The Dissertation Committee for Prateek Jain
certifies that this is the approved version of the following dissertation:

Large Scale Optimization Methods for Metric and Kernel Learning

Committee:

Inderjit S. Dhillon, Supervisor

Constantine Caramanis

Joydeep Ghosh

Kristen Grauman

Raymond Mooney

Large Scale Optimization Methods for Metric and Kernel Learning

by

Prateek Jain, B.Tech., M.S.Comp.Sci.

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

December 2009

Dedicated to my parents...

Acknowledgments

It is remarkable that even a modest enterprise such as this thesis, is impossible without contributions from so many people and proper alignment of so many factors. So, I begin by thanking almighty God for making it all work out.

I would like to thank my thesis advisor Prof. Inderjit S. Dhillon for simply being a great advisor and helping me out in possibly the toughest situation of my life. I hope his passion for research and ability to think clearly and objectively about a problem have rubbed off, at least partially, on me also. I will also like to thank my committee members Prof. Constantine Caramanis, Prof. Joydeep Ghosh, Prof. Kristen Grauman, and Prof. Raymond Mooney for their advice and support. Special thanks to Kristen for helping me get started with my research and being a great role model as well as a great source of advice.

I would also like to thank all of my lab-mates in the data mining lab, especially Jason Davis, Brian Kulis , Zhengdong Lu, Suvrit Sra and Wei Tang, for helping me out with so many things including research and for making our lab Tay-137 the most fun place to work on the UT campus. A special thanks to Raghu Meka, an unofficial member of our group and my collaborator on many research and sports projects. Thanks to Chinmayi Krishnappa for proof-

reading large portions of this thesis.

While research is a lot of fun, still Austin would have been an incredibly boring place if not for all my friends here, including David Montoya, Mitul Tiwari, Misha Sra, Prince Mahajan, Sudheendra Narasimhan, Vinay Siddhavanhalli. Special thanks to my apartment mates Indrajit Roy, Pushpraj Shukla and Siddharth Chauhan for bearing with my weird idiosyncrasies and me in general for a long time.

Next, I would like to thank my sweetest and best friends Anand Sinha, Bharti Jain, Chinmayi Krishnappa, Hemanta Maji, Manku Jain, Naveen Gupta and Rishi Dhingra. I am heavily indebted to all of them for being there for me during happiness, sorrows and everything else. Thanks also to my friends from my undergraduate studies espcially my wing-mates and department-mates.

I would like to thank my whole family for imparting me all the values that I have and for making me feel so secure that I never felt scared taking up challenges. Thanks to my uncle Vinod Jain, aunt Chameli Jain, and cousins Manjari , Shivani Jain for their constant love and support. Special thanks to my sisters Swati and Prerna Jain, my brother-in-law Sanjay Patni, and my neices Soumya and Tithi Jain for making life worth living. Finally, thanks to my mom and dad for everything. I know it is impossible to thank you in words and so I put to rest this futile exercise.

Large Scale Optimization Methods for Metric and Kernel Learning

Publication No. _____

Prateek Jain, Ph.D.
The University of Texas at Austin, 2009

Supervisor: Inderjit S. Dhillon

A large number of machine learning algorithms are critically dependent on the underlying distance/metric/similarity function. Learning an appropriate distance function is therefore crucial to the success of many methods. The class of distance functions that can be learned accurately is characterized by the amount and type of supervision available to the particular application. In this thesis, we explore a variety of such distance learning problems using different amounts/types of supervision and provide efficient and scalable algorithms to learn appropriate distance functions for each of these problems.

First, we propose a generic regularized framework for Mahalanobis metric learning and prove that for a wide variety of regularization functions, metric learning can be used for efficiently learning a kernel function incorporating the available side-information. Furthermore, we provide a method for fast nearest neighbor search using the learned distance/kernel function. We show that a

variety of existing metric learning methods are special cases of our general framework. Hence, our framework also provides a kernelization scheme and fast similarity search scheme for such methods.

Second, we consider a variation of our standard metric learning framework where the side-information is incremental, streaming and cannot be stored. For this problem, we provide an efficient online metric learning algorithm that compares favorably to existing methods both theoretically and empirically.

Next, we consider a contrasting scenario where the amount of supervision being provided is extremely small compared to the number of training points. For this problem, we consider two different modeling assumptions: 1) data lies on a low-dimensional linear subspace, 2) data lies on a low-dimensional non-linear manifold. The first assumption, in particular, leads to the problem of matrix rank minimization over polyhedral sets, which is a problem of immense interest in numerous fields including optimization, machine learning, computer vision, and control theory. We propose a novel online learning based optimization method for the rank minimization problem and provide provable approximation guarantees for it. The second assumption leads to our geometry-aware metric/kernel learning formulation, where we jointly model the metric/kernel over the data along with the underlying manifold. We provide an efficient alternating minimization algorithm for this problem and demonstrate its wide applicability and effectiveness by applying it to various machine learning tasks such as semi-supervised classification,

colored dimensionality reduction, manifold alignment etc.

Finally, we consider the task of learning distance functions under no supervision, which we cast as a problem of learning disparate clusterings of the data. To this end, we propose a discriminative approach and a generative model based approach and we provide efficient algorithms with convergence guarantees for both the approaches.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The success of many learning algorithms, such as k-Nearest Neighbor and, Support Vector Machines, is critically dependent on the distance/similarity measure used to compare the input data points. A standard, but somewhat ad-hoc approach to select a distance measure is to try a few off-the-shelf distance or similarity (kernel) functions such as the Euclidean distance function or the Gaussian kernel function. However, such an approach is cumbersome and does not scale well. Moreover, for most real-world problems standard distance or similarity functions are not appropriate and fail to capture the true relationships between objects.

Recently, distance/kernel learning approaches have come forward as a more principled and data-dependent method to select a distance/kernel function for the task at hand. Numerous approaches have been proposed that attempt to learn distance/kernel functions, e.g., [29, 90, 70, 110]. Typically, these approaches assume a specific parameterization of the distance function and optimize for the parameters using the provided supervision. Examples of common parameterizations include Gaussian kernels with variable width [90], weighted sum of a few given metrics/kernels [70], and weighted sum of

eigenvectors of a Laplacian over the data [112].

Ideally, a distance/kernel learning algorithm should have the following properties: 1) flexible parameterization so that it can be applied to a variety of applications, 2) efficiency in the computation of provably optimal distance functions, 3) good generalization to unseen points, i.e., it also works in inductive settings, 4) can be used efficiently by many applications, such as sub-linear time similarity search.

Unfortunately, most of the existing distance/kernel learning approaches are lacking in at least one of the above properties. Of the existing approaches, one of the most successful approaches is to learn a Mahalanobis distance function. Mahalanobis distance functions have been shown to be successful in a number of different domains, e.g., text mining [23], computer vision [57, 45], software support [46]. They are amenable to simple and elegant formulations, have good generalization properties and wide applicability.

However, most of the Mahalanobis metric learning methods are designed for semi-supervised learning with small dimensionality and large amounts of supervision. Different forms and amounts of supervision dictate the class of distance functions that can be learned accurately. For example, if limited supervision is available, then learning a distance function with a large number of parameters is infeasible. Furthermore, in most of the real-world applications, especially in the domain of computer-vision and text-mining, the data-points are complicated data structures, such as images, text documents, and web documents. Typically, a feature representation of such data structures is high-

dimensional and is difficult to obtain. Instead, the inner product function (or kernel function) over the feature space is provided. Thus, there is a need to formulate scalable and theoretically sound distance learning methods that perform well in the provided domain and conditions, and can also perform computations implicitly in the feature (or kernel) space.

In this thesis, we explore various large scale optimization techniques for the problems of learning distance functions and kernel functions under varying amounts and types of supervision. Our primary focus is on learning *Mahalanobis* distance/kernel function in a variety of different contexts. We also explore a few other parameterizations for distance functions in Chapter 7 and 8. In the next section, we provide a broad overview of the distance/kernel learning problems considered in this thesis and our contributions.

1.1 Thesis Overview

In Chapter 3, we propose a generic regularized framework for Mahalanobis metric learning[1] that can be seen as a generalization of a variety of metric learning methods, e.g, ITML [24], LMNN [108]. Specifically, we parameterize the distance function, i.e., the metric, over the data points using a Mahalanobis metric[2], i.e., $d_W(\boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{x} - \boldsymbol{y})^T W (\boldsymbol{x} - \boldsymbol{y})$. Side-information is provided to the framework in the form of convex constraints over the Maha-

[1]Throughout this thesis, the term *metric learning* refers to Mahalanobis metric learning.

[2]Throughout this thesis, the term *Mahalanobis metric* W refers to the Mahalanobis distance function parameterized by W

lanobis metric. Examples of such constraints include commonly used pair-wise distance constraints, relative-distance constraints etc. Assuming a convex regularization function, a metric learning problem under our generic framework is a convex program and can be solved in polynomial time using standard convex programming software.

Solving for Mahalanobis metric using a standard convex programming software requires explicit representation of the feature vectors $\boldsymbol{x} \in \mathbb{R}^d$ and computes a $d \times d$ matrix W explicitly. As stated earlier, for most real-world applications d is large and explicit representation for feature vectors $\boldsymbol{x}$ is not available explicitly. Hence, direct computation of W is infeasible. However, we show that under a fairly mild assumption on the regularization function, our metric learning framework can be used for implicitly learning a metric over the feature space. Furthermore, the learned metric can be used to define a new kernel function for the given application.

One of the most important applications of metric learning is in enhancing the solution to a fundamental problem - the nearest neighbor search problem, which is critical to several machine learning problems including k-NN classification, pose estimation, and ranking. Furthermore, most real-world applications involve large databases, ruling out linear search for nearest neighbor. Hence, sub-linear time search for nearest neighbor using the learned metric is vital.

In chapter 4, we introduce a method for fast approximate similarity search with learned distance functions. We formulate randomized hash func-

tions that incorporate side-information from partially labeled data or paired constraints, thus allowing examples to be efficiently indexed according to the learned metric without resorting to a naive linear scan of all the items. We present a straightforward solution for the case of relatively low-dimensional input vector spaces, where the Mahalanobis metric (W) can be computed explicitly. We further derive a method for the fast computation of hash functions over high-dimensional datasets for which explicit input space computations are infeasible. Our method makes fast indexing accessible for numerous existing metric learning methods (e.g., [110, 10, 24]). We demonstrate the approach applied to a variety of real-world datasets. We find that our learned metrics improve accuracy relative to commonly-used metric baselines, while our hashing construction enables efficient indexing with learned distances and very large databases.

The metric learning framework we introduce in Chapter 3 provides useful distance functions for a variety of domains, and is accurate for applications where the learner can access all distance constraints at once. However, in many real-world applications, distance constraints are only available incrementally, thus necessitating methods that can perform online updates to the learned metric. Existing online algorithms offer bounds on worst-case performance, but typically do not perform well in practice as compared to their offline counterparts (i.e. ones with static distance constraints). In Chapter 5, we present a new online metric learning algorithm that updates a learned Mahalanobis metric based on LogDet regularization and gradient descent. We prove theo-

retical worst-case performance bounds, and empirically compare the proposed method against existing online metric learning algorithms. As in Chapter 3, to further boost the practicality of our approach, we develop an online locality-sensitive hashing scheme which leads to efficient updates of data structures used for fast approximate similarity search. We demonstrate our algorithm on multiple datasets and show that it outperforms relevant baselines.

Learning a Mahalanobis metric using methods described in Chapters 3 and 4 consists of learning $O(\min(d^2, n^2))$ parameters, where d is the dimensionality of the input data and n is the number of input data points. As a result, if the amount of supervision is very small then learning a full-rank Mahalanobis metric or the corresponding kernel function leads to over-fitting and poor generalization. In Chapter 3 we partially address this problem by restricting the Mahalanobis metric to be a diagonal plus low-rank matrix, thus reducing the number of parameters to linear in $O(\min(d, n))$. However, the low-rank part needs to be restricted to a pre-selected basis which is undesirable in many applications. In Chapters 6 and 7, we attempt to get rid of this requirement by making assumptions on the structure of the data.

In Chapter 6, we explore the scenario where the data points lie on a small-dimensional smooth manifold. Given pairwise (dis-)similarity constraints, we learn a kernel matrix over the data that respects the provided side-information as well as the local geometry of the data. We extend the framework introduced in Chapter 3; we jointly model the metric/kernel over the data along with the underlying manifold. Furthermore, we show that for

some important parameterized forms of the underlying manifold model, we can estimate the model parameters and the kernel matrix efficiently. Our resulting algorithm is able to incorporate local geometry into the metric learning task; at the same time it can handle a wide class of constraints. Finally, our algorithm is fast and scalable, is able to exploit the low dimensional manifold structure and does not require semi-definite programming. We demonstrate wide applicability and effectiveness of our framework by applying to various machine learning tasks such as semi-supervised classification, colored dimensionality reduction, manifold alignment etc. On each of the tasks our method performs competitively or better than the respective state-of-the-art method.

In Chapter 7, we assume that the data points lie in a low-dimensional subspace, i.e., the learned kernel matrix is low-rank. This implies that the number of parameters to be learned is reasonably small and can be learned accurately using a small amount of supervision. We handle the low-rank kernel learning problem in both transductive and inductive settings. For the transductive setting, we propose a novel low-rank kernel matrix learning problem that incorporates the provided distance constraints. For the inductive setting, we use our metric learning framework introduced in Chapter 3 to learn a low-parameter kernel *function*[3].

In both transductive and inductive settings, our formulation reduces to the affine constrained rank minimization problem, which in itself is a problem

[3]Throughout this thesis, the term *rank k-kernel function* refers to a kernel function which leads to at most rank k-kernel matrix for any set of points

of immense interest with numerous applications in fields like optimization, computer vision, and control theory. We show that the rank minimization with affine constraints is an NP-hard problem and assuming $P \neq NP$, does not even admit a logarithmic approximation. Furthermore, we propose two online learning based approaches for rank minimization - our first algorithm is a multiplicative update method based on a generalized experts framework [84, 4], while our second algorithm is a novel application of the online convex programming framework [113]. In the latter, we flip the role of the decision maker by making the decision maker search over the constraint space instead of feasible points, as is usually the case in online convex programming. A salient feature of our online learning approach is that it allows us to give provable approximation guarantees for the rank minimization problem over polyhedral sets. We evaluate the performance of our methods for low-rank kernel learning on UCI datasets. On all the datasets, our algorithms improve the accuracy of the baseline kernel while also significantly decreasing the rank.

In Chapter 8, we study the problem of unsupervised distance learning. We observe that a distance function is used to capture a particular semantic notion in the data. When no supervision is provided, we formulate the problem as that of learning all the distance functions associated with the dominant semantics – a hard problem. We relax the problem to that of finding all the clusterings associated with the dominant distance functions or semantics in the data.

We propose two new approaches for the problem of recovering disparate

clusterings. In the first approach we aim to find good clusterings of the data that are also *decorrelated* with one another. To this end, we give a new and tractable characterization of decorrelation between clusterings, and present an objective function to capture it. We provide an efficient "decorrelated" k-means type algorithm to minimize this objective function and provide convergence guarantees. In the second approach, we model the data as a sum of mixtures and associate each mixture with a clustering. This approach leads us to the problem of learning a convolution of mixture distributions. Though the latter problem can be formulated as one of factorial learning [34, 50, 87], the existing formulations and methods do not perform well on many real high-dimensional data sets. We propose a new regularized factorial learning framework that is more suitable for capturing the notion of disparate clusterings in modern, high-dimensional data sets. We propose a generalized EM-algorithm to find out parameters for new regularized factorial learning framework and prove convergence for the same. The resulting algorithm does well in uncovering multiple clusterings, and is much improved over existing methods. We evaluate our methods on two real-world data sets - a music data set from the text mining domain, and a portrait data set from the computer vision domain. Our methods achieve a substantially higher accuracy than existing factorial learning as well as traditional clustering algorithms.

1.2 Summary

We introduce a generic framework for Mahalanobis distance learning (metric learning) that generalizes most of the existing convex Mahalanobis distance learning formulations. We then extend our framework to handle large dimensional datasets and also to implicitly learn metrics, or equivalently kernel functions, in feature space. We then propose a method for fast similarity search for metric learned by our regularized framework.

Additionally, we consider a few specific distance learning problems where our metric learning framework cannot be applied directly. In particular, we consider the problem of online metric learning, geometric metric learning, low-rank metric/kernel learning and unsupervised distance learning. For each of the problems, we propose theoretically justified formulations and provide efficient algorithms to solve the corresponding problem.

1.3 Notation

We adhere to the following notation throughout this thesis. Lowercase bold letters ($\boldsymbol{x}$) denote vectors, $\boldsymbol{x}^T\boldsymbol{y}$ denotes inner product between two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$, and $\|\cdot\|_p$ denotes L_p norm. Matrices are denoted by capital letters (X). $\|\cdot\|_F$ denotes Frobenius norm, $\|\cdot\|_p$ denotes vector induced p-norm for matrices. $\mathrm{Tr}(\cdot)$ denotes the trace of a matrix and A^T denotes transpose of A. The term *Mahalanobis metric* W refers to the Mahalanobis distance function parameterized by W and the term *metric learning* refers to Mahalanobis distance learning.

Chapter 2

Background Material and Related Work

In this chapter, we cover background material relevant to the thesis proposal, including Mahalanobis metrics, locality sensitive hashing, the multiplicative weights update algorithm, and online convex programming. We also briefly review related work.

2.1 Parameterized Mahalanobis Metrics

Given n points $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$, with all $\boldsymbol{x}_i \in \Re^d$, positive-definite (p.d.) $d \times d$ matrix A parameterizes the squared Mahalanobis distance:

$$d_A(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\boldsymbol{x}_i - \boldsymbol{x}_j)^T A (\boldsymbol{x}_i - \boldsymbol{x}_j), \tag{2.1}$$

for all $i, j = 1, \ldots, n$. Note that the Mahalanobis metric is a linear mapping of data, i.e., $\boldsymbol{x}' \rightarrow A^{1/2}\boldsymbol{x}$ and it is induced by generalazied inner product (kernel) that measures the pairwise similarity: $s_A(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{x}_i^T A \boldsymbol{x}_j$.

Given a set of inter-point distance constraints, one can directly learn a matrix A to yield a distance measure that is more accurate for a given classification or clustering problem. Many methods have been proposed for Mahalanobis metric learning [110, 10, 24]. Of particular interest is the information-

theoretic metric learning method of [24] because it is easier to implement and faster than most of the other existing metric learning methods.

2.1.1 Information-Theoretic Metric Learning

Given an initial $d \times d$ p.d. matrix A_0 specifying prior knowledge about inter-point distances, the learning task is posed as an optimization problem that minimizes the LogDet divergence between matrices A and A_0, subject to a set of constraints specifying pairs of examples that are similar or dissimilar. In semi-supervised multi-class settings, the constraints are taken directly from the provided labels: points in the same class must be similar, points in different classes are constrained to be dissimilar.

To compute A, the LogDet divergence is minimized while enforcing desired constraints:

$$\begin{aligned} \min_{A \succeq 0} \quad & D_{\ell \mathrm{d}}(A, A_0) \\ \text{s. t.} \quad & d_A(\boldsymbol{x}_i, \boldsymbol{x}_j) \leq u \quad (i,j) \in \mathcal{S}, \\ & d_A(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq \ell \quad (i,j) \in \mathcal{D}, \end{aligned} \tag{2.2}$$

where $D_{\ell \mathrm{d}}(A, A_0) = \mathrm{tr}(AA_0^{-1}) - \log\det(AA_0^{-1}) - d$, $\mathcal{S}$ and $\mathcal{D}$ are sets containing pairs of points constrained to be similar and dissimilar, respectively, and ℓ and u are large and small values, respectively (defined below).

Computing the optimal solution to (2.2) involves repeatedly projecting the current solution onto a single constraint, via the update:

$$A_{t+1} = A_t + \beta_t A_t (\boldsymbol{x}_{i_t} - \boldsymbol{x}_{j_t})(\boldsymbol{x}_{i_t} - \boldsymbol{x}_{j_t})^T A_t, \tag{2.3}$$

where $\boldsymbol{x}_{i_t}$ and $\boldsymbol{x}_{j_t}$ are the constrained data points for iteration t, and β_t is a projection parameter computed by the algorithm.

When the dimensionality of the data is very high, one cannot explicitly work with A, and so the update in (2.3) cannot be performed. However, one may still implicitly update the Mahalanobis matrix A via updates in kernel space for an equivalent kernel learning problem in which $K = X^T A X$ for $X = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n]$. If K_0 is an input kernel matrix for the data, the appropriate update is:

$$K_{t+1} = K_t + \beta_t K_t (\boldsymbol{e}_{i_t} - \boldsymbol{e}_{j_t})(\boldsymbol{e}_{i_t} - \boldsymbol{e}_{j_t})^T K_t, \tag{2.4}$$

where the vectors $\boldsymbol{e}_{i_t}$ and $\boldsymbol{e}_{j_t}$ refer to the i_t-th and j_t-th standard basis vectors, respectively, and the projection parameter β_t is the same as in (2.3) (see [24]). Note that it is possible for the set of examples involved in constraints to be a superset of the set of examples in the input kernel.

2.2 Locality Sensitive Hashing

A family of locality-sensitive hash functions $\mathcal{F}$ is a distribution of functions where the following holds: for any two objects $\boldsymbol{x}$ and $\boldsymbol{y}$,

$$\Pr_{h \in \mathcal{F}}[h(\boldsymbol{x}) = h(\boldsymbol{y})] = sim(\boldsymbol{x}, \boldsymbol{y}), \tag{2.5}$$

where $h(x)$ is a hash function mapping vector $\boldsymbol{x}$ to a bit value, $sim(\boldsymbol{x}, \boldsymbol{y})$ is some similarity function defined on the collection of objects [21, 53]. A k-bit hash key can be defined using hash functions $< h_1, h_2, \ldots, h_k >$ sampled from

$\mathcal{F}$ as:

$$g(\boldsymbol{x}) = [h_1(\boldsymbol{x}), h_2(\boldsymbol{x}), \ldots, h_k(\boldsymbol{x})].$$

When $g(\boldsymbol{x}) = g(\boldsymbol{y})$, $\boldsymbol{x}$ and $\boldsymbol{y}$ collide in the hash table. Because the probability that two inputs collide is equal to the similarity between them, highly similar objects are indexed together in the hash table with high probability.

In the hashing table construction phase, each database example is hashed into l different hash tables using random permutations of the bits of the hash key given by $g(\boldsymbol{x})$. Each list of permuted hash keys is sorted lexicographically to form l sorted orders. For a query point, k-bit hash key is computed using g and is indexed into each sorted order using binary search, thus retrieving at most $2l$ examples. This technique guarantees that to retrieve $(1 + \epsilon)$ approximate neighbors at most $l = O(N^{1/(1+\epsilon)}$ examples need to be looked up[21].

2.3 Multiplicative Weights Update Algorithm

The Multiplicative Weights Update algorithm (MW algorithm) is an extension of the Winnow algorithm [74] for a generalized experts framework [4]. This framework was implicitly used by [84] for solving several fractional packing and covering problems. This approach was formalized and extended to semi-definite programs in [3].

In the generalized experts (GE) framework there is a set of n experts, a set of events $\mathcal{E}$, and a penalty matrix M such that the i-th expert incurs a

penalty of $M(i,j)$ for an event $j \in \mathcal{E}$. The penalties are assumed to be bounded and lie in the interval $[-\rho, \rho]$ for a fixed $\rho > 0$. At each time step $t = 1, 2, \ldots,$ an adversary chooses an event $j^t \in \mathcal{E}$ so that the i-th expert incurs a penalty of $M(i, j^t)$. The goal in the GE framework is to formulate a *prediction algorithm* that chooses a distribution $\mathcal{D}^t = (p_1^t, \ldots, p_n^t)$ on the experts at time step t, so that the total expected loss incurred by the prediction algorithm is not much worse than the total loss incurred by the best expert. Formally, the goal of the prediction algorithm is to minimize

$$\sum_{t=1}^{T} \sum_{l=1}^{n} p_l^t M(l, j^t) - \min_i \sum_{t=1}^{T} M(i, j^t).$$

Note that the distribution in round t, $\mathcal{D}^t$, must be chosen without knowledge of the event j^t chosen at time step t. At every step t, the MW algorithm has a weight w_i^t assigned to expert i, and sets the distribution $\mathcal{D}^t = (p_1^t, \ldots, p_n^t)$, where $p_i^t = w_i^t / \sum_j w_j^t$. The weights at time step $t+1$ are updated as:

$$w_i^{t+1} = \begin{cases} w_i^t (1-\delta)^{M(i,j^t)/\rho} & \text{if } M(i,j^t) \geq 0, \\ w_i^t (1+\delta)^{M(i,j^t)/\rho} & \text{if } M(i,j^t) < 0. \end{cases}$$

It can be shown that the multiplicative updates algorithm achieves bounded expected loss.

Theorem 1 (Corollary 4 of [4])**.** *Suppose that for all i and $j \in \mathcal{E}$, $M(i,j) \in [-\rho, \rho]$. Let $\epsilon > 0$ be an error parameter and let $\delta = \min\{\frac{\epsilon}{4\rho}, \frac{1}{2}\}$, and $T = \frac{16\rho^2 \ln n}{\epsilon^2}$. Then, the following bound holds for the average expected loss of the MW algorithm*

$$\frac{\sum_{t=1}^{T} \sum_{l=1}^{n} p_l^t M(l, j^t)}{T} \leq \epsilon + \frac{\sum_t M(k, j^t)}{T}, \ \forall k.$$

2.3.1 Online Convex Programming

The online convex programming (OCP) framework [113, 60, 48] models various useful online learning problems like industrial production and network routing. The OCP framework involves a fixed convex set K and a sequence of unknown cost functions $f_1, f_2, \ldots : K \to \mathbb{R}$. At each time step t, a decision maker must choose a point $z_t \in K$ and incurs a cost $f_t(z_t)$. However, the choice of z_t must be made with the knowledge of $z_1, \ldots, z_{t-1}$ and $f_1, \ldots, f_{t-1}$ alone i.e., without knowing f_t. The total cost incurred by the algorithm after T steps equals $\sum_t f_t(z_t)$. The objective in OCP is to minimize the *regret* as defined below:

$$R(T) = \sum_{t=1}^{T} f_t(z_t) - \min_{z \in K} \sum_{t=1}^{T} f_t(z). \tag{2.6}$$

[113] has shown that in the case when the functions f_t are convex and differentiable with bounded gradient, one can achieve a regret of $O(\sqrt{T})$. Let $\|K\| = \max_{z_1, z_2 \in K} \|z_1 - z_2\|$ and $G = \max_{z \in K, t \in \{1,\ldots\}} \| \bigtriangledown f^t(z)\|$, where $\|\cdot\|$ denotes the Euclidean norm (or Frobenius norm if the set K is defined over matrices). Also, assume that $\bigtriangledown f^t$ can be evaluated efficiently at any given point z. Under the above assumptions [113] proposed a Generalized Infinitesimal Gradient Ascent algorithm which achieves a regret of $O((G^2 + \|K\|^2)\sqrt{T})$. The function GIGA in Algorithm 3 describes a slightly modified version of [113]'s algorithm that achieves the following improved regret bound.

Theorem 2 (Adaptation of Theorem 1 of [113])**.** *The following bound holds*

for the regret of the GIGA sub-routine of Algorithm 3 after T rounds,

$$R(T) \leq G \cdot \|K\|\sqrt{T} \tag{2.7}$$

Proof sketch: Using the modified step-size in Algorithm 2, the theorem follows from Zinkevich's original proof. □

2.4 Related Work

We briefly review related work to the research presented in this proposal.

Distance Learning: Recently, a number of techniques have been proposed for distance learning, including several techniques to learn a Mahalanobis metric [110, 10, 24], and methods to learn example-specific local distance functions [29]. Embedding functions can be useful both to capture (as closely as possible) a desired set of provided distances between points, as well as to provide an efficient approximation for a known but computationally expensive distance function of interest [6, 42]. In contrast to learned metrics, such geometric embeddings are meant to mirror a fixed distance function and do not adapt to reflect supervised constraints.

Kernel Learning: Existing kernel learning methods can be broadly divided into two categories. The first category includes primarily task-dependent approaches, where the intrinsic structure in the data is assumed, and the goal is to maximally tune the kernel to the provided side-information for the given

task, *e.g.*, class labels for classification, must (cannot)-link constraints for semi-supervised clustering. Prominent methods include multiple kernel learning [70], hyper-kernels [83], hyper-parameter cross validation [90], etc. In Chapter 3, we present a Mahalanobis metric based framework for learning kernel functions, which can then be extended to learn low-rank kernels. A drawback of this approach is that it is primarily based on "local" kernels (e.g., the RBF kernel), and do not exploit the geometry of the unlabeled data.

The other category of kernel learning methods consist of data-dependent approaches, which explicitly model the geometry of the data, *e.g.*, underlying *manifold* structure. These methods appear in both unsupervised and semi-supervised learning scenarios. For the unsupervised case, [109] proposed a method to recover the underlying low dimensional manifold by learning a kernel over it. More generally, [13] show that a large class of manifold learning methods are equivalent to learning certain types of kernels. For the semi-supervised setting, data-dependent kernels are used to enforce smoothness on a graph or a similar structure composed from *all* of the data. Like in the unsupervised case, the kernel captures the manifold and/or cluster structure of the data, and after integrated a regularized classification model, often provides good generalization performance [94, 18].

Our geometry-aware kernel learning method (see Chapter 6) combines the two kernel learning paradigms, thereby exploiting the geometry of the data while retaining the task-specific feature. Related work in this direction is limited and largely focuses on learning parameters for a specific family

of data-dependent kernels, *e.g.*, spectral kernels [112, 69]. In comparison, our method is based on a non-parametric information-theoretic metric/kernel learning method and is more flexible. Furthermore, existing methods are typically designed for a particular application only, *e.g.*, semi-supervised classification, and are not able to handle different type of constraints, such as distance constraints.

Low-rank Kernel Learning: For large scale distance learning, low-rank kernel learning algorithms are of great importance as they help in scaling the existing kernel learning methods [70] to large data sets. [8] introduced a low-rank kernel learning method where they use incomplete Cholesky factorization to relax the rank constraint. [68] introduces a spectral Bregmann divergences based objective function where the objective function itself constrains the final kernel to have the same rank as the input kernel K_0. A disadvantage of this method is that the initial kernel K_0 needs to be low-rank, while in practice this is often not the case. Also, most of the existing low-rank kernel learning methods are restricted to transductive setting. In Chapter 7, we introduce kernel learning methods for both transductive and inductive settings. Furthermore, our methods do not require the initial kernel to be low-rank.

Our low-rank kernel learning formulations lead to the problem of rank minimization over intersection of polyhedral sets and a small number of convex constraints. Most existing methods for rank minimization over convex sets are based on relaxing the non-convex rank function to a convex function, e.g., the

trace-norm [26, 85] or the logarithm of the determinant [27]. Unfortunately, these heuristics do not have any guarantees on the quality of the solution in general. Recently, there a number of methods have been proposed for solving the affine constrained rank minimization problem exactly [85, 79, 72, 38], most of which extends the corresponding techniques for compressed sensing to rank minimization [20, 31, 101, 16]. However, these methods assume a *restricted isometry property* on the affine constraints matrix which typically does not hold for the constraints matrix encountered in kernel learning. Furthermore, most of the existing methods are shown to be optimal for only linear equality constraints. Thus these approaches are limited in its applicability and it is not clear how to extend it to a larger class of rank minimization problems. We demonstrate empirically that our online learning based algorithms (see Chapter 7) for rank minimization outperform the trace-norm relaxation based method [26, 85]. We also remark that minimizing the trace-norm is computationally expensive, which further limits its applicability.

Several specific instances of the general low-rank matrix approximation problems have been widely researched in the machine learning community. Examples include low-rank kernel learning, SDE, sparse PCA and NNMA. Most methods for these problems can be broadly grouped into the following two categories: a) methods which drop the rank constraint and use the top k eigenvectors of the solution to the relaxed optimization problem e.g., [109]; b) methods which factor the matrix X in RMP into AB^T and optimize the resultant non-convex problem e.g., [71, 62]. However, typically these methods

do not have any provable guarantees.

Fast Similarity Search: In order to efficiently index multi-dimensional data, data structures based on spatial partitioning and recursive hyperplane decomposition have been developed, e.g. $k-d$-trees [28] and metric trees [103]. Due to the particular importance of indexing local patch features, several tree-based strategies have also been proposed [11, 82] in the vision community. Some such data structures support the use of arbitrary metrics. However, while their expected query time requirement may be logarithmic in the database size, selecting useful partitions can be expensive and requires good heuristics; worse, in high-dimensional spaces all exact search methods are known to provide little query time improvement over a naive linear scan [53].

As such, researchers have considered the problem of *approximate* similarity search, where a user is afforded explicit tradeoffs between the guaranteed accuracy versus speed of a search. Several randomized approximate search algorithms have been developed that allow high-dimensional data to be searched in time sub-linear in the size of the database, notably the locality-sensitive hashing (LSH) methods of [53, 21]. Data-dependent variants of LSH have been proposed: the authors of [32] select partitions based on where data points are concentrated, while in [92] boosting is used to select feature dimensions that are most indicative of similarity in the parameter space. This tunes the hash functions according to the estimation problem of interest; however, indexed examples must be sorted according to the input space (non-learned) distance.minimas

While randomized algorithms such as LSH have been employed heavily in various fields to mitigate the time complexity of identifying similar examples [91], their use has been restricted to generic measures for which the appropriate hash functions are already defined; that is, direct application to learned metrics was not possible. In Chapter 4, we devise a method that allows knowledge attained from partially labeled data or paired constraints to be incorporated into the hash functions. Our algorithm is theoretically sound: there is provably no additional loss in accuracy relative to the learned metric beyond the quantifiable loss induced by the approximate search technique.

Disparate Clustering: For disparate clustering, most of the existing work has been in the semi-supervised setting. The semi-supervised clustering problem of finding a clustering consistent with a given set of constraints has been extensively studied ([105, 110, 14]). This approach has been applied to the problem of recovering multiple clusterings by providing appropriate constraints. Must-link and cannot-link constraints have been extensively used for semi-supervised clustering ([105, 106, 14]). Recently, Davidson et al.[22] proposed an efficient incremental algorithm for must-link and cannot-link constraints. An alternative approach to the problem is taken by [9, 41, 40] where it is assumed that a clustering of the data is given and the objective is to find a clustering different from the given one. Our work for disparate clustering (see Section 8) differs from the above approaches in that our methods for discovering the disparate clusterings are completely unsupervised.

A supervised approach to the related problem of learning hidden two-

factor structures from the observed data was suggested in [100]. Their method, named Separable Mixture Model (SMM), models the data using a bilinear function of the factors and can also be used for obtaining two clusterings of the data. An advantage of our methods for disparate clustering over SMM is that our methods are unsupervised compared to the supervised approach of SMM. Also, our model can be extended to more than two factors, whereas it is unclear how SMM could be extended to a data generated from more than two-factors.

One of our approaches ("sum of parts" approach), introduced in Section 8.1.2, is closely related to the factorial learning problem where each data point is assumed to be generated by combining multiple factors. Ghahramani[34] introduced a novel architecture named co-operative vector quantization (CVQ), in which a set of multiple vector quantizers (VQ) combine linearly to generate the input data. However, a drawback of CVQ is that it can have multiple solutions. Many of these solutions give poor results for the problem of discovering disparate clusterings, especially on our real-world applications. Also, CVQ can be seen as a special case of our model. Another recent model related to factorial learning is multiple cause vector quantization (MCVQ) (Ross and Zemel [87]). In MCVQ it is assumed that the dimensions of the data can be separated into several disjoint factors, which take on values independently of each other. The factors are then modeled using a vector quantizer as in CVQ. However, MCVQ also faces the same drawbacks of CVQ - existence of multiple solutions - which leads to poor performance for our application of discovering

disparate clusterings.

The problem of learning convolutions of distributions that forms the basis of our "sum of parts" approach (see Section 8.1.2) has been considered in the statistics community - see for instance [30], [89], [88]. However, these methods deal with learning convolutions of simple distributions like binomial, Gaussian and Poisson, and do not consider mixtures of distributions. A fundamental problem with learning a convolution of Gaussians, as mentioned in [89], is that the problem is not well-defined - there exist many solutions to the learning problem. We face a similar problem in the M-step of our algorithm for learning the convolution of mixtures of Gaussians, where the maximum likelihood estimation has multiple solutions. We deal with this issue by regularizing the solution space in a way suitable for the purpose of recovering disparate clusterings so that the problem becomes well-posed.

We emphasize that though we state the problem of recovering disparate clusterings as one of learning independent components from the data, the problem we address is completely different from that of independent component analysis (ICA) [52]. ICA tries to separate a multivariate signal into independent additive univariate signals, whereas in our problem we try to decompose the signal into independent multivariate signals, each of which may have high correlation between its different dimensions.

Chapter 3

Metric and Kernel Learning: A Generic Framework

In this chapter, we introduce a generic framework for metric learning that encompasses various existing metric learning methods [24, 108, 93]. Our framework learns a Mahalanobis distance function that is parameterized by a positive semi-definite matrix W[1]:

$$d_W(\boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{x} - \boldsymbol{y})^T W (\boldsymbol{x} - \boldsymbol{y}).$$

Mahalanobis distance functions are a powerful class of distance functions and have been shown to be successful in a variety of domains, e.g., computer vision [57], text mining [23], software analysis [46]. Recently, numerous methods have been proposed for the task of Mahalanobis metric learning [108, 110, 36, 93]. These methods work by exploiting distance information that is intrinsically available in many learning settings. Our metric learning framework incorporates the available distance information as convex constraints on the Mahalanobis metric W, and can handle a number of different types of distance constraints. To avoid over-fitting, and to ensure good generalization

[1]Throughout this thesis, the term *Mahalanobis metric* W refers to the Mahalanobis distance function parameterized by W.

bounds, a regularization function can be specified. Furthermore, we extend our framework to handle high-dimensional data where learning full Mahalanobis metric W with a quadratic number of parameters is not feasible. To this end, we restrict the learned metric to be a low-rank plus diagonal matrix and show that under mild conditions, such a metric can be learned efficiently.

Another important problem that we address in this chapter is that of kernel (or similarity) function learning. Kernel methods have been successful in many machine learning problems. The basic idea behind kernel methods is to use the inner product over a feature space rather than explicitly use the feature space vectors. This is particularly useful when: 1) feature representation of the data points may not be available or is high dimensional, or 2) the data set has non-linear decision boundaries.

Typically, the success of a kernel method is heavily dependent on the kernel function that is selected. Unfortunately, most existing kernel function learning methods are either restricted to the transductive setting (i.e. test points are known at the time of training) or have a specific parametric form. In this chapter, we show that Mahalanobis distance functions can also be used naturally and efficiently for the problem of kernel function learning. Our learned kernel function is of the form $k(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{x}^T W \boldsymbol{y}$, and can be computed efficiently using an initial kernel function.

In summary, the contributions of this chapter are:

- We introduce a generic regularized framework for the problem of Ma-

halanobis metric learning. Our framework can handle a variety of side-information and different regularization functions, and guarantees that the optimal metric can be obtained in a polynomial number of time steps.

- We extend our framework to handle high dimensional metric learning where learning a full Mahalanobis metric is expensive. Instead, we restrict our metric to have a small number of parameters and show that under certain mild conditions, the metric can be learned efficiently.

- We show that for a large class of regularization functions, our framework admits implicit computation of the metric in kernel space. Consequently, our method can be used for learning kernel functions while incorporating the provided side-information. Moreover, for large training databases, our methods can be scaled by restricting the learned kernel function to a small number of basis points.

- We show that a variety of existing metric learning methods are specific instances of our framework, and hence, can be kernelized efficiently (i.e. computed implicitly in kernel space). Furthermore, by restricting the learned kernel to a small basis set, we also improve the scalability properties of these methods.

Most of the material presented in this chapter is based on our work [55, 24].

3.1 A Generic Framework for Metric Learning

Given a set of points $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$, such that $\boldsymbol{x}_i \in \mathbb{R}^d$, the task it to learn a Mahalanobis distance function parameterized by a positive definite matrix matrix $W \in \mathbb{S}_+^{d \times d}$:

$$d_W(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\boldsymbol{x}_i - \boldsymbol{x}_j)^T W (\boldsymbol{x}_i - \boldsymbol{x}_j). \tag{3.1}$$

We assume that the side-information is provided in the form of convex constraints over the Mahalanobis matrix W. One of the most common form of side-information is distance relationships between training points, e.g., pairwise distance constraints, relative distance constraints etc. Another commonly available side-information is class label information that can be incorporated in our framework as distance constraints or non-parametric probability estimation constraints.

Given a set of convex constraints specifying the available side-information, the metric learning problem is to learn a positive-definite matrix W parameterizing the corresponding Mahalanobis distance function (3.1). To avoid overfitting to the provided side-information and guarantee good generalization, we introduce a regularization on W which leads to the following optimization problem:

$$\begin{aligned} &\min_W \ \mathrm{Tr}(f(W)) \\ &\text{s.t. } g_i(X^T W X) \leq b_i, \ 1 \leq i \leq m, \\ &\quad W \succeq 0, \end{aligned} \tag{3.2}$$

where (g_i, b_i) specifies the available side-information and $f : \mathbb{R}^{d\times d} \to \mathbb{R}^{d\times d}$ is a regularization function. Now, if f and g_i's are convex functions such that a sub-gradient of f and of each g_i is computable efficiently (in time poly(d)) then the above problem is solvable in polynomial time using standard convex programming methods such as the ellipsoid method [44].

Next, we give a few examples of the different types of side-information that can be handled in our framework:

1. **Pair-wise distance/similarity constraints**: Pairwise distance/similarity constraints is one of the most commonly available side-information in the context of metric learning. For example, in the problem of semi-supervised clustering, points are constrained to be either similar (pairwise distance should be relatively small) or dissimilar (pairwise distance should be larger). In fully supervised settings, constraints can be inferred so that points in the same class have small distances (or high similarity) and points in the different class have large distances. A pairwise distance constraint between a pair of points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ can be formulated as:

$$\pm\mathrm{Tr}(X^T W X(\boldsymbol{e}_i - \boldsymbol{e}_j)(\boldsymbol{e}_i - \boldsymbol{e}_j)^T) \leq b,$$

where $\boldsymbol{e}_i$ is the i-th standard basis vector. Similarly, a pairwise similarity constraint is specified by:

$$\pm\mathrm{Tr}(W X \boldsymbol{e}_i \boldsymbol{e}_j^T X) \leq b,$$

2. **Relative distance/similarity constraints**: Relative distance/similarity constraints is another popular class of supervision. For example, in information retrieval settings, relative distance constraints between triplets of points can be gathered through click-through feedback. In fully supervised settings, points in the same class should have smaller distance compared to the points in different classes. Also, relative distance similarity constraints are useful in the area of psychology and market analysis where it is hard to assign a distance/similarity value between two points but it is easier to compare the distance between a triplet of points. Relative distance constraints for triplet $(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{x}_k)$ is specified using:

$$\pm \mathrm{Tr}(X^T W X (\boldsymbol{e}_i - \boldsymbol{e}_j)(\boldsymbol{e}_i - \boldsymbol{e}_j)^T - (\boldsymbol{e}_i - \boldsymbol{e}_k)(\boldsymbol{e}_k - \boldsymbol{e}_k)^T) \leq b.$$

3. **Non-parametric probability estimation constraints**: Non-parametric probability estimation constraints are mostly used in the supervised/semi-supervised setting [54], where conditional probability of a class given a data point $(p(c|x))$ is estimated using Parzen's window method. Such constraints are specified using:

$$p(c|x) = \pm \frac{\sum_{i \in c} \mathrm{Tr}(W \boldsymbol{x} \boldsymbol{x}_i^T)}{\sum_{t=1}^{C} \sum_{j \in t} \mathrm{Tr}(W \boldsymbol{x} \boldsymbol{x}_j^T)} \geq b,$$

or equivalently,

$$\mathrm{Tr}\left(\boldsymbol{x}^T W \left(\pm \sum_{i \in c} \boldsymbol{x}_i - b \sum_{t=1}^{C} \sum_{j \in t} \boldsymbol{x}_j \right)^T \right) \geq 0.$$

3.1.1 Relations to Existing Metric Learning Methods

Our metric learning framework generalizes almost all the existing convex metric learning problem formulations. Here, we give examples of a few such formulations.

- **Information Theoretic Metric Learning (ITML)**: Davis et al. [24] proposed the following metric learning problem formulation:

$$\begin{aligned} \min_{W \succeq 0} \ & \mathrm{Tr}(WW_0^{-1}) - \log\det(WW_0^{-1}), \\ s.t. \ & d_W(\boldsymbol{x}_i, \boldsymbol{x}_j) \leq b_{ij}, \ (i,j) \in \mathcal{S}, \\ & d_W(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq b_{ij}, \ (i,j) \in \mathcal{D}, \end{aligned}$$

  where $\mathcal{S}$ and $\mathcal{D}$ specify pairs of similar and dissimilar points respectively. Clearly, ITML is an instantiation of our framework with the regularization function $f(W) = W_0^{-1/2} W W_0^{-1/2} - \log(W_0^{-1/2} W W_0^{-1/2})$ and linear pairwise distance constraints.

- **Mahalanobis Metric for Clustering (MMC)**: In their seminal work, Xing et al. [110] introduced the following problem:

$$\begin{aligned} \max_{W} \ & \sum_{ij} (1 - y_{ij}) \sqrt{(\boldsymbol{x}_i - \boldsymbol{x}_j)^T W (\boldsymbol{x}_i - \boldsymbol{x}_j)}, \\ \text{s.}t. \ & \sum_{ij} y_{ij} (\boldsymbol{x}_i - \boldsymbol{x}_j)^T W (\boldsymbol{x}_i - \boldsymbol{x}_j) \leq 1, \\ & W \succeq 0, \end{aligned}$$

where y_i is the class label for $\boldsymbol{x}_i$, $y_{ij} = 1$ if $y_i = y_j$ and 0 otherwise. A variational formulation of the above problem is:

$$\begin{aligned}
&\max_{W,t} \; t, \\
&\text{s.}t. \; \sum_{ij} y_{ij}(\boldsymbol{x}_i - \boldsymbol{x}_j)^T W(\boldsymbol{x}_i - \boldsymbol{x}_j) \leq 1, \\
&\quad \sum_{ij}(1 - y_{ij})\sqrt{(\boldsymbol{x}_i - \boldsymbol{x}_j)^T W(\boldsymbol{x}_i - \boldsymbol{x}_j)} \geq t, \\
&\quad W \succeq 0.
\end{aligned}$$

For fixed t, clearly the above problem is an instance of (3.2) with $f(W)$ as a constant function. Similarly, Large Margin Nearest Neighbor (LMNN) [108] and Maximally Collapsing Metric Learning (MCML) [36] can also be seen as instantiations of our generic framework with constant regularization function f.

- **Relevant Component Analysis (RCA)**: RCA [10] computes the Mahalanobis metric W in closed form using:

$$W^{-1} = \frac{1}{n}\sum_{l=1}^{L}\sum_{i\in\Omega_l}(\boldsymbol{x}_i - \mu_l)(\boldsymbol{x}_i - \mu_l)^T, \tag{3.3}$$

where Ω_l is a chunklet or sub-class membership assignment and μ_l is the mean of a chunklet Ω_l. A chunklet is a subset of a class; all the points in a chunklet belong to the same class, but points in different chunklets need not belong to different classes. Note that, W computed using (3.3)

is the optimal solution to the following problem:

$$\begin{aligned} \min_{W} \ & -\log\det W, \\ \text{s.t. } & \frac{1}{n}\sum_{l=1}^{L}\sum_{i\in\Omega_l}(\boldsymbol{x}_i-\mu_l)^T W(\boldsymbol{x}_i-\mu_l)\leq 1, \\ & W\succeq 0. \end{aligned} \tag{3.4}$$

Furthermore, μ_l can be written as a linear combination of $\boldsymbol{x}_i$'s. Hence, the first constraint in (3.4) can be written as a convex function of X^TWX. Therefore, RCA can also be seen as an instantiation of our metric learning framework with regularization function $f(W) = -\log\det W$.

- **Pseudo Online Metric Learning (POLA)**: Shalev-Shwartz et al. [93] proposed the following metric learning formulation:

$$\begin{aligned} \min_{W} \ & \|W\|_F^2, \\ \text{s.t. } & y_{ij}(b-(\boldsymbol{x}_i-\boldsymbol{x}_j)^T W(\boldsymbol{x}_i-\boldsymbol{x}_j))\geq 1, \quad \forall (i,j)\in\mathcal{P}, \\ & W\succeq 0, \end{aligned} \tag{3.5}$$

where $y_{ij} = 1$ if $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are similar, and $y_{ij} = -1$ if $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are dissimilar. $\mathcal{P}$ is a set of pairs of points with known distance constraints. Clearly, POLA is an instantiation of our metric learning framework with $f(W) = \|W\|_F^2$ and the side-information available in the form of pairwise distance constraints.

3.2 High-dimensional Metric Learning

In Section 3.1, we proposed a generic framework for Mahalanobis distance learning, that can be seen as a generalization of a number of metric learning methods. The number of parameters involved in these problems is $O(\min(n^2, d^2))$, where n is the number of training points and d is the dimensionality of the data. This quadratic dependency effects not only the running time for both training and testing, but also poses tremendous challenges in estimating a quadratic number of parameters. For example, a data set with 10,000 dimensions leads to a Mahalanobis matrix with 100 million values. This represents a fundamental limitation of existing approaches, as many modern data mining problems possess relatively high dimensionality.

In this section, we present a generic framework for learning structured Mahalanobis distance (kernel) functions that scale linearly with the dimensionality (or training set size). Instead of representing the Mahalanobis distance/kernel matrix as a full $d \times d$ (or $n \times n$) matrix with $O(\min(n^2, d^2))$ parameters, our methods use compressed representations, admitting matrices parameterized by $O(\min(n, d))$ values. This enables the Mahalanobis distance/kernel function to be learned, stored, and evaluated efficiently in the context of high-dimensional large training sets.

Now, we formulate our high-dimensional metric learning framework. Consider a low-dimensional subspace in $\mathbb{R}^d$ and let the columns of U form an orthogonal basis of this subspace. We will constrain the learned Mahalanobis

distance matrix to be of the form:

$$W = \alpha I^d + W_l = W_0 + ULU^T, \tag{3.6}$$

where α is a parameter, I^d is the d-dimensional identity matrix, W_l denotes the low-rank part of W and $L \in \mathbb{S}_+^{k \times k}$ with $k \ll \min(n, d)$. Similar to problem (3.2), we propose the following problem to learn a fixed matrix (W_0 plus low-rank Mahalanobis metric):

$$\begin{aligned} \min_{W,L} \quad & \mathrm{Tr}(f(W)) \\ \text{s.t.} \quad & g_i(X^T W X) \leq b_i,\ 1 \leq i \leq m, \\ & W = \alpha I^d + ULU^T, \\ & W \succeq 0. \end{aligned} \tag{3.7}$$

Note that the above problem is identical to (3.2) except for an added constraint $W = \alpha I^d + ULU^T$. Assuming sub-gradient of f and g_i can be computed efficiently, the above problem can also be solved using standard convex programming methods. However, sub-gradients of f and g_i can potentially cost $\Omega(d^2)$ computational steps which is expensive for large d. Instead, we would like to solve for W in time at most *linear* in d.

Below, we show that for a large class of functions f, the above problem can be solved in time linear in d, and polynomial in k, m.

Theorem 3. *Let $f : \mathbb{R} \to \mathbb{R}$ be a function defined over the reals. Consider the extension of f to the spectrum of $W \in S_d^+$, i.e. $f(W) = Uf(\Lambda)U^T$, where $W = U\Lambda U^T$ is the eigenvalue decomposition of W (Definition 1.2, [49]). For*

this class of functions f, (3.7) reduces to:

$$\begin{aligned} \min_{L} \quad & Tr(f(\alpha I^k + L)) \\ s.t. \quad & g_i(X^T X + X^T U L U^T X) \leq b_i,\ 1 \leq i \leq m, \\ & L \succeq -\alpha I^k. \end{aligned} \tag{3.8}$$

Proof. Let $U' \in \mathbb{R}^{d \times d}$ be an orthonormal basis for $\mathbb{R}^d$ obtained by completing the basis represented by U, i.e., $U' = [U\ U_\perp]$ for a $U_\perp \in \mathbb{R}^{d \times (d-k)}$ s.t. $U^T U_\perp = 0$ and $U_\perp^T U_\perp = I^{d-k}$. Now,

$$\begin{aligned} W = \alpha I^d + U L U^T &= \alpha I^d + U' \begin{bmatrix} L & 0 \\ 0 & 0 \end{bmatrix} U' \\ &= U' \left(\alpha I^d + \begin{bmatrix} L & 0 \\ 0 & 0 \end{bmatrix} \right) U'^T. \end{aligned} \tag{3.9}$$

Now, it is easy to see that for a spectral function (Definition 1.2, [49]) f,

$$f(UWU^T) = U f(W) U^T, \tag{3.10}$$

where U is an orthogonal matrix. Also, for any $A, B \in \mathbb{R}^{d \times d}$,

$$f \begin{pmatrix} A & 0 \\ 0 & B \end{pmatrix} = \begin{pmatrix} f(A) & 0 \\ 0 & f(B) \end{pmatrix}, \tag{3.11}$$

Using (3.9), (3.10), and (3.11), we get:

$$\begin{aligned} \mathrm{Tr}(f(W)) &= \mathrm{Tr} \left(U' f \left(\alpha U'^T I^d U' + \begin{bmatrix} L & 0 \\ 0 & 0 \end{bmatrix} \right) U'^T \right), \\ &= \mathrm{Tr}(f \left(\begin{bmatrix} \alpha I^k + L & 0 \\ 0 & \alpha I^{d-k} \end{bmatrix} \right)), \\ &= \mathrm{Tr}(f(\alpha I^k + L)) + (d-k) f(\alpha). \end{aligned} \tag{3.12}$$

Theorem now follows using (3.12) and substituting for $W = \alpha I^k + U L U^T$ in the constraints of (3.7). □

We now present conditions on f and g_i so that (3.7) can be solved in time $poly(n, m, k)$ and linear in d.

Theorem 4. *Let $f : \mathbb{R} \to \mathbb{R}$ be a function defined over the reals such that:*

- *$f(x)$ is a convex function.*
- *A sub-gradient of $f(x)$ can be computed efficiently in $O(1)$ time.*

Similar to Theorem 3, consider the matrix generalization of f, i.e., $f(W) = Uf(\Lambda)U^T$, where $W = U\Lambda U^T$ is the eigenvalue decomposition of W.

Let $g_i(B), B \in \mathbb{R}^{n\times n}$ be a convex function and its sub-gradient can be computed in time $poly(n)$. Then, problem (3.7) *can be solved in time* **linear** *in d and polynomial in n, m, k.*

Proof. Using Theorem 3, problem (3.7) reduces to (3.8). Now consider the objective function in (3.8). Let $L = V_L\Lambda_L V_L^T$ be the eigenvalue decomposition of L, then $\nabla_L f(\alpha I^k + L) = V_L f'(\Lambda_L)V_L^T$. As a sub-gradient of $f(x)$ can be computed in $O(1)$ for all $x \in \mathbb{R}$, a sub-gradient of $f(\alpha I^k + L)$ can be computed using $O(k^3)$ operations.

Now, consider a constraint in (3.8), $g_i(X^TX + X^TULU^TX) \leq b_i$. Note that $X^TX + X^TULU^TX$ can be computed in time $\mathrm{O}((n^2 + nk)d + k^3)$ and sub-gradient of $g_i(\cdot)$ can be computed in time $poly(n)$. Hence, sub-gradient of $g_i(X^TX + X^TULU^TX)$ can be computed in time linear in d and poly in n, k. Finally, the positive definitess constraint is over a $k \times k$ matrix, and is independent of d.

So, (3.8) is a convex program with m constraints and where all the subgradients can be computed in time linear in d. Hence, it can be solved in time poly(n, m, k) and linear in d using standard convex programming methods like ellipsoid method [44]. □

3.3 Kernel Function Learning

In this section we study the problem of kernel function learning. We utilize the high-dimensional metric learning framework presented in the previous section to learn a kernel function. Specifically, we learn a Mahalanobis metric W using our metric learning framework (3.7) and the corresponding kernel function is given by: $k(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{x}^T W \boldsymbol{y}$. For the low-dimensional case, W can be solved for explicitly using standard convex programming softwares and can be used to form the kernel function $k(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{x}^T W \boldsymbol{y}$.

The more interesting scenario is the high-dimensional case where the feature vectors and the Mahalanobis metric W cannot be represented explicitly, and hence kernel learning is critical. The goal is to efficiently learn a metric W such that the kernel function $k(\boldsymbol{x}, \boldsymbol{y}) = \phi(\boldsymbol{x})^T W \phi(\boldsymbol{y})$ can be computed implicitly in the feature space[2]. In this section we show that by selecting appropriate basis U, we can use high-dimensional metric learning framework (3.7) introduced in previous section to solve for W implicitly in the feature space, i.e., the problem (3.7) is *kernelizable*. We assume that the kernel func-

[2]We denote high-dimensional inputs by $\phi(\boldsymbol{x})$ to mark their distinction from the small-dimensional inputs $\boldsymbol{x}$.

tion $K_0(\boldsymbol{x}, \boldsymbol{y}) = \phi(\boldsymbol{x})^T \phi(\boldsymbol{y})$ between any two data points can be computed in $O(1)$ time. Denote W^* as an optimal solution for (3.7). Now, we formally define *kernelizable* metric learning problems.

Definition 3.3.1. *An instance of metric learning problem* (3.7) *is kernelizable if the following conditions hold:*

- *Problem* (3.7) *is solvable efficiently in time poly(n, m) without explicit use of feature space vectors* $X = [\phi(\boldsymbol{x}_1), \phi(\boldsymbol{x}_2), \ldots, \phi(\boldsymbol{x}_n)]$.
- $Tr(W^* Y C Y^T)$*, where* $\phi_Y \in \mathbb{R}^{d \times N}$ *is the feature space representation of any given data points* $Y = [\phi(\boldsymbol{y}_1), \phi(\boldsymbol{y}_2), \ldots, \phi(\boldsymbol{y}_n)]$*, can be computed in time poly(N) for all* $C \in \mathbb{R}^{N \times N}$.

Theorem 5. *Let* $K_0 = X^T X$ *and let* R *be a* k*-dimensional basis such that* $R = XJ$*, where* $J \in \mathbb{R}^{n \times k}$*. Assuming* f *and* g_i*'s satisfy the conditions specified in Theorem 4, problem* (3.7) *with* $U = R(R^T R)^{-1/2} = XJ(J^T K_0 J)^{-1/2}$ *is kernelizable (see Definition 3.3.1)*[3].

Proof. Using Theorem 3, problem (3.7) reduces to (3.8). Now, consider a constraint $g_i(X^T X + X^T U L U^T X) \leq b_i$ specified in (3.8). As $U = XJ(J^T K_0 J)^{-1/2}$, $g_i(X^T X + X^T U L U^T X) = g_i(K_0 + K_0 J(J^T K_0 J)^{-1/2} L (J^T K_0 J)^{-1/2} J^T K_0)$. Hence, input to g_i can be computed in time $\mathrm{O}(n^2 + nk + k^3)$. Using similar arguments to Theorem 4, (3.7) can be solved efficiently in time $\mathrm{poly}(n, m)$. Furthermore,

[3]If $J^T K_0 J$ is rank deficient then pseudo inverse of $J^T K_0 J$ can be used.

learned metric is of the form $W^* = \alpha I^k + XJ(J^T K_0 J)^{-1/2} L^* J(J^T K_0 J)^{-1/2} X^T$. Hence, $\mathrm{Tr}(W^* YCY^T) = \mathrm{Tr}(\alpha Y^T Y + Y^T XJ(J^T K_0 J)^{-1/2} L^* J(J^T K_0 J)^{-1/2} X^T Y)$, i.e., $\mathrm{Tr}(W^* YCY^T)$ can be computed efficiently in terms of inner product in the feature space, $K_0(X, Y) = X^T Y$. □

3.3.1 Special Cases

In the previous section, we proved a general result on kernelization of metric learning. In this section, we further consider a few special cases of interest: the von Neumann divergence, the squared Frobenius norm and semi-definite programming. For each of the cases, we derive the required optimization problem to be solved and mention the relevant optimization algorithms that can be used.

3.3.1.1 von Neumann Divergence

The von Neumann divergence is a generalization of the well known KL-divergence to matrices. It is used extensively in quantum computing to compare density matrices of two different systems [81]. It is also used in the exponentiated matrix gradient method by [102], online-PCA method by [107] and fast SVD solver by [5]. The von Neumann divergence between W and W_0 is defined to be:

$$D_{\mathrm{vN}}(W, W_0) = \mathrm{Tr}(W \log W - W \log W_0 - W + W_0),$$

where both W and W_0 are positive definite. Now, consider an instance of the metric learning problem with linear constraints and von Neumann divergence

as the regularization function:

$$\begin{aligned} \min_{W} \quad & D_{\text{vN}}(W, I) \\ \text{s.t.} \quad & \text{Tr}(WXC_iX^T) \leq \boldsymbol{b}_i, \quad \forall 1 \leq i \leq m, \\ & W \succeq 0. \end{aligned} \tag{3.13}$$

It is easy to see that $D_{\text{vN}}(W, I) = \text{Tr}(f_{\text{vN}}(W))$, where

$$f_{\text{vN}}(W) = W \log W - W + I = U f_{\text{vN}}(\Lambda) U^T,$$

where $W = U\Lambda U^T$ is the eigenvalue decomposition of W and $f_{\text{vN}} : \mathbb{R} \to \mathbb{R}, f_{\text{vN}}(x) = x \log x - x + 1$. Also, note that $f_{\text{vN}}(x)$ is a strictly convex function with $\text{argmin}_x f_{\text{vN}}(x) = 1$ and $f_{\text{vN}}(1) = 0$. Hence, using Theorem 5, problem (3.13) is kernelizable since $D_{\text{vN}}(W, I)$ satisfies the required conditions. Using (3.8) with $U = X(X^TX)^{-1/2}$, the optimization problem to be solved is given by:

$$\begin{aligned} \min_{L} \quad & D_{\text{vN}}\left(I^n + L, I^n\right) \\ \text{s.t.} \quad & \text{Tr}(C_iK_0 + C_iK_0^{1/2}LK_0^{1/2}) \leq b_i, \quad \forall 1 \leq i \leq m \\ & L \succeq -I^n, \end{aligned} \tag{3.14}$$

Next, we derive a simplified version of the above optimization problem. Note that $D_{\text{vN}}(\cdot, \cdot)$ is defined only for positive semi-definite matrices. Hence, the constraint $L \succeq -I^n$ should be satisfied if the above problem is feasible. Thus, the reduced optimization problem is given by:

$$\begin{aligned} \min_{L} \quad & D_{\text{vN}}\left(I^n + L, I^n\right) \\ \text{s.t.} \quad & \text{Tr}(C_iK_0 + C_iK_0^{1/2}LK_0^{1/2}) \leq b_i, \quad \forall 1 \leq i \leq m. \end{aligned} \tag{3.15}$$

Now, we prove a lemma for general Bregman matrix divergences, of which the von Neumann divergence is a special case. Consider the following general optimization problem:

$$\begin{aligned} \min_{W} \quad & D_\phi(W, W_0) \\ \text{s.t.} \quad & \text{Tr}(WR_i) \leq s_i, \quad \forall 1 \leq i \leq m, \\ & W \succeq 0, \end{aligned} \tag{3.16}$$

where $D_\phi(W, W_0)$ is a Bregman matrix divergence [65] generated by a real-valued strictly convex function over symmetric matrices $\phi : \mathbb{R}^{n \times n} \to \mathbb{R}$, i.e.,

$$D_\phi(W, W_0) = \phi(W) - \phi(W_0) - \text{Tr}((W - W_0)^T \nabla\phi(W_0)). \tag{3.17}$$

Note that the von-Neumann divergence is a Bregman matrix divergence (see Equation (3.17)) with the generating function $\phi(X) = \text{Tr}(X \log X - X)$.

Lemma 1. *The solution to the dual of the primal formulation* (3.16) *is given by:*

$$\begin{aligned} \max_{W,\lambda,Z} \quad & \phi(W) - \phi(W_0) - \mathit{Tr}(W\nabla\phi(W)) + \mathit{Tr}(W_0\nabla\phi(W_0)) - s(\lambda) \\ \mathit{s.t.} \quad & \nabla\phi(W) = \nabla\phi(W_0) - R(\lambda) + Z, \qquad (3.18) \\ & \lambda \geq 0, \quad Z \succeq 0, \qquad (3.19) \end{aligned}$$

where $s(\lambda) = \sum_{i=1}^m \lambda_i s_i$ *and* $R(\lambda) = \sum_{i=1}^m \lambda_i R_i$.

Proof. First, consider the Lagrangian of (3.16):

$$\begin{aligned} L(W, \lambda, Z) &= D_\phi(W, W_0) + \text{Tr}(WR(\lambda)) - s(\lambda) - \text{Tr}(WZ), \\ \text{where } R(\lambda) = \sum_{i=1}^m \lambda_i R_i, \quad & s(\lambda) = \sum_{i=1}^m \lambda_i s_i, \quad Z \succeq 0, \lambda \geq 0. \end{aligned} \tag{3.20}$$

Now, note that

$$\nabla_W D_\phi(W, W_0) = \nabla\phi(W) - \nabla\phi(W_0). \tag{3.21}$$

Setting the gradient of the Lagrangian with respect to W to be zero and using (3.21), we get:

$$\nabla\phi(W) - \nabla\phi(W_0) + R(\lambda) - Z = 0, \tag{3.22}$$

$$\text{and so, } \mathrm{Tr}(W\nabla\phi(W_0)) = \mathrm{Tr}(W\nabla\phi(W)) + \mathrm{Tr}(WR(\lambda)) - \mathrm{Tr}(WZ). \tag{3.23}$$

Now, substituting (3.23) into the Lagrangian, we get:

$$L(W, \lambda, Z) = \phi(W) - \phi(W_0) - \mathrm{Tr}(W\nabla\phi(W)) + \mathrm{Tr}(W_0\nabla\phi(W_0)) - s(\lambda),$$

where $\nabla\phi(W) = \nabla\phi(W_0) - R(\lambda) + Z$. The lemma now follows directly. □

Using Lemma 1 and simplifying using the fact that $\frac{\partial \mathrm{Tr}(X \log X)}{\partial X} = \log X$, we get the following dual for problem (3.13):

$$\begin{aligned} \max_{\lambda} \quad & -\mathrm{Tr}(\exp(-K_0^{1/2} C(\lambda) K_0^{1/2})) - b(\lambda) \\ \text{s.t.} \quad & \lambda \geq 0, \end{aligned} \tag{3.24}$$

where $C(\lambda) = \sum_i \lambda_i C_i$ and $\boldsymbol{b}(\lambda) = \sum_i \lambda_i(\boldsymbol{b}_i - \mathrm{Tr}(C_i K_0))$.

Now, using $\mathrm{Tr}(AB) = \mathrm{Tr}(BA)$ we see that:

$$\mathrm{Tr}\left(\left(-K_0^{1/2} C(\lambda) K_0^{1/2}\right)^k\right) = \mathrm{Tr}\left(\left(-C(\lambda)K_0\right)^k\right).$$

Next, using the Taylor series expansion for the matrix exponential:

$$\begin{aligned}\mathrm{Tr}(\exp(-K_0^{1/2}C(\lambda)K_0^{1/2})) &= \mathrm{Tr}\left(\sum_{i=0}^{\infty}\frac{(-K_0^{1/2}C(\lambda)K_0^{1/2})^i}{i!}\right)\\ &= \sum_{i=0}^{\infty}\frac{\mathrm{Tr}\left((-K_0^{1/2}C(\lambda)K_0^{1/2})^i\right)}{i!}\\ &= \sum_{i=0}^{\infty}\frac{\mathrm{Tr}\left((-C(\lambda)K_0)^i\right)}{i!} = \mathrm{Tr}(\exp(-C(\lambda)K_0)).\end{aligned}$$

Hence, the resulting dual problem is given by:

$$\begin{aligned}\min_{\lambda} \quad & F(\lambda) = \mathrm{Tr}(\exp(-C(\lambda)K_0)) + \boldsymbol{b}(\lambda)\\ \text{s.t.} \quad & \lambda \geq 0.\end{aligned} \tag{3.25}$$

Also, $\frac{\partial F}{\partial \lambda_i} = \mathrm{Tr}(\exp(-C(\lambda)K_0)C_iK_0) + \boldsymbol{b}_i$. Hence, any first order smooth optimization method can be used to solve the above dual problem. Also, similar to [65], a Bregman's cyclic projection method can be used to solve the primal problem (3.15).

3.3.1.2 Squared Frobenius Divergence

The squared Frobenius norm divergence is defined as:

$$D_{\mathrm{frob}}(W, W_0) = \frac{1}{2}\|W - W_0\|_F^2,$$

and is a popular measure of distance between matrices. Consider the following instance of (3.7) with the squared Frobenius divergence as the objective

function and linear inequality constraints:

$$\begin{aligned} \min_{W} \quad & D_{\text{frob}}(W, \eta I) \\ \text{s.t.} \quad & \text{Tr}(WXC_iX^T) \leq \boldsymbol{b}_i, \quad \forall 1 \leq i \leq m, \\ & W \succeq 0. \end{aligned} \tag{3.26}$$

Note that for $\eta = 0$ and $C_i = (\boldsymbol{e}_a - \boldsymbol{e}_b)(\boldsymbol{e}_a - \boldsymbol{e}_b)^T - (\boldsymbol{e}_a - \boldsymbol{e}_c)(\boldsymbol{e}_a - \boldsymbol{e}_c)^T$ (relative distance constraints), the above problem (3.26) is the same as the one proposed by [93]. Below we see that, similar to [93], Theorem 5 in Section 3.3 guarantees kernelization for a more general class of Frobenius divergence based objective functions.

It is easy to see that $D_{\text{frob}}(W, \eta I) = \text{Tr}(f_{\text{frob}}(W))$, where

$$f_{\text{frob}}(W) = (W - \eta I)^T(W - \eta I) = U f_{\text{frob}}(\Lambda) U^T,$$

$W = U\Lambda U^T$ is the eigenvalue decomposition of W and $f_{\text{frob}} : \mathbb{R} \to \mathbb{R}$, $f_{\text{frob}}(x) = (x-\eta)^2$. Note that $f_{\text{frob}}(x)$ is a strictly convex function with $\text{argmin}_x f_{\text{frob}}(x) = \eta$ and $f_{\text{frob}}(\eta) = 0$. Hence, using Theorem 5, problem (3.26) is kernelizable since $D_{\text{frob}}(W, \eta I)$ satisfies the required conditions. Using (3.8), the optimization problem to be solved is given by:

$$\begin{aligned} \min_{L} \quad & \|L\|_F^2 \\ \text{s.t.} \quad & \text{Tr}(\eta C_i K_0 + C_i K_0^{1/2} L K_0^{1/2}) \leq b_i, \quad \forall 1 \leq i \leq m, \\ & L \succeq -\eta I^n. \end{aligned} \tag{3.27}$$

The above problem can be solved using standard convex optimization techniques like interior point methods.

3.3.1.3 SDPs

In this section we consider the case when the regularization function in (3.7) is a linear function. Examples of similar formulations for metric learning include MMC [110], LMNN [108]. We consider the following generic semidefinite program (SDP) to learn a linear transformation W:

$$\begin{aligned} \min_{W} \quad & \mathrm{Tr}(XC_0X^TW) \\ \text{s.t.} \quad & \mathrm{Tr}(WXC_iX^T) \leq \boldsymbol{b}_i, \quad \forall 1 \leq i \leq m \\ & W \succeq 0. \end{aligned} \tag{3.28}$$

Here we show that this problem can be efficiently solved for high dimensional data in its kernel space.

Theorem 6. *Problem* (3.28) *is kernelizable.*

Proof. (3.28) has a constant regularization function, i.e., it is a non-strict convex problem that may have multiple solutions. A variety of regularizations can be considered that lead to slightly different solutions. Here, we consider two regularizations:

- **Frobenius norm**: We add a squared Frobenius norm regularization to (3.28) so as to find the minimum Frobenius norm solution to (3.28) (when γ is sufficiently small):

$$\begin{aligned} \min_{W} \quad & \mathrm{Tr}(XC_0X^TW) + \frac{\gamma}{2}\|W\|_F^2 \\ \text{s.t.} \quad & \mathrm{Tr}(WXC_iX^T) \leq \boldsymbol{b}_i, \quad \forall 1 \leq i \leq m, \\ & W \succeq 0. \end{aligned} \tag{3.29}$$

Consider the following variational formulation of the problem:

$$\begin{aligned} \min_{t} \min_{W} \quad & t + \gamma \|W\|_F^2 \\ \text{s.t.} \quad & \mathrm{Tr}(WXC_iX^T) \leq \boldsymbol{b}_i, \quad \forall 1 \leq i \leq m \\ & \mathrm{Tr}(XC_0X^TW) \leq t \\ & W \succeq 0. \end{aligned} \tag{3.30}$$

Note that for constant t, the inner minimization problem in the above problem is similar to (3.26) and hence can be kernelized. Corresponding optimization problem is given by:

$$\begin{aligned} \min_{L,t} \quad & t + \gamma \|L\|_F^2 \\ \text{s.t.} \quad & \mathrm{Tr}(C_iK_0^{1/2}LK_0^{1/2}) \leq b_i, \quad \forall 1 \leq i \leq m \\ & \mathrm{Tr}(C_0K_0^{1/2}LK_0^{1/2}) \leq t \\ & L \succeq 0, \end{aligned} \tag{3.31}$$

Similar to (3.27), the above problem can be solved using convex optimization methods.

- **Log determinant**: In this case we seek the solution to (3.28) with minimum determinant. To this effect, we add a log-determinant regularization:

$$\begin{aligned} \min_{W} \quad & \mathrm{Tr}(XC_0X^TW) - \gamma \log\det W \\ \text{s.t.} \quad & \mathrm{Tr}(WXC_iX^T) \leq \boldsymbol{b}_i, \quad \forall 1 \leq i \leq m, \\ & W \succeq 0. \end{aligned} \tag{3.32}$$

The above regularization was also considered by [65], which provided a fast projection algorithm for the case when each C_i is a one-rank matrix and discussed conditions for which the optimal solution to the regularized problem is an optimal solution to the original SDP.

Consider the following variational formulation of (3.32):

$$\begin{aligned} \min_{t} \min_{W} \quad & t - \gamma \log\det W \\ \text{s.t.} \quad & \operatorname{Tr}(WXC_iX^T) \leq \boldsymbol{b}_i, \quad \forall 1 \leq i \leq m, \\ & \operatorname{Tr}(XC_0X^TW) \leq t, \\ & W \succeq 0. \end{aligned} \tag{3.33}$$

Note that the objective function of the inner optimization problem of (3.33) satisfies the conditions of Theorem 5, and hence (3.33) or equivalently (3.32) is kernelizable.

□

In summary, in this section, we showed that for a large class of regularization functions f, general metric learning problem (3.2) can be kernelized, i.e., can be used to learn a kernel function satisfying the provided side information. Additionally, we studied a few existing metric learning formulations that are instantiations of (3.2) and derive the corresponding optimization problem for kernel learning.

3.4 Summary

In this chapter, we proposed a generic regularized framework for metric learning (3.2). Our framework can handle a variety of distance constraints and regularization functions. In fact, almost all the existing convex metric learning methods can be seen as specific instances of our framework. For high-dimensional datasets, we extended our framework to reduce the number of learned parameters from quadratic to linear in the dimensionality d. Under certain mild conditions (which almost all the existing methods satisfy), our high-dimensional metric learning framework can be used to learn the metric efficiently in time linear in d.

We also considered the problem of kernel function learning, and show that under certain mild conditions, our metric learning framework can be used for kernel learning as well. Similar to the metric learning case, we showed that the learned kernel function can be restricted to a small number of parameters, and hence, can be scaled for large datasets. This is particularly useful for applications where sub-linear time approximate nearest neighbor search is important. Finally, we considered a few examples of existing metric learning methods and studied their corresponding kernel learning problem.

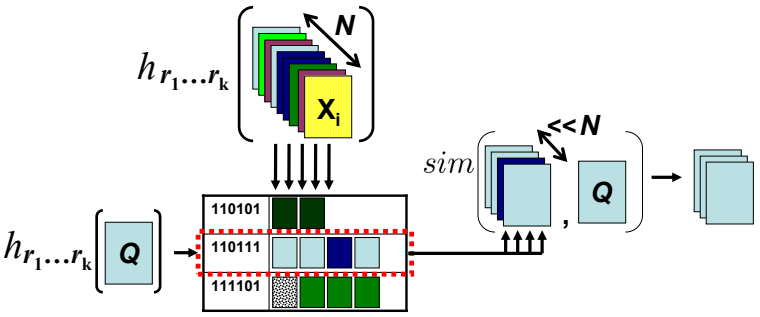
Chapter 4

Fast Similarity Search for Learned Metrics

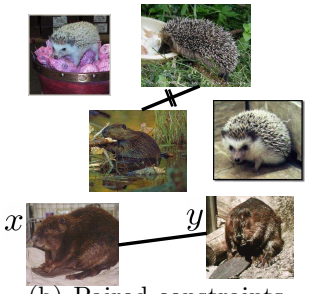
In this chapter, we study a method for fast approximate similarity search with learned Mahalanobis metrics. We formulate randomized hash functions that incorporate side-information from partially labeled data or paired constraints, so that the input examples may be efficiently indexed according to the learned metric without resorting to a naive exhaustive scan of all items. We present a straightforward solution for the case of relatively low-dimensional input vector spaces, and further derive a solution to accommodate very high-dimensional data for which explicit input space computations are infeasible. The former contribution makes fast indexing accessible for numerous existing metric learning methods (e.g., [110, 10, 24]), while the latter is of particular interest for commonly used image representations, such as bags-of-words, multi-dimensional multi-resolution histograms, and other high-dimensional features. Some of the material presented in this chapter was published in [57].

4.1 Hashing for Semi-Supervised Similarity Search

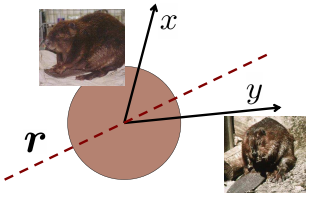
The main idea of our approach is to learn a parameterization of a Mahalanobis metric based on the provided side-information, e.g., partial labels or



(a) Main idea of locality-sensitive hashing

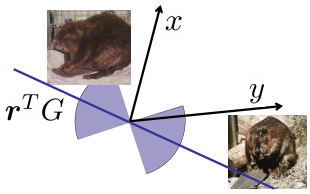


(b) Paired constraints



$h_{\boldsymbol{r}}(x) \neq h_{\boldsymbol{r}}(y)$

(c) Original hash function



$h_{\boldsymbol{r},A}(x) = h_{\boldsymbol{r},A}(y)$

(d) Semi-supervised hash function

Figure 4.1: (a) When learning a metric, some paired constraints can be obtained for a portion of the image database, specifying some examples that ought to be treated as similar (straight line) or dissimilar (crossed out line). (b) Whereas existing randomized LSH functions hash examples similar under the original distance together, (c) our semi-supervised hash functions incorporate the learned constraints, so that examples constrained to be similar—or other pairs like them—will with high probability hash together. The circular red region in (b) denotes that the existing LSH functions generate a hyperplane uniformly at random to separate images, in contrast, as indicated by the blue "hourglass" region in (c), our hash functions bias the selection of random hyperplane to reflect the specified (dis)similarity constraints.

paired constraints for some training examples, while simultaneously encoding the learned information into randomized hash functions. These functions will guarantee that the more similar inputs are under the learned metric, the more likely they are to collide in a hash table. After constructing hash tables containing all of the initial training (database) examples, examples similar to a

new instance are found in sub-linear time in the size of the database by evaluating the learned metric between the new example and any examples with which it shares a hash bucket.

We learn Mahalanobis metric parameterized by A using the metric learning framework introduced in Section 3.1, Chapter 3. Recall that, A is obtained by solving a convex optimization problem of the form:

$$\begin{aligned} &\min_{W} \operatorname{Tr}(f(W)) \\ &\text{s.t. } g_i(X^T W X) \leq b_i,\ 1 \leq i \leq m, \\ &\quad W \succeq 0. \end{aligned} \tag{4.1}$$

Also, assuming f satisfies conditions provided in the Theorem 5, the learned metric is of the form $W = I + XSX^T$, where $S \in \mathbb{R}^{n \times n}$.

Now, we introduce a family of hash functions that accommodate learned Mahalanobis distances, where we want to retrieve examples $\boldsymbol{x}_i$ for an input $\boldsymbol{x}_q$ for which the value $d_W(\boldsymbol{x}_i, \boldsymbol{x}_q)$ resulting from (2.1) is small, or, in terms of the kernel form, for which the value of $s_A(\boldsymbol{x}_i, \boldsymbol{x}_q) = \boldsymbol{x}_q^T W \boldsymbol{x}_i$ is high.

4.1.1 Explicit Formulation

For the explicit case, we can adapt the randomized hyperplane hashing approach of [21], which further follows from results in [37]. In the process of designing a randomized algorithm for the MAX-CUT problem, Goemans and Williamson demonstrated that, given a collection of vectors $\boldsymbol{x}_1, ..., \boldsymbol{x}_n$ on the unit sphere, and a randomly generated vector $\boldsymbol{r}$, the following relationship

holds:

$$\Pr[\text{sign}(\boldsymbol{x}_i^T \boldsymbol{r}) \neq \text{sign}(\boldsymbol{x}_j^T \boldsymbol{r})] = \frac{1}{\pi} \cos^{-1}(\boldsymbol{x}_i^T \boldsymbol{x}_j).$$

In [21], Charikar uses this result to design hash functions for LSH for the inner product similarity function. In particular, we let

$$h_{\boldsymbol{r}}(\boldsymbol{x}) = \begin{cases} 1, & \text{if } \boldsymbol{r}^T \boldsymbol{x} \geq 0 \\ 0, & \text{otherwise} \end{cases}, \tag{4.2}$$

which is a valid LSH function as given in (2.5).

This may be naturally extended to the setting when we learn a Mahalanobis distance. For the explicit case, we assume that metric is learned using our general metric learning framework (4.1) where function f is any convex function for which a sub-gradient can be computed efficiently. Given the matrix learned metric W, such that $W = G^T G$, we generate the following randomized hash functions $h_{\boldsymbol{r},W}$, which accept an input point and return a hash key bit:

$$h_{\boldsymbol{r},W}(\boldsymbol{x}) = \begin{cases} 1, & \text{if } \boldsymbol{r}^T G \boldsymbol{x} \geq 0 \\ 0, & \text{otherwise} \end{cases}, \tag{4.3}$$

where the vector $\boldsymbol{r}$ is chosen at random from a d-dimensional Gaussian distribution with zero mean and unit variance. This construction leverages earlier results showing that (i) the probability of two unit vectors having a dot product with random vector $\boldsymbol{r}$ that are opposite in sign is proportional to the angle between them [37], and (ii) the sign of $\boldsymbol{r}^T \boldsymbol{x}_i$ is therefore a locality-sensitive function for the inner product of any two inputs $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ [21].

Thus by parameterizing the hash functions instead by G (which is computable since W is p.d.), we obtain the following relationship:

$$\Pr\left[h_{\boldsymbol{r},W}(\boldsymbol{x}_i) = h_{\boldsymbol{r},W}(\boldsymbol{x}_j)\right] = 1 - \frac{1}{\pi}\cos^{-1}\left(\frac{\boldsymbol{x}_i^T W \boldsymbol{x}_j}{\sqrt{|G\boldsymbol{x}_i||G\boldsymbol{x}_j|}}\right),$$

which sustains the LSH requirement of (2.5) for a learned Mahalanobis metric, whether W is computed using the method of [24] or otherwise [110, 108, 10]. Essentially we have shifted the random hyperplane $\boldsymbol{r}$ according to W, and by factoring it by G we allow the random hash function itself to "carry" the information about the learned metric. The denominator in the cosine term normalizes the learned kernel values.

In this case, we could transform all the data according to W *prior* to hashing, i.e, $\boldsymbol{x} \rightarrow W^{1/2}x$ and use standard Euclidean nearest neighbor hashing techniques; however such a technique would require transformation of test points also which would require additional $O(d^2)$ computation. Furthermore, the choice of presentation here helps set up the more complex formulation we derive below. Note that (4.3) requires that the input dimension d be low enough that W can be handled in memory directly, allowing explicit optimization of (4.1).

4.1.2 Implicit Formulation

We are particularly interested in the case where the dimensionality d may be very high—say on the order of 10^4 to 10^6—but the examples are sparse and therefore representable (e.g., bags of words or histogram pyramids [95, 42]). Even though the examples are each sparse, *the matrix A can*

be dense, with values for each dimension. In this case, the kernelized metric learning techniques are necessary. However, this complicates the computation of hash functions, as they can no longer be computed directly as in (4.3) above. Thus, in this section we derive a new algorithm to make simultaneous implicit updates to both the hash functions and the metric.

Here, we assume that the metric is learned using our high-dimensional metric learning framework (see Section 3.2). Recall that, by setting $\alpha = 1$ and $U = \Phi K_0^{-1/2}$, our high-dimensional metric learning framework learns a metric of the form $W = I + \Phi K_0^{-1/2} L K_0^{-1/2} \Phi^T$, where $\Phi = [\phi(\boldsymbol{x}_1), \ldots, \phi(\boldsymbol{x}_c)]$ be the $d \times c$ matrix of the initial c data points selected from the training points that forms the basis $R = \Phi$ to which W is restricted to. We denote high-dimensional inputs by $\phi(\boldsymbol{x})$ to mark their distinction from the dense inputs $\boldsymbol{x}$ handled earlier. Let $\phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}_j)$ be the initial (non-learned) kernel value between example $\boldsymbol{x}_i$ and the input $\boldsymbol{x}_j$. Initially, $K_0 = \Phi^T \Phi$, and so, implicitly, $W_0 = I$. As in the explicit formulation above, the goal is to wrap G into the hash function, i.e. compute $\boldsymbol{r}^T G \phi(\boldsymbol{x})$, but now we must do so without working directly with G.

In the following, we will show that an appropriate hash function $h_{\boldsymbol{r},W}$ for inputs $\phi(\boldsymbol{x})$ can be defined as:

$$h_{\boldsymbol{r},W}(\phi(\boldsymbol{x})) = \begin{cases} 1, & \text{if } \boldsymbol{r}^T \phi(\boldsymbol{x}) + \sum_{i=1}^{c} \gamma_i^r \phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}) \geq 0 \\ 0, & \text{otherwise} \end{cases}, \tag{4.4}$$

where $\phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x})$ is the original kernel value between $\boldsymbol{x}_i$ and the query $\boldsymbol{x}$, and γ_i^r are coefficients computed once (offline) during metric learning (and will be

defined below). Note that while G is dense and therefore not manageable, computing $\boldsymbol{r}^T\phi(\boldsymbol{x})$ is computationally inexpensive, as only the entries of $\boldsymbol{r}$ corresponding to non-zero entries in $\phi(\boldsymbol{x})$ need to be generated. Should the inputs be high-dimensional but dense, our implicit form is still valuable, as we bypass computing $O(d^2)$ products with G and require only $O(d)$ inner products for $\boldsymbol{r}^T\phi(\boldsymbol{x})$.

Next we show that for $W = G^TG$ learned using our high-dimensional metric learning framework, G can be computed efficiently in terms of the initially chosen c basis points, and hence, (4.4) can be computed efficiently. Our construction relies on the following lemma.

Lemma 2. *Let $W = I + \Phi K_0^{-1/2} L K_0^{-1/2} \Phi^T$, where $W \succeq 0$ and $L \in \mathbb{R}^{c\times c}$. Also, assume Φ has full column rank. Then, $W^{1/2} = I + \Phi S \Phi^T$, with*

$$S = K_0^{-1/2}\left(-I + (I+L)^{1/2}\right)K_0^{-1/2}, \tag{4.5}$$

where $K_0 = \Phi^T\Phi$.

Proof. Note that S given above is well defined as, $W \succeq 0 \implies I + L \succeq 0$. Lemma now follows by substituting for S in $W = W^{1/2}W^{1/2}$. □

Recall that using (3.7) with the basis $R = \Phi$, Mahalanobis metric W can be computed efficiently in terms of $K_0 = \Phi^T\Phi$ and is of the form $W = I + \Phi K_0^{-1/2} L K_0^{-1/2} \Phi^T$. Using Lemma 2, $G = I + \Phi S \Phi^T$, where $W = G^TG$,

$S = K_0^{-1/2}\left(I + K_0^{1/2} L K_0^{1/2}\right) K_0^{-1/2}$. Therefore, we have

$$\begin{aligned} \boldsymbol{r}^T G\phi(\boldsymbol{x}) &= \boldsymbol{r}^T\phi(x) + \sum_{i=1}^{c}\sum_{j=1}^{c} S_{ij}\boldsymbol{r}^T\phi(\boldsymbol{x}_j)\phi(\boldsymbol{x}_i)^T\phi(\boldsymbol{x}) \\ &= \boldsymbol{r}^T\phi(\boldsymbol{x}) + \sum_{i=1}^{c}\gamma_i^r\phi(\boldsymbol{x}_i)^T\phi(\boldsymbol{x}), \end{aligned}$$

where $\gamma_i^r = \sum_j S_{ij}\boldsymbol{r}^T\phi(\boldsymbol{x}_j)$, and is a notation substitution for the first equality. This notation reflects that the values of each γ_i^r rely only on known constrained points, and thus can be efficiently computed in the training phase, prior to hashing anything into the database. Finally, having determined the expression for $\boldsymbol{r}^T G\phi(\boldsymbol{x})$, we arrive at our hash function definition in (4.4). Note the analogy between the use of $\boldsymbol{r}^T G\boldsymbol{x}$ and $\boldsymbol{r}^T G\phi(\boldsymbol{x})$ in (4.3) and (4.4), respectively.

In summary, in this section, we presented a method for efficient computation of learned hash function for high-dimensional data points in terms of inner product between just a few of the training points. Our hash function is parameterized by a Mahalanobis metric W obtained using the high-dimensional metric learning framework introduced in Section 3.2.

Note that, to compute the hash function defined in (4.4) we need to compute S given in (4.5). This in turn requires computation of K_0^{-1} which can be prohibitive for large databases. In the next section, we show that for the special case of the Information Theoretic Metric Learning (ITML) method, S can be obtained during the metric learning phase itself and do not need to explicitly invert K_0.

4.1.2.1 ITML based Hashing

In this section, we consider Information Theoretic Metric Learning (ITML) approach, a special instance of the general metric learning problem (3.7), in the context of locality sensitive hashing. Specifically, we show that the parameters γ_i^r required for efficient computation of hash functions can be obtained via simple updates while learning the metric itself.

Recall that, W^1 is obtained by repeatedly projecting the current solution onto a single constraint, via the update:

$$W_{t+1} = W_t + \beta_t W_t(\boldsymbol{x}_{i_t} - \boldsymbol{x}_{j_t})(\boldsymbol{x}_{i_t} - \boldsymbol{x}_{j_t})^T W_t, \tag{4.6}$$

where $\boldsymbol{x}_{i_t}$ and $\boldsymbol{x}_{j_t}$ are the constrained data points for iteration t, and β_t is a projection parameter computed by the algorithm. See [24] for further details. For the case of high dimensional data, W is implicitly updated using:

$$K_{t+1} = K_t + \beta_t K_t(\boldsymbol{e}_{i_t} - \boldsymbol{e}_{j_t})(\boldsymbol{e}_{i_t} - \boldsymbol{e}_{j_t})^T K_t, \tag{4.7}$$

where the vectors $\boldsymbol{e}_{i_t}$ and $\boldsymbol{e}_{j_t}$ refer to the i_t-th and j_t-th standard basis vectors, respectively, and the projection parameter β_t is the same as in (4.6)(see [24]).

Our construction relies on two technical lemmas, which are given below.

Lemma 3. *Let $B = I + \beta \boldsymbol{y}\boldsymbol{y}^T$ be positive semi-definite. Then $B^{1/2} = I + \alpha \boldsymbol{y}\boldsymbol{y}^T$, with $\alpha = (\pm\sqrt{1 + \boldsymbol{y}^T\boldsymbol{y}\beta} - 1)/\boldsymbol{y}^T\boldsymbol{y}$.*

Proof. The lemma follows directly using Lemma 2 where $\Phi = \boldsymbol{y}$. □

[1] A variable without a subscript t denotes its value after convergence.

Lemma 4. *For all t, if $G_0 = I$ and $S_0 = 0$, then*

$$G_{t+1} = I + \Phi S_{t+1} \Phi^T$$
$$S_{t+1} = S_t + \alpha_t (I + S_t K_0)(\boldsymbol{e}_{i_t} - \boldsymbol{e}_{j_t})(\boldsymbol{e}_{i_t} - \boldsymbol{e}_{j_t})^T (I + K_0 S_t^T)(I + K_0 S_t).$$

Proof. We prove this lemma using induction. In the base case, $S_0 = 0$, implying $G_0 = I$ and $G_0^T G_0 = W_0 = I$. Now, let the hypothesis holds for step t, i.e. $G_t = I + \Phi S_t \Phi^T$. Note that this form for G_t is analogous to the form for $W = I + XMX^T$ (however, the matrices S and M are not equivalent). Now update for matrix G at step $t+1$ is given by:

$$G_{t+1} = (I + \beta_t G_t \boldsymbol{v}_t \boldsymbol{v}_t^T G_t^T)^{1/2} G_t = (I + \alpha_t G_t \boldsymbol{v}_t \boldsymbol{v}_t^T G_t^T) G_t, \tag{4.8}$$

where, $\boldsymbol{v}_t = \phi(\boldsymbol{y}_t) - \phi(\boldsymbol{z}_t)$ and α is given by Lemma (3). Now substituting for G_t we get,

$$G_{t+1} = I + \Phi S_t \Phi^T + \alpha_t G_t \boldsymbol{v}_t \boldsymbol{v}_t^T G_t^T G_t \tag{4.9}$$

Now, $\boldsymbol{v}_t = \Phi(\boldsymbol{e}_{i_t} - \boldsymbol{e}_{j_t})$. Thus,

$$\begin{aligned} G_t \boldsymbol{v}_t = (I + \Phi S_t \Phi^T)\Phi(\boldsymbol{e}_{i_t} - \boldsymbol{e}_{j_t}) &= (\Phi + \Phi S_t \Phi^T \Phi)(\boldsymbol{e}_{i_t} - \boldsymbol{e}_{j_t}) \\ &= \Phi(I + S_t K_0)(\boldsymbol{e}_{i_t} - \boldsymbol{e}_{j_t}), \end{aligned} \tag{4.10}$$

where last equality follows from $\Phi^T \Phi = K_0$. Similarly,

$$\Phi G_t = \Phi^T (I + \Phi S_t \Phi) = (\Phi + \Phi^T \Phi S_t \Phi) = (I + K_0 S_t)\Phi. \tag{4.11}$$

Using Equation (4.9), (4.10) and (4.11),

$$\begin{aligned} G_{t+1} = I + \Phi S_t \Phi^T + \alpha_t \Phi (I + S_t K_0)(\boldsymbol{e}_{i_t} - \boldsymbol{e}_{j_t}) \\ (\boldsymbol{e}_{i_t} - \boldsymbol{e}_{j_t})^T (I + K_0 S_t^T)(I + K_0 S_t)\Phi. \end{aligned} \tag{4.12}$$

Thus substituting,

$$S_{t+1} = S_t + \alpha_t(I + S_t K_0)(\boldsymbol{e}_{i_t} - \boldsymbol{e}_{j_t})(\boldsymbol{e}_{i_t} - \boldsymbol{e}_{j_t})^T(I + K_0 S_t^T)(I + K_0 S_t),$$

proves the lemma. □

Recall the update rule for W from (4.6): $W_{t+1} = W_t + \beta_t W_t \boldsymbol{v}_t \boldsymbol{v}_t^T W_t$, where $\boldsymbol{v}_t = \phi(\boldsymbol{y}_t) - \phi(\boldsymbol{z}_t)$, if points y_t and z_t are involved in the constraint under consideration at iteration t. We emphasize that just as this update must be implemented implicitly via (4.7), so too we must derive an *implicit* update for the G_t matrix required by our hash functions.

Since W_t is p.d., we can factorize it as $W_t = G_t^T G_t$, which allows us to rewrite the update as:

$$W_{t+1} = G_t^T(I + \beta_t G_t \boldsymbol{v}_t \boldsymbol{v}_t^T G_t^T) G_t.$$

As a result, if we factorize $I + \beta_t G_t \boldsymbol{v}_t \boldsymbol{v}_t^T G_t^T$, we can derive an update for G_{t+1}:

$$G_{t+1} = (I + \beta_t G_t \boldsymbol{v}_t \boldsymbol{v}_t^T G_t^T)^{1/2} G_t = (I + \alpha_t G_t \boldsymbol{v}_t \boldsymbol{v}_t^T G_t^T) G_t, \tag{4.13}$$

where the second equality follows from Lemma 3 using $\boldsymbol{y} = G_t \boldsymbol{v}_t$, and α_t is defined accordingly.

Using (4.13) and Lemma 4, G_t can be expressed as $G_t = I + \Phi S_t \Phi^T$, where S_t is a $c \times c$ matrix of coefficients that determines the contribution of each of the c points to G. Initially, S_0 is set to be zero matrix, and from there every S_{t+1} is iteratively updated in $O(c^2)$ time via $S_{t+1} =$

$$S_t + \alpha_t(I + S_t K_0)(\boldsymbol{e}_{i_t} - \boldsymbol{e}_{j_t})(\boldsymbol{e}_{i_t} - \boldsymbol{e}_{j_t})^T(I + K_0 S_t^T)(I + K_0 S_t).$$

Using this result, at convergence of the metric learning algorithm we can compute $\boldsymbol{r}^T G\phi(\boldsymbol{x})$ in terms of the c inner products $\phi(\boldsymbol{x}_i)^T\phi(\boldsymbol{x})$ as follows:

$$\begin{aligned}\boldsymbol{r}^T G\phi(\boldsymbol{x}) &= \boldsymbol{r}^T\phi(x) + \sum_{i=1}^{c}\sum_{j=1}^{c} S_{ij}\boldsymbol{r}^T\phi(\boldsymbol{x}_j)\phi(\boldsymbol{x}_i)^T\phi(\boldsymbol{x}) \\ &= \boldsymbol{r}^T\phi(\boldsymbol{x}) + \sum_{i=1}^{c}\gamma_i^r\phi(\boldsymbol{x}_i)^T\phi(\boldsymbol{x}),\end{aligned}$$

where $\gamma_i^r = \sum_j S_{ij}\boldsymbol{r}^T\phi(\boldsymbol{x}_j)$. Note that, inversion of K_0 is not required and S can be computed efficiently while solving for W.

In this section we presented the main technical contribution of this chapter: explicit and implicit methods to construct semi-supervised hash functions. For the implicit case, we presented a generic method to compute hash functions for our kernel learning framework and considered the special case of kernels learned using ITML.

4.2 Searching Hashed Examples

Having constructed LSH functions for learned metrics, we can apply existing methods [53, 21] to perform sub-linear time approximate similarity search. Given N data points in a Hamming space and an input $\boldsymbol{x}_q$, approximate near-neighbor (ANN) techniques guarantee retrieval of example(s) within the radius $(1+\epsilon)D$ from $\boldsymbol{x}_q$ in $O(N^{1/(1+\epsilon)})$ time, where the true nearest neighbor is at a distance of D from $\boldsymbol{x}_q$. We employ the method of [21], which requires searching $M = 2N^{1/(1+\epsilon)}$ examples to obtain the first ANN. (Note that $M << N$ for large databases.) After hashing, we only need to compute

the learned kernel values between the query and the examples with which it collided. The hashed neighbors are ranked according to these scores, and this ranked list is used for k-NN classification, clustering, etc., depending on the application.

To generate b-bit hash keys, we select b random vectors $[\boldsymbol{r}_1, \ldots, \boldsymbol{r}_b]$ to form b hash functions and concatenate the resulting bits from (4.3) or (4.4). There is a tradeoff in the selection of b: larger values will increase the accuracy of how well the keys themselves reflect the learned metric, but will increase computation time and can lead to too few collisions in the hash tables. On the other hand, lower values of b make hashing faster, but the key will only coarsely reflect our metric, and too many collisions may result.

Table 4.1 summarizes the computational complexity for the main steps of our algorithm: projections during offline metric learning, computing each hash bit for a given point, and computing the ANNs for a hashed query. z is the number of non-zero entries in the query, $z \leq d$. Having defined theoretically sound locality-sensitive hash functions for learned metrics, we can apply existing methods [53, 21] to perform sub-linear time approximate similarity search. Given N data points in a Hamming space and an input $\boldsymbol{x}_q$, approximate near-neighbor techniques guarantee retrieval of example(s) within the radius $(1+\epsilon)D$ from $\boldsymbol{x}_q$ in $O(N^{1/(1+\epsilon)})$ time, where the true nearest neighbor is at a distance of D from $\boldsymbol{x}_q$.

To generate a b-bit hash key for every example, we select b random vectors $[\boldsymbol{r}_1, \ldots, \boldsymbol{r}_b]$ to form b hash functions. The hash key for an input $\boldsymbol{x}$

Step	Explicit	Implicit
Metric learning projection (offline)	$O(d^2)$	$O(c^2)$
Hashing: compute $h_{\boldsymbol{r},A}(\boldsymbol{x})$	$O(d)$	$O(z)$
Search: identify the query's ANNs	$O(Md)$	$O(Mz)$

Table 4.1: Computational complexity for the proposed method, using variables defined in the text.

is then the concatenation of the outputs of (4.3) (or similarly, the outputs of (4.4) for an input $\phi(\boldsymbol{x})$). The tradeoff in the selection of b is as follows: larger values will increase the accuracy of how well the keys themselves reflect the metric of interest, but will also increase computation time and can lead to too few collisions in the hash tables. On the other hand, if b is lower, hashing will be faster, but the key will only coarsely reflect our metric, and too many collisions may result. A query hashes to certain buckets in the hash table, where it collides with some small portion of the stored examples. We employ the search method of [21], which requires searching $M = 2N^{1/(1+\epsilon)}$ examples for the $k = 1$ approximate-NN. Then we compute the learned kernel (or metric) values only between the query and those examples. The hashed neighbors are ranked according to these scores, and this ranked list is used for k-nn classification, clustering, etc., depending on the application.

4.3 Results

In the following we first apply our algorithm in the low-dimensional setting to a nearest neighbor classification problem for software support. Then we apply our algorithm in the implicit setting for image search in three distinct

domains: exemplar-based recognition, pose estimation, and feature indexing. In all cases, our experimental goal is twofold: 1) to evaluate the impact on accuracy a learned metric has relative to both standard baseline metrics and state-of-the-art methods, and 2) to test how reliably our semi-supervised hash functions preserve the learned metrics in practice when performing sub-linear time database searches. We therefore report results in terms of both accuracy improvements as well as speedups realized.

For all the experiments, we use ITML method for learning metric and in the implicit hashing case, we compute the hash function using the updates derived in Section 4.1.2.1. Throughout we select examples for (dis)similarity constraints randomly from among a pool of examples. For categorical data, (dis)similarity constraints are associated with points having different (same) labels; for data with parameter vectors, constraints are determined based on examples' nearness in the parameter space. We compute the distance between all pairs of a subset (≈ 100) of the database examples according to the non-learned metric, and then let the distance constraints' lower ℓ and upper u limits be the 1-st and 99-th percentile of those values, respectively. We measure accuracy in terms of the error of the retrieved nearest neighbors' labels, which is either a parameter vector (in the case of the pose data) or a class label (in the case of the patches and object images).

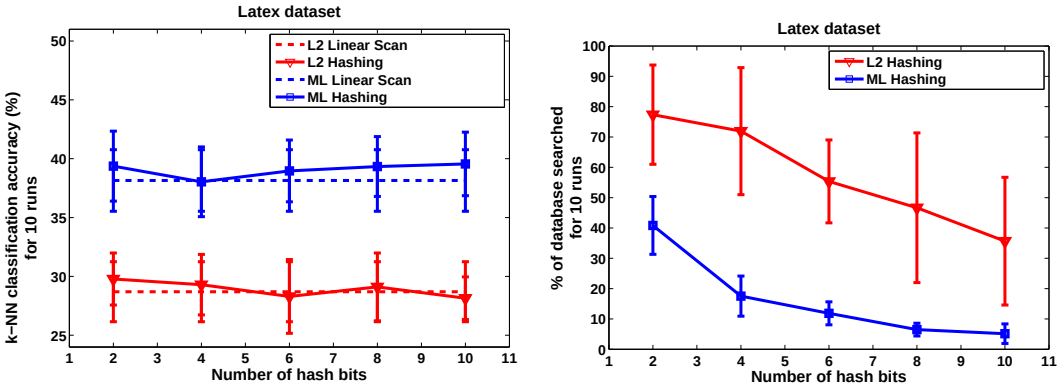


Figure 4.2: Comparison of the accuracy (left) and time requirements (right) when hashing with the original and learned metrics for the Latex dataset. Left plot shows k-nn classification accuracy; right plot shows the search time in terms of the percentage of database items searched per query, both as a function of the number of hash bits b. Results are from ten runs with random query / database partitions, with $\epsilon = 1.5$. Our learned hash functions reduce the search to about 5% of the database, with virtually no loss in accuracy over the exhaustive linear scan.

4.3.1 Clarify

We first evaluate our method for learned metric hashing on a nearest neighbor (nn) classification problem using data from the CLARIFY system of Ha et al. [46]. CLARIFY assists a programmer in diagnosing errors by identifying previously seen abnormal termination reports with similar program features, and pointing the programmer to other users who have had similar problems. We experiment with a database of n=3825 such examples collected from the Latex typesetting program. The features are d = 20-dimensional, and so our explicit formulation for learning hash functions is most appropriate. Paired similarity constraints are generated with information-theoretic metric

learning using 20 labeled examples from each class. For 10 random partitions of the data, we extract 30 examples for each of its nine classes, and treat the remainder as database examples. We measure the $k = 4$-nearest-neighbor classification accuracy and search times over all 270 queries per run, under four settings: the original Euclidean distance metric and a linear scan, the original distance with LSH, the learned metric with a linear scan, and the learned metric with LSH. For both hashing cases we fixed $\epsilon = 1.5$.

Figure 4.2 shows the resulting accuracy and complexity gains. By incorporating the paired constraints, the learned metric shows clear accuracy gains over the unconstrained Euclidean distance, yielding about 10% higher correct classification rates. The k-nn rates for the both associated hashed results are on average as good as the linear scan results, and in this case, have little dependence on the number of hash functions used. As b increases, however, the hash keys become more specific and allow larger amounts of the database to be ignored for any given query (right-hand plot). When searching only 5% of the database, our learned hash functions suffer no loss in accuracy yet enable an average $13x$ speedup (maximum speedup $34x$) relative to an exhaustive scan with the learned metric (including the overhead cost of computing the hash keys). Interestingly, for the same values of ϵ and b, the number of examples searched with the learned hash functions is noticeably lower than that of the generic hash functions, and has a tighter distribution. While the indexing guarantees remain the same, we infer that just as the learned metric adjusts the feature space so that in-class examples are more closely clustered,

Method	d	k=1	k=7	k=50
L_2 linear scan	24K	8.9	12.0	15.1
L_2 hashing	24K	9.4	12.8	15.6
PSH, linear scan	1.5K	9.4	12.2	15.9
PCA, linear scan	60	13.5	14.0	16.8
ML PCA, lin. scan	60	13.1	13.8	16.2
ML linear scan	24K	8.4	11.5	14.1
ML hashing	24K	8.8	12.1	14.9

Table 4.2: Mean pose error (in cm) obtained with each method. Our approach (denoted ML) outperforms the L_2 baseline and PSH [92].

the learned hash functions better map them to distinct keys.

4.3.2 Human Body Pose Estimation.

Next we demonstrate our method applied to single-frame human body pose estimation. Example-based techniques to infer pose (e.g. [7, 92]) store a large database of image examples that are labeled with their true pose (i.e., 3d joint positions or angles). A query image is indexed into the database according to image similarity, and the query's pose is estimated based on the pose parameters attached to those nearest neighbors (NN). Thus our objective for this task is to learn a metric for the image features that reports small distances for examples that are close in pose space, and to make the search scalable by hashing according to the learned metric. This is similar to the goals of the parameter-sensitive hashing (PSH) method of [92]. However our approach is distinct from [92] in that it allows one to seamlessly both hash and search according to the learned metric. As a result it may provide more

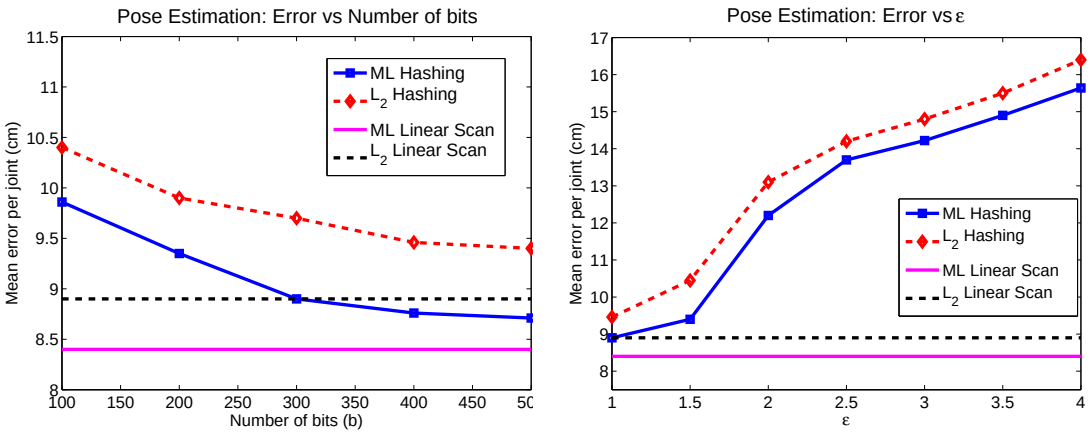


Figure 4.3: **Left**: Error as a function of the number of hash bits. Fast search with the learned metric is more accurate than the L_2 baseline. For both, the error converges around b=500 bits. **Right**: Hashing error relative to an exhaustive linear scan as a function of ϵ, which controls the search time required.

accurate retrievals, as we show empirically below.

We use a database of half a million examples provided by the authors of [99], where PSH is employed within a pose tracker. The images were generated with Poser graphics software: human figures in a variety of clothes are rendered in many realistic poses drawn from mocap data. Each image is represented by a $d = 24,016$-dimensional multi-scale edge detection histogram (EDH). The vectors' high dimension requires our implicit formulation for semi-supervised hash functions. We use a linear kernel over $c = 50$ randomly selected examples as the initial kernel (K_0). We hold out 1000 test queries examples, and generate $1,000,000$ similarity constraints among $50K$ of the remaining training examples. For each, we constrain the distance of the 10 nearest exemplars (in terms of pose parameters) to be less than ℓ. Simi-

Figure 4.4: Examples of pose estimates. Each column contains a different pose. Top row contains query images, remaining rows show the best pose retrieved by each method. Second row shows best pose obtained by our method

larly, of all the examples with a pose distance greater than a threshold t, 10 are randomly picked and their distance to the example is constrained to be greater than u. The values of t and c are selected with cross-validation.

As baselines, we compute results for NN search with both the Euclidean distance (L_2) on the EDH's, and the Hamming distance on the PSH embeddings provided by the authors of [99]. To hash with the L_2 baseline we simply

apply [21]. We also use PCA to reduce the dimensionality of the EDH vectors in order to apply our explicit formulation. We measure the error for a query by the mean distance of its true joint positions to the poses in the k-NN. To give a sense of the variety of the data, a random database example is on average at a distance of 34.5 cm from a query.

Table 4.2 shows the overall errors for each method. (Throughout our approach is denoted by 'ML'.) With a linear scan, ML yields the most accurate retrievals of all methods, and with hashing it outperforms all the hashing-based techniques. The PCA-based results are relatively poor, indicating the need to use the full high-d features and thus our implicit formulation. A paired-error T-test reveals that our improvements over PSH and L_2 are statistically significant, with 99.95% confidence.

Figure 4.4 shows the NN retrieved by each method for five typical queries. In most examples, L_2 and PSH estimate the overall pose reasonably well, but suffer on one or more limbs, whereas our approach more precisely matches all limbs and yields a lower total error. While PSH does not improve over the L_2 baseline for this dataset (as it did for data in [92]), it does do nearly as well as L_2 when using about $16x$ fewer dimensions; it appears its main advantage here is the ability to significantly reduce the dimension.

Our semi-supervised hash functions maintain the accuracy of the learned metric, but for orders of magnitude less search time than the linear scan. With our Matlab implementation, a linear scan requires 433.25 s per query, while our hashing technique requires just 1.39 s. On average, metric learning with

hashing searches just 0.5% of the database. Figure 4.3 compares the error obtained by ML+hashing and L_2+hashing when varying the number of hash bits (middle plot) and the search time allowed (right plot). For a large number of bits, the hash keys are more precise and hence the error drops (although hashing overhead increases). Similarly, since $M = 2N^{1/(1+\epsilon)}$, for higher values of ϵ we must search fewer examples, but accuracy guarantees decrease.

4.3.3 Exemplar-based Object and Scene Categorization

Next we evaluate our method applied for exemplar-based object and scene recognition with the Caltech-101, a common benchmark, and a dataset of scene images downloaded from Flickr. The goal is to predict the object or scene class of a test example by finding the most visually similar examples in the labeled database, and then allowing those neighbors to cast votes on the label. In this set of experiments we demonstrate the flexibility of our approach relative to the choice of a base metric, with results using three different kernels defined in the vision literature [42, 73, 1].

4.3.3.1 Caltech-101 database

To compare the Caltech-101 images we consider learning kernels on top of Grauman and Darrell's pyramid match kernel (PMK) [42] applied to SIFT features, and the kernel designed by Zhang and colleagues [1] applied to geometric blur features. As described above, the PMK uses multi-resolution

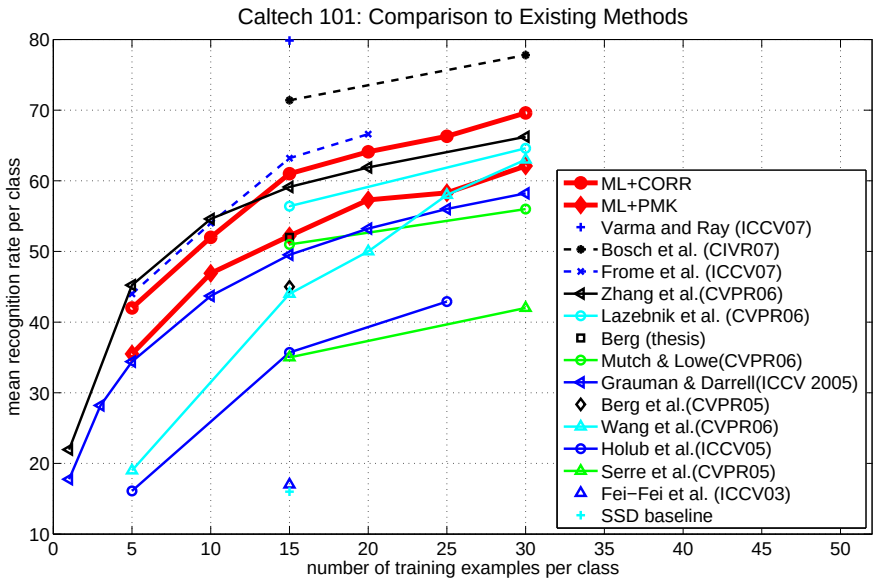


Figure 4.5: Comparison against existing techniques. Our method outperforms all other single metric/kernel approaches.

histograms to estimate the correspondence between two sets of local image features. To hash with the non-learned PMK, the pyramids are embedded as described in [67]. The pyramid inputs are sparse but extremely high-dimensional ($d = O(10^6)$), thus explicitly representing A is infeasible, and the implicit form of our technique is necessary. The kernel in [1] also measures the correspondences between local features, but by averaging over the minimum distance to matching features in terms of the descriptors and their position in the image; we will refer to it as CORR. Note that we can learn kernels for both the PMK and CORR using the kernel learning formulation from [24], but can only hash with the learned PMK, since an explicit vector space representation ($\phi(\boldsymbol{x})$) for the CORR kernel is unknown.

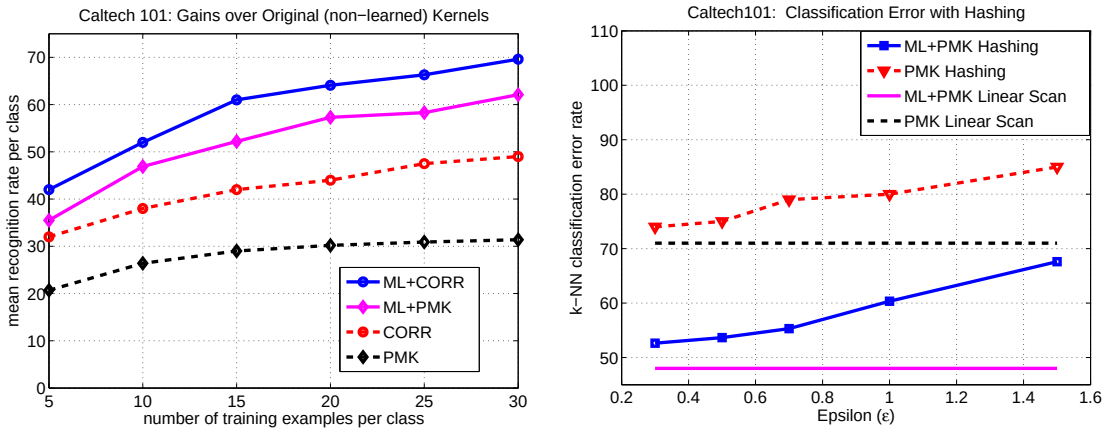


Figure 4.6: **Left**: Our learned kernels significantly improve NN search accuracy relative to their non-learned counterparts, the CORR and PMK kernels. **Right**: Comparison of the k-NN classification error when hashing with the original and learned PMK. This plot shows the accuracy-search time tradeoff when using the original or learned hashing functions. CORR refers to the kernel proposed by Zhang et al. [1]. (Best viewed in color.)

We first evaluate the effectiveness of metric learning itself on this dataset. We pose a k-NN classification task, and evaluate both the original (PMK or CORR) and learned kernels when used in a linear scan mode. We set $k = 1$ for our experiments; this value was chosen arbitrarily. We vary the number of training examples T per class for the database, using the remainder as test examples, and measure accuracy in terms of the mean recognition rate per class, as is standard practice for this dataset.

Figure 4.5 shows our results relative to all other existing techniques that have been applied to this dataset. Our approach outperforms all existing single-kernel classifier methods when using the learned CORR kernel: we achieve 61.0% accuracy for $T = 15$ and 69.6% accuracy for $T = 30$. Our

learned PMK achieves 52.2% accuracy for $T = 15$ and 62.1% accuracy for $T = 30$. Figure 4.6 shows specifically the comparison of the original baseline kernels for NN classification. The plot on the left reveals gains in NN retrieval accuracy; notably, our learned kernels with simple NN classification also outperform the baseline kernels when used with SVMs [1, 42]. Only the results of recent multiple-metric approaches [29, 104, 17] (shown with dashed lines in Figure 4.5) are more accurate, though they also incur the greater cost of applying each of the base kernels in sequence to all examples, while our method requires only one comparison to be computed per example.

Now we consider hashing over the learned PMK. For $T = 15$, our learned hash functions achieve 47% accuracy, and require about $10x$ less computation time than a linear scan when accounting for the hash key computation (here $N = 1515$, which is modest compared to the pose data evaluated above). The righthand plot in Figure 4.6 shows the error of our learned PMK-based hashing compared to the baseline [43] as a function of ϵ. For this data the value of b had little effect on accuracy. As with the linear scan search, we still realize significant accuracy improvements, but now with a guaranteed sub-linear time search.

4.3.3.2 Flickr scene database

To evaluate our learned hash functions when the base kernel is the proximity distribution kernel (PDK), we performed experiments with a dataset of 5400 images of 18 different tourist attractions from the photo-sharing site

Flickr. We took three cities in Europe that have major tourist attractions: Rome, London, and Paris. The tourist sites for each city were taken from the top attractions in www.TripAdvisor.com under the headings Religious site, Architectural building, Historic site, Opera, Museum, and Theater. Overall, the list yielded 18 classes: eight from Rome, five from London, and five from Paris. The classes are: Arc de Triomphe, Basilica San Pietro, Castel SantAngelo, Colosseum, Eiffel Tower, Globe Theatre, Hotel des Invalides, House of Parliament, Louvre, Notre Dame Cathedral, Pantheon, Piazza Campidoglio, Roman Forum, Santa Maria Maggiore, Spanish Steps, St. Pauls Cathedral, Tower Bridge, and Westminister Abbey. We downloaded the first 300 images returned from each search query to represent the data for each class. Since not all images downloaded for a given tag actually contain the proper scene, we manually added ground truth labels. About 90% of the initial tags on the downloaded images were accurate.

Duplicate images and images that had no response from the interest point detectors were removed and then replaced with lower ranked images so that the number of images per category remained at 300. All images were scaled down to have moderate width (320 pixels).

Note that the regular viewpoints and scales in the Caltech-101 images above make it possible to improve the unordered set representation using simply image coordinate positions, which means the PMK with features including spatial position are adequate. For the Flickr data, however, the viewpoint and scale varies significantly across instances of the same scene, so the loose con-

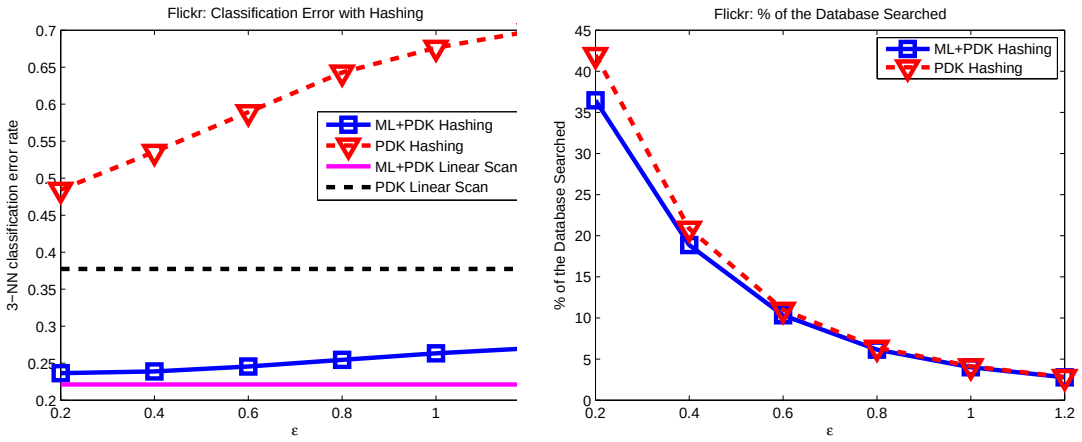


Figure 4.7: Scene classification on the Flickr dataset. **Left**: Hashing error relative to an exhaustive linear scan as a function of ϵ, which controls the search time required. **Right**: Amount of the database searched as a function of ϵ. Clearly our method ML+PDK Hashing achieves significantly higher accuracy while searching for a smaller size of the database.

figurations of features captured by the PDK provide a better way to preserve semi-local geometry without being overly restrictive. We detect corner and blob-like regions in the images using the Harris-affine [80] and Maximally Stable Extremal Regions (MSER) [77] detectors, and represent all regions with SIFT descriptors [75]. The PDK requires a codebook to quantize features; following [73], we set the number of visual words to be $k = 200$, and include neighbors up to rank $R = 64$. We use the embedding described in [67] for hashing with PDK.

We again pose a nearest-neighbor classification task, and compare results when using either a linear scan or hashing, with the original base PDK or the learned PDK. We randomly select 275 images per class for training and

the remaining images for testing. Figure 4.7 shows the results. The linear scan error for either hashing method show the best possible performance, and again as we decrease the ϵ parameter, we can expect more accurate results at the cost of longer query times. The learned PDK achieves a significantly higher accuracy then the original base PDK while searching a smaller amount of the Flickr database.

4.3.4 Indexing Local Patch Descriptors.

Finally, we evaluate our approach on a patch matching task using data provided from the Photo Tourism project [96] and [51]. The dataset contains about 300K local patches extracted from interest points in multiple users' photos of scenes from different viewpoints. The objective is to be able to rapidly identify any matching patches from the same 3d scene point in order to provide correspondences to a structure from motion algorithm. For this application, *classifying* patches is not so useful; rather, one wants to find all relevant patches. Thus we measure accuracy in terms of precision and recall.

We add random jitter (scaling, rotations, and shifts) to all patches as prescribed in [51], extract both the raw patch intensities and SIFT descriptors, and then pose the retrieval task to the L_2 baseline and our learned metrics for each representation. To learn metrics we gather constraints from 10,000 matching and non-matching patch pairs, with a 50-50 mix taken from the Trevi and Halfdome portions of the data. All methods are tested on 100K pairs from the Notre Dame portion. The left plot in Figure 4.8 compares their accuracy

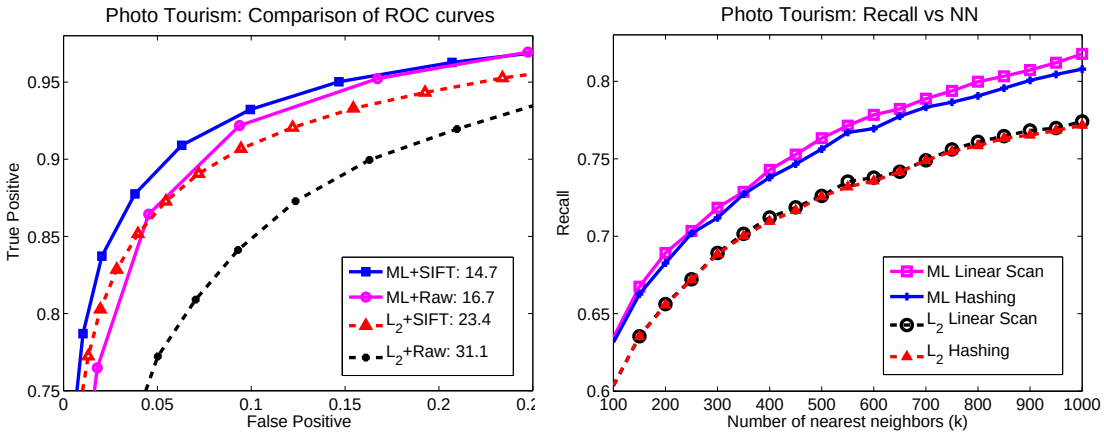


Figure 4.8: **Left**: This plot illustrates accuracy improvements of the learned metric (ML) relative to L_2 baselines, for both raw patches (dimensionality $d = 4096$) and SIFT descriptors (dimensionality $d = 128$). **Right**: This plot shows the recall as a function of the number of SIFT patches retrieved, for our method and the L_2 baseline. Our semi-supervised hash functions maintain accuracy close to a linear scan, while requiring much less search time.

via ROC curves for each feature and metric combination; the numbers in the legend summarize the error in terms of the false positive rate once 95% of the true positives are retrieved. ML+raw intensities yields a significant gain over L_2+raw, while ML+SIFT also gives some improvement.[2]

Finally, we consider our ML-hashing algorithm for the SIFT patches. We measure accuracy by the relevance of the NN ranking: for increasing values of k, we compute the recall rate within the top k-NN. We calculate this score with and without hashing, and before and after metric learning. In order to

[2]In this experiment we were able to reproduce the baseline for L_2 given in [51], however we were unable to do so for their SIFT baseline, for which 6% error is obtained. We suspect this is due to our un-optimized SIFT extraction, and that ML would continue to yield similar improvements as above if provided better descriptors.

control k for the hashing, we consider as many nearby hash bins as necessary. In the right plot in Figure 4.8, we see that the learned metric outperforms the L_2 baseline, and that hashing does not noticeably degrade accuracy. When $k = 1000$, we search only 16.1% of the database when hashing over the learned metric, and when $k = 1$, we search only 0.8%, leading to substantial gains in retrieval time (about a factor of 80 vs. linear scan).

4.4 Summary

In this chapter, we introduced a method to enable efficient approximate similarity search for learned metrics. Our main contribution is a new algorithm to construct theoretically sound locality-sensitive hash functions for both implicit and explicit parameterizations of the Mahalanobis distance function. For high-dimensional data, we show that a hash function can be computed efficiently for metrics learned using our high-dimensional metric learning framework (3.7). Furthermore, for the special case of the ITML method [24], we derive simultaneous implicit updates for both the hash function and the learned metric. Experiments demonstrate our technique's accuracy and flexibility for a number of large-scale search tasks.

Chapter 5

Online Metric Learning

The metric learning framework we introduced in Chapter 3 assumes that all the side-information is available at once and is used in an offline training phase. However, in many real applications, the desired distance function may need to change gradually over time as additional information or constraints are received. For instance, in image search applications on the internet, online click-through data that is continually collected may impact the desired distance function. To address this need, recent work on *online* metric learning algorithms attempts to handle constraints that are received one at a time [93, 24]. Unfortunately, current methods suffer from a number of drawbacks, including speed, bound quality, and empirical performance.

Further complicating this scenario is the fact that fast retrieval methods must be in place on top of the learned metrics for many applications dealing with large-scale databases. For example, in image search applications, relevant images within very large collections must be quickly returned to the user, and constraints and user queries may often be intermingled across time. Thus a good online metric learner must also be able to support fast similarity search routines. This is problematic since existing methods (e.g., locality-sensitive

hashing [35, 21] or kd-trees) assume a static distance function, and are expensive to update when the underlying distance function changes.

In this chapter, we consider the problem of online metric learning. Our goal is to make metric learning practical for real-world learning tasks in which both constraints and queries must be handled efficiently in an online manner. To that end, we first develop an online metric learning algorithm that uses LogDet regularization and exact gradient descent. The new algorithm is inspired by the metric learning algorithm studied in [24]; however, while the loss bounds for the latter method are dependent on the input data, our loss bounds are independent of the sequence of constraints given to the algorithm. Furthermore, unlike the Pseudo-metric Online Learning Algorithm (POLA) [93], another recent online technique, our algorithm requires no eigenvector computation, making it considerably faster in practice. We further show how our algorithm can be integrated with large-scale approximate similarity search. We devise a method to incrementally update locality-sensitive hash keys during the updates of the metric learner, making it possible to perform accurate sub-linear time nearest neighbor searches over the data in an online manner.

We compare our algorithm to related existing methods using a variety of standard data sets. We show that our method outperforms existing approaches, and even performs comparably to several offline metric learning algorithms. To evaluate our approach for indexing a large-scale database, we include experiments with a set of 300,000 image patches; our online algorithm effectively learns to compare patches, and our hashing construction allows

accurate fast retrieval for online queries.

5.1 Online Metric Learning

In this section we introduce our model for online metric learning, develop an efficient algorithm to implement it, and prove regret bounds.

5.1.1 Formulation and Algorithm

As in previous chapters, we restrict ourselves to learning a Mahalanobis distance function over our input data, which is a distance function parameterized by a $d \times d$ positive definite matrix W. Recall that, the squared Mahalanobis distance between given d-dimensional vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ is defined as

$$d_W(\boldsymbol{u}, \boldsymbol{v}) = (\boldsymbol{u} - \boldsymbol{v})^T W (\boldsymbol{u} - \boldsymbol{v}).$$

In this chapter, we assume that the side-information is available in form of distance or similarity constraints that arise from supervised information—for example, the distance between two points in the same class should be "small".

In contrast to offline approaches, which assume all constraints are provided up front, online algorithms assume that constraints are received one at a time. That is, we assume that at time step t, there exists a current distance function parameterized by W_t. A constraint is received, encoded by the triple $(\boldsymbol{u}_t, \boldsymbol{v}_t, y_t)$, where y_t is the target distance between $\boldsymbol{u}_t$ and $\boldsymbol{v}_t$. Using W_t, we first *predict* the distance $\hat{y}_t = d_{W_t}(\boldsymbol{u}_t, \boldsymbol{v}_t)$ using our current distance function, and incur a loss $\ell(\hat{y}_t, y_t)$. Then we *update* our matrix from W_t to W_{t+1}. The goal

is to minimize the sum of the losses over all time steps, i.e. $L_W = \sum_t \ell(\hat{y}_t, y_t)$. One common choice is the squared loss: $\ell(\hat{y}_t, y_t) = \frac{1}{2}(\hat{y}_t - y_t)^2$. We also consider a variant of the model where the input is a quadruple $(\boldsymbol{u}_t, \boldsymbol{v}_t, y_t, b_t)$, where $b_t = 1$ if we require that the distance between $\boldsymbol{u}_t$ and $\boldsymbol{v}_t$ be less than or equal to y_t, and $b_t = -1$ if we require that the distance between $\boldsymbol{u}_t$ and $\boldsymbol{v}_t$ be greater than or equal to y_t. In that case, the corresponding loss function is $\ell(\hat{y}_t, y_t, b_t) = \max(0, \frac{1}{2}b_t(\hat{y}_t - y_t))^2$.

A typical approach [63, 24, 93] for the above given online learning problem is to solve for W_{t+1} by minimizing a regularized loss at each step:

$$W_{t+1} = \underset{W \succ 0}{\operatorname{argmin}} D(W, W_t) + \eta \ell(d_W(\boldsymbol{u}_t, \boldsymbol{v}_t), y_t), \tag{5.1}$$

where $D(W, W_t)$ is a regularization function and $\eta_t > 0$ is the regularization parameter. As in [24], we use the *LogDet* divergence $D_{\ell d}(W, W_t)$ as the regularization function. It is defined over positive definite matrices and is given by $D_{\ell d}(W, W_t) = \operatorname{tr}(WW_t^{-1}) - \log\det(WW_t^{-1}) - d$. This divergence has previously been shown to be useful in the context of metric learning [24]. It has a number of desirable properties for metric learning, including scale-invariance, automatic enforcement of positive definiteness, and a maximum-likelihood interpretation.

Existing approaches solve for W_{t+1} by approximating the gradient of the loss function, i.e. $\ell'(d_W(\boldsymbol{u}_t, \boldsymbol{v}_t), y_t)$ is approximated by $\ell'(d_{W_t}(\boldsymbol{u}_t, \boldsymbol{v}_t), y_t)$ [63, 24, 93]. While for some regularization functions (e.g. Frobenius divergence, von-Neumann divergence) such a scheme works out well, for LogDet

regularization it can lead to non-definite matrices for which the regularization function is not even defined. This results in a scheme that has to adapt the regularization parameter in order to maintain positive definiteness [24].

In contrast, our algorithm proceeds by *exactly* solving for the updated parameters W_{t+1} that minimize (5.1). Since we use the exact gradient, our analysis will become more involved; however, the resulting algorithm will have several advantages over existing methods for online metric learning. By setting gradient of (5.1) with LogDet regularization to be zero w.r.t. W:

$$W_{t+1}^{-1} = W_t^{-1} + \eta(\bar{y} - y_t)\boldsymbol{z}_t\boldsymbol{z}_t^T,$$

where $\boldsymbol{z}_t = \boldsymbol{u}_t - \boldsymbol{v}_t$ and $\bar{y} = d_{W_{t+1}}(\boldsymbol{u}_t, \boldsymbol{v}_t) = \boldsymbol{z}_t^T W_{t+1}\boldsymbol{z}_t$. Using straightforward algebra and the Sherman-Morrison inverse formula, we can show that the resulting solution to the minimization of (5.1) is:

$$W_{t+1} = W_t - \frac{\eta(\bar{y} - y_t)W_t\boldsymbol{z}_t\boldsymbol{z}_t^T W_t}{1 + \eta(\bar{y} - y_t)\boldsymbol{z}_t^T W_t\boldsymbol{z}_t}. \tag{5.2}$$

It is not immediately clear that this update can be applied, since $\bar{y}$ is a function of W_{t+1}. However, by multiplying the update in (5.2) on the left by $\boldsymbol{z}_t^T$ and on the right by $\boldsymbol{z}_t$ and noting that $\hat{y}_t = \boldsymbol{z}_t^T W_t\boldsymbol{z}_t$, we obtain the following:

$$\bar{y} = d_{W_{t+1}}(\boldsymbol{u}_t, \boldsymbol{v}_t) = \boldsymbol{z}_t^T W_{t+1}\boldsymbol{z}_t = \frac{\hat{y}_t}{1 + \eta(\bar{y} - y_t)\hat{y}_t}. \tag{5.3}$$

(5.3) is a quadratic equation in $\bar{y}$, and can be solved as

$$\bar{y} = \frac{\eta y_t\hat{y}_t - 1 + \sqrt{(\eta y_t\hat{y}_t - 1)^2 + 4\eta\hat{y}_t^2}}{2\eta\hat{y}_t}. \tag{5.4}$$

We justify ignoring the other solution to the quadratic equation later in the proof of Theorem 7. We can solve directly for $\bar{y}$ using the above given formula, and then plug this into the update (5.2). For the case when the input is a quadruple and the loss function is the squared hinge loss, we only perform the update (5.2) if the new constraint is violated.

It is possible to show that the resulting matrix W_{t+1} is positive definite; the proof appears below in Theorem 7.

Theorem 7. *Suppose W_t is positive-definite, then W_{t+1} given by the LEGO update* (5.2) *is positive definite.*

Proof. We prove the theorem using mathematical induction. The base case holds as the initial matrix $W_0 = \frac{1}{d}I$ is positive definite.

Now we show that W_{t+1} is positive-definite(p.d.) if W_t is p.s.d. If W_t is a p.d. matrix then $\hat{y} = \boldsymbol{z}^T W_t \boldsymbol{z} \geq 0$ for all $\boldsymbol{z}$. Now,

$$\sqrt{(\eta y_t \hat{y}_t - 1)^2 + 4\eta \hat{y}_t^2} \geq |\eta y_t \hat{y}_t - 1|,$$

as $\eta > 0$. Thus, using (5.4), $\bar{y} = \boldsymbol{z}_t^T W_{t+1} \boldsymbol{z}_t \geq 0$ for all $\boldsymbol{z}_t$. Hence, W_{t+1} is p.d. This also justifies our rejection of the other solution to the quadratic equation (5.3), since otherwise $\bar{y} \leq 0$, implying the resulting W_{t+1} would be indefinite. □

The fact that this update maintains positive definiteness is a key advantage of our method over existing methods; POLA, for example, requires

projection to the positive semidefinite cone via an eigendecomposition. The final loss bound in [24] depends on the regularization parameter η_t from each iteration and is in turn dependent on the sequence of constraints, an undesirable property for online algorithms. In contrast, by minimizing the function f_t we designate above in (5.1), our algorithm's updates automatically maintain positive definiteness. This means that the regularization parameter η need not be changed according to the current constraint, and the resulting bounds (Section 5.1.2) and empirical performance are notably stronger.

We refer to our algorithm as LogDet Exact Gradient Online (LEGO), and use this name throughout to distinguish it from POLA [93] (which uses a Frobenius regularization) and the Information Theoretic Metric Learning (ITML)-Online algorithm [24] (which uses an approximation to the gradient).

5.1.2 Analysis

We now analyze the regret bounds for our online metric learning algorithm.

To evaluate the online learner's quality, we want to compare the loss of the online algorithm (which has access to one constraint at a time in sequence) to the loss of the best possible offline algorithm (which has access to all constraints at once). Let $\hat{d}_t = d_{W^*}(\boldsymbol{u}_t, \boldsymbol{v}_t)$ be the learned distance between points $\boldsymbol{u}_t$ and $\boldsymbol{v}_t$ with a fixed positive definite matrix W^*, and let $L_{W^*} = \sum_t \ell(\hat{d}_t, y_t)$ be the loss suffered over all t time steps. Note that the loss L_{W^*} is with respect to a single matrix W^*, whereas L_W (Section 5.1.1) is with respect to a matrix

that is being updated every time step. Let W^* be the optimal offline solution, i.e. it minimizes total loss incurred (L_{W^*}). The goal is to demonstrate that the loss of the online algorithm L_W is competitive with the loss of any offline algorithm. To that end, we now show that $L_W \leq c_1 L_{W^*} + c_2$, where c_1 and c_2 are constants.

To compute the regret bound, we present a lemma (Lemma 7) to bound the loss at each step incurred by the algorithm in terms of the loss incurred by the optimal offline solution. In the result below, we assume that the length of the data points is bounded: $\|\boldsymbol{u}\|_2^2 \leq R$ for all $\boldsymbol{u}$, i.e. $\mathrm{Tr}(X_t) \leq R$ for all t. Also, let the optimal W^* to be $0 \prec W^* \prec I$. Thus $y_t \in [0,\ R]$, if a y_t is provided out of this range than we can just clip it to be between either 0 or R. We first present a few useful lemmas that will be used for proving Lemma 7.

Lemma 5. *At each step t of the LEGO algorithm,*

$$\bar{y} \leq \frac{R}{2} + \sqrt{\frac{R^2}{4} + \frac{1}{\eta}}.$$

Proof. Using (5.4),

$$\bar{y} = \frac{\eta y_t - 1/\hat{y}_t + \sqrt{(\eta y_t - 1/\hat{y}_t)^2 + 4\eta}}{2\eta}.$$

Now $\hat{y}_t \geq 0$, $\eta \geq 0$ and $y_t \leq R$. Thus, simplifying we get:

$$\bar{y} \leq \frac{R}{2} + \sqrt{\frac{R^2}{4} + \frac{1}{\eta}}.$$

Hence proved. □

Lemma 6. *At each step t of the LEGO algorithm,*

$$\frac{d\bar{y}}{dy_t} = \frac{\eta\bar{y}^2}{1+\eta\bar{y}^2}, \ \frac{d(\bar{y}-y_t)}{dy_t} = \frac{-1}{1+\eta\bar{y}^2}.$$

Proof. Using (5.3),

$$1+\eta(\bar{y}-y_t)\hat{y}_t = \frac{\hat{y}_t}{\bar{y}}.$$

Thus,

$$\eta\left(\frac{d\bar{y}}{dy_t}-1\right) = -\frac{1}{\bar{y}^2}\frac{d\bar{y}}{dy_t}.$$

Simplifying we get:

$$\frac{d\bar{y}}{dy_t} = \frac{\eta\bar{y}^2}{1+\eta\bar{y}^2}. \tag{5.5}$$

By subtracting 1 from both the sides of (5.5) and simplifying the resulting expression, we get:

$$\frac{d(\bar{y}-y_t)}{dy_t} = \frac{-1}{1+\eta\bar{y}^2}.$$

Hence proved. □

Lemma 7. *At each step t,*

$$\frac{1}{2}\alpha_t(\hat{y}_t-y_t)^2 - \frac{1}{2}\beta_t(d_{W^*}(\boldsymbol{u}_t,\boldsymbol{v}_t)-y_t)^2 \leq D_{ld}(W^*,W_t) - D_{ld}(W^*,W_{t+1}),$$

where $0 \leq \alpha_t \leq \frac{\eta}{1+\eta\left(\frac{R}{2}+\sqrt{\frac{R^2}{4}+\frac{1}{\eta}}\right)^2}$, $\beta_t = \eta$, *and W^* is the optimal offline solution.*

Proof.

$$D_{ld}(W^*, W_t) - D_{ld}(W^*, W_{t+1}) = \log\left(\frac{\det(W_t)}{\det(W_{t+1})}\right) + \mathrm{Tr}((W_t^{-1} - W_{t+1}^{-1})W^*).$$

Since $\log(\det(W)) = \mathrm{Tr}(\log(W))$ and $\mathrm{Tr}(AB) = \mathrm{Tr}(BA)$, we have that

$$\begin{aligned} D_{ld}(W^*, W_t) - D_{ld}(W^*, W_{t+1}) &= \mathrm{Tr}(\log(W_t) - \log(W_{t+1})) + \mathrm{Tr}((W_t^{-1} - W_{t+1}^{-1})W^*) \\ &= \mathrm{Tr}(\log(W_t W_{t+1}^{-1})) - \eta(\bar{y} - y_t)\mathrm{Tr}(X_t W^*) \\ &= \mathrm{Tr}(\log(I + \eta(\bar{y} - y_t)W_t X_t)) - \eta(a_t - y_t)\mathrm{Tr}(W^* X_t) \\ &= \log(1 + \eta_t(\bar{y} - y_t)\hat{y}_t) - \eta_t(\bar{y} - y_t)r, \end{aligned}$$

where $r = \mathrm{Tr}(W^* X_t)$.

Proving the lemma amounts to showing that:

$$\begin{aligned} D_{ld}(W^*, W_t) - D_{ld}(W^*, W_{t+1}) &= \log(1 + \eta_t(\bar{y} - y_t)\hat{y}_t) - \eta_t(\bar{y} - y_t)r \\ &\geq \frac{1}{2}\alpha_t(b_t - y_t)^2 - \frac{1}{2}\beta_t(r - y_t)^2, \end{aligned} \tag{5.6}$$

for some positive constants α_t and β_t. Consider the function

$$F(r) = \frac{1}{2}\alpha_t(\hat{y}_t - y_t)^2 - \frac{1}{2}\beta_t(r - y_t)^2 - \log(1 + \eta(\bar{y} - y_t)\hat{y}_t) + \eta(\bar{y} - y_t)r.$$

Equation 5.6 is equivalent to $F(r) \leq 0, \forall r$. It can be seen that $F(r)$ is maximized when $r = y_t + \frac{\eta}{\beta_t}(\bar{y} - y_t)$. Substituting for r in $F(r)$ and simplifying, we get:

$$\frac{1}{2}\alpha_t(\hat{y}_t - y_t)^2 + \frac{\eta^2}{2\beta_t}(\bar{y} - y_t)^2 - \log(1 + \eta(\bar{y} - y_t)\hat{y}_t) + \eta_t(\bar{y} - y_t)y_t.$$

Hence, we need to prove that

$$G(y_t) = \frac{1}{2}\alpha_t(\hat{y}_t - y_t)^2 + \frac{\eta^2}{2\beta_t}(\bar{y} - y_t)^2 - \log(1 + \eta(\bar{y} - y_t)\hat{y}_t) + \eta_t(\bar{y} - y_t)y_t \leq 0,$$

for all y_t.

Using Lemma 6,

$$\frac{dG}{dy_t} = -\alpha_t(\hat{y}_t - y_t) - \frac{\eta^2}{\beta_t}\frac{(\bar{y} - y_t)}{1 + \eta\bar{y}^2} + \frac{1}{1 + \eta(\bar{y} - y_t)\hat{y}_t}\frac{\eta\hat{y}_t}{1 + \eta\bar{y}^2} + \eta(\bar{y} - y_t) - \frac{\eta y_t}{1 + \eta\bar{y}^2}.$$

Now $\frac{1}{1+\eta(\bar{y}-y_t)\hat{y}_t} = \frac{\bar{y}}{\hat{y}_t}$. Therefore,

$$\frac{dG}{dy_t} = -\alpha_t(\hat{y}_t - y_t) - \frac{\eta^2}{\beta_t}\frac{(\bar{y} - y_t)}{1 + \eta\bar{y}^2} + \frac{\eta(\bar{y} - y_t)}{1 + \eta_t\bar{y}^2} + \eta(\bar{y} - y_t).$$

Now let $\beta_t = \eta \geq 0$. Hence,

$$\frac{dG}{dy_t} = -\alpha_t(\hat{y}_t - y_t) + \eta(\bar{y} - y_t).$$

Now it can be seen that if $y_t = \hat{y}_t \implies y_t = \bar{y}$. Therefore, the optimum for $G(y_t)$ is achieved at $y_t = \bar{y} = \hat{y}_t$ and the optimal value is $G = 0$.

Now consider:

$$\frac{d^2G}{dy_t^2} = \alpha_t - \frac{\eta}{1 + \eta\bar{y}^2}.$$

Hence $y_t = \bar{y}$ is maxima for G iff,

$$\alpha_t \leq \frac{\eta}{1 + \eta\bar{y}^2},$$

for all $\bar{y}$. Using Lemma 5, $y_t = \bar{y}$ is maxima for G if

$$\alpha_t \leq \frac{\eta}{1 + \eta\left(\frac{R}{2} + \sqrt{\frac{R^2}{4} + \frac{1}{\eta}}\right)^2}.$$

As $G = 0$ at $y_t = \bar{y}$, for $\beta_t = \eta$ and $\alpha \leq \frac{\eta}{1+\eta\left(\frac{R}{2}+\sqrt{\frac{R^2}{4}+\frac{1}{\eta}}\right)^2}$, the lemma holds. $\square$

Theorem 8.

$$L_W \leq \left(1+\eta\left(\frac{R}{2}+\sqrt{\frac{R^2}{4}+\frac{1}{\eta}}\right)^2\right)L_{W^*}+\left(\frac{1}{\eta}+\left(\frac{R}{2}+\sqrt{\frac{R^2}{4}+\frac{1}{\eta}}\right)^2\right)D_{ld}(W^*,W_0),$$

where $L_W = \sum_t \ell(\hat{y}_t, y_t)$ *is the loss incurred by the series of matrices* W_t *generated by Equation* (5.4)*,* $W_0 \succ 0$ *is the initial matrix, and* W^* *is the optimal offline solution.*

Proof. The bound is obtained by summing the loss at each step using Lemma 7:

$$\sum_t \left(\frac{1}{2}\alpha_t(\hat{y}_t-y_t)^2-\frac{1}{2}\beta_t(d_{W^*}(\boldsymbol{u}_t,\boldsymbol{v}_t)-y_t)^2\right) \leq \sum_t \left(D_{ld}(W^*,W_t)-D_{ld}(W^*,W_{t+1})\right).$$

The result follows by plugging in the appropriate α_t and β_t, and observing that the right-hand side telescopes to $D_{ld}(W^*,W_0)-D_{ld}(W^*,W_{t+1}) \leq D_{ld}(W^*,W_0)$ since $D_{ld}(W^*,W_{t+1}) \geq 0$. □

For the squared hinge loss $\ell(\hat{y}_t, y_t, b_t) = \max(0, b_t(\hat{y}_t - y_t))^2$, the corresponding algorithm has the same bound.

The regularization parameter affects the tradeoff between L_{W^*} and $D_{ld}(W^*,W_0)$: as η gets larger, the coefficient of L_{W^*} grows while the coefficient of $D_{ld}(W^*,W_0)$ shrinks. In most scenarios, R is small; for example, in the case when $R=2$ and $\eta=1$, then the bound is $L_W \leq (4+\sqrt{2})L_{W^*}+2(4+\sqrt{2})D_{ld}(W^*,W_0)$. Furthermore, in the case when there exists an offline solution with zero error, i.e., $L_{W^*}=0$, then with a sufficiently large regularization parameter, we know that $L_W \leq 2R^2 D_{ld}(W^*,W_0)$. This bound is analogous to the bound proven in Theorem 1 of the POLA method [93]. Note, however, that

our bound is much more favorable to scaling of the optimal solution W^*, since the bound of POLA has a $\|W^*\|_F^2$ term while our bound uses $D_{ld}(W^*, W_0)$: if we scale the optimal solution by c, then the $D_{ld}(W^*, W_0)$ term will scale by $O(c)$, whereas $\|W^*\|_F^2$ will scale by $O(c^2)$. Similarly, our bound is tighter than that provided by the ITML-Online algorithm since, in the ITML-Online algorithm, the regularization parameter η_t for step t is dependent on the input data. An adversary can always provide an input $(\boldsymbol{u}_t, \boldsymbol{v}_t, y_t)$ so that the regularization parameter has to be decreased arbitrarily; that is, the need to maintain positive defininteness for each update can prevent ITML-Online from making progress towards an optimal metric.

In summary, we have proven a regret bound for the proposed LEGO algorithm, an online metric learning algorithm based on LogDet regularization and gradient descent. Our algorithm automatically enforces positive definiteness every iteration and is simple to implement. The bound is comparable to POLA's bound but is more favorable to scaling, and is stronger than ITML-Online's bound.

5.2 Fast Online Similarity Searches

As seen in Chapter 4, for many real-world applications, metric learning is used in conjunction with nearest-neighbor searching, and data structures to facilitate such searches are essential. For online metric learning to be practical for large-scale retrieval applications, we must be able to efficiently index the data as updates to the metric are performed. This poses a problem for most

fast similarity searching algorithms, since each update to the online algorithm would require a costly update to their data structures.

Our goal is to avoid expensive naive updates, where all database items are re-inserted into the search structure. We employ *locality-sensitive hashing* (LSH) to enable fast queries; but rather than re-hash all database examples every time an online constraint alters the metric, we show how to incorporate a second level of hashing that determines which hash bits are changing during the metric learning updates. This allows us to avoid costly exhaustive updates to the hash keys, though occasional updating is required after substantial changes to the metric are accumulated.

Recall, that LSH [35, 21] produces a binary hash key $H(\boldsymbol{u}) = [h_1(\boldsymbol{u})h_2(\boldsymbol{u})...h_b(\boldsymbol{u})]$ for every data point. The locality-sensitive hash function $h(\cdot)$ must satisfy the following property: $Pr[h_i(\boldsymbol{u}) = h_i(\boldsymbol{v})] = \text{sim}(\boldsymbol{u}, \boldsymbol{v})$, where $\text{sim}(\cdot, \cdot)$ denotes a similarity function with values between 0 and 1. In this work, we use the following hash function developed in Section 4.1:

$$h_{\boldsymbol{r},W}(\boldsymbol{u}) = \begin{cases} 1, & \text{if } \boldsymbol{r}^T G \boldsymbol{u} \geq 0 \\ 0, & \text{otherwise,} \end{cases} \tag{5.7}$$

where $W = G^T G$ and $\boldsymbol{r}$ is the normal to a random hyperplane.

5.2.1 Online Hashing Updates

The approach described in Chapter 4 is not immediately amenable to online updates. We can imagine producing a series of LSH functions $h_{\boldsymbol{r}_1,W}, ..., h_{\boldsymbol{r}_b,W}$, and storing the corresponding hash keys for each data point

in our database. However, the hash functions as given in (5.7) are dependent on the Mahalanobis distance; when we update our matrix W_t to W_{t+1}, the corresponding hash functions, parameterized by G_t, must also change. To update all hash keys in the database would require $O(nd)$ time, which may be prohibitive. In the following we propose a more efficient approach.

Recall the update for W: $W_{t+1} = W_t - \frac{\eta(\bar{y}-y_t)W_t\boldsymbol{z}_t\boldsymbol{z}_t^TW_t}{1+\eta(\bar{y}-y_t)\hat{y}_t}$, which we will write as $W_{t+1} = W_t + \beta_t W_t\boldsymbol{z}_t\boldsymbol{z}_t^TW_t$, where $\beta_t = -\eta(\bar{y}-y_t)/(1+\eta(\bar{y}-y_t)\hat{y}_t)$. Let $G_t^TG_t = W_t$. Then $W_{t+1} = G_t^T(I+\beta_tG_t\boldsymbol{z}_t\boldsymbol{z}_t^TG_t^T)G_t$. The square-root of $I+\beta_tG_t\boldsymbol{z}_t\boldsymbol{z}_t^TG_t^T$ is $I+\alpha_tG_t\boldsymbol{z}_t\boldsymbol{z}_t^TG_t^T$, where $\alpha_t = (\sqrt{1+\beta_t\boldsymbol{z}_t^TW_t\boldsymbol{z}_t}-1)/(\boldsymbol{z}_t^TW_t\boldsymbol{z}_t)$. As a result, $G_{t+1} = G_t + \alpha_tG_t\boldsymbol{z}_t\boldsymbol{z}_t^TW_t$. The corresponding update to (5.7) is to find the sign of

$$\boldsymbol{r}^TG_{t+1}\boldsymbol{x} = \boldsymbol{r}^TG_t\boldsymbol{u} + \alpha_t\boldsymbol{r}^TG_t\boldsymbol{z}_t\boldsymbol{z}_t^TW_t\boldsymbol{u}. \tag{5.8}$$

Suppose that the hash functions have been updated in full at some time step t_1 in the past. Now at time t, we want to determine which hash bits have flipped since t_1, or more precisely, which examples' product with some $\boldsymbol{r}^TG_t$ has changed from positive to negative, or vice versa. This amounts to determining all bits such that $\text{sign}(\boldsymbol{r}^TG_{t_1}\boldsymbol{u}) \neq \text{sign}(\boldsymbol{r}^TG_t\boldsymbol{u})$, or equivalently, $(\boldsymbol{r}^TG_{t_1}\boldsymbol{u})(\boldsymbol{r}^TG_t\boldsymbol{u}) \leq 0$. Expanding the update given in (5.8), we can write $\boldsymbol{r}^TG_t\boldsymbol{u}$ as $\boldsymbol{r}^TG_{t_1}\boldsymbol{u} + \sum_{\ell=t_1}^{t-1}\alpha_\ell\boldsymbol{r}^TG_\ell\boldsymbol{z}_\ell\boldsymbol{z}_\ell^TW_\ell\boldsymbol{u}$. Therefore, finding the bits that have changed sign is equivalent to finding all $\boldsymbol{u}$ such that $(\boldsymbol{r}^TG_{t_1}\boldsymbol{u})^2 + (\boldsymbol{r}^TG_{t_1}\boldsymbol{u})\left(\sum_{\ell=t_1}^{t-1}\alpha_\ell\boldsymbol{r}^TG_\ell\boldsymbol{z}_\ell\boldsymbol{z}_\ell^TW_\ell\boldsymbol{u}\right) \leq 0$. We can use a second level of locality-sensitive hashing to *approximately* find all such $\boldsymbol{u}$. Define a vector $\bar{\boldsymbol{u}} =$

$[(\boldsymbol{r}^T G_{t_1} \boldsymbol{u})^2; (\boldsymbol{r}^T G_{t_1} \boldsymbol{u})\boldsymbol{u}]$ and a "query" $\bar{\boldsymbol{q}} = [-1; -\sum_{\ell=t_1}^{t-1} \alpha_\ell \boldsymbol{r}^T W_\ell \boldsymbol{z}_\ell \boldsymbol{z}_\ell^T G_\ell]$. Then the bits that have changed sign can be approximately identified by finding all examples $\bar{\boldsymbol{u}}$ such that $\bar{\boldsymbol{q}}^T \bar{\boldsymbol{u}} \geq 0$. In other words, we look for all $\bar{\boldsymbol{u}}$ that have a large inner product with $\bar{\boldsymbol{q}}$, which translates the problem to a similarity search problem. This may be solved approximately using the locality-sensitive hashing scheme given in [21] for inner product similarity. Note that finding $\bar{\boldsymbol{u}}$ for each $\boldsymbol{r}$ can be computationally expensive, so we search $\bar{\boldsymbol{u}}$ for only a randomly selected subset of the vectors $\boldsymbol{r}$.

In summary, when performing online metric learning updates, instead of updating all the hash keys at every step (which costs $O(nd)$), we delay updating the hash keys and instead determine approximately which bits have changed in the stored entries in the hash table since the last update. When we have a nearest-neighbor query, we can quickly determine which bits have changed, and then use this information to find a query's approximate nearest neighbors using the current metric. Once many of the bits have changed, we perform a full update to our hash functions.

Finally, we note that the above can be extended to the case where computations are done in kernel space.

5.3 Experimental Results

In this section we evaluate the proposed algorithm (LEGO) over a variety of data sets, and examine both its online metric learning accuracy as well as the quality of its online similarity search updates. As baselines, we

consider the most relevant techniques from the literature: the online metric learners POLA [93] and ITML-Online [24]. We also evaluate a baseline offline metric learner associated with our method. For all metric learners, we gauge improvements relative to the original (non-learned) Euclidean distance, and our classification error is measured with the k-nearest neighbor algorithm.

First we consider the same collection of UCI data sets used in [24]. For each data set, we provide the online algorithms with 10,000 randomly-selected constraints, and generate their target distances as in [24]—for same-class pairs, the target distance is set to be equal to the 5th percentile of all distances in the data, while for different-class pairs, the 95th percentile is used. To tune the regularization parameter η for POLA and LEGO, we apply a pre-training phase using 1,000 constraints. (This is not required for ITML-Online, which automatically sets the regularization parameter at each iteration to guarantee positive definiteness). The final metric (A_T) obtained by each online algorithm is used for testing (T is the total number of time-steps). Figure 5.1 shows the k-nn error rates for all five data sets. LEGO outperforms the Euclidean baseline as well as the other online learners, and even approaches the accuracy of the offline method (see [24] for additional comparable offline learning results using [36, 108]). LEGO and ITML-Online have comparable running times. However, our approach has a significant speed advantage over POLA on these data sets: on average, learning with LEGO is 16.6 times faster, most likely due to the extra projection step required by POLA.

Next we evaluate our approach on a handwritten digit classification

task, reproducing the experiment used to test POLA in [93]. We use the same settings given in that paper. Using the MNIST data set, we pose a binary classification problem between each pair of digits (45 problems in all). The training and test sets consist of 10,000 examples each. For each problem, 1,000 constraints are chosen and the final metric obtained is used for testing. Figure 5.2 compares the test error between POLA and LEGO. Note that LEGO beats or matches POLA's test error in 33/45 (73.33%) of the classification problems. Based on the additional baselines provided in [93], this indicates that our approach also fares well compared to other offline metric learners on this data set.

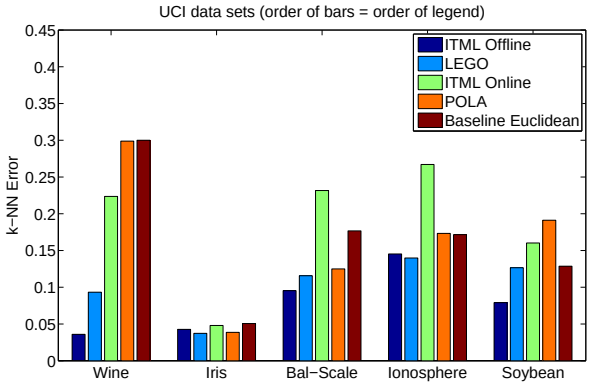


Figure 5.1: Comparison with existing online metric learning methods on the UCI datasets. Our method (LEGO) outperforms both the Euclidean distance baseline as well as existing metric learning methods, and even approaches the accuracy of the offline algorithm.

We next consider a set of image patches from the Photo Tourism

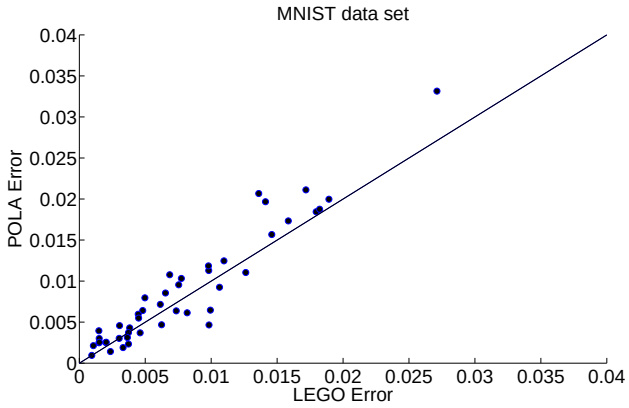


Figure 5.2: Comparison of errors for LEGO and POLA on 45 binary classification problems using the MNIST data; LEGO matches or outperforms POLA on 33 of the 45 total problems.

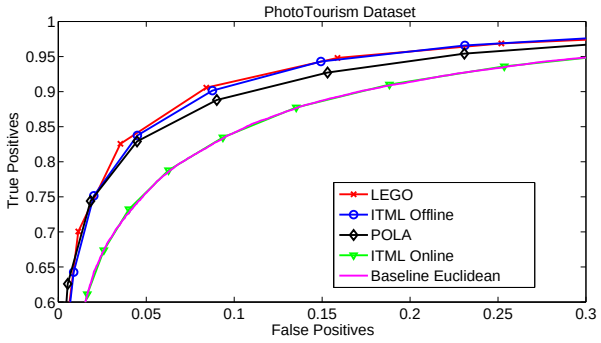


Figure 5.3: Comparison with existing online metric learning methods on the Photo Tourism data. Our online algorithm significantly outperforms the L_2 baseline and ITML-Online, does well relative to POLA, and nearly matches the accuracy of the offline method.

project [96], where user photos from Flickr are used to generate 3-d reconstructions of various tourist landmarks. Forming the reconstructions requires solving for the correspondence between local patches from multiple images of the same scene. We use the publicly available data set that contains about

300,000 total patches from images of three landmarks[1]. Each patch has a dimensionality of 4096, so for efficiency we apply all algorithms in kernel space, and use a linear kernel. The goal is to learn a metric that measures the distance between image patches better than L_2, so that patches of the same 3-d scene point will be matched together, and (ideally) others will not. Since the database is large, we can also use it to demonstrate our online hash table updates. Following [51], we add random jitter (scaling, rotations, shifts) to all patches, and generate 50,000 patch constraints (50% matching and 50% non-matching patches) from a mix of the Trevi and Halfdome images. We test with 100,000 patch pairs from the Notre Dame portion of the data set, and measure accuracy with precision and recall.

Figure 5.3 shows that LEGO and POLA are able to learn a distance function that significantly outperforms the baseline squared Euclidean distance. However, LEGO is more accurate than POLA, and again nearly matches the performance of the offline metric learning algorithm. On the other hand, the ITML-Online algorithm does not improve beyond the baseline. We attribute the poor accuracy of ITML-Online to its need to continually adjust the regularization parameter to maintain positive definiteness; in practice, this often leads to significant drops in the regularization parameter, which prevents the method from improving over the Euclidean baseline. In terms of training time, on this data LEGO is 1.42 times faster than POLA (on average over 10 runs).

[1] http://phototour.cs.washington.edu/patches/default.htm

Finally, we present results using our online metric learning algorithm together with our online hash table updates described in Section 5.2.1 for the Photo Tourism data. For our first experiment, we provide each method with 50,000 patch constraints, and then search for nearest neighbors for 10,000 test points sampled from the Notre Dame images. Figure 5.4 shows the recall as a function of the number of patches retrieved for four variations: LEGO with a linear scan, LEGO with our LSH updates, the L_2 baseline with a linear scan, and L_2 with our LSH updates. The results show that the accuracy achieved by our LEGO+LSH algorithm is comparable to the LEGO+linear scan (and similarly, L_2+LSH is comparable to L_2+linear scan), thus validating the effectiveness of our online hashing scheme. Moreover, LEGO+LSH needs to search only 10% of the database, which translates to an approximate speedup factor of 4.7 over the linear scan for this data set.

Next we show that LEGO+LSH performs accurate and efficient retrievals in the case where constraints and queries are interleaved in any order. Such a scenario is useful in many applications: for example, an image retrieval system such as Flickr continually acquires new image tags from users (which could be mapped to similarity constraints), but must also continually support intermittent user queries. For the Photo Tourism setting, it would be useful in practice to allow new constraints indicating true-match patch pairs to stream in while users continually add photos that should participate in new 3-d reconstructions with the improved match distance functions. To experiment with this scenario, we randomly mix online additions of 50,000 constraints with

10,000 queries, and measure performance by the recall value for 300 retrieved nearest neighbor examples. We recompute the hash-bits for all database examples if we detect changes in more than 10% of the database examples.

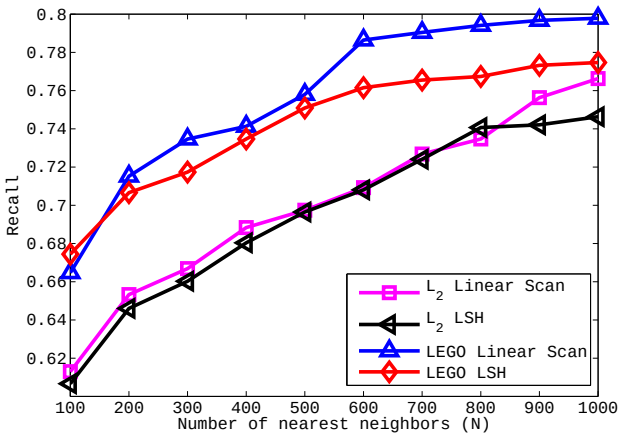


Figure 5.4: The plot shows the recall value for increasing numbers of nearest neighbors retrieved. 'LEGO LSH' denotes LEGO metric learning in conjunction with online searches using our LSH updates, 'LEGO Linear' denotes LEGO learning with linear scan searches. L_2 denotes the baseline Euclidean distance. The right plot shows the average recall values for all methods at different time instances as more queries are made and more constraints are added. Our online similarity search updates make it possible to efficiently interleave online learning and querying. See text for details.

Figure 5.5 compares the average recall value for various methods after each query. As expected, as more constraints are provided, the LEGO-based accuracies all improve (in contrast to the static L_2 baseline, as seen by the straight line in the plot). Our method achieves similar accuracy to both the linear scan method (LEGO Linear) as well as the naive LSH method where

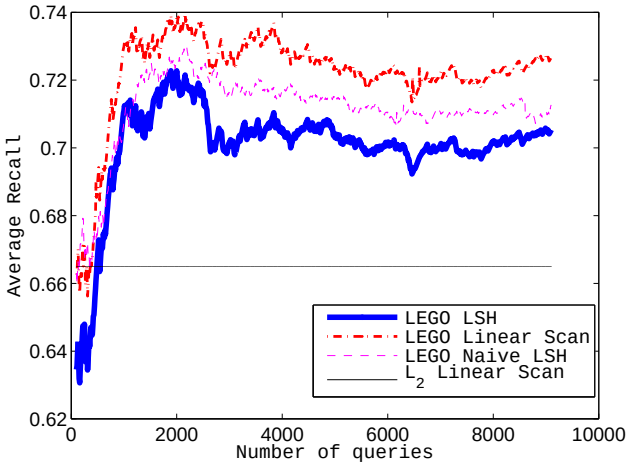


Figure 5.5: 'LEGO LSH' denotes LEGO metric learning in conjunction with online searches using our LSH updates, 'LEGO Linear' denotes LEGO learning with linear scan searches. L_2 denotes the baseline Euclidean distance. The plot shows the average recall values for all methods at different time instances as more queries are made and more constraints are added. Our online similarity search updates make it possible to efficiently interleave online learning and querying. See text for details.

the hash table is fully recomputed after every constraint update (LEGO Naive LSH). The curves stack up appropriately given the levels of approximation: LEGO Linear yields the upper bound in terms of accuracy, LEGO Naive LSH with its exhaustive updates is slightly behind that, followed by our LEGO LSH with its partial and dynamic updates. In reward for this minor accuracy loss, however, our method provides a speedup factor of 3.8 over the naive LSH update scheme. (In this case the naive LSH scheme is actually slower than a linear scan, as updating the hash tables after every update incurs a large overhead cost.) For larger data sets, we can expect even larger speed

improvements.

5.4 Summary

In this chapter, we considered the problem of online metric learning where side-information is provided in an online fashion. For this problem, we developed an online metric learning algorithm that performs inexpensive updates to the metric each time some new side-information in the form of distance constraints is available. We showed that our online learner offers improved reliability over state-of-the-art online metric learning methods in terms of regret bounds, and empirical performance. As in Chapter 4, we applied our learned metric to the problem of fast similarity search. For this task, we proposed a method to perform online updates to the fast similarity search structures, and demonstrated its applicability and advantages on a variety of data sets.

Chapter 6

Geometry-aware Kernel Learning

In previous chapters, we studied the problem of distance/kernel learning in offline and online settings, and applied it to the task of sublinear time similarity search. We parameterized the distance/kernel function using a Mahalanobis metric W with a quadratic number of parameters in the dimensionality d, or the training set size n. For large datasets, we restricted the number of parameters to be linear in $\min(d, n)$. However, this requires pre-specifying a small-dimensional basis upfront which might be too restrictive for many real-world applications.

For typical real-world applications, even though the dimensionality of the feature space is large, the intrinsic dimensionality of the data is significantly smaller. In the next two chapters, we exploit this intuition by imposing different kinds of structural assumptions on the data to learn kernel functions using a small amount of supervision.

In this chapter, we assume that the data is obtained from a low-dimensional manifold. The goal is then to incorporate the intrinsic structure in the data, while learning a task-dependent kernel using the provided side-information. Specifically, we jointly model a task-dependent kernel as well as

a data-dependent kernel that reflects the local geometry or manifold structure of the data. We show that for some important parameterizations of the set of data-dependent kernels, our formulation admits convexity, and the proposed optimization algorithm efficiently learns an appropriate kernel function for the given task. Our algorithm is fast, scalable, does not involve semi-definite programming, and crucially, is able to exploit the low dimensional structure of the underlying manifold that is often present in real-world datasets.

Our proposed framework is generic and can be easily tailored for a variety of tasks. We apply our method to the task of classification (inductive and transductive setting), automatic model selection for standard kernel functions, and semi-supervised manifold learning. For each application, we empirically demonstrate that our method can achieve comparable or better performance than the respective state-of-the-art.

This work on geometric kernel learning was published in [76].

6.1 Methodology

Given a set of n points $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\} \in \mathbb{R}^d$, we seek a positive semi-definite kernel matrix K that can be later used for various tasks, *e.g.* classification, retrieval, etc. Our goal is two-fold: 1) use the provided supervision over the data, 2) exploit the unlabeled or unsupervised data, i.e., we want to learn a kernel that respects the underlying manifold structure in the data while also incorporating the side-information provided. Previous kernel learning approaches typically handle this problem by learning a spectral kernel

$K = \sum_i \alpha_i \boldsymbol{v}_i \boldsymbol{v}_i^T$, where the vectors $\boldsymbol{v}_i$ are the low-frequency eigenvectors of the Laplacian of a k-NN graph. However, constraining the eigenvectors to be unchanged severely restricts the class of kernels that can be learned.

A contrasting task-dependent approach to kernel learning is based on the metric learning paradigm introduced in Chapter 3. Here, the goal is to learn a kernel K that is "close" to a pre-defined baseline kernel K_0 and satisfies the provided pairwise (or relative) constraints that are specific to the task at hand. Formally, K is obtained by solving the following problem:

$$\min_K \; D(K, K_0), \quad \text{s.t.} \quad K \in \mathcal{K},$$

where $\mathcal{K}$ is a convex set of kernel K that satisfy

$$\begin{aligned} &K_{ii} + K_{jj} - 2K_{ij} \leq u \quad (i,j) \in \mathcal{S}, \\ &K_{ii} + K_{jj} - 2K_{ij} \geq l \quad (i,j) \in \mathcal{D}, \\ &K \succeq 0. \end{aligned} \tag{6.1}$$

In the above $\mathcal{S}$ is the given set of similar points, $\mathcal{D}$ is the given set of dis-similar points, and $D(\cdot, \cdot)$ is a distance function for comparing two kernel matrices. We will denote the set of kernel that satisfy (6.1) as the set of task-dependent kernel. Although flexible and effective for various problems, this framework does not account for the unlabeled data and their geometry. As a result, large amount of supervision is required to capture the intrinsic structure in the data.

In this chapter, we propose a geometry-aware metric learning (G-ML) framework that combines both the data-dependent and task-dependent kernel

learning approaches. Our model maintains the flexibility of the metric learning based approach while exploiting the intrinsic structure in the data, and as we shall show later, engenders multiple competitive machine learning models.

6.1.1 Geometry-aware Metric Learning

In this section, we describe our geometry-aware metric learning (G-ML) model, where we learn the kernel K, as well as the kernel M that explicitly exploits the intrinsic structure in the data through the optimization problem:

$$\min_{K,M} \; D(K, M), \quad \text{s.t.} \quad K \in \mathcal{K}, \; M \in \mathcal{M}, \tag{6.2}$$

where the set $\mathcal{M}$ is a parametric set of kernels that capture the intrinsic geometry of the labeled as well as unlabeled data and $D(\cdot, \cdot)$ is a distance function over matrices. The above optimization problem computes kernel K that satisfies task specific constraints (6.1) and is also close to kernel M, thus incorporating data geometry into the kernel K (see Figure 6.1). Later in the section, we give a few interesting examples of the set $\mathcal{M}$.

A key component of our framework is the distance function $D(K, M)$ that is being used. Similar to ITML (see Section 2.1.1), we also use the LogDet matrix divergence, $D_{\ell d}(K, M)$, where $D_{\ell d}(K, M) = \text{tr}(KM^{-1}) - \log\det(KM^{-1}) - n$.

Now, we give an important example of the set $\mathcal{M}$ based on spectral learning methods [112] that captures the underlying structure in the data. First, define a graph $\mathcal{G}$ over the data points that captures local structure of

data, *e.g.*, a k-NN graph or an ϵ-ball graph. Let W be the adjacency matrix of $\mathcal{G}$, D be the degree matrix, L be the graph Laplacian[1] $L = D - W$, and $V = [\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_r]$ be the r eigenvectors of L (typically $r \ll n$) corresponding to the smallest eigenvalues of L: $\lambda_1 \leq \cdots \leq \lambda_r$. Then, the set $\mathcal{M}$ we consider is given by:

$$\mathcal{M} = \left\{ \sum_i^r \alpha_i \boldsymbol{v}_i \boldsymbol{v}_i^T | \ \alpha_1 \geq \alpha_2 \geq \cdots \geq \alpha_r \geq 0. \right\} \tag{6.3}$$

where the order constraints $\alpha_1 \geq \alpha_2 \geq \cdots \geq \alpha_r \geq 0$ further ensure smoothness (the eigenvector $\boldsymbol{v}_i$ is known to be smoother than $\boldsymbol{v}_{i+1}$).

For this particular choice of $\mathcal{M}$, the kernel K is obtained by solving the following optimization problem:

$$\begin{aligned} \min_{K, \alpha_1, \alpha_2, \ldots, \alpha_r} \quad & D_{\ell d}(K, M) \\ \text{s.t. } & K \in \mathcal{K}, \ M = \sum_i^r \alpha_i \boldsymbol{v}_i \boldsymbol{v}_i^T, \\ & \alpha_1 \geq \alpha_2 \geq \cdots \geq \alpha_r \geq 0. \end{aligned} \tag{6.4}$$

Solving above problem yields $\{\alpha_1, \alpha_2, \ldots, \alpha_r\}$ in the cone $\alpha_1 \geq \alpha_2 \geq \cdots \geq \alpha_r \geq 0$ and a feasible kernel K that is close to $M = \sum_i^r \alpha_i \boldsymbol{v}_i \boldsymbol{v}_i^T$ (see Figure 6.1). Slack variables can be incorporated in our framework to ensure that the set $\mathcal{K}$ is always feasible, even under noisy constraints.

6.1.2 Alternative $\mathcal{M}$

In the above subsection, we discussed an example of $\mathcal{M}$ as a particular subset of spectral kernels. However our framework is general and depending on

[1] We can also use the normalized Laplacian $I - D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$.

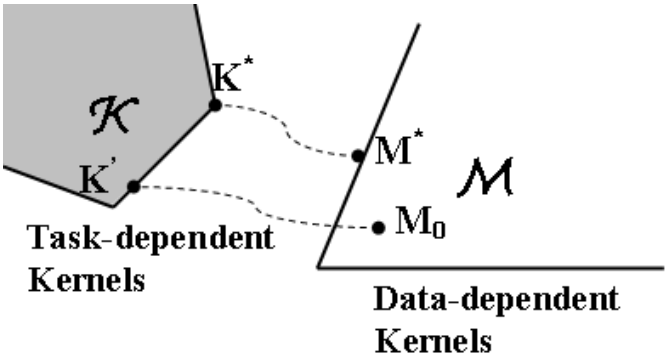


Figure 6.1: Illustration of G-ML. The shadowed polygon stands for the feasible set of kernels K specified by the task dependent pairwise constraints. The cone stands for data-dependent kernels that exploits the intrinsic geometry of the data. Using a fixed M_0 would lead to sub-optimal kernel K', while the joint optimization (as in (6.2)) over both M and K leads to a better solution K^*.

the application it can admit other parametric sets also. For example, consider the set:

$$\mathcal{M} = \{S - S(I + TS)^{-1}TS |\ T = \sum_{i}^{r} \theta_i \boldsymbol{v}_i \boldsymbol{v}_i^T, \\ \theta_1 \geq \cdots \geq \theta_r \geq 0.\} \quad (6.5)$$

where S is a fixed given kernel and the vectors $\boldsymbol{v}_i$ are eigenvectors of the graph Laplacian L. This set generalizes the data-dependent kernel proposed by [94] by replacing the graph Laplacian with a more flexible T. Note that $\mathcal{M}$ given by (6.5) reduces to $\mathcal{M}$ given by (6.3) in the limit $\|S^{-1}\| \to 0$. This set of kernel is interesting in that, unlike most spectral kernels that are usually evaluated in a transductive setting, the kernel value can be naturally extended to unseen samples as

$$M(\boldsymbol{x}, \boldsymbol{x}') = S(\boldsymbol{x}, \boldsymbol{x}') - S(\boldsymbol{x}, .)(I + TS)^{-1}TS(., \boldsymbol{x}')$$

As will be shown in Section 4, the set $\mathcal{M}$ given by (6.3) as well as (6.5) both lead to convex sub-problems for finding T with fixed K. In general, the convexity holds if $\{\boldsymbol{v}_1, \cdots, \boldsymbol{v}_r\}$ are orthogonal, which allows us to extend our model to other manifold learning models [13], such as Isomap or LLE. The set $\mathcal{M}$ can also be adapted to perform automatic model selection for *supervised learning*, for example we can tune the parameter for the RBF kernels by letting

$$\mathcal{M} = \left\{ \alpha \exp(-\frac{||\boldsymbol{x}_i - \boldsymbol{x}_j||^2}{2\sigma^2}) \mid \alpha > 0, \sigma > 0 \right\}, \tag{6.6}$$

where α and σ are parameters to be learned by G-ML.

6.2 Algorithm

In this section, we analyze properties of the proposed optimization problem (6.4) and propose a fast and scalable algorithm. First, note that although the constraints specified in (6.4) are all linear, the objective function $D_{\ell d}(K, M)$ is not jointly convex in K and M. However, the problem can be shown to be convex individually in K and M^{-1}. Here and in the remainder of this chapter, whenever the inverse of a matrix does not exist, we use its Moore-Penrose inverse. It is easy to see that on fixing M, the problem is strictly convex in K as $D_{\ell d}(K, M)$ is known to be convex in K [24]. The following lemma shows that (6.4) is also convex in the parameters $\frac{1}{\alpha_i}, 1 \leq i \leq r$, when K is fixed.

Lemma 8. *Assuming K to be fixed, Problem* (6.4) *is convex in $\beta_1 = 1/\alpha_1, \beta_2 = 1/\alpha_2, \ldots, \beta_r = 1/\alpha_r$.*

Algorithm 1 Geometry-aware Metric Learning(G-ML)
Optimization procedure when $\mathcal{M}$ is given by (6.3)

Input: X: input $d \times n$ matrix, $\mathcal{S}$: similarity constraints
$\mathcal{D}$: dis-similarity constraints, $\boldsymbol{\alpha}^0$: initial $\boldsymbol{\alpha}$
γ: slack parameter, r: number of eigenvectors
Output: K, M

$\mathcal{G}$ =kNN-graph(X), W =Affinity matrix of $\mathcal{G}$
$L = D - W$, $L = \sum_i^n \mu_i \boldsymbol{v}_i \boldsymbol{v}_i^T$
$M = \sum_i^r \alpha_i^0 \boldsymbol{v}_i \boldsymbol{v}_i^T$
repeat
  $K = \mathsf{ITML}(M,\ \mathcal{S},\ \mathcal{D},\ \gamma)$ //(Step A)
  $\boldsymbol{\alpha} = \mathsf{FindAlpha}(K, \boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_r)$ //(Step B)
  $M = \sum_i \alpha_i \boldsymbol{v}_i \boldsymbol{v}_i^T$
until convergence

---

function $\boldsymbol{\alpha} = \mathsf{FindAlpha}(K, \boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_r)$
Cyclic projection method to solve (6.4) with fixed K

$\alpha_i = \boldsymbol{v}_i^T K \boldsymbol{v}_i, 1 \leq i \leq r$
$\boldsymbol{\nu} = 0, i = 0$
repeat
  $c = \min(\nu_i, (\alpha_{i+1} - \alpha_i)/2)$
  $\nu_i = \nu_i - c, \alpha_{i+1} = \alpha_{i+1} - c, \alpha_i = \alpha_i + c$
  $i =\ \mod (i+1, n)$
until convergence

Proof. Since $M^{-1} = \sum_i \beta_i \boldsymbol{v}_i \boldsymbol{v}_i^T$, where $\beta_i = \frac{1}{\alpha_i}$, the fact that $D_{\ell d}(K, M) = D_{\ell d}(M^{-1}, K^{-1})$ is convex in M^{-1} implies convexity in $\beta_i, \forall i$. Furthermore, the constraints $\alpha_1 \geq \alpha_2 \geq \cdots \geq \alpha_r \geq 0$ can be equivalently written as a set of linear constraints $\beta_r \geq \cdots \geq \beta_2 \geq \beta_1 > 0$. □

Now, we describe our proposed alternating minimization algorithm for solving (6.4). Our algorithm is based on individual convexity of (6.4) w.r.t K and M^{-1}. It iterates by fixing M (or equivalently $\alpha_1, \alpha_2, \ldots, \alpha_r$) to solve for

K (denoted Step A), and then fixing K to solve for $\alpha_1, \alpha_2, \ldots, \alpha_r$ (Step B). In Step A, to find K, we use the cyclic projection algorithm where at each step we project the current solution onto one of the constraints. The projection problem that needs to be solved at each step is:

$$\min_{K} \; D_{\ell d}(K, K_t), \quad \text{s.t.} \;\; K_{ii} + K_{jj} - 2K_{ij} \leq u,$$

i.e., projection w.r.t. single (dis-)similarity constraint. As shown in Section 2.1.1, the above problem can be solved in closed form using a one-rank update to K_t. Furthermore, the update can be computed in just $\mathcal{O}(nk)$ operations, where $r \ll n$ is the rank of the kernel M. Now in Step B, to obtain $\alpha_1, \alpha_2, \ldots, \alpha_r$, we solve the equivalent optimization problem:

$$\begin{aligned} \min_{\beta_1, \beta_2, \ldots, \beta_r} \; & D_{\ell d}(\sum_i \beta_i \boldsymbol{v}_i \boldsymbol{v}_i^T, K^{-1}) \\ & \text{s.t. } \beta_r \geq \beta_{r-1} \geq \cdots \geq \beta_1 \geq 0, \end{aligned}$$

where $\beta_i = 1/\alpha_i$. This problem can also be solved using cyclic projection, where at each step the current solution is projected onto one of the inequality constraints. Every projection step can be performed in just $\mathcal{O}(k)$ operations.

In summary, we have presented a highly scalable and easy to implement algorithm (Algorithm 1) for solving (6.4). Furthermore, the objective function value achieved by our algorithm is guaranteed to converge.

Alternative $\mathcal{M}$ As mentioned in Section 6.1.2, an alternate set $\mathcal{M}$ given by (6.5) induces an natural out-of-sample extension. Although it is not further

pursued in this thesis, we would like to point out that, similar to (6.4), this alternative set $\mathcal{M}$ also leads to a convex optimization problem for computing M when K is fixed.

Lemma 9. *Assuming K to be fixed, Problem* (6.4) *is convex in $\theta_1, \theta_2, \ldots, \theta_r$.*

Proof. Restricting the kernel function to the provided samples, we get $M = S - S(I + TS)^{-1}TS$. Using the Sherman-Morrison-Woodbury formula, $M^{-1} = S^{-1} + T$. Now, $D_{\ell d}(K, M)$ is convex in M^{-1}. Using the property that a function $g(x) = f(a + x)$ is convex if f is convex, $D_{\ell d}(K, M)$ is convex in T. As T is a linear function of $\theta_i, 1 \leq i \leq r$, $D_{\ell d}(K, M)$ is convex in $\theta_1, \cdots, \theta_r$. □

Using the above lemma, we can adapt Algorithm 6.1.2 to obtain a suboptimal solution to (6.2) where $\mathcal{M}$ is given by (6.5).

Unlike the kernels in (6.3) and (6.5), the set $\mathcal{M}$ given by (6.6) does not admit a convex subprolem when fixing K. However, since only two parameters are involved, we can still adapt our alternative minimization framework to obtain a reasonably efficient method for optimizing (6.2) using $\mathcal{M}$ specified in (6.6).

6.3 Discussion

6.3.1 Connection to Regularization Theory

Now, we present a regularization theory based interpretation of our methodology for estimating kernel K (Problem (6.4)). Using duality theory,

it can be shown that the general form of the solution to (6.4) is given by:

$$K = (\sum_i \alpha_i^{-1} \boldsymbol{v}_i \boldsymbol{v}_i^T + \sum_{(i,j)\in\mathcal{S}} \gamma_{ij}^{\mathcal{S}} (\boldsymbol{e}_i - \boldsymbol{e}_j)(\boldsymbol{e}_i - \boldsymbol{e}_j)^T - \sum_{(i,j)\in\mathcal{D}} \gamma_{ij}^{\mathcal{D}} (\boldsymbol{e}_i - \boldsymbol{e}_j)(\boldsymbol{e}_i - \boldsymbol{e}_j)^T)^{-1} \quad (6.7)$$

with $\gamma_{ij}^{\mathcal{S}}, \gamma_{ij}^{\mathcal{D}} \geq 0$ and $\boldsymbol{e}_i$ being the vector with the i^{th} entry one and rest zeros. Let $f : X \to \mathbb{R}$ be a real valued function over the feature space and $\boldsymbol{f} = [f(\boldsymbol{x}_1), f(\boldsymbol{x}_2), \cdots, f(\boldsymbol{x}_n)]^T$, we then have

$$\boldsymbol{f}^T K^{-1} \boldsymbol{f} = \boldsymbol{f}^T \left(\sum_i \frac{1}{\alpha_i} \boldsymbol{v}_i \boldsymbol{v}_i^T \right) \boldsymbol{f} + \sum_{(i,j)\in\mathcal{S}} \gamma_{ij}^{\mathcal{S}} (f_i - f_j)^2 - \sum_{(i,j)\in\mathcal{D}} \gamma_{ij}^{\mathcal{D}} (f_i - f_j)^2 \quad (6.8)$$

where the first term addresses the overall smoothness of function f on the graph, while the last two terms measures the violation of pairwise constraints. Formulation (6.8) generalizes the joint regularization framework proposed by [94] to include non-positive definite term (dis-similarity term $\sum_{(i,j)\in\mathcal{D}} \gamma_{ij}^{\mathcal{D}} (f_i - f_j)^2$ in our case) in the regularization, while the overall positive definiteness is still ensured either explicitly through another constraint ($K \succeq 0$) or implicitly through the particular optimization algorithm (Bregman projection in our case).

6.3.2 Connection to Gaussian Processes(GP)

Next, we present an interesting connection of our method to that of GP based methods for estimating M. Let $K = \Phi(X)\Phi(X)^T$, where $\Phi(X) =$

$[\phi(\boldsymbol{x}_1)\,\phi(\boldsymbol{x}_2)\,\cdots\,\phi(\boldsymbol{x}_n)]^T$ and $\phi(\boldsymbol{x}_i) \in \mathbb{R}^m$ is the feature space representation of point $\boldsymbol{x}_i$. As in standard GP based methods, assume that each of the feature dimension of $\phi(\boldsymbol{x}_i)$'s are jointly Gaussian with mean 0 and covariance M that needs to be estimated. Thus, the likelihood of the data is given by:

$$\mathcal{L} = \frac{1}{(2\pi)^{n/2}|M|^{1/2}} \exp\left(-\frac{1}{2}\operatorname{tr}\left(\Phi(X)^T M^{-1}\Phi(X)\right)\right).$$

It is easy to see that maximizing the above given likelihood is equivalent to minimizing $D_{\ell d}(K, M)$ with fixed K. Assuming a parametric form for $M = \sum_i \alpha_i \boldsymbol{v}_i \boldsymbol{v}_i^T$, GP based spectral kernel learning is equivalent to learning M using our method. Furthermore, typical GP based methods use one-rank target alignment kernel $K = yy^T$, where y_i is the label of i-th point. In contrast, we use a more robust learned kernel K that not only accounts for the labels, but also the similarity in the data points itself, i.e. our learned kernel K is less likely to overfit to the provided labels and is applicable to a wider class of problems where supervision need not be in the form of labels.

6.4 Applications

In this section, we describe a few applications of our geometry-aware metric learning framework (G-ML) for kernel learning. Besides enhancing existing metric/kernel learning methods, our method also extends the application of kernel learning to a few previously inapplicable tasks as well, e.g., manifold learning tasks.

6.4.1 Classification

First, we describe application of our method to the task of classification in two scenarios: 1) supervised case where the test points are unknown in the training phase, and 2) semi-supervised case where the test/unlabeled points are also part of the training. For both the cases, pairwise similarity/dissimilarity constraints are obtained using the provided labels over the data, and the k nearest neighbor classifier with the learned kernel K is used for predicting the labels. In the supervised learning case, we apply G-ML to the task of automatic model selection by learning the parameters for the baseline kernel M. For semi-supervised learning, G-ML jointly learns the kernel K and the eigenvalues of the spectral kernel M, thereby taking into account the geometry of the unlabeled data. Note that the optimization step for M (step B) is similar to the kernel-target alignment technique for selecting a spectral kernel [112]. However, [112] treat the kernel as a long vector, while our method respects the two-dimensional structure and positive definiteness of the matrix M.

6.4.2 Manifold Learning

G-ML is applicable to semi-supervised manifold learning where the task is to learn and exploit the underlying manifold structure using the provided supervision (pairwise (dis-)similarity constraints). In particular, we apply G-ML to the task of non-linear dimensionality reduction and manifold alignment. In contrast to other metric learning methods [110, 24] that learns the metric over the ambient space, G-ML learns the metric *on the manifold*, where $\{\boldsymbol{v}_i\}$

are the approximate coordinates of the data on the manifold [12].

Colored Dimensionality Reduction Here we consider the semi-supervised dimensionality reduction task where we want to retain both the intrinsic manifold structure of data and the (partial) label information. G-ML naturally merges the two sources of information; the learned kernel K incorporates the manifold structure (as expressed in $\{\alpha_i\}$ and $\{\boldsymbol{v}_i\}$) while reflecting the provided side information (expressed through constraints). Hence, the leading eigenvectors of K should provide a better low-dimensional representation of the data. In absence of any constraints, this dimension reduction model degenerates to Laplacian Eigenmaps [12]. Furthermore, compared to [97], our model is able to learn a more accurate embedding of the data (Figure 6.5).

Manifold Alignment Finally, we apply our method to the task of manifold alignment, where the goal is to align previously disconnected (or weakly connected) manifolds according to some common property. For example, consider images of different objects under a particular transformation, *e.g.* rotation, illumination, scaling etc, which will form a low-dimensional manifold called Lie group. The goal is to estimate information about the *transformation* of the object in the image, rather than the object itself. We show that G-ML accurately represents the corresponding Lie group manifold by aligning the image manifold of different objects under the same transformation (captured by a joint graph Laplacian). This alignment is achieved through learning the

kernel K by constraining a small subset of images with similar transformations to have small distance.

6.5 Experimental Results

In this section, we evaluate our method for geometry-aware metric learning (G-ML) on the applications mentioned in the previous section. Specifically, we apply our method to the task of classification, semi-supervised classification, non-linear dimensionality reduction, and manifold alignment. For each task we compare our method with the respective state-of-the-art methods.

6.5.1 Classification: Supervised Learning

First, we apply our G-ML framework to the task of classification in a supervised learning scenario (Section 6.4.1). For this task, we consider the feasible set $\mathcal{M}$ for M to be scaled Gaussian RBF kernels with unknown scale α and kernel width σ, as in (6.6). Unlike the spectral kernel case, the subproblem for finding α and σ is non-convex and a local optimum for the non-convex subproblem is found with conjugate gradient descent (Matlab function fminsearch). The resulting K is then used for k-NN classification. We evaluate our method (G-ML) on four standard UCI datasets (iris, wine, balance-scale and ionosphere). For each dataset we use 20 points for training and the rest for testing. Figure 6.2 compares 4-NN classification error incurred by our

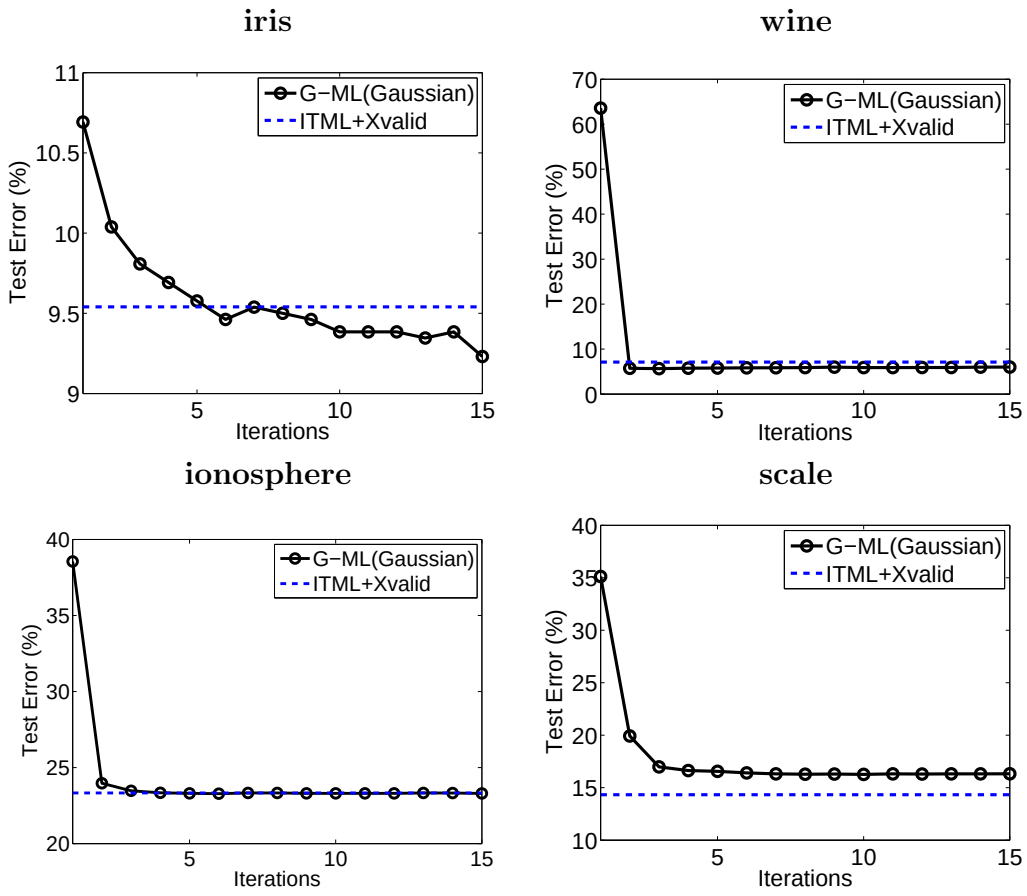


Figure 6.2: 4-NN classification error via kernels learned using our method (G-ML) and ITML [24]. The data-dependent kernel M is the RBF kernel. Clearly, G-ML is able to achieve competitive error rate while learning the kernel width for M, while ITML requires cross validation.

method to the ITML method [24]. For ITML, the kernel width of Gaussian RBF M is selected using leave-one-out cross validation. Clearly, G-ML is able to automatically select a good kernel width, while ITML requires slower cross

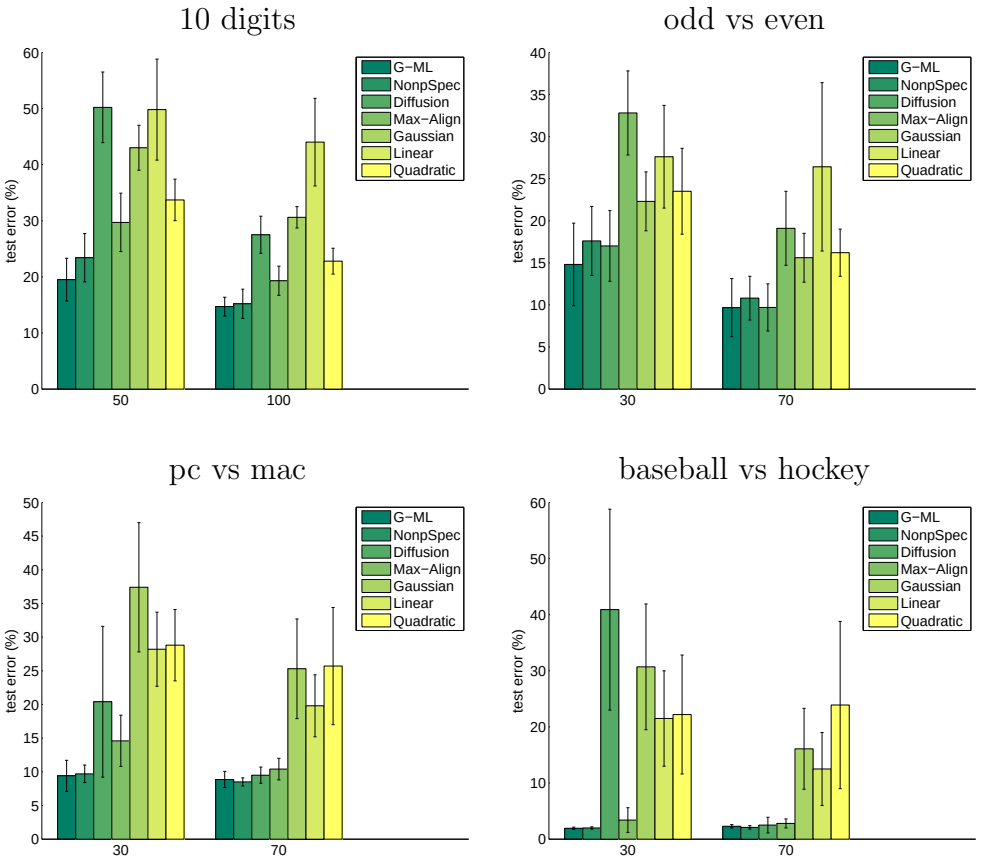


Figure 6.3: Classification error for various methods on four standard datasets using different number of labeled samples. Note that G-ML consistently performs comparably or better than the best semi-supervised learning methods and significantly outperforms the supervised learning methods.

validation to obtain a similar width parameter.

6.5.2 Classification: Semi-supervised Learning

Next, we evaluate our method for classification in the semi-supervised setting (Section 6.4.1). We evaluate our method on four datasets that fall

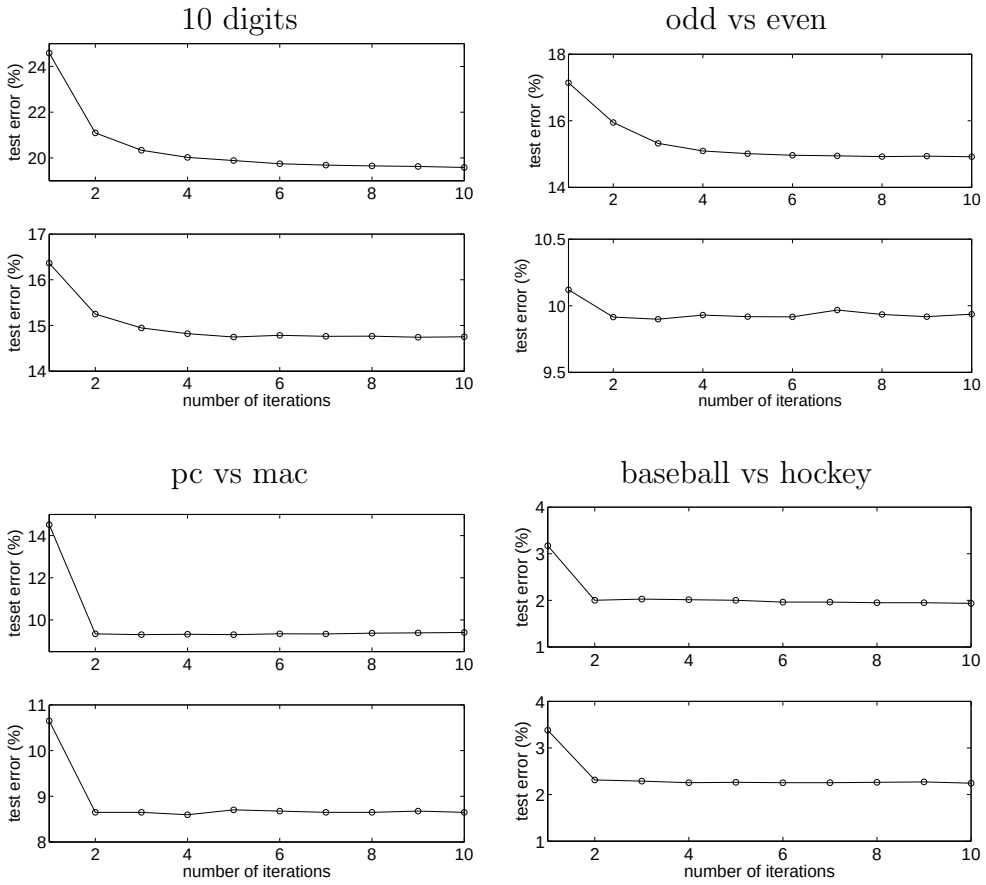


Figure 6.4: **Top Row** and **Bottom row**: Classification error rate for our method (G-ML) with 30 labeled samples and 70 labeled data (50, 100 for 10 digits) as the number of iterations increase. In both the cases G-ML improves over the initial (diffusion) kernel .

in two broad categories: a) text classification: two standard subsets of 20-newsgroup dataset, namely, baseball-hockey (1993 instances/ 2 classes), and pc-mac (1943/2). b) digit classification: two subsets of USPS digits dataset, odd-even (4000/2) and ten digits (4000/10). Odd-even involves classifying odd "1, 3, 5, 7, 9" vs. even "0, 2, 4, 6, 8" digits, while ten digits is the standard

10-way digit classification.

To form the k-NN graph, we use cosine similarity over tf-idf representation for text classification datasets and RBF-kernel function over gray-scale pixel values for the digits dataset. We compare G-ML (k-NN classifier with $k = 4$) with three state-of-the-art semi-supervised kernels: non-parametric spectral kernel [112], diffusion kernel [64], and maximal-alignment kernel [70]. For all four semi-supervised learning models we use 10-NN unweighted graphs on all the datasets. The non-parametric spectral kernel uses the first 200 eigenvectors [112], whereras G-ML uses the first 20 eigenvectors to form M. For the three competitor semi-supervised kernels, we use support vector machines (one-vs-all classification). We also compare against three standard kernels: RBF kernel (bandwidth learned using 5-fold cross validation), linear kernel, and quadratic kernel. We use the diffusion kernel $K = \exp(-tL)$ with $t = 0.1$ for initializing our alternating minimization algorithm. Note that the various parameter values are set arbitrarily without optimizing and do not give an unfair advantage to the proposed method.

We report the classification error of G-ML averaged over 30 random training/testing splits; the results of competing methods are from [112]. Figure 6.3 compares error incurred by various methods on each of the four datasets, the second row shows the test error rate at each iteration of G-ML using 30 labeled examples (except for 10 digits dataset where we use 50 examples), while the third row shows the same for 70 labeled examples (100 examples for 10 digits). Clearly, on all the four data sets, G-ML gives comparable or

better performance than state-of-the-art semi-supervised learning algorithms and significantly outperforms the supervised learning algorithms.

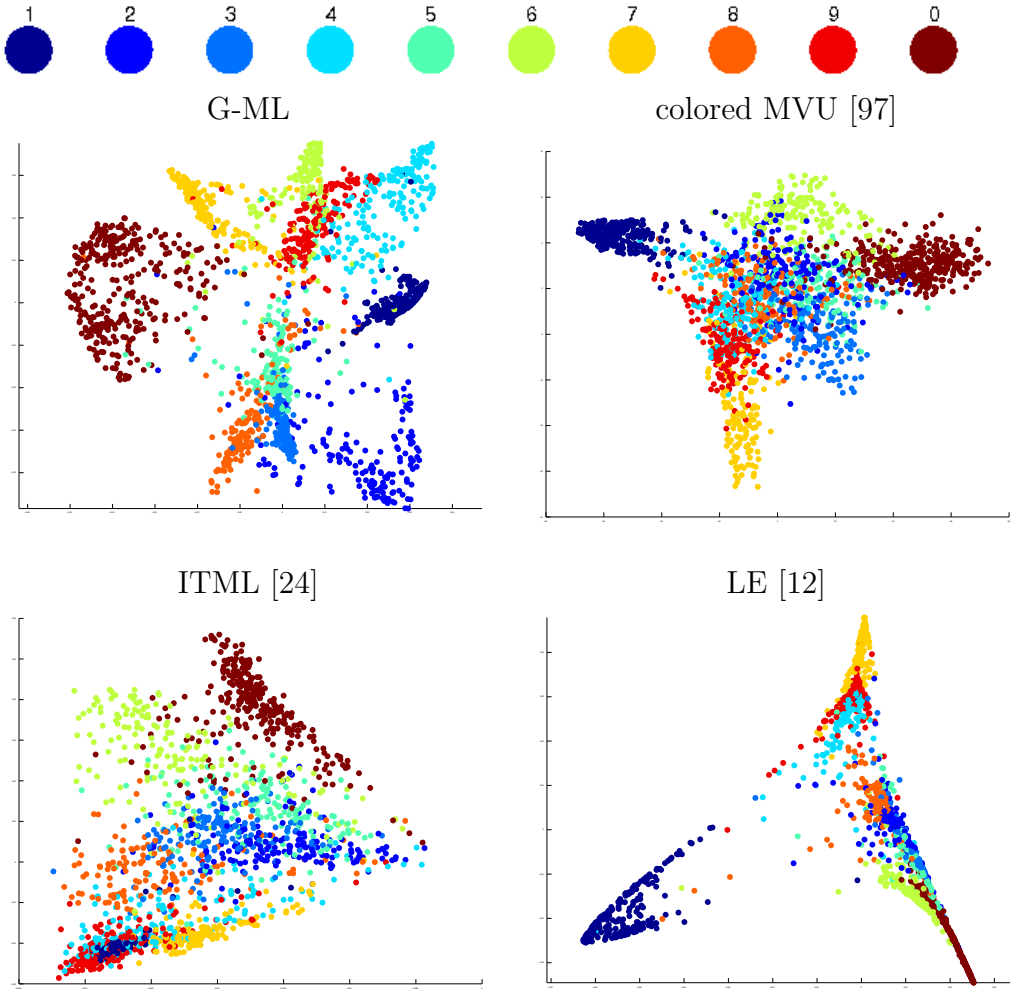


Figure 6.5: Two dimensional embedding of 2007 USPS digits using different methods. Color of the dots represents different classes of digits (color coding is provided in the top row). We observe that compared to other methods, our method separates the respective manifolds of different digits more accurately, *e.g.* digit 4. (Better viewed in color)

6.5.3 Colored Dimensionality Reduction

Next, we apply our method to the task of semi-supervised non-linear dimensionality reduction. We evaluate our method on standard USPS digits dataset, and compare it to the state-of-the-art colored Maximum Variance Unfolding (colored MVU) [97] method which also performs dimensionality reduction for labelled data. We also compare our method to ITML [24] that does not take the local geometry into account and Laplacian Eigenmaps [12] that does not exploit the label information. For visualization, we reduce the dimensionality of the data to two and plot each of the 10 classes of digits with different color (Figure 6.5). For the proposed G-ML method, we use 200 samples to generate the pairwise constraints, while colored MVU is supplied with all the labels. Note that other than digit 5, G-ML is able to separate manifolds of all the digits in the two-dimensional embedding. In contrast, colored MVU is unable to clearly separate manifolds of digits 4, 5, 8, and 2 while using more labels than the proposed G-ML method.

6.5.4 Manifold Alignment

In this experiment, we evaluate our method for the task of manifold alignment (Section 6.4.2) on two datasets, each associated with a different type of transformation. The first dataset consists of images of two subjects sampled from the Yale face B dataset, each with 64 different illumination conditions (varying angles of two illumination sources). Note that the images of each of

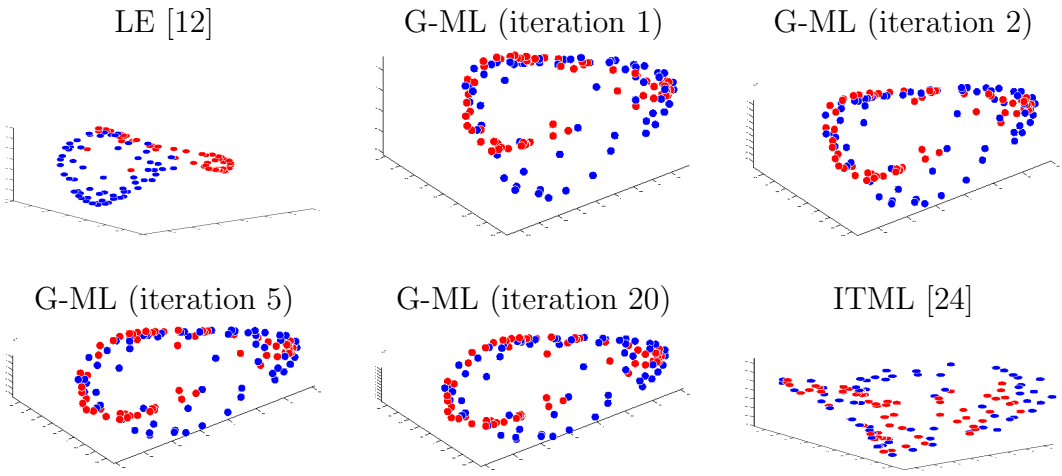


Figure 6.6: Manifold alignment results for the Yale Face dataset. The plot show 3-dimensional embedding of the images of two subjects with different illumination obtained by various methods.

the subjects lie on an arbitrary oriented two-dimensional manifold. In order to align the two manifolds, we randomly sample 10 must-links for the images with the same illumination conditions. Figure 6.6 shows three-dimensional embedding of the images using Laplacian Eigenmaps [12], proposed G-ML method at various iterations, and ITML method with RBF kernel as the baseline kernel [24]. We observe that G-ML is able to capture the manifold structure of the Lie group and successfully align them within five iterations. Next, we apply our method to the task of illumination estimation, where the goal is to retrieve the image with the most similar illumination to the given query image. Figure 6.7 shows that G-ML is able to accurately retrieve similar illumination images irrespective of the identity of the person. The ITML method, which does not capture the local geometry of the unsupervised data, is unable to

Figure 6.7: Retrieval result for two queries based on kernel learned using G-ML and using ITML kernel. We observe that G-ML is able to capture the local geometry of the manifold, which is further confirmed by the illumination retrieval results, where unlike ITML, G-ML is able to retrieve similar illumination images irrespective of the subject. (Better viewed in color)

align the data points w.r.t. the illumination transform and hence unable to accurately retrieve similar illumination images.

To give a quantitative evaluation of manifold alignment, we also performed a similar experiment on a subset of COIL-20 data datasets, which contains images of three subjects with different degree of rotation (72 points uniformly sampled from 0$\sim$360 degree). Images of each subjects should lie on a circular one-dimensional manifold. We apply our method to retrieve images

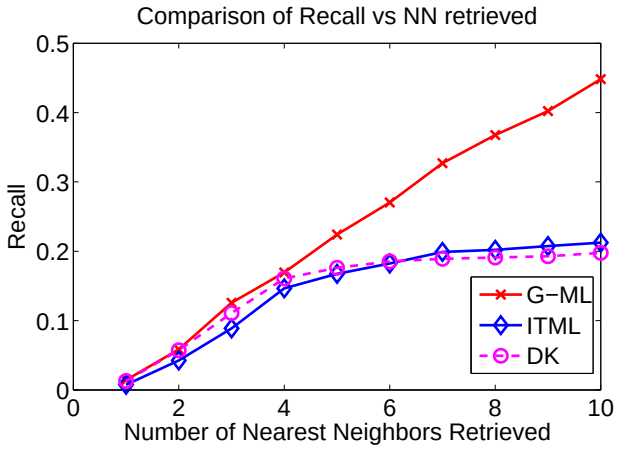


Figure 6.8: This plot shows recall as a function of number of retrieved images, for G-ML, ITML, and Diffusion Kernel (DK).

with similar "angle" to a given query image. Figure 6.8 shows that with 10 randomly chosen similarity-constraints, our method is able to obtain recall of 0.47, significantly outperforming the ITML (0.24) and the diffusion kernel [64] method (0.23).

Chapter 7

Low Rank Kernel Learning

In this chapter, we focus on the problem of kernel learning with limited supervision. In contrast to the previous chapter, we assume that the data lies in a low-dimensional *linear* subspace, and the goal is to learn the corresponding low-rank kernel matrix/function[1] over the data.

We consider two different settings for kernel learning:

- **Transductive Learning**: In this setting, all the training and test data is provided upfront. Hence, learning a low-rank kernel **matrix** suffices as kernel values between out-of-sample points need not be computed.
- **Inductive Learning**: For inductive learning, a kernel **function** needs to be learned as the kernel function value between any pair of points might be needed.

In both the cases, we formulate the low-rank kernel learning problem as a rank minimization problem subject to several affine constraints and a

[1]Throughout this chapter, the term *rank k-kernel function* refers to a kernel function which leads to at most rank k-kernel matrix for any set of points

heavily structured convex set. The problem of rank minimization over polyhedral sets is an important problem in itself, with numerous applications in machine learning, computer vision, control theory etc. Apart from low-rank kernel learning, other important machine learning problems like feature efficient linear classification, semi-definite embedding (SDE), non-negative matrix approximation (NNMA), etc., can also be viewed as rank minimization problems over a polyhedron with additional convex constraints such as a Frobenius norm constraint and/or a semi-definiteness constraint. Even though there has been extensive work on the specific problems mentioned above, the general problem of rank minimization over polyhedral sets is not well understood.

In this chapter, we first address the problem of rank minimization when there are a large number of trace constraints along with a few convex constraints that are relatively "easy" in a precise sense defined below. We further study specific algorithms for low-rank kernel learning in both transductive and inductive settings, which are special cases of the general rank minimization problem.

We first formulate the rank minimization problem we study. Let $A_1, \ldots, A_m \in \mathbb{R}^{n\times n}$, $b_1, \ldots, b_m \in \mathbb{R}$ and let $\mathcal{C} \subseteq \mathbb{R}^{n\times n}$ be a convex set of matrices. Then, consider the following optimization problem which we refer to as RMP (for

Rank Minimization over Polyhedron):

$$\begin{aligned} \min \quad & \operatorname{rank}(X) \\ \text{s.t} \quad & \operatorname{Tr}(A_i X) \geq b_i \quad 1 \leq i \leq m \\ & X \in \mathcal{C}. \end{aligned} \tag{RMP}$$

The set $\mathcal{C}$ represents the "easy" constraints in the sense that for such a set $\mathcal{C}$, we assume that RMP with a single trace constraint can be solved efficiently. Note that this holds for many typical convex sets $\mathcal{C}$, e.g., the unit ball under any L_p or Frobenius norm, the semi-definite cone, and the intersection of the unit ball with the semi-definite cone. Furthermore, low-rank kernel learning, SDE and NNMA can all be seen as instantiations of the above general formulation.

The general RMP problem as stated above is non-convex, NP-hard and, as we show, cannot be approximated unless $P = NP$. Due to the computational hardness of the problem much of the previous work has concentrated on providing heuristics, with no guarantees on the quality of the solution. We remark that the recent trace-norm based approach of [85] does guarantee an optimal solution for a simplified instance of RMP where only well-conditioned linear equality constraints are allowed. However, it is not clear how to extend their guarantees to the more general RMP problem.

We now list the main contributions of this chapter:

- We show that for the RMP problem, the minimum feasible rank cannot be approximated unless $P = NP$ (see Theorem 9). To get over this hurdle we introduce a relaxed notion of approximation, where along with approximating the optimal rank we also allow small violations in the constraints. In practice, this relaxed notion is as meaningful as the standard notion of approximation as almost all real-life problems have noisy measurements.
- We provide an algorithm for RMP based on the Multiplicative Weights Update framework of [84, 4] and under the relaxed notion of approximation, we prove approximation guarantees for the algorithm.
- We provide an algorithm for RMP based on the framework of online convex programming (OCP) introduced by [113]. We use the OCP framework in a novel way by changing the role of the decision maker, which searches over the constraints instead of the feasible points, as is usually the case. We prove that under the relaxed notion of approximation, the algorithm provides approximation guarantees.
- We apply our method to the problem of low-rank kernel matrix learning in transductive setting which can be seen as a specific instance of general RMP. Furthermore, we demonstrate empirically that our algorithm for low-rank kernel learning is able to learn low-rank kernels while improving on the accuracy of the baseline kernel.
- Using our kernel learning framework introduced in Section 3.3, we reduce the problem of kernel function learning to an instance of RMP and show

that using both of our algorithms for RMP can be used efficiently in this context.

Most of the material of this chapter is based on our work [78, 56].

7.1 Computational Complexity

As was mentioned in the introduction, RMP is NP-hard in general. Further, by a reduction to the problem of support minimization over convex sets, and using hardness of approximation results from [2] we show that RMP is hard to approximate within a factor of $2^{\log^{1-\epsilon} n}$, for every $\epsilon > 0$. Further, if we let $\Delta = \max\{\|A_i\|_F + |b_i| : 1 \leq i \leq m\}$ for a particular instance of RMP, then following the techniques of [2] we also show that RMP is hard to approximate within a factor of $2^{\log^{1-\epsilon} \Delta}$, for every $\epsilon > 0$.

Theorem 9. *There exists no polynomial time algorithm for approximating* RMP *within a factor of* $2^{\log^{1-\epsilon} n}$*, for every* $\epsilon > 0$ *unless* $P = NP$*. Further, if we let* $\Delta = \max\{\|A_i\|_F + |b_i| : 1 \leq i \leq m\}$ *then* RMP *cannot be approximated within a factor of* $2^{\log^{1-\epsilon} \Delta}$ *unless* $P = NP$[2]*.*

In view of the above hardness result we introduce a weaker notion of approximation. We believe the relaxed notion of approximation to be of equal use, if not more, as the standard notion of approximation in practice. For

[2]This hardness result holds even when $\mathcal{C}$ is fixed to be the unit ball under an L_p or Frobenius norm or many other sets common in practice.

an instance of RMP, let $\mathbb{F}(A_1, \ldots, A_m, \boldsymbol{b}, \mathcal{C})$ denote the feasible region, where $\boldsymbol{b} = (b_1, \ldots, b_m)$:

$$\mathbb{F}(A_1, \ldots, A_m, \boldsymbol{b}, \mathcal{C}) = \{X : X \in \mathcal{C}, \mathrm{Tr}(A_i X) \geq b_i, \forall i\}. \tag{7.2}$$

Definition 7.1.1. *Given a function $c : \mathbb{R} \to \mathbb{R}_+$, we say that a matrix $\overline{X}$ is a $(c(\epsilon), \epsilon)$-approximate solution to* RMP *if the following hold:*

$$\overline{X} \in \mathbb{F}(A_1, \ldots, A_m, \boldsymbol{b} - \epsilon\mathbf{1}, \mathcal{C})$$
$$\mathit{rank}(\overline{X}) \leq c(\epsilon) \min\{\mathit{rank}(X) : X \in \mathbb{F}(A_1, .., A_m, \boldsymbol{b}, \mathcal{C})\}.$$

Further, we say that RMP *is $(c(\epsilon), \epsilon)$-approximable, if there exists a polynomial time algorithm that given inputs $A_1, \ldots, A_m, \boldsymbol{b}, \epsilon$, outputs a $(c(\epsilon), \epsilon)$-approximate solution to* RMP.

Thus, along with approximating the minimum feasible rank we also allow a small violation, quantified by ϵ, of the constraints. Note that for $\epsilon = 0$, we recover the normal notion of approximation with an approximation factor of $c(0)$.

7.2 Methodology

Our approach to RMP crucially exploit the fact that even though RMP is hard in general, it is efficiently solvable for certain convex sets $\mathcal{C}$ when there is a single trace constraint. For instance, when $\mathcal{C} = \{X : X \succeq 0, \|X\|_F \leq 1\}$, with a single trace constraint the RMP problem can be solved efficiently using a singular value decomposition of A.

In our approach, we assume the existence of an oracle $\mathcal{O}$ that solves the following RMP problem with a single trace constraint, and returns an optimal X or declares the problem infeasible:

$$\mathcal{O}: \quad \min \ \text{rank}(X) \ \text{ s.t. } \text{Tr}(AX) \geq b, X \in \mathcal{C}. \tag{7.3}$$

As discussed above, for certain convex sets $\mathcal{C}$, oracle $\mathcal{O}$ solves a hard non-convex problem. In our approach, we exploit this fact by making several queries to the oracle where the trace constraint $\text{Tr}(AX) \geq b$ is obtained by a weighted combination of the original trace constraints. The trick then is to choose the combinations in such a way that after a small number of iterations, we can find a low-rank X that satisfies all the constraints with at most an ϵ-violation.

Based on the above intuition, we give two approaches to solve the RMP problem - one based on the Multiplicative Weights Update algorithm and the other based on online convex programming.

Before we describe our algorithms, we need to introduce additional notation. For an instance of RMP specified by matrices $A_1, \ldots, A_m$, scalars $b_1, \ldots, b_m$ and convex set $\mathcal{C}$, let $D = \max\{\|X\|_F : X \in \mathcal{C}\}$. We assume, without loss of generality, that $D \geq 1$. Recall that $\mathbb{F}((A_1, \ldots, A_m), \boldsymbol{b}, \mathcal{C})$ and $\mathbb{F}((A_1, \ldots, A_m), \boldsymbol{b} - \epsilon\mathbf{1}, \mathcal{C})$ denote the feasibility sets as defined in (7.2) and $\Delta = \max\{\|A_i\|_F + |b_i| : 1 \leq i \leq m\}$. Further, let k^* be the rank of the optimal solution to RMP. That is,

$$k^* = \min\{\text{rank}(X) : \ X \in \mathbb{F}((A_1, \ldots, A_m), \boldsymbol{b}, \mathcal{C})\}.$$

7.2.1 Rank Minimization via Multiplicative Weights Update

Algorithm 2 RMP-MW (Multiplicative Updates)

Input: Constraints (A_i, b_i), $1 \leq i \leq m$, ϵ

Input: Oracle $\mathcal{O}(A, b)$ which solves

$$\min \ \text{rank}(X) \quad s.t. \quad \text{Tr}(AX) \geq b, \ X \in \mathcal{C}$$

Initialize: $w_i^1 = 1, \ \forall i$ and $t = 1$
repeat
  Set $(A^t, b^t) = \sum_i w_i^t (A_i, b_i)$
  if Oracle $\mathcal{O}(A^t, b^t)$ declares infeasibility **then**
    return Problem is infeasible
  else
    Obtain X^t using Oracle $\mathcal{O}(A^t, b^t)$
    Set $M(i, X^t) = \text{Tr}(A_i X^t) - b_i$
    Set $\rho = \max_i M(i, X^t)$
    Set $\boldsymbol{w}^{t+1}$=MultUpdate$(\boldsymbol{w}^t, M, \rho, \epsilon)$
  end if
  Set $t = t + 1$
until $t > T$
return $\overline{X} = \sum_t X^t / T$

Output: $\boldsymbol{w}^{t+1}$=MultUpdate$(\boldsymbol{w}^t, M, \rho, \epsilon)$

Set $\delta = \min\{\frac{\epsilon}{4\rho}, \frac{1}{2}\}$
for all $1 \leq i \leq m$ **do**
  if $M(i, X^t) \geq 0$ **then**
    $w_i^{t+1} = w_i^t (1 - \delta)^{M(i, X^t)/\rho}$
  else
    $w_i^{t+1} = w_i^t (1 + \delta)^{-M(i, X^t)/\rho}$
  end if
end for

In this section we present an approach to RMP based on the generalized experts (GE) framework described in Section 2.3. To adapt the GE framework for the RMP problem, we first need to select a set of experts, a set of events

and the associated penalties. We associate each RMP constraint $\mathrm{Tr}(A_i X) \geq b_i$ with an expert and let the events correspond to the elements of $\mathcal{C}$. The penalty for expert i corresponding to the i-th constraint and event X is then given by $\mathrm{Tr}(A_i X) - b_i$. Note that rather than rewarding a satisfied constraint, we penalize it. This strategy is motivated by the work of [84, 3] and is similar to boosting, where a distribution is skewed towards an example for which the current hypothesis made an incorrect prediction.

We assign weights w_i^t to the i-th expert in the t-th iteration, and initialize the weights $w_i^1 = 1$, for all i. In the t-th iteration we query the oracle $\mathcal{O}$ with $(A, b) = \sum_i w_i^t (A_i, b_i)$ to obtain a solution $X^{t+1} \in \mathcal{C}$. We then use the Multiplicative Weights Update algorithm with updates as described in function MultUpdate of Algorithm 2 to compute the weights w_i^{t+1} for the $(t+1)$-th iteration. Algorithm 2 describes our multiplicative update based algorithm for RMP. In the following theorem we prove approximation guarantees for the solution output by Algorithm 2.

Theorem 10. *Given the existence of an oracle $\mathcal{O}$ solving the problem of* (7.3)*, Algorithm 2 outputs an $(O(\frac{\Delta^2 D^2 \log n}{\epsilon^2}), \epsilon)$-approximate solution to* RMP*.*

Proof. Observe that, if the oracle declares infeasibility at any time step t, the original problem is also infeasible. Hence, we assume that the oracle returned a feasible point X^t at time-step t, for all $1 \leq t \leq T$.

Now, $|\mathrm{Tr}(A_i X) - b_i| \leq \|A_i\|_F \|X\|_F + |b_i| \leq \Delta D$. Thus, the penalties $\mathrm{Tr}(A_i X) - b_i$ lie in $[-\Delta D, \Delta D]$. As Algorithm 2 uses multiplicative updates

to update the weights as in Theorem 1, for $T = 16(\Delta D)^2 \log n/\epsilon^2$, we have

$$\frac{\sum_t \sum_j p_j^t [A_j X^t - b_j]}{T} \leq \epsilon + \frac{\sum_t [A_i X^t - b_i]}{T},$$

where $p_j^t = w_j^t / \sum_l w_l^t$. Note that, by the choice of the oracle the $LHS \geq 0$. Thus, for $\overline{X} = \sum_t X^t/T$ we have

$$\mathrm{Tr}(A_i \overline{X}) \geq b_i - \epsilon, \quad \forall i. \tag{7.4}$$

We now bound the rank of $\overline{X}$ compared to the optimal value. Let t be such that X_t has the highest rank, say k, among $X_1, \ldots, X_T$. Then, $k^* \geq k$, as for a particular convex combination of (A_i, b_i) the minimum rank possible was k. Thus, $\mathrm{rank}(\overline{X}) \leq kT = O(\frac{(\Delta^2 \cdot D^2) k^*}{\epsilon^2})$. Using (7.4) we have that $\overline{X} \in \mathbb{F}((A_1, \ldots, A_m), b - \epsilon \mathbf{1}, \mathcal{C})$. Thus, by Definition 7.1.1 $\overline{X}$ is an $(O(\frac{\Delta^2 D^2 \log n}{\epsilon^2}), \epsilon)$-approximate solution to RMP. □

The running time of Algorithm 2 is $O(\frac{\Delta^2 D^2 \log n}{\epsilon^2}(T_{\mathcal{O}} + mn^2))$, where $T_{\mathcal{O}}$ denotes the oracle's running time.

7.2.2 Rank Minimization via OCP

In this section, we present a novel application of online convex programming described in Section 2.3.1 to obtain an approximate solution to RMP. The intuition behind this approach is similar to that of Section 7.2.1; in fact this approach can be viewed as a generalization of the approach of Section 7.2.1.

Algorithm 3 RMP-OCP (Online Convex Programming)

Input: Constraints (A_i, b_i), $1 \leq i \leq m$, ϵ

Input: Oracle $\mathcal{O}(A, b)$ which solves $\min \text{ rank}(X) \quad s.t. \quad \text{Tr}(AX) \geq b, \; X \in \mathcal{C}$

Initialize: $A^1 = \frac{\sum_i A_i}{m}$ and $b^1 = \frac{\sum_i b_i}{m}$, $t = 1$
Set $K = \{\sum_i \lambda_i (A_i, b_i) : \sum_i \lambda_i = 1, \lambda_i \geq 0 \; \forall i\}$
repeat
  if Oracle $O(A^t, b^t)$ declares infeasibility **then**
    return Problem is infeasible
  else
    Obtain X^t using Oracle $\mathcal{O}(A^t, b^t)$
    Define function $f^t(A, b) = \text{Tr}(AX^t) - b$
    Set (A^{t+1}, b^{t+1})=GIGA$((A^t, b^t), f^t(A, b), K, t)$
  end if
  Set $t = t + 1$
until $t > T$
return $\overline{X} = \sum_t X^t / T$

Output: $\boldsymbol{z^{t+1}}$=GIGA$(\boldsymbol{z^t}, f^t(\boldsymbol{z}), K, t)$

Set $\eta_t = \frac{\Delta}{2D\sqrt{t}}$
Set $\boldsymbol{z^{t+1}} = \Pi_K \left(\boldsymbol{z^t} - \eta_t \nabla f^t(\boldsymbol{z^t})\right)$, where Π_K represents the orthogonal projection onto K

In the OCP framework one generally associates the convex set K with a feasible region and the cost functions with penalty functions. In our application of OCP to RMP we flip this view and choose K to be the space of convex combinations of the constraints and associate cost functions with feasible points of RMP. In particular, we set $K \subseteq \mathbb{R}^{n \times n} \times \mathbb{R}$ to be the convex hull of $(A_1, b_1), \ldots, (A_m, b_m)$, i.e.,

$$K = \left\{ \sum_i \lambda_i (A_i, b_i) : \sum_i \lambda_i = 1, \lambda_i \geq 0 \; \forall i \right\}.$$

Given a matrix X, we define a cost function $f_X : K \to \mathbb{R}$ by $f_X(A, b) =$

$\mathrm{Tr}(AX) - b$.

We initialize $A^1 = \sum_i A_i/m$ and $b^1 = \sum_i b_i/m$. Given $(A^t, b^t) \in K$ for the t-th iteration, we query the oracle $\mathcal{O}$ with $(A, b) = (A^t, b^t)$ to obtain a solution $X^t \in \mathcal{C}$. We then set the cost function $f^t(A, b) = f_{X^t}(A, b) = \mathrm{Tr}(AX^t) - b$ and use the OCP algorithm [113] as described in function GIGA of Algorithm 3 to compute (A^{t+1}, b^{t+1}) for the $(t+1)$-st iteration. Algorithm 3 describes our OCP based algorithm for RMP. In the following theorem we prove approximation guarantees for the output of Algorithm 3.

Theorem 11. *Given the existence of an oracle $\mathcal{O}$ to solve the problem* (7.3)*, Algorithm 3 outputs an $(O(\frac{\Delta^2 D^2}{\epsilon^2}), \epsilon)$-approximate solution to* RMP*.*

Proof. As in Theorem 10 we assume that the oracle returns a feasible point at all time steps. Note that using the terminology of Theorem 2, $G = \max_{z \in K, t \in \{1,\ldots\}} \| \bigtriangledown f^t(z) \| \leq D$ and $\|K\| \leq \Delta$. Thus, using Theorem 2 we have

$$\sum_{t=1}^{T} (\mathrm{Tr}(A^t X^t) - b^t) \leq \min_{(A,b) \in K} \sum_{t=1}^{T} (\mathrm{Tr}(AX^t) - b) + \Delta D \sqrt{T}.$$

Note that the above LHS ≥ 0 since oracle returns a feasible X^t, $\forall t$. Thus, for $T = \Delta^2 D^2/\epsilon^2$ and $\overline{X} = \sum_t X^t/T$,

$$\mathrm{Tr}(A\overline{X}) \geq b - \epsilon, \tag{7.5}$$

for all $(A, b) \in K$. In particular, we have for every i, $\mathrm{Tr}(A_i \overline{X}) \geq b_i - \epsilon$. We now bound the rank of $\overline{X}$ compared to the optimal value. Let t be such that X_t has the highest rank, say k, among $X_1, \ldots, X_T$. Then, we must have

$k^* \geq k$, and so we have $\text{rank}(\overline{X}) \leq kT \leq O((\Delta D)^2 k^*/\epsilon^2)$. Also, from (7.5) we have that $\overline{X} \in \mathbb{F}((A_1, \ldots, A_m), b - \epsilon\mathbf{1}, \mathcal{C})$. Thus by Definition 7.1.1, $\overline{X}$ is a $(O((\Delta^2 D^2)/\epsilon^2), \epsilon)$-approximate solution to RMP. □

The running time of Algorithm 3 is $O(\frac{\Delta^2 D^2}{\epsilon^2}(T_{\mathcal{O}} + T_{OCP} + mn^2))$, where $T_{\mathcal{O}}$ denotes the running time of the oracle, and T_{OCP} denotes the time taken in each round by the GIGA algorithm of Theorem 2.

7.2.3 Discussion

Oracle: The oracle for solving the problem of (7.3) plays a crucial role in our approach. As discussed previously, for typical cases of $\mathcal{C}$, like the unit ball under an L_p or Frobenius norm etc., (7.3) can be solved by the singular value decomposition of A. Further, in the case when the set $\mathcal{C}$ involves a quadratic or ellipsoid constraint we can use the S-procedure [86] to solve (7.3).

Limitations: A drawback of our methods is the dependence on Δ, ϵ in the bound of Theorem 10. This limits the applicability of our method to problems, like NNMA, with a large number of non-negativity constraints where the ratio $\frac{\Delta}{\epsilon}$ is typically large. However, our algorithms can be used as a heuristic for such problems and can be used to initialize other methods which require a good low-rank solution for initialization. Moreover, the lower bounds for the experts framework and boosting suggest that the dependence on Δ, ϵ in our bound may be optimal for the general RMP problem.

7.3 Low-rank Kernel Learning

In this section we apply both our rank minimization algorithms to the problem of low-rank kernel learning in both transductive and inductive setting.

7.3.1 Low-rank Kernel Matrix Learning: Transductive Setting

In the transductive setting, low-rank kernel learning involves finding a low-rank positive semi-definite (p.s.d.) matrix that satisfies linear constraints typically derived from labeled data. Due to the rank constraint, this problem is non-convex and is in general hard to solve. As described below, both our online learning approaches can be applied naturally to this problem. We provide provable guarantees on the rank of the obtained kernel.

Formally, the low-rank kernel learning problem can be cast as the following optimization problem:

$$\begin{aligned} \min_{K} \quad & \|K - K_0\|_F \\ \text{s.t.} \quad & \mathrm{Tr}(S_i K) \leq \ell, \quad \forall\ 1 \leq i \leq |\mathcal{S}|, \\ & \mathrm{Tr}(D_j K) \geq u, \quad \forall\ 1 \leq j \leq |\mathcal{D}|, \\ & \mathrm{rank}(K) \leq r, \quad K \succeq 0, \end{aligned} \tag{7.6}$$

where $\mathcal{S}$ is a set of pairs of points from the same class that are constrained to have distance less than ℓ. Similarly, $\mathcal{D}$ is a set of pairs of points from different classes that are constrained to have distance greater than u, with $\ell \ll u$. For a similarity constraint matrix S_i, $S_i(i_1, i_1) = S_i(i_2, i_2) = 1$, $S_i(i_1, i_2) = S_i(i_2, i_1) = -1$ and all other entries 0. The dissimilarity constraint matrices D_j

can be constructed similarly. Assuming $\|K_0\|_F = 1$, (7.6) can be reformulated as:

$$\begin{aligned} \min_{K} \quad & \text{rank}(K) \\ \text{s.t.} \quad & \text{Tr}(S_i K) \leq \ell \ \ \forall i, \quad \text{Tr}(D_j K) \geq u \ \ \forall j, \\ & \text{Tr}(K K_0) \geq \beta, \quad \|K\|_F \leq 1, \quad K \succeq 0, \end{aligned} \tag{7.7}$$

where β is a function of r and can be computed using binary search. Note that (7.7) is a special case of RMP with the convex set $\mathcal{C}$ being the intersection of the p.s.d. cone and the unit Frobenius ball. Hence, we can use RMP-MW and RMP-OCP to solve (7.7). Given (A, b) the oracle for both the methods solves:

$$\min_{K} \ \text{rank}(K) : \ \text{Tr}(AK) \geq b, \|K\|_F \leq 1, \ K \succeq 0. \tag{7.8}$$

Let $A = U\Sigma U^T$ be the eigenvalue decomposition of A, and let Λ be a diagonal matrix with just the positive entries of Σ. Then the minimum k s.t. $\sqrt{\sum_{i=1}^{k} \Lambda(i,i)^2} \geq b$ is the solution to (7.8). This follows from elementary linear algebra. Note that for the oracle solving (7.8), $T_{\mathcal{O}} = O(n^3)$.

Now, $D = 1$ and $\Delta = O(1 + l^2 + u^2)$ as $\|S_i\|_F = \|D_j\|_F = 2$. Using Theorem 10, the RMP-MW algorithm obtains a solution with rank $r \leq O(\frac{1+u^2+l^2}{\epsilon^2} \log n) r^*$ where r^* is the optimal rank. Similarly, RMP-OCP obtains an $\left(O(\frac{1+u^2+l^2}{\epsilon^2}), \epsilon\right)$-approximate solution. In Section 7.4.2, we present empirical results for RMP-MW and RMP-OCP algorithms on some standard UCI datasets.

7.3.2 Low-rank Kernel Function Learning: Inductive Setting

Consider an instance of the metric learning framework (3.2) introduced in Section 3.1, with $\text{rank}(W) + \|W\|_F^2$ as the regularization function and linear inequality constraints g_i. Formally,

$$\begin{aligned} \min_{W} \quad & \text{rank}(W) \\ \text{s.t.} \quad & \text{Tr}(WXC_iX^T) \leq \boldsymbol{b}_i, \quad \forall 1 \leq i \leq m, \\ & \|W\|_F^2 \leq r, \\ & W \succeq 0, \end{aligned} \tag{7.9}$$

where $W \in \mathbb{S}_+^d$ is a positive semi-definite matrix to be learned, $C_i \in \mathbb{R}^{n \times n}, 1 \leq i \leq m$ are constraint matrices, $X = \{\phi(\boldsymbol{x}_1), \phi(\boldsymbol{x}_2,) \ldots, \phi(\boldsymbol{x}_n)\}$, $\phi(\boldsymbol{x}_i)$ is the feature vector representation for i-th data point. Note that if the constraint matrices C_i are of the form $\pm(\boldsymbol{e}_p - \boldsymbol{e}_q)(\boldsymbol{e}_p - \boldsymbol{e}_q)^T$, then the above problem is a metric learning problem using similarity and dissimilarity constraints.

Now, we show that as in Section 3.3, we can extend the above formulation for learning a low-rank kernel function. First, we provide two lemmas that we use to reduce (7.9) to a kernel learning problem.

Lemma 10. *Let W^* be the optimal solution of* (7.9)*, then*

$$W^* = XS^*X^T,$$

where $S^* \in \mathbb{S}_+^n$.

Proof. Let $W = U\Lambda U^T = \sum_j \lambda_j \boldsymbol{u}_j \boldsymbol{u}_j^T$ be the eigenvalue decomposition of W. Consider a linear constraint $\text{Tr}(WXC_iX^T) \leq b_i$ as specified in problem (7.9). Note that $\text{Tr}(WXC_iX^T) = \sum_j \lambda_j \boldsymbol{u}_j^T XC_iX^T \boldsymbol{u}_j$. Note that if the j-th eigenvector $\boldsymbol{u}_j$ of W is orthogonal to the range-space of X, i.e. $X^T\boldsymbol{u}_j = 0$, then the corresponding eigenvalue λ_j is not affected by the constraint. Now, clearly setting $\lambda_j = 0$ decreases rank of W and doesn't effect constraints $\|W\|_F^2 \leq r$ and $W \succeq 0$. Furthermore, the range-space of X is at most n-dimensional. Thus, without loss of generality we can assume that $\lambda_j = 0, \forall j > n$. Furthermore, $\boldsymbol{u}_j \forall j \leq n$ lie in the range-space of X, i.e., $\boldsymbol{u}_j = X\boldsymbol{\alpha}_j \forall j \leq n$ for some $\boldsymbol{\alpha}_j \in \mathbb{R}^n$. Hence,

$$\begin{aligned} W^* &= \sum_{j=1}^{n} \lambda_i^* \boldsymbol{u}_j^* \boldsymbol{u}_j^{*T}, \\ &= \sum_{j=1}^{n} X(\lambda_j^* \boldsymbol{\alpha}_j^* \boldsymbol{\alpha}_j^{*T}) X^T, \\ &= XS^*X^T, \end{aligned}$$

where $S^* = \sum_{j=1}^{n} \lambda_j^* \boldsymbol{\alpha}_j^* \boldsymbol{\alpha}_j^{*T}$. □

Lemma 11. *If $n < d$ and X has full column rank, i.e. X^TX is invertible then:*

$$XSX^T \succeq 0 \iff S \succeq 0.$$

Proof. $\implies$

$XSX^T \succeq 0 \implies \boldsymbol{v}^T XSX^T \boldsymbol{v} \geq 0, \forall \boldsymbol{v} \in \mathbb{R}^d$. Since X has full column rank,

$\forall \boldsymbol{q} \in \mathbb{R}^n \; \exists \boldsymbol{v} \in \mathbb{R}^d$ s.t. $X^T\boldsymbol{v} = \boldsymbol{q}$. Hence, $\boldsymbol{q}^T S \boldsymbol{q} = \boldsymbol{v}^T X S X^T \boldsymbol{v} \geq 0, \forall \boldsymbol{q} \in \mathbb{R}^n \implies S \succeq 0$

$\Longleftarrow$

Now $\forall \boldsymbol{v} \in \mathbb{R}^d$, $\boldsymbol{v}^T X S X^T \boldsymbol{v} \geq 0$ as $S \succeq 0$. Thus $XSX^T \succeq 0$. $\square$

Using Lemma 10, problem (7.9) reduces to:

$$\begin{aligned} \min_{S} \quad & \text{rank}(XSX^T) \\ \text{s.t.} \quad & \text{Tr}(XSX^T X C_i X^T) \leq \boldsymbol{b}_i, \quad \forall 1 \leq i \leq m, \\ & \|XSX^T\|_F^2 \leq r, \\ & XSX^T \succeq 0. \end{aligned} \tag{7.10}$$

Now, assuming X to be full-rank and dimensionality of X to be greater than n (number of points), $\text{rank}(XSX^T) = \text{rank}(XK_0^{-1/2} L K_0^{-1/2} X^T) = \text{rank}(L)$, where $L = K_0^{1/2} S K_0^{1/2}$. Additionally, $\text{Tr}(XSX^T X C_i X^T) = \text{Tr}(K_0^{1/2} L K_0^{1/2} C_i)$, where $K_0 = X^T X$ is the input kernel matrix. Similarly, $\|XSX^T\|_F^2 = \|L\|_F^2$. Using Lemma 11, $XSX^T \succeq 0 \iff L \succeq 0$. Hence, (7.10) reduces to following kernel learning problem:

$$\begin{aligned} \min_{L} \quad & \text{rank}(L) \\ \text{s.t.} \quad & \text{Tr}(K_0^{1/2} L K_0^{1/2} C_i) \leq \boldsymbol{b}_i, \quad \forall 1 \leq i \leq m, \\ & \|L\|_F^2 \leq r, \\ & L \succeq 0. \end{aligned} \tag{7.11}$$

Note that the above problem is the same as the low-rank kernel *matrix* learning problem (7.7) defined in the previous section and is an instantiation

of general RMP. Hence, we can use RMP-MW and RMP-OCP to solve (7.11). The Oracle required by both the methods can be constructed as described in Section 7.3.1. Furthermore, given learned L^*, the learned kernel function, $k(\cdot, \cdot)$, between a pair of points $\boldsymbol{y}_i$ and $\boldsymbol{y}_j$ is given by:

$$K(\boldsymbol{y}_i, \boldsymbol{y}_j) = \phi(\boldsymbol{y}_i)^T X K_0^{-1/2} L K_0^{-1/2} X^T \phi(\boldsymbol{y}_j),$$

i.e., the learned kernel can be computed efficiently using an initial kernel function $K_0(\boldsymbol{x}, \boldsymbol{y}) = \phi(\boldsymbol{x})^T \phi(\boldsymbol{y})$.

7.4 Experimental Results

In this section, we present empirical evaluation and comparison of our algorithms to existing methods for general RMP as well as low-rank kernel learning. For general RMP, we use synthetic examples to compare our methods against the trace-norm heuristic [85] and the log-det heuristic [27]. The trace-norm heuristic relaxes the rank objective to the trace-norm of the matrix, which is given by the sum of its singular values. Note that the trace-norm of a matrix is a convex function. The log-det heuristic relaxes the rank objective to the log of the determinant of the matrix. For the application of RMP to low-rank kernel learning, we use standard UCI datasets. All the presented results represent the average over 20 runs.

7.4.1 Synthetic Datasets

First we use synthetic datasets by generating random matrices $A_i \in \mathbb{S}_n$, where $\mathbb{S}_n$ is the set of $n \times n$ symmetric matrices. We also generate a random

positive semi-definite matrix $X_0 \in \mathbb{S}_n$ with $\|X_0\|_F \leq 1$, and use the obtained X_0 to generate constraints $\mathrm{Tr}(A_i X) \geq b_i = \mathrm{Tr}(A_i X_0)$. The convex set $\mathcal{C}$ is fixed to be the intersection of the p.s.d cone and the unit ball under the Frobenius norm. We fix the number of constraints to be 200 and the tolerance ϵ for RMP-MW and RMP-OCP to be 5%. We use SeDuMi to implement the trace-norm and log-det heuristics.

In Table 7.1, we compare the ranks of the solutions obtained by our algorithms against the ones obtained by the trace-norm and log-det heuristics. For small n, both trace-norm and log-det heuristic perform better than RMP-MW and RMP-OCP. Note that since the constraint matrices A_i are random, they satisfy (with high probability) the restricted isometry property used in the analysis of [85]. However, RMP-OCP outperforms trace-norm heuristic for large n (Table 7.1, $n = 100$) and RMP-MW performs comparably. We attribute this phenomenon to the Frobenius norm constraint for which the theoretical guarantees of [85] are not applicable. Also, both trace-norm and log-det heuristic scale poorly with the problem size and fail to obtain a result in reasonable time even for moderately large n. In contrast, both our algorithms scale well with n, with RMP-MW in particular able to solve problems of sizes up to $n = 5000$.

7.4.2 Low-rank Kernel Learning: Transductive Setting

We evaluate the performance of our methods applied to the problem of low-rank kernel learning in transductive setting, as described in Section

Method\n	50	75	100	200	300
RMP-MW	23.25	11.25	7.3	2	2
RMP-OCP	12.8	7.5	5.3	2	2
Trace-norm	6.8	6.7	6.5	-	-
LogDet	5	4.2	4.0	-	-

Table 7.1: Rank of the matrices obtained by different RMP methods for varying size of the constraint matrices (n). The number of constraints generated (m) is fixed to be 200. A "-" represents that the method could not find a solution within 3 hours on a 2.6GHz Pentium 4 machine. Note that for large problem sizes, both the trace-norm and the log-det heuristics are not computationally viable. Both our approaches outperform the trace-norm heuristic as the problem size increases.

7.3.1, for k-NN classification on standard UCI datasets. We use two-fold cross validation with $k = 5$. The lower and upper bounds for the similarity and dissimilarity constraints (l, u) are set using the 30-th and 70-th percentiles of the observed distribution of distances between pairs of points. We randomly select a set of $40c^2$ pairs of points for constraints, where c is the number of classes in the dataset. We run both RMP-MW and RMP-OCP for $T = 50$ iterations. Empirically our algorithms significantly outperform the theoretical rank guarantees of Theorems (10) and (11).

Table 7.2 shows the accuracies achieved by the baseline Gaussian kernel (with $\sigma = 0.1$), RMP-MW, RMP-OCP and the Burg divergence (also called as LogDet divergence) based low-rank kernel learning algorithm (BurgKernel) of [68]. It can be seen from the table that both RMP-MW and RMP-OCP obtain a significantly lower rank kernel than the baseline Gaussian kernel. Further, RMP-MW and RMP-OCP achieve a substantially higher accuracy than the

Dataset\Method	GK	MW	OCP	BK
Musk	80.80 (476)	93.11 (44.1)	**98.15** (61.2)	81.51 (61.2)
Heart	77.44 (267)	91.05 (46.8)	**91.13** (39.5)	83.91 (39.5)
Ionosphere	90.34 (350)	**91.26** (40)	91.17 (27.9)	90.67 (27.9)
Cancer	90.12 (569)	93.14 (82)	91.46 (94)	**93.38** (94)
Scale	66.34 (607)	**73.78** (146)	72.46 (91)	72.11 (91)

Table 7.2: Accuracies for 5-Nearest Neighbor classification using kernels obtained by different methods. Numbers in parentheses represent the rank of the obtained solution. GK represents Gaussian Kernel ($\sigma = 0.1$), MW represents RMP-MW, OCP represents RMP-OCP and BK represents BurgKernel[68]. Overall, RMP-OCP obtains the best accuracy.

Gaussian kernel. Our algorithms also achieve a substantial improvement in accuracy over the BurgKernel method. Note that we iterate our algorithms for fewer iterations compared to the ones suggested by the theoretical bounds, hence few of the constraints maybe unsatisfied. This suggests that these unsatisfied constraints maybe noisy constraints and have small effect on the generalization error. We leave further investigation into generalization error of our methods as a topic for future research.

Note that the BurgKernel method needs to be initialized with a low-rank kernel. Typically, a few top eigenvectors of the baseline kernel are used for this initialization. However, selecting only a few top eigenvectors can lead to a poor initial kernel, especially if the rank of the initial kernel is high. This can further lead to poor accuracy for the BurgKernel method, as indicated

by our experiments. Instead, the kernels obtained by our algorithms could be used to *initialize* the BurgKernel algorithm. For example, for the case of the Heart dataset, initialization of BurgKernel algorithm with the low-rank solution obtained by RMP-OCP method achieves an accuracy of 94.29 compared to 83.91 achieved when initialized with the top eigenvectors of the baseline Gaussian kernel. Note that this also improves upon the accuracy achieved by RMP-MW and RMP-OCP.

7.5 Summary

In this chapter, we considered the problem of low-rank kernel learning in both the transductive and inductive settings. For both the settings, low-rank kernel learning reduces to the rank minimization problem over the intersection of a polyhedra and a heavily structured convex set. For this problem, we proposed two online learning based approximation algorithms — the multiplicative weights update based algorithm and the online convex programming based algorithm. Both the algorithms can be applied to a large class of general rank minimization problems and provide provable approximation guarantees.

Chapter 8

Unsupervised Distance Learning

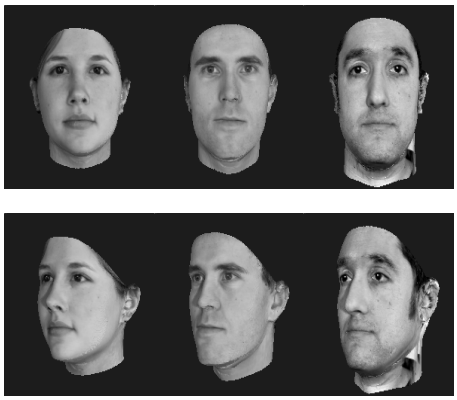


Figure 8.1: Images of different persons in different poses. Each row has different persons in the same pose. Each column has the same person in different poses.

For most real-world problems, the data has complicated semantics and several interpretations. For example, a set of face images can be categorized according to person identity, pose, angle of camera etc. In previous chapters, we studied the problem of learning a metric/kernel function using available side-information. Broadly speaking, the goal of supervision is to stress a particular semantic or interpretation of the data and use it to compress the data, e.g., if the label of an image containing the face of a person is determined

according to the identity of the person in the image, then the data is compressed or clustered according to the person-identity semantic. However, in the absence of any side-information, it is desirable to uncover all the dominant semantics.

In this chapter, we study the problem of distance learning in the unsupervised setting. We model a particular semantic as the clustering [1] it induces over the data. Hence, in the absence of any supervision, the goal is to uncover all the disparate or alternative clusterings. As an example, consider a set of pictures of different persons in different poses (see Figure 8.1). Given such a dataset the goal is to recover two disparate clusterings of the data - one based on the identity of the person and the other based on their pose. The above problem arises naturally for many other widely used datasets, for instance: news articles (can be clustered by the main topic, or by the news source), reviews of various musical albums (can be clustered by composers, or by other characteristics like genre of the album), and movies (can be clustered based on actors/actresses or genre).

In this chapter, we present two novel unsupervised approaches for discovering disparate clusterings in a given dataset. In the first approach we aim to find multiple clusterings of the data which satisfy two criteria: a) the clustering error of each individual clustering is small and b) different clusterings have small *correlation* between them. To this end, we present a new and compu-

[1] Throughout this chapter, a *clustering* will refer to a set of disjoint clusters of the data.

tationally tractable characterization of *correlation* (or decorrelation) between different clusterings. We use this characterization to formulate a k-means type objective function which contains error terms for each individual clustering along with a regularization term corresponding to the correlation between clusterings. We provide a computationally efficient k-means type algorithm for minimizing this objective function.

In the second approach we model the problem of finding disparate clusterings as one of learning the component distributions when the given data is sampled from a convolution of mixture distributions. This formulation is appropriate when the different clusterings come from independent additive components of the data. The problem of learning a convolution of mixture distributions is closely related to factorial learning [34, 50, 87]. However, the methods of [34, 50, 87] are not suited for recovering multiple clusterings. The problem with applying factorial learning directly is that there are multiple solutions to the problem of learning a convolution of mixture distributions. Out of all such possible solutions, the desirable solutions are the ones that give maximally disparate clusterings. To address this problem we propose a regularized factorial learning model that intuitively captures the notion of decorrelation between clusterings and aims to estimate the parameters of the decorrelated model.

An important aspect of both our approaches is the notion of decorrelation between clusterings. The decorrelation measures that we propose quantify the "orthogonality" between the mean vectors corresponding to different

clusterings. We show that the characterization of disparity between different clusterings by the "orthogonality" between the mean vectors of the respective cluster centers has a well-founded theoretical basis (see Section 8.1.3.1).

We evaluate our methods on synthetic and real-world datasets that have multiple disparate clusterings. We consider real-world datasets from two different domains - a music dataset from the text-mining domain and a portrait dataset from the computer vision domain. We compare our methods to two factorial learning algorithms, Co-operative Vector Quantization (CVQ)[34] and Multiple Cause Vector Quantization (MCVQ)[87]. We also compare against traditional single clustering algorithms like k-means and non-negative matrix approximation (NNMA)[71]. On all the datasets, both of our algorithms significantly outperform the factorial learning as well as the single clustering algorithms. The factorial learning methods work reasonably well on a few synthetic datasets which exactly satisfy their respective model assumptions. But they are not robust in the case where model assumptions are even slightly violated. Because of this, their performance is poor on real-world datasets and other synthetic datasets. In comparison, our algorithms are more robust and perform significantly better on all the datasets. For the music dataset both our algorithms achieve around 20% improvement in accuracy over the factorial learning and single clustering algorithms (k-means and NNMA). Similarly, for the portrait dataset we achieve an improvement of 30% over the baseline algorithms.

This work on disparate clustering was published in [58, 59].

8.1 Disparate Clustering

For simplicity, we present our methods for uncovering two disparate clusterings from the data; our techniques can be generalized to uncover more than two clusterings. We propose the following approaches:

- Decorrelated-kmeans approach: In this approach we try to fit each clustering to the entire data, while requiring that different clusterings be decorrelated with each other. To this end, we introduce a novel measure for correlation between clusterings. This measure is motivated by the fact that if the representative vectors of two clusterings are orthogonal to one another, then the labellings generated by nearest neighbor assignments for these representative vectors are independent under some mild conditions (see Section 8.1.3.1).

- Sum of parts approach: In this approach we model the data as a sum of independent components, each of which is a mixture model. We then associate each component with a clustering. Further, as the distribution of the sum of two independent random variables is the convolution of the distributions (see [25]), we model the observed data as being sampled from a convolution of two mixtures. Thus, our approach leads us to the problem of learning a convolution of mixtures. Note that the individual components uncovered by this approach may not be good approximations to the data by themselves, but their sum is. This is in complete contrast

to the first approach where we try to approximate the data individually by each component.

8.1.1 First Approach: Decorrelated-kmeans

Given a set of data points $Z = \{\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_n\} \subseteq \mathbb{R}^m$, we aim to uncover two clusterings C^1 and C^2. Specifically, we wish to partition the set Z into k_1 groups for the first clustering C^1 and k_2 groups for the second clustering C^2. To achieve this task, we try to find *decorrelated* clusterings each of which approximates the data as a whole. We propose the following objective function:

$$G(\boldsymbol{\mu}_{1\ldots k_1}, \boldsymbol{\nu}_{1\ldots k_2}) = \sum_i \sum_{\boldsymbol{z}\in C_i^1} \|\boldsymbol{z} - \boldsymbol{\mu}_i\|^2 + \sum_j \sum_{\boldsymbol{z}\in C_j^2} \|\boldsymbol{z} - \boldsymbol{\nu}_j\|^2 + \lambda \sum_{i,j} (\boldsymbol{\beta}_j^T \boldsymbol{\mu}_i)^2 + \lambda \sum_{i,j} (\boldsymbol{\alpha}_i^T \boldsymbol{\nu}_j)^2, \quad (8.1)$$

where C_i^1 is cluster i of the first clustering, C_j^2 is cluster j of the second clustering, and $\lambda > 0$ is a regularization parameter. The vector $\boldsymbol{\mu}_i$ is the *representative* vector of C_i^1, $\boldsymbol{\nu}_j$ is the *representative* vector of C_j^2, $\boldsymbol{\alpha}_i$ is the mean vector of C_i^1 and $\boldsymbol{\beta}_j$ is the mean vector of C_j^2.

The first two terms of (8.1) correspond to a k-means type error term for the clusterings, with a crucial difference being that the "representative" vector of a cluster may not be its mean vector. The last two terms are regularization terms that measure the decorrelation between the two clusterings. In order to extend this formulation for $T \geq 2$ clusterings, we add k-means type error

terms for each of the T clusterings. Furthermore, we add $T \times (T-1)/2$ terms corresponding to the decorrelation between pairs of clusterings.

The decorrelation measure given above is motivated by the intuition that if the "representative" vectors of two clusterings are orthogonal to one another, then the labellings generated by nearest neighbor assignments for these vectors are independent. We provide a theoretical basis for the above intuition in Section 8.1.3.1. Also, an important advantage of the proposed decorrelation measure is that the objective function remains strictly and jointly convex in the $\boldsymbol{\mu}_i$'s and $\boldsymbol{\nu}_j$'s (assuming fixed C_i^1's and C_j^2's).

To minimize the objective function (8.1), we present an iterative algorithm which we call Decorrelated-kmeans (Algorithm 1). We fix C^1 and C^2 to obtain $\boldsymbol{\mu}_i$'s and $\boldsymbol{\nu}_j$'s that minimize (8.1) and then assign each point $\boldsymbol{z}$ to C_i^1 such that $i = \text{argmin}_l \|\boldsymbol{z} - \boldsymbol{\mu}_l\|^2$ and to C_j^2 such that $j = \text{argmin}_l \|\boldsymbol{z} - \boldsymbol{\nu}_l\|^2$. We initialize one of the clusterings using k-means with $k = k_1$ and the other clustering randomly.

For computing the $\boldsymbol{\mu}_i$'s and $\boldsymbol{\nu}_j$'s, we need to minimize (8.1). The gradient of the objective function in (8.1) w.r.t $\boldsymbol{\mu}_i$ is given by:

$$\frac{\partial G}{\partial \boldsymbol{\mu}_i} = -2\left(\sum_{\boldsymbol{z} \in C_i^1} \boldsymbol{z}\right) + 2\left(\sum_j n_{ij}\right)\boldsymbol{\mu}_i + 2\lambda \sum_j (\boldsymbol{\beta}_j^T \boldsymbol{\mu}_i)\boldsymbol{\beta}_j,$$

where n_{ij} is the number of points that belong to C_i^1 and C_j^2.

Now, $(\boldsymbol{\beta}_j^T \boldsymbol{\mu}_i)\boldsymbol{\beta}_j = (\boldsymbol{\beta}_j \boldsymbol{\beta}_j^T)\boldsymbol{\mu}_i$ and $\boldsymbol{\alpha}_i = \frac{\left(\sum_{\boldsymbol{z} \in C_i^1} \boldsymbol{z}\right)}{\sum_j n_{ij}}$. Thus,

$$\frac{\partial G}{\partial \boldsymbol{\mu}_i} = -2 \sum_j n_{ij} \boldsymbol{\alpha}_i + 2 \sum_j n_{ij} \boldsymbol{\mu}_i + 2\lambda \left(\sum_j \boldsymbol{\beta}_j \boldsymbol{\beta}_j^T \right) \boldsymbol{\mu}_i.$$

Similarly,

$$\frac{\partial G}{\partial \boldsymbol{\nu}_j} = -2 \sum_i n_{ij} \boldsymbol{\beta}_j + 2 \sum_i n_{ij} \boldsymbol{\nu}_j + 2\lambda \left(\sum_i \boldsymbol{\alpha}_i \boldsymbol{\alpha}_i^T \right) \boldsymbol{\nu}_j.$$

Setting the gradients to zero gives us the following equations:

$$\boldsymbol{\mu}_i = \left(I + \frac{\lambda}{\sum_j n_{ij}} \sum_j \boldsymbol{\beta}_j \boldsymbol{\beta}_j^T \right)^{-1} \boldsymbol{\alpha}_i, \tag{8.2}$$

$$\boldsymbol{\nu}_j = \left(I + \frac{\lambda}{\sum_i n_{ij}} \sum_i \boldsymbol{\alpha}_i \boldsymbol{\alpha}_i^T \right)^{-1} \boldsymbol{\beta}_j. \tag{8.3}$$

Since the objective function (8.1) is strictly and jointly convex in both $\boldsymbol{\mu}_i$'s and $\boldsymbol{\nu}_j$'s, the above updates lead to a global minima of the objective function (8.1) for *fixed* C^1 and C^2.

8.1.1.1 Computing the updates efficiently:

Computing the updates given by (8.2) and (8.3) requires computing the inverse of an $m \times m$ matrix, where m is the dimensionality of the data. Thus updating all the $\boldsymbol{\mu}_i$'s and $\boldsymbol{\nu}_j$'s directly would seem to require $\mathrm{O}(k_1 m^3 + k_2 m^3)$ operations, which is cubic in the dimensionality of the data. We now give a substantially faster way to compute the updates in time linear in the dimensionality. Using the Sherman-Morrison-Woodbury formula (see [39]) for the inverse in (8.2), we get

$$\left(I + \xi_i V V^T\right)^{-1} = I - \xi_i V \left(I + \xi_i V^T V\right)^{-1} V^T,$$

where $\xi_i = \frac{\lambda}{\sum_j n_{ij}}$ and $V = [\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_{k_2}]$. Using the eigenvalue decomposition $V^T V = Q \Sigma Q^T$ we see that

$$\left(I + \xi_i V^T V\right)^{-1} = Q \left(I + \xi_i \Sigma\right)^{-1} Q^T.$$

Since $V^T V$ is a $k_2 \times k_2$ matrix its eigenvalue decomposition can be computed in $\mathrm{O}(k_2^3)$ time. Also, as $(I + \xi_i \Sigma)^{-1}$ is a diagonal matrix, calculating its inverse requires just $\mathrm{O}(k_2)$ operations. The updates for $\boldsymbol{\mu}_i$'s can now be rewritten as,

$$\boldsymbol{\mu}_i = \left(I - \xi_i V Q \left(I + \xi_i \Sigma\right)^{-1} Q^T V^T\right) \boldsymbol{\alpha}_i. \tag{8.4}$$

Similarly, the updates for $\boldsymbol{\nu}_j$'s can now be written as,

$$\boldsymbol{\nu}_j = \left(I - \zeta_j M U \left(I + \zeta_j \Lambda\right)^{-1} U^T M^T\right) \boldsymbol{\beta}_j, \tag{8.5}$$

where $\zeta_j = \frac{\lambda}{\sum_i n_{ij}}$, $M = [\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_{k_1}]$, and $U \Lambda U^T$ is the eigenvalue decomposition of $M^T M$.

Using these updates reduces the computational complexity of computing all the $\boldsymbol{\mu}_i$'s and $\boldsymbol{\nu}_j$'s to $O(mk_1^2 + mk_2^2 + k_1^3 + k_2^3)$. If $m > k = \max(k_1, k_2)$, which is typically the case, the above bound becomes $O(mk^2)$.

8.1.1.2 Determining λ:

The regularization parameter λ plays an important role in the Decorrelated-kmeans algorithm. It determines the tradeoff between minimizing the individual clustering error of each clustering (first two terms in (8.1)) and finding decorrelated cluster centers for the different clusterings (last two terms in

Algorithm 4 Decorrelated-kmeans (Dec-kmeans)

Input: Data $\mathcal{Z} = \{\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_n\}$
k_1: Number of clusters in first clustering (C^1)
k_2: Number of clusters in second clustering(C^2)
λ: regularization parameter

Output: C^1, C^2: Two different clusterings

1. $C^1 \leftarrow k$-means($\mathcal{Z}$), $C^2 \leftarrow$ Random assignment
2. **repeat**
   - 2.1. $\boldsymbol{\alpha}_i \leftarrow$ ComputeMean(C_i^1), for all $1 \leq i \leq k_1$
   - 2.2. $\boldsymbol{\beta}_j \leftarrow$ ComputeMean(C_j^2), for all $1 \leq j \leq k_2$
   - 2.3. Update $\boldsymbol{\mu}_i$ and $\boldsymbol{\nu}_j$ for all i, j using (8.4), (8.5)
   - 2.4. $\forall \boldsymbol{z}$, $C_i^1 \leftarrow C_i^1 \cup \{\boldsymbol{z}\}$, if $i = \arg\min_l \|\boldsymbol{z} - \boldsymbol{\mu}_l\|^2$.
   - 2.5. $\forall \boldsymbol{z}$, $C_j^2 \leftarrow C_j^2 \cup \{\boldsymbol{z}\}$, if $j = \arg\min_l \|\boldsymbol{z} - \boldsymbol{\nu}_l\|^2$.
4. **until** convergence

return $C^1,\ C^2$

(8.1)). Empirically, we observe that the clustering accuracies are good when $\lambda \in [100, 10000]$, which is a large range. But, a different scaling of the data can change this range for λ. Hence, we determine λ using a simple heuristic. Note that for small values of λ, the Decorrelated-kmeans algorithm finds approximately the same clusters for both the clusterings. While for high value of λ it tries to find clusterings which are orthogonal to each other, even though both the clusterings may not fit the data well. Thus, a suitable λ balances out both the objectives and hence generally there is a large change in objective function value when λ is perturbed slightly. Based on this intuition we form a heuristic to determine λ: start with a large λ and find different clusterings of

the data while decreasing λ, and select a λ for which the drop in the objective function is the highest. Note that different variants of the heuristic can be used depending on the data and domain knowledge. For example, if the data is large, then a subset of the data can be used for finding clusterings or if the data is noisy then a more robust measure like average change in objective function should be preferred over the maximum change measure for selecting λ.

8.1.2 Second Approach: Sum of Parts

In this section, we describe our "sum of parts" approach. Let $\mathcal{Z} = \{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_n\}$ be the observed m-dimensional data sampled from a random variable Z. We model Z as a sum $X + Y$, where X, Y are independent random variables and are drawn from mixtures of distributions. Specifically,

$$p_X = \sum_{i=1}^{k_1} a_i p_{X_i}, \qquad p_Y = \sum_{j=1}^{k_2} b_j p_{Y_j}.$$

The problem of learning independent components can now be stated as: Given data sampled according to Z, recover the parameters of the probability distributions p_{X_i}, p_{Y_j} along with the mixing weights a_i, b_j.

As $Z = X + Y$, the probability density function of Z is the convolution of p_X and p_Y [25, Section A.4.11]. Thus,

$$p_Z(\boldsymbol{z}) = (p_X * p_Y)(\boldsymbol{z}) = \sum_{i=1}^{k_1} \sum_{j=1}^{k_2} (a_i b_j) \cdot (p_{X_i} * p_{Y_j})(\boldsymbol{z}) \ , \tag{8.6}$$

where $f_1 * f_2(\boldsymbol{z}) = \int_{\mathbb{R}^m} f_1(\boldsymbol{x}) \cdot f_2(\boldsymbol{z} - \boldsymbol{x}) d\boldsymbol{x}$ denotes the convolution of f_1 and f_2.

From (8.6) it follows that when the distributions p_{X_i} and p_{Y_j} belong to a family of distributions closed under convolution, Z can be viewed as a mixture of $k_1 \times k_2$ distributions. However, the problem of learning the components X and Y from Z is harder than that of simply learning the parameters of a mixture model, as along with learning the $k_1 \times k_2$ component distributions one must also be able to factor them out. In the following section, we give a generalized Expectation Maximization (EM) algorithm for learning the parameters of the component mixtures when the base distributions are spherical multi-variate Gaussians. Our techniques can be extended to more general distributions like non-spherical Gaussians and potentially to other families closed under convolution.

8.1.2.1 Learning the convolution of a mixture of Gaussians

Let the components X and Y be mixtures of spherical Gaussians, i.e., $p_X = \sum_{i=1}^{k_1} a_i \mathcal{N}(\boldsymbol{\mu_i}, \sigma^2)$ and $p_Y = \sum_{i=1}^{k_2} b_i \mathcal{N}(\boldsymbol{\nu_i}, \sigma^2)$. As in our first approach we initialize the EM algorithm (Algorithm 2) by k-means for the first clustering and a random assignment for the second clustering. We initialize $\boldsymbol{\mu}_i^0$'s and $\boldsymbol{\nu}_j^0$'s to be the means of the first and second clusterings respectively. To initialize σ we use a heuristic presented in [15],

$$\sigma = \frac{1}{\sqrt{2m}} \min \left(\min_{i \neq j} \|\boldsymbol{\mu}_i^0 - \boldsymbol{\mu}_j^0\|, \min_{i \neq j} \|\boldsymbol{\nu}_i^0 - \boldsymbol{\nu}_j^0\| \right).$$

E-step:

Let $p_{ij}^t(\boldsymbol{z})$ denote the conditional probability that $\boldsymbol{z}$ comes from the Gaussian $p_{X_i} * p_{Y_j}$ given the current parameters. As our main objective is to cluster the data, we use hard assignments in the E-step to ease the computations involved. The E-step in this case will be:

$$p_{ij}^{t+1}(\boldsymbol{z}) = \begin{cases} 1, & \text{if } (i,j) = \\ & \text{argmax}_{(r,s)}\{a_r^t b_s^t \cdot \mathcal{N}\left(\boldsymbol{\mu}_r^t + \boldsymbol{\nu}_s^t, 2(\sigma^t)^2\right)(\boldsymbol{z})\} \\ 0, & \text{otherwise.} \end{cases} \tag{8.7}$$

Note that, to uncover T different clusterings from the data, $O(k^T)$ computational operations are required for each data point in the E-step. Ghahramani[34] suggested various approximation methods to reduce the time complexity of this estimation, and the same can be applied to our setting as well. In our implementation, we use Gibbs sampling for approximating the distribution of labels, $p_{ij}^{t+1}(z)$, when the parameters of the base distributions are fixed.

M-step:

In the M-step, we use the clusterings updated in the E-step (specified by p_{ij}^{t+1}'s) to estimate the parameters of the distributions. Formally, we maximize the log-likelihood:

$$\left(\boldsymbol{\mu}_{1\ldots k_1}^{t+1}, \boldsymbol{\nu}_{1\ldots k_2}^{t+1}, \sigma^{t+1}, a_{1\ldots k_1}^{t+1}, b_{1\ldots k_2}^{t+1}\right) = \underset{\substack{\boldsymbol{\mu}_{1\ldots k_1}, \boldsymbol{\nu}_{1\ldots k_2}, \sigma, \\ a_{1\ldots k_1}, b_{1\ldots k_2}}}{\text{argmax}} \sum_{i,j,\boldsymbol{z}} p_{ij}^{t+1}(\boldsymbol{z}) \ \log\left(a_i b_j \mathcal{N}(\boldsymbol{\mu}_i + \boldsymbol{\nu}_j, \sigma)(\boldsymbol{z})\right).$$

The mixture weights and variance σ can be easily computed by differentiating w.r.t. a_i's, b_j's, σ and setting the derivatives to zero. This gives us the following expressions:

$$a_i^{t+1} = \frac{1}{n} \sum_j \sum_{\boldsymbol{z}} p_{ij}^{t+1}(\boldsymbol{z}), \tag{8.8}$$

$$b_j^{t+1} = \frac{1}{n} \sum_i \sum_{\boldsymbol{z}} p_{ij}^{t+1}(\boldsymbol{z}), \tag{8.9}$$

$$(\sigma^{t+1})^2 = \frac{1}{2mn} \sum_{i,j,\boldsymbol{z}} p_{ij}^{t+1}(\boldsymbol{z}) \ \|\boldsymbol{z} - \boldsymbol{\mu}_i^t - \boldsymbol{\nu}_j^t\|^2. \tag{8.10}$$

Computing the means to maximize the log-likelihood is more involved and it reduces to minimizing the following objective function:

$$\min_{\boldsymbol{\mu}_{1\ldots k_1}, \boldsymbol{\nu}_{1\ldots k_2}} \quad F(\boldsymbol{\mu}_{1\ldots k_1}, \boldsymbol{\nu}_{1\ldots k_2}) = \sum_{i,j,\boldsymbol{z}} p_{ij}^{t+1}(\boldsymbol{z}) \|\boldsymbol{z} - \boldsymbol{\mu}_i - \boldsymbol{\nu}_j\|^2. \tag{8.11}$$

Note that there exist multiple solutions for the above equation; since we can translate the means $\boldsymbol{\mu}_i$'s by a fixed vector $\boldsymbol{w}$ and the means $\boldsymbol{\nu}_j$'s by $-\boldsymbol{w}$ to get another set of solutions. Note that the CVQ [34] algorithm also suffers from the same problem of multiple solutions. Out of all the solutions to (8.11), the solutions which give maximally disparate clusterings are more desirable. To obtain such solutions we regularize the $\boldsymbol{\mu}_i$'s and $\boldsymbol{\nu}_j$'s to have small correlation with each other. To this end we introduce a regularization term to make the $\boldsymbol{\mu}_i$'s and $\boldsymbol{\nu}_j$'s orthogonal to one another. This correlation measure is similar to the measure discussed in the previous Decorrelated-kmeans approach (Section 8.1.1). Formally, we minimize the following objective function:

$$\tilde{F}(\boldsymbol{\mu}_{1\ldots k_1}, \boldsymbol{\nu}_{1\ldots k_2}) \quad = \quad \sum_{i,j,\boldsymbol{z}} p_{ij}^{t+1}(\boldsymbol{z}) \|\boldsymbol{z} \ - \ \boldsymbol{\mu}_i \ - \ \boldsymbol{\nu}_j\|^2 \ + \ \lambda \sum_{i,j} (\boldsymbol{\mu}_i^T \boldsymbol{\nu}_j)^2, \tag{8.12}$$

where $\lambda > 0$ is a regularization parameter and can be selected using a heuristic similar to the one described in Section 8.1.1.2.

Observe that the above objective is not jointly convex in $\boldsymbol{\mu}_i$ and $\boldsymbol{\nu}_j$ but is strictly convex in $\boldsymbol{\mu}_i$ for fixed $\boldsymbol{\nu}_j$'s and vice-versa. To minimize $\tilde{F}$, we use the block coordinate descent algorithm ([111]) where we fix $\boldsymbol{\nu}_j$'s to minimize $\boldsymbol{\mu}_i$ and vice-versa. By differentiating (8.12) w.r.t. $\boldsymbol{\mu}_i$ and $\boldsymbol{\nu}_j$ and setting the derivatives to zero we get,

$$\left(I + \frac{\lambda \sum_j \boldsymbol{\nu}_j \boldsymbol{\nu}_j^T}{\sum_j n_{ij}}\right) \boldsymbol{\mu}_i + \frac{\sum_j n_{ij} \boldsymbol{\nu}_j}{\sum_j n_{ij}} = \boldsymbol{\alpha}_i,$$
$$\left(I + \frac{\lambda \sum_i \boldsymbol{\mu}_i \boldsymbol{\mu}_i^T}{\sum_i n_{ij}}\right) \boldsymbol{\nu}_j + \frac{\sum_i n_{ij} \boldsymbol{\mu}_i}{\sum_i n_{ij}} = \boldsymbol{\beta}_j,$$

where $n_{ij} = \sum_{\boldsymbol{z}} p_{ij}^{t+1}(\boldsymbol{z})$ is the number of data-points that belong to cluster i of the first clustering and cluster j of the second clustering, $\boldsymbol{\alpha}_i$ denotes the mean of all points that are assigned to cluster i in the first clustering and $\boldsymbol{\beta}_j$ denotes the mean of points assigned to cluster j in the second clustering, i.e. ,

$$\boldsymbol{\alpha}_i = \frac{1}{n a_i} \sum_j \sum_{\boldsymbol{z}} \boldsymbol{z} p_{ij}^{t+1}(\boldsymbol{z}), \tag{8.13}$$

$$\boldsymbol{\beta}_j = \frac{1}{n b_j} \sum_i \sum_{\boldsymbol{z}} p_{ij}^{t+1}(\boldsymbol{z}). \tag{8.14}$$

To solve for $\boldsymbol{\mu}_i$ and $\boldsymbol{\nu}_j$ in the above equations we use an alternative minimization scheme - we iteratively update the $\boldsymbol{\mu}_i$ and $\boldsymbol{\nu}_j$ as follows:

$$\boldsymbol{\mu}_i = \left(I + \frac{\lambda \sum_j \boldsymbol{\nu}_j \boldsymbol{\nu}_j^T}{\sum_j n_{ij}}\right)^{-1} \left(\boldsymbol{\alpha}_i - \frac{\sum_j n_{ij} \boldsymbol{\nu}_j}{\sum_j n_{ij}}\right) \tag{8.15}$$

$$\boldsymbol{\nu}_j = \left(I + \frac{\lambda \sum_i \boldsymbol{\mu}_i \boldsymbol{\mu}_i^T}{\sum_i n_{ij}}\right)^{-1} \left(\boldsymbol{\beta}_j - \frac{\sum_i n_{ij} \boldsymbol{\mu}_i}{\sum_i n_{ij}}\right). \tag{8.16}$$

For initialization we set $\boldsymbol{\nu}_j$ to be $\boldsymbol{\beta}_j$ for each j. Below we prove that this scheme converges to a local minima of (8.12).

Lemma 12. *The updates for $\boldsymbol{\mu}_i$ and $\boldsymbol{\nu}_j$ given by* (8.15) *converge to a local minimum of the regularized objective function given by* (8.12).

Proof. As the updates (8.15) minimize the objective function at each iteration, the updates converge to a fixed point[111]. Also, the objective function (8.12) is strictly-convex in $\boldsymbol{\mu}_i$ for fixed $\boldsymbol{\nu}_j$'s and vice-versa. Thus, any fixed point of (8.12) is also a local minimum. It now follows that our updates converge to a local minimum of the objective function. □

Theorem 12. *Algorithm 2 monotonically decreases the objective function:*

$$F = \sum_{i,j,\boldsymbol{z}} p_{ij}^{t+1}(\boldsymbol{z}) \frac{\|\boldsymbol{z} - \boldsymbol{\mu}_i - \boldsymbol{\nu}_j\|^2}{2\sigma^2} + \lambda \sum_{i,j} (\boldsymbol{\mu}_i^T \boldsymbol{\nu}_j)^2 \tag{8.17}$$

Proof. Let F_t be the objective function value at the start of t-th iteration, F_t^E be the objective function value after the E-step of t-th iteration and $F_t^M = F_{t+1}$ be the objective function after M-step of t-th iteration. The E-step assigns new labels according to 8.7, which is equivalent to minimizing:

$$\sum_{i,j,\boldsymbol{z}} p_{ij}^{t+1}(\boldsymbol{z}) \frac{\|\boldsymbol{z} - \boldsymbol{\mu}_i - \boldsymbol{\nu}_j\|^2}{2\sigma^2},$$

with μ_i and ν_j being fixed.

Thus, the first term of the objective function (8.17) is decreased by the E-step while the second term remains fixed. Hence, $F_t \geq F_t^E$. Using Lemma 12, $F_t^E \geq F_t^M$, as only μ_i's and ν_j's are variables with p_{ij} fixed (σ can be absorbed in λ). Thus, $F_t \geq F_{t+1}$. □

8.1.2.2 Computing the updates efficiently:

Using techniques similar to Section 8.1.1.1, the update for $\boldsymbol{\mu}_i$ can be written as:

$$\boldsymbol{\mu}_i = \left(I - \xi_i V Q \left(I + \xi_i \Sigma\right)^{-1} Q^T V^T\right) \left(\boldsymbol{\alpha}_i - \frac{\sum_j n_{ij} \boldsymbol{\nu}_j}{\sum_j n_{ij}}\right), \tag{8.18}$$

where, $\xi_i = \frac{\lambda}{\sum_j n_{ij}}$ and $V = [\boldsymbol{\nu}_1, \ldots, \boldsymbol{\nu}_{k_2}]$ and $Q\Sigma Q^T$ is the eigenvalue decomposition of $V^T V$.

Similarly, the update for $\boldsymbol{\nu}_j$ can be written as,

$$\boldsymbol{\nu}_j = \left(I - \zeta_j M U \left(I + \zeta_j \Lambda\right)^{-1} U^T M^T\right) (\boldsymbol{\beta}_j - \frac{\sum_i n_{ij} \boldsymbol{\mu}_i}{\sum_i n_{ij}}), \tag{8.19}$$

where, $\zeta_j = \frac{\lambda}{\sum_i n_{ij}}$, $M = [\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_{k_1}]$ and $M^T M = U \Lambda U^T$.

As in Section 8.1.1.1, the above updates reduce the computational complexity of computing all the $\boldsymbol{\mu}_i$'s and $\boldsymbol{\nu}_j$'s from $\mathrm{O}(k_1 m^3 + k_2 m^3)$ to $O(m(k_1^2 + k_2^2))$.

8.1.3 Discussion

8.1.3.1 Decorrelation measure:

Now we motivate the decorrelation measures used in equations (8.1) and (8.12). For this, we will need the following two lemmas about uniqueness of projection and multivariate Gaussians. In the following lemmas, for a subspace S of $\mathbb{R}^m$ let $P_S : \mathbb{R}^m \to \mathbb{R}^m$ be the orthogonal projection operator onto the subspace S.

Algorithm 5 Convolutional-EM (Conv-EM)

Input: Data $\mathcal{Z} = \{\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_n\}$
k_1: Number of clusters in first clustering (C^1)
k_2: Number of clusters in second clustering(C^2)
λ: regularization parameter

Output: C^1, C^2: Two different clusterings

1. $C^1 \leftarrow k$-means($\mathcal{Z}$), $C^2 \leftarrow$Random assignment
2. $\boldsymbol{\mu}_i \leftarrow$ComputeMean($C_i^1$), $\boldsymbol{\nu}_j \leftarrow$ComputeMean($C_j^2$)
3. $a_i = \frac{1}{k_1}$, $b_j = \frac{1}{k_2}$
4. **repeat**

   E Step:

   4.1. For each $\boldsymbol{z}$, assign $p_{ij}(\boldsymbol{z})$ using (8.7).

   M Step:

   4.2. Assign a_i, b_j and σ using (8.8), (8.9), (8.10).

   4.3. Assign $\boldsymbol{\alpha}_i$ and $\boldsymbol{\beta}_j$ using (8.13), (8.14).

   4.4. $\boldsymbol{\nu}_j \leftarrow \boldsymbol{\beta}_j$

   4.5. **repeat until convergence**

   - Update $\boldsymbol{\mu}_i$ using (8.18).
   - Update $\boldsymbol{\nu}_j$ using (8.19).

5. **until** convergence
6. $C_i^1 = \{\boldsymbol{z} | p_{ij}(\boldsymbol{z}) = 1, \forall j\}$, $C_j^2 = \{\boldsymbol{z} | p_{ij}(\boldsymbol{z}) = 1, \forall j\}$

return $C^1,\ C^2$

Lemma 13. *Let S_1, S_2 be subspaces of $\mathbb{R}^m$ such that $S_1 \cap S_2 = \{\mathbf{0}\}$. Then, for all $x \in S_1$, and $y \in S_2$, there exists a unique $u \in S_1 + S_2$ such that $P_{S_1}(u) = x$ and $P_{S_2}(u) = y$.*

Proof. Let $x \in S_1$ and $y \in S_2$. Also, let P_1, P_2 be the projection matrices for the projection operators P_{S_1} and P_{S_2} respectively. We first formulate the hypothesis that $S_1 \cap S_2 = \{\mathbf{0}\}$ in terms of the matrices P_1, P_2 by showing that

$I - P_1P_2$ and $I - P_2P_1$ are invertible. Suppose on the contrary that $I - P_1P_2$ is not invertible. Then, for some non-zero z we must have, $(I - P_1P_2)z = 0$, i.e., $z = P_1P_2z$. Recall that for a projection matrix P into a subspace S we always have $\|Pu\| \leq \|u\|$ with equality if and only if $u \in S$. Thus, we have

$$\|z\| = \|P_1P_2z\| \leq \|P_2z\| \leq \|z\|.$$

Therefore, $z = P_1P_2z = P_2z$, which is possible only if $z \in S_1$ and $z \in S_2$. This contradicts the assumption that $S_1 \cap S_2 = \{\mathbf{0}\}$, $I - P_1P_2$ must be invertible. Similarly, we can also show that $I - P_2P_1$ is invertible.

Now, to prove the lemma we need to show that there exists a unique $u \in S_1 + S_2$ such that '$x = P_1u$ and $y = P_2u$'. Since, $S_1 \cap S_2 = \{\mathbf{0}\}$, solving the above system of equations is equivalent to solving for $v \in S_1$, and $w \in S_2$ such that

$$x = P_1(v + w), \quad y = P_2(v + w).$$

Manipulating the above equations, we get:

$$(I - P_1P_2)v = x - P_1y, \quad (I - P_2P_1)w = y - P_2x.$$

The existence and uniqueness of v, w follow from the fact that $I - P_1P_2$ and $I - P_2P_1$ are invertible. □

Lemma 14. *Let $Z \in \mathbb{R}^m$ denote a random variable with spherical Gaussian distribution. Let $S_1, S_2 \subseteq \mathbb{R}^m$ be two subspaces such that $S_1 \cap S_2 = \{0\}$ and let $Z_1 = P_{S_1}(Z)$, $Z_2 = P_{S_2}(Z)$ be the random variables obtained by projecting Z onto S_1, S_2 respectively. Then, the random variables Z_1 and Z_2 are independent if and only if the subspaces S_1 and S_2 are orthogonal.*

Proof. $\Longleftarrow$ If S_1 and S_2 are orthogonal, then for $u_1 \in S_1$ and $u_2 \in S_2$, $Pr[Z = u_1 + u_2] = Pr[Z_1 = u_1, Z_2 = u_2]$. Further, since Z has a spherical Gaussian distribution so do Z_1 and Z_2. The independence of Z_1 and Z_2 follows easily from the above observations.

$\Longrightarrow$ Let the random variables Z_1 and Z_2 be independent. Note that without loss of generality we can assume that Z has mean $\mathbf{0}$ (as else we can translate Z). Furthermore, we can also assume that the support of Z is contained in $S_1 + S_2$. This is because, $P_{S_1} = P_{S_1} \circ P_{S_1+S_2}$ and $P_{S_1+S_2}(Z)$ is also distributed as a spherical multivariate Gaussian. For the rest of the proof we will suppose that $S_1 + S_2 = \mathit{support}(Z) = \mathbb{R}^m$ and that Z has mean $\mathbf{0}$.

Using Lemma 13, for $u \in \mathbb{R}^m$, we have

$$Pr[Z = u] = Pr[P_{S_1}(Z) = P_{S_1}(u), P_{S_2}(Z) = P_{S_2}(u)].$$

As Z_1 and Z_2 are independent the above can be rewritten as

$$Pr[Z = u] = Pr[P_{S_1}(Z) = P_{S_1}(u)] \cdot Pr[P_{S_2}(Z) = P_{S_2}(u)].$$

Now, since the projection of a spherical multivariate Gaussian is also a spherical multivariate Gaussian, substituting probability density formulae in the above equation we get the following:

$$\frac{1}{(2\pi)^{m/2}} e^{-\frac{1}{2}\|u\|^2} = \frac{1}{(2\pi)^{m_1/2}} e^{-\frac{1}{2}\|u_1\|^2} \cdot \frac{1}{(2\pi)^{m_2/2}} e^{-\frac{1}{2}\|u_2\|^2},$$

where, m_1, m_2 denote the dimensions of S_1, S_2 respectively and $u_1 = P_{S_1}(u), u_2 = P_{S_2}(u)$. Noting that $m = m_1 + m_2$ (since $S_1 \cap S_2 = \{\mathbf{0}\}$) the above equation

can be simplified to

$$\|u\|^2 = \|P_{S_1}(u)\|^2 + \|P_{S_2}(u)\|^2.$$

As the above equation holds for all u it also holds in particular for $u \in S_1$. Now, for $u \in S_1$ we have $P_{S_1}(u) = u$, thus we get

$$\forall u \in S_1, P_{S_2}(u) = 0.$$

The above condition can easily be shown to be equivalent to S_1 and S_2 being orthogonal. □

We now give the motivation for our decorrelation measures. Let $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_{k_1}$ and $\boldsymbol{\nu}_1, \ldots, \boldsymbol{\nu}_{k_2}$ be vectors in $\mathbb{R}^m$ such that $\boldsymbol{\mu}_i$ and $\boldsymbol{\nu}_j$ are orthogonal for all i, j. Let S_1 be the space spanned by $\boldsymbol{\mu}_i$'s and S_2 be the space spanned by $\boldsymbol{\nu}_j$'s. Define the "nearest-neighbor" random variables, $NN_1(Z), NN_2(Z)$ as follows:

$$\begin{aligned} NN_1(Z) &= \text{argmin}\{\|Z - \boldsymbol{\mu}_i\| : 1 \leq i \leq k_1\}, \\ NN_2(Z) &= \text{argmin}\{\|Z - \boldsymbol{\nu}_j\| : 1 \leq j \leq k_2\}. \end{aligned} \tag{8.20}$$

Then as S_1 and S_2 are orthogonal to each other, it follows from Lemma 14 that when Z is a spherical multivariate Gaussian, the random variables $NN_1(Z)$ and $NN_2(Z)$ are independent. Similarly, it can be shown that when Z is a spherical multivariate Gaussian, the random variables $\overline{NN}_1$, and $\overline{NN}_2$ defined by,

$$(\overline{NN}_1(Z), \overline{NN}_2(Z)) = \underset{(i,j)}{\text{argmin}}\{\|Z - \boldsymbol{\mu}_i - \boldsymbol{\nu}_j\|\}, \tag{8.21}$$

are independent. Note that in equations (8.1), (8.12) we use inner products involving the mean vectors of different clusterings as the correlation measure. Thus, minimizing the correlation measure ideally leads to the mean vectors of different clusterings being orthogonal. Also, observe that we use nearest neighbor assignments of the form (8.20), (8.21) in our algorithms in Decorrelated-kmeans and Convolutional-EM. Thus, the decorrelation measures specified in equations (8.1) and (8.12) intuitively correspond to the labellings of the clusterings being independent.

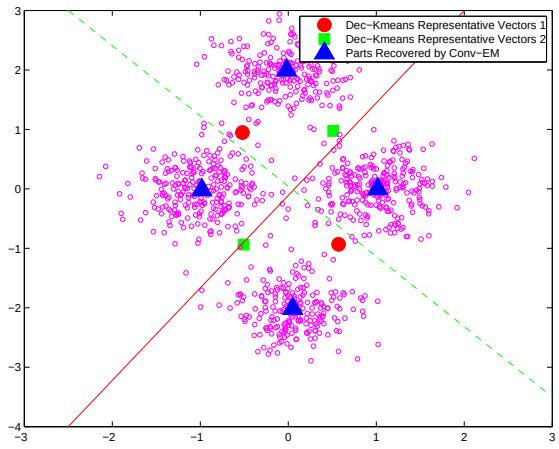


Figure 8.2: Representative vectors obtained by Dec-kmeans and the parts obtained by Conv-EM. The bold line represents the separating hyperplane for the first clustering, while the dotted line represents the separating hyperplane for the second clustering. Conv-EM produced mean vectors $\{\mu_1, \mu_2, \nu_1, \nu_2\}$ and subsequently each of the four parts are obtained by $\mu_i + \nu_j$ $(i \in \{1, 2\}, j \in \{1, 2\})$.

8.2 Decorrelated-kmeans vs Convolutional-EM

Decorrelated-kmeans (Algorithm 1) has a three-fold advantage over the "sum of the parts" approach (Algorithm 2):

- Computing the E-step exactly in the "sum of the parts" approach requires $O(k^T)$ computation for each data point, where T is the number of alternative clusterings. On the other hand, in Decorrelated-kmeans, each label assignment step requires just $O(kT)$ computations as the error terms for different clusterings are independent in (8.1). Thus, Decorrelated-kmeans is more scalable than Convolutional-EM with respect to the number of alternative clusterings.

- The M-step in the "sum of the parts" approach solves a non-convex problem and requires an iterative procedure to reach a local minimum. In the Decorrelated-kmeans approach, computing the representative vectors (the equivalent of M-step) requires solving a convex problem and the optimal solution can be written down in closed form. Hence, estimation of representative vectors is more accurate and efficient for Decorrelated-kmeans.

- Decorrelated-kmeans is a discriminative approach, while Convolutional-EM is a generative model based approach. Thus, the model assumptions are more stringent for the latter approach. This is observed empirically also, where Decorrelated-kmeans works well for all the datasets, but Convolutional-EM suffers on one of the real-life datasets.

On the flip side, there is no natural interpretation of the "representative" vectors given by Decorrelated-kmeans. On the other hand, the means given by Convolutional-EM can naturally be interpreted as giving a part-based representation of the data. This argument is illustrated by Figure 8.2. The representative vectors obtained from Decorrelated-kmeans partition the data into two clusters accurately. But, they don't give any intuitive characterization of the data. In contrast, Convolutional-EM is able to recover the four clusters in the data generated by the addition of two mixtures of Gaussians.

8.3 Experiments

We now provide experimental results on synthetic as well as real-world datasets to show the applicability of our methods. For real-world datasets we consider a music dataset from the text-mining domain and a portrait dataset from the computer-vision domain. We compare our methods against the factorial learning algorithms Co-operative Vector Quantization (CVQ)[34] and Multiple Cause Vector Quantization (MCVQ)[87]. We also compare against single-clustering algorithms such as k-means and NNMA. We will refer to the methods of Sections 8.1.1, 8.1.2 as Dec-kmeans (for Decorrelated-kmeans) and Conv-EM (for Convolutional-EM) respectively.

We also compare our methods against two simple heuristics:

1. *Feature Removal (FR)*: In this approach, we first cluster the data using k-means. Then, we remove the coordinates that have the most *correlation*

with the labels in the obtained clustering. Next, we cluster the data again using the remaining features to obtain the alternative clustering. The correlation between a feature and the labels is taken to be proportional to the total weight of the mean vectors for the feature and inversely proportional to the entropy of the particular feature in the mean vectors. Formally:

$$C(i) = \frac{\sum_j \mu_j^i}{-\sum_j \left(\frac{\mu_j^i}{\sum_l \mu_l^i} \log \frac{\mu_j^i}{\sum_l \mu_l^i} \right)},$$

where μ_j^i is the i-th dimension of the j-th cluster.

2. *Orthogonal Projection (OP)*: This heuristic is motivated by principal gene shaving[47]. The heuristic proceeds by projecting the data onto the subspace orthogonal to the means of the first clustering and uses the projected data for computing the second clustering.

   (a) Cluster the data using a suitable method of clustering.

   (b) Let the means of the obtained clustering be $\boldsymbol{m}_1, \ldots, \boldsymbol{m}_k$. Project the input matrix X onto the space orthogonal to the one spanned by the means $\boldsymbol{m}_1, \ldots, \boldsymbol{m}_k$ to get X'.

   (c) Cluster the columns of X' to obtain a new set of labels, and compute the cluster means $\tilde{\boldsymbol{m}}_1, \ldots, \tilde{\boldsymbol{m}}_k$.

   (d) Repeat steps (b),(c) with means $\tilde{\boldsymbol{m}}_1, \ldots, \tilde{\boldsymbol{m}}_k$.

   (e) Until convergence, repeat steps (a)-(d).

8.3.1 Implementation Details:

All the methods have been implemented in MATLAB. The implementation of MCVQ was obtained from the authors of [87]. Lee and Seung's algorithm[71] is used for NNMA. Experiments were performed on a Linux machine with a 2.4 GHz Pentium IV processor and 1 GB main memory. For the real-world datasets, we report results in terms of accuracy with the true labels. As the number of clusters can be high in the synthetic datasets, we report results in terms of normalized mutual information (NMI) [98]. For all the experiments, accuracy/NMI is averaged over 100 runs.

8.3.2 Synthetic Datasets:

For our experiments we generate synthetic datasets as a sum of independent components. Let $\mathcal{X}$ and $\mathcal{Y}$ be samples drawn from two independent mixtures of multivariate Gaussians. To evaluate our methods in various settings, we generate the final dataset $\mathcal{Z}$ by combining $\mathcal{X}$ and $\mathcal{Y}$ in three different ways. By viewing $\mathcal{X}$ and $\mathcal{Y}$ as the *components* of the datasets, and clustering based on these components we get two different clusterings of the data.

1. Concatenated dataset: This dataset is produced by simply concatenating the features of $\mathcal{X}$ and $\mathcal{Y}$, i.e., $\mathcal{Z} = \begin{bmatrix} \mathcal{X} \\ \mathcal{Y} \end{bmatrix}$.

2. Partial overlap dataset: In this dataset we allow a few of the features of $\mathcal{X}$ and $\mathcal{Y}$ to overlap. Specifically, let $\mathcal{X} = \begin{bmatrix} \mathcal{X}_1 \\ \mathcal{X}_2 \end{bmatrix}$ and $\mathcal{Y} = \begin{bmatrix} \mathcal{Y}_1 \\ \mathcal{Y}_2 \end{bmatrix}$, where $\mathcal{X}_1$, $\mathcal{X}_2$, $\mathcal{Y}_1$ and $\mathcal{Y}_2$ all have the same dimensionality. Then, we

form $\mathcal{Z} = \begin{bmatrix} \mathcal{X}_1 \\ \mathcal{X}_2 + \mathcal{Y}_1 \\ \mathcal{Y}_2 \end{bmatrix}$.

3. Sum dataset: In this dataset, all of the features of $\mathcal{X}$ and $\mathcal{Y}$ overlap, i.e., $\mathcal{Z} = \mathcal{X} + \mathcal{Y}$.

In our experiments the dimensionality of $\mathcal{X}$ and $\mathcal{Y}$ was set to 30 and there were 3000 data points. We label each x_i and y_j according to the Gaussian from which they were sampled. Thus, each z is associated with two true-labels. Both our methods produce two disparate clusterings, and we associate each clustering with a unique true-labeling and report NMI with respect to that true-labeling. We use the same procedure for CVQ and MCVQ. For k-means and NNMA[2], which produce just one clustering, we report the NMI of the clustering with respect to the true-labellings.

Figure 8.3 compares the NMI achieved by various methods on the Concatenated dataset. It can be seen from the figure that Conv-EM and Dec-kmeans outperform k-means and NNMA for both the clusterings, achieving an average improvement of $50 - 60\%$ in NMI. Similarly, both Conv-EM and Dec-kmeans achieve significantly higher NMI than CVQ. Note that the Concatenated dataset satisfies MCVQ's assumption that each dimension of the data is generated from one of the two factors. This is empirically confirmed

[2]As NNMA is useful for the non-negative data only, we made the data non-negative by choosing the means sufficiently far away from origin.

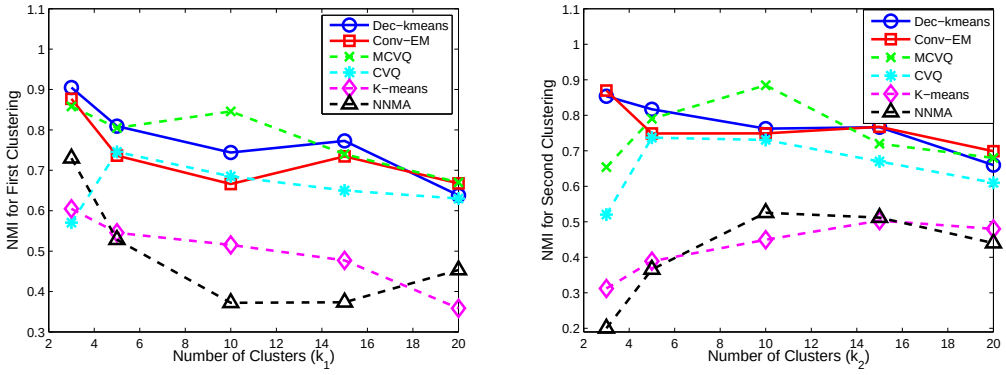


Figure 8.3: NMI achieved by various methods on the Concatenated Dataset. Top figure shows NMI for the first clustering and bottom figure shows NMI for the second clustering. Overall, Dec-kmeans achieves the highest NMI, while MCVQ also performs well on this dataset.

by the results, as MCVQ not only outperforms CVQ but also performs competitively with Conv-EM and Dec-kmeans.

Figure 8.4 compares the NMI for various methods on the Overlap dataset. Clearly, Conv-EM and Dec-kmeans perform better than both CVQ and MCVQ. Note that when the number of clusters is small, NMI of MCVQ with respect to both the clusterings drops to around 0.6. This is probably because the overlap dataset does not satisfy the model assumptions of MCVQ.

Figure 8.5 shows the NMI achieved by various methods on the Sum dataset. For this dataset also, both Conv-EM and Dec-kmeans perform comparably and both the methods achieve significantly higher NMI than other methods. Interestingly, NMI for MCVQ is even lower than the single-clustering algorithms (k-means and NNMA). This could be because the modeling assumption of MCVQ – each dimension in the data is generated by exactly one

factor – is completely violated in the Sum dataset. Note that, although CVQ is designed to model the Sum datasets, it performs poorly compared to Conv-EM and Dec-kmeans. This trend can be attributed to the fact that due to the lack of regularization CVQ selects one of the many possible solutions to its optimization problem, which may or may not correspond to good disparate clusterings.

Also note that Conv-EM does not perform significantly better than Dec-kmeans, even though the datasets fit the Conv-EM model well. This is probably because of the non-convex nature of the optimization problem for the M-step in Conv-EM, due to which which the maximum likelihood estimation gets stuck in a local minimum.

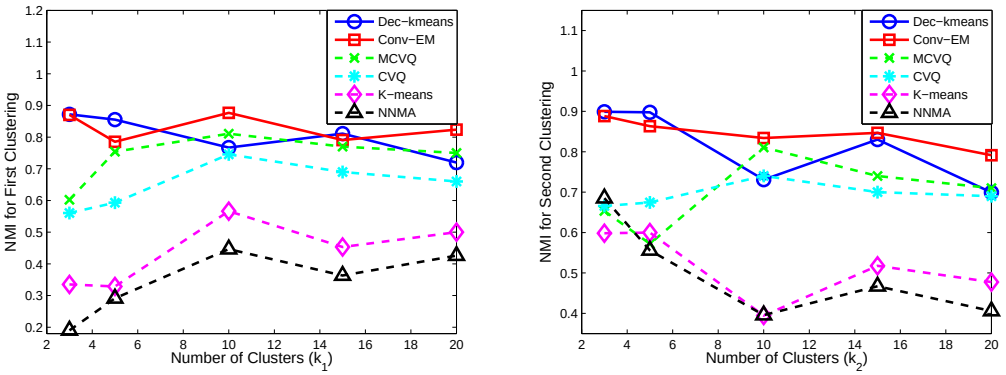


Figure 8.4: NMI achieved by various methods on the Overlap Dataset. Top figure shows NMI for the first clustering and bottom figure shows NMI for the second clustering. Dec-kmeans and Conv-EM achieves similar NMI. Both achieve higher NMI than MCVQ or CVQ.

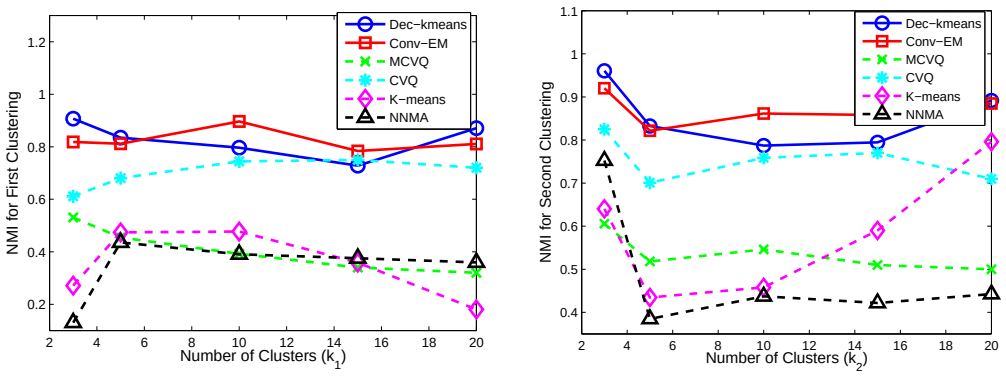


Figure 8.5: NMI achieved by various methods on the Sum Dataset. Top figure shows NMI for the first clustering and bottom figure shows NMI for the second clustering. Dec-kmeans and Conv-EM achieves similar NMI. NMI achieved by MCVQ is very low.

8.3.3 Real-World Datasets

8.3.3.1 Music Dataset:

The music dataset is a collection of 270 documents, with each document being a review of a classical music piece taken from *amazon.com*. Each music piece is composed by one of Beethoven, Mozart or Mendelssohn and is in one of symphony, sonata or concerto forms. Thus, the documents can be clustered based on the composer or the genre of the musical piece. For the experiments a term-document matrix was formed with dimensionality 258 after stop word removal and stemming.

Table 8.1 shows that although all the methods are able to recover the true clustering for composer, most of the algorithms perform poorly for the genre based clustering. In particular, k-means and NNMA perform very poorly for the clustering based on genre as they produce just one clustering which

Method\Type	Composer	Genre
NNMA	1.00	0.40
k-means	0.89	0.41
Feature Removal	0.97	0.64
Orthogonal Projection	0.99	0.66
CVQ	0.97	0.57
MCVQ	0.91	0.53
Conv-EM	1.00	0.65
Dec-kmeans	**1.00**	**0.69**

Table 8.1: Accuracy achieved by various methods on the Music dataset, which is a collection of text documents. Dec-kmeans performs the best on this dataset. CVQ and MCVQ perform very poorly compared to Conv-EM

has high NMI with the clustering based on composers. Note that in this dataset, the sets of features (words) that determine clustering with respect to composer and genre respectively are almost disjoint. Hence, methods like Feature Removal and Orthogonal Projection, which try to identify disjoint sets of features for the clusterings work fairly well. But, both MCVQ and CVQ algorithm achieve very low accuracy as they do not try to find *decorrelated* clusterings. Both our methods outperform the baseline algorithms.

8.3.3.2 Portrait Dataset:

The Portrait dataset consists of 324 images obtained from Yale Face Dataset B [33]. Each image in the dataset is a portrait of one of three people, in one of three poses in different backgrounds. The dimensionality of each image is 64×64. As a first step we reduce the dimensionality of the data to 300

by using principal component analysis. As in the music dataset, the current dataset can be clustered in two natural ways - by the person in the picture or the pose. Table 8.2 shows that both k-means and NNMA perform poorly with respect to both the clusterings. This shows that in the datasets where there is more than one natural clustering, traditional clustering algorithms could fail to find even one good clustering. Hence, it can be beneficial to use alternative clustering methods even if one is interested in obtaining a single clustering.

Our hypothesis is that unlike the music dataset, there are no dominant features for any of the clusterings in this dataset. This hypothesis can be justified by observing the poor accuracies of methods like Feature Removal and Orthogonal Projection. Conv-EM outperforms baseline algorithms CVQ and MCVQ significantly, but interestingly Dec-kmeans achieves an even higher accuracy of 84% and 78% for the two clusterings.

8.4 Summary

We formulated the unsupervised distance learning problem as that of learning disparate clusterings of the data. We provided two novel approaches to simultaneously uncover all the dominant clusterings in the data. Our decorrelated k-means approach is a discriminative approach based on the k-means clustering algorithm. We introduced a new regularization to uncover decorrelated clusterings of the data and provided theoretical justification for the same. Our sum of parts approach is a generative approach and leads to the

Method\Type	Person	Pose
NNMA	0.51	0.49
k-means	0.66	0.56
Feature Removal	0.56	0.48
Orthogonal Projection	0.66	0.70
CVQ	0.53	0.51
MCVQ	0.64	0.51
Conv-EM	0.69	0.72
Dec-kmeans	**0.84**	**0.78**

Table 8.2: Accuracy achieved by various methods on the Portrait dataset, which is a collection of images. Dec-kmeans outperforms all other methods by a significant margin. Conv-EM achieves better accuracy than all other methods, especially CVQ and MCVQ.

problem of learning a convolution of mixture models. For the special case of spherical Gaussians, we provided a generalized EM algorithm and added a regularization to address the identifiability issue. We evaluated our methods on two real-world data sets - a music data set from the text mining domain, and a portrait data set from the computer vision domain. Our methods achieved a substantially higher accuracy than existing factorial learning as well as traditional clustering algorithms.

Chapter 9

Conclusions and Future Directions

In this thesis, we studied the important problems of metric and kernel learning. In Chapters 3, 4, and 5, we primarily focused on various Mahalanobis distance/kernel learning problems. In Chapters 6, 7, and 8 we considered a few other models for distance/kernel learning.

First, we introduced a generic framework for metric learning that generalizes almost all the existing convex metric learning formulations. We specified conditions under which our framework learns a metric efficiently. We also extended our framework to handle high-dimensional data by restricting the number of parameters involved, leading to an efficient kernel learning framework.

Metric learning owes its wide applicability to its ability to significantly improve the accuracy of nearest neighbor retrieval. Most of the real-world applications involve huge databases, and hence, fast nearest neighbor search is fundamental to the success of any metric learning method. For the problem of fast similarity search, we developed a method to learn randomized hash functions which enables sub-linear time similarity search based on the learned metric. We provided efficiently computable hash functions that can be used

with locality sensitive hashing to retrieve $(1+\epsilon)$-nearest neighbors (according to the learned distance function) by searching just $O(N^{1/(1+\epsilon)})$ examples. As our formulation is generic, our method can be used with a variety of metric learning formulations and for a wide range of similarity search problems. Applications include fast pose retrieval, fast object recognition using k-NN and patch matching for fast 3-D reconstruction.

Additionally, we studied more structured distance learning problems that do not fall directly under the generic metric/kernel learning framework we introduced in Chapter 3.

We considered the online metric learning problem which is particularly useful in many web-based applications, e.g., image tagging and search. We developed an online metric learning algorithm together with a method to perform online updates to fast similarity search structures, and demonstrated their applicability and advantages on a variety of data sets. Our online learner offers improved reliability over state-of-the-art methods in terms of regret bounds and empirical performance.

Next, we studied two approaches to kernel learning problems where the amount of available information is small and the intrinsic structure of data needs to be exploited. In the first approach we assumed that the data lies on a low-dimensional non-linear manifold and we proposed a kernel learning algorithm that simultaneously models the intrinsic geometry of the data while incorporating the provided side-information. Our second approach assumed that the data lies in a low-dimensional linear subspace. This led to the general

rank minimization problem – a problem of immense interest in numerous research areas. For this problem, we provided an efficient algorithm with provable approximation guarantees. Furthermore, we demonstrated empirically that our approach leads to accurate and low-rank kernels.

Finally, we formulated the problem of unsupervised distance learning as that of uncovering disparate clusterings from the data in a completely unsupervised setting. We proposed two novel approaches for the problem - a decorrelated k-means approach and a sum of parts approach. In the first approach, we introduced a new regularization for k-means to uncover decorrelated clusterings and provided theoretical justification for it. The sum of parts approach lead us to the interesting problem of learning a convolution of mixture models and we presented a regularized EM algorithm for learning a convolution of mixtures of spherical Gaussians. We demonstrated the effectiveness and robustness of our algorithms on synthetic and real-world datasets. On each of these datasets, we significantly improved upon the accuracy achieved by existing factorial learning methods as well as traditional clustering algorithms like k-means and NNMA.

In summary, this thesis demonstrates the importance of the distance/kernel learning problem and its numerous applications. This thesis also validates the power and significance of Mahalanobis metric/kernel functions as distance functions. We show that these functions admit simple and elegant problem formulations, are amenable to theoretical analysis, and can be easily adapted to a large number of contrasting problem scenarios.

Some future directions for the research discussed in this thesis include:

1. **Generalization Bounds**: We introduced regularization in our metric learning framework (see Section 3.1) to avoid over-fitting to the available side-information and to have good generalization error. In Section 5.1.2, we proved regret bounds (or equivalently generalization bounds) for LogDet divergence based regularization and pairwise distance constraints. A similar analysis of our general metric learning framework should be of interest and is expected to provide additional insight into the selection of an appropriate regularization function.

2. **Kernelized Hashing**: In Chapter 4, we introduced a method for efficiently computing hash functions for sparse high-dimensional feature representations of data objects. Recently, [66] introduced a heuristic to enable hashing for any Hilbert space that does not require explicit representation of the feature vectors. It would be of interest to further study their construction and provide theoretical bounds on the number of samples required to compute the hash function.

3. **Matrix Completion Methods for Low-rank Kernel Learning**: The low-rank kernel learning problem (7.7) introduced in Section 7.3.1 can be reduced to a matrix completion problem. Recently, numerous methods have been proposed for the matrix completion problem [61, 19, 79] and it would be interesting to study these methods in the context of low-rank kernel learning.

4. **Novel Metric Learning Problems**: As shown in this thesis, distance learning problems have widespread applicability and a large number of metric learning problems specific to particular contexts/applications are yet unexplored. Examples of such problems include multiple-instance metric learning, local metric learning, and metric learning by combination of a few provided baseline metrics.

References

[1] Hao Zhang 0002, Alexander C. Berg, Michael Maire, and Jitendra Malik. Svm-knn: Discriminative nearest neighbor classification for visual category recognition. In *Proccedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2126–2136, 2006.

[2] Edoardo Amaldi and Viggo Kann. On the approximability of minimizing non-zero variables or unsatisfied relations in linear systems. *Theoretical Computer Science*, 209:237–260, 1998.

[3] Sanjeev Arora, Elad Hazan, and Satyen Kale. Fast algorithms for approximate semidefinite programming using the multiplicative weights update method. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 339–348, 2005.

[4] Sanjeev Arora, Elad Hazan, and Satyen Kale. Multiplicative weights method: a meta-algorithm and its applications. http://www.cs.princeton. edu/˜arora/pubs/MWsurvey.pdf, 2005.

[5] Sanjeev Arora and Satyen Kale. A combinatorial, primal-dual approach to semidefinite programs. In *Proceedings of the ACM Symposium on Theory of Computing (STOC)*, pages 227–236, 2007.

[6] Vassilis Athitsos, Jonathan Alon, Stan Sclaroff, and George Kollios. Boostmap: An embedding method for efficient nearest neighbor retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 30(1):89–104, 2008.

[7] Vassilis Athitsos and Stan Sclaroff. Estimating 3d hand pose from a cluttered image. In *Proccedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 432–442, 2003.

[8] Francis R. Bach and Michael I. Jordan. Predictive low-rank decomposition for kernel methods. In *Proccedings of the International Conference on Machine Learning (ICML)*, pages 33–40, 2005.

[9] Eric Bae and James Bailey. Coala: A novel approach for the extraction of an alternate clustering of high quality and high dissimilarity. In *Proceedings of the International Conference on Data Mining (ICDM)*, pages 53–62, 2006.

[10] Aharon Bar-Hillel, Tomer Hertz, Noam Shental, and Daphna Weinshall. Learning a mahalanobis metric from equivalence constraints. *Journal of Machine Learning Research (JMLR)*, 6:937–965, 2005.

[11] Jeffrey S. Beis and David G. Lowe. Shape indexing using approximate nearest-neighbour search in high-dimensional spaces. In *Proccedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1000–1006, 1997.

[12] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003.

[13] Yoshua Bengio, Olivier Delalleau, Nicolas Le Roux, Jean-Francois Paiement, Pascal Vincent, and Marie Ouimet. Learning eigenfunc-

tions links spectral embedding and kernel PCA. *Neural Computation*, 16(10):2197–2219, 2004.

[14] Mikhail Bilenko, Sugato Basu, and Raymond J. Mooney. Integrating constraints and metric learning in semi-supervised clustering. In *Proccedings of the International Conference on Machine Learning (ICML)*, 2004.

[15] Christopher M Bishop. *Neural networks for pattern recognition*. Oxford University Press, Oxford, UK, 1996.

[16] Thomas Blumensath and Mike E. Davies. Iterative hard thresholding for compressed sensing. *Applied and Computational Harmonic Analysis*, 27(3):265 – 274, 2009.

[17] Anna Bosch, Andrew Zisserman, and Xavier Muñoz. Representing shape with a spatial pyramid kernel. In *Proceedings of the ACM International Conference on Image and Video Retrieval (CIVR)*, pages 401–408, 2007.

[18] Olivier Bousquet, Olivier Chapelle, and Matthias Hein. Measure based regularization. In *Proccedings of Advances in Neural Information Processing Systems (NIPS)*, 2003.

[19] Emmanuel J. Candès and Benjamin Recht. Exact matrix completion via convex optimization, 2008.

[20] Emmanuel J. Candès and Terence Tao. Decoding by linear programming. *IEEE Transactions on Information Theory*, 51(12):4203–4215, 2005.

[21] Moses Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings of the ACM Symposium on Theory of Computing*

(STOC), pages 380–388, 2002.

[22] Ian Davidson, S. S. Ravi, and Martin Ester. Efficient incremental constrained clustering. In *Proceedings of the ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD)*, pages 240–249, 2007.

[23] Jason V. Davis and Inderjit S. Dhillon. Structured metric learning for high dimensional problems. In *Proceedings of the ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD)*, pages 195–203, 2008.

[24] Jason V. Davis, Brian Kulis, Prateek Jain, Suvrit Sra, and Inderjit S. Dhillon. Information-theoretic metric learning. In *Proccedings of the International Conference on Machine Learning (ICML)*, pages 209–216, 2007.

[25] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification*. Wiley-Interscience, November 2000.

[26] M. Fazel, H. Hindi, and S. Boyd. A rank minimization heuristic with application to minimum order system approximation. In *American Control Conference, Arlington, Virginia*, 2001.

[27] M. Fazel, H. Hindi, and S. Boyd. Log-det heuristic for matrix rank minimization with applications to hankel and euclidean distance matrices. In *American Control Conference*, 2003.

[28] J. Freidman, J. Bentley, and A. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software*, 3(3):209–226, September 1977.

[29] Andrea Frome, Yoram Singer, Fei Sha, and Jitendra Malik. Learning globally-consistent local distance functions for shape-based image retrieval and classification. In *Proccedings of the International Conference on Computer Vision (ICCV)*, pages 1–8, 2007.

[30] William R. Gaffey. A consistent estimator of a component of a convolution. *The Annals of Mathematical Statistics*, 30(1):198–205, Mar 1959.

[31] Rahul Garg and Rohit Khandekar. Gradient descent with sparsification: an iterative algorithm for sparse recovery with restricted isometry property. In *Proccedings of the International Conference on Machine Learning (ICML)*, 2009.

[32] Bogdan Georgescu, Ilan Shimshoni, and Peter Meer. Mean shift based clustering in high dimensions: A texture classification example. In *Proccedings of the International Conference on Computer Vision (ICCV)*, pages 456–463, 2003.

[33] Athinodoros S. Georghiades, Peter N. Belhumeur, and David J. Kriegman. From few to many: Illumination cone models for face recognition under variable lighting and pose. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 23(6):643–660, 2001.

[34] Zoubin Ghahramani. Factorial learning and the EM algorithm. In *Proccedings of Advances in Neural Information Processing Systems (NIPS)*, pages 617–624, 1994.

[35] Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Similarity search in

high dimensions via hashing. In *Proceedings of International Conference on Very Large Data Bases (VLDB)*, pages 518–529, 1999.

[36] Amir Globerson and Sam T. Roweis. Metric learning by collapsing classes. In *Proccedings of Advances in Neural Information Processing Systems (NIPS)*, 2005.

[37] Michel X. Goemans and David P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of ACM*, 42(6):1115–1145, 1995.

[38] Donald Goldfarb and Shiqian Ma. Convergence of fixed point continuation algorithms for matrix rank minimization, 2009.

[39] Gene H. Golub and Charles F. van Loan. *Matrix Computations*. Johns Hopkins Univ. Press, second edition, 1989.

[40] David Gondek and Thomas Hofmann. Non-redundant clustering with conditional ensembles. In *Proceedings of the ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD)*, pages 70–77, 2005.

[41] David Gondek, Shivakumar Vaithyanathan, and Ashutosh Garg. Clustering with model-level constraints. In *Proceedings of the SIAM Conference on Data Mining (SDM)*, 2005.

[42] Kristen Grauman and Trevor Darrell. The pyramid match kernel: Discriminative classification with sets of image features. In *Proccedings of the International Conference on Computer Vision (ICCV)*, pages 1458–1465, 2005.

[43] Kristen Grauman and Trevor Darrell. Pyramid match hashing: Sub-

linear time indexing over partial correspondences. In *Proccedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

[44] M. Groschel, Laszlo Lovasz, and Alexander Schrijver. *Geometric Algorithms and Combinatorial Optimization*. Springer-Verlag, 1988.

[45] Matthieu Guillaumin, Thomas Mensink, Jakob Verbeek, and Cordelia Schmid. Tagprop: Discriminative metric learning in nearest neighbor models for image auto-annotation. In *Proccedings of the International Conference on Computer Vision (ICCV)*, 2009.

[46] Jungwoo Ha, Christopher J. Rossbach, Jason V. Davis, Indrajit Roy, Hany E. Ramadan, Donald E. Porter, David L. Chen, and Emmett Witchel. Improved error reporting for software that uses black-box components. In *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation*, pages 101–111, 2007.

[47] Trevor Hastie, Robert Tibshirani, Michael B Eisen, Ash Alizadeh, Ronald Levy, Louis Staudt, Wing C Chan, David Botstein, and Patrick Brown. Gene shaving as a method for identifying distinct sets of genes with similar expression patterns. *Genome Biology*, 2000.

[48] Elad Hazan, Adam Kalai, Satyen Kale, and Amit Agarwal. Logarithmic regret algorithms for online convex optimization. In *Proceedings of the Conference on Computational Learning Theory*, pages 499–513, 2006.

[49] Nick J. Higham. *Functions of Matrices: Theory and Computation*. SIAM, 2008.

[50] Geoffrey E. Hinton and Richard S. Zemel. Autoencoders, minimum description length and Helmholtz free energy. In *Proccedings of Advances in Neural Information Processing Systems (NIPS)*, pages 3–10, 1993.

[51] Gang Hua, Matthew Brown, and Simon A. J. Winder. Discriminant embedding for local image descriptors. In *Proccedings of the International Conference on Computer Vision (ICCV)*, pages 1–8, 2007.

[52] Aapo Hyvärinen and Erkki Oja. Independent component analysis: algorithms and applications. *Neural Networks*, 13(4-5):411–430, 2000.

[53] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of the ACM Symposium on Theory of Computing (STOC)*, pages 604–613, 1998.

[54] Prateek Jain and Ashish Kapoor. Active learning for large multi-class problems. In *Proccedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[55] Prateek Jain, Brian Kulis, Jason V. Davis, and Inderjit S. Dhillon. Metric and kernel learning using a linear transformation, 2009.

[56] Prateek Jain, Brian Kulis, and Inderjit S. Dhillon. Generalized metric and kernel learning framework with applications to low-rank kernel function learning. *In preparation*, 2009.

[57] Prateek Jain, Brian Kulis, and Kristen Grauman. Fast image search for learned metrics. In *Proccedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[58] Prateek Jain, Raghu Meka, and Inderjit S. Dhillon. Simultaneous unsu-

pervised learning of disparate clusterings. In *Proceedings of the SIAM Conference on Data Mining (SDM)*, pages 858–869, 2008.

[59] Prateek Jain, Raghu Meka, and Inderjit S. Dhillon. Simultaneous unsupervised learning of disparate clusterings. *Statistical Analysis and Data Mining*, 1(3):195–210, 2008.

[60] Adam Tauman Kalai and Santosh Vempala. Efficient algorithms for online decision problems. *Journal on Computer and System Sciences (JCSS)*, 71(3):291–307, 2005.

[61] Raghunandan H. Keshavan, Sewoong Oh, and Andrea Montanari. Matrix completion from a few entries, 2009.

[62] Dongmin Kim, Suvrit Sra, and Inderjit S. Dhillon. Fast newton-type methods for the least squares nonnegative matrix approximation problem. In *Proceedings of the SIAM Conference on Data Mining (SDM)*, 2007.

[63] Jyrki Kivinen and Manfred K. Warmuth. Exponentiated gradient versus gradient descent for linear predictors. *Inf. Comput.*, 132(1):1–63, 1997.

[64] Risi Imre Kondor and John D. Lafferty. Diffusion kernels on graphs and other discrete input spaces. In *Proccedings of the International Conference on Machine Learning (ICML)*, pages 315–322, 2002.

[65] B. Kulis, S. Sra, and I. S. Dhillon. Convex perturbations for scalable semidefinite programming. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2009.

[66] Brian Kulis and Kristen Grauman. Kernelized locality-sensitive hashing

for scalable image search. In *Proccedings of the International Conference on Computer Vision (ICCV)*, 2009.

[67] Brian Kulis, Prateek Jain, and Kristen Grauman. Fast similarity search for learned metrics. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 31(12):2143–2157, 2009.

[68] Brian Kulis, Mátyás Sustik, and Inderjit S. Dhillon. Learning low-rank kernel matrices. In *Proccedings of the International Conference on Machine Learning (ICML)*, pages 505–512, 2006.

[69] John Lafferty and Guy Lebanon. Diffusion kernels on statistical manifolds. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 6:129–163, 2005.

[70] Gert R. G. Lanckriet, Nello Cristianini, Peter L. Bartlett, Laurent El Ghaoui, and Michael I. Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research (JMLR)*, 5:27–72, 2004.

[71] Daniel D. Lee and H. Sebastian Seung. Algorithms for non-negative matrix factorization. In *Proccedings of Advances in Neural Information Processing Systems (NIPS)*, pages 556–562, 2000.

[72] Kiryung Lee and Yoram Bresler. Admira: Atomic decomposition for minimum rank approximation, 2009.

[73] Haibin Ling and Stefano Soatto. Proximity distribution kernels for geometric context in category recognition. In *Proccedings of the International Conference on Computer Vision (ICCV)*, pages 1–8, 2007.

[74] Nick Littlestone and Manfred K. Warmuth. The weighted majority algorithm. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 256–261, 1989.

[75] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[76] Zhengdong Lu, Prateek Jain, and Inderjit S. Dhillon. Geometry-aware metric learning. In *Proccedings of the International Conference on Machine Learning (ICML)*, page 85, 2009.

[77] Jiri Matas, Ondrej Chum, Martin Urban, and Tomás Pajdla. Robust wide baseline stereo from maximally stable extremal regions. In *British Machine Vision Conference*, 2002.

[78] Raghu Meka, Prateek Jain, Constantine Caramanis, and Inderjit S. Dhillon. Rank minimization via online learning. In *Proccedings of the International Conference on Machine Learning (ICML)*, pages 656–663, 2008.

[79] Raghu Meka, Prateek Jain, and Inderjit S. Dhillon. Guaranteed rank minimization via singular value projection, 2009.

[80] Krystian Mikolajczyk and Cordelia Schmid. Scale & affine invariant interest point detectors. *International Journal of Computer Vision*, 60(1):63–86, 2004.

[81] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.

[82] David Nistér and Henrik Stewénius. Scalable recognition with a vo-

cabulary tree. In *Proccedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2161–2168, 2006.

[83] Cheng Soon Ong, Alexander J. Smola, and Robert C. Williamson. Learning the kernel with hyperkernels. *Journal of Machine Learning Research (JMLR)*, 6:1043–1071, 2005.

[84] Serge A. Plotkin, David B. Shmoys, and Éva Tardos. Fast approximation algorithms for fractional packing and covering problems. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 495–504, 1991.

[85] Benjamin Recht, Maryam Fazel, and Pablo A. Parrilo. Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization, 2007. Submitted to SIAM Review.

[86] R. Tyrell Rockafellar. *Convex Analysis*. Princeton University Press, 1970.

[87] David A. Ross and Richard S. Zemel. Learning parts-based representations of data. *Journal of Machine Learning Research (JMLR)*, 7:2369–2397, 2006.

[88] F. J. Samaniego and L. E. Jones. Maximum likelihood estimation for a class of multinomial distributions arising in realibility. *Journal of the Royal Statistical Society. Series B (Methodological)*, 43(1):46–52, 1981.

[89] Stanley L. Sclove and Garrett J. van Ryzin. Estimating the parameters of a convolution. *Journal of the Royal Statistical Society. Series B*

(Methodological), 31(1):181–191, 1969.

[90] Matthias Seeger. Cross-validation optimization for large scale hierarchical classification kernel methods. In *Proccedings of Advances in Neural Information Processing Systems (NIPS)*, pages 1233–1240, 2006.

[91] G. Shakhnarovich, T. Darrell, and P. Indyk, editors. *Nearest-Neighbor Methods in Learning and Vision: Theory and Practice*. MIT Press, 2006.

[92] Gregory Shakhnarovich, Paul A. Viola, and Trevor Darrell. Fast pose estimation with parameter-sensitive hashing. In *Proccedings of the International Conference on Computer Vision (ICCV)*, pages 750–759, 2003.

[93] Shai Shalev-Shwartz, Yoram Singer, and Andrew Y. Ng. Online and batch learning of pseudo-metrics. In *Proccedings of the International Conference on Machine Learning (ICML)*, 2004.

[94] Vikas Sindhwani, Partha Niyogi, and Mikhail Belkin. Beyond the point cloud: from transductive to semi-supervised learning. In *Proccedings of the International Conference on Machine Learning (ICML)*, pages 824–831, 2005.

[95] Josef Sivic and Andrew Zisserman. Video data mining using configurations of viewpoint invariant regions. In *Proccedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 488–495, 2004.

[96] Noah Snavely, Steven M. Seitz, and Richard Szeliski. Photo tourism: exploring photo collections in 3d. *ACM Transactions on Graphics*, 25(3):835–846, 2006.

[97] Le Song, Alex Smola, Karsten M. Borgwardt, and Arthur Gretton. Colored maximum variance unfolding. In *Proccedings of Advances in Neural Information Processing Systems (NIPS)*, pages 1385–1392, 2007.

[98] Alexander Strehl and Joydeep Ghosh. Cluster ensembles — a knowledge reuse framework for combining multiple partitions. *Journal of Machine Learning Research (JMLR)*, 3:583–617, 2002.

[99] Leonid Taycher, David Demirdjian, Trevor Darrell, and Gregory Shakhnarovich. Conditional random people: Tracking humans with crfs and grid filters. In *Proccedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 222–229, 2006.

[100] Joshua B. Tenenbaum and William T. Freeman. Separating style and content with bilinear models. *Neural Computation*, 12(6):1247–1283, 2000.

[101] Joel A. Tropp and Deanna Needell. Cosamp: Iterative signal recovery from incomplete and inaccurate samples, 2008.

[102] Koji Tsuda, Gunnar Rätsch, and Manfred K. Warmuth. Matrix exponentiated gradient updates for on-line learning and bregman projection. *Journal of Machine Learning Research (JMLR)*, 6:995–1018, 2005.

[103] Jeffrey K. Uhlmann. Satisfying general proximity/similarity queries with metric trees. *Information Processing Letters*, 40(4):175–179, 1991.

[104] Manik Varma and Debajyoti Ray. Learning the discriminative power-invariance trade-off. In *Proccedings of the International Conference on*

Computer Vision (ICCV), pages 1–8, 2007.

[105] Kiri Wagstaff and Claire Cardie. Clustering with instance-level constraints. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, page 1097, 2000.

[106] Kiri Wagstaff, Claire Cardie, Seth Rogers, and Stefan Schrödl. Constrained k-means clustering with background knowledge. In *Proccedings of the International Conference on Machine Learning (ICML)*, pages 577–584, 2001.

[107] Manfred K. Warmuth and Dima Kuzmin. Randomized pca algorithms with regret bounds that are logarithmic in the dimension. In *Proccedings of Advances in Neural Information Processing Systems (NIPS)*, pages 1481–1488, 2006.

[108] Kilian Q. Weinberger, John Blitzer, and Lawrence K. Saul. Distance metric learning for large margin nearest neighbor classification. In *Proccedings of Advances in Neural Information Processing Systems (NIPS)*, 2005.

[109] Kilian Q. Weinberger, Fei Sha, and Lawrence K. Saul. Learning a kernel matrix for nonlinear dimensionality reduction. In *Proccedings of the International Conference on Machine Learning (ICML)*, 2004.

[110] Eric P. Xing, Andrew Y. Ng, Michael I. Jordan, and Stuart J. Russell. Distance metric learning with application to clustering with side-information. In *Proccedings of Advances in Neural Information Processing Systems (NIPS)*, pages 505–512, 2002.

[111] Willard I. Zangwill. *Nonlinear Programming: A Unified Approach*. Englewood Cliffs: Prentice-Hall, 1969.

[112] Xiaojin Zhu, Jaz Kandola, Zoubin Ghahramani, and John Lafferty. Nonparametric transforms of graph kernels for semi-supervised learning. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Proccedings of Advances in Neural Information Processing Systems (NIPS)*, volume 17, pages 1641–1648, 2005.

[113] Martin Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proccedings of the International Conference on Machine Learning (ICML)*, pages 928–936, 2003.

Vita

Prateek Jain was born in Nagaur, Rajasthan, India, the son of Late Ren Manjusha Jain and Nathmal Jain. He spent most of his childhood in five different cities in the state of Rajasthan. He graduated from Hind Zinc High School, Chittorgarh, Rajasthan in 1999. He obtained his B.Tech. degree from Computer Science and Engineering Department, IIT Kanpur in 2004. He then spent one year as a project trainee with IBM-India Research Lab, New Delhi and later joined the Computer Science Department at University of Texas at Austin as a Phd Student in the Fall semester of 2005. Prateek obtained his M.A. degree from the Computer Science Department, UT Austin in 2008. During the course of his tenure as a Phd student, Prateek won three paper awards and was nominated for a best student paper award. He is also a recipient of an MCD fellowship and several travel-grants from different international conferences. After his doctorate, Prateek plans to join Microsoft Research Lab, Bangalore, India as an associate researcher.

Permanent address: Mr. Kamal Kishore Jain,
C/O Rajesh Drug Store,
Sadar Bazar, Nagaur, India (Raj.)

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.