://arxiv.org/pdf/1301.4010, 2013. 596

[Rot17] Thomas Rothvoss. Constructive discrepancy minimization for convex sets. *SIAM Journal on Computing (A preliminary of version of this paper appeared in FOCS 2014)*, 46(1):224–234, 2017. 596

[RPK86] Robert Roy, Arogyaswami Paulraj, and Thomas Kailath. Esprit–a subspace rotation approach to estimation of parameters of cisoids in noise. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 34(5):1340–1342, 1986. 762

[RR17] Shravas Rao and Oded Regev. A sharp tail bound for the expander random sampler. *arXiv preprint arXiv:1703.10205*, 2017. 26, 502

[RR19] Victor Reis and Thomas Rothvoss. Linear size sparsifier and the geometry of the operator norm ball. *arXiv preprint arXiv:1907.02145*, 2019. 1912

[RRT17] Maxim Raginsky, Alexander Rakhlin, and Matus Telgarsky. Non-convex learning via stochastic gradient langevin dynamics: a nonasymptotic analysis. In *COLT*. arXiv preprint arXiv:1702.03849, 2017. 17, 172

[RS67] A Rényi and G Szekeres. On the height of trees. *J. Austral. Math. Soc*, 7(4):497–5, 1967. 560

[RS19a] Victor Reis and Zhao Song. Tbd. *Manuscript*, 2019. xxix

[RS19b] Aviad Rubinstein and Zhao Song. Reducing approximate longest common subsequence to approximate edit distance. *arXiv preprint arXiv:1904.05451*, 2019. xxix, 1915, 2190, 2216, 2515

[RSSS19] Aviad Rubinstein, Saeed Seddighin, Zhao Song, and Xiaorui Sun. Approximation algorithms for lcs and lis with truly improved running times. In *FOCS*, 2019. xxix, 1914, 1915, 2192, 2213, 2515

[RST10] Steffen Rendle and Lars Schmidt-Thieme. Pairwise interaction tensor factorization for personalized tag recommendation. In *Proceedings of the third ACM international conference on Web search and data mining(WSDM)*, pages 81–90. ACM, 2010. 57, 1469

[RSW16] Ilya Razenshteyn, Zhao Song, and David P Woodruff. Weighted low rank approximations with provable guarantees. In *Proceedings of the 48th Annual*

Symposium on the Theory of Computing (STOC), 2016. xxvii, 6, 56, 60, 1131, 1339, 1428, 1506, 1649, 1652, 1654, 1655, 1657, 1658, 1664, 1665, 1672, 1898, 2027, 2514

[RTP16] Thomas Reps, Emma Turetsky, and Prathmesh Prabhu. Newtonian program analysis via tensor product. In *Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages(POPL)*, volume 51:1, pages 663–677. ACM, 2016. 57, 1469

[RTV05] Cornelis Roos, Tamás Terlaky, and J-Ph Vial. *Interior point methods for linear optimization*. Springer Science & Business Media, 2005. 6

[Rub18] Aviad Rubinstein. Approximating edit distance. *https://theorydish.blog/2018/07/20/approximating-edit-distance/*, 2018. 1914

[Rud99] Mark Rudelson. Random vectors in the isotropic position. *Journal of Functional Analysis*, 164(1):60–72, 1999. 25, 26, 30, 503, 548, 553, 590, 2141

[RV08] Mark Rudelson and Roman Vershynin. On sparse reconstruction from Fourier and gaussian measurements. *Communications on Pure and Applied Mathematics*, 61(8):1025–1045, 2008. lxvii, lxviii, 37, 38, 623, 762, 918, 919, 923, 925

[RV09] Mark Rudelson and Roman Vershynin. Smallest singular value of a random rectangular matrix. *Communications on Pure and Applied Mathematics*, 62(12):1707–1739, 2009. 1545

[RV10] Mark Rudelson and Roman Vershynin. Non-asymptotic theory of random matrices: extreme singular values. *arXiv preprint arXiv:1003.2990*, 2010. 1226

[RVW00] Omer Reingold, Salil Vadhan, and Avi Wigderson. Entropy waves, the zig-zag graph product, and new constant-degree expanders and extractors. In *Foundations of Computer Science, 2000. Proceedings. 41st Annual Symposium on*, pages 3–13. IEEE, 2000. 27, 503

[RVW16] Tim Roughgarden, Sergei Vassilvitskii, and Joshua R. Wang. Shuffles and circuits: (on lower bounds for modern parallel computation). In *Proceedings of the 28th ACM Symposium on Parallelism in Algorithms and Architectures, SPAA 2016, Asilomar State Beach/Pacific Grove, CA, USA, July 11-13, 2016*, pages 1–12, 2016. 2316

[RVW18] Tim Roughgarden, Sergei Vassilvitskii, and Joshua R Wang. Shuffles and circuits (on lower bounds for modern parallel computation). *Journal of the ACM (JACM)*, 65(6):41, 2018. 1916

[RYP03] Ryan Rifkin, Gene Yeo, and Tomaso Poggio. Regularized least-squares classification. *Nato Science Series Sub Series III Computer and Systems Sciences*, 190:131–154, 2003. 1923

[SA96] Emilio Salinas and Laurence F. Abbott. A model of multiplicative neural responses in parietal cortex. *Proceedings of the National Academy of Sciences*, 93(21):11956–11961, 1996. 23, 353

[SA15] Hanie Sedghi and Anima Anandkumar. Provable methods for training neural networks with sparse connectivity. In *ICLR*. arXiv preprint arXiv:1412.2693, 2015. 358

[SAH+13] Lixin Shi, O Andronesi, Haitham Hassanieh, Badih Ghazi, Dina Katabi, and Elfar Adalsteinsson. Mrs sparse-FFT: Reducing acquisition time and artifacts for in vivo 2d correlation spectroscopy. In *ISMRM'13, Int. Society for Magnetic Resonance in Medicine Annual Meeting and Exhibition*, 2013. 41, 676, 687

[Sar06] Tamás Sarlós. Improved approximation algorithms for large matrices via random projections. In *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS) , 21-24 October 2006, Berkeley, California, USA, Proceedings*, pages 143–152, 2006. 52, 13, 1109, 1117, 1337, 1338, 1370, 1426, 1468, 1907, 1920, 1922, 1924, 1927, 1966

[SAZ09] Noam Shental, Amnon Amir, and Or Zuk. Rare-allele detection using compressed se (que) nsing. *arXiv preprint arXiv:0909.0400*, 2009. 33, 617

[SBG04] Age K. Smilde, Rasmus Bro, and Paul Geladi. *Multi-way Analysis with Applications in the Chemical Sciences*. Wiley, 2004. 57, 1469

[SBS+17] Hojjat Salehinejad, Julianne Baarbe, Sharan Sankar, Joseph Barfett, Errol Colak, and Shahrokh Valaee. Recent advances in recurrent neural networks. *arXiv preprint arXiv:1801.01078*, 2017. 23, 353, 357

[SBT17] David Sutter, Mario Berta, and Marco Tomamichel. Multivariate trace inequalities. *Communications in Mathematical Physics*, 352(1):37–58, 2017. arXiv:1604.03023. 501, 507, 508, 517, 518, 522, 524

[SC15] Jimin Song and Kevin C Chen. Spectacle: fast chromatin state annotation using spectral learning. *Genome biology*, 16(1):33, 2015. 57, 1469

[SC16] Daniel Soudry and Yair Carmon. No bad local minima: Data independent training error guarantees for multilayer neural networks. *arXiv preprint arXiv:1605.08361*, 2016. 21, 251

[Sch61] Craige Schensted. Longest increasing and decreasing subsequences. *Canadian Journal of Mathematics*, 13:179–191, 1961. 2218

[Sch81] Ralph Otto Schmidt. A signal subspace approach to multiple emitter location spectral estimation. *Ph. D. Thesis, Stanford University*, 1981. 762

[Sch12] Leonard J Schulman. Cryptography from tensor problems. In *IACR Cryptology ePrint Archive*, volume 2012, page 244. https://eprint.iacr.org/2012/244, 2012. 57, 1469

[Sch18] Aaron Schild. An almost-linear time algorithm for uniform random spanning tree generation. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, 2018. 29, 550, 555

[SGGSD17] Samuel S. Schoenholz, Justin Gilmer, Surya Ganguli, and Jascha Sohl-Dickstein. Deep information propagation. In *ICLR*, 2017. 358

[SGS15] Rupesh K Srivastava, Klaus Greff, and Jürgen Schmidhuber. Training very deep networks. In *Advances in neural information processing systems (NeurIPS)*, pages 2377–2385, 2015. 255

[SH05] Amnon Shashua and Tamir Hazan. Non-negative tensor factorization with applications to statistics and computer vision. In *Proceedings of the 22nd international conference on Machine learning(ICML)*, pages 792–799. ACM, 2005. 57, 1469

[SH11] Martin Slawski and Matthias Hein. Sparse recovery by thresholded non-negative least squares. In *Advances in Neural Information Processing Systems*, pages 1926–1934, 2011. 988

[Sha11] Ohad Shamir. A variant of azuma's inequality for martingales with subgaussian tails. *ArXiv e-prints*, abs/1110.2392, 10 2011. 283

[SHM+16] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016. 20, 250, 351

[Shp90] D. Shpak. A weighted-least-squares matrix decomposition method with applications to the design of two-dimensional digital filters. In *IEEE Thirty Third Midwest Symposium on Circuits and Systems*, 1990. 51, 1038

[SHW+16] Mao Shaowu, Zhang Huanguo, Wu Wanqing, Zhang Pei, Song Jun, and Liu Jinhui. Key exchange protocol based on tensor decomposition problem. *China Communications*, 13(3):174–183, 2016. 57, 1469

[Sie] Siemens. Compressed sensing beyond speed. https://www.healthcare.siemens.com/magnetic-resonance-imaging/clinical-specialities/compressed-sensing. 36, 917

[SJ03] Nathan Srebro and Tommi S. Jaakkola. Weighted low-rank approximations. In *ICML*, 2003. 50, 51, 1038

[SK18] Eric Price Sushrut Karmalkar. Compressed sensing with adversarial sparse noise via l1 regression. *arXiv preprint arXiv:1809.08055*, 2018. 990

[SL09] Roman Sandler and Michael Lindenbaum. Nonnegative matrix factorization with earth mover's distance metric. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1873–1880. IEEE, 2009. 1262, 1263

[SLC+17] Fanhua Shang, Yuanyuan Liu, James Cheng, KW Ng, and Yuichi Yoshida. Variance reduced stochastic gradient descent with sufficient decrease. *arXiv preprint arXiv:1703.06807*, 2017. 13, 74

[SLJ+15] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015. 22, 251

[SLRB17] Mark Schmidt, Nicolas Le Roux, and Francis Bach. Minimizing finite sums with the stochastic average gradient. *Mathematical Programming*, 162(1-2):83–112, 2017. 15, 77

[Slu77] Eric V Slud. Distribution inequalities for the binomial law. *The Annals of Probability*, 5(3):404–412, 1977. 2506, 2507

[Sma98] Steve Smale. Mathematical problems for the next century. *The mathematical intelligencer*, 20(2):7–15, 1998. 1895

[SMH11] Ilya Sutskever, James Martens, and Geoffrey Hinton. Generating text with recurrent neural networks. In *International Conference on Machine Learning (ICML)*, pages 1017–1024, 2011. 23, 353

[SMT+18] Max Simchowitz, Horia Mania, Stephen Tu, Michael I Jordan, and Benjamin Recht. Learning without mixing: Towards a sharp analysis of linear system identification. In *Conference on Learning Theory (COLT)*. arXiv preprint arXiv:1802.08334, 2018. 257

[Son17] Zhao Song. High dimensional Fourier transform in the continuous setting. In *Manuscript*, 2017. 21, 250, 351, 1901, 2140, 2309

[SP97] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997. 353

[Spe85] Joel Spencer. Six standard deviations suffice. *Trans. Amer. Math. Soc.*, 289(2):679–706, 1985. 30, 590

[SS91] Hava T Siegelmann and Eduardo D Sontag. Turing computability with neural nets. *Applied Mathematics Letters*, 4(6):77–80, 1991. 354

[SS10] Michael Saks and C Seshadhri. Estimating the longest increasing sequence in polylogarithmic time. In *Proceedings of the Fifty-First Annual IEEE Symposium on Foundations of Computer Science*, 2010. 2214, 2218, 2219

[SS11] Daniel A Spielman and Nikhil Srivastava. Graph sparsification by effective resistances. *SIAM Journal on Computing*, 40(6):1913–1926, 2011. 25, 29, 13, 548, 551, 552, 559, 581, 596

[SS14] Zhao Song and Wen Sun. Probabilistic recharging model in uncertain environments. In *AAMAS*, pages 1343–1344, 2014. xxix

[SS16] Shai Shalev-Shwartz. SDCA without duality, regularization, and individual convexity. In *International Conference on Machine Learning*, pages 747–754, 2016. 15, 77

[SS17] Tselil Schramm and David Steurer. Fast and robust tensor decomposition with applications to dictionary learning. *manuscript*, 2017. 58, 1470

[SS18] Itay Safran and Ohad Shamir. Spurious local minima are common in two-layer ReLU neural networks. In *International Conference on Machine Learning (ICML)*. http://arxiv.org/abs/1712.08968, 2018. 21, 251, 351

[SS19] Zhao Song and Wen Sun. Efficient model-free reinforcement learning in metric spaces. *arXiv preprint arXiv:1905.00475*, 2019. xxix

[SSB14] Haşim Sak, Andrew Senior, and Françoise Beaufays. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *Fifteenth annual conference of the international speech communication association*, 2014. 22, 352

[SSBD14] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014. 14, 74

[SSN12] Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. Lstm neural networks for language modeling. In *Thirteenth annual conference of the international speech communication association*, 2012. 22, 352

[SSS+17] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of Go without human knowledge. *Nature*, 550(7676):354, 2017. 20, 250, 351

[SSV12] Zhao Song, Seyed Abbas Sadat, and Richard T. Vaughan. Mo-lost: Adaptive ant trail untangling in multi-objective multi-colony robot foraging. In *Eleventh International Conference on Autonomous Agents and Multiagent Systems(AAMAS2012), June 4-8 2012, Universitat Politecnica de Valencia, Valencia, Spain.*, 2012. xxix

[SSZ13] Shai Shalev-Shwartz and Tong Zhang. Stochastic dual coordinate ascent methods for regularized loss minimization. *Journal of Machine Learning Research*, 14(Feb):567–599, 2013. 13, 74

[SSZ14] Shai Shalev-Shwartz and Tong Zhang. Accelerated proximal stochastic dual coordinate ascent for regularized loss minimization. In *International Conference on Machine Learning*, pages 64–72, 2014. 15, 22, 77, 251

[ST04] Daniel A. Spielman and Shang-Hua Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *Proceedings of the Thirty-sixth Annual ACM Symposium on Theory of Computing (STOC)*, pages 81–90, New York, NY, USA, 2004. ACM. 25, 6, 372, 381, 464, 548, 551, 596

[ST11] Daniel A Spielman and Shang-Hua Teng. Spectral sparsification of graphs. *SIAM Journal on Computing*, 40(4):981–1025, 2011. 29

[Ste06] Alwin Stegeman. Degeneracy in candecomp/parafac explained for p × p × 2 arrays of rank p+1 or higher. *Psychometrika*, 71(3):483–501, 2006. 59, 1471

[Ste08] Alwin Stegeman. Low-rank approximation of generic p × q × 2 arrays and diverging components in the candecomp/parafac model. *SIAM Journal on Matrix Analysis and Applications*, 30(3):988–1007, 2008. 59, 1471

[STLS14] Marco Signoretto, Dinh Quoc Tran, Lieven De Lathauwer, and Johan A. K. Suykens. Learning with tensors: a framework based on convex optimization and spectral regularization. *Machine Learning*, 94(3):303–351, 2014. 57, 1469

[Str69] Volker Strassen. Gaussian elimination is not optimal. *Numerische Mathematik*, 13(4):354–356, 1969. 89, 761, 775, 792, 1133, 1426, 1468, 1515

[SV82] Yossi Shiloach and Uzi Vishkin. An $O(\log n)$ parallel connectivity algorithm. *J. Algorithms*, 3(1):57–67, 1982. 2315

[SV12a] Nariankadu D. Shyamalkumar and Kasturi R. Varadarajan. Efficient subspace approximation algorithms. *Discrete & Computational Geometry*, 47(1):44–63, 2012. 54, 1112

[SV12b] Zhao Song and Richard T. Vaughan. Multi-robot, multi-patch foraging with maximum sustainable yield. In *Artificial Life(ALIFE XIII), July 19–22, 2012, Michigan State University, East Lansing, Michigan, USA*, 2012. xxix

[SV13] Zhao Song and Richard T. Vaughan. Sustainable robot foraging: Adaptive fine-grained multi-robot task allocation for maximum sustainable yield of biological resources. In *IROS*, pages 3309–3316, 2013. xxix

[SV16] Damian Straszak and Nisheeth K Vishnoi. Irls and slime mold: Equivalence and convergence. *arXiv preprint arXiv:1601.02712*, 2016. 34, 617

[SVL14] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems (NeurIPS)*, pages 3104–3112, 2014. 22, 352

[SVWX17] Le Song, Santosh Vempala, John Wilmes, and Bo Xie. On the complexity of learning neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 5514–5522, 2017. 258

[SW11] Christian Sohler and David P Woodruff. Subspace embeddings for the ℓ_1-norm with applications. In *Proceedings of the forty-third annual ACM sym-*

posium on Theory of computing, pages 755–764. ACM, 2011. 52, 1109, 1117, 1118, 1170, 1185, 1191, 1253, 1336, 1338, 1341, 1344, 1345, 1975

[SW15] Tselil Schramm and Benjamin Weitz. Low-rank matrix completion with adversarial missing entries. *CoRR*, abs/1506.03137, 2015. 51, 1038, 1043

[SW18] Christian Sohler and David P. Woodruff. Strong coresets for k-median and subspace approximation, goodbye dimension. In *FOCS*. IEEE, 2018. 2094

[SWY+19] Zhao Song, Ruosong Wang, Lin F. Yang, Hongyang Zhang, and Peilin Zhong. Efficient symmetric norm regression via linear sketching. *Manuscript*, 2019. xxix, 1965, 2515, 2516

[SWZ16] Zhao Song, David P. Woodruff, and Huan Zhang. Sublinear time orthogonal tensor decomposition. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems (NIPS) 2016, December 5-10, 2016, Barcelona, Spain*, pages 793–801, 2016. xxviii, 3, 4, 58, 61, 1469, 2025, 2514

[SWZ17] Zhao Song, David P Woodruff, and Peilin Zhong. Low rank approximation with entrywise ℓ_1-norm error. In *Proceedings of the 49th Annual Symposium on the Theory of Computing (STOC)*. ACM, https://arxiv.org/pdf/1611.00898, 2017. xxvii, 3, 6, 56, 60, 1145, 1334, 1335, 1336, 1337, 1338, 1339, 1342, 1345, 1357, 1370, 1373, 1399, 1428, 1429, 1433, 1476, 1488, 1506, 1604, 1617, 1619, 1623, 1624, 1628, 1629, 1645, 1646, 1647, 1717, 1718, 1720, 1725, 1727, 1728, 1729, 1736, 1737, 1738, 1739, 1779, 1784, 1788, 2027, 2082, 2094, 2514

[SWZ18] Zhao Song, David P Woodruff, and Peilin Zhong. Towards a zero-one law for entrywise low rank approximation. *arXiv preprint arXiv:1811.01442*, 2018. xxviii, 60, 1007, 1967, 2027, 2082, 2083, 2514

[SWZ19a] Zhao Song, David P Woodruff, and Peilin Zhong. Average case column subset selection for entrywise ℓ_1-norm loss. *Manuscript*, 2019. 2514

[SWZ19b] Zhao Song, David P Woodruff, and Peilin Zhong. Relative error tensor low rank approximation. In *SODA*. arXiv preprint arXiv:1704.08246, 2019. xxviii, 3, 6, 7, 60, 246, 1341, 1357, 1433, 1899, 2025, 2027, 2042, 2094, 2514

[SY19] Zhao Song and Xin Yang. Quadratic suffices for over-parametrization via matrix chernoff bound. *arXiv preprint arXiv:1906.03593*, 2019. xxix, 25, 548, 1908, 1909, 2139, 2516

[SZ13] Zhao Song and Yuke Zhu. Graphical model-based learning in high dimensional feature spaces. In *Twenty-Seventh AAAI Conference on Artificial Intelligence*, 2013. xxix

[TBR15] Gongguo Tang, Badri Narayan Bhaskar, and Benjamin Recht. Near minimax line spectral estimation. *Information Theory, IEEE Transactions on*, 61(1):499–512, 2015. 763

[TBSR13] Gongguo Tang, Badri Narayan Bhaskar, Parikshit Shah, and Benjamin Recht. Compressed sensing off the grid. *Information Theory, IEEE Transactions on*, 59(11):7465–7490, 2013. 41, 676, 686, 763

[TD99] Françoise Tisseur and Jack Dongarra. A parallel divide and conquer algorithm for the symmetric eigenvalue problem on distributed memory architectures. *SIAM Journal on Scientific Computing*, 20(6):2223–2236, 1999. 1318, 1784, 1788

[Ter13] Tamás Terlaky. *Interior point methods of mathematical programming*, volume 5. Springer Science & Business Media, 2013. 6

[TG07] Joel Tropp and Anna C Gilbert. Signal recovery from partial information via orthogonal matching pursuit. *IEEE Trans. Inform. Theory*, 53(12):4655–4666, 2007. 38, 918

[Tho65] Colin J Thompson. Inequality with applications in statistical mechanics. *Journal of Mathematical Physics*, 6(11):1812–1813, 1965. 28, 506

[Tia17] Yuandong Tian. An analytical formula of population gradient for two-layered ReLU network and its applications in convergence and critical point analysis. In *International Conference on Machine Learning (ICML)*. http://arxiv.org/abs/1703.00560, 2017. 21, 250, 351, 358, 2140

[Tib96] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996. 14, 74, 1334

[Tij] H Tijms. Understanding probability: Chance rules in everyday life. 2004. 25, 548

[TK11] Petr Tichavsky and Zbyněk Koldovsky. Weight adjusted tensor method for blind separation of underdetermined mixtures of nonstationary sources. *IEEE Transactions on Signal Processing*, 59(3):1037–1047, 2011. 58, 1470

[TLW+06] Dharmpal Takhar, Jason N Laska, Michael B Wakin, Marco F Duarte, Dror Baron, Shriram Sarvotham, Kevin F Kelly, and Richard G Baraniuk. A new compressive imaging camera architecture using optical-domain compression. In *Computational Imaging IV*, volume 6065, page 606509. International Society for Optics and Photonics, 2006. 33, 617

[TM17] Davoud Ataee Tarzanagh and George Michailidis. Fast monte carlo algorithms for tensor operations. In *arXiv preprint*. https://arxiv.org/pdf/1704.04362, 2017. 1476, 1504

[Tre01] Luca Trevisan. Non-approximability results for optimization problems on bounded degree instances. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing (STOC)*, pages 453–461. ACM, 2001. 1671, 1703

[Tro11a] Joel A Tropp. Freedman's inequality for matrix martingales. *Electronic Communications in Probability*, 16:262–270, 2011. 25, 548

[Tro11b] Joel A Tropp. Improved analysis of the subsampled randomized hadamard transform. *Advances in Adaptive Data Analysis*, 3(01n02):115–126, 2011. 1907, 1922

[Tro11c] Joel A Tropp. User-friendly tail bounds for matrix martingales. Technical report, CALIFORNIA INST OF TECH PASADENA, 2011. 553, 558, 2510

[Tro12] Joel A Tropp. User-friendly tail bounds for sums of random matrices. *Foundations of computational mathematics*, 12(4):389–434, 2012. 25, 26, 30, 31, 503, 511, 548, 551, 553, 557, 559, 590, 591, 2141

[Tro15] Joel A. Tropp. An introduction to matrix concentration inequalities. *Foundations and Trends® in Machine Learning*, 8(1-2):1–230, 2015. 504, 2141, 2510

[TS19] Ewin Tang and Zhao Song. Tbd. *Manuscript*, 2019. xxix

[TSHK11] Ryota Tomioka, Taiji Suzuki, Kohei Hayashi, and Hisashi Kashima. Statistical performance of convex tensor decomposition. In *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems (NIPS). Proceedings of a meeting held 12-14 December 2011, Granada, Spain.*, pages 972–980, 2011. 58, 1470

[Tso10] Charalampos E. Tsourakakis. MACH: fast randomized tensor decompositions. In *SDM*, pages 689–700, 2010. 1797, 1798, 1804

[TSSW00] Luca Trevisan, Gregory B Sorkin, Madhu Sudan, and David P Williamson. Gadgets, approximation, and linear programming. *SIAM Journal on Computing*, 29(6):2074–2097, 2000. 1269

[Tsy09] Alexandre B Tsybakov. Introduction to nonparametric estimation. revised and extended from the 2004 french original. translated by vladimir zaiats, 2009. 1219

[Tuk60] John W Tukey. A survey of sampling from contaminated distributions. *Contributions to probability and statistics*, pages 448–485, 1960. 992, 993, 1007

[Tur41] Paul Turán. On an external problem in graph theory. *Mat. Fiz. Lapok*, 48:436–452, 1941. 2284

[TWZS15] Hsiao-Yu Fish Tung, Chao-Yuan Wu, Manzil Zaheer, and Alexander J Smola. Spectral methods for the hierarchical dirichlet process. 2015. 1797, 1799, 1814, 1887

[TZ12] Mikkel Thorup and Yin Zhang. Tabulation-based 5-independent hashing with applications to linear probing and second moment estimation. *SIAM Journal on Computing*, 41(2):293–331, 2012. 621

[UHZ+16] Madeleine Udell, Corinne Horn, Reza Zadeh, Stephen Boyd, et al. Generalized low rank models. *Foundations and Trends® in Machine Learning*, 9(1):1–118, 2016. 55, 56, 1342, 1427, 1428, 1450

[Vai87] Pravin M Vaidya. An algorithm for linear programming which requires $O(((m+n)n^2+(m+n)^{1.5}n)L)$arithmetic operations. In *FOCS*. IEEE, 1987. 10, 3, 11

[Vai89a] Pravin M Vaidya. A new algorithm for minimizing convex functions over convex sets. In *30th Annual Symposium on Foundations of Computer Science(FOCS)*, pages 338–343. IEEE, 1989. 10, 3

[Vai89b] Pravin M Vaidya. Speeding-up linear programming using fast matrix multiplication. In *FOCS*. IEEE, 1989. 10, 13, 15, 3, 20, 66, 77

[Val90] Leslie G. Valiant. A bridging model for parallel computation. *Commun. ACM*, 33(8):103–111, 1990. 2315

[Vap92] Vladimir Vapnik. Principles of risk minimization for learning theory. In *Advances in neural information processing systems*, pages 831–838, 1992. 13, 14, 74

[Vap13] Vladimir Vapnik. *The nature of statistical learning theory*. Springer science & business media, 2013. 13, 74

[Vas09] M Alex O Vasilescu. *A multilinear (tensor) algebraic framework for computer graphics, computer vision, and machine learning*. PhD thesis, Citeseer, 2009. 57, 1469

[vdBLSS19] Jan van den Brand, Yin Tat Lee, Aaron Sidford, and Zhao Song. Tbd. *Manuscript*, 2019. xxix

[vdH03] Joris van der Hoeven. *Majorants for formal power series*. Citeseer, 2003. 232

[Ver10] Roman Vershynin. Introduction to the non-asymptotic analysis of random matrices. *arXiv preprint arXiv:1011.3027*, 2010. 1942

[Ver14] Sergio Verdú. Total variation distance and the distribution of relative information. In *ITA*, pages 1–3. Citeseer, 2014. 1219

[VGT12] Gaël Varoquaux, Alexandre Gramfort, and Bertrand Thirion. Small-sample brain mapping: sparse recovery on spatially correlated designs with randomization and clustering. *arXiv preprint arXiv:1206.6447*, 2012. 988

[VL92] Charles F Van Loan. *Computational frameworks for the fast Fourier transform*, volume 10 of *Frontiers in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1992. 2024

[VL00] Charles F Van Loan. The ubiquitous kronecker product. *Journal of computational and applied mathematics*, 123(1-2):85–100, 2000. 2029, 2042, 2065, 2080

[VLP93] Charles F Van Loan and N. Pitsianis. Approximation with Kronecker products. In *Linear algebra for large scale and real-time applications (Leuven, 1992)*, volume 232 of *NATO Adv. Sci. Inst. Ser. E Appl. Sci.*, pages 293–314. Kluwer Acad. Publ., Dordrecht, 1993. 2024

[Voe11] David G Voelz. *Computational Fourier Optics: A MATLAB Tutorial (SPIE Tutorial Texts Vol. TT89)*. SPIE press, 2011. 36, 917

[VT01] H. L. Van Trees. *Detection, estimation, and modulation theory*. Wiley, 1 edition, September 2001. 53, 1111

[VT02] M Alex O Vasilescu and Demetri Terzopoulos. Multilinear analysis of image ensembles: Tensorfaces. In *European Conference on Computer Vision*, pages 447–460. Springer, 2002. 57, 1469

[VT04] M Alex O Vasilescu and Demetri Terzopoulos. Tensortextures: Multilinear image-based rendering. In *ACM Transactions on Graphics (TOG)*, volume 23:3, pages 336–342. ACM, 2004. 57, 1469

[VW18] Santosh Vempala and John Wilmes. Polynomial convergence of gradient descent for training one-hidden-layer neural networks. *arXiv preprint arXiv:1805.02677*, 2018. 351

[WA03] Hongcheng Wang and Narendra Ahuja. Facial expression decomposition. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 958–965. IEEE, 2003. 57, 1469

[WA16] Yining Wang and Animashree Anandkumar. Online and differentially-private tensor decomposition. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems (NIPS) 2016, December 5-10, 2016, Barcelona, Spain*. https://arxiv.org/pdf/1606.06237, 2016. 58, 1469, 1809, 1810, 1826, 1850, 1855, 1861

[Wag08] R. Wagner. Tail estimates for sums of variables sampled by a random walk. *Combinatorics, Probability and Computing*, 17(2), 2008. 26, 502

[Wag11] David Wagner. Multivariate stable polynomials: theory and applications. *Bulletin of the American Mathematical Society*, 48(1):53–84, 2011. 599

[Wai19] M.J. Wainwright. *High-Dimensional Statistics: A Non-Asymptotic Viewpoint*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 2019. 2068

[Wal77] Alastair J Walker. An efficient method for generating discrete random variables with general distributions. *ACM Transactions on Mathematical Software (TOMS)*, 3(3):253–256, 1977. 1807

[Wan16] Yining Wang. Personal communication. 2016. 1808

[Wan19a] Lan Wang. A new tuning-free approach to high-dimensional regression. ., 2019. 2026

[Wan19b] Lan Wang. Personal communication. ., 2019. 2026

[Wat64] Geoffrey S Watson. Smooth regression analysis. *Sankhyā: The Indian Journal of Statistics, Series A*, pages 359–372, 1964. 14, 74

[Wea04] Nik Weaver. The Kadison-Singer problem in discrepancy theory. *Discrete Math.*, 278(1-3):227–239, 2004. 593

[Wes94] Carl-Fredrik Westin. *A tensor framework for multidimensional signal processing*. PhD thesis, Linköping University Electronic Press, 1994. 57, 1469

[Whi12] Tom White. *Hadoop: the definitive guide*. O'Reilly, 2012. 2315

[Wik16] Wikipedia. Random permutation. In *https://en.wikipedia.org/wiki/Random_permutation*, 2016. 1819

[Wil96] David Bruce Wilson. Generating random spanning trees more quickly than the cover time. In *Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing (STOC)*, pages 296–303, New York, NY, USA, 1996. ACM. 29, 550

[Wil05] Herbert S Wilf. *Generating functionology*. AK Peters/CRC Press, 2005. 666

[Wil12] Virginia Vassilevska Williams. Multiplying matrices faster than coppersmith-winograd. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing (STOC)*, pages 887–898. ACM, 2012. 12, 4, 75, 89, 158, 159, 761, 775, 792, 881, 1133, 1426, 1468, 1515, 1921

[WKL09] Lan Wang, Bo Kai, and Runze Li. Local rank inference for varying coefficient models. *Journal of the American Statistical Association*, 104(488):1631–1645, 2009. 2026

[WL09] Lan Wang and Runze Li. Weighted wilcoxon-type smoothly clipped absolute deviation method. *Biometrics*, 65(2):564–571, 2009. 2026

[WLSH14] Chi Wang, Xueqing Liu, Yanglei Song, and Jiawei Han. Scalable moment-based inference for latent dirichlet allocation. In *ECML-PKDD*, pages 290–305, 2014. 1797

[WLY+15] Ye Wang, Meng Li, Xinyang Yi, Zhao Song, Michael Orshansky, and Constantine Caramanis. Novel power grid reduction method based on l1 regularization. In *DAC*, 2015. xxix

[WM01] B. Walczak and DL Massart. Dealing with missing data: Part i. *Chemometrics and Intelligent Laboratory Systems*, 58(1):15–27, 2001. 58, 1469

[Woj96] P. Wojtaszczyk. *Banach spaces for analysts*, volume 25. Cambridge University Press, 1996. 1987

[Woo50] Max A Woodbury. Inverting modified matrices. *Memorandum report*, 42(106):336, 1950. 14

[Woo14a] David P Woodruff. Low rank approximation lower bounds in row-update streams. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1781–1789, 2014. 1922, 1928, 1934, 1946

[Woo14b] David P. Woodruff. Sketching as a tool for numerical linear algebra. *Foundations and Trends in Theoretical Computer Science*, 10(1-2):1–157, 2014. 52, 53, 770, 1056, 1109, 1110, 1116, 1134, 1377, 1479, 1482, 1516, 1561, 1767, 1781, 1786, 1791, 1992, 2000, 2025, 2027, 2031, 2037, 2042, 2077

[WPB+18] Lan Wang, Bo Peng, Jelena Bradic, Runze Li, and Yunan Wu. A tuning-free robust and efficient approach to high-dimensional regression. Technical report, School of Statistics, University of Minnesota, 2018. 2026

[Wri73] Farrol Tim Wright. A bound on tail probabilities for quadratic forms in independent random variables whose distributions are not necessarily symmetric. *The Annals of Probability*, 1(6):1068–1070, 1973. 2505

[Wri97] Stephen J Wright. *Primal-dual interior-point methods*, volume 54. Siam, 1997. 6

[WS15] Yining Wang and Aarti Singh. Column subset selection with missing data via active sampling. In *The 18th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 1033–1041, 2015. 1433, 1476

[WTSA15] Yining Wang, Hsiao-Yu Tung, Alexander J Smola, and Anima Anandkumar. Fast and guaranteed tensor decomposition via sketching. In *Advances in Neural Information Processing Systems (NIPS)*, pages 991–999. https://arxiv.org/pdf/1506.04448, 2015. 58, 1469, 1518, 1797, 1799, 1800, 1803, 1808, 1809, 1810, 1811, 1813, 1814, 1824, 1826, 1828, 1887

[WW19] Ruosong Wang and David P. Woodruff. Tight bounds for ℓ_p oblivious subspace embeddings. In *SODA*, pages 1825–1843, 2019. 1966, 1971, 1972, 2031, 2032

[WWS+05] Hongcheng Wang, Qing Wu, Lin Shi, Yizhou Yu, and Narendra Ahuja. Out-of-core tensor approximation of multi-dimensional matrices of visual data. *ACM Transactions on Graphics (TOG)*, 24(3):527–535, 2005. 57, 1469

[WX05] Avi Wigderson and David Xiao. A randomness-efficient sampler for matrix-valued functions and applications. In *Foundations of Computer Science, 2005. FOCS 2005. 46th Annual IEEE Symposium on*, pages 397–406. IEEE, 2005. 26, 27, 502, 503, 504, 505, 508

[WX06] Avi Wigderson and David Xiao. Retraction of Wigderson-Xiao: A randomness-efficient sampler for matrix valued functions, eccc tr05-107. 2006. 28, 501, 504

[WX08] Avi Wigderson and David Xiao. Derandomizing the AW matrix-valued Chernoff bound using pessimistic estimators and applications. *Theory of Computing*, 4(3):53–76, 2008. 505

[WY13] Naiyan Wang and Dit-Yan Yeung. Bayesian robust matrix factorization for image and video processing. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1785–1792, 2013. 1334

[WYWY12] Naiyan Wang, Tiansheng Yao, Jingdong Wang, and Dit-Yan Yeung. A probabilistic approach to robust matrix factorization. In *European Conference on Computer Vision*, pages 126–139. Springer, 2012. 1334

[WZ13] David P. Woodruff and Qin Zhang. Subspace embeddings and ℓ_p-regression using exponential random variables. In *COLT 2013 - The 26th Annual Conference on Learning Theory, June 12-14, 2013, Princeton University, NJ, USA*, pages 546–567, 2013. 52, 1109, 1117, 1119, 1456, 1966, 1971

[WZ16] David P Woodruff and Peilin Zhong. Distributed low rank approximation of implicit functions of a matrix. In *32nd IEEE International Conference on Data Engineering (ICDE)*. https://arxiv.org/pdf/1601.07721, 2016. 1318, 1784, 1788

[WZC+18] Tsui-Wei Weng, Huan Zhang, Hongge Chen, Zhao Song, Cho-Jui Hsieh, Duane Boning, Inderjit S Dhillon, and Luca Daniel. Towards fast computation of certified robustness for relu networks. In *ICML*. https://arxiv.org/pdf/1804.09699, 2018. xxix

[XBSD+18] Lechao Xiao, Yasaman Bahri, Jascha Sohl-Dickstein, Samuel S. Schoenholz, and Jeffrey Pennington. Dynamical isometry and a mean field theory of CNNs: How to train 10,000-layer vanilla convolutional neural networks. In *International Conference on Machine Learning (ICML)*, 2018. 358

[XCS10] Huan Xu, Constantine Caramanis, and Sujay Sanghavi. Robust pca via outlier pursuit. In *Advances in Neural Information Processing Systems*, pages 2496–2504, 2010. 1334, 1370

[XCS11] Liang Xiong, Xi Chen, and Jeff Schneider. Direct robust matrix factorizatoin for anomaly detection. In *Data Mining (ICDM), 2011 IEEE 11th International Conference on*, pages 844–853. IEEE, 2011. 1334

[XJR02] Eric P Xing, Michael I Jordan, and Stuart Russell. A generalized mean field algorithm for variational inference in exponential families. In *Proceedings of the Nineteenth conference on Uncertainty in Artificial Intelligence*, pages 583–591. Morgan Kaufmann Publishers Inc., 2002. 14, 74

[XZ14] Lin Xiao and Tong Zhang. A proximal stochastic gradient method with progressive variance reduction. *SIAM Journal on Optimization*, 24(4):2057–2075, 2014. 15, 77

[Yan18] Wei Yang. Classification on CIFAR-10/100 and ImageNet with PyTorch, 2018. Accessed: 2018-04. lxxi, lxxii, 265, 344, 345, 346, 347, 348, 349

[Yao77] Andrew Chi-Chin Yao. Probabilistic computations: Toward a unified measure

of complexity. In *Foundations of Computer Science, 1977., 18th Annual Symposium on*, pages 222–227. IEEE, 1977. 1242

[Yao82] Andrew Chi-Chih Yao. On constructing minimum spanning trees in k-dimensional spaces and related problems. *SIAM Journal on Computing*, 11(4):721–736, 1982. 1334

[YC09] Jiho Yoo and Seungjin Choi. Weighted nonnegative matrix co-tri-factorization for collaborative prediction. In *ACML*, 2009. 1044

[YC14] Tatsuya Yokota and Andrzej Cichocki. Multilinear tensor rank estimation via sparse tucker decomposition. In *Soft Computing and Intelligent Systems (SCIS), 2014 Joint 7th International Conference on and Advanced Intelligent Systems (ISIS), 15th International Symposium on*, pages 478–483. IEEE, 2014. 1706

[YCRM16] Jiyan Yang, Yin-Lam Chow, Christopher Ré, and Michael W Mahoney. Weighted sgd for ℓ_p regression with randomized preconditioning. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 558–569. Society for Industrial and Applied Mathematics, https://arxiv.org/pdf/1502.03571, 2016. 1717

[YCS11] Yusuf Kenan Yilmaz, Ali Taylan Cemgil, and Umut Simsekli. Generalised coupled tensor factorisation. In *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011. Proceedings of a meeting held 12-14 December 2011, Granada, Spain.*, pages 2151–2159, 2011. 57, 1469

[YCS16] Xinyang Yi, Constantine Caramanis, and Sujay Sanghavi. Solving a mixture of many random linear equations by tensor decomposition and alternating minimization. In *arXiv preprint*. https://arxiv.org/pdf/1608.05749, 2016. 57, 1469

[Ye97] Yinyu Ye. *Interior point algorithms: theory and analysis*. Springer, 1997. 6

[Yel14] Yelp. Yelp dataset. *http://www.yelp.com/dataset_challenge*, 2014. 1426

[YFS16] Yuning Yang, Yunlong Feng, and Johan AK Suykens. Robust low-rank tensor recovery with regularized redescending m-estimator. *IEEE transactions on neural networks and learning systems*, 27(9):1933–1946, 2016. 58, 1470

[YLPR15] Dong Yin, Kangwook Lee, Ramtin Pedarsani, and Kannan Ramchandran. Fast and robust compressive phase retrieval with sparse-graph codes. In *2015 IEEE International Symposium on Information Theory (ISIT)*, pages 2583–2587. IEEE, 2015. 635

[YMM13] J. Yang, X. Meng, and M. Mahoney. Quantile regression for large-scale applications. In *ICML*, pages 881–887, 2013. 1966, 1971

[YMM14] Jiyan Yang, Xiangrui Meng, and Michael W. Mahoney. Quantile regression for large-scale applications. *SIAM J. Scientific Computing*, 36(5), 2014. 1450

[You40] G. Young. Maximum likelihood estimation and factor analysis. *Psychometrika*, 6(1):49–53, 1940. 50, 1038

[YS17] Greg Yang and Samuel Schoenholz. Mean field residual networks: On the edge of chaos. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 7103–7114, 2017. 358

[YS18] Greg Yang and Sam S. Schoenholz. Deep mean field theory: Layerwise variance and width variation as methods to control gradient explosion. *ICLR open review*, 2018. 358

[YTM94] Yinyu Ye, Michael J Todd, and Shinji Mizuno. An $O(\sqrt{nL})$-iteration homogeneous and self-dual linear programming algorithm. *Mathematics of Operations Research*, 19(1):53–67, 1994. 9

[YV18] Grigory Yaroslavtsev and Adithya Vadapalli. Massively parallel algorithms and hardness for single-linkage clustering under ℓ_p distances. In *International Conference on Machine Learning*, pages 5596–5605, 2018. 1917

[YX15] Zai Yang and Lihua Xie. Achieving high resolution for super-resolution via reweighted atomic norm minimization. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 3646–3650. IEEE, 2015. 686, 763

[YZD12] Linbin Yu, Miao Zhang, and Chris Ding. An efficient algorithm for ℓ_1-norm principal component analysis. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1377–1380. IEEE, 2012. 53, 1111, 2506, 2507

[ZBH+17] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations (ICLR)*, 2017. 20, 250, 351

[ZBSG05] Yunhui Zheng, David J Brady, Michaell E Sullivan, and Bob D Guenther. Fiber-optic localization by geometric space coding with a two-dimensional gray code. *Applied optics*, 44(20):4306–4314, 2005. 33, 617

[ZCF+10] Matei Zaharia, Mosharaf Chowdhury, Michael J Franklin, Scott Shenker, and Ion Stoica. Spark: Cluster computing with working sets. *HotCloud*, 10(10-10):95, 2010. 2315

[ZCS+19] Huan Zhang, Hongge Chen, Zhao Song, Duane Boning, Inderjit S Dhillon, and Cho-Jui Hsieh. The limitations of adversarial training and the blind-spot attack. In *ICLR*. arXiv preprint arXiv:1901.04684, 2019. xxix

[ZCZG18] Difan Zou, Yuan Cao, Dongruo Zhou, and Quanquan Gu. Stochastic gradient descent optimizes over-parameterized deep relu networks. In *arXiv preprint*. https://arxiv.org/pdf/1811.08888, 2018. 1909, 2140

[ZCZJ14] Yuchen Zhang, Xi Chen, Denny Zhou, and Michael I Jordan. Spectral methods meet em: A provably optimal algorithm for crowdsourcing. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1260–1268. https://arxiv.org/pdf/1406.3824, 2014. 57, 1469

[ZF13] Anastasios Zouzias and Nikolaos M. Freris. Randomized extended kaczmarz for solving least squares. *SIAM J. Matrix Analysis Applications*, 34(2):773–793, 2013. 1927

[ZG01] Tong Zhang and Gene H. Golub. Rank-one approximation to high order tensors. *SIAM J. Matrix Analysis Applications*, 23(2):534–550, 2001. 57, 1469

[ZH05] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005. 14, 74

[Zha97] Zhengyou Zhang. Parameter estimation techniques: A tutorial with application to conic fitting. *Image and vision Computing*, 15(1):59–76, 1997. 56, 1428, 1430

[ZK16] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016. 255

[ZL17] Yi Zhou and Yingbin Liang. Critical points of neural networks: Analytical forms and landscape properties. *arXiv preprint arXiv:1710.11205*, 2017. 351

[ZLC17] Yuchen Zhang, Percy Liang, and Moses Charikar. A hitting time analysis of stochastic gradient langevin dynamics. In *COLT*. arXiv preprint arXiv:1702.05575, 2017. 17, 172

[ZLS+12] Yinqiang Zheng, Guangcan Liu, Shigeki Sugimoto, Shuicheng Yan, and Masatoshi Okutomi. Practical low-rank matrix approximation under robust

ℓ_1-norm. In *2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, June 16-21, 2012*, pages 1410–1417, 2012. 53, 1110, 1370

[ZLSD18] Jiong Zhang, Yibo Lin, Zhao Song, and Inderjit S Dhillon. Learning long term dependencies via fourier recurrent units. In *International Conference on Machine Learning (ICML)*, 2018. xxix, 339, 353, 988

[ZMKL05] Cai-Nicolas Ziegler, Sean M McNee, Joseph A Konstan, and Georg Lausen. Improving recommendation lists through topic diversification. In *Proceedings of the 14th international conference on World Wide Web*, pages 22–32. ACM, 2005. 1426

[Zol86] Vladimir M Zolotarev. *One-dimensional stable distributions*, volume 65. American Mathematical Soc., 1986. 667

[Zou12] Anastasios Zouzias. A matrix hyperbolic cosine algorithm and applications. In *International Colloquium on Automata, Languages, and Programming*, pages 846–858. Springer, 2012. 29

[ZP04] Peng Zhang and Jing Peng. Svm vs regularized least squares classification. In *ICPR (1)*, pages 176–179, 2004. 1923

[ZPB06] Yunhui Zheng, Nikos P Pitsianis, and David J Brady. Nonadaptive group testing based fiber sensor deployment for multiperson tracking. *IEEE Sensors Journal*, 6(2):490–494, 2006. 33, 617

[ZS16] Ruohan Zhang and Zhao Song. Maximum sustainable yield problem for robot foraging and construction system. In *IJCAI*, pages 2725–2731, 2016. xxix

[ZSB15] Ruohan Zhang, Zhao Song, and Dana H Ballard. Global policy construction in modular reinforcement learning. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015. xxix

[ZSD17] Kai Zhong, Zhao Song, and Inderjit S Dhillon. Learning non-overlapping convolutional neural networks with multiple kernels. *arXiv preprint arXiv:1711.03440*, 2017. xxix, 21, 25, 251, 257, 351, 548, 2140

[ZSJ+17] Kai Zhong, Zhao Song, Prateek Jain, Peter L. Bartlett, and Inderjit S. Dhillon. Recovery guarantees for one-hidden-layer neural networks. In *ICML*, 2017. xxix, 21, 25, 57, 251, 257, 351, 358, 548, 1451, 1469, 2140

[ZSJD18] Kai Zhong, Zhao Song, Prateek Jain, and Inderjit S Dhillon. Nonlinear inductive matrix completion based on one-layer neural networks. *arXiv preprint arXiv:1805.10477*, 2018. xxix

[ZW13] Syed Zubair and Wenwu Wang. Tensor dictionary learning with sparse tucker decomposition. In *Digital Signal Processing (DSP), 2013 18th International Conference on*, pages 1–6. IEEE, 2013. 1706

[ZWX+17] Shun Zheng, Jialei Wang, Fen Xia, Wei Xu, and Tong Zhang. A general distributed dual coordinate optimization framework for regularized loss minimization. *The Journal of Machine Learning Research*, 18(1):4096–4117, 2017. 13, 74

[ZWZ16] Junyu Zhang, Zaiwen Wen, and Yin Zhang. Subspace methods with local refinements for eigenvalue computation using low-rank tensor-train format. *Journal of Scientific Computing*, pages 1–22, 2016. 1711

[ZX17] Yuchen Zhang and Lin Xiao. Stochastic primal-dual coordinate method for regularized empirical risk minimization. *The Journal of Machine Learning Research*, 18(1):2939–2980, 2017. 13, 58, 74, 1470

[ZYJ17] Lijun Zhang, Tianbao Yang, and Rong Jin. Empirical risk minimization for stochastic convex optimization: $O(1/n)$-and $O(1/n^2)$-type of risk bounds. *arXiv preprint arXiv:1702.02030*, 2017. 13, 74

[ZZGM15] Xiaodong Zheng, Shanfeng Zhu, Junning Gao, and Hiroshi Mamitsuka. Instance-wise weighted nonnegative matrix factorization for aggregating partitions with locally reliable clusters. In *IJCAI*, 2015. 1044

Vita

Zhao Song was born in Xi'an, Shaanxi, China, the son of Li Song (father) and Mei Sun (mother), the grandson of Kexin Song (father's father), Weifeng Guan (father's mother), Yongzhe Sun (mother's father), Fenglin Yang (mother's mother).

Permanent address: Jinhua North Road 280, C-22-06
Xi'an, Shaanxi, China 710032

This dissertation was typeset with LaTeX[†] by the author.

[†]LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.