The Dissertation Committee for Michael Minyi Zhang
certifies that this is the approved version of the following dissertation:

Scalable Inference for Bayesian Non-parametrics

Committee:

Sinead A. Williamson, Supervisor

Peter Müller

James G. Scott

Eric P. Xing

Scalable Inference for Bayesian Non-parametrics

by

Michael Minyi Zhang,

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN

August 2018

Dedicated to my parents.

Acknowledgments

Over the course of my doctoral studies, I have had the pleasure to meet many intelligent people who have done a tremendous amount of work to help me achieve my goal of obtaining a PhD in statistics. First and foremost, I would like to thank my advisor, Sinead Williamson, who has been incredibly supportive and helpful in my path through graduate school since my first day of the PhD program and, beyond that, an excellent researcher and collaborator. Next, I want to thank Lizhen Lin who has essentially been a second mentor to me for her kindness and generosity in helping develop my early research career. I would also like to acknowledge funding from the National Science Foundation (NSF 1447721), for supporting my advisor and me.

Additionally, I have had the good luck and benefit of working with several researchers who have contributed a great deal to the work I have developed over the last four years, namely Fernando Pérez-Cruz, Henry Lam, Avinava Dubey, Pablo Olmos, Howard Huang, Daniele Schiavazzi, Paul Damien, Zach Phillips, and Ulrich Müller. Furthermore, I would like to thank Stephen Walker, Purnamrita Sarkar and, again, Lizhen Lin for their generosity in helping me prepare for the preliminary exam. Next, I want to thank the other members of my committee–Peter Müller, James Scott and Eric Xing for taking the time to critique my thesis. I would also like to thank Vicki Keller and all the staff members of the UT Department of Statistics for their wonderful

work supporting the students and faculty of this department. I also want to thanky my high school statistics teacher, Victoria Stewart, for putting me on the initial path towards my career in statistics. Lastly, I would like to thank my family for all of their love and support.

Scalable Inference for Bayesian Non-parametrics

Publication No. _____

Michael Minyi Zhang, Ph.D.
The University of Texas at Austin, 2018

Supervisor: Sinead A. Williamson

Bayesian non-parametric models, despite their theoretical elegance, face a serious computational burden that prevents their use in serious "big data" scenarios. Furthermore, we cannot expect the data in "big data" to exist solely on one processor, so we must have parallel algorithms that are valid Bayesian inference samplers. However, inherent dependencies in Bayesian non-parametric models make this task very difficult. Instead, we must either construct good approximations or develop clever reformulations of our models so that we perform inference with provably accurate results. This thesis will discuss four methods developed to parallelize inference in the Bayesian and Bayesian non-parametric setting.

Table of Contents

Acknowledgments **v**

Abstract **vii**

List of Tables **xi**

List of Figures **xii**

Chapter 1. Introduction **1**

Chapter 2. Background **4**

2.1 Bayesian Models 4
2.2 Bayesian Inference 5
2.2.1 Monte Carlo Estimation and Importance Sampling . . . 5
2.2.2 Markov Chain Monte Carlo 7
2.3 Bayesian Non-parametric Models 10
2.3.1 The Dirichlet Process 11
2.3.2 Example: The Infinite Mixture Model 13
2.3.3 The Beta-Bernoulli Process 14
2.3.4 Example: The Infinite Latent Factor Model 19
2.3.5 Gaussian Processes 21
2.3.6 Example: Bayesian Non-linear Regression 22
2.4 Bayesian Non-parametric Inference 24
2.4.1 Discrete Models 24
2.4.2 Continuous Models 26
2.5 Distributed Inference 28
2.5.1 Embarrassingly Parallel MCMC 28
2.5.2 Exact Parallel MCMC 31
2.6 Further Reading 34

Chapter 3. Distributed Inference in Bayesian Nonparametric Models **36**
3.1 Completely Random Measures 36
3.2 Inference Approaches 39
3.3 Distributed Inference Methods 40
3.4 Hybrid Algorithms for Distributed Inference 42
3.4.1 Distributed Hybrid Inference in the Dirichlet Process . . 45
3.4.2 Distributed Hybrid Inference in the Indian Buffet Process 48
3.5 Experimental Evaluation 50
3.5.1 Limitations of an Entirely Uncollapsed Approach 51
3.5.2 Experimental Evaluation: Dirichlet Process 53
3.5.3 Experimental Evaluation: Indian Buffet Process 55
3.6 Discussion 59

Chapter 4. Accelerated Inference for Latent Variable Models **63**
4.1 Related Work 65
4.1.1 Dirichlet Process Inference 66
4.1.2 Inferential Problems 67
4.2 Method 69
4.3 Experimental Results 72
4.3.1 Location Clustering Example 72
4.3.2 Image Data Sets 73
4.4 Discussion 76

Chapter 5. Robust and Parallel Bayesian Model Selection **83**
5.1 Bayesian Model Selection 84
5.2 Divide-and-Conquer and Robust Bayesian Model Selection . . 86
5.3 Improved Concentration and Robustness 91
5.4 Simulations and Data Analysis 106
5.5 Discussion 112

Chapter 6. Embarrassingly Parallel Inference for Gaussian Processes **122**

6.1 Related Work 124

6.1.1 Reducing the Cost of Matrix Inversion With Covariance Approximations 125

6.1.2 Distributed Inference for Gaussian Processes 128

6.1.3 Fast Bayesian Inference via Stochastic Approximations . 129

6.2 Embarrassingly Parallel Inference with the "Importance Gaussian Process Sampler" 130

6.2.1 Minibatched Importance Samples 134

6.3 Experimental Evaluation 135

6.3.1 Evaluation on Synthetic Data 135

6.3.1.1 Comparison with Competing Methods 135

6.3.1.2 The Importance of Importance Sampling 141

6.3.2 Evaluation on Real Data 147

6.3.2.1 Sensitivity to Model Settings 147

6.3.2.2 Comparison with Competing Methods 150

6.3.2.3 Applications Beyond Regression 151

6.4 Discussion 152

Chapter 7. Conclusion **154**

Bibliography **155**

List of Tables

6.1 Comparison of inference complexity. N is the number of data points, K is the number of experts or local GPs, and $M = N/K$ is the number of inducing points. For the Monte Carlo based methods, J is the number of MCMC iterations or importance samples. 135

6.2 Test set performance on synthetic datasets 137

6.3 Test set log likelihood and MSE for various weighting schemes. Standard errors are in parentheses 146

6.4 Test set log likelihood and MSE for two different covariate partitioning schemes. Standard errors are in parentheses. 146

6.5 Test set log likelihood and AUC on three classification datasets. 151

List of Figures

2.1 Draws from a stick breaking Dirichlet process. 13
2.2 Draws from a stick breaking beta process. 18
2.3 Draws from a Chinese restaurant process and Indian buffet process. 20
2.4 Draws from a Gaussian process with an RBF kernel. 22

3.1 Comparison of F1 scores over iteration for the collapsed, uncollapsed and hybrid samplers 53
3.2 F1 score for test set synthetic data. 54
3.3 F1 score over iterations for synthetic data set with small separation 55
3.4 F1 score over iterations for synthetic data set with large separation 56
3.5 Top: The true features present in the synthetic data set. Bottom: Examples of observations in the synthetic data set. . . . 57
3.6 Test set log likelihood on synthetic data without warm-start initialization. 58
3.7 Number of features over iterations for synthetic data without warm-start initialization. 59
3.8 Test set log likelihood on synthetic data with warm-start initialization. 60
3.9 Number of features over iterations for synthetic data with warm-start initialization. 60
3.10 Test set log likelihood on synthetic data with all processors introducing features. 61
3.11 Number of features over iterations for synthetic data with all processors introducing features. 61

4.1 Test set predictive log likelihood vs. log time (seconds) and number of features vs. log time. 73
4.2 Clustering results for each sampling method. Different colors represent different clusters. 74

4.3 Test set predictive log likelihood vs. log time (seconds) and number of features vs. log time. 77
4.4 Popularity of each instantiated feature. 77
4.5 Test set predictive log likelihood vs. log time (seconds) and number of features vs. log time. 78
4.6 Popularity of each instantiated feature. Uncollapsed sampler put all observations into one cluster. 78
4.7 Yale faces features (left) and MNIST features (right) obtained via accelerated sampling, sorted in descending order of popularity. 79
4.8 Yale faces features (left) and MNIST features (right) obtained via uncollapsed sampling, sorted in descending order of popularity. 79
4.9 First 10 Yale faces features (top) and MNIST features (bottom) obtained via collapsed sampling, sorted in descending order of popularity. 80
4.10 Yale faces features (left) and MNIST features (right) obtained via variational inference, sorted in descending order of popularity. 80
4.11 First 10 MNIST features obtained via collapsed sampling initialized randomly over 100 partitions (top) or initialized with 100 clusters in K-means (bottom), sorted in descending order of popularity. 81
4.12 First 10 MNIST features obtained via uncollapsed sampling initialized randomly over 100 partitions (top) or initialized with 100 clusters in K-means (bottom), sorted in descending order of popularity. 81
4.13 First 10 Yale faces features obtained via collapsed sampling initialized randomly over 100 partitions (top) or initialized with 100 clusters in K-means (bottom), sorted in descending order of popularity. 81
4.14 First 10 Yale faces features obtained via uncollapsed sampling initialized randomly over 100 partitions (top) or initialized with 100 clusters in K-means (bottom), sorted in descending order of popularity. 82

5.1 Contamination test. 107
5.2 Magnitude of outlier test. 108
5.3 Testing empirical coverage of predictive value. 110
5.4 Posterior regression parameter coverage test results, estimate combination. 113

5.5 Posterior regression parameter coverage test results, model combination. 114
5.6 Contamination test, BMA and median model selection 115
5.7 Contamination test, AIC and BIC 116
5.8 Contamination test, spike and slab 116
5.9 Coverage test. 117
5.10 Magnitude test, AIC and BIC. 118
5.11 Magnitude test, BMA and median model selection. 119
5.12 Magnitude test, spike and slab. 119
5.13 Synthetic big data results. 120
5.14 Diabetes test data results. 121

6.1 Posterior mean and 95% predictive intervals on Synthetic 1 (stationary, long length scale). 136
6.2 Posterior mean and 95% predictive intervals on Synthetic 2 (stationary, short length scale). 136
6.3 Posterior mean and 95% predictive intervals on Synthetic 3 (non-stationary). 138
6.4 Evaluation on synthetic data of the effect of the number of samples J on the test set log likelihood of IGPS (with 95% confidence intervals), for various values of B and K. 142
6.5 Evaluation on synthetic data of the effect of the number of samples J on the test set MSE of IGPS (with 95% confidence intervals), for various values of B and K. 143
6.6 Comparison of different covariance matrices for a latent function with a long lengthscale (left) and a short lengthscale (right). "True Function" is the latent function being modeled. "Dense" is the dense covariance matrix $\Sigma(X, X')$. "Averaged" is an importance averaging of several block diagonal partitioned covariance matrices. "Partitioned" is one instance of a block diagonal covariance matrix. 145
6.7 Evaluation on the AIRS dataset of the effect of B, K, and J on the test set log likelihood of IGPS (with 95% confidence intervals). 148
6.8 Evaluation on the AIRS dataset of the effect of B, K, and J on the test set MSE of IGPS (with 95% confidence intervals). . . 149
6.9 Comparison of IGPS and SVI on AIRS dataset for various K, with $J = 100$ and $B = 1000$. SVI parameters chosen to have equivalent computational cost. 150

6.10 Binary classification task: label probabilities obtained using the full GP and the IGPS. 152

Chapter 1

Introduction

Bayesian non-parametric models are elegant ways to discover underlying latent features within a data set. These types of models have proven useful for learning functions [76], clustering data [4], topic modeling [90], or learning low dimensional representations of data [37] without having to enforce strong assumptions on the model (like the parametric form of the function, or the number of clusters with which to model data). This flexibility becomes critical as the data we try to model becomes more complex. For applications of machine learning to audio, images, videos, and other complicated objects we need methods that can accommodate and adapt model complexity to the data of interest.

But as the complexity of the data we are trying to model increases, so too does the scale of the problem at hand, both in terms of the number of observations ("big N", examples of which include topic models of all entries on Wikipedia or modeling user behavior on Facebook or Netflix) and the dimension of the model fit to the data ("big P", examples of which include bioinformatic applications where there may be tens of thousands of biomarkers for a particular observation or in cases where we may expect the number of

parameters to grow with the size of the data, as is the case in the Bayesian non-parametric setting). This "big data" problem therefore requires that our models not only can handle difficult problems but also must handle massive volumes of data as well.

However appealing Bayesian non-parametric models are to statistical practitioners, inference for such models is difficult. Inference in the Bayesian paradigm faces different challenges from the frequentist approach. Whereas under the frequentist ideology we have to optimize a difficult objective function, in Bayesian statistics we have to integrate difficult functions for which closed from expressions usually do not exist. This procedure becomes even more complicated in the non-parametric setting with infinite dimension priors.

Parallelization, if possible, eases some of the computational burden because we can divide one large inference problem into several smaller ones. A common form of parallelization is store the entire data set and model on a single machine and distribute the computation to available processors ("computational parallelization") [82]. Under the "big data" scenario, we assume the entire data set of interest cannot exist on one machine but instead divided over several machines. Therefore, we need Bayesian inference techniques that are not only fast, but can be parallelizable across multiple machines by physically dividing the data and the model across different machines and perform inference on the divided components ("memory parallelization").

Constructing good parallel inference algorithms is difficult because the

algorithm should have theoretical guarantees for correct inference while minimizing expensive inter-processor communication. Satisfying both of these constraints is difficult. Naive distribution and combination of Bayesian inference leads to posterior distributions with excessively high variance [102]. Exact inference often will require constant processor communication which requires a huge amount of computational overhead [3]. In total, these problems limit the popularity Bayesian non-parametric methods for the statistical and machine learning practitioner.

The scope of this dissertation covers various scalable inference strategies to address the challenges of Bayesian modeling–particularly Bayesian non-parametric modeling–in the "big data" era. The structure of the dissertation is as follows: Chapter 2 will introduce Bayesian modeling and inference and then continue with a discussion of the Bayesian non-parametric paradigm. Chapter 3 details an exact parallel inference strategy for feature allocation in completely random measures (with demonstrations for the Dirichlet process and Indian buffet process). Chapter 4 builds on the idea from the previous chapter by discussing a problem in parallel inference not addressed by other works through a parallelizable approach for Bayesian non-parametric latent variable models and proposing new features in high-dimensions. Chapter 5 looks at the problem of parallelizing model selection in the Bayesian domain. Chapter 6 proposes an entirely embarrassingly parallel approach to Gaussian process inference. And finally, the dissertation concludes with Chapter 7 and a discussion of future work.

Chapter 2

Background

2.1 Bayesian Models

This thesis revolves around the philosophy of Bayesian statistics, in which we assign subjective, probabilistic, *a priori* beliefs on the values of the parameters θ, which represent our objects of interest. These beliefs are captured using a prior distribution, $P(\theta)$. Given the likelihood $P(X|\theta)$, a functional representation of the relationship between the parameters and the data, $X = \{x_1, \ldots, x_N\}$, we apply Bayes's theorem to obtain the posterior, $P(\theta|X)$:

$$P(\theta|X) = \frac{P(X|\theta)P(\theta)}{\int P(X|\theta)P(\theta)\,\mathrm{d}\theta} \tag{2.1}$$

The posterior is the central object of importance in Bayesian statistical inference as it encodes our beliefs about the unknown quantities of interest, having observed information which depends on the parameters. With Bayes's theorem we have a clear way to update our prior beliefs on θ upon observing the data. Furthermore, the posterior provides a way to perform predictions on some new observations X^* with the posterior predictive distribution:

$$P(X^*|X) = \int P(X^*|\theta)P(\theta|X)\,\mathrm{d}\theta. \tag{2.2}$$

However, in many practical situations, it is impossible to obtain the posterior distribution in closed form because of the intractable integral in the denominator, $P(X) = \int P(X|\theta)P(\theta)\,\mathrm{d}\theta$. As we will see in Section 2.3, Bayes's theorem is unavailable in the Bayesian non-parametric setting as θ is infinite dimensional and thus does not admit a density.

2.2 Bayesian Inference

Because the posterior is generally not available in an exact, closed form, we need to resort to other methods for posterior inference. Broadly, Bayesian inference techniques fall under two categories: Sampling based Monte Carlo approaches, which generate simulations designed to draw samples from the posterior distribution, and variational approaches, which attempt to approximate the posterior with a simplet distribution chosen by minimizing the Kullback-Leibler divergence between the variational distribution and the true posterior. Because the scope of this dissertation will only focus on fast Markov chain Monte Carlo (MCMC) inference, this section will only provide background information on the classic inferential tools in MCMC. For details about variational approaches, read [11, 95, 10, 19, 92].

2.2.1 Monte Carlo Estimation and Importance Sampling

Monte Carlo methods are a class of numerical simulation algorithms that are commonly used in situations where we may want to simulate from a distribution or calculate an integral that is not easily obtainable, as is the

case in Bayesian inference. The basic idea is that we can approximate a distribution, $P(Y)$ by using N samples $y^{(i)} \sim P(Y)$ and forming an empirical distribution. Given this distribution of empirical samples, $\left\{y^{(i)}, i = 1, \ldots N\right\}$, we can also calculate unbiased estimates of integrals, $\bar{f} = \int f(Y)P(Y)\,\mathrm{d}Y$ with $\bar{f} \approx \frac{1}{N}\sum_{i=1}^{N} f\left(y^{(i)}\right)$ due to the law of large numbers. Unfortunately, we cannot even take samples from a posterior distribution $P(\theta|X)$ due to its intractable marginal likelihood term, so in general, we cannot use a straightforward Monte Carlo scheme in Bayesian inference.

Instead of sampling from the distribution of interest directly, we may instead sampling from some known distribution which is easy to draw samples from and weight these samples depending on how well they represent the true distribution. This is the basic idea behind importance sampling, one of the most popular methods for Monte Carlo estimations. If we define an importance distribution $Q(Y)$ that we can draw samples from, then we can again form unbiased estimates of some quantity $\bar{f}$ by noting

$$\begin{aligned} \bar{f} &= \int f(Y)P(Y)\,\mathrm{d}Y = \int f(Y)\frac{P(Y)}{Q(Y)}Q(Y)\,\mathrm{d}Y \\ &\approx \frac{1}{N}\sum_{i=1}^{N} f\left(y^{(i)}\right)\frac{P\left(y^{(i)}\right)}{Q\left(y^{(i)}\right)}, \end{aligned} \tag{2.3}$$

where $y^{(i)} \sim Q(Y)$. The quality of an importance sampler therefore depends on selecting a suitable importance distribution, $Q(Y)$ that closely resembles our distribution of interest, $P(Y)$ and does not produce estimators with infinite variance. Although the optimal importance distribution which minimizes the

estimator variance exists:

$$Q^*(Y) = \frac{|f(Y)|P(Y)}{\int |f(Y)|P(Y)\,\mathrm{d}Y}, \tag{2.4}$$

this distribution is generally not one that is easy to sample from. Furthermore, as the dimensionality of Y increases will become harder to generate good samples from $Q(Y)$ and empirically, importance samplers often face the problem where only one importance sample has non-negligible weight–meaning in actuality we do not average over different samples but only select one out of N importance samples [70].

2.2.2 Markov Chain Monte Carlo

If we want to obtain samples from some posterior distribution but cannot do so directly, we can instead simulate from a stochastic process that is guaranteed produce samples from the posterior distribution. In order to do this, we need to construct a Markov chain with the posterior, $P(\theta|X)$, as its stationary distribution. As long as a Markov chain is *irreducible*, meaning for any state of the chain it has non-zero probability moving to any other state, and *aperiodic*, meaning the chain is not a repeating cycle, then it is *ergodic* and has the target distribution as its unique stationary distribution.

The "classic" MCMC method is the Metropolis-Hastings algorithm [62, 38]. Given a proposal distribution, $Q(\theta)$, which provides candidate values for the next state of the Markov chain, the probability of transitioning to a state

$\theta' \sim Q(\theta)$ from a state $\theta^{(t)}$ at time t is

$$r = \min\left\{1, \frac{P(\theta'|X)Q(\theta^{(t)})}{P(\theta^{(t)}|X)Q(\theta')}\right\}. \tag{2.5}$$

Because this Markov chain stays at the previous state with probability $1 - r$ it is aperiodic, and if the proposal distribution of Q is chosen to have positive probability on the space of θ then the Metropolis-Hastings sampler is irreducible and therefore ergodic. Additionally, if a Markov chain exhibits detailed balance–meaning that for a given distribution $P(X)$ and a transition kernel $K(X|X')$ then if $P(X)K(X'|X) = P(X')K(X|X')$ then $P(X)$ is the limiting distribution of the Markov chain. The transition kernel for the Metropolis-Hastings algorithm is

$$K(\theta|\theta') = r(\theta, \theta')Q(\theta'|\theta) + \left(1 - \int r(\theta')Q(\theta'|\theta)\,\mathrm{d}\theta'\right)\delta_\theta, \tag{2.6}$$

given the acceptance probability $r(\cdot, \cdot)$ from Equation 2.5 and the proposal distribution Q conditioned on θ. It is possible to show that the Metropolis-Hastings algorithm exhibits detailed balance with the posterior as its limiting distribution: $P(\theta|X)K(\theta|\theta') = P(\theta'|X)K(\theta'|\theta)$. Thus, by detailed balance, the Metropolis-Hastings algorithm has a stationary distribution and by ergodicity this stationary distribution is unique. Moreover, the ratio $P(\theta'|X)/P(\theta^{(t)}|X)$ means we can ignore the posterior denominator $P(X)$ so that we may now sample from the posterior distribution without the problem of intractable integrals.

A special case of the Metropolis-Hastings algorithm commonly used in Bayesian inference is the Gibbs sampler. For a posterior of dimension P,

Algorithm 1: The Metropolis-Hastings Algorithm

for $t = 1, \ldots, T$ **do**
- Draw $\theta' \sim Q(\theta)$
- Calculate
$$r = \frac{P(X|\theta')P(\theta')Q(\theta^{(t)})}{P(X|\theta^{(t)})P(\theta^{(t)})Q(\theta')}$$
- Draw $u \sim \text{Uniform}(0, 1)$
- **if** $u < r$ **then**
  - $\theta^{(t+1)} := \theta'$
- **else**
  - $\theta^{(t+1)} := \theta^{(t)}$

$\theta = \{\theta_1, \ldots, \theta_P\}$, we can sample the joint distribution of $P(\theta|X)$ by iteratively sampling from the conditional distribution $P(\theta_p|X, \theta_{-p})$ for $p = 1, \ldots P$. This choice of proposal distribution for θ_p guarantees that the acceptance probability in the Metropolis-Hastings algorithm is always 1. If our choice of prior for $P(\theta_p)$ is conjugate, meaning that the posterior is in the same family of distributions as the prior, then the Gibbs sampler provides us with a computationally convenient Bayesian inference algorithm.

Algorithm 2: The Gibbs Sampling Algorithm

for $t = 1, \ldots, T$ **do**
- $\theta_1^{(t+1)} := \theta_1' \sim P(\theta_1^{(t)}|X, \theta_2^{(t)}, \ldots, \theta_P^{(t)})$
- $\theta_2^{(t+1)} := \theta_2' \sim P(\theta_2^{(t)}|X, \theta_1^{(t+1)}, \theta_3^{(t)}, \ldots, \theta_P^{(t)})$
- $\vdots$
- $\theta_p^{(t+1)} := \theta_p' \sim P(\theta_p^{(t)}|X, \theta_1^{(t+1)}, \ldots, \theta_{p-1}^{(t+1)}, \theta_{p+1}^{(t)}, \ldots, \theta_P^{(t)})$
- $\vdots$
- $\theta_P^{(t+1)} := \theta_P' \sim P(\theta_P^{(t)}|X, \theta_1^{(t+1)}, \ldots, \theta_{P-1}^{(t+1)})$

2.3 Bayesian Non-parametric Models

In a typical Bayesian parametric model, we have a set of parameters $\theta = \{\theta_1, \ldots, \theta_P\}$ on which we place prior distributions, $P(\theta)$, and using Bayes's theorem we obtain posterior distributions of the parameters given observed data, $P(\theta|X)$. However, in many modeling situations we may have a variety of competing models to fit to the data. In the Bayesian parametric setting, we might do this by computing posterior model probabilities

$$P(M_k|X) = \frac{P(M_k)\int P(X|\theta, M_k)P(\theta|M_k)\,\mathrm{d}\theta}{\sum_{k=1}^{K} P(X|M_k)P(M_k)}, \tag{2.7}$$

given prior model probability $P(M_k)$ for $k = 1, \ldots, K$ models. A model in this case could refer to the number of mixture components in a mixture model or the number of factors in a latent factor model. Fitting all possible models and selecting one is generally infeasible, especially because it is typically not obvious how many mixtures or factors, for example, to include.

The non-parametric approach to this problem would instead allow us to avoid specifying the number or size of the models to investigate and instead let the model complexity adapt to the data. The parameter space is fixed in the typical Bayesian setting, so the Bayesian non-parametric (BNP) solution is to instead assume an infinite dimensional parameter space for the prior, but choose a likelihood that uses a finite number of parameters per observation. The rest of this section will proceed with brief introductions to three of the most popular non-parametric priors–the Dirichlet process, the beta-Bernoulli process and the Gaussian process, and their practical applications to statistical

modeling.

2.3.1 The Dirichlet Process

The Dirichlet process (DP) [22] is a distribution over the space of functions, specifically over the space of probability distributions, with the property that marginal distributions of the Dirichlet process are Dirichlet distributed. D is Dirichlet process-distributed, given a base distribution $H(\cdot)$ and concentration parameter α, if for any finite, disjoint partition of θ, $\{A_1, \ldots, A_K\}$:

$$D(A_1), \ldots, D(A_K) \sim \text{Dirichlet}(\alpha H(A_1), \ldots, \alpha H(A_K)). \tag{2.8}$$

Given a sequence of observations $\theta_1, \ldots, \theta_n$ drawn from a Dirichlet process, D, the posterior of D is again Dirichlet process-distributed

$$D|\theta_1, \ldots, \theta_n \sim \text{DP}\left(\alpha + n, \frac{\alpha}{\alpha + n} H + \frac{1}{\alpha + n} \sum_{i=1}^{n} \delta_{\theta_i}\right). \tag{2.9}$$

If we have an observation θ_{n+1} and integrate out D, then we obtain a predictive distribution [9]

$$\theta_{n+1}|\theta_1, \ldots \theta_n \sim \frac{1}{\alpha + n}\left(\alpha H + \sum_{i=1}^{n} \delta_{\theta_i}\right). \tag{2.10}$$

This representation of the Dirichlet process, commonly known as the Pólya urn representation [9], is related to the Chinese restaurant process for clustering and mixture models. Suppose we observe a sequence $\theta_1, \ldots, \theta_n$ from a Dirichlet process where observation i of this sequence is located at a position in $\{\phi_1, \ldots, \phi_K\}$ for the K locations currently occupied by $\theta_1, \ldots, \theta_n$. We can

obtain a posterior predictive distribution for θ_{n+1} with

$$\theta_{n+1}|\theta_1, \ldots, \theta_n \sim \frac{1}{\alpha + n}\left(\alpha H + \sum_{k=1}^{K} n_k \phi_k\right). \tag{2.11}$$

This distribution produces clustering behavior where new sequences of θ decide to join a cluster based on the cluster's popularity, n_k, or join a new cluster depending on the value of α (also known as the "rich get richer" property). This distribution is known as the Chinese restaurant process [2] due to an analogy which anthropomorphically describes this process as a customer entering a Chinese restaurant and the customer will either sit at a new table with probability proportional to α or will sit at a table with other customers with probability proportional to the number of customers sitting at that table. As we will see later, this representation of the Dirichlet process becomes useful for inference as we can avoid the difficulties of performing Bayesian inference on an infinite dimensional object.

Another insightful representation of the Dirichlet process is available in the stick breaking construction [81], which takes a "stick" of length one and recursively breaks off $\text{Beta}(1, \alpha)$ length pieces from each previous segment. The stick pieces represent point masses on distributions located at ϕ_k. Therefore, if $D \sim DP(\alpha, H)$ then we can define D as:

$$\beta_k \sim \text{Beta}(1, \alpha),\ \pi_k = \beta_k \prod_{\ell=1}^{k-1}(1 - \beta_\ell),\ \phi_k|H \sim H,\ D = \sum_{k=1}^{\infty} \pi_k \delta_{\phi_k}. \tag{2.12}$$

From the representation of D as $D = \sum_{k=1}^{\infty} \pi_k \delta_{\phi_k}$, it is now evident that the Dirichlet process is a discrete distribution. Thus, draws from the Dirichlet

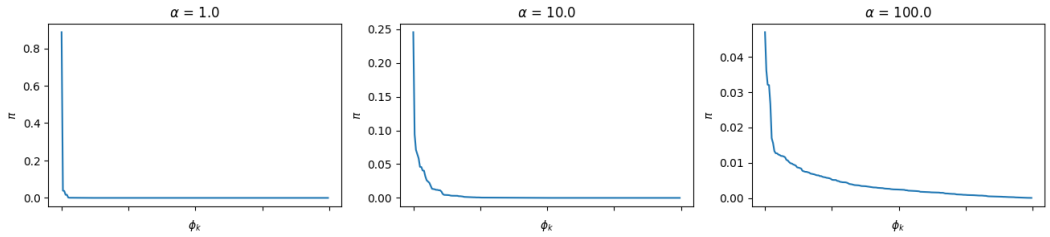

Figure 2.1: Draws from a stick breaking Dirichlet process.

process are guaranteed to have ties. This discrete property becomes meaningful when using Dirichlet processes in the context of mixture modeling and clustering, as illustrated with the Chinese restaurant process.

2.3.2 Example: The Infinite Mixture Model

The most common application of Dirichlet processes in Bayesian non-parametric modeling is the example of infinite mixture models. A *finite* Dirichlet mixture model with K components is generally represented as

$$\phi_k \sim H,\ \pi \sim \text{Dirichlet}(\alpha, \ldots, \alpha),\ X \sim \prod_{i=1}^{N} \sum_{k=1}^{K} \pi_k L(x_i, \phi_k), \tag{2.13}$$

for data $X = \{x_1, \ldots, x_N\}$ and a likelihood function $L(\cdot, \cdot)$. To facilitate inference for this mixture model, we use a data augmentation scheme by introducing a latent variable $z_i \sim \pi$ which indicates the mixture or cluster membership of observation i. With this data augmentation, the mixture model is now

$$\phi_k \sim H,\ \pi \sim \text{Dirichlet}(\alpha, \ldots, \alpha),\ z_i \sim \pi,\ x_i \sim L(x_i, \phi_{z_i}). \tag{2.14}$$

In the Bayesian non-parametric setting, we now replace the Dirichlet distribution with a Dirichlet process and we can represent a Dirichlet process mixture

model [4, 74] as

$$D|\alpha, H \sim DP(\alpha, H),\ \theta_i \sim D,\ x_i|\theta_i \sim L(x_i, \theta_i). \tag{2.15}$$

Alternatively, we can explicitly model the atom weights using the stick breaking construction to represent the infinite mixture model as

$$\phi_k \sim H,\ \beta_k \sim \text{Beta}(1, \alpha),\ \pi_k = \beta_k \prod_{\ell=1}^{k-1}(1-\beta_\ell),\ z_i \sim \pi,\ x_i \sim L(x_i, \phi_{z_i}). \tag{2.16}$$

Here, we have replaced the mixing parameter with the stick breaking construction of the mixing weights seen in Equation 2.12. In the Chinese restaurant process representation, we marginalize π so that the mixture model is now

$$\phi_k \sim H,\ z_i = k \sim \begin{cases} \frac{n_{-ik}}{N+\alpha-1}, & n_k > 0 \\ \frac{\alpha}{N+\alpha-1}, & n_k = 0 \end{cases},\ x_i \sim L(x_i, \phi_{z_i}), \tag{2.17}$$

where $n_{-ik} = |\{z_j : z_j = k, j \neq i\}|$ and n_k is the number of observations allocated to cluster k. Even though there is an *a priori* assumption that there are infinite mixture or clusters, we are guaranteed to have data associated with only finitely many of the clusters. In this framework, the number of clusters occupied will depend on the data and the concentration parameter, α. As we will see later, the quality and speed of our inference algorithms will depend heavily on whether we choose to explicitly represent the infinite dimensional mixing component with the stick-breaking representation or marginalize out the mixing component with the Chinese restaurant process.

2.3.3 The Beta-Bernoulli Process

Section 2.3.1 introduced a distribution over distributions through the Dirichlet process and described how various constructions of the DP can pro-

duce clustering and mixture models with an assumed *a priori* infinite number of components. Similarly, this section will introduce distribution over infinite binary components through the beta-Bernoulli process and demonstrate its utility in modeling data composed of infinite latent factors.

The beta process [41, 91], $B \sim BP(\alpha, H)$, is defined by a concentration parameter α on a base measure H that has total mass γ on the space Φ. The beta process can then be described as $B = \sum_{k=1}^{\infty} \pi_k \delta_{\phi_k}$. This representation is similar to the point mass representation in the Dirichlet process, except in the Beta process the weights π_k need not sum to one. An alternative representation of the beta process [72] can be defined, given a finite process where

$$\pi_k \sim \text{Beta}\left(\frac{\alpha\gamma}{K}, \alpha\left(1 - \frac{\gamma}{K}\right)\right),\ \phi_k\ \sim H/\gamma,\ B_K = \sum_{k=1}^{K} \pi_k \delta_{\phi_k}. \tag{2.18}$$

Letting $K \to \infty$ means $B_K \to B$, a beta process.

Like the stick breaking representation for the Dirichlet process in Equation 2.12, the beta process also has many stick breaking representations [89, 72, 12]. Represented below is the stick breaking representation of [72]. If $B \sim BP(\alpha, H)$ then we can represent it as

$$\begin{aligned} C_k &\sim \text{Poisson}(\gamma),\ \phi_{kj} \sim H/\gamma,\ \beta_{kj}^{(\ell)} \sim \text{Beta}(1, \alpha), \\ B &= \sum_{k=1}^{\infty} \sum_{j=1}^{C_k} \beta_{kj}^{(\ell)} \prod_{\ell=1}^{k-1} \left(1 - \beta_{kj}^{(\ell)}\right) \delta_{\phi_k} \end{aligned} \tag{2.19}$$

In other words, draws from a beta process are an infinite collection of binary probabilities at located in the space of Φ. With infinite binary probabilities, we can then produce infinite binary draws through the Bernoulli process.

If X_i is a Bernoulli process and B is a discrete measure (assume it is a beta process $B \sim BP(\alpha, H)$ though it need not be) then $X_i \sim \text{BeP}(B)$ [91]. We can generate samples of X_i with the following generative process:

$$z_{ik}|\pi_k \sim \text{Bernoulli}(\pi_k),\ X_i = \sum_{k=1}^{\infty} z_{ik}\delta_{\phi_k}. \tag{2.20}$$

Here we can see the infinite dimensional modeling capability of the beta-Bernoulli process where an observation X_i contains the presence of features ϕ_k with probability π_k. If B is a continuous measure, then $\text{BeP}(B)$ will be a Poisson process with B as its intensity function and $X_i = \sum_{k=1}^{M} \delta_{\phi_k}$, where $M \sim \text{Poisson}(B(\Phi))$. In this case, the Bernoulli process acts as the process which selects new features to allocate to observations. If B is continuous and discrete then the Bernoulli process is a sum of the continuous and discrete Bernoulli processes. Moreover, the beta process is conjugate with the Bernoulli process. After observing $X_1, \ldots, X_n$ observations from

$$X_i|B \sim \text{BeP}(B),\ B \sim \text{BP}(\alpha, H), \tag{2.21}$$

the posterior distribution is

$$B|X_1, \ldots X_n \sim BP\left(\alpha + n, \frac{\alpha}{\alpha + n}H + \frac{1}{\alpha + n}\sum_{i=1}^{n} X_i\right). \tag{2.22}$$

After marginalizing out B, the predictive distribution is

$$X_{n+1}|X_1, \ldots X_n \sim BeP\left(\frac{\alpha}{\alpha + n}H + \frac{1}{\alpha + n}\sum_{i=1}^{n} X_i\right) \tag{2.23}$$

[91] showed that this predictive posterior distribution is a two parameter Indian buffet process with parameters (α, γ). We can see this when we separate

X_{n+1} into its discrete and continuous components. The discrete component $\text{BeP}\left(\frac{1}{\alpha+n}\sum_{i=1}^{n} X_i\right)$ allocates new observations to features proportional to the number of previous observations that select this feature while the continuous component, $\text{BeP}\left(\frac{\alpha}{\alpha+n}H\right)$ generates $\text{Poisson}\left(\frac{\alpha\gamma}{\alpha+n}\right)$ new features for observation $n+1$ to occupy.

The original, one parameter Indian buffet process [36] is defined with the finite probability model

$$\pi_k \sim \text{Beta}\left(\frac{\alpha}{K}, 1\right), \; z_{ik} \sim \text{Bernoulli}(\pi_k), \tag{2.24}$$

and letting $K \to \infty$. This distribution $Z \sim \text{IBP}(\alpha)$ is represented by

$$P(Z) = \frac{\alpha^K}{\prod_{h=1}^{2^{N-1}} K_h!} \exp\{-\alpha H_N\} \prod_{k=1}^{K} \frac{\left(N - \sum_{i=1}^{N} z_{ik}\right)! \left(\sum_{i=1}^{N} z_{ik} - 1\right)!}{N!}, \tag{2.25}$$

where H_N is the N-th harmonic number. This representation is equivalent to the posterior predictive distribution in Equation 2.22, where $\gamma = 1$. Similar to the Chinese restaurant process, we can describe the behavior of this process as the i-th customer (observation) who enters an Indian restaurant and selects $\text{Poisson}(\alpha/i)$ new dishes (features) and chooses a previous dish k with probability proportional to the number of previous customers who also selected dish k. With the Indian buffet process, we now have a distribution from which to draw infinite dimension binary matrices that we could use for latent factor modeling.

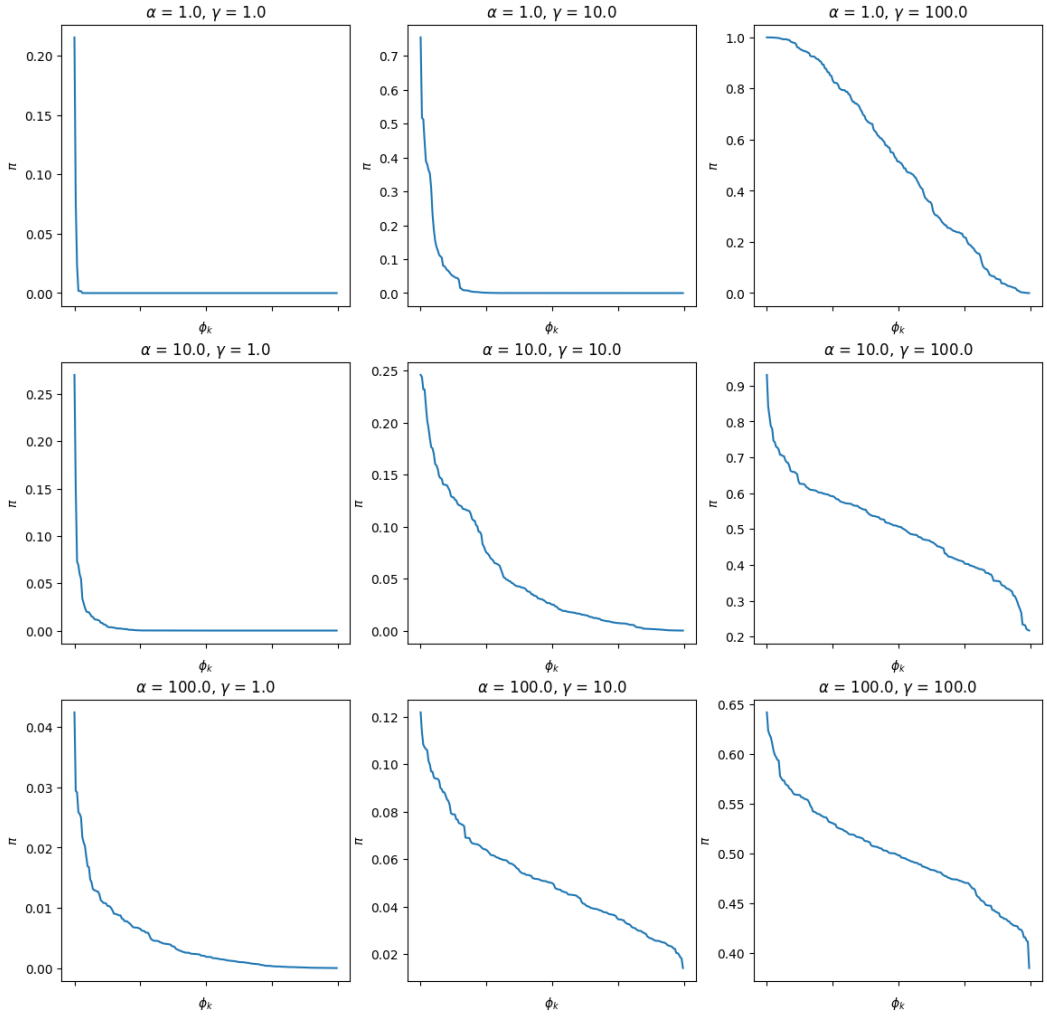

Figure 2.2: Draws from a stick breaking beta process.

2.3.4 Example: The Infinite Latent Factor Model

A latent factor model assumes that an $N \times D$ dataset X can be decomposed as a combination of $K < N$ features. One popular example of a finite factor model is factor analysis [15]. In factor analysis, we can represent the data in a lower dimensional format as $X = LF + \epsilon$ where F is a $K \times D$ matrix representing the "factors" or latent features and L is a $N \times K$ is the "factor loading" matrix which represent the intensity an observation exhibits a particular factor and ϵ represents the noise term centered at 0 with variance σ^2.

We can extend the factor analysis model into a sparse infinite dimensional case where the factor loading matrix is represented by a $N \times \infty$ binary matrix Z indicating the presence or absence of a certain feature A_k, a row in the $\infty \times D$ matrix A for observation X_i. Again, the infinite latent factor model is represented with the Indian buffet process as

$$Z \sim \text{IBP}(\alpha),\ A_k \sim N(0, \sigma_A^2),\ \epsilon_i \sim N(0, \sigma_X^2 I),\ X = ZA + \epsilon, \tag{2.26}$$

or with the beta-Bernoulli process by assuming $Z|B \sim \text{BeP}(B), B \sim (\alpha, H)$. Examples of latent factor models can be found in applications of image modeling [44], where images can be decomposed into basic components that are assumed to be superimposed on each other. Or in choice modeling [32], where N users use D or some subset of D objects (movies, products, etc.) and we can decompose their consumption into latent factors which can then be used to identify similar consumers.

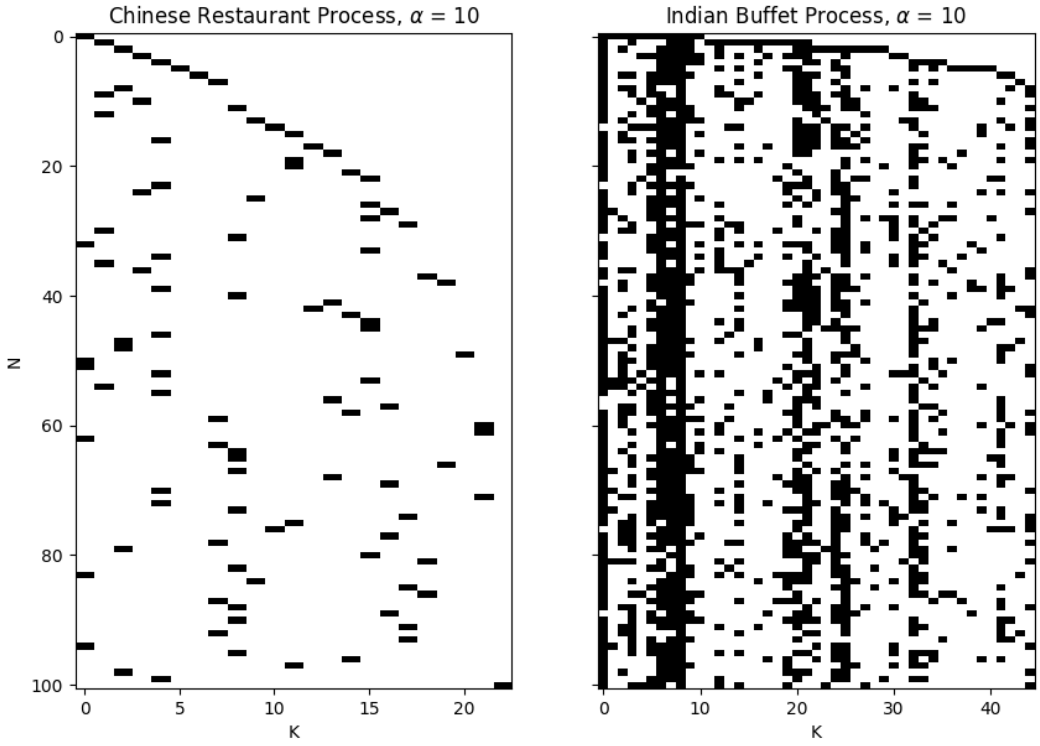

Figure 2.3: Draws from a Chinese restaurant process and Indian buffet process.

2.3.5 Gaussian Processes

In Sections 2.3.1 and 2.3.3, we examined examples of Bayesian non-parametric models where the latent processes were assumed to have a discrete distribution (i.e. the cluster membership of data or the features allocated to data). In the example of Gaussian process models, however, we now assume that the latent process we wish to model is instead continuous or more specifically–a real-valued function.

We may be interested in using Gaussian processes to model functions in situation where we do not or cannot place a parametric form on a latent function we wish to model. Function learning is a common problem in machine learning and statistics, but if we are in a situation where we do not want to place strong assumptions of the nature of the latent function (e.g. a linear function) then the function learning problem becomes very difficult. However, the space of functions that we would like to consider which maps some inputs X to an output Y should be smooth so we should somehow only consider functions , $f : X \to Y$, that "behave nicely".

The Bayesian non-parametric solution to function learning is to place a prior distribution on the space of functions (assumed to be infinite dimensional) and learn the latent function from the posterior distribution. The Gaussian process (GP) [76] is a popular choice of prior over the space of smooth functions, as we will see, due to its convenience and tractability in inference.

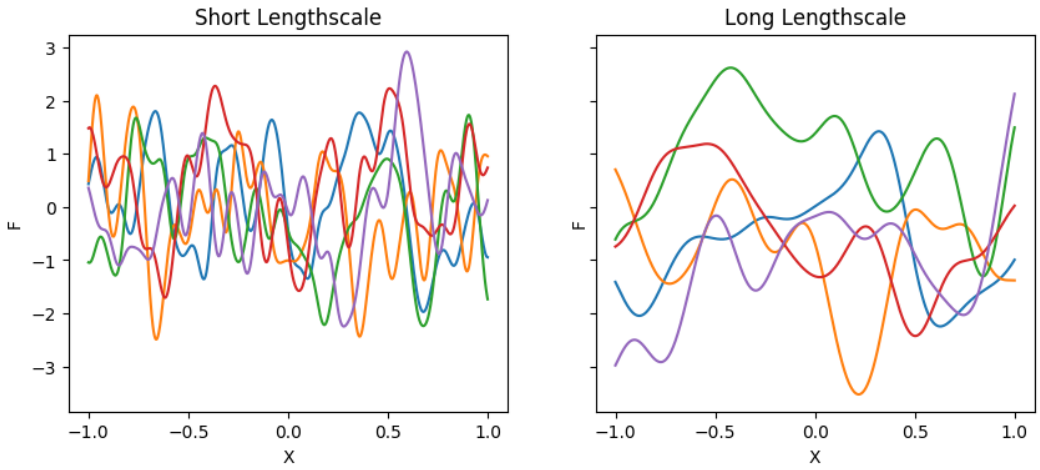

Figure 2.4: Draws from a Gaussian process with an RBF kernel.

A Gaussian process is a distribution over real valued functions, $f : \mathbb{R}^D \to \mathbb{R}$, defined by a mean function $m(\cdot)$ and a covariance kernel function $\Sigma(\cdot, \cdot)$ with the property that, evaluated at any finite set of points X, a Gaussian process distributed function is multivariate normal centered at $m(X)$ with covariance $\Sigma(X, X')$

$$f \sim \mathrm{GP}(m, \Sigma) \to f|X \sim N(m(X), \Sigma(X, X')). \tag{2.27}$$

Because of this multivariate normal property and the fact that functions drawn from this prior are smooth, the Gaussian process is often an attractive choice of function prior as this can lead to tractable posterior inference.

2.3.6 Example: Bayesian Non-linear Regression

A basic example of a Gaussian process model can be seen in a non-linear regression problem. Suppose we observe inputs X and outputs Y and

assume the relation between X and Y is some function f. Then using the Gaussian process we typically represent this model as

$$f \sim \mathrm{GP}(0, \Sigma),\ \epsilon \sim N(0, \sigma^2 I),\ Y = f(X) + \epsilon. \tag{2.28}$$

Because of the multivariate normal property of the Gaussian process, we have a normally distributed likelihood that is conjugate to a normally distributed. Thus, we have posterior, marginal likelihood and predictive posterior distributions available in closed form. Given predictive inputs X^*, the posterior predictive distribution of the latent function f^* is

$$\begin{aligned} P(f^*|-) &\sim N(\mu_f, \Sigma_f) \\ \mu_f &= \Sigma(X^*, X)\left[\Sigma(X, X) + \sigma^2 I\right]^{-1} Y \\ \Sigma_f &= \Sigma(X^*, X^*) - \Sigma(X^*, X)\left[\Sigma(X, X) + \sigma^2 I\right]^{-1} \Sigma(X, X^*) \end{aligned} \tag{2.29}$$

Because the posterior and posterior predictive distributions are available in closed form, the only procedure remaining in Gaussian process regression inference is learning the hyperparameters θ associated with the covariance function $\Sigma(\cdot, \cdot)$. Perhaps the most common approach in the regression setting is to optimize the hyperparameters with respect to the marginal likelihood $\int P(Y|f, X)P(f|X)\,\mathrm{d}f$:

$$P(Y|X) = (2\pi)^{-\frac{N}{2}}\left|\Sigma(X, X') + \sigma^2 I\right|^{-\frac{1}{2}} \exp\left\{-\frac{1}{2} Y^T\left[\Sigma(X, X') + \sigma^2 I\right]^{-1} Y\right\}. \tag{2.30}$$

Section 2.4.2 will further discuss different methods of inference for Gaussian process inference.

2.4 Bayesian Non-parametric Inference

2.4.1 Discrete Models

An immediate, non-trivial question for Bayesian non-parametrics is how to perform inference for models with infinite dimensional priors. For discrete non-parametric models, like the Dirichlet process or the beta-Bernoulli process, the question for inference is how to represent the infinite random measure π. For Bayesian non-parametric inference, we can represent π with a finite approximation [48, 71, 103]. Instead of relying on a fixed approximation of a Bayesian non-parametric model, instead we can use a slice sampler for inference under the stick breaking representation of the Dirichlet process or the beta-Bernoulli process which only instantiates a finite number of components through random truncations that the auxiliary slice variable produces [96, 50, 89].

For example, in the stick breaking representation of the Dirichlet process mixture model from Equation 2.16, the likelihood for observation i is $P(x_i|\pi,\phi) = \sum_{k=1}^{\infty} \pi_k L(x_i,\phi_k)$. With the slice variable, this likelihood becomes $P(x_i,u_i|\pi,\phi) = \sum_{k=1}^{\infty} \mathbf{I}(u_i < \pi_k) L(x_i,\phi_k)$, where $\mathbf{I}(\cdot)$ is the indicator function. Because the stick-breaking weights are decreasing as k increases, only a finite number of atoms will be instantiated at a given slice truncation level u_i and thus we only need to consider the atoms $\{k : \pi_k \geq u_i\}$ when sampling $P(z_i = k|-)$. Because we recover the original model when we integrate out u_i, this slice sampling approach [96] is a valid sampler for the Dirichlet process mixture model.

Slice sampling in the stick-breaking Indian buffet representation of the beta-Bernoulli process [89] operates similarly to its Dirichlet process equivalent. Again, we draw a slice variable $u \sim \text{Uniform}(0, \beta^*)$ where β^* is the smallest stick weight of features that are turned on (i.e. allocated to an observation) which represents the truncation level for instantiating feature weights. Thus, the infinite dimensional matrix Z is now represented by a truncated, finite matrix.

$$P(Z|u,\beta) = \frac{1}{\beta^*}\mathbf{I}(0 \leq u \leq \beta^*)P(Z|\beta),\ \beta^* = \min\left\{1, \min_{k:\exists i, z_{ik}=1} \beta_k\right\}. \quad (2.31)$$

Otherwise, we can marginalize π which allows us to completely avoid issues with representing an infinite dimensional object (which will be referred to as "collapsed" samplers). For Dirichlet process mixture models, we can integrate out the random mixing measure to obtain the Chinese restaurant process as seen in Equation 2.17 from which we can sample the cluster allocation directly. Radford Neal's "Algorithm 8" is perhaps the most well known collapsed MCMC sampler for Dirichlet process mixtures [66]. The essential idea of Algorithm 8 is that it introduces m auxiliary components that represent unoccupied clusters. If K^+ is the set of K occupied clusters and K^- represent the m unoccupied, auxiliary variables then the probability of observation i joining a cluster in K^+ is proportional to its popularity without observation i, n_{-ik}, times its likelihood, whereas joining a cluster in K^- is proportional to α/m times the likelihood. We then update the posterior for feature $\phi_k,\ k \in K^+$ and draw new features from H for $\phi_k,\ k \in K^-$.

For the beta-Bernoulli process, integrating out π gives us the Indian buffet process representation. Similar to the Chinese restaurant process, we can now sample feature allocations directly with probabilities dependent on the number of observations allocated to feature k. [37] provides a simple Gibbs sampling routine for inference in Indian buffet process models with application in the latent Gaussian factor model as seen in Equation 2.26.

2.4.2 Continuous Models

Instead of optimizing the hyperparameters with respect to the (approximate) marginal likelihood we may approach the problem of learning the hyperparameters in the Bayesian framework and place a prior on the hyperparameters, $P(\theta)$. In the Bayesian setting we could perform hyperparameter inference by computing a Laplace approximation of θ or if we are interested in the exact posterior distribution of the hyperparameters we may sample them via Hamiltonian Monte Carlo.

However, this is still a serious, inherent problem in Gaussian process models–inference requires inverting an $N \times N$ matrix which generally scales $O(N^3)$. Numerous approaches for fast Gaussian process have been developed, most of which attempt to reduce the complexity of inverting a large matrix. Scalable Gaussian process methods largely fall under two categories: (1) Sparse methods, which learn the function posterior through representative "inducing points" and will be discussed below, and (2) local methods, which partition the data and assume a block diagonal structure in the covariance matrix, and

will be discussed in Section 2.5.

One of the most popular sparse methods, FITC [84], learns the locations of these inducing points, X_u, by maximum likelihood estimation jointly with the model hyperparameters. We assume there are M pseudo-inputs, X_u (or inducing points) with associated pseudo-targets $f_u|X_u \sim N(0, \Sigma(X_u, X_u')$, which is chosen to match the prior on the latent function f. Using results from conditional Gaussians, we can see that

$$\begin{aligned} P(Y|X, X_u, f_u) \sim N(\mu_u, \Sigma_u),\ \mu_u = \Sigma(X, X_u)\Sigma^{-1}(X_u, X_u')f_u, \\ \Sigma_u = \Sigma(X, X') - \Sigma(X, X_u)\Sigma^{-1}(X_u, X_u')\Sigma(X_u, X) + \sigma^2 I. \end{aligned} \tag{2.32}$$

By assuming $\Lambda = \text{diag}\left\{\Sigma(X, X') - \Sigma(X, X_u)\Sigma^{-1}(X_u, X_u')\Sigma(X_u, X)\right\}$, we can form a likelihood that factorizes over the N observations given the pseudo-input and targets:

$$P(Y|X, X_u, f_u) = \prod_{i=1}^{N} P(Y_i|X_i, X_u, f_u) \sim N\left(\mu_u, \Lambda + \sigma^2 I\right) \tag{2.33}$$

Given the prior $P(f_u|X_u)$ and the likelihood $P(Y|X, X_u, f_u)$ we can obtain the posterior distribution:

$$\begin{aligned} Q_u = \Sigma(X_u, X_u') + \Sigma(X_u, X)\left(\Lambda + \sigma^2 I\right)^{-1}\Sigma(X, X_u) \\ \mu_f = \Sigma(X_u, X_u')Q_u^{-1}\Sigma(X_u, X)\left(\Lambda + \sigma^2 I\right)^{-1} Y \\ \Sigma_f = \Sigma(X_u, X_u')Q_u^{-1}\Sigma(X_u, X_u'),\ P(f_u|X, YX_u) \sim N(\mu_f, \Sigma_f) \end{aligned} \tag{2.34}$$

and the posterior predictive distribution:

$$\begin{aligned} P(Y^*|X^*, X, Y, X_u) \sim N(\mu^*, \Sigma^*), \\ \mu^* = \Sigma(X^*, X_u)Q_u^{-1}\Sigma(X_u, X)\left(\Lambda + \sigma^2 I\right)^{-1} Y \\ \Sigma^* = \Sigma(X^*, X^{*\prime}) - \Sigma(X^*, X_u)(\Sigma(X_u, X_u')^{-1} - Q_u^{-1})\Sigma(X_u, X^*) + \sigma^2 I. \end{aligned} \tag{2.35}$$

Lastly we can integrate out f_u to obtain the marginal likelihood:

$$P(Y|X, X_u) \sim N(0, \Sigma(X, X_u)\Sigma^{-1}(X_u, X_u')\Sigma(X_u, X) + \Lambda + \sigma^2 I) \qquad (2.36)$$

Inference in FITC learns the hyperparameters and the inducing locations jointly by optimizing with respect to $P(Y|X, X_u)$.

2.5 Distributed Inference

The central content of this dissertation will focus on scalable and, in particular, *distributable* MCMC methods for Bayesian non-parametric inference. The majority of the parallel methods introduced in this dissertation are data parallel algorithms, in which the data is distributed across multiple machines but are assumed to have the same model. Broadly speaking, data parallel MCMC algorithms fall under two categories: (1) embarrassingly parallel MCMC algorithms and (2) exact, parallel MCMC algorithms.

2.5.1 Embarrassingly Parallel MCMC

Each embarrassingly parallel MCMC algorithm generally consists of two stages. The first stage is where exact posterior sampling is performed on a processor p containing only a subset of the data. After local processor MCMC inference, each processor generates a *subposterior*. For many of these approaches, the likelihood of the data is either upweighted, $P(X^{(p)}|\theta) = \prod_{i=1}^{N_p} P(X_i^{(p)}|\theta)^P$ to form an estimate of the full likelihood using only a subset of the data or the prior is downweighted, $P(\theta^{(p)}) = P(\theta)^{\frac{1}{P}}$ so that the com-

bined subposterior, $P(\theta^{(p)}|X^{(p)}) \propto \prod_{p=1}^{P} P(\theta)^{\frac{1}{P}} \prod_{i=1}^{N_p} P(X_i^{(p)}|\theta)$, is equal to the full posterior.

The second stage, after subposterior inference, is combining subposteriors to form a final posterior. Existing embarrassingly parallel methods differ in how to combine these subposterior to generate an estimate for the full posterior. This type of approach is called *embarrassingly parallel* because we can trivially distribute computation in this scheme across as many processors available and only require interprocessor communication when combining estimates to form a final posterior.

The consensus Monte Carlo algorithm [79] places parametric assumptions on the subposterior distribution–namely that it assumes the combined posterior is Gaussian. Supposing each processor p produces T subposterior samples, $\{\theta_{p1}, \ldots, \theta_{pT}\}$. The combined posterior draw is

$$\hat{\theta}_t = \left(\sum_{p=1}^{P} W_p \right)^{-1} \sum_{p=1}^{P} W_p \theta_{pt}, \tag{2.37}$$

given a weighting matrix W_p typically chosen to be the subposterior variance of θ on processor p. If each subposterior is Gaussian distributed then $\theta_t \sim N\left(\sum_{p=1}^{P} W_p \theta_{pt}, \left(\sum_{p=1}^{P} W_p \right)^{-1} \right)$ and we may draw posterior samples from this combined posterior distribution.

Instead of assuming a Gaussian distribution on the consensus posterior, [67] fits a non-parametric kernel density estimator with Gaussian kernels on

the subposterior samples. This produces a mixture model on the subposteriors,

$$\hat{P}(\theta|X^{(p)}) = \frac{1}{T}\sum_{t=1}^{T} N(\theta_{pt}, h^2 I) \tag{2.38}$$

for a bandwidth parameter h. Combining all P of these kernel density estimates produces a product estimate of the full posterior,

$$\hat{P}(\theta|X) = \frac{1}{T^P}\prod_{p=1}^{P}\sum_{t=1}^{T} N(\theta_{pt}, h^2 I). \tag{2.39}$$

Equivalently, the estimate of the full posterior is a mixture of T^P Gaussians. To sample from the combined posterior, first we must sample a mixture component and then sample from that mixture component's distribution. This MCMC algorithm is proven to be an asymptotically exact sampler as the number of subposterior draws T approaches infinity.

Alternately, we can obtain a combined posterior by taking the geometric median of the P subposteriors. The geometric median is a generalization of the univariate median to multiple dimensions which, for a finite collection of points $\{\theta_1, \ldots, \theta_P\} \in \Theta$, is

$$\theta^* = \underset{\theta\in\Theta}{\operatorname{argmin}} \sum_{p=1}^{P} ||\theta - \theta_p||, \tag{2.40}$$

given a normed space Θ equipped with norm $||\cdot||$. [64] has shown that using the geometric median to combine subposteriors in the embarrassingly parallel MCMC framework has good frequentist posterior concentration qualities and is provably robust to outliers in the data. Furthermore, the theoretical properties of geometric median combination, unlike consensus Monte Carlo, do not rely on

distribution assumptions of the posterior and, through the Weiszfeld algorithm [99], is faster than sampling the posterior from T^M Gaussian mixtures.

For Bayesian non-parametric models, embarrassingly parallel methods are particularly suited for a class Gaussian process inference techniques known as "product of experts" [68], which assumes the marginal likelihood of the Gaussian process regression models factorizes across partitions,

$$P(Y|X,\theta) \approx \prod_{p=1}^{P} P(Y^{(p)}|X^{(p)},\theta). \tag{2.41}$$

In this formulation, the dense covariance matrix becomes a low-rank block diagonal matrix and therefore training this model scales $O\left(N_p^3\right)$ per partition. Each variation of the product of expert models differs only in how to combine predictions formed on each "expert" (partition). The product of experts model generates predictions by taking weighted products of each expert:

$$P(f^*|X^*,X,Y,\theta) = \prod_{p=1}^{P} P(f^*|X^{(p)},Y^{(p)},X^*,\theta)^{\beta_p}. \tag{2.42}$$

Since the marginal likelihood is assumed to factorize along P, then computation, specifically the matrix inversion, is now distributable with each machine only having to invert a smaller N_p sized matrix.

2.5.2 Exact Parallel MCMC

While the embarrassingly parallel framework is attractive for its conceptual simplicity and computational convenience, it is not obvious how a given combined subposterior differs from the true posterior when the true posterior

is not Gaussian. Furthermore, these methods are unsuitable for problems like mixture models or latent factor models because those types of models with exhibit a serious label switching problem when trivially distributed in such a framework.

Another approach for distributable inference is to exploit conditional independence in model structure. Conditional independence in the Bayesian framework means that if a set of parameters $Z = \{\theta_1, \ldots, \theta_P\}$ parameters are independent given the data and some other parameters Z^c and then we can update the parameters in Z in parallel while still maintaining the exactness of the MCMC sampler. The most simple example of a parallelizable conditional structure is one where the likelihood completely factorizes across N observations given the parameters:

$$P(\theta|X) \propto P(\theta) \prod_{i=1}^{N} L(x_i, \theta). \tag{2.43}$$

In this setting we can distribute the data across P processors and we may calculate the full likelihood function $\prod_{i=1}^{N} L(x_i, \theta)$. in parallel and send that value (or if in the case of Gibbs sampling, the summary statistics) to a master processor to sample a new value of θ from $P(\theta|X)$. Unfortunately in this simple setting, we need to have constant communication between the worker processors and the master to sample new values of θ which we need to avoid considering processor communication carries a substantial computational overhead, unlike in the case of the embarrassingly parallel samplers.

A more reasonable example of when this type of parallel approach can

be useful would be in the case of a mixture model. Suppose we model our data using a Dirichlet mixture of K Gaussians:

$$\mu_k \sim N(\mu_0, \Sigma_0), \pi \sim \text{Dirichlet}(\alpha, \ldots, \alpha), z_i \sim \pi, x_i \sim N(\mu_{z_i}, \Sigma_X) \tag{2.44}$$

Given the mixing weights π, the mixture components μ_k and observation x_i, the posterior distribution of $Z = (z_1, \ldots, z_N)$ is conditionally independent, and hence we can distribute X and the latent indicators Z across processors and update Z in parallel. After some L sub-iterations on each processor we can send the summary statistics given Z to the master processor to sample μ_k and π from its full conditional distributions and send the new values of (μ_k, π) to the worker processors.

Algorithm 3: Parallel Bayesian Gaussian Mixture Model Inference

for $p = 1, \ldots, P$ *in parallel* **do**
  Receive new values of π and μ_k from master processor
  for $\ell = 1, \ldots L$ **do**
    for $i = 1, \ldots N_p$ **do**
      Sample $z_{ip} \sim P(z_i = k|-) \propto \pi_k N(x_i; \mu_k, \Sigma_X)$
  Send summary statistics, $n_{kp} := |\{z_i = k : i = 1, \ldots, N_p\}|$ and $\bar{x}_{kp} := \sum_{i:z_{ip}=k} x_i$ for $k = 1, \ldots, K$ to master processor
if *master processor* **then**
  Combine summary statistics from worker processors, $n_k := \sum_{p=1}^{P} n_{kp}$ and $\bar{x}_k := \sum_{p=1}^{P} \bar{x}_{kp}$
  Sample $\pi \sim \text{Dirichlet}(\alpha + n_1, \ldots, \alpha + n_K)$
  Sample $\mu_k \sim N\left(\left(\Sigma_0^{-1} + n_k\Sigma_X^{-1}\right)^{-1}\left(\mu_0 + \bar{x}_k\right), \left(\Sigma_0^{-1} + n_k\Sigma_X^{-1}\right)^{-1}\right)$
  Send new values of π and μ_k to worker processors

We can see an obvious extension of this parallel inference framework

illustrated in Algorithm 3 for Bayesian models like the infinite mixture model or the infinite latent factor model where we sample the "local" variables Z in parallel and update the latent features given Z on the master processor. This type of parallelism is most amenable when Bayesian non-parametric models are represented in their stick-breaking formats. In the stick breaking setting, we explicitly represent the mixing measure π which means the posterior of z_i is conditionally independent given π and the latent feature ϕ_k.

If we integrate out mixing measure, then $P(z_i = k|-)$ will depend on the number of observations allocated to feature k and therefore not parallelizable unless we allow for excessive processor communication. If we can integrate ϕ_k from the likelihood then the likelihood will depend on all observations and produce the same processor communication problem. But, instantiating these parameters will lead to a MCMC sampler that mixes slower even though parallel inference is possible. Chapters 3 and 4 will demonstrate methods developed to overcome this exact trade-off between the speed of an uncollapsed sampler (meaning the mixing weights and features are instantiated) and a collapsed sampler (meaning the mixing weights and features are marginalized) with a parallel MCMC inference algorithm that contains the quality of both types of samplers.

2.6 Further Reading

The interested reader who intends to devote serious time to the study of Bayesian modeling, Bayesian non-parametrics and inference should read

the following books, articles and theses each of which were influential to the creation of this dissertation. As a introduction to Bayesian statistics and modeling, Gelman et al.'s *Bayesian Data Analysis* [26] is an excellent first read to understanding the intuition and power of modeling data under the Bayesian philosophy. The obvious next direction is to learn about Bayesian inference and computation. For Monte Carlo and Markov chain Monte Carlo techniques, Robert and Cassella's *Monte Carlo Statistical Methods* [77] and Brooks et al.'s *Handbook of Markov Chain Monte Carlo* [13]. Angelino et al. [3] extends the disucssion of Bayesian inference for scalable and distributable methods.

As a gentle introduction to Bayesian non-parametrics in practical modeling scenarios, Gershman and Blei's introduction to Bayesian non-parametrics [29] is an easy and enjoyable tutorial for those interested understanding the motivation to approach modeling from an infinite dimensional perspective. Rasmussen and Williams wrote a comprehensive book introducing Gaussian processes [76]. To understand Bayesian non-parametrics with more technical depth, Emily Fox's PhD thesis [23] and Peter Orbanz's lecture notes on Bayesian non-parametrics [69] offer a readable yet theoretically detailed overview of introductory ideas in BNP.

Chapter 3

Distributed Inference in Bayesian Nonparametric Models

In this chapter, we focus on two popularly used nonparametric Bayesian methods, the Dirichlet process and the Indian buffet process. We choose these because they are the most commonly used nonparametric priors for mixture models and latent feature models, respectively. However, they are both members of a larger class of models based on a Completely Random Measure framework, and the techniques in this chapter can easily be extended to apply to several other members of this class.

3.1 Completely Random Measures

Many commonly used nonparametric priors, including the Dirichlet process and the Indian buffet process, can be expressed in terms of a class of distributions known as completely random measures (CRM) [54]. A completely random measure is a random measure μ on some space Ω where the masses $\mu(A_i), \mu(A_j)$ assigned to disjoint subsets $A_i, A_j \subset \Omega$ are independent.

A commonly used example of a completely random measure is the gamma process. If H is some probability measure on Ω, and $\alpha > 0$, then

the gamma process assigns a Gamma($\alpha H(A)$) mass to any subset $A \subset \Omega$. The Dirichlet process is a distribution over random probability measure $D \sim \mathrm{DP}(\alpha, H)$ on Ω that corresponds to the normalization of a gamma process with parameters α and H. Since the gamma process assigns independent gamma-distributed masses to disjoint subsets of Ω, the masses assigned by a Dirichlet process to a finite partition of Ω are necessarily Dirichlet distributed, i.e. if $A_1, \ldots, A_K$ is a partition of Ω, then $(D(A_1), \ldots, D(A_k)) \sim \mathrm{Dirichlet}(\alpha H(A_1), \ldots, \alpha H(A_K))$.

We can use a Dirichlet process-distributed random measure

$$D := \sum_{k=1}^{\infty} \pi_k \delta_{\theta_k} \tag{3.1}$$

to assign parameters to observations:

$$\begin{aligned} D \sim & \mathrm{DP}(\alpha, H) \\ \phi_n \sim & D \end{aligned} \tag{3.2}$$

The discrete nature of D means that a single atom $\pi_k \delta_{\theta_k}$ can be associated with multiple observations, so we will get repeated values of ϕ_n, so that we are partitioning observations into an unbounded number of clusters. If we let z_n be the cluster indicator for the nth data point, we can directly obtain the predictive distribution over $z_{n+1} | z_{1:n}$:

$$P(z_{n+1} = k | z_{1:n}, \alpha) = \begin{cases} \frac{m_k}{n+\alpha} & \text{for } k \leq K \\ \frac{\alpha}{n+\alpha} & \text{for } k = K+1 \end{cases} \tag{3.3}$$

where K is the number of clusters present in the first n observations, and m_k is the number of observations in the kth cluster. To recover a full mixture

model from Equation 3.3, we can sample a cluster parameter $\theta_k \sim H$ for each cluster and set $\phi_n := \theta_{z_n}$.

Another commonly used CRM is the beta process [41, 91]. The (homogenous) beta process is a completely random measure where the distribution over atom sizes is given by the limit, as $K \to \infty$, of a Beta $\left(c\frac{\alpha}{K}, c\left(1-\frac{\alpha}{K}\right)\right)$ distribution, and whose atom locations are i.i.d. according to some probability measure H. The beta process can be used as a prior for latent feature models. If $B := \sum_{k=1}^{\infty} \mu_k \delta_{\theta_k} \sim \mathrm{BP}(\alpha, c, H)$, then we can interpret the μ_k as the probability of an observation containing the kth latent feature θ_k. We can select a finite subset of these latent features for the nth data point by sampling a sequence Z_n of binary indicator variables $z_{n,k} \sim \text{Bernoulli}(\mu_k)$.

As with the Dirichlet process, it is possible to integrate out the latent measure and work directly with the predictive distribution over the binary sequences. If K is the number of features seen in the first $n-1$ data points, then we can sample the next sequence $Z_n = (z_{n,k})_{k=1}^{\infty}$ as

$$\begin{aligned} z_{n,k} &\sim \text{Bernoulli}\left(\frac{m_k}{n}\right), \qquad k = 1, \ldots, K \\ J &\sim \text{Poisson}\left(\frac{\alpha}{n}\right) \\ z_{n,j} &= 1, \qquad j = K+1, \ldots, K+J \\ z_{n,j} &= 0, \qquad j > K+J \\ K &\leftarrow K+J \end{aligned} \tag{3.4}$$

If we stack the sequences Z_n into a matrix $\mathbf{Z}$, Equation 3.4 describes a distribution over binary matrices known as the Indian buffet process [36].

3.2 Inference Approaches

In Section 3.1, we saw two complementary representations for the cluster/latent variable allocations obtained using a Dirichlet process and an Indian buffet process. We can either explicitly instantiate the latent measure and sample each data point's allocations independently given this measure, or we can integrate out the latent measure and work directly with the predictive distribution.

When designing an MCMC algorithm, these two options lead directly to two different inference approaches, which we will refer to as *collapsed* and *uncollapsed* samplers. In an uncollapsed sampler, we alternate between sampling the random measure given the cluster/latent feature allocations, and sampling the allocations given the random measure. To avoid having to instantiate the infinitely many atoms of the random measure, we can either replace the random measure with a finite-dimensional approximation (see [48] for the Dirichlet process or [71, 103] for the Indian buffet process), or we can construct a slice sampler that employs a random truncation that maintains the correct posterior distribution (see [96, 25] for the Dirichlet process or [89] for the Indian buffet process).

A collapsed sampler (see [46, 66] for the Dirichlet process and [36] for the Indian buffet process) integrates out the random measure, and works only with the cluster allocations (Dirichlet process) or latent feature allocations (Indian buffet process). Since we only observe a finite number of clusters or latent features, we do not need to introduce approximations or slice vari-

ables. We make use of the fact that the sequences obtained by integrating out the random variables are exchangeable, meaning we can adapt the predictive distributions in Equations 3.3 and 3.4 to give a conditional distribution $P(Z_n|Z_{-n})$. This can be combined with a likelihood term $P(X_n|Z_n, \Theta)$ to give the full conditional distribution, which can in term be used to construct a Gibbs sampler.

3.3 Distributed Inference Methods

A number of parallel inference algorithms have been proposed for the Dirichlet process and its variants. [83] proposed an approximate distributed scheme for the hierarchical Dirichlet process, which is easily adaptable to the Dirichlet process mixture model. On each local step, each node assumes that the sufficient statistics from the other nodes are unchanged from the previous global step. On global steps, the sufficient statistics are updated and new clusters are (approximately) aligned. As [102] showed, this approach works well where the clusters are large, but when working with small clusters suffers from alignment issues, due to both problems matching up new clusters, and the possibility of small clusters drifting in location over the course of a local iteration.

[102] proposed a distributed method based on partitioning the Dirichlet process into a mixture of conditionally independent Dirichlet processes. These conditionally independent Dirichlet processes are updated independently on separate processors during local steps. Global moves are used to move data

between the conditionally independent Dirichlet processes to ensure correct mixing. While this approach works well for relatively small numbers of processors, its scalability is limited by the fact that each cluster in the overall DPMM must reside on a single processor (as discussed by [102] and [24]). Further, the approach assumes a shared-memory architecture where there is minimal cost to moving data between processors; in a distributed-memory context this would cause significant communication bottlenecks.

A more recent approach, that is explicitly designed for the distributed-memory, low-communication setting, was proposed by [25]. Unlike the previously described approaches, this paper uses an uncollapsed approach, explicitly instantiating the cluster probabilities and parameters. A slice sampler is used to ensure only a finite number of atoms need to be represented. When new atoms are introduced by the slice sampler, a shared seed is used in the pseudo-random number generators on each node, to ensure that the same atoms are proposed without the need for direct communication.

The downside of this approach is that, by necessity, the new atoms are sampled from the prior, rather than from their conditional distribution given observations. In high dimensions, this means that the proposed atoms are likely to be very far from data, and will therefore tend not to be used, resulting in slow mixing.

Compared with the Dirichlet process, there has been relatively little work on distributed inference for the Indian buffet process. The main contribution in this area is by [20], who deploy an approach similar to that of

[83]. Each processor approximates the current feature counts from the other processors with the counts from the previous time step; there is no explicit merging of new features. As we will see in Section 3.5, this approach will lead to over-estimation of the number of new features, and poor mixing.

3.4 Hybrid Algorithms for Distributed Inference

A key goal of a distributed algorithm is to minimize communication between agents. This can be achieved by breaking the algorithm into independent sub-algorithms, which can be run independently on different agents. In practice, we usually cannot split an MCMC sampler on a Bayesian hierarchical model into entirely independent sub-algorithms, since there are typically some global dependencies implied by the hierarchy. We can either replace these global dependencies with appropriate approximations, or we can make use of conditional independencies to temporarily partition our algorithm. In this paper, we describe methods that take the latter approach.

In Section 3.2, we considered two inference paradigms: collapsed and uncollapsed samplers. Both these approaches lead to difficulties when attempting to parallelize inference. In the collapsed setting, we face the problem that, since the cluster probabilities and parameters are marginalized out, the probability of the nth data point belonging to the kth cluster depends on the cluster allocations of all other data points. In particular, we need up-to-date knowledge of the total number of data points in the kth cluster, and the sufficient statistics associated with that cluster's distribution. If the kth cluster is in-

stantiated on multiple machines, maintaining these statistics requires frequent global communication. Algorithms have been proposed that ensure all data points in a given cluster are associated with the same processor [102]; however this can lead to bottlenecks and limited scalability due to large clusters. Further, we replace near-constant communication about sufficient statistics with less frequent, yet bandwidth-heavy, reallocation of datapoints to different processors. Another approach is to approximate the true counts with "stale" values, in effect assuming counts on other agents have not changed [20]; however this introduces errors and suffers from alignment issues, particularly in the nonparametric section where the number of clusters changes from iteration to iteration [102].

On the surface, uncollapsed samplers are much more suited to the distributed setting. We can make use of the fact that, conditioned on the latent measure, the cluster/latent feature allocations are conditionally independent. This means that if we split our data between the available agents and send a copy of our latent measure to all these agents, then the agents can independently sample the allocations for their subset of the data. Global communication is then required to sample from the conditional distribution of the random measure given the allocations. Parallelization in an uncollapsed representation for the Dirichlet process has been proposed by [25].

When working with an uncollapsed representation, we need a way of introducing new features. One option is to use a random slice variable, and sample a set of a atoms that are above that slice [96, 25, 50]. Another option

is to combine the probabilities for all the uninstantiated clusters into one, and sample a set of auxiliary variables from the prior that act as proposal locations for new clusters [66]. The performance of such algorithms will depend on how close the proposed new atoms or auxiliary variables are to the true cluster parameters. For a low-dimensional parameter space, we are likely to have reasonably good proposals; however as the dimensionality increases our proposals are unlikely to be near the data.

In this section, we propose a hybrid approach that offers the advantages of both a collapsed and an uncollapsed representation. We are able to do this because of the complete randomness of the underlying random measure (beta process in the case of the Indian buffet process; gamma process in the case of the Dirichlet process). This allows us to split the prior into two independent (in the case of the IBP) or conditionally independent (in the case of the DP) random measures. We choose to split the prior into a finite-dimensional measure corresponding to the currently observed clusters/features, and an infinite-dimensional "tail". We use uncollapsed inference on the finite measure, allowing straightforward parallelization but avoiding ever needing to expand our representation. For the infinite tail, we use a collapsed representation that allows us to efficiently introduce new features even in a high-dimensional setting. We present a hybrid sampler for the Dirichlet process in Section 3.4.1, and for the Indian buffet process in Section 3.4.2.

3.4.1 Distributed Hybrid Inference in the Dirichlet Process

Assume we wish to perform inference in a Dirichlet process mixture model with some arbitrary mixture kernel $f(\theta)$ parametrized by $\theta \in \Omega$. We can write this model as

$$D := \sum_k \pi_k \delta_{\theta_k} \sim \mathrm{DP}(\alpha, H),\ \phi_n \sim D,\ x_n \sim f(\phi_n). \tag{3.5}$$

Let A be some subset of Ω (where A may have measure zero) and let A^c be its complement in Ω. We can represent this Dirichlet process as the weighted superposition of two independent Dirichlet processes, one on A and one on A^c. Concretely, if $H|_A$ is the restriction of H to A, i.e.

$$H|_A(d\theta) = \begin{cases} H(d\theta) & \theta \in A \\ 0 & \theta \notin A \end{cases} \tag{3.6}$$

then we can represent the $\mathrm{DP}(\alpha, H)$ mixture model described in Equation 3.5 as

$$\begin{aligned} B \sim &\mathrm{Beta}(\alpha H(A), \alpha H(A^c)) & \\ G_1 \sim &\mathrm{DP}(\alpha H|_A) & \phi_n \sim &BG_1 + (1-B)G_2 \\ G_2 \sim &\mathrm{DP}(\alpha H|_{A^c}) & x_n \sim &f(\phi_n) \end{aligned} \tag{3.7}$$

Conditioned on the current cluster counts m_k, the posterior distribution over D is given by

$$D|m_1, \ldots, m_K \sim \mathrm{DP}\left(\alpha + N, \frac{\alpha H + \sum_k m_k \delta_{\theta_k}}{\alpha + N}\right) \tag{3.8}$$

Following from Equation 3.7, we can re-write this as

$$\begin{aligned} B \sim & \text{Beta}(N, \alpha) \\ G_1 = \sum_{k=1}^{K} \pi_k \delta_{\theta_k} \sim & \text{DP}\left(N, \frac{\sum_k m_k \delta_{\theta_k}}{N}\right) \qquad \phi_n \sim BG_1 + (1-B)G_2 \\ & \qquad\qquad\qquad\qquad\quad x_n \sim f(\phi_n) \\ G_2 = \sum_{k=K+1}^{\infty} \pi_k \delta_{\theta_k} \sim & \text{DP}(\alpha, H) \end{aligned} \tag{3.9}$$

We note that $G_1 = \sum_{k=1}^{K} \pi_k \delta_{\theta_k}$, where K is the number of currently occupied clusters, and $(\pi_1, \ldots, \pi_K) \sim \text{Dirichlet}(m_1, \ldots, m_K)$. We have partitioned our Dirichlet process into a weighted combination of a finite-dimensional Dirichlet distribution, with elements corresponding to the currently occupied clusters, and an infinite-dimensional Dirichlet process, with atoms corresponding to the currently unoccupied clusters.

We can now instantiate the finite measure G_1 on all processors, and integrate out the infinite dimensional tail. We randomly select one out of P processors, by sampling $P^* \sim \text{Uniform}(1, \ldots, P)$. For every other processor, i.e. for each processor $j \neq P^*$, we perform restricted Gibbs sampling [66], only allowing observations to choose between the K clusters in G_1.

On processor P^*, we allow observations to pick a cluster from G_1 with probability B, or a cluster from G_2 with probability $1 - B$. Concretely, the probability a data point x_n on cluster P^* being assigned to cluster k, conditioned on B, G_1 and the other data points on processor P^*, is given by

$$P(\phi_n = \theta_k|-) \propto \begin{cases} B\pi_k f(x_n;\theta_k) & k \leq K \\ \frac{(1-B)m_k}{N} f(x_n; \{x_i : \phi_i = \theta_k, i \neq n\}) & K < k \leq K+J \\ \frac{(1-B)\alpha}{N} & k = K+J+1 \end{cases} \tag{3.10}$$

where J is the number of atoms in G_2 which are associated with data. Note that the only data points that can be associated with atoms in G_2 are those on processor P^*, so we can evaluate $\frac{m_k}{N} f(x_n; \{x_i : \phi_i = \theta_k, i \neq n\})$ without any knowledge about the other processors. At each global step, we gather the sufficient statistics from all instantiated clusters – from both G_1 and G_2 – and sample parameters for those clusters. We then create a new partition, redefining the support of G_1 as the set of instantiated cluster parameters, and resample $B \sim \text{Beta}(N, \alpha)$.

While asymptotically correct, an unfortunate consequence of this sampler is that it is slow to instantiate new clusters. With only $1/P$ of the data points eligible to start a new cluster, the rate at which new clusters are added will decrease with the number of processors. While this is of less concern once we have converged to an appropriate number of clusters, it can lead to slow convergence if we start with too few clusters. To avoid this problem, we initialize our algorithm by allowing *all* processors to instantiate new clusters. At each global step, we decrease the number of randomly selected processors eligible to instantiate new clusters, until we end up with a single processor. This warm start allows us to expand our initial state space with data-driven cluster proposals.

We note that a sampler with multiple processors instantiating new clusters will not converge to the true posterior; instead it will tend to over-estimate the number of clusters. However, the procedure acts in a manner similar to simulated annealing, by encouraging large moves early in the algorithm but gradually decreasing the excess stochasticity until we are sampling from the correct algorithm.

3.4.2 Distributed Hybrid Inference in the Indian Buffet Process

If $B \sim \mathrm{BP}(\alpha, c, H)$ and $Z_n \sim \mathrm{BeP}(B)$, then the posterior distribution over B is given by

$$B|Z_1, \ldots, Z_n \sim \mathrm{BP}\left(\frac{c\alpha + \sum_k m_k}{c+n}, c+n, \frac{c\alpha H + \sum_k m_k \delta_{\theta_k}}{c\alpha + \sum_k m_k}\right) \tag{3.11}$$

Since the beta process is a completely random measure, we can partition this into the superposition of two independent completely random measures, so that

$$\begin{aligned} B_1 := \sum_{k=1}^{K} \mu_k \delta_{\theta_k} \sim & \mathrm{BP}\left(\frac{\sum_k m_k}{c+n}, c+n, \frac{\sum_k m_k \delta_{\theta_k}}{\sum_k m_k}\right) \\ B_2 := \sum_{k=K+1}^{\infty} \mu_k \delta_{\theta_k} \sim & \mathrm{BP}\left(\frac{c\alpha}{c+n}, c+n, H\right) \\ B = & B_1 + B_2 \end{aligned} \tag{3.12}$$

We note that the distribution over the atom sizes $\mu_1, \ldots, \mu_k$ of B_1 is simply a sequence of K $\mathrm{Beta}(N - m_k, m_k + c)$ random variables. This allows

us to split the IBP into two independent feature selection mechanisms: one (controlled by B_1) with a finite number of currently instantiated features, and one (controlled by B_1) with an unbounded number of currently uninstantiated features.

This observation allows us to construct a distributed MCMC sampler where, at any given time, only data on a single processor is allowed to sample from the full conditional distribution over features. On all the other processors, data points can only use features from B_1. Ergodicity is ensured by periodically re-defining B_1 to include all instantiated features, and randomly resampling the single processor that is able to instantiate new features.

Concretely, at each global iteration, we randomly select one processor with indicator $P^* \sim \text{Uniform}(1, \ldots, P)$. For every other processor, i.e. for each processor $j \neq P^*$, we perform restricted Gibbs sampling [66], only allowing data points to select subsets of the K features in B_1. The probability that $z_{nk} = 1$ for such a data point is given by:

$$P(z_{nk} = z|-) \propto \begin{cases} \mu_k f(x_n|z_{nk} = 1, z_{n,-k}, \theta_1, \ldots, \theta_k) & z = 1 \\ (1 - \mu_k) f(x_n|z_{nk} = 0, z_{n,-k}, \theta_1, \ldots, \theta_k) & z = 0. \end{cases} \tag{3.13}$$

On processor P^*, data points can select features from B_1 or B_2. Let K be the number of atoms in B_1, and let J be the number of instantiated features in B_2. The first K features are selected according to Equation 3.13. If we are able to marginalize over the feature locations θ_k, the next J features are selected according to

$$P(z_{nk} = z|Z_{-nk}, x_n) \propto \begin{cases} m_k f(x_n|z_{nk} = 1, Z_{-nk}) & z = 1 \\ (N - m_k) f(x_n|z_{nk} = 0, Z_{-nk}) & z = 0. \end{cases} \tag{3.14}$$

If we are unable to marginalize over the θ_k, we can instantiate them as described in [20] and include them in the appropriate likelihood term. Finally, we propose adding Poisson(α/N) new features, using a Metropolis-Hastings step.

At each global step, we gather the sufficient statistics from each instantiated feature, and sample a feature value θ_k for each one conditioned on the data points exhibiting that feature. We redefine B_1 and B_2 so that B_1 contains all (and only) instantiated features, and sample $\mu_k \sim \text{Beta}(m_k, N - m_k + c)$ for each feature in B_1. We then sample a new processor indicator $P^* \sim \text{Uniform}(1, \ldots, P)$ and repeat.

As with the Dirichlet process sampler described in Section 3.2, the distributed sampler will be slow to add features, since to ensure correctness of the transition distribution, only one processor can add features at a time. We can dramatically improve the time to convergence by using a warm-start procedure as described in Section 3.4.1, where we initially allow all processors to instantiate new features, and gradually reduce the number of processors adding new features until we have the correct sampling distribution.

3.5 Experimental Evaluation

We begin by showing that, while parallelizable, an entirely uncollapsed sampler is a poor choice when dimensionality increases. We go on to compare our distributed hybrid samplers with a range of other parallel methods for the DP and the IBP.

3.5.1 Limitations of an Entirely Uncollapsed Approach

In an entirely uncollapsed sampler, we must ensure that a global set of atom sizes and locations is shared across all processors. This means that we must sample new parameters from the prior. One method of doing so is obtained by modifying Algorithm 8 of [66]. Algorithm 8 describes a scheme for Gibbs sampling a Dirichlet process mixture with a non-conjugate likelihood, where we can integrate out the atom sizes but must sample the atom locations. We can modify this to give a fully uncollapsed algorithm, where both atom sizes and atom locations are instantiated.

At each global step, let K be the total number of occupied clusters, and let $\mathbf{m} = m_1, \ldots, m_K$ be the cluster counts. We can proceed as follows:

- Discard any atoms that are not associated with any data points, leaving only K instantiated atoms.
- Sample new atom locations $\theta_1, \ldots, \theta_K$ for the K atoms, using the conditional distribution given the associated data points.
- Sample J new atom locations $\theta_{K+1}, \ldots, \theta_J$ from the base measure, where $J \geq 1$.
- Sample atom weights for all $K + J$ atoms

$$\boldsymbol{\pi} := (\pi_1, \ldots, \pi_K, \ldots, \pi_{K+J}) \sim \text{Dirichlet}(m_1, \ldots, m_k, \alpha/J, \ldots, \alpha/J)$$

- For each data point x_n, sample a cluster indicator c_n according to

$$P(c_n = k|\boldsymbol{\pi}, \boldsymbol{\theta}) \propto \pi_k f(x_n; \theta_k)$$

  where $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_K, \ldots, \theta_{K+J})$.

The final, time-consuming step, where we sample the cluster indicators, can be parallelized.

Unfortunately, we can run into problems when it comes to proposing new parameter values $\theta_{K+1}, \ldots, \theta_{K+J}$. In high dimensions, it is unlikely that a proposed parameter will be near our data, so the associated likelihood of any given data point will be low. This is in contrast to the collapsed setting, where we integrate over all possible locations.

Figure 3.1 shows convergence plots for three algorithms: The uncollapsed algorithm described above; a standard collapsed Gibbs sampler; and the single-processor version of our hybrid algorithm. The data set is a D dimensional synthetic data set consisting of 100 observations of Gaussian mixtures with 2 true mixture components centered at $5 \times \{1\}^D$ and $-5 \times \{1\}^D$ with an identity covariance matrix. We can clearly see that the collapsed sampler performs better than the uncollapsed sampler for 10 dimensional data. Since the hybrid sampler only uses collapsed sampling for newly introduced features, the performance of the hybrid sampler in this situation is expected to be similar to the uncollapsed sampler although the hybrid sampler reaches its maximum F1 score faster than the uncollapsed sampler.

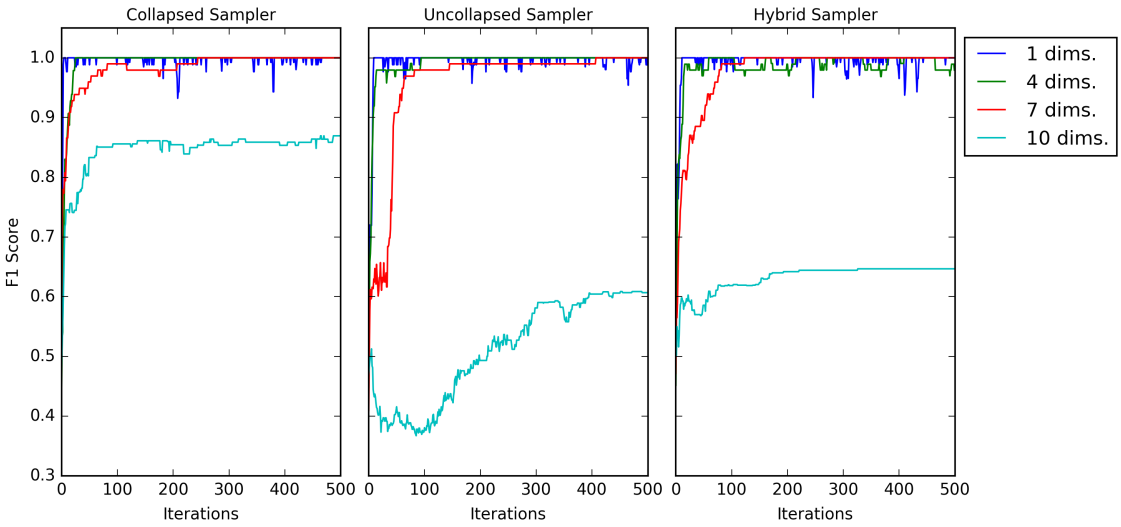

Figure 3.1: Comparison of F1 scores over iteration for the collapsed, uncollapsed and hybrid samplers

3.5.2 Experimental Evaluation: Dirichlet Process

In this section, we show how our inference algorithm can speed up inference in a Dirichlet process mixture of normals. To evaluate our algorithm, we generated parameter weights given the α parameter from the stick breaking Dirichlet process prior. Then, we sample locations for the clusters from a Normal-Inverse Wishart prior and for n observations we sample a cluster indicator from the cluster weights and then sample from the cluster parameter. Our experiment shows the F1 score of test set data as the number of processors increases. As we can see in Figure 3.2, our hybrid method is capable of achieving a higher F1 score faster than the lower processor runs.

Next, we evaluate the performance of our DPMM sampler by adjusting

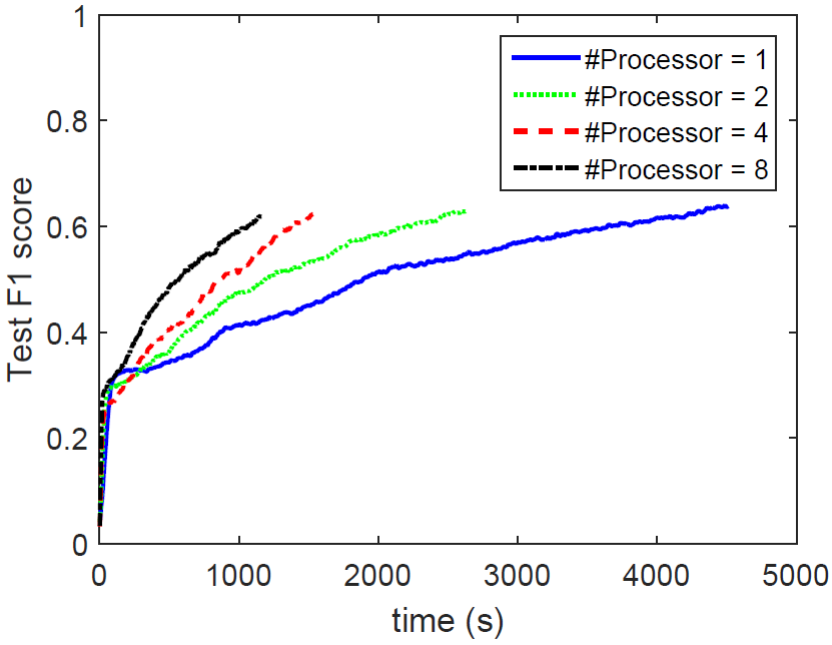

Figure 3.2: F1 score for test set synthetic data.

the separability of the true cluster locations. Intuitively, we observe that our algorithm performs poorly when there is little separation between the clusters (Figure 3.3) and performs well when there is large separation between clusters (Figure 3.4).

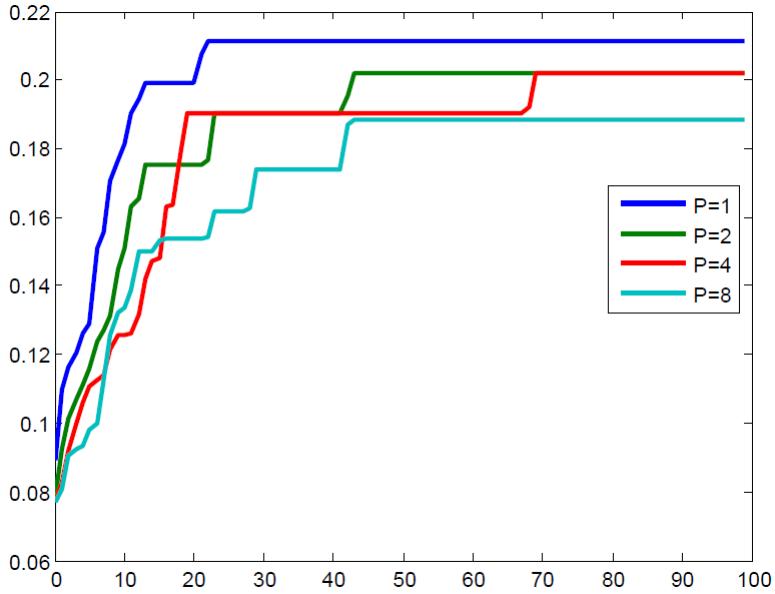

Figure 3.3: F1 score over iterations for synthetic data set with small separation

3.5.3 Experimental Evaluation: Indian Buffet Process

Next, we show how our inference algorithm can speed up inference in an Indian buffet process latent feature model. We use a linear Gaussian likelihood, modeling the data as

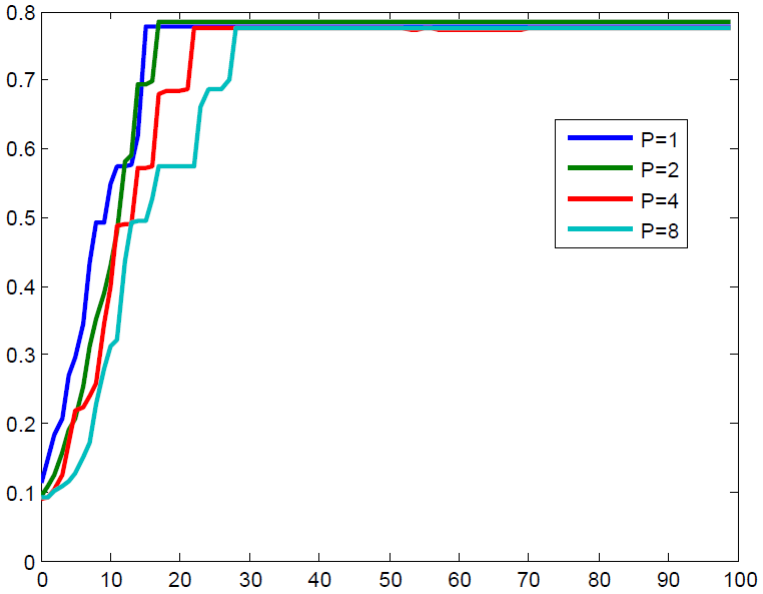

Figure 3.4: F1 score over iterations for synthetic data set with large separation

$$\begin{aligned} Z &\sim \text{IBP}(\alpha, 1) \\ A_k &\sim \text{Normal}(0, \sigma_A^2 \mathbf{I}) \\ X_n &\sim \text{Normal}\left(\sum_k z_{nk} A_k, \sigma_X^2 \mathbf{I}\right) \end{aligned} \tag{3.15}$$

We evaluated the model on a synthetic data set consisting of 10,000 observations. This dataset was an extension of the "Cambridge" dataset, used in the original IBP paper [37], where each data point contains a randomly selected subset of 4 binary features, plus Gaussian noise ($\sigma_X = 0.5$). In the IBP experiments, we ran the hybrid sampler for 1000 total observations with a synchronization step every 5 iterations and we run the hybrid sampler for 4, 8, 16, 32, 64 and 128 processors.

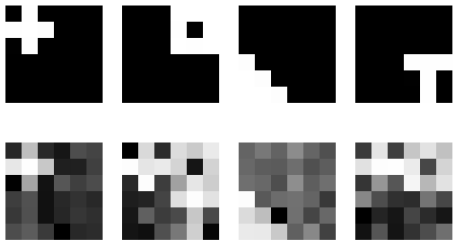

Figure 3.5: Top: The true features present in the synthetic data set. Bottom: Examples of observations in the synthetic data set.

We run the Hybrid IBP sampler under a "cold start", where only one processor is allowed to introduce new features for the entire duration of the sampler. We can see that the cold start results in the test set log likelihood of the higher processors failing to converge properly (Fig. 3.6). Since the number of features in each experiment is fixed at 2, we observe that the sampler, over 8 processors, accepts few features than in subsequent examples (Fig. 3.7).

Next, we evaluated the effect of warm-start initialization, where initally all processors could propose new features; we reduced the number of processors able to add new features by 0.99 at each global step. Figure 3.8 shows the predictive log likelihood over time (shown on a log scale), for 4, 8, 16, 32, 64, and 128 processors. Clearly, the convergence rate for high processor trials is better than in the cold-start trials and the number of features is generally close to the true number of features, 4. Additionally, we can see that the 128 processor run converges the fastest and all the other processor runs converge in descending order of number of processors.

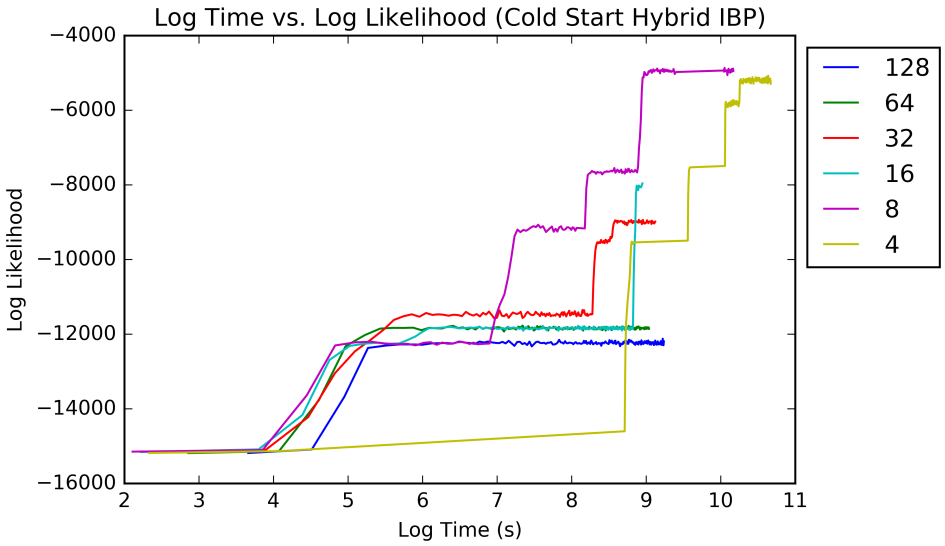

Figure 3.6: Test set log likelihood on synthetic data without warm-start initialization.

Finally, we allowed all the processors to propose new features for the entire duration of the sampler ("always-hot"). Using the same experimental synthetic data scenario described earlier, we can see that all the processor runs roughly converge to the same test log likelihood. However, the number of features introduced is much greater than the warm start experiment, and the number of features introduces as the number of processors increase too. Moreover, the difference in convergence rates between processors is not as dramatic as in the warm-start trials. The results of the always-hot trials approximately replicate the behavior of the parallel IBP sampler in [20].

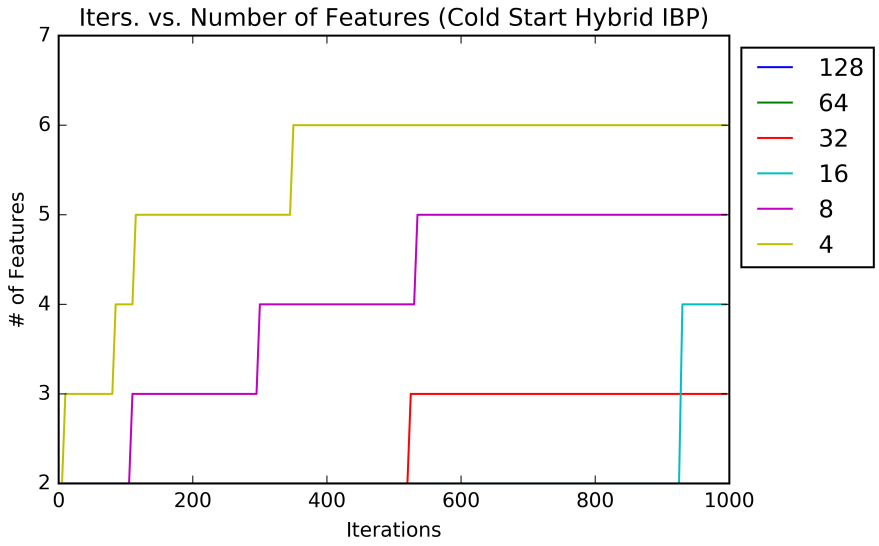

Figure 3.7: Number of features over iterations for synthetic data without warm-start initialization.

3.6 Discussion

As seen in the previous experiments, we now have a general strategy of parallelizing inference for a potentially wide class of Bayesian nonparametric models. Due to the conditional independence between the infinite dimensional latent components in the BNP models considered in this paper, we can partition the latent components into the finite-sized instantiated partition and the infinite-sized uninstantiated partition. We can take advantage of the inherent parallelizability of the uncollapsed sampler on the finite partition, which performs adequately for popular features. But collapsed sampling will perform better for proposing new components and allocating observations to newly added components. Thus, we restrict collapsed sampling only to the infinite

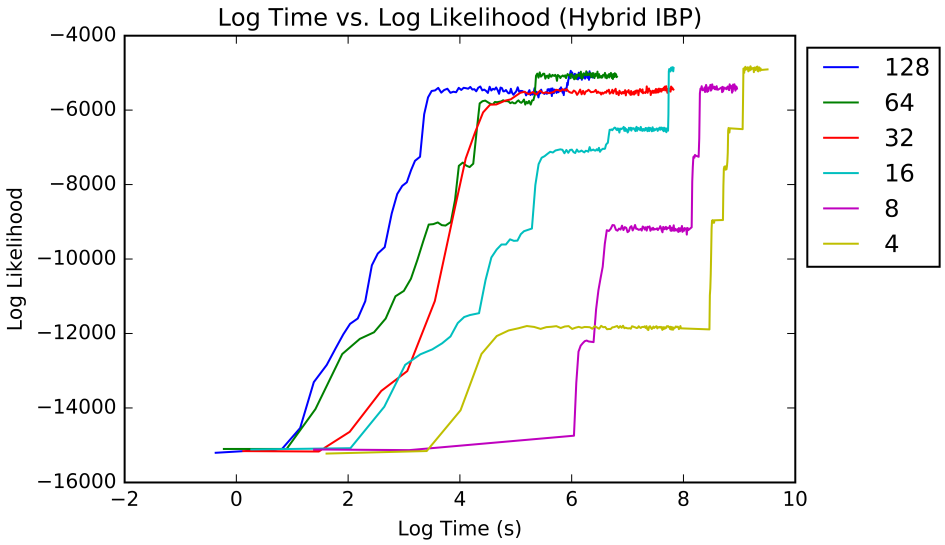

Figure 3.8: Test set log likelihood on synthetic data with warm-start initialization.

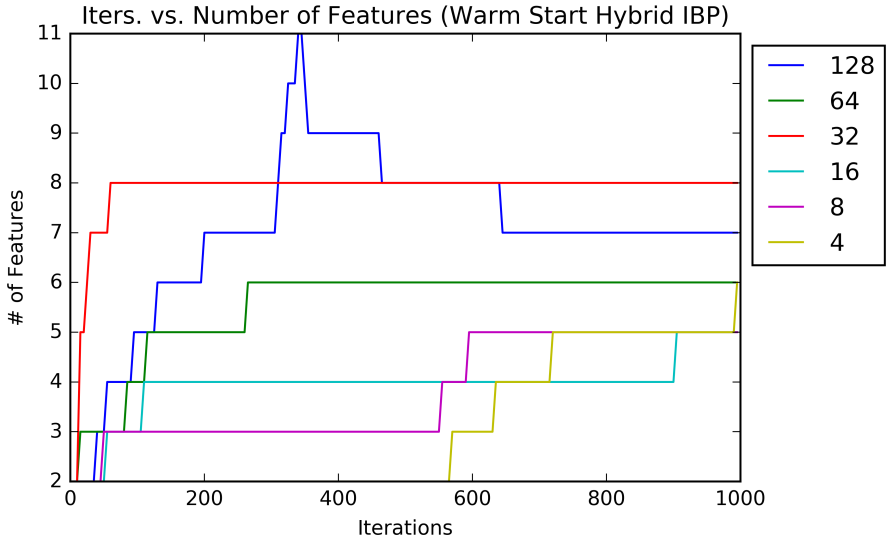

Figure 3.9: Number of features over iterations for synthetic data with warm-start initialization.

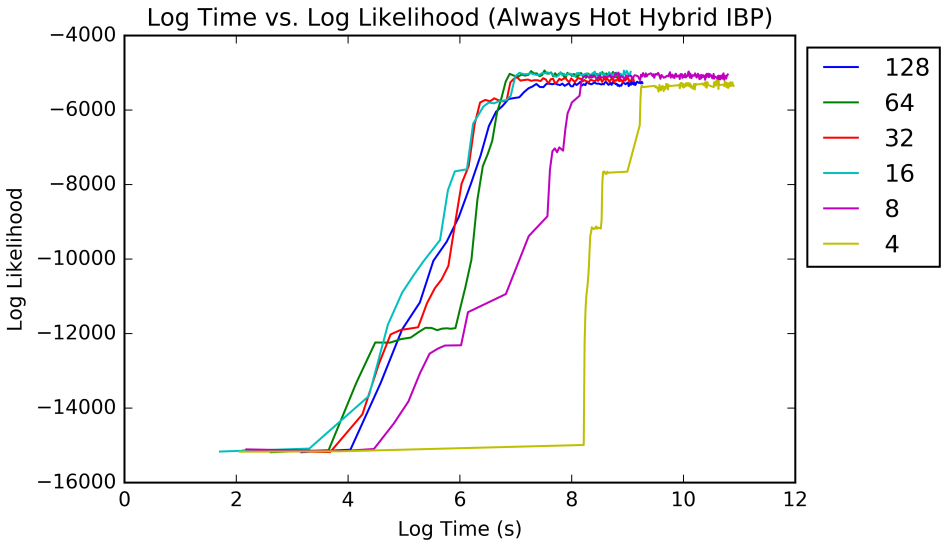

Figure 3.10: Test set log likelihood on synthetic data with all processors introducing features.

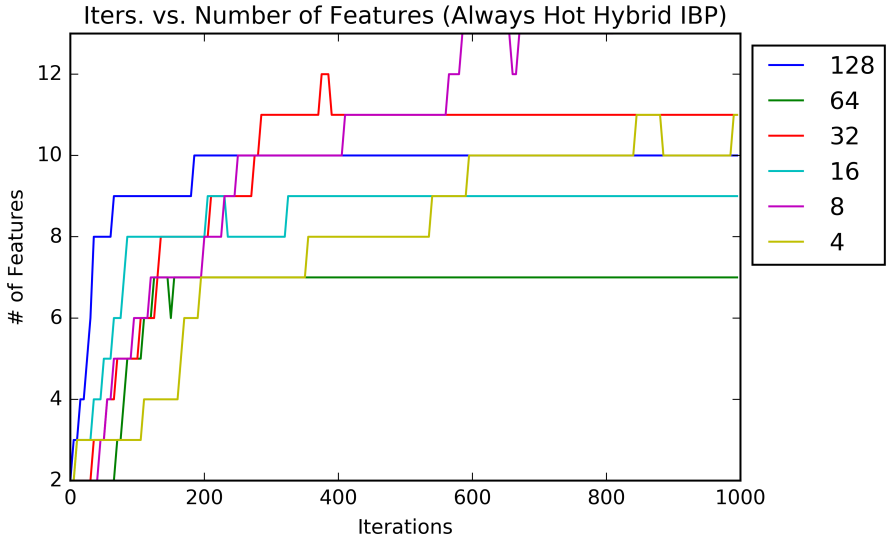

Figure 3.11: Number of features over iterations for synthetic data with all processors introducing features.

dimensional partition of the latent components. After partitioning the data across P processors, each processor will independently sample the allocation of the latent components to observations of the data and on a global step, a master processor will gather summary statistics from each machine and send new features and posterior values for parameters to all other machines.

In a distributed setting, we must restrict collapsed sampling to only one processor to have a valid MCMC algorithm. But we have seen that convergence will generally be poor under a large number of processors because the sampler can only propose new features on $1/P$-th of the data. To overcome this issue, we suggest using a "warm-start" procedure where all processors may introduce new features and we gradually reduce the number of processors introducing features each global step until only one processor may perform collapsed sampling.

One of the major issues regarding MCMC inference methods is that they are generally slow, especially as the size of the data increases. Big data is an increasingly important concern for machine learning tasks because the nature of the data available now has grown to such a massive size that the scalability of an algorithm needs to be a primary concern in developing machine learning tools. Inference in the Bayesian nonparametrics, especially for the IBP, has generally been difficult but we have developed an inference algorithm that has made BNP models amenable for huge data sets.

Chapter 4

Accelerated Inference for Latent Variable Models

Bayesian nonparametrics (BNP) models appear to be perfectly suited for the era of big data [49], in which ever-expanding databases of high-dimensional data cannot be dealt with simplistically. Generative processes priors like the Dirichlet process [22] or the Indian buffet process [37] allow for modeling latent variables like clusters or otherwise unobservable features in our data and adapting the complexity of the model in accordance to the complexity of the data. Even if we had some understanding of the latent structure in the data, we would not necessarily know their exact forms and implications in the model *a priori*. The BNP solution, which divides the data into discrete features and clusters, fosters interpretable models that would naturally lead to new hypotheses about the information in such databases [52]. For example, in a general medical records dataset containing billions of observations, a cluster (or feature) composed of 0.001% of the population still includes tens of thousands of people. But in looking at a small fraction of the data, these clusters may be seen as outliers at best and it would be difficult to characterize their meaning. Exploring large, complicated databases with the intention of finding interpretable new features by adapting the model complexity to the complexity

of the data is BNP's most prominent and promising feature. Unfortunately, BNP fails to deliver on this promise for at least three intrinsic limitations:

1.) Base measure definition: The base measure divides the available probability mass over the whole input space for a predefined likelihood model. It controls which models are feasible and which are not. For high-dimensional data, we would need to spread the probability mass thin even if we had the "right" base measure, which might prevent finding the correct representation of our data. When we do not, the base measure can be better understood as a regularizer to avoid the curse of dimensionality than a distribution that encodes all our prior knowledge about the problem at hand [27].

2.) Shallow likelihood models: Likelihood models are typically hand-engineered features that simplistically pass information about the parameters from the prior to the observable data. These simple likelihoods limit the type of information that can be represented and cannot capture high level complex features in images or audio, for example. We need different likelihood models, which should be learned from the data, to be able to better characterize the relation between the observed data and an interpretable model.

3.) Computational complexity: Inference in BNP models cannot be readily parallelized because traditional collapsed sampling methods and variational approximations requires processing the data sequentially. Uncollapsed sampling can be parallelized because the sufficient statistics are explicitly defined, but finding good features by sampling from a high-dimensional base measure is improbable.

This chapter will propose a method for addressing the first and last limitation. The second limitation is also relevant and there are many papers working toward that goal: Variational autoencoders [53], adversarial variational Bayes [61] or deep hierarchical implicit models [93], among many others. But, incorporating them is not straightforward in our current inference strategy.

The main idea in this paper is an algorithm that relies on an adaptation of Neal's Algorithm 8 [66] together with a parallelizable procedure in which every few iterations the different nodes share summary statistics and newly introduced features. To improve convergence, instead of sampling from the base measure, we propose a simple mechanism that samples from the data points that are not well characterized by their current clusters. With this method, we are able to find new cluster without needing to sample from the base measure. We illustrate the algorithms with high dimensional, high signal to noise ratio in which standard MCMC inference procedures for BNP models will fail to find any meaningful information and our algorithm is able to find relevant clusters.

4.1 Related Work

One of the most common examples of a latent variable model is the mixture model. In this chapter we will apply our technique to the Dirichlet process mixture model (DPMM) [4]

4.1.1 Dirichlet Process Inference

The Dirichlet process in particular is attractive because its marginal Dirichlet property allows for tractable inference. [66] provides an overview of different MCMC sampling techniques for the DPMM. Sampling methods for the Dirichlet process broadly fall into two categories: methods that integrate out the mixing parameter, π [66, 46], and methods that leave π instantiated [48, 96, 25]. Inference methods that allow π to be instantiated are inherently parallelizable since the cluster allocation probability is conditionally independent given the mixing proportion, but proposing new features under this setting is difficult due to the infinite dimension of the mixing proportion. On the other hand, integrating out π allows us to deal only with the cluster allocation and not the mixing proportion. This, in conjunction with a likelihood $P(X_i|\theta_{z_i})$, provides a full conditional distribution to be used in a Gibbs sampler. However, $\int P(Z|\pi,\alpha)P(\pi|\alpha)\,\mathrm{d}\pi$ means that the marginal distribution of z_i becomes dependent on all other cluster allocations z_{-i} which is unparallelizable without excessive and costly communication between processors.

Numerous distributed inference procedures have been developed for fast inference of DP models. [83] propose an asynchronous method for the hierarchical Dirichlet process [90] which distributes the data across P processors and performs Gibbs sampling based on approximate marginalized distribution of $z_i|z_{-i}$. [102] propose an exact sampler for the DPMM, but requires that each observation associated with a particular cluster k must exist on the same processor. Chapter 3 described an exact sampler for completely random mea-

sures (CRM) which exploits the conditional independencies of the features by partitioning the random measure into a finite instantiated partition and an infinite uninstantiated partition. The instantiated portion runs the inherently parallel sampler with the mixing parameter, π, instantiated and at random one processor is selected to sample and propose new features based on the predictive distribution of the cluster assignments.

Algorithm 4: Slice Sampling for the Dirichlet Process Mixture Model

for $i = 1, \ldots, N$ **do**
  $u_i \sim \text{Uniform}(0, \pi_{z_i})$
$u^* := \min_i u_i$
$K^* := K$
while $u^* \geq \pi^*$ **do**
  $K^* := K^* + 1$
  $\beta_{K^*} \sim \text{Beta}(1, \alpha)$
  $\pi_{K^*} := \pi^* \beta_{K^*}$
  $\phi_{K^*} \sim H$
  $\pi^* := \pi^*(1 - \beta_{K^*})$
for $i = 1, \ldots, N$ **do**
  $z_i \sim P(z_i|-) \propto \begin{cases} L(x_i, \phi_k) & \pi_k \geq u_i \\ 0 & \text{o.w.} \end{cases}$
for $k \in \{k : n_k > 0\}$ **do**
  $\phi_k \sim P(\phi_k|-)$
  $\beta_k \sim \text{Beta}\left(1 + n_k, \alpha + \sum_{\ell=k+1}^{K} n_\ell\right)$
  $\pi_k := \beta_k \prod_{\ell=1}^{k-1}(1 - \beta_\ell)$

4.1.2 Inferential Problems

However, none of these techniques deal with the problem of sampling good feature locations in high dimensional space. The distributed samplers

Algorithm 5: Neal's Algorithm 8

for $i = 1, \ldots, N$ **do**

  Sample $z_i \sim P(z_i = k|-) \propto \begin{cases} n_{-ik} L(x_i, \phi_{z_i}), & k \in K^+ \\ (\alpha/m) L(x_i, \phi_{z_i}), & k \in K^- \end{cases}$

Update $K^+ := \{k : n_k > 0\}$

for $k \in K^+$ **do**

  Sample $\phi_k \sim P(\phi_k|-)$

for $k \in K^-$ **do**

  Sample $\phi_k \sim H$

described above only seek to accomplish (approximately for [83]) correct inference. The method in Chapter 3 proposed to allow all processors to propose new features to encourage better mixing of the MCMC sampler before proceeding with exact inference and only allowing one processor to propose new features. But, the proposal of new features themselves poses a serious problem in the DPMM. In proposing new features, we can either propose features from an *uncollapsed* representation–meaning, drawing features from the prior, $P(\theta)$, and assign observations to clusters with likelihood $P(X|\theta)$. Or, we can marginalize out the clusters and sample cluster assignments from a *collapsed* likelihood, $\int P(X|\theta)P(\theta)\,\mathrm{d}\theta$.

In high dimensional settings, the uncollapsed sampler is likely to fail to find good locations for features. The collapsed sampler is more likely to sample better feature locations because it draws from the expectation of the likelihood with respect to the prior distribution on features but this requires us to obtain the collapsed distribution in closed form which, in general, is not available. Even if the collapsed distribution were available in closed form,

situations where the features occupy a sparse region of its domain will also lead the collapsed sampler to fail to find good features (we will demonstrate an example of this failure in the Section 4.3).

4.2 Method

Our novel sampling method allows for both distributed parallel sampling for the Dirichlet process and a better sampling and proposing method of new features. Essentially, we propose new features centered at observations that are poorly fit to its assigned cluster in parallel and after the accelerated portion of sampling we proceed with an inherently parallel uncollapsed sampler with instantiated mixing parameter π.

The sampling method first begins in an "accelerated" stage, where we draw proposals for features from the observations as opposed to the prior. In order to deal with the problem of proper mixing, we propose a method that samples locations for features from observations where its current latent feature assignment has low likelihood. As opposed to other DP samplers that sample feature locations from $P(\theta)$ or assign cluster allocations based on $\int P(X|\theta)P(\theta)\,\mathrm{d}\theta$, our accelerated sampler proposes features centered at observations with the lowest likelihood given its current feature allocation. In the Gaussian distributed examples in Section 4.3.1, we set the covariance of each proposal to be the covariance of its current allocated feature divided

by some constant greater than one[1]. Additionally, we write that the feature should be proportional to the data because the feature locations in the multinomial-Dirichlet examples in Section 4.3.2 have features proposed from the observations normalized so that the proposed feature sums to 1.

We further approximate inference for a DPMM during accelerated sampling with a modified version of Algorithm 8 [66] which introduces S auxiliary features θ_k for $k = 1, \ldots, S$ that represent finite realizations of the clusters. If N_k represents the number of observations assigned to cluster k then let K^+ represent $\{k : k \in (1, \ldots, S), N_k > 0\}$ and K^- represent $\{k : k \in (1, \ldots, S), N_k = 0\}$. Algorithm 8 assigns cluster assignments according to the following probability:

$$P(z_i = k|-) \propto \begin{cases} N_{-ik} \cdot P(X_i|\theta_k), & k \in K^+ \\ (\alpha/S) \cdot P(X_i|\theta_k), & k \in K^- \end{cases} \tag{4.1}$$

where N_{-ik} represents the number of observations besides observation i allocated to cluster k. Then, draw new realizations for $\theta_k, k \in K^-$ from H and update posterior values for $\theta_k, k \in K^+$.

In a single processor setting, we will always have the exact value of N_{-ik}. However, in the parallel setting we no longer have the precise feature counts, which forces inter-processor communication for every state transition of z_i. Therefore, we approximate N_{-ik} with M_{-ikp} which is the local count of cluster k on processor p excluding observation i when running the algorithm in parallel. [5] applies the same approximation for latent Dirichlet allocation

[1]For all examples seen in Section 4.3.1 we divide by 10

and the hierarchical Dirichlet process for their inference technique. This approximation works well for large clusters, as seen in [102] but we note that we only apply this approximation in the accelerated stage and not for the entire duration of the sampler in order to maintain the exactness of our method.

We assume that new features accepted on different processors are different from each other, thus we do not consider the problem of feature alignment. Furthermore, by dividing across multiple processors we can propose P times more features to explore possible new clusters that will persist after acceleration. After L subiterations of our sampler, we trigger a global synchronization step where each processor sends updated features, feature counts and feature summary statistics to instantiate new features on all processors and to update posterior values for global parameters.

This proposed accelerated stage of our sampler is obviously not a correct MCMC sampler for a DPMM, but because the correctness of an MCMC sampler is theoretically invariant to its starting position we first proceed with the accelerated sampler before switching to a correct sampler. We allow the sampler to initially run in this accelerated sampling mode for some partial duration after which we can either elect to sample from an easily parallelizable, finite Dirichlet mixture model with an instantiated mixing parameter: $P(z_i = k|-) \propto \pi_k P(X_i|\theta_k)$. Or, if we wish to continue with an asymptotically exact Dirichlet process sampler, the methods proposed either in [102] or Chapter 3. The major additional benefit of our accelerated sampler is that we have an efficient sampler to encourage fast mixing of the MCMC sampler with-

out the need to integrate the latent features out of the likelihood. Thus, we can now use a variety of priors for features without encountering the problem of proposing features from the prior in high dimensional space. Our method in the DPMM case is suitably general for a wide class of data modeling scenarios. Although the most common type of mixture is the Gaussian mixture model (GMM), we do not place any assumption on the form of likelihood for $P(X|\theta)$, and we will see an example of our method applied to count data to demonstrate the flexibility of this method.

4.3 Experimental Results

4.3.1 Location Clustering Example

To first demonstrate the basic utility of our sampler, we apply our method to an empirical 2 dimensional data set that tracked the cell phone coordinates of a psychiatric patient. Using a DPMM with a Gaussian-Inverse Wishart prior on the mean and covariance parameters and compare against the collapsed, uncollapsed and variational inference samplers:

$$\begin{aligned} &\theta_k = (\mu_k, \Sigma_k),\ P(X_i|\theta_{z_i}, z_i) \sim \text{Normal}(\mu_{z_i}, \Sigma_k), \\ &H \sim \text{Normal-Inv. Wishart}(\mu_0, \Psi, \lambda, \nu) \end{aligned} \tag{4.2}$$

We can see that the collapsed, accelerated, and variational inference[2] DP samplers are able to find many meaningful clusters whereas the uncollapsed sampler is only able to find 2 clusters. Because we are examining a 2 dimensional

[2]We do not have timing data for variational inference, so we only report the final log likelihood and number of clusters

data set, the collapsed sampler is able to sample mostly good locations and we can see that the variational method performs very well. Additionally, we also initialize the collapsed and uncollapsed samplers with random and K-means initializations to 100 clusters. Though the log likelihood predictive performance is good for these methods, the clusters that they produce besides the collapsed sampler initialized with K-means are qualitatively less meaningful than ones found in the accelerated, collapsed and variational inference sampler. We will see, in Section 4.3.2, empirical datasets where even the collapsed sampler is not able to find good feature locations in high dimensional space and where accelerated sampling performs comparably variational inference.

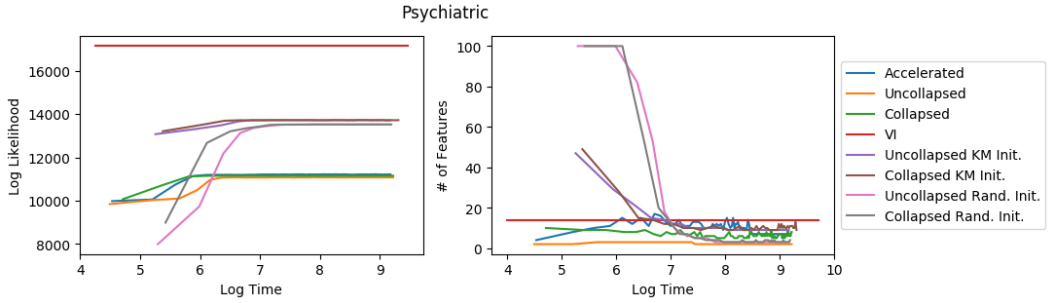

Figure 4.1: Test set predictive log likelihood vs. log time (seconds) and number of features vs. log time.

4.3.2 Image Data Sets

Next we will apply our accelerated inference technique on two large dimensional image data sets, the 28×28 dimensional MNIST handwritten digit dataset [57], consisting of a training set of 60,000 images and test set of 10,000 images and the 168×192 dimensional Cropped Extended Yale Face

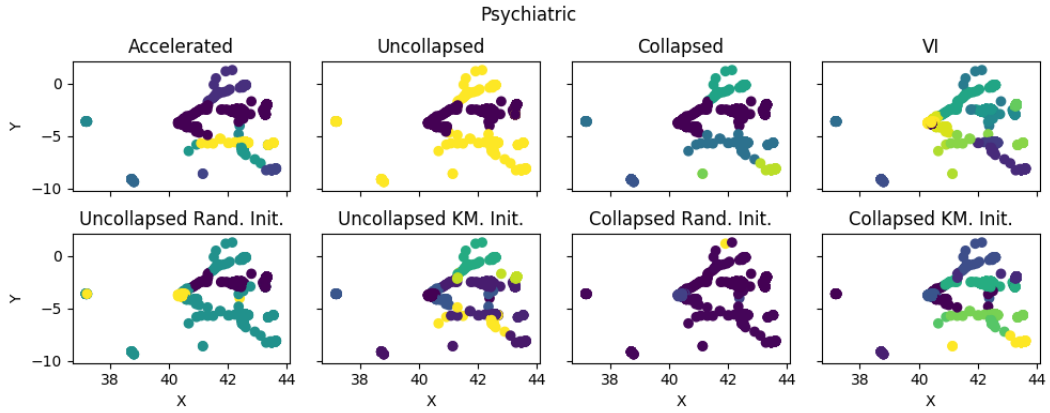

Figure 4.2: Clustering results for each sampling method. Different colors represent different clusters.

Dataset B [58], divided into 1,818 training images and 606 test set images. For both of these datasets, we model the data with a multinomial likelihood and a Dirichlet prior:

$$\begin{aligned} P(X_i|\theta_{z_i}, z_i) &\sim \text{Multinomial}(\theta_{z_i}), \\ H &\sim \text{Dirichlet}(\gamma) \end{aligned} \tag{4.3}$$

In each experiment we initialized all observations to the same cluster and distributed the data randomly across 10 processors. We ran the sampler for 100 iterations with a global synchronization step every 10 iterations and stopped the accelerated sampling after 50 iterations. For the random and K-means initialization tests (labeled "Rand. Init." and "KM Init." on the figures, respectively), we distributed data to 100 initial clusters either by random sampling or K-means allocation.

For each dataset, we can see that both the collapsed and uncollapsed samplers have difficulty proposing good features in high dimensional space.

For the uncollapsed sampler, the results demonstrate that it is difficult for the sampler to propose good locations from the prior in high dimensional space and as a result, very few new features are instantiated relative to the size of the data. While the collapsed sampler can still propose accepted features, Figures 4.4 and 4.6 show that most of the data is assigned to one or two clusters and the rest are generally singleton features. The reason for this is that in datasets with a high signal-to-noise ratio, like the image datasets used in the paper, the base measure is flat over the entire space of features and thus accepting good features tends to rarely happen. The features that are accepted under collapsed sampling tend to either be large clusters that contain most of the data or singletons with tight concentration around its assigned observation. As evidenced by the features in Figure 4.9, the most popular features tend to be very blurry and uninformative.

On the other hand, the features proposed under the accelerated sampler tend to be more popular so that the data are distributed more evenly among the clusters than in the collapsed and uncollapsed samplers (examples of the accelerated features can be seen in Figure 4.7). Additionally, the features accepted under accelerated sampling tend to be more "pure" in the sense that the data assigned to the clusters provide a clear interpretation of the cluster. This is most obviously seen in the MNIST dataset where each learned feature generally represents a digit that fits multiple observations well (as evidenced in Figure 4.4).

Out of the other sampling methods, the variational inference method

works well for discovering good features and obtaining a good predictive likelihood– however, deriving the evidence lower bound to optimize in variational inference is a non-trivial task and often reduces the choice of likelihoods and priors with a lower bound available in closed form. Again, our method could be used for any general class of likelihoods and priors, though we choose to model the data with conjugate priors to the likelihood in order to obtain a closed form expression of the predictive likelihood so that we may compare easily between different sampling methods.

Furthermore, random and K-means initialization for the uncollapsed and collapsed samplers demonstrate that we need a smarter way of learning new features beyond just having good initial states. Although Figures 4.11, 4.12, 4.13, and 4.14 show that, under these initializations, we can learn some features of the data. But we can also see, by the poor quality of many of the other features learned by these methods, that our base measure has difficulty accurately representing out high-dimensional data set. This difficulty is one which we raised in the first point of our introduction. Thus, we need our accelerated method to learn features because our method better represents complicated datasets.

4.4 Discussion

We have introduced a novel method to overcome a problem inherent in Bayesian nonparametric latent variable models of learning unobservable features in a high-dimensional regime while also providing a data-parallel in-

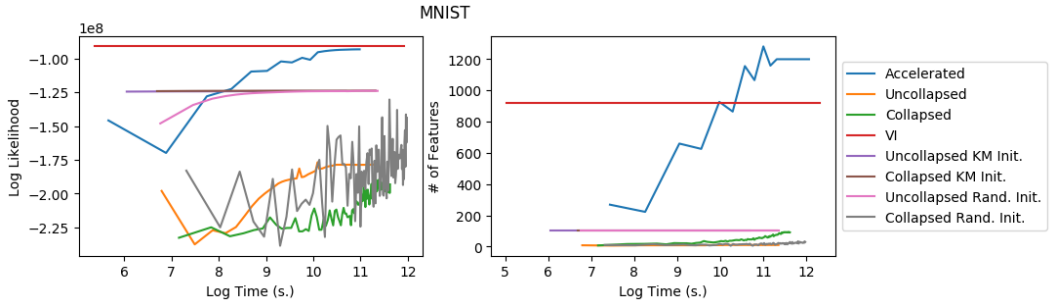

Figure 4.3: Test set predictive log likelihood vs. log time (seconds) and number of features vs. log time.

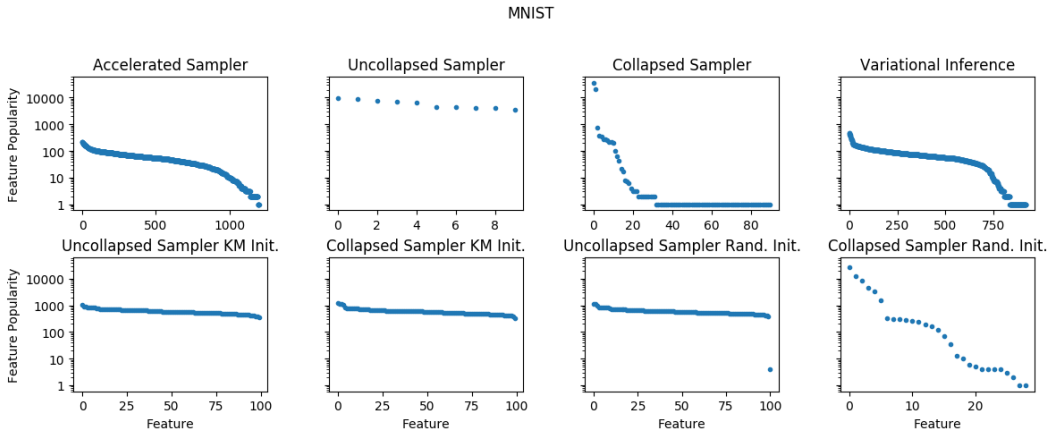

Figure 4.4: Popularity of each instantiated feature.

ference method suitable for "big data" scenarios. In order to accelerate the mixing of the MCMC sampler, we propose feature locations from observations that are poorly fit to its currently allocated feature during the accelerated stage of our sampler in order to find better locations for features.

After running accelerated sampling, our method then reverts to a completely uncollapsed model, which is easily and inherently parallelizable without

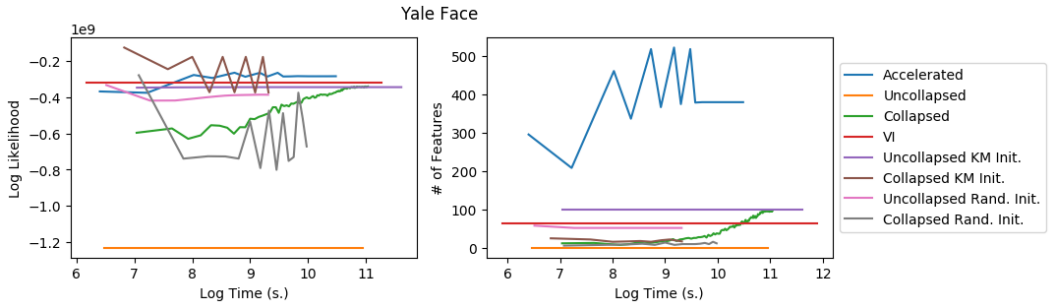

Figure 4.5: Test set predictive log likelihood vs. log time (seconds) and number of features vs. log time.

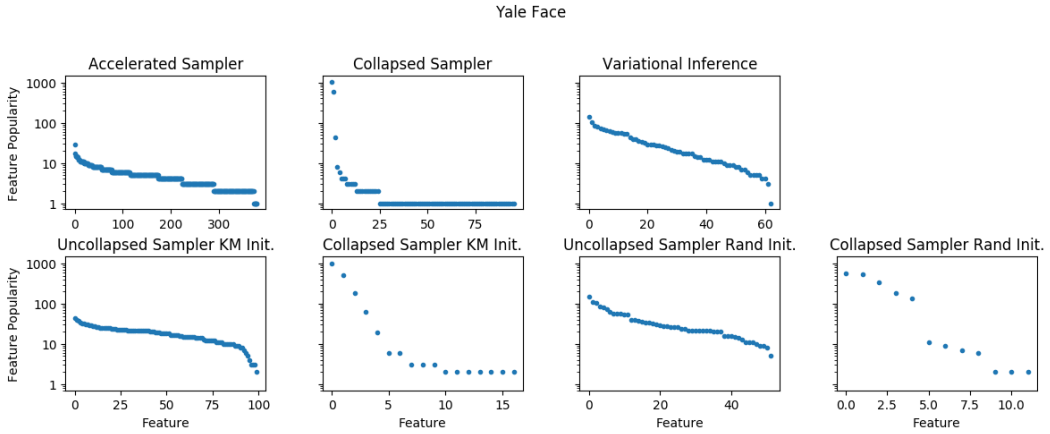

Figure 4.6: Popularity of each instantiated feature. Uncollapsed sampler put all observations into one cluster.

excessive processor communication, in order to maintain the theoretical correctness of our inference algorithm converging to the correct limiting posterior distribution. The additional benefit of our sampler is that our technique works for a general choice of likelihood and prior, whereas using a collapsed sampler limits the model choice to a narrow range of options for data modeling.

At first, it may seem that the number of features discovered for the

Figure 4.7: Yale faces features (left) and MNIST features (right) obtained via accelerated sampling, sorted in descending order of popularity.

Figure 4.8: Yale faces features (left) and MNIST features (right) obtained via uncollapsed sampling, sorted in descending order of popularity.

accelerated method is excessive but we are using a very simple likelihood to model the digits instead of a more complicated model that is invariant to rotations or scalings of the data. This is apparent in the features discovered for accelerated sampling and variational inference, where a large proportion of the popular clusters learned are various forms of "ones". Furthermore, we

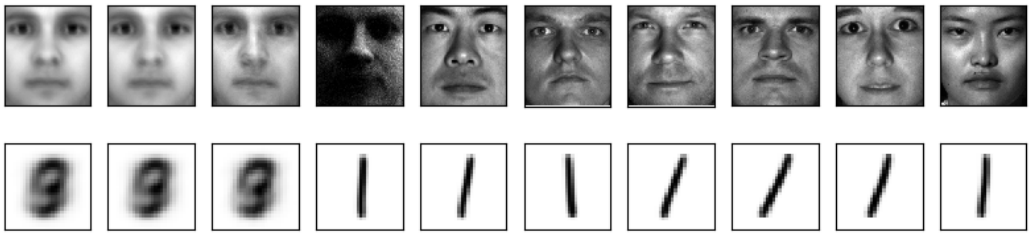

Figure 4.9: First 10 Yale faces features (top) and MNIST features (bottom) obtained via collapsed sampling, sorted in descending order of popularity.

Figure 4.10: Yale faces features (left) and MNIST features (right) obtained via variational inference, sorted in descending order of popularity.

could propose merge steps for features in order to prune the number of features if we are concerned about the number of features learned. Additionally, our method is suitable for other latent feature models as well–for example, sparse

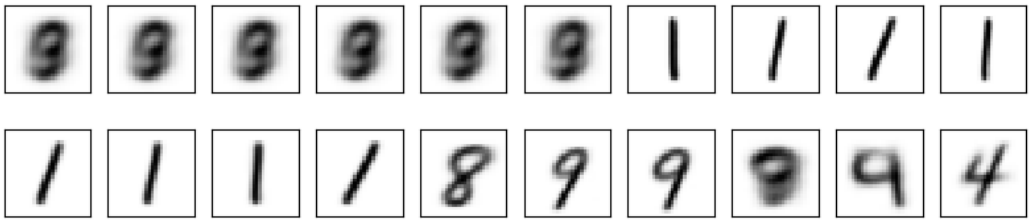

Figure 4.11: First 10 MNIST features obtained via collapsed sampling initialized randomly over 100 partitions (top) or initialized with 100 clusters in K-means (bottom), sorted in descending order of popularity.

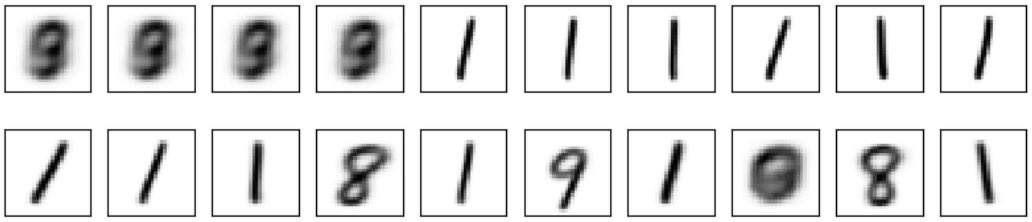

Figure 4.12: First 10 MNIST features obtained via uncollapsed sampling initialized randomly over 100 partitions (top) or initialized with 100 clusters in K-means (bottom), sorted in descending order of popularity.

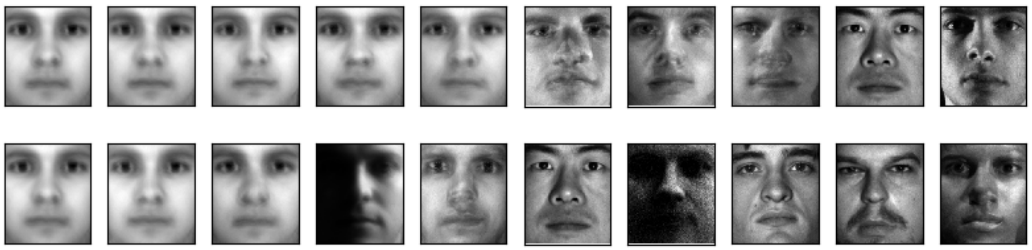

Figure 4.13: First 10 Yale faces features obtained via collapsed sampling initialized randomly over 100 partitions (top) or initialized with 100 clusters in K-means (bottom), sorted in descending order of popularity.

latent factor models like the Indian Buffet Process [36] but for demonstration purposes we only examined our method on the DPMM in this paper.

Figure 4.14: First 10 Yale faces features obtained via uncollapsed sampling initialized randomly over 100 partitions (top) or initialized with 100 clusters in K-means (bottom), sorted in descending order of popularity.

Given our Bayesian formulation of the problem, we now have a natural generative model from which we can simulate GAN-type behavior. In contrast, the DPMM and BNP models in general do not need to train a discriminator to generate data but instead we fit a latent variable model through MCMC inference which then provides a method to generate data from the specified hierarchical model placed on the data. Because our method allows us to learn features in complex, high-dimensional settings where previously it was difficult to do so, we now have an opportunity to realize the promises and theoretical benefits of Bayesian nonparametric modeling in complicated datasets.

In future work, we hope to demonstrate the continued success of Bayesian non-parametrics in modeling complex data while demonstrating the additional benefit that Bayesian and Bayesian nonparametric methods have in providing a natural representation of uncertainty quantification of our results and predictions while having a theoretically motivated methodology of adapting model complexity to the data.

Chapter 5

Robust and Parallel Bayesian Model Selection

In many data modeling scenarios, many plausible models are available to fit to the data, each of which may result in drastically different predictions and conclusions. Being able to select the right model for inference is a crucial task. As our main example, we consider model selection for a normal linear model:

$$Y = X\beta + \epsilon, \;\; \epsilon \sim N(0, \sigma^2 I), \tag{5.1}$$

where Y is an N dimensional response vector, X is an $N \times D$ dimensional design matrix and β is a D dimensional vector of regression parameters. Here the candidate models to be selected could refer to the sets of significant variables. In a Bayesian setting, we have a natural probabilistic evaluation of models through posterior model probabilities. Depending on the objectives of the data analysis, we may be interested in assessing the belief on which is the "best" model or obtaining predictions with minimum error.

Existing procedures to accomplish the aforementioned goals, however, will perform poorly under the presence of outliers and contaminations. In addition, Markov chain Monte Carlo algorithms for these methods do not scale to big data situations. This chapter will investigate a "divide-and-conquer"

method that integrates with existing Bayesian model selection techniques, in a way that is robust to outliers and, moreover, allows us to perform Bayesian model selection in parallel.

Our "divide-and-conquer" strategy is based on the ideas for robust inference using the notion of the geometric median [63], especially the median posterior in the Bayesian context [98, 65]. Previous work in this area has focused on the performance in parametric inference. Our contribution in this paper is to demonstrate the effectiveness of these ideas in selecting the correct class of models on top of the parameters. In particular, we show that the model aggregated across different subsets (the "divide") has improved concentration to the true model class compared to the one using the full data set. This concentration is in terms of the posterior model probabilities to the point mass assigned to the true model. The result also holds jointly with the concentration of the parameter estimates, and under the presence of outliers and hence demonstrates robustness. We carry out extensive numerical studies on simulation data and a real data example to demonstrate the performance of our proposed approach.

5.1 Bayesian Model Selection

In Bayesian model selection, we define the prior model probability $Pr(M_k)$ for each of the model M_k $(k = 1, \ldots, K)$ under consideration. For model M_k, we additionally have parameters (β_k, σ_k^2) with prior $Pr(\beta_k, \sigma_k^2|M_k)$, which leads to a likelihood $Pr(Y|\beta_k, \sigma_k^2, M_k)$. Thus, the posterior model prob-

ability for model M_k, $Pr(M_k|-)$, is proportional to

$$Pr(M_k) \int Pr(Y|\beta_k, \sigma_k^2, M_k) Pr(\beta_k, \sigma_k^2|M_k)\, \mathrm{d}\beta_k \mathrm{d}\sigma_k^2.$$

However, as noted in [7], choosing the model with the highest posterior model probability is not always the best option nor should one neglect the risk of model uncertainty. Instead of resorting to a single model for predicted values $\tilde{Y}$ (or some quantity of interest in general), [42] proposes to average over the model uncertainty with Bayesian model averaging (BMA) to obtain a posterior mean and variance of $\tilde{Y}$ at a covariate level $\tilde{X}$:

$$\begin{aligned} E[\tilde{Y}|\tilde{X}, Y] &= \sum_{k=1}^{K} E[\tilde{Y}|\tilde{X}, Y, M_k] Pr(M_k|\tilde{X}, Y), \\ Var(\tilde{Y}|\tilde{X}, Y) &= \sum_{k=1}^{K} Pr(M_k|X, Y) \Big(Var(\tilde{Y}|\tilde{X}, Y, M_k) + \\ &\qquad E[\tilde{Y}|\tilde{X}, Y, M_k]^2 \Big) - E[\tilde{Y}|X, Y]^2. \end{aligned}$$

We will focus on BMA in our theoretical developments in this paper. Our numerical experiments, however, will show that our divide-and-conquer strategy is also effective in applying on other model selection methods.

The first alternative to BMA is the median probability model, which can be shown to be optimal if we must choose one model for prediction [7]. In this approach, we define the posterior inclusion probability of each predictor x_d $(d = 1, \ldots, D)$ as the sum of posterior model probabilities of the models that include predictor x_d, namely $p_d = \sum_{k: x_d \in M_k} Pr(M_k|X, Y)$. The median probability model is the model that includes the predictors x_d if $p_d \geq 1/2$.

Second, using the maximum value of the likelihood for each model $Pr(Y|\hat{\beta}_k, \hat{\sigma}_k^2, M_k)$, where $(\hat{\beta}_k, \hat{\sigma}_k^2)$ is the maximum likelihood estimate of (β_k, σ_k^2), we can perform penalized model selection through the Akaike information criterion (AIC) [1] or the Bayesian information criterion (BIC) [78] by selecting the model with the lowest information criterion:

$$\text{AIC} = -2 \log Pr(Y|\hat{\beta}_k, \hat{\sigma}_k^2, M_k) + 2(D+1),$$
$$\text{BIC} = -2 \log Pr(Y|\hat{\beta}_k, \hat{\sigma}_k^2, M_k) + (D+1) \log N.$$

The final model selection technique we will consider is stochastic variable selection through the spike and slab model [28], which allows for variable shrinkage under high-dimensional models. For the purposes of this paper, we will use the rescaled spike and slab model [47]. To perform posterior inference in this model, we first define $Y' = \sqrt{\frac{N}{\hat{\sigma}^2}} Y$ where $\hat{\sigma}^2$ is the unbiased estimate of σ^2 under the full model and let $\nu_0 > 0$ be some small number. The model is defined to be the following mixture model:

$$Y' \sim N(X\beta, N\sigma^2 I),\ \beta_{SS_d} \sim N(0, J_d \tau_d^2),$$
$$\sigma_{ss}^{-2} \sim \text{Gamma}(a, b),\ J_d \sim (1-w)\delta_{J_d}(\nu_0) + w\delta_{J_d}(1),$$
$$\tau_d^{-2} \sim \text{Gamma}(a_\tau, b_\tau),\ w \sim \text{Uniform}(0, 1).$$

5.2 Divide-and-Conquer and Robust Bayesian Model Selection

In our robust model selection strategy, we divide N observations into R subsets of roughly equal sample size. Then inference, model selection and

prediction is performed for the linear model $Y_{(j)} = X_{(j)}\beta + \epsilon_{(j)}$ independently across $j = 1, \ldots, R$ subsets using the existing Bayesian model selection procedures, which are then combined to form a final model or a combined prediction value.

Given linear model (5.1), we first define the following priors on a normal likelihood with response variable Y and D-dimensional predictor X. The N observations are divided into R subsets with s observations within each subset. One has,

$$Pr\left(\sigma_{(j)}^{-2}\right) = \text{Gamma}(a, b),$$
$$Pr(\beta_{(j)}|\sigma_{(j)}^2) = N(\beta_0, \sigma_{(j)}^2\Sigma_0).$$

To compensate for the data division, we raise the likelihood of the divided data $Pr(Y_{(j)}|X_{(j)}, \beta, \sigma^2)$ to the R-th power and adjust the normalizing constant accordingly so that the likelihood for Y_j is:

$$\left(\frac{R}{2\pi\sigma_{(j)}^2}\right)^{N/2} \exp\left\{\frac{-R}{2\sigma^2}\left(Y_{(j)} - X_{(j)}\beta_{(j)}\right)^T\left(Y_{(j)} - X_{(j)}\beta_{(j)}\right)\right\}.$$

The intuition and motivation for raising the subset likelihood to R-th power is to adjust the potentially inflated variance of the subset posterior distribution. Exploiting conjugacy, we obtain the full conditionals for data subset

$j = 1, \ldots, R$:

$$
\begin{aligned}
Pr(\beta_{(j)}|-) &= N\left(\mu_\beta, \sigma^2\Sigma_\beta\right), \\
\mu_\beta &= \Sigma_\beta\left(\beta_0\Sigma_0^{-1} + RX_{(j)}^T Y_{(j)}\right), \\
\Sigma_\beta &= \left(\Sigma_0^{-1} + RX_{(j)}^T X_{(j)}\right)^{-1}, \\
Pr\left(\sigma_{(j)}^{-2}|-\right) &= \text{Gamma}\left(a', b'\right), \\
a' &= a + \frac{N+D}{2}, \\
b' &= b + \frac{R}{2}\epsilon^T\epsilon + \frac{1}{2}\left(\beta_{(j)} - \beta_0\right)^T \Sigma_0^{-1}\left(\beta_{(j)} - \beta_0\right), \\
\epsilon &= \left(Y_{(j)} - X_{(j)}\beta_{(j)}\right).
\end{aligned}
$$

Let $\Sigma_X = I + RX_{(j)}\Sigma_0 X_{(j)}^T$, then integrating out the parameters gives us the following marginal distribution $Pr(Y_{(j)}|X_{(j)})$:

$$
\frac{\left(\frac{R}{2\pi}\right)^{\frac{N}{2}} b^a \Gamma(a + \frac{N}{2}) \left|\Sigma_X\right|^{-\frac{1}{2}} / \Gamma(a)}{\left(b + \frac{R}{2}\left(Y_{(j)} - X_{(j)}\beta_0\right)^T \Sigma_X^{-1}\left(Y_{(j)} - X_{(j)}\beta_0\right)\right)^{a+\frac{N}{2}}}.
$$

For distributed AIC and BIC model evaluation, we raise the likelihood term of the AIC and BIC formula to the power of R:

$$
\begin{aligned}
\text{AIC}_R &= -2R\log Pr(Y_{(j)}|\hat{\beta}_k, \hat{\sigma}_k^2, M_k) + 2(D+1), \\
\text{BIC}_R &= -2R\log Pr(Y_{(j)}|\hat{\beta}_k, \hat{\sigma}_k^2, M_k) + (D+1)\log N.
\end{aligned}
$$

In applying our procedure with the spike and slab prior, we derived the full Gibbs sampler for our procedure. For posterior inference in the spike and slab model, let $\Delta = \text{diag}\left\{J_1\tau_1^2, \ldots, J_D\tau_D^2\right\}$, we can perform Gibbs sampling

by drawing from the following posteriors:

$$
\begin{aligned}
Pr(\beta_{SS(j)}|-) &= N(\mu_{\beta_{SS}}, \Sigma_{\beta_{SS}}), \\
\Sigma_{\beta_{SS}} &= \left(\Delta^{-1} + \frac{R}{N\sigma^{-2}_{SS(j)}} X^T_{(j)} X_{(j)}\right)^{-1}, \\
\mu_{\beta_{SS}} &= \Sigma_{\beta_{SS}} \left(\frac{R}{N\sigma^{-2}_{SS(j)}} X^T_{(j)} Y_{(j)}\right), \\
Pr\left(\sigma^{-2}_{SS(j)}|-\right) &= \text{Gamma}\left(a'_{SS}, b'_{SS}\right), \\
a'_{SS} &= a + \frac{N}{2}, \\
b'_{SS} &= b + \frac{R}{2N}\left(Y_{(j)} - X_{(j)}\beta_{SS(j)}\right)^T \left(Y_{(j)} - X_{(j)}\beta_{SS(j)}\right), \\
Pr\left(J_d|-\right) &\propto w_{d1}\delta_{J_d}(\nu_0) + w_{d2}\delta_{J_d}(1), \\
w_{d1} &= (1-w)\nu_0^{-1/2} \exp\left\{-\frac{\beta^2_{SS(j)d}}{2\nu_0\tau_d^2}\right\}, \\
w_{d2} &= w \exp\left\{-\frac{\beta^2_{SS(j)d}}{2\tau_d^2}\right\}, \\
Pr\left(\tau_d^{-2}|-\right) &= \text{Gamma}\left(a_\tau + \frac{1}{2}, b_\tau + \frac{\beta^2_{SS(j)d}}{2J_d}\right), \\
Pr(w|-) &= \text{Beta}\left(1 + |\{d : J_d = 1\}|, 1 + |\{d : J_d = \nu_0\}|\right).
\end{aligned}
$$

Once inference is built on each subset, the key step is to aggregate the subset models (or estimates) together into a final model (or estimate). To aggregate our results, we collect the R number of subset models or estimates and find the geometric median between these R elements. The geometric median for a set of elements $\{x_1, \ldots, x_R\}$ valued on a Hilbert space $\mathbb{H}$, is

defined as

$$x_* = \text{med}_g(x_1, \ldots, x_R) = \text{argmin}_{y \in \mathbb{H}} \sum_{j=1}^{R} \|y - x_j\|, \tag{5.2}$$

where $\|\cdot\|$ is the norm associated with the inner product in $\mathbb{H}$ [65]. The solution can generally be effectively approximated using the Weiszfeld algorithm [99].

For instance, in the case of aggregating the posterior model probabilities across R subsets of data, the geometric median operates on the space of posterior distributions and the geometric median posterior model probability, $Pr_*(M_k|X, Y)$, is defined as:

$$\underset{P \in \Pi_K}{\text{argmin}} \sum_{j=1}^{R} \left|\left|P - Pr(M_k|X_{(j)}, Y_{(j)})\right|\right|, \tag{5.3}$$

where $Pr(M_k|X_{(j)}, Y_{(j)})$ is the posterior model probabilities for subset j, and Π_K denotes the space of distributions on K support points. The metric $\|\cdot\|$ here can be taken as the Euclidean metric, or an integral probability metric (IPM) defined as $||P - Q|| = \sup_{f \in \mathcal{F}} \left|\int f(x)\, \mathrm{d}(P - Q)(x)\right|$ for some class of functions $\mathcal{F}$ [87, 86].

For the model selection techniques discussed earlier (AIC, BIC, and the median model selection), we can choose a final model in two ways: One, we can select the best model locally on each subset, use it for prediction, and then aggregate the results (estimate combination). Or two, we can take the median of the model selection criteria and choose that particular model on each subset and then aggregate the results to get a final model (model combination).

However, in Bayesian model averaging and spike and slab modeling we do not choose a final model. We can still perform model or estimate combination by aggregating the posterior model probabilities. We consider both model and estimate combinations in our experiments and show that they yield similar results in our experimental settings.

Algorithm 6: Algorithm for robust model selection in the case of BMA.

for $j \in \{1, \ldots, R\}$ **do**
- Raise likelihood to R-th power
- Compute inference for $P(\theta|M_k, X_{(j)}, Y_{(j)})$ for $k = 1, \ldots, K$
- Draw predictive values from predictive posterior $P(\tilde{Y}|M_k, X_{(j)}, Y_{(j)})$ for $k = 1, \ldots, K$
- Calculate posterior model probabilities $\{P(M_k|X_{(j)}, Y_{(j)})\}_{k=1,\ldots,K}$

Calculate geometric median of posterior model probabilities over the subsets using (5.3).

Approximate geometric medians of posterior parameter probabilities or predictive values given individual models over the subsets using (5.2).

Obtain BMA estimate: $E[\tilde{Y}|Y, X] = \sum_{k=1}^{K} E_*[\tilde{Y}|X, Y, M_k]Pr_*(M_k|X, Y)$

5.3 Improved Concentration and Robustness

In this section we provide theoretical justification on the robustness in the divide-and-conquer strategy. In particular, we focus on BMA. Additionally, we show that the aggregated model class from our strategy concentrates faster, in terms of posterior model probabilities, to the correct class compared to using the whole data set at once. This concentration result can be joint

with parameter estimation, and also applies in a way that exhibits robustness against outliers. Note that we do not raise the subset likelihood to R-th power in our current theoretical analysis, but the results can be generalized by imposing slightly stronger entropy conditions on the model.

Let $\mathcal{S}$ be the domain of $\theta = (M_k, \beta, \sigma^2)$, our set of model indices and parameters. Let θ_0 be the true data generating parameter, and let (X_1, Y_1) be a generic data point. Let $p_0(y|x) := p(y|x, \theta_0)$ be the true conditional density of Y_1 given X_1, and $p_0(x)$ be the true density of the covariates X_1. We denote $p_\theta(y|x) := p(y|x, \theta)$. Let P_θ be the distribution defined by $p_0(x) \times p_\theta(y|x)$ and P_0 is the true distribution $p_0(x) \times p_0(y|x)$. For convenience, we denote $P_0 f = P_0 f(X_1, Y_1) = E_{p_0}[f(X_1, Y_1)]$ where $E_{p_0}[\cdot]$ is the expectation under $p_0(y|x) \times p_0(x)$. We denote P_0^N as the true probability measure taken on the data (X, Y) of size N and $P_0^N f = E_{P_0^N}[f(X, Y)]$. Lastly, we denote $\mathcal{D}(\epsilon, \mathcal{P}, d)$ as the ϵ-packing number of a set of probability measures $\mathcal{P}$ under the metric d, which is the maximal number of points in $\mathcal{P}$ such that the distance between any pair is at least ϵ. We implicitly assume here that $\mathcal{P}$ is separable. The following Theorem 1 follows from a modification of Theorem 2.1 in [30]:

Theorem 1. *Assume that there is a sequence ε_N such that $\varepsilon_N \to 0$ and $N\varepsilon_N^2 \to \infty$ as $N \to \infty$, a constant C, and a set $\mathcal{S}_N \in \mathcal{S}$ so that*

1. $\log \mathcal{D}(\varepsilon_N/2, \mathcal{P}_{\mathcal{S}_N}, d_H) \leq N\varepsilon_N^2$.

2. $Pr(\mathcal{S} \setminus \mathcal{S}_N) \leq e^{-N\varepsilon_N^2(C+4)}$.

3. $Pr\left(\theta : -P_0 \log \frac{p_\theta(Y_1|X_1)}{p_0(Y_1|X_1)} \leq \varepsilon_N^2, \right.$
$\left. P_0\left(\frac{p_\theta(Y_1|X_1)}{p_0(Y_1|X_1)}\right)^2 \leq \varepsilon_N^2\right) \geq e^{-N\varepsilon_N^2 C}.$

where $\mathcal{P}_{\mathcal{S}_N} = \{p_0(x) \times p_\theta(y|x) : \theta \in \mathcal{S}_N\}$ *and* d_H *is the Hellinger distance. Then we have*

$$P_0^N\left(Pr(\theta : d_H(P_\theta, P_0) > T\varepsilon_N^2|X, Y) > \delta\right) \leq \frac{1}{C^2 N\varepsilon_N^2 \delta} + \frac{2e^{-LN\varepsilon_N^2}}{\delta} + \frac{2e^{-2N\varepsilon_N^2}}{\delta}, \tag{5.4}$$

for any $0 < \delta < 1$ *and sufficiently large* $T > 0$ *such that* $LT^2 \geq C + 4$ *and* $LT^2 - 1 > L$*, where* L *is a universal constant.*

Proof of Theorem 1.

The proof is a modification of that for Theorem 2.1 in [30]. Take any $\epsilon > 2\varepsilon_N$, we have, by Assumption 1,

$$\log \mathcal{D}\left(\frac{\epsilon}{2}, \mathcal{P}_{\mathcal{S}_N}, d\right) \leq \log \mathcal{D}\left(\varepsilon_N, \mathcal{P}_{\mathcal{S}_N}, d\right) \leq N\varepsilon_N^2.$$

Then, by Theorem 7.1 in [30], there exists tests ϕ_N and a large enough constant T (chosen later) such that

$$P_0^N \phi_N \leq e^{N\varepsilon_N^2} e^{-LNT^2\varepsilon_N^2} \frac{1}{1 - e^{-LNT^2\varepsilon_N^2}}, \tag{5.5}$$

and

$$\sup_{\theta \in \mathcal{S}_N : d(P_\theta, P_0) > T\varepsilon_N} P_\theta^N(1 - \phi_N) \leq e^{-LNT^2\varepsilon_N^2}, \tag{5.6}$$

for a universal constant $L > 0$, any $N > 0$, and P_θ^N denotes the probability measure on (X, Y) under $(X_1, Y_1) \sim p_0(x) \times p_\theta(y|x)$.

By (5.5), we have

$$P_0^N Pr(\theta : \theta : d(P_\theta, P_0) > L\varepsilon_N^2 | X, Y)\phi_N \leq P_0^N \phi_N \leq 2e^{-LN\varepsilon_N^2}, \tag{5.7}$$

as $N \to \infty$, if we choose $LT^2 - 1 > L$. Now, since

$$P_0 \frac{p_\theta(Y_1|X_1)}{p_0(Y|X)} = \int \frac{p_\theta(y|x)}{p_0(y|x)} p_0(dy|x)p_0(dx) = \int p_\theta(dy|x)p_0(dx) = 1,$$

by Fubini's theorem, we have

$$P_0^N \int_{\mathcal{S}\setminus\mathcal{S}_N} \prod_{i=1}^N \frac{p_\theta(Y_i|X_i)}{p_0(Y_i|X_i)} Pr(\mathrm{d}\theta) \leq Pr(\mathcal{S} \setminus \mathcal{S}_N).$$

Hence, by Fubini's theorem again,

$$\begin{aligned}
& P_0^N \int_{\theta\in\mathcal{S}: d(P_\theta,P_0)>T\varepsilon_N} \prod_{i=1}^N \frac{p_\theta(Y_i|X_i)}{p_0(Y_i|X_i)} Pr(\mathrm{d}\theta)(1-\phi_N) \\
\leq\ & \Pi(\mathcal{S} \setminus \mathcal{S}_N) + \int_{\theta\in\mathcal{S}_N : d(P_\theta,P_0)>T\varepsilon_N} P_\theta^N (1-\phi_N) Pr(\mathrm{d}\theta) \\
\leq\ & \Pi(\mathcal{S} \setminus \mathcal{S}_N) + e^{-LNT^2\varepsilon_N^2} \quad \text{by (5.6)} \\
\leq\ & 2e^{-N\varepsilon_N^2(C+4)},
\end{aligned} \tag{5.8}$$

if $KM^2 \geq C + 4$, by Assumption 2.

By Lemma 1 (stated below) and Assumption 3, with probability at least $1 - 1/(C^2 N\varepsilon_N^2)$, we have

$$\int \prod_{i=1}^N \frac{p_\theta(Y_i|X_i)}{p_0(Y_i|X_i)} Pr(\mathrm{d}\theta) \geq e^{-2N\varepsilon_N^2} Pr(B_n) \geq e^{-N\varepsilon_N^2(2+C)}, \tag{5.9}$$

where

$$B_n = \left\{ \theta : -P_0 \log \frac{p_\theta(Y_1|X_1)}{p_0(Y_1|X_1)} \leq \varepsilon_N^2,\ P_0 \left(\frac{p_\theta(Y_1|X_1)}{p_0(Y_1|X_1)} \right)^2 \leq \varepsilon_N^2 \right\}.$$

Let A_N be the event that (5.9) holds. We have

$$\begin{aligned}
& P_0^N Pr(\theta : d(P_\theta, P_0) > T\varepsilon_N | X, Y)(1-\phi_N)\mathbf{1}_{A_N} \\
= \; & P_0^N \frac{\int_{\theta: d(P_\theta,P_0)>T\varepsilon_N} \prod_{i=1}^N \frac{p_\theta(Y_i|X_i)}{p_0(Y_i|X_i)} Pr(\mathrm{d}\theta)}{\int \prod_{i=1}^N \frac{p_\theta(Y_i|X_i)}{p_0(Y_i|X_i)} Pr(\mathrm{d}\theta)} (1-\phi_N)\mathbf{1}_{A_N} \\
\leq \; & e^{N\varepsilon_N^2(2+C)} 2e^{-N\varepsilon_N^2(C+4)} \text{ by (5.8) and (5.9)} \\
= \; & 2e^{-2N\varepsilon_N^2}.
\end{aligned}$$

Therefore,

$$\begin{aligned}
& P_0^N Pr(\theta : d(P_\theta, P_0) > T\varepsilon_N | X, Y) \\
= \; & P_0^N Pr(\theta : d(P_\theta, P_0) > T\varepsilon_N | X, Y)\phi_N + P_0^N Pr(\theta : d(P_\theta, P_0) \\
& > T\varepsilon_N | X, Y)(1-\phi_N)\mathbf{1}_{A_N} + P_0^N Pr(\theta : d(P_\theta, P_0) \\
& > T\varepsilon_N | X, Y)(1-\phi_N)(1-\mathbf{1}_{A_N}) \\
\leq \; & P_0^N Pr(\theta : d(P_\theta, P_0) > T\varepsilon_N | X, Y)\phi_N + P_0^N Pr(\theta : d(P_\theta, P_0) \\
& > T\varepsilon_N | X, Y)(1-\phi_N)\mathbf{1}_{A_N} + P_0^N(A_N^c) \text{ for sufficiently large } T \\
\leq \; & 2e^{-LN\varepsilon_N^2} + 2e^{-2N\varepsilon_N^2} + \frac{1}{C^2 N\varepsilon_N^2},
\end{aligned}$$

by (5.7), (5.10) and the property of A_N. By Chebyshev's inequality, we have

$$P_0^N \left(Pr(\theta : d(P_\theta, P_0) > T\varepsilon_N^2 | X, Y) > \delta \right) \leq \frac{1}{C^2 N\varepsilon_N^2 \delta} + \frac{2e^{-LN\varepsilon_N^2}}{\delta} + \frac{2e^{-2N\varepsilon_N^2}}{\delta},$$

which concludes the theorem. □

Lemma 1. *For any $\epsilon > 0$ and probability distribution Π defined on the set*

$$\left\{ \theta : -P_0 \log \frac{p_\theta(Y|X)}{p_0(Y_1|X_1)} \leq \epsilon^2, \; P_0 \left(\frac{p_\theta(Y|X)}{p_0(Y_1|X_1)} \right)^2 \leq \epsilon^2 \right\}, \tag{5.10}$$

we have, for every $C > 0$,

$$P_0^N \left(\int \prod_{i=1}^{N} \frac{p_\theta(Y_i|X_i)}{p_0(Y_i|X_i)} \Pi(\mathrm{d}\theta) \leq e^{-(1+C)N\epsilon^2} \right) \leq \frac{1}{C^2 N \epsilon^2}. \tag{5.11}$$

Theorem 2 (Adopted from [63])**.** *Consider a Hilbert space* $(\mathbb{H}, \langle \cdot, \cdot \rangle)$ *and* $\xi_0 \in \mathbb{H}$*. Let* $\hat{\xi}_1, \ldots, \hat{\xi}_R \in \mathbb{H}$ *be a collection of independent random* $\mathbb{H}$*-valued elements. Let* α, q, ν *be constants such that* $0 < q < \alpha < 1/2$ *and* $0 \leq \nu < (\alpha - q)/(1 - q)$*. Suppose that there exists* $\epsilon > 0$ *such that for all* j*, where* $1 \leq j \leq \lfloor (1 - \nu)R \rfloor + 1$*,*

$$P(\|\hat{\xi}_j - \xi_0\| > \epsilon) \leq q.$$

Let $\hat{\xi}_* = med_g(\hat{\xi}_1, \ldots, \hat{\xi}_R)$ *be the geometric median of* $\{\hat{\xi}_1, \ldots, \hat{\xi}_R\}$*. Then*

$$P(\|\hat{\xi}_* - \xi_0\| > C_\alpha \epsilon) \leq \left(e^{(1-\nu)\psi(\frac{\alpha-\nu}{1-\nu}, q)} \right)^{-R},$$

where $C_\alpha = (1 - \alpha)\sqrt{1/(1 - 2\alpha)}$*, and*

$$\psi(\alpha, q) = (1 - \alpha) \log \frac{1 - \alpha}{1 - q} + \alpha \log \frac{\alpha}{q}.$$

As noted by [30], the important assumptions are Assumptions 1 and 3. Essentially, Assumption 1 constrains the size of the parameter domain $\mathcal{S}$ to be not too big, whereas Assumption 3 ensures sufficient mass of the prior on a neighborhood of the true parameter. The concentration result (5.4) states that the posterior distribution of θ is close to the true θ_0 with high probability,

where the closeness is measured in terms of the Hellinger distance between the likelihoods. Note that the RHS of (5.4) consists of three terms. The dominant term is the power-law decay in $N\varepsilon_N^2$. The other two exponential decay terms result from technical arguments in the existence of tests that sufficiently distinguish between distributions [8, 56].

Next we describe the concentration behavior of BMA. We focus on the situations where all the candidate models are non-nested, i.e. only one model contains distributions that are arbitrarily close to the truth. Without loss of generality, we let M_1 be the true model.

Theorem 3 (BMA of Non-Nested Models)**.** *Suppose the assumptions in Theorem 1 hold. Also assume that, for sufficiently small $\epsilon > 0$, $d(P_\theta, P_0) > \epsilon$ for any $\theta \in \mathcal{S}_{-1} := \{(M_k, \beta, \sigma^2) : k \neq 1\}$. Let L be the same universal constant arising in Theorem 1. We have*

1. *For any given $0 < \delta < 1$,*

$$P_0^N \left(Pr(M_1|X,Y) < 1 - \delta \right) \leq \frac{1}{C^2 N \varepsilon_N^2 \delta} + \frac{2e^{-LN\varepsilon_N^2}}{\delta} + \frac{2e^{-2N\varepsilon_N^2}}{\delta}, \tag{5.12}$$

   for sufficiently large N.

2. *For any given $0 < \delta < 1$,*

$$P_0^N \left(d_E(Pr(M_k|X,Y), \mathbf{e}_1) > \delta \right) \leq \frac{\sqrt{2}}{C^2 N \varepsilon_N^2 \delta} + \frac{2\sqrt{2}e^{-LN\varepsilon_N^2}}{\delta} + \frac{2\sqrt{2}e^{-2N\varepsilon_N^2}}{\delta}, \tag{5.13}$$

   for sufficiently large N, where d_E is the Euclidean distance, and $\mathbf{e}_1$ is the point mass on M_1.

3. For any $0 < \delta < \sqrt{(\sqrt{2}-1)^2+1}/2$,

$$P_0^N\left(d_H(Pr(M_k|X,Y),\mathbf{e}_1) > \delta\right) \leq \frac{(\sqrt{2}-1)^2+1}{\sqrt{2}C^2N\varepsilon_N^2\delta^2} + \frac{((\sqrt{2}-1)^2+1)e^{-LN\varepsilon_N^2}}{\delta^2} + \frac{((\sqrt{2}-1)^2+1)e^{-2N\varepsilon_N^2}}{\delta^2}, \tag{5.14}$$

for sufficiently large N.

Proof of Theorem 3.

Proof of 1. Consider large enough N and fix a sufficiently large $T > 0$. We have

$$\begin{aligned} & Pr(\theta : d(P_\theta, P_0) \leq T\varepsilon_N^2|X,Y) \\ = \; & E_{Pr}\left[Pr(\theta : d(P_\theta, P_0) \leq T\varepsilon_N^2|M_k,X,Y)|X,Y\right], \qquad (5.15) \end{aligned}$$

where $E_{Pr}[\cdot|X,Y]$ denotes the posterior expectation

and $Pr(\cdot|M_k,X,Y)$ denotes the posterior distribution given model M_k

$$= \; Pr(M_1|X,Y)Pr(\theta : d(P_\theta, P_0) \leq T\varepsilon_N^2|M_1,X,Y), \qquad (5.16)$$

by the condition that $d(P_\theta, P_0) > T\varepsilon_N^2$ for any $\theta \in \mathcal{S}_{-1}$ and any $T > 0$ eventually. Hence

$$Pr(\theta : d(P_\theta, P_0) \leq T\varepsilon_N^2|X,Y) \geq 1-\delta, \tag{5.17}$$

implies

$$Pr(M_1|X,Y) \geq 1-\delta. \tag{5.18}$$

The result then follows from Theorem 1, which implies that (5.17) occurs with probability at least

$$1 - \left(\frac{1}{C^2 N \varepsilon_N^2 \delta} + \frac{2e^{-LN\varepsilon_N^2}}{\delta} + \frac{2e^{-2N\varepsilon_N^2}}{\delta} \right),$$

Proof of 2. Note that (5.18) implies

$$d_E(Pr(M_k|X,Y), \mathbf{e}_1) = \sqrt{(1 - Pr(M_1|X,Y))^2 + \sum_{k \neq 1} Pr(M_k|X,Y)^2} \leq \sqrt{2}\delta, \tag{5.19}$$

since $(1 - Pr(M_1|X,Y))^2 \leq \delta^2$ and $(\delta, 0, \ldots, 0)$ is an optimizer of the optimization

$$\max \sum_{i=2}^{K} x_i^2 \text{ subject to } \sum_{i=2}^{K} x_i \leq \delta.$$

Hence (5.12) and (5.19) together imply

$$P_0^N \left(d_E(Pr(M_k|X,Y), \mathbf{e}_1) \geq \sqrt{2}\delta \right) \leq \frac{1}{C^2 N \varepsilon_N^2 \delta} + \frac{2e^{-LN\varepsilon_N^2}}{\delta} + \frac{2e^{-2N\varepsilon_N^2}}{\delta}.$$

By redefining $\tilde{\delta} = \sqrt{2}\delta$, we get (5.13).

Proof of 3. Note that (5.18) implies

$$\begin{aligned} d_H(Pr(M_k|X,Y), \mathbf{e}_1) &= \sqrt{\frac{1}{2}\left((\sqrt{1 - Pr(M_1|X,Y)})^2 + \sum_{k \neq 1} Pr(M_k|X,Y) \right)} \\ &\leq \sqrt{\frac{1}{2}\left((1 - \sqrt{1-\delta})^2 + \delta \right)}, \end{aligned} \tag{5.20}$$

since $x_i = \delta/(k-1)$ for all $i \neq 0$ gives the optimizer of the optimization

$$\max \sum_{i \neq 0} \sqrt{x_i} \text{ subject to } \sum_{i \neq 0} x_i \leq \delta.$$

Hence (5.12) and (5.20) together imply

$$P_0^N\left(d_H(Pr(M_k|X,Y),\mathbf{e}_1) > \sqrt{\frac{1}{2}\left((1-\sqrt{1-\delta})^2+\delta\right)}\right) \leq \frac{1}{C^2N\varepsilon_N^2\delta} + \frac{2e^{-LN\varepsilon_N^2}}{\delta} + \frac{2e^{-2N\varepsilon_N^2}}{\delta}. \tag{5.21}$$

Note that $(1-\sqrt{1-\delta})^2$ is a convex function in δ for $0<\delta<1$ and is equal to 0 at $\delta=0$. Thus $(1-\sqrt{1-\delta})^2 \leq (\sqrt{2}-1)^2\delta$ for $0<\delta<1/2$, where $(\sqrt{2}-1)^2$ is the slope of the line between $(0,0)$ and $(1/2,(1-\sqrt{1-1/2})^2$. Hence, for $0<\delta<1/2$, we have

$$\sqrt{\frac{1}{2}\left((1-\sqrt{1-\delta})^2+\delta\right)} \leq \sqrt{((\sqrt{2}-1)^2+1)\frac{\delta}{2}}.$$

Combining with (5.21), we have

$$P_0^N\left(d_H(Pr(M_k|X,Y),\mathbf{e}_1) > \sqrt{((\sqrt{2}-1)^2+1)\frac{\delta}{2}}\right) \leq \frac{1}{C^2N\varepsilon_N^2\delta} + \frac{2e^{-LN\varepsilon_N^2}}{\delta} + \frac{2e^{-2N\varepsilon_N^2}}{\delta}. \tag{5.22}$$

By redefining $\tilde{\delta} = \sqrt{((\sqrt{2}-1)^2+1)\delta/2}$, we get (5.14). □

Note that the assumption $d(P_\theta, P_0) > \epsilon$ for any $\theta \in \mathcal{S}_{-1}$ and sufficiently small ϵ is a manifestation of the non-nested model situation, asserting that only one model is "correct". Result 1 is a concentration on the posterior probability of picking the correct model to be close to 1.

Result 2 translates this in terms of the Euclidean distance between the model posterior probability and the point mass on the correct model. Result 3 is an alternative using the Hellinger distance. Note that the concentration

bound for Hellinger distance (5.14) is inferior to that for Euclidean distance (5.13) for small δ since δ^2 instead of δ shows up in the RHS of (5.14). This is because in our proof, the function $\sqrt{(1-\sqrt{1-\delta})^2+\delta}$ that appears in (5.21) has derivative $1/(2\sqrt{(1-\delta)((1-\sqrt{1-\delta})^2+\delta)})$ which is ∞ at $\delta = 0$, and thus no linearization is available when δ is close to 0.

Theorem 3 can be modified to handle the case where multiple models contain the truth. In particular, the expression inside the probability in (5.12) becomes

$$\sum_{r\in\mathcal{M}} Pr(M_r|X,Y) < 1-\delta,$$

where $\mathcal{M}$ is the collection of all r such that M_r contains the true model. In (5.13) and (5.14), the use of $\mathbf{e}_1$ is replaced by an existence of some probability vector (dependent on N) supported on the indices in $\mathcal{M}_r$. In other words, one now allows comparing with an arbitrary allocation of probability masses to all true models in the concentration bound. These modifications can be seen by following the arguments in the proof of Theorem 3. Specifically, (5.16) would be modified as

$$\sum_{r\in\mathcal{M}} Pr(M_r|X,Y)Pr(\theta : d(P_\theta,P_0) \le T\varepsilon_N^2|M_r,X,Y).$$

Then (5.17) would imply a modified version of (5.18), namely

$$\sum_{r\in\mathcal{M}} Pr(M_r|X,Y) \ge 1-\delta,$$

giving the claimed modification for (5.12). Then, following (5.19), we could find a probability vector to make all $(1-Pr(M_r|X,Y))^2$ terms vanish except

one, which is in turn bounded by δ^2. This gives the claimed modifications for (5.13) and (5.14).

The following result states how a divide-and-conquer strategy can improve the concentration rate of the posterior model probabilities towards the correct model:

Theorem 4 (Concentration Improvement)**.** *Suppose the assumptions in Theorem 3 hold. Let* $s = N/R$*, and* $q = \frac{\sqrt{2}}{C^2 s\varepsilon_s^2 \delta} + \frac{2\sqrt{2}e^{-Ls\varepsilon_s^2}}{\delta} + \frac{2\sqrt{2}e^{-2s\varepsilon_s^2}}{\delta}$*. For sufficiently large* s*, letting* α, ν *be constants such that* $0 < q < \alpha < 1/2$ *and* $0 \leq \nu < (\alpha - q)/(1 - q)$*, we have:*

1. $Pr_*(M_k|X, Y)$*, the geometric median under* d_E *of* $\{Pr(M_k|(X_{(j)}, Y_{(j)}))\}_{j=1,\ldots,R}$*, satisfies*

$$P_0^N \left(d_E(Pr_*(M_k|X, Y), \mathbf{e}_1) > C_\alpha \delta\right) \leq \left(e^{(1-\nu)\psi(\frac{\alpha-\nu}{1-\nu}, q)}\right)^{-R}, \tag{5.23}$$

   where $C_\alpha = (1 - \alpha)\sqrt{1/(1 - 2\alpha)}$*, and* $\psi(\alpha, q) = (1 - \alpha) \log \frac{1-\alpha}{1-q} + \alpha \log \frac{\alpha}{q}$*.*

2. *Let* K *be the number of model classes, then:*

$$P_0^N \left(Pr_*(M_1|X, Y) < 1 - C_\alpha \delta \sqrt{\frac{K - 1}{K}}\right) \leq \left(e^{(1-\nu)\psi(\frac{\alpha-\nu}{1-\nu}, q)}\right)^{-R}. \tag{5.24}$$

3. *Suppose in addition that, for any* $P_{\theta^1}, P_{\theta^2}$ *such that* $\theta^i = (M_1, \beta^i, (\sigma^2)^i)$ *for* $i = 1, 2$*, we have*

$$d_H(P_{\theta^1}, P_{\theta^2}) \geq \tilde{C} \rho_k(\theta^1, \theta^2)^\gamma, \tag{5.25}$$

where $\rho_k(\theta^1, \theta^2) = \|k(\cdot, \theta^1) - k(\cdot, \theta^2)\|_{\mathbb{H}}$*, with* k *being a characteristic kernel defined on the space* $\{\theta = (M_1, \cdot, \cdot)\}$ *and* $\mathbb{H}$ *is the corresponding reproducing kernel Hilbert space (RKHS), and* $\tilde{C} > 0$ *and* $\gamma > 0$ *are constants. Moreover, assume that there is a universal constant* $\tilde{K}$ *such that* $e^{-\tilde{K}s\varepsilon_s^2/2} \leq \varepsilon_s$ *for all* s*, and we choose* ε_s *such that* $\tilde{q} = \frac{1}{Cs\varepsilon_s^2} + 4e^{-\tilde{K}s\varepsilon_s^2/2} < 1/2$*. Then*

$$P_0^N \Bigg(Pr_*(M_1|X,Y) > 1 - C_\alpha \delta \sqrt{\frac{K-1}{K}}, \\ \|Pr_*(\theta|M_1,X,Y) - \delta_0\|_{\mathcal{F}_k} \leq C_\alpha \tilde{T} \epsilon_s^{1/\gamma} \Bigg) \\ \geq 1 - \left(e^{(1-\nu)\psi\left(\frac{\alpha-\nu}{1-\nu}, q\right)}\right)^{-R} - \left(e^{\psi(\alpha,q)}\right)^{-R},$$

where $\|\cdot\|_{\mathcal{F}_k}$ *is defined as* $\|P - Q\|_{\mathcal{F}_k} = \|\int k(x,\cdot) d(P-Q)(x)\|_{\mathbb{H}}$*,* $\tilde{T} > 0$ *is a sufficiently large constant,* $Pr_*(\theta|M_1, X, Y)$ *is the geometric median of* $\{Pr(\theta|M_1, (X_{(j)}, Y_{(j)}))\}_{j=1,\ldots,R}$ *under the* $\|\cdot\|_{\mathcal{F}_k}$*-norm, and* δ_0 *is the delta measure at the true parameter.*

The significance of Theorem 4 is the improvement of the concentration from power-law decay in Theorem 3 to exponential decay, as the number of subsets grows. Such type of results is known in the case of parameter estimation (e.g., [98, 65]). Theorem 4 generalizes to the case of model selection. Results 1 and 2 describe the exponential concentration for the model posteriors to the correct model, while Result 3 states the joint concentration in both the model posterior and the parameter posterior given the correct model, when one adopts a second layer of divide-and-conquer on the parameter posterior conditional on each individual candidate model. Result 3 in particular combines with the parameter concentration result in [65].

Note that we have taken a hybrid viewpoint here that we assume a "correct" model and parameters in a frequentist sense. Under this view, a posterior probability more concentrated towards the truth is more desirable. This constitutes our main claim that the divide-and-conquer strategy is attractive. This view has been used in existing work like [98, 65].

Finally, the following theorem highlights that the concentration improvement still holds even if the data are contaminated to a certain extent:

Theorem 5 (Robustness to Outliers)**.** *Using the notation in Theorem 4, but assume instead that, for j where $1 \leq j \leq \lfloor (1-\nu)R \rfloor + 1$,*

$$P_0^s \left(d_E(Pr(M_k|X_{(j)}, Y_{(j)}), \mathbf{e}_1) > \delta \right) \leq \frac{\sqrt{2}}{C^2 s\varepsilon_s^2 \delta} + \frac{2\sqrt{2}e^{-Ls\varepsilon_s^2}}{\delta} + \frac{2\sqrt{2}e^{-2s\varepsilon_s^2}}{\delta},$$

the conclusion of Theorem 4 still holds.

Theorem 5 stipulates that when a small number of subsets are contaminated by arbitrary nature, the geometric median approach still retains the same exponential concentration.

Proofs of Theorems 4 and 5.

The proofs of both theorems rely on a key theorem on geometric median in [63], restated in the Appendix. We focus on Theorem 4, as the proof for Theorem 5 is a straightforward modification in light of Theorem 2.

Proof of 1. Immediate by noting that

$$P_0^s \left(d_E(Pr(M_k|X_{(j)}, Y_{(j)}), \mathbf{e}_1) > \delta \right) \leq q,$$

for all $j = 1, \ldots, R$, and applying Theorem 2.

Proof of 2. Note that

$$d_E(Pr_*(M_k|X,Y), \mathbf{e}_1) \geq (1 - Pr_*(M_1|X,Y))\sqrt{\frac{K}{K-1}}. \tag{5.26}$$

To see this, let $a = Pr_*(M_1|X,Y)$. We have

$$d_E(Pr_*(M_k|X,Y), \mathbf{e}_1) = \sqrt{(1-a)^2 + \sum_{i=2}^{K} x_i^2},$$

where x_i's satisfy $\sum_{i=2}^{K} x_i = 1 - a$. Since $(1-a)/(K-1)$ is the optimizer of the optimization

$$\min \sum_{i=2}^{K} x_i^2 \;\text{ subject to }\; \sum_{i=2}^{K} x_i = 1 - a,$$

we get $\sqrt{(1-a)^2 + \sum_{i=2}^{K} x_i^2} \geq (1-a)\sqrt{K/(K-1)}$.

Hence (5.23) and (5.26) together give

$$P_0^N \left(Pr_*(M_1|X,Y) < 1 - C_\alpha \delta \sqrt{\frac{K-1}{K}} \right) \leq \left(e^{(1-\nu)\psi(\frac{\alpha-\nu}{1-\nu}, q)} \right)^{-R}.$$

Proof of 3. Under the additional assumptions, we can invoke Corollary 3.5 in [65] to obtain that

$$P_0^N \left(\|Pr_*(\theta|M_1, X, Y) - \delta_0\|_{\mathcal{F}_k} > C_\alpha \tilde{T} \epsilon_s^{1/\gamma} \right) \leq \left(e^{\psi(\alpha,q)} \right)^{-R}.$$

The result follows from applying a union bound and together with (5.24). □

5.4 Simulations and Data Analysis

For the BMA, AIC, BIC and median probability model tests, we generate data from a model $Y = X\beta + \epsilon$, where X is a 5000×10 matrix and β is a 10 dimensional vector with 3 true predictors. We assess the aforementioned model selection techniques with four tests, over 10 trials for the contamination and magnitude tests and over 20 trials for the coverage test on 1 and 10 subsets for the magnitude and coverage tests and 1 and 50 subsets for the contamination tests with 1,000 iterations on each MCMC chain and a burn-in period of the initial 500 iterations.

The first test is the contamination test which examines the root mean square error (RMSE) of held-out test data $\tilde{Y}$ of size 50 against the number of outliers present (as many as 5 in our experiments) in the training data, Y. We generate outliers by taking the maximum of the absolute value of the data and add a given magnitude value. Each outlier has a relative magnitude of 10,000 meaning that we find the largest output, Y_{i^*} such that $i^* = \text{argmax}_i \{|Y_i| : i = 1, \ldots, N\}$, so that the value of the outlier is $Y_{i^*} + (\text{sgn}(Y_{i^*}) \times 10000)$. For the contamination test, we expect to see superior performance with regards to RMSE of the 50 subset median posterior as long as the number of outliers per subset does not exceed 1. Figure 5.1 demonstrates the robustness of our technique to the number of outliers when we divide the data into subsets. We can see that the empirical 95% distribution of the RMSE over 10 trials for 50 subsets (green dashed line) falls dramatically below that of the RMSE distribution of 1 subset for each model selection technique when

outliers are present except in the case when 50 outliers are present for Bayesian model averaging which approaches the point where the theoretical guarantees of our method are violated.

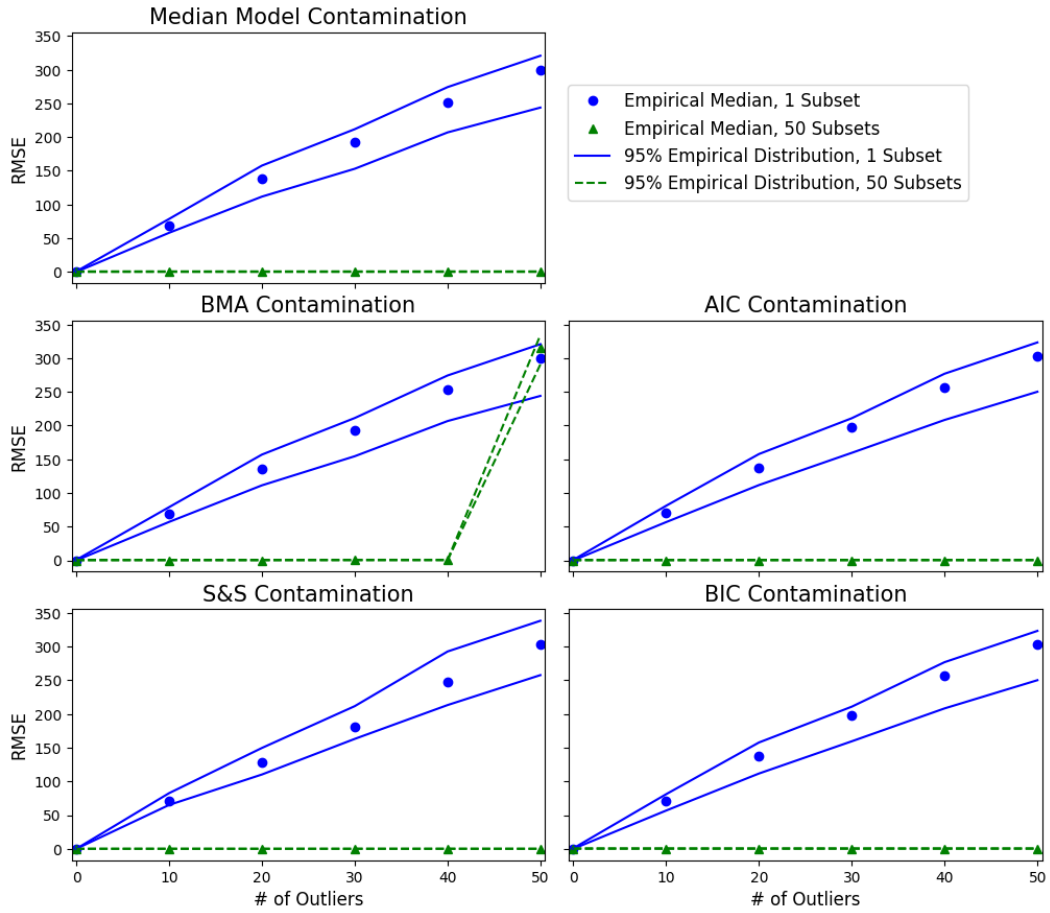

Figure 5.1: Contamination test.

The second test assesses the RMSE of the held-out test data of size 50 against the *increasing relative magnitude of one outlier* present in the training data. We expect to see nearly constant RMSE on the 10 subset run as the

relative magnitude of a single outlier increases, thus the procedure is robust. We can see in Figure 5.2 that the RMSE of distributed variants of the model selection techniques are lower than the single processor variants as the number of outliers increases. In the magnitude test, we can categorically observe that 10 subset RMSE is invariant to the relative magnitude of one outlier present in the data whereas the RMSE grows rapidly on one subset.

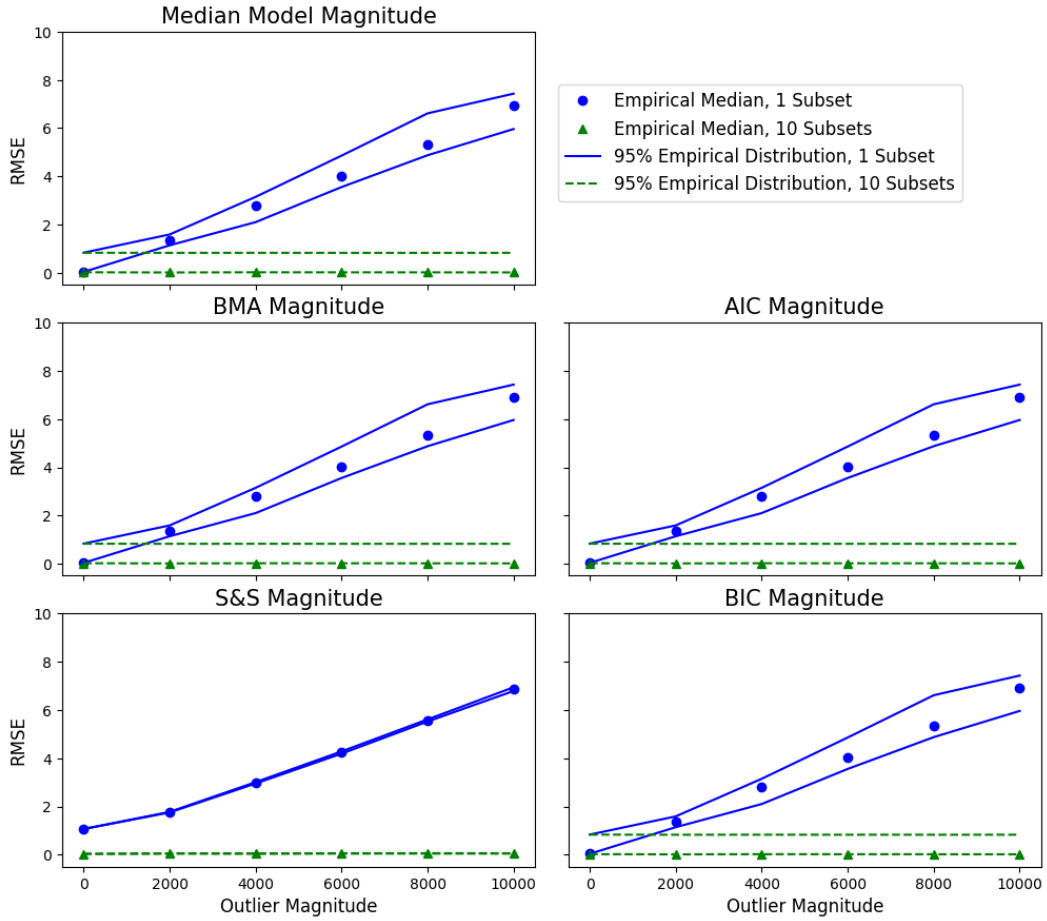

Figure 5.2: Magnitude of outlier test.

The next test assesses the 95% frequentist posterior coverage of the true held-out predictive value of size 1, $\tilde{Y}$, against the *increasing relative magnitude* of one outlier in the training data. To calculate coverage we generate 50 independent MCMC chains at each level of outlier magnitude and calculate the proportions of chains which include the true predictive value within the 2.5% and 97.5% percentiles of the posterior predictive draws. For the coverage test we see that the empirical coverage of a single predictive value for the distributed subsets is, on average, 95% regardless of the magnitude of the outlier as opposed to the empirical coverage for the single subset. In the 1 subset case, we can see that the empirical coverage degrades almost to zero as the magnitude of the outlier grows. (see Figure 5.3).

Our last evaluation is the coverage of the regression coefficients and the ability for our model selection techniques to *choose the correct model* under the distributed setting with a single outlier of magnitude 10,000. We compare the posterior credible interval of the regression coefficients for 1 and 10 subsets. Note that we do not include nested models in our evaluations or models larger than the true model (i.e models with more than 3 covariates included). Furthermore, we perform this evaluation under two settings: One, where we combine the optimal local model seleceted on each subset ("Model Combination") or if we combine the subposterior estimates and select the optimal model globally ("Estimate Combination") As seen in Figure 5.4, the parallel technique is able to select the correct model 1 subset test, the outlier leads to the incorrect model being selected. Additionally, Figure 5.5 demonstrates

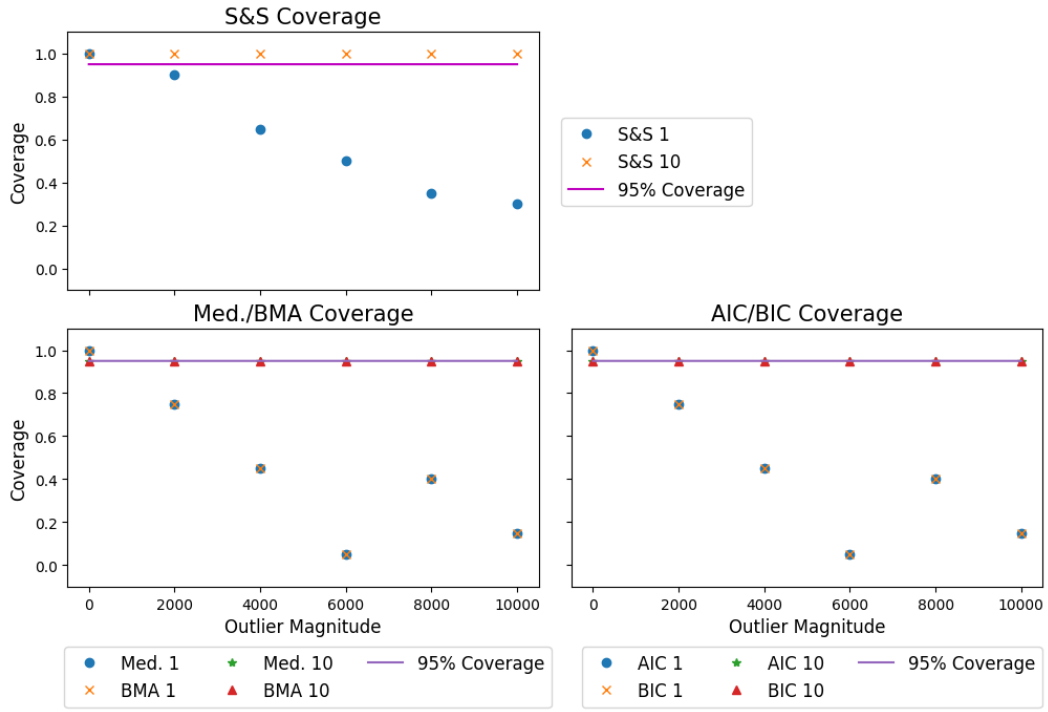

Figure 5.3: Testing empirical coverage of predictive value.

that model and estimate combination yield similar results with the regression coefficient coverage test.

Also, we would like to see if the results still hold between model and estimate combination for the other simulation studies performed. Figs. 5.6, 5.7, 5.8, 5.9, 5.10, 5.11, and 5.12 show that there is little difference in how we combine the information for model selection in each of the tests evaluated.

Furthermore, we wish to evaluate our method a large synthetic dataset with the same synthetic generating process as above, but with *one million observations* divided over 50 processors. Here, we examine the behavior of

our method when we increase the magnitude of one outlier in the dataset and when we increase the number of outliers with fixed magnitude. In Figure 5.13, we can see that our performance is robust when the number of outliers per subset fulfills Theorem 4. When the number of outliers reaches 40 and 50, we see start to see a noticeable degradation of our method's predictive ability. However, this degradation is still small relative to what we might observe in the case where we do not divide the data into subsets.

Additionally, we would like to see the computational gain of dividing the data for this situation in terms of CPU time for running the model selection and inference procedure. For one subset the average computation time is 91,829.15 seconds with a standard error of 190.80 seconds. For ten subsets, the average computation time is 10,301.60 seconds with a standard error of 81.28 seconds. And for fifty subsets, the average computation time is 29,49.74 seconds with a standard error of 16.61 seconds which signifies that we obtain critical computational performance when dividing our method across multiple processors.

Lastly, we evaluate our parallel model selection method on the diabetes data set used in [21]. The diabetes data consists of a 442×10 dimension design matrix scaled with unit norm and zero mean and a single response vector. We held out 45 observations for test evaluation and plotted the posterior 95% credible intervals for the predictive values centered at zero after subtracting the true predictive value. We can see in Fig. 5.14 that, after dividing the data across 5 subsets, we can attain a tighter credible interval over the true value

for each model selection technique.

5.5 Discussion

While a substantial body of work exists for fast and scalable Bayesian inference methods, few research methods are available on robust and scalable model selection. We have studied in this paper a divide-and-conquer strategy that contributes to filling this gap. This strategy operates by taking the geometric median of posterior model probabilities or other selection criteria that extends previous results focusing on parametric inference. We show theoretically how the strategy, particularly in the setting of BMA, can be robust to outliers and, moreover, exhibits faster concentration to the true model in terms of posterior model probabilities. The concentration result also applies to the joint setting of model selection and parameter estimation. We illustrate with both simulation data and a real data example how a variety of our strategy leads to more robust inference compared to standard approach that does not divide data into subsets. The strategy we present is simple to execute and is foreseen to have good practical value.

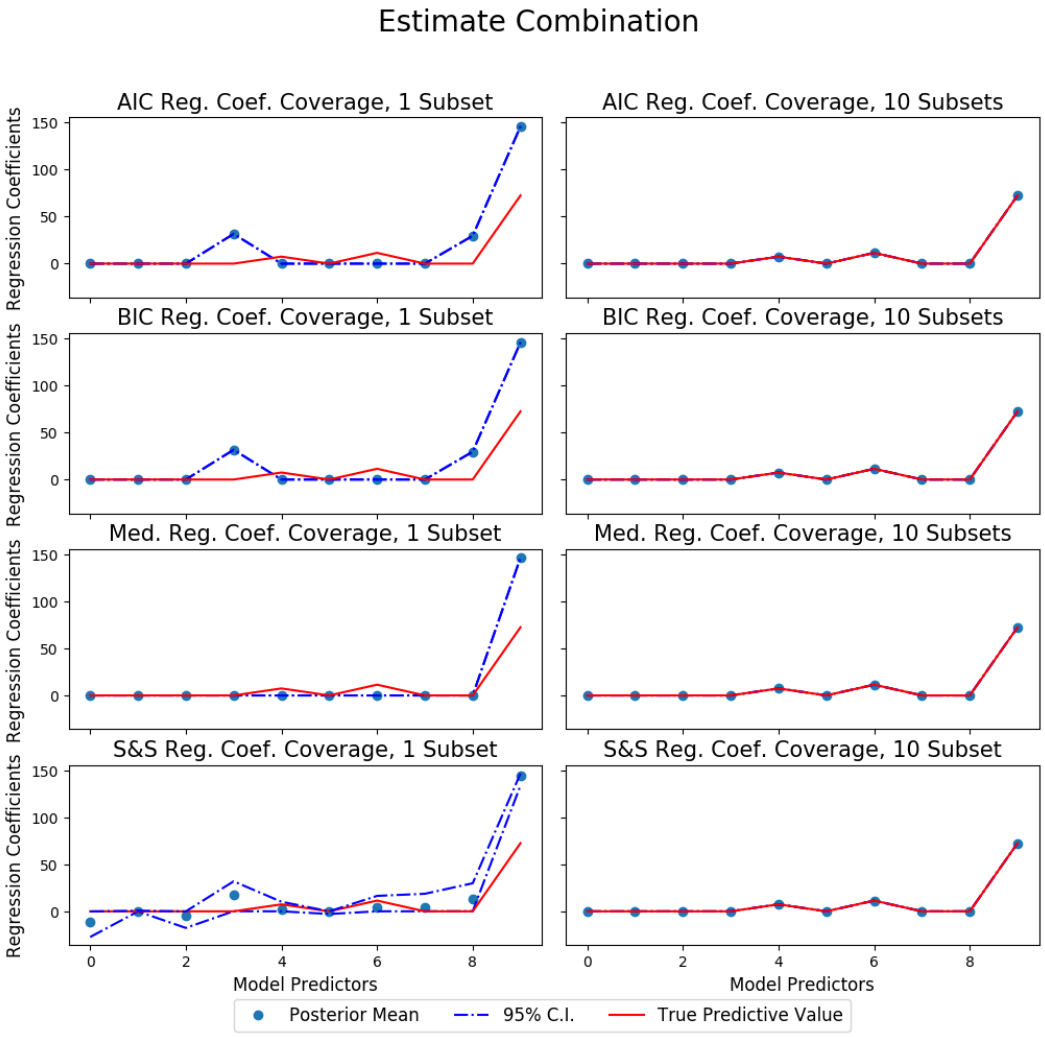

Figure 5.4: Posterior regression parameter coverage test results, estimate combination.

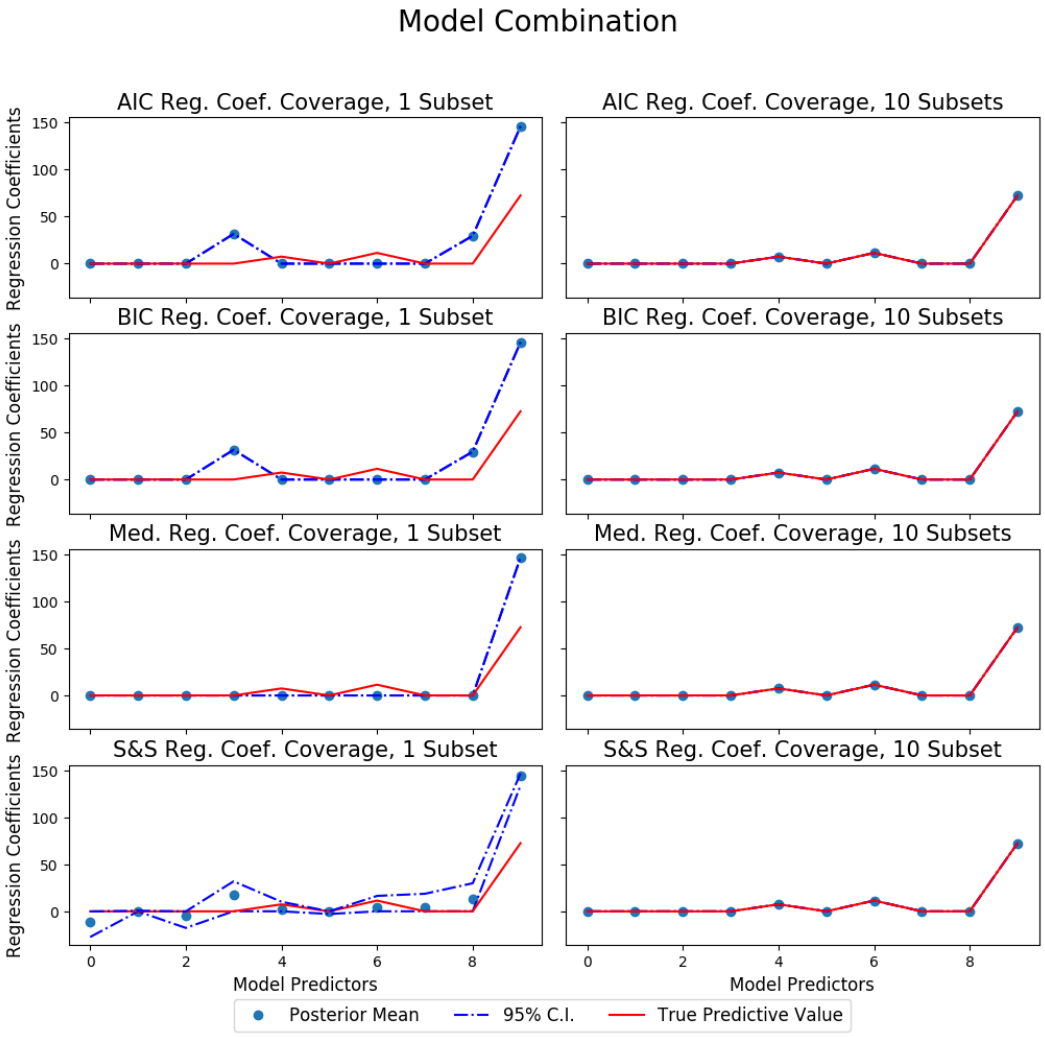

Figure 5.5: Posterior regression parameter coverage test results, model combination.

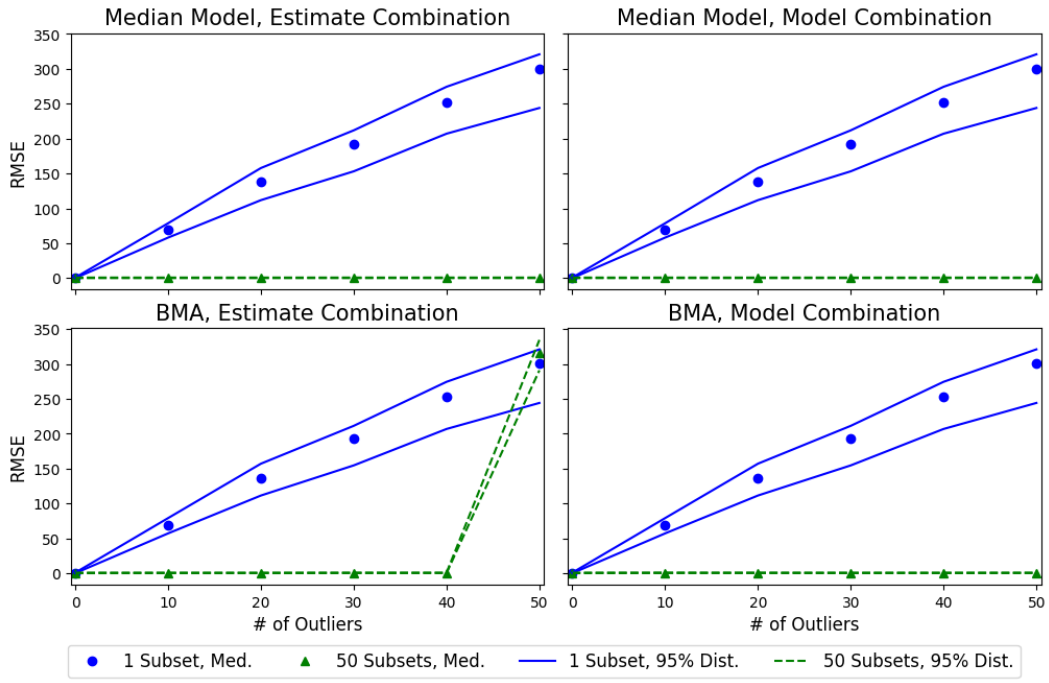

Figure 5.6: Contamination test, BMA and median model selection

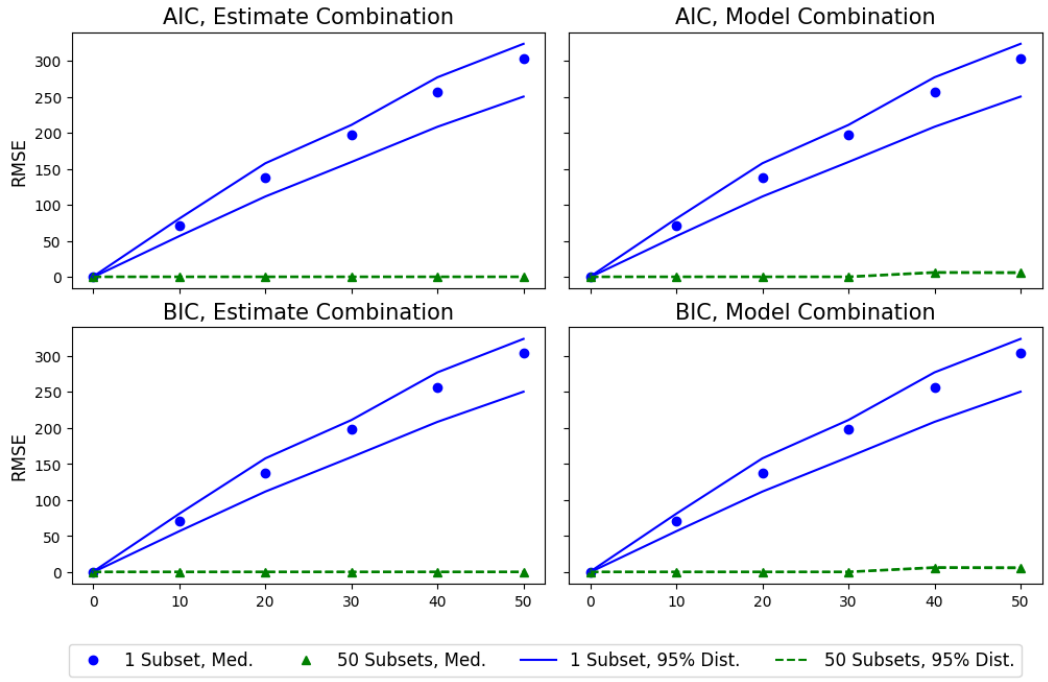

Figure 5.7: Contamination test, AIC and BIC

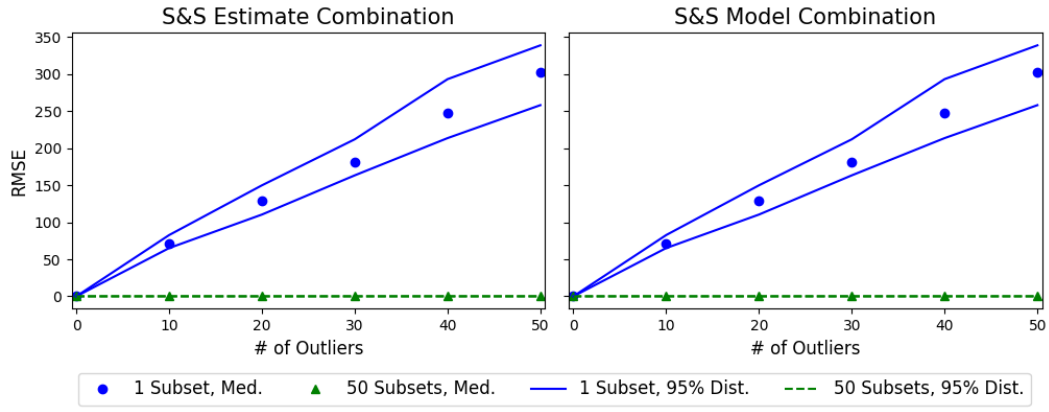

Figure 5.8: Contamination test, spike and slab

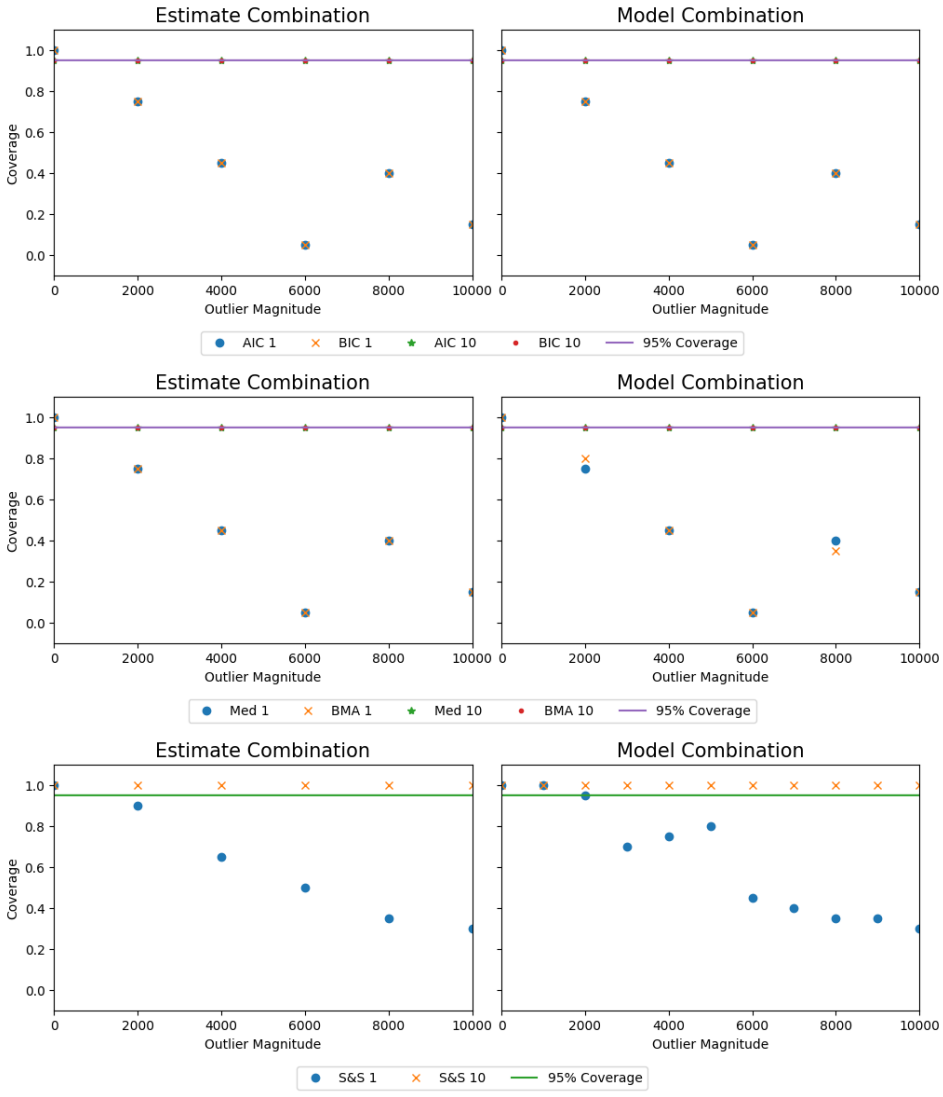

Figure 5.9: Coverage test.

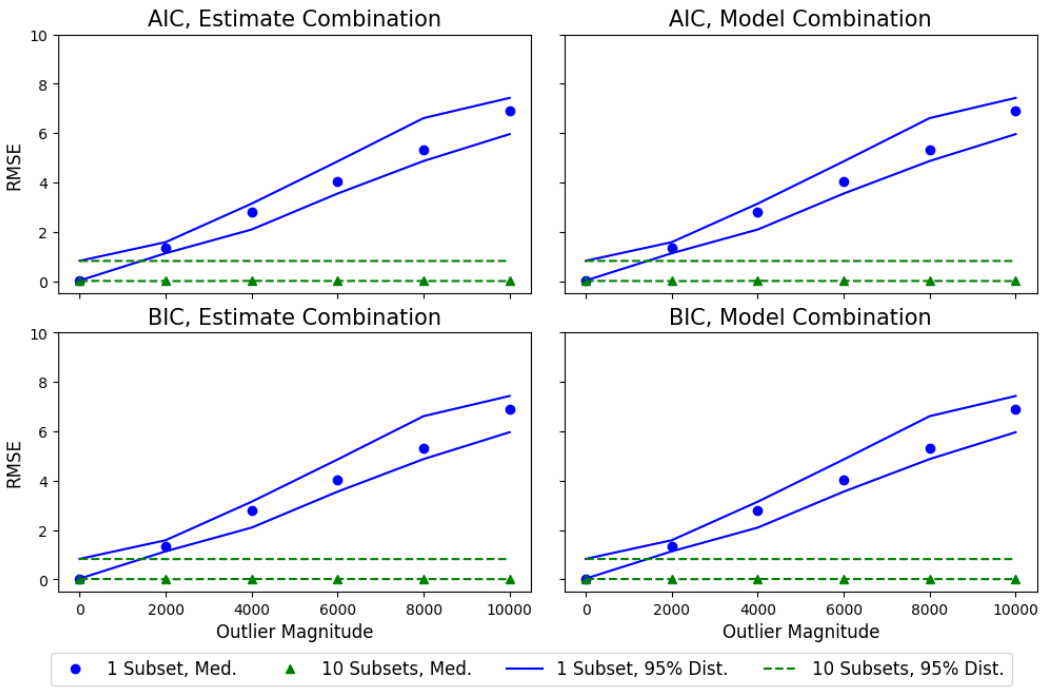

Figure 5.10: Magnitude test, AIC and BIC.

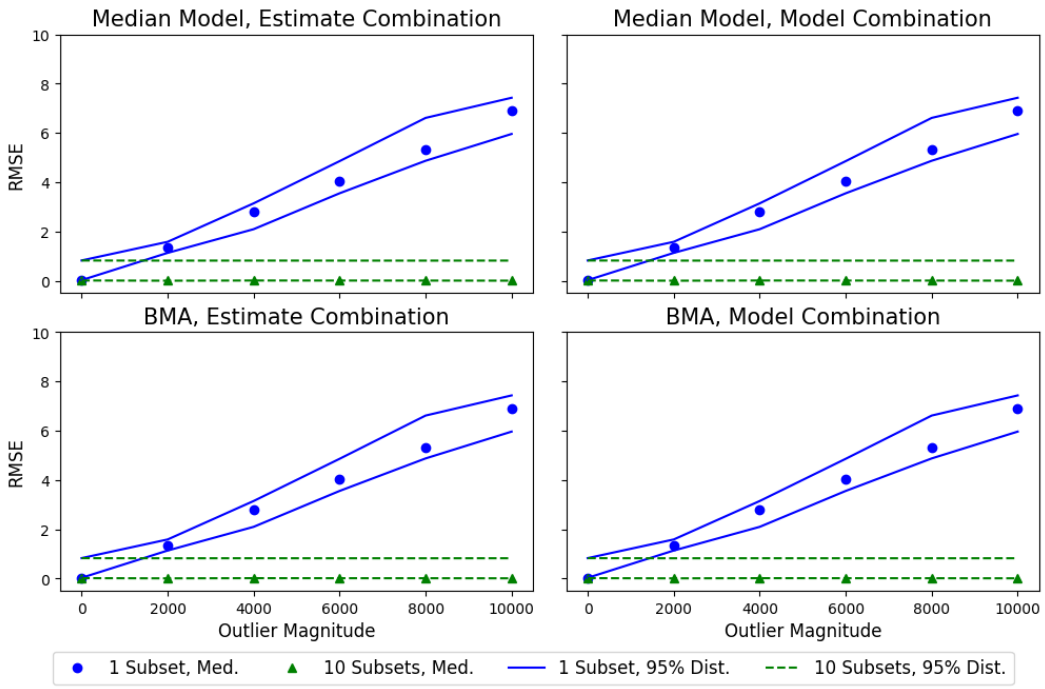

Figure 5.11: Magnitude test, BMA and median model selection.

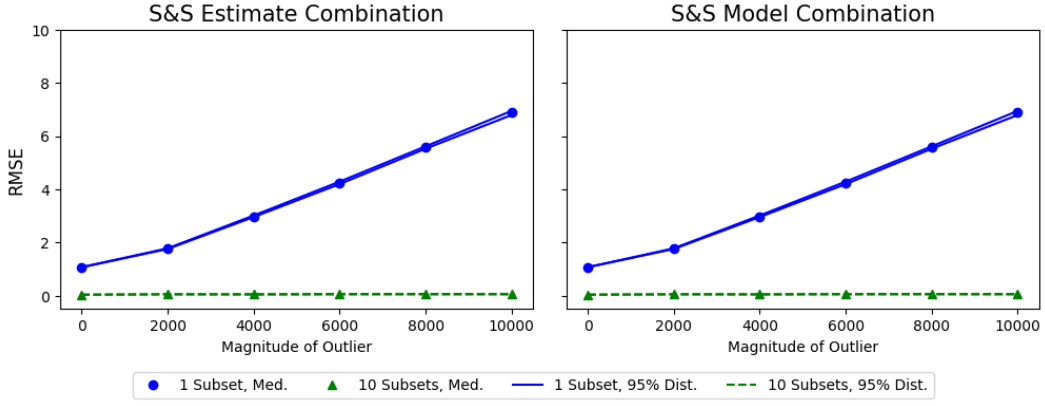

Figure 5.12: Magnitude test, spike and slab.

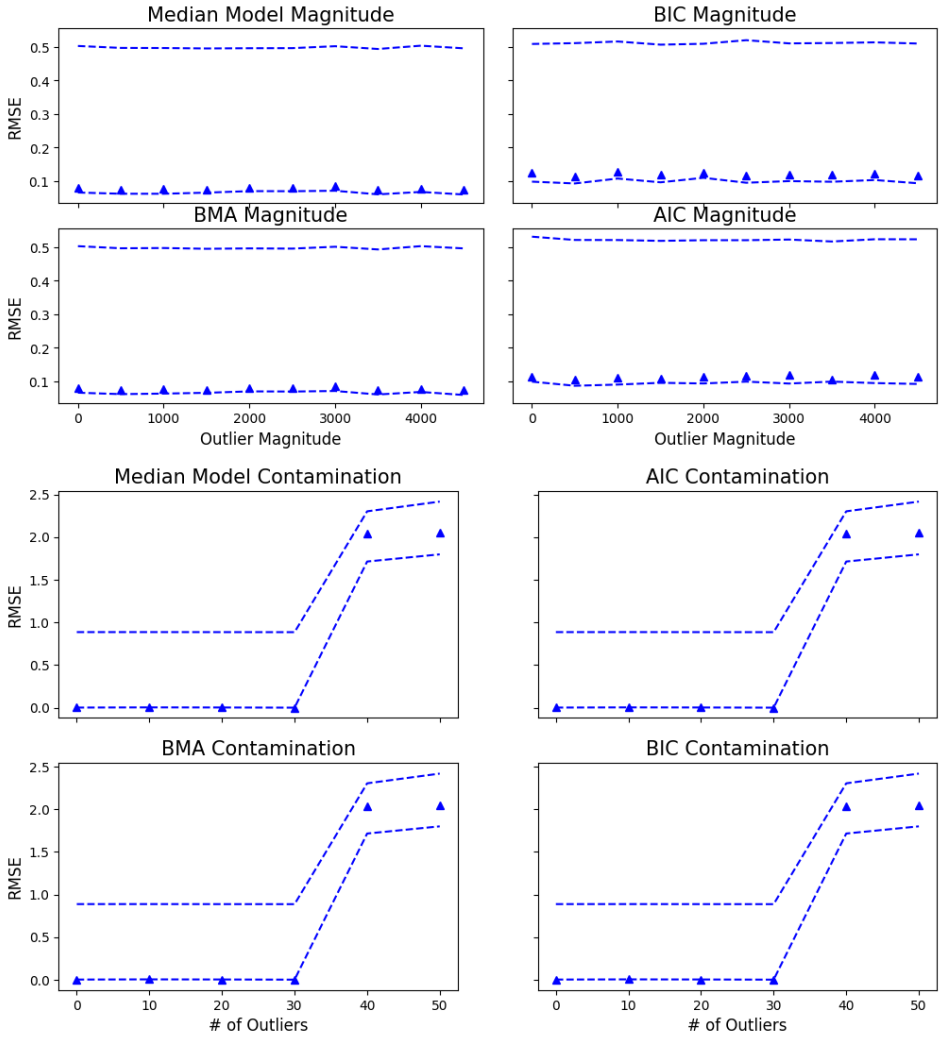

Figure 5.13: Synthetic big data results.

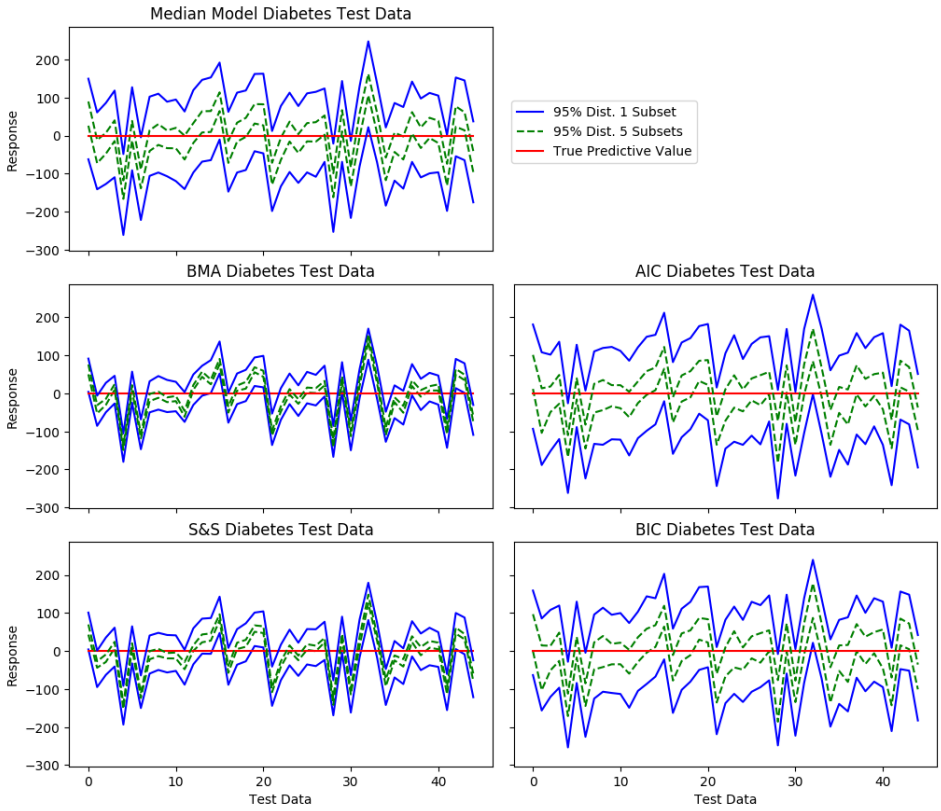

Figure 5.14: Diabetes test data results.

Chapter 6

Embarrassingly Parallel Inference for Gaussian Processes

Many problems in statistics and machine learning can be framed in terms of learning a latent function $f(x)$. For example, in regression problems, we represent our dependent variables as a (noisy) function of our independent variables. In classification, we learn a function that maps an input to a class (or a probability distribution over classes). In parameter optimization, we have some function that maps parameters to their likelihoods and want to find the optima of this function. However, if we assume the relationship between x and $f(x)$ is nonlinear then learning the latent function is a non-trivial task.

Gaussian processes (GPs) provide a flexible family of distributions over functions, that have been widely adopted for problems including regression, classification and optimization due to their ease of use in modeling latent functions. Unfortunately, inference in GP models with N observations involves repeated inversion of an $N \times N$ matrix, which typically scales as $O(N^3)$. This computational bottleneck has thus far prevented Gaussian processes from being used in so-called "big data" situations.

Two main approaches have been proposed to ameliorate the computa-

tional complexity for inference: sparse methods, that aim to reduce the size of the matrix to be inverted, and local methods, that aim to simplify its structure. Unfortunately, both methods exhibit key failure modes as we reduce the computational cost: local methods can miss long-range correlations, and sparse methods tend to miss short-range fluctuations. Further, methods of these types are not typically parallelizable to run efficiently on a distributed architecture.

Moreover, the typical Gaussian process regression model fit with a stationary covariance kernel is not flexible enough to model latent functions that exhibit idiosyncratic behavior at different locations in the input space. While non-stationary covariance kernels are certainly available, in practice they are slow to use. Using mixtures of Gaussian processes or combining partitioned Gaussian processes with stationary covariance kernels is another flexible way of modeling non-stationary latent functions but typical inference procedures for these models are slow. To this end we wish to construct an inference procedure for such mixtures with the explicit goal of scalable computation.

This chapter proposes a novel inference algorithm for fitting mixtures of partitioned Gaussian processes that is scalable, flexible and easily distributed. The "Importance Gaussian Process Sampler" (IGPS) approximates the covariance matrix with an importance averaging over block diagonal matrices. We learn, in parallel, multiple partitioned Gaussian processes sampled from the posterior partitioning distribution over the inputs, allowing us to take advantage of the lower inversion cost of a block-diagonal matrix. We then use

importance sampling to combine these estimates in a principled manner – the only step in our algorithm requiring global communication. To obtain even greater speedups while maintaining competitive performance, we use stochastic approximations obtained using minibatches. The resulting posterior predictive distribution has a more expressive expected covariance matrix than a block diagonal matrix, avoiding edge effects common with local methods and allowing for an expressive covariance structure that can model both long- and short-range covariance as well as non-stationary behavior in the latent function.

6.1 Related Work

A Gaussian process, as introduced in Section 2.3.5, is a distribution over functions $f : \mathbb{R}^D \to \mathbb{R}$, parametrized by some mean function $m(x)$, typically taken as zero, and a covariance function $\Sigma(x, x')$. For a given m and Σ, a GP is a unique distribution over functions f such that for any finite set of points, $x_1, \ldots, x_N \in \mathbb{R}^D$, the function evaluated at those points is multivariate normally distributed with mean and covariance given by m and Σ evaluated at these inputs.

The behavior of functions sampled from a Gaussian process is largely determined by the covariance function. A covariance function where $\Sigma(x, x')$ only depends on $|x - x'|$ will produce stationary, isotropic functions. The most commonly used covariance function of this type is the squared exponential (or RBF) kernel, $\Sigma(x, x') = v \exp\{-\gamma(x - x')^2\}$. Here, the (inverse) lengthscale γ

determines how quickly $f(x)$ varies with x. If we believe that our function is non-stationary, a covariance function that depends on location is appropriate. While a number of such functions exist [31], they are generally harder to work with than stationary kernels with more parameters to tune.

Gaussian processes have been used to provide prior distributions over functions in a variety of applications [see for example 76, 85, 97]. In this exposition we focus primarily on regression but we note that the extension to other settings is straightforward. In the regression setting we want to learn a function $f \sim \mathrm{GP}(0, \Sigma)$ that maps our inputs $X = (x_i)_{i=1}^N$ to our outputs $Y = (y_i)_{i=1}^N$ such that $y_i \sim \mathrm{N}(f(x_i), \sigma^2)$. In this setting, conjugacy means that inferring f given the data and covariance function is straightforward. The challenge comes in inferring the hyperparameters, Θ, that control the form of the covariance function. Optimizing or sampling these hyperparameters involves inverting the covariance matrix $\mathbf{\Sigma}$ obtained by evaluating $\Sigma(\cdot, \cdot)$ at the inputs $x_1, \ldots, x_N$. In general, the computational cost of inverting this matrix is $O(N^3)$.

6.1.1 Reducing the Cost of Matrix Inversion With Covariance Approximations

One way to reduce the $O(N^3)$ cost is to use an easily-invertible class of covariance matrices (either as an approximation to some desirable but computationally demanding covariance matrix, or as a design choice in specifying the GP). Two broad classes of methods have been proposed: "sparse" meth-

ods which parametrize the covariance based on $M << N$ inducing inputs, and "local" methods that replace the dense $N \times N$ covariance matrix with a block-diagonal matrix.

Sparse GP approximations parameterize the covariance matrix of the GP model with M pseudo-inputs, where $M << N$. The pseudo-input locations are chosen so that the posterior function evaluated at these points is a good approximation to the true posterior, for example by maximum likelihood optimization [84] or variational inference [92]. The computational saving comes from replacing an $N \times N$ covariance matrix with an $M \times M$ matrix. In the regression case, this reduces the training cost to $O(NM^2)$. Further computational savings can be obtained by using stochastic variational inference (SVI) to update the inducing points by calculating necessary gradients based only on size-B subsets of the N datapoints, reducing computational cost to $O\left(M^2 \min\{M, B\}\right)$ [39]. While sparse methods can yield impressive speedups, they tend to have a decreased ability to model high-frequency fluctuations in the function, since the number of inducing points limits the amount of variation we can capture. Additionally, as with the full-covariance GP it approximates, the sparse GP cannot naturally model non-stationary data without resorting to a non-stationary kernel.

Local Gaussian process methods make local approximations to the dense covariance matrix so that a low-rank representation of the covariance matrix is inverted instead of the full-rank matrix. [51] applies a "tapering" function to the covariance matrix so that observation pairs with low corre-

lation are set to zero and provides theorems for estimator consistency when the covariance function used is a Matérn kernel. [33] tries to learn the local approximation by taking the n-nearest neighbors of a predictive value X^* to the data X and learns both the function hyperparameters and predictive distribution jointly by iteratively increasing the size of the nearest neighbors until a stopping criteria is satisfied for all predictive inputs. [6] approximates a full Gaussian process model using a smaller subset of the data to form a prediction of the entire model.

Product of experts models [94, 14, 68, 18] multiply the predictions of multiple local Gaussian processes. Conditioned on the partitioning, inference in the local GP scales approximately as $O(N^3/K^2)$, since we need to invert K matrices of average size $\frac{N}{K} \times \frac{N}{K}$. A naive implementation of this type of local GP method leads to discontinuities at partition boundaries, motivating various techniques for mitigating this effect. Park et al. [73] use a boundary value function to ensure continuity between regions. Mixture of experts models [34, 75, 60] average over multiple local GPs, for example using MCMC.

One advantage of local methods is that they allow us to use different covariance hyperparameters in different blocks. Without resorting to complex non-stationary kernels, this allows us to capture behavior which is locally approximately stationary, but where the lengthscale varies across the input space [94, 75]. This is in contrast with sparse methods, which can only capture non-stationarity if we use an explicitly non-stationary covariance function. The disadvantage of the local methods, however, is that they risk ignoring impor-

tant correlations since they assume zero correlation between different blocks in the partition. If the data points are partitioned based on location, this means that long-range correlations will be ignored; if they are partitioned randomly, the model will tend to perform poorly if the number of observations in some region of $\mathbb{R}^D$ is low. This can be ameliorated by using MCMC to explore a distribution over partitions of Gaussian processes, as in the mixture of experts models. For example, Gramacy and Lee [34] show that jointly learning the partition and the local GPs can improve performance over a fixed partition. This improved performance comes at a cost however: performing MCMC-based inference over partitions can be expensive and (due to the Markovian relationship between samples) precludes direct parallelization.

6.1.2 Distributed Inference for Gaussian Processes

The sparse and local approximations described above aim to reduce the overall computational burden by reducing the size of matrices to be inverted. When run on a single machine, this reduction in computational cost leads directly to faster inference. However, we may also be interested in distributing computation cost across multiple threads or machines. Even if the total computational cost is the same, we can reduce total time by distributing computation onto multiple parallel threads. Alternatively, if we increase the computational budget then we may be able to improve our posterior estimate by running multiple samplers in parallel and then combining the results without increasing the time budget.

Local GP methods that do not average over partitions are well suited to this sort of parallelism. They split a single GP problem into K independent problems whose parameters can be inferred in parallel. We only need to communicate between the K subproblems at the end when we combine their predictions. This type of algorithm, where global communication occurs only once after all the local computation is complete, is known as "embarrassingly parallel". Ng and Deisenroth [68] exploit these independences, in a weighted product-of-experts model, to obtain an embarrassingly parallel algorithm appropriate for large datasets. The embarrassingly parallel nature of these algorithms relies on the fact that we are using a single partition over the data and an importance sampler to average over the space of partitions. The ability to parallelize is lost if we use an MCMC sampler to learn a distribution over partitions, due to the dependence between samples.

6.1.3 Fast Bayesian Inference via Stochastic Approximations

When performing Bayesian inference on large datasets, much of the computational cost is due to calculating functions – for example, gradients or likelihoods. One way to reduce computational costs is to approximate these functions using noisy estimates based on much smaller subsets of the data. For example, sparse variational inference [43] uses minibatches of data to approximate gradients in a variational context. Stochastic gradient MCMC methods [59, 100] perform a similar approximation in a gradient-based MCMC setting. Several embarrassingly parallel MCMC methods combine noisy posterior es-

timates obtained using subsets of the data [88, 64]. The Bayesian coresets approach aims to learn the posterior based on a reweighted posterior [45]. In a Gaussian process context, as mentioned in Section 6.1.1, SVI has been used to speed up inference in sparse Gaussian processes from $O(NM^2)$ to $O\left(M^2 \min\left\{M, B\right\}\right)$, where B is the minibatch size.

6.2 Embarrassingly Parallel Inference with the "Importance Gaussian Process Sampler"

As described in Section 6.1, local GP methods reduce the computational burden of inverting the $N \times N$ covariance matrix by replacing or approximating it with a block-diagonal matrix with K blocks. Local methods using a fixed partition are easily parallelized to use K threads, each costing $O(N^2/K^2)$, but doing so ignores some of the correlations between data points. In particular, if the partitioning is based on input location (as is common), we ignore long-range correlations. However, averaging over partitions using MCMC is expensive and difficult to parallelize.

Our approach uses a distribution over partitions—allowing long-range correlations and a dense expected covariance matrix—but uses an importance-sampling-based inference scheme that is trivially parallelizable to approximate the marginalization over the space of partitions. In this sense, our method is a *mixture of partitioned Gaussian processes* and not an approximation of the standard Gaussian process regression model. By this method, we can both approximate the covariance matrix in a stationary model or provide a cheap

and fast way to model non-stationary functions if we allow each partition to have its own set of hyperparameters.

Other mixtures of Gaussian processes have been developed for the purpose of forming more flexible models that could model, for example, non-stationarity in the latent function without resorting to a non-stationary covariance kernel [75]. Partitioning approaches like the Bayesian treed Gaussian process [34], like our method, partition the input space and average over the partitions for generate predictions. Unlike previous approaches, our intended objective is to develop a fast, distributable Gaussian process model that, as a consequence of importance averaging over partitions, is also a more flexible approach than the typical Gaussian process regression model.

We wish to explore the posterior distribution over $\mathbf{\Sigma}$ (given the prior implied by the distribution over partitions) while ensuring our algorithm can be distributed. To this end, we independently sample J partitions from the distribution $Z^{(j)} \sim p(Z|X)$ and assign them weights $w_j \propto p(Z|X,Y)/p(Z|X)$, such that $\sum_j w_j = 1$. We then use the weighted samples to obtain asymptotically unbiased[1] estimates of functions of the posterior, such as predictions at new inputs X^*.

Calculating the w_j involves integrating over the covariance parameters

[1]Since we are working with self-normalized weights, the estimate has a bias of $O(1/J)$ [55], but will often have a lower variance than the unbiased estimate obtained with $w_j = p(Z|X,Y)/p(Z|X)$.

Θ_k:

$$\frac{p(Z|X,Y)}{p(Z|X)} \propto \frac{p(X,Y|Z)p(Z)}{p(X|Z)p(Z)} = p(Y|X,Z) = \int p(Y|X,Z,\Theta)p(\Theta)d\Theta. \quad (6.1)$$

We must approximate this intractable integral. Depending on our accuracy/speed trade-off, we can obtain an unbiased estimate of the w_j using a sample-based approximation. Alternatively, we can perform a Laplace approximation about the MAP $\hat{\Theta}$ or we can directly use the MAP approximation $p(Y|X,Z) \approx p(Y|X,Z,\hat{\Theta})$. In our experiments, we choose a MAP approximation; while this is not as accurate as sampling hyperparameters it is significantly faster, and mirrors the choices made by our comparison methods.

To perform prediction under our method we must first calculate the prediction on importance proposal j, by averaging the predictions produced on each of the K partitions weighted by the posterior cluster assignment probabilities of the predictive inputs[2]:

$$P(f_j^*|-) = \sum_{k=1}^{K} P(f_j^*|X_{k,j}, Y_{k,j}, Z_j, X^*, Z_j^*)P(Z_j^*|X_{k,j}, Z_j). \quad (6.2)$$

Next, we must obtain the weights to calculate the importance averaging of the predictions f_j^*. Our choice for the proposal distribution in the importance sampler results in an importance weight proportional to the marginal likelihood of the model, given the partition assignments:

$$w_j \propto P(Y|X,Z_j) = \prod_{k=1}^{K} P(Y_{k,j}|X_{k,j},Z_j). \quad (6.3)$$

[2]In the interest of computational speed, we approximate this distribution with the MAP estimate: $P(f_j^*|-) \approx \max_k P(f_j^*|X_{k,j}, Y_{k,j}, Z_j, X^*, Z_j^*)$.

We already calculate this marginal likelihood in fitting the model produced on each of the J proposals so we, in effect, obtain the importance weights (up to proportionality) for "free". Next we must normalize the weights and average the prediction from each proposal to calculate the final prediction,

$$P(\bar{f}^*|-) = \sum_{j=1}^{J} w_j P(f_j^*|Z_j, -). \tag{6.4}$$

The overall computational cost of the IGPS, using J importance-weighted samples and K blocks, is therefore $O(JN^3/K^2)$. In Table 6.1, we compare this with the overall computational cost of the full GP, sparse approximations (FITC, DTC and SVI), the Bayesian treed GP (BTGP), and the robust Bayesian committee machine (RBCM). While the $O(JN^3/K^2)$ cost is $O(J)$ higher than sparse methods[3] and local methods based on a fixed partition such as RBCM, we note that the J samples can be performed and weighted in parallel—meaning the time taken is comparable if we are willing to sacrifice computational resources. In this procedure, the only communication between processors occurs at the end of the prediction step when we normalize the weights, w_j, and obtain the importance averaged predictions, $\bar{f}^*$. This is vital in any distributed computation algorithm due to the high overhead cost of inter-processor communication. We can also make use of the independence of the K partitions to parallelize further, using JK threads each taking $O(N^3/K^3)$. As shown in Table 6.1, this leads to an equivalent wall-time cost

[3]In general, a sparse model with M inducing points obtains comparable accuracy to a local method with N/K local GPs, and has equivalent computational complexity.

comparable with the distributed RBCM. As we will see in Section 6.3, the extra computational cost required to ensure a full posterior predictive distribution yields improved performance over methods that are based on a fixed partition.

6.2.1 Minibatched Importance Samples

Although we can obtain significant computational and memory saving advantages using IGPS's low-rank approximation, we still may encounter major bottlenecks from attempting to approximate the covariance matrix of the full training set. To overcome this issue, we propose a "minibatching" solution, where each importance sample is obtained and weighted based only on a subset of size $B << N$.

Given a random subset B of observations, we can approximate $p(\Theta|X, Y, Z)$ with the subset posterior $p(\Theta|X^{mb}, Y^{mb}, Z)$ evaluated on a size-B minibatch (X^{mb}, Y^{mb}). Such a posterior estimate is strongly consistent, but will tend to underestimate the posterior variance [88]. To achieve more realistic credible intervals, we can assume we have seen each pair (x_i, y_i) in our minibatch N/B times; mathematically, this corresponds to raising the contribution of the likelihood to the subset posterior to the (N/B)-th power. This approach, introduced by Minsker et al. [64], is known as the stochastic approximation (SA) trick; Srivastava et al. [88] show that the SA sub-posteriors are also strongly consistent.

We use this stochastic approximation trick to estimate the posterior

Table 6.1: Comparison of inference complexity. N is the number of data points, K is the number of experts or local GPs, and $M = N/K$ is the number of inducing points. For the Monte Carlo based methods, J is the number of MCMC iterations or importance samples.

	Full GP	Sparse	SVI	
Complexity	N^3	NM^2	$M^2 \min(M, B)$	
	RBCM	BTGP	**IGPS**	**SA-IGPS**
Complexity	N^3/K^2	JN^3/K^2	JN^3/K^2	JB^3/K^2
Comp/thread	N^3/K^3	$\times$	N^3/K^3	B^3/K^3

distribution over parameters for each importance sample, allowing us to reduce our overall complexity from $O(JN^3/K^2)$ to $O(JB^3/K^2)$. Empirical results in Section 6.3 will show that this stochastic approximation performs favorably on large datasets in comparison with both the non-SA IGPS method and other scalable GP inference methods.

6.3 Experimental Evaluation

To showcase the performance of our method, we compare it with a number of competing methods on both synthetic and real data sets.

6.3.1 Evaluation on Synthetic Data

6.3.1.1 Comparison with Competing Methods

We begin by evaluating our method on synthetically generated data, in order to allow us to explore and visualize a range of regimes, and to allow comparison with methods that do not scale to our real-world dataset. In our

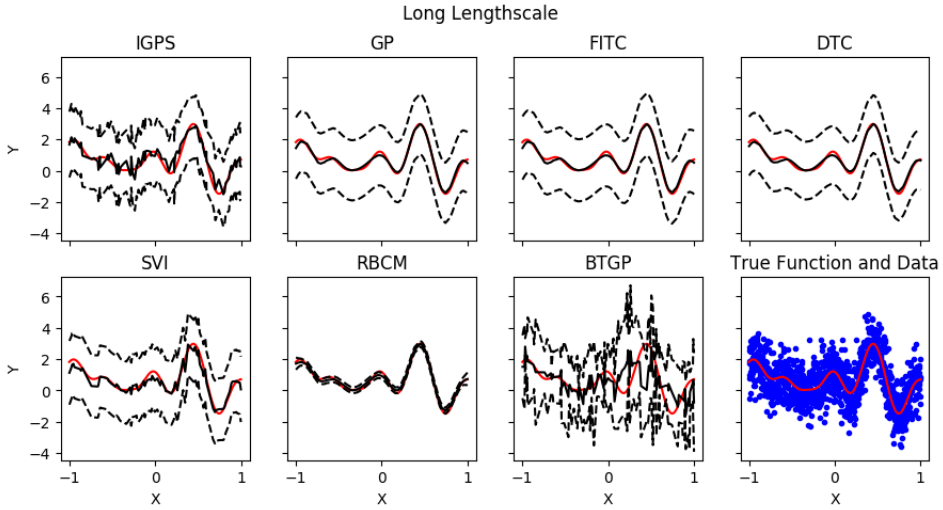

Figure 6.1: Posterior mean and 95% predictive intervals on Synthetic 1 (stationary, long length scale).

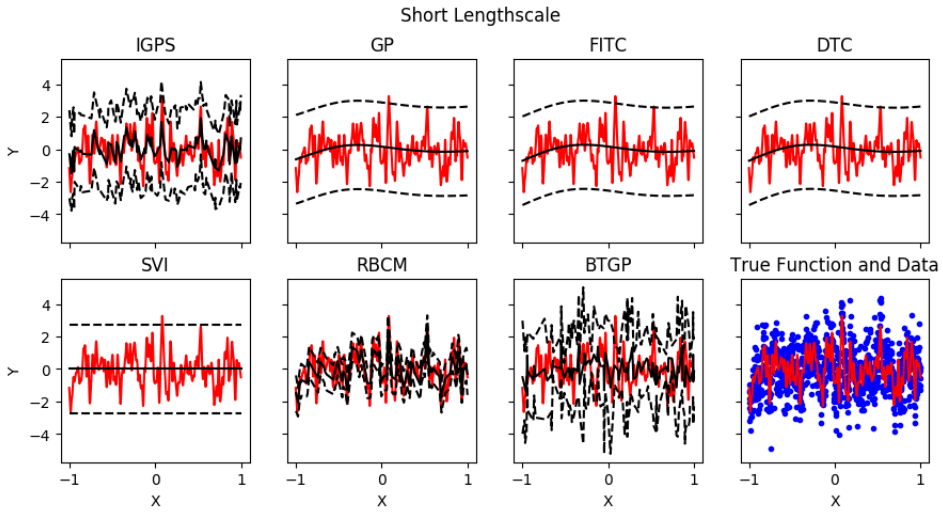

Figure 6.2: Posterior mean and 95% predictive intervals on Synthetic 2 (stationary, short length scale).

Table 6.2: Test set performance on synthetic datasets

(a) Log likelihood

Data	**IGPS**	GP	FITC	DTC
Long Lengthscale	-152.41	-143.52	-143.39	-143.81
Short Lengthscale	-157.16	-172.32	-172.26	-172.26
Non-stationary	-158.21	-181.40	-181.30	-181.30

Data	SVI	RBCM	BTGP
Long Lengthscale	-156.51	-5207.89	-231.84
Short Lengthscale	-173.38	-251.50	-212.61
Non-stationary	-198.00	-910.54	-256.73

(b) MSE

Data	**IGPS**	GP	FITC	DTC	SVI	RBCM	BTGP
Long Lengthscale	1.20	1.03	1.02	1.02	1.30	1.03	2.07
Short Lengthscale	1.35	1.83	1.83	1.83	1.87	1.06	2.46
Non-stationary	1.39	2.18	2.17	2.17	2.12	1.00	2.64

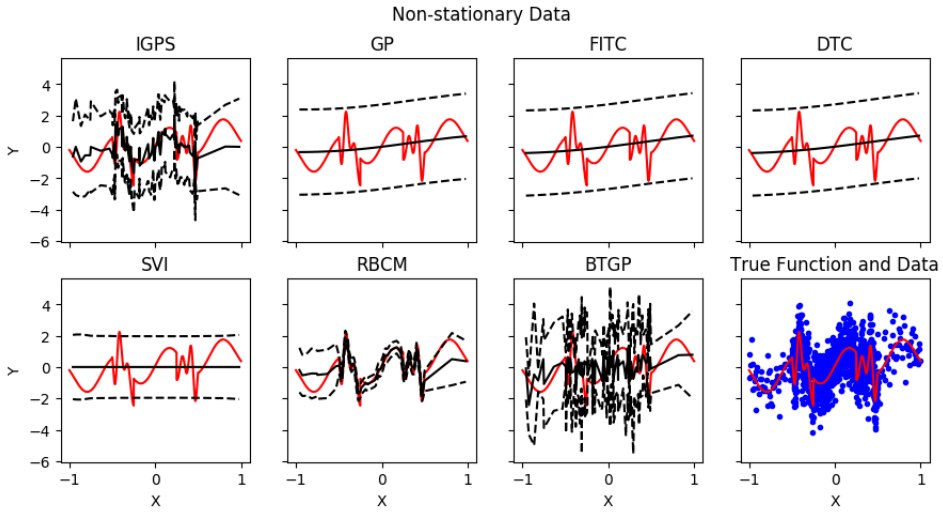

Figure 6.3: Posterior mean and 95% predictive intervals on Synthetic 3 (non-stationary).

studies, we will compare the Importance Gaussian Process Sampler against a full GP with squared exponential covariance (GP); three sparse approximations to this model, FITC [84], DTC [80], and SVI [39]; the Bayesian treed GP [34, BTGP]; and the robust Bayesian committee machine [18, RBCM]. Our IGPS code uses the Gaussian process modules in `GPy` in Python with parallelization executed through `MPI4py` [17]. For these experiments we did not use minibatching. We ran the full GP, FITC, DTC and SVI implementations also through `GPy`, BTGP in `tgp`, and RBCM in `gptf`.

We consider three data settings, generated on a linearly spaced grid of values on $[-1, 1]$.

1. **Stationary, long-range correlations** generated with inverse length scale $\gamma = 15$.

2. **Stationary, short-range correlations** generated with inverse length scale $\gamma = 5000$.

3. **Non-stationary** generated piecewise with fast and slow moving periodic functions.

In examples (1) and (2), we generated data from a GP with zero mean squared exponential covariance kernel with amplitude $\nu = 1$. For all examples we added Gaussian noise $\sigma^2 = 1$ to the observed outputs. We generated a training data set with 1,000 observations and a test set with 100 observations.

For fitting the stationary data, we restrict the hyperparameters on our IGPS method to be the same on all K blocks. For (3), we allowed each mixture to have its own hyperparameters in order to model the non-stationarity of the data. In all methods except BTGP we infer hyperparameters through optimization, and for BTGP we infer the hyperparameters through MCMC sampling. For the sparse methods, we used $M = 100$ inducing points, and for the local methods (including the IGPS) we used $K = 10$ partitions to have a comparable level of computational complexity. For the BTGP we ran the MCMC sampler for 10 iterations; for the IGPS we used $J = 10$ independent importance-weighted samples. Figures 6.1, 6.2, and 6.3 shows the posterior predictive results and predictive intervals obtained using the five methods, and Tables 6.2a and 6.2b show the corresponding test set log likelihoods and mean squared errors.

We first consider the one-dimensional stationary examples. Recall that,

in general, sparse methods perform well when the covariance structure is dominated by longer-range correlations, and local methods perform well when we have significant local variation in our function. Looking at the results on the dataset with long-range correlations (Figure 6.1), we see that the IGPS performs better than the other local methods, and performs nearly as well as the full GP and the sparse approximations.

If we look at the dataset with short-range correlation (Figure 6.2), we see the sparse methods struggle to learn the function: with a small number of inducing points, it is impossible to capture the high-frequency variation. Looking at the quantitative results in Tables 6.2a and 6.2b, we see that the IGPS out-performs the RBCM. We posit that this is because the IGPS makes use of the distribution of the covariates to learning a distribution over partitions, while the RBCM relies on a single fixed partition. While the BTGP also averages over partitions, its poor predictive performance is likely because of lack of convergence of the MCMC chain: the underlying model is fairly complex and likely to mix slowly.

Finally, consider the non-stationary example, which combines known failure modes of local and sparse GPs. We have a combination of slowly varying behavior (which is poorly captured by local methods) and fast-varying behavior (which is poorly captured by sparse methods). The full GP, RBCM and sparse methods, fitted with stationary kernels, obviously cannot account for the non-stationary components in the data, and by assuming a stationary covariance they give poorer test-set performance. The BTGP does a reasonable

job at capturing the function; again its performance is likely to be hampered by slow mixing and lack of convergence. Figure 6.3 shows that the IGPS is able to capture the function, and Tables 6.2a and 6.2b show that it can provide confident predictions at all regions of the function.

6.3.1.2 The Importance of Importance Sampling

The IGPS falls under the "local" framework, much like the RBCM and the BTGP; however it out-performs both methods. This can be attributed to importance sampling a distribution over partitions. To demonstrate this, we consider performance on a synthetic dataset of 10,000 training observations from a 12000×100 dimensional input which is generated from a 50 mixture GMM with outputs drawn from a GP model with zero mean and an long lengthscale RBF kernel.

The RBCM uses a single, fixed partition. Conversely, the IGPS uses a distribution over partitions, combined using importance sampling weights. Figures 6.4 and 6.5 show how varying the number of importance samples, for a range of values of K and B (remember, $K = 1, B = N = 10,000$ corresponds to the full Gaussian process, and as K increases or B decreases, we expect a drop in quality). In most cases, we see a similar pattern: there is a clear improvement in performance between $J = 1$ to around $J = 50$, but beyond that the improvements level off. This confirms that averaging over partitions improves performance, but suggests that in this setting, we need relatively few samples to approximate the posterior. In Figure 6.6 we can visualize why this is

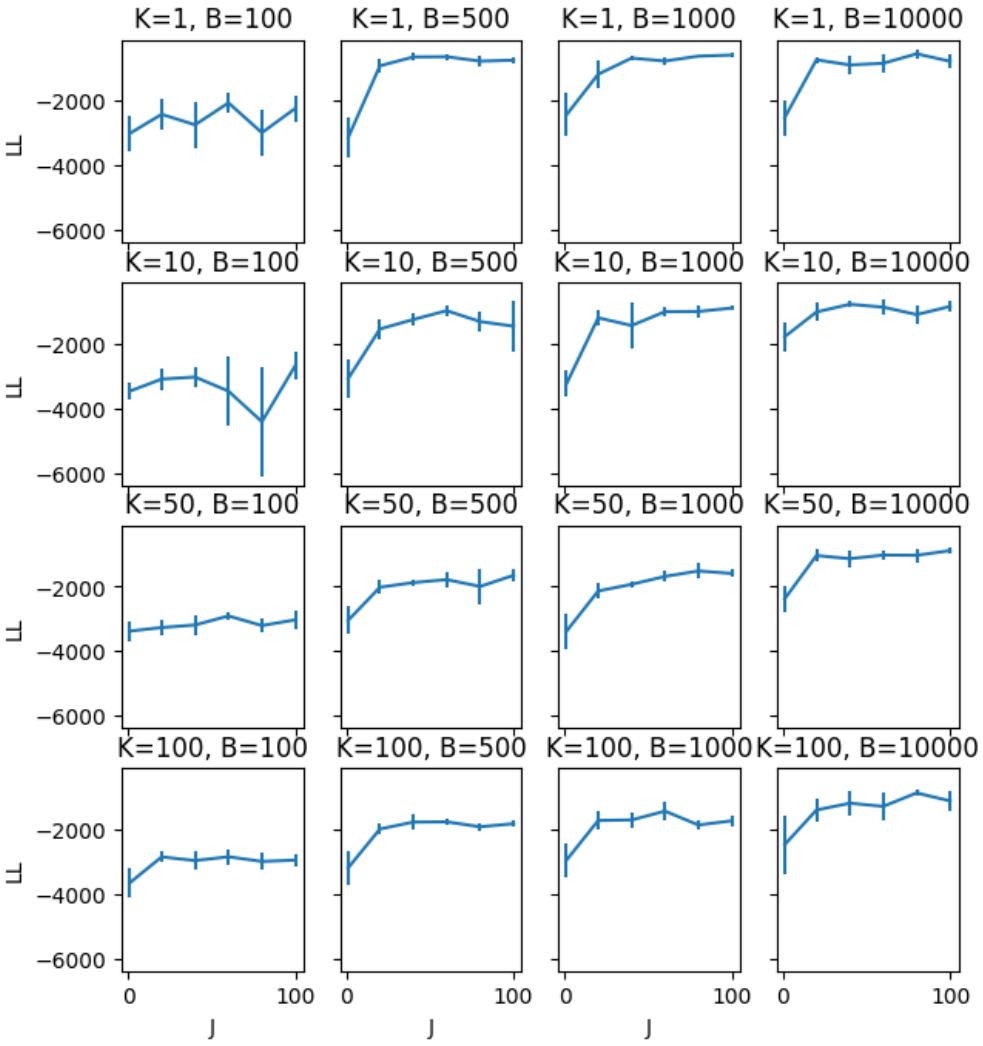

Figure 6.4: Evaluation on synthetic data of the effect of the number of samples J on the test set log likelihood of IGPS (with 95% confidence intervals), for various values of B and K.

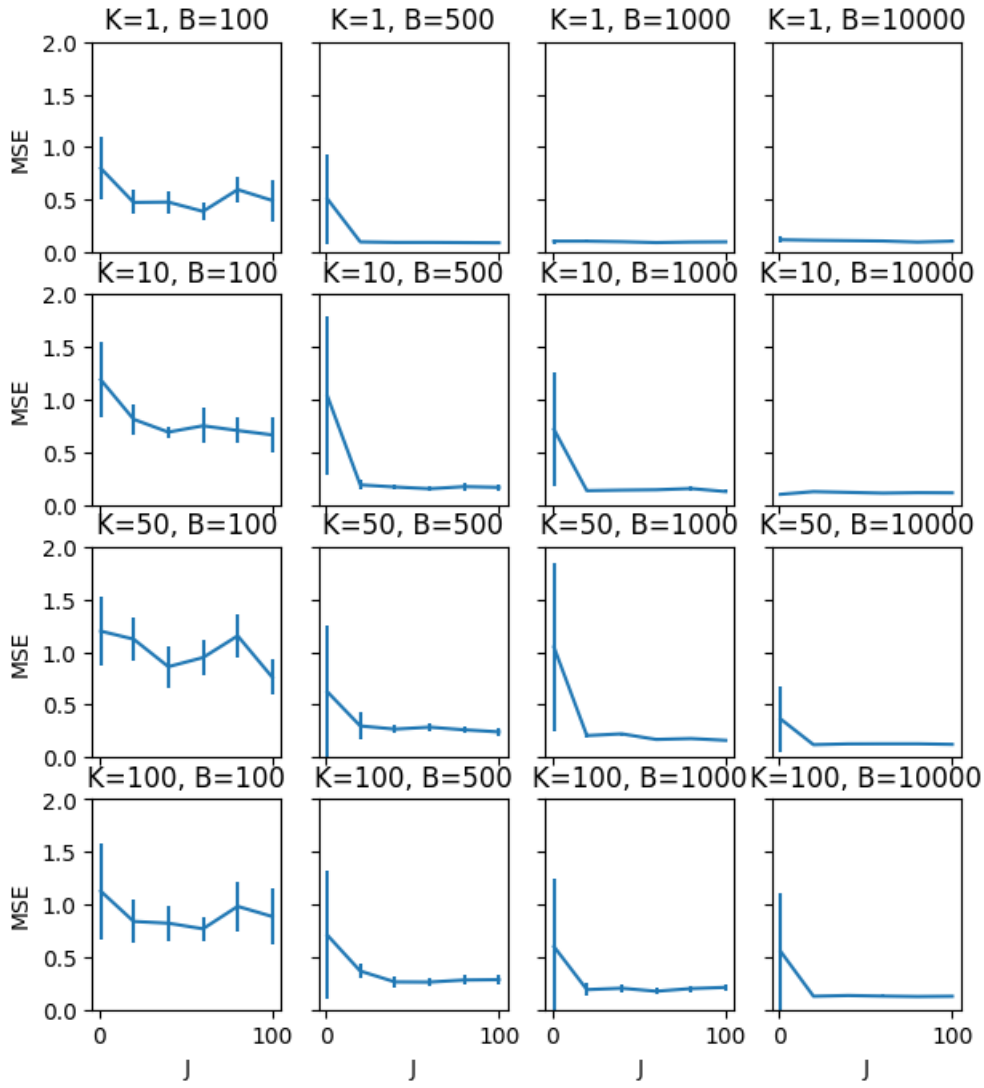

Figure 6.5: Evaluation on synthetic data of the effect of the number of samples J on the test set MSE of IGPS (with 95% confidence intervals), for various values of B and K.

the case if we compare the resulting covariance matrices in a product of expert type approach like the RBCM with a mixture of expert approach like ours. We note that BTGP also averages over partitions and, as our previous experiments show, can achieve high quality predictions as a result; however the slow mixing of the MCMC algorithm and the inability to distribute inference means we get worse performance for the same computational effort, and precludes the use of BTGP on large datasets.

The IGPS uses importance sampled weights to average over partitions; in the minibatch setting these weights, and the samples themselves, are obtained using a stochastic approximation. It is reasonable to question whether either the calculation of importance weights, or the up-weighting of the likelihood to obtain a stochastic approximation, affect the performance – would we do as well using uniform weights or avoiding the stochastic approximation? As we see in Table 6.3 (which uses the same synthetic dataset as above, with $J = 10$, $K = 10$ and $B = 1000$), using importance samples with reweighted likelihood minibatches results in better predictive performance than either not upweighting the minibatched likelihood or using uniform weights to combine predictions.

A final difference from the RBCM is the choice of the distribution over partitions that the IGPS is able to explore. The IGPS uses a distribution based on covariate location, while the RBCM generates its single partition uniformly. To evaluate the importance of this, we use the same synthetic dataset to train two variants of the IGPS: one that uses a Gaussian mixture

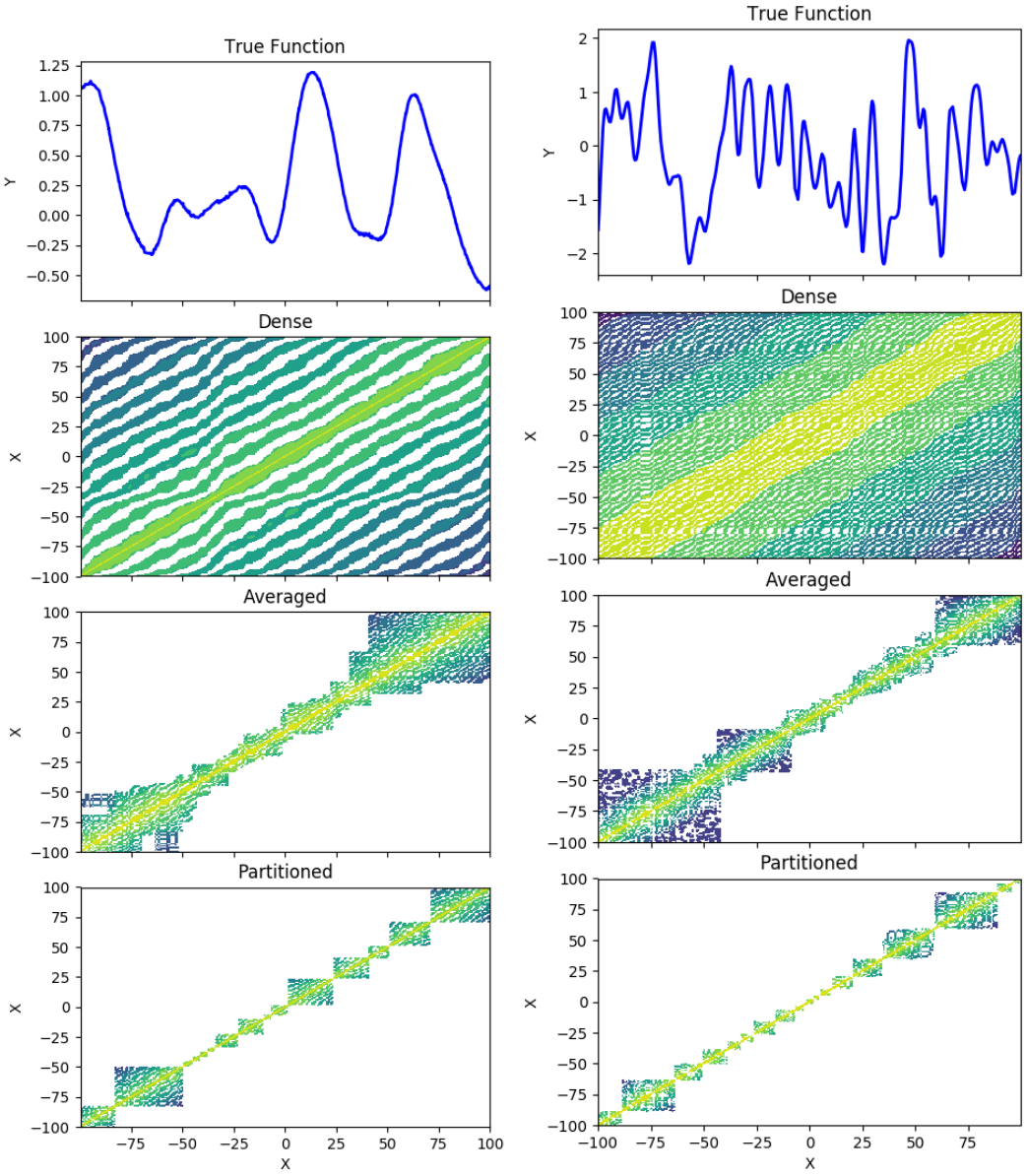

Figure 6.6: Comparison of different covariance matrices for a latent function with a long lengthscale (left) and a short lengthscale (right). "True Function" is the latent function being modeled. "Dense" is the dense covariance matrix $\Sigma(X, X')$. "Averaged" is an importance averaging of several block diagonal partitioned covariance matrices. "Partitioned" is one instance of a block diagonal covariance matrix.

Table 6.3: Test set log likelihood and MSE for various weighting schemes. Standard errors are in parentheses

Setting	LL	MSE
IS with SA	-354.96 (28.59)	0.096 (0.003)
IS without SA	-423.93 (41.24)	0.097 (0.002)
Unif. with SA	-726.67 (14.19)	0.19 (0.019)
Unif. without SA	-842.88 (9.82)	0.25 (0.337)

Table 6.4: Test set log likelihood and MSE for two different covariate partitioning schemes. Standard errors are in parentheses.

Setting	LL	MSE
GMM	-429.19 (41.57)	0.14 (0.01)
Random Clusters	-789.37 (37.43)	0.53 (0.02)

model to partition data, and one which uses random partitions. As before, we used $J = 10$, $K = 10$ and $B = 1000$. We can see in Table 6.4 that we indeed perform better when we place structure in the input clustering as opposed to purely random partitioning.

Next we wish to understand how our method behaves under different parameter settings. Again, using the same synthetic data set, we examine the behavior of our method when increasing J under different settings of K and B. Figure 6.4 shows that the optimal settings for the best predictive performance are when J and B are high while K is low, though we can still obtain good results when these settings are relaxed to something less computationally burdensome to run.

6.3.2 Evaluation on Real Data

As seen in our experiments on synthetic data, the IGPS is applicable to many different data regimes where other approximations may fail. Its inherently parallelizable nature also makes it an appealing choice for larger, real-world datasets where use of a full GP is computationally infeasible. To evaluate performance in this "big data" regime, we used an empirical dataset consisting of 209,631 mid-tropospheric CO2 measurements over space and time from the Atmospheric Infrared Sounder (AIRS)[4].

6.3.2.1 Sensitivity to Model Settings

Clearly, both the performance and the cost of inference of our model will depend on the number of blocks K in our approximation, the number of importance samples J, and the minibatch size B. On the one hand, inference scales as $O(JB^3/K^2)$, so we can speed up inference by decreasing J or B or by increasing K. On the other hand, a smaller number of blocks will allow us to better approximate a dense covariance matrix; a larger number of importance samples helps us explore the full posterior; larger minibatches reduce the noise in our estimators.

In order to pick values for K, J and B, we must understand how they affect our overall estimates. We trained the IGPS using a range of values for B, K and J, over 20 cross-validation splits. As expected, we find that as we

[4]Available in the R package `FRK` as `AIRS_05_2003`

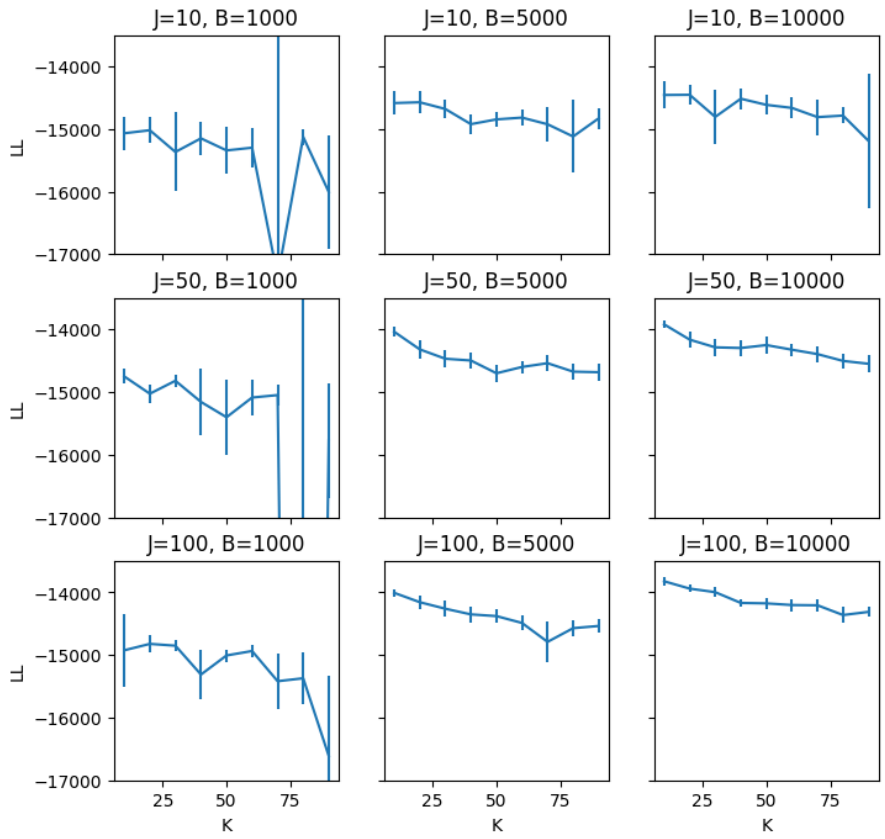

Figure 6.7: Evaluation on the AIRS dataset of the effect of B, K, and J on the test set log likelihood of IGPS (with 95% confidence intervals).

increase K or decrease J and B our performance deteriorates. Figure 6.7 shows that as the B and J increases, the average predictive log likelihood increases and the variance of the log likelihood decreases, and that as K increases the quality of our inference method degrades. However, looking at Figures 6.7 and 6.8, we see the deterioration in predictive likelihood and MSE is fairly gradual for most values: we only see a dramatic degradation when we have both a small minibatch size and a large number of partitions. This suggests that the

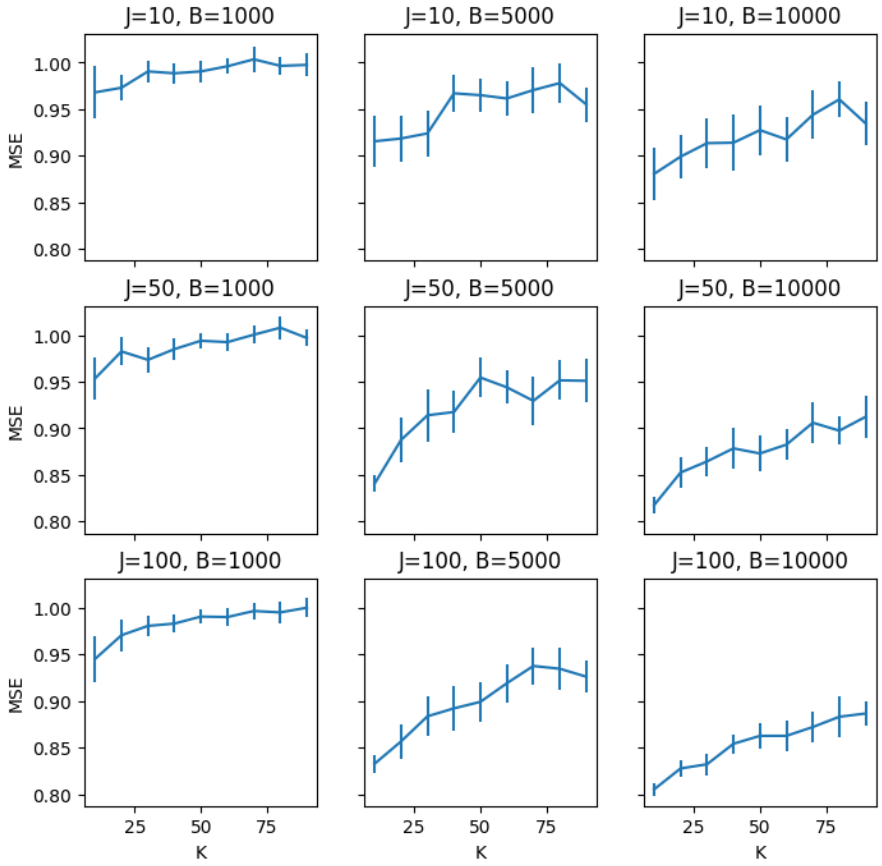

Figure 6.8: Evaluation on the AIRS dataset of the effect of B, K, and J on the test set MSE of IGPS (with 95% confidence intervals).

practitioner can modify B, J and K within a wide range to achieve acceptable computational costs without a dramatic drop in quality.

6.3.2.2 Comparison with Competing Methods

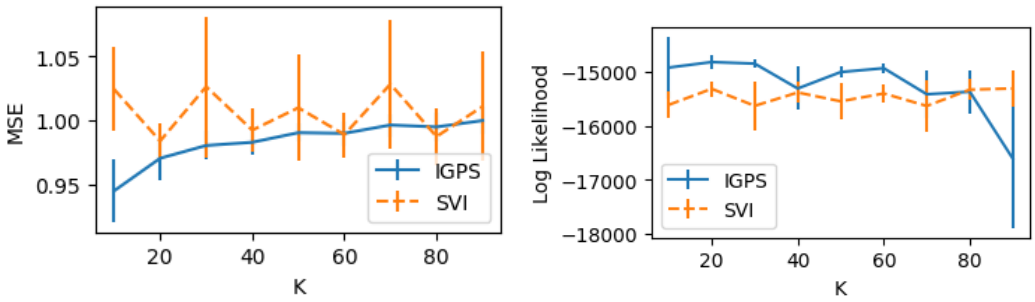

Figure 6.9: Comparison of IGPS and SVI on AIRS dataset for various K, with $J = 100$ and $B = 1000$. SVI parameters chosen to have equivalent computational cost.

Using the same CO2 dataset and squared exponential kernel as before, we compare the IGPS with SVI – the only other method that would scale to this dataset.[5] For IGPS, we set $J = 100$ and $B = 1000$ and explored a range of values of K; for SVI we chose values that gave a comparable level of computational complexity. We evaluated performance over 20 cross-validation splits. Our importance sampling method provides for a richer predictive model due to the averaging over importance proposals, and we see the benefit of this in our results. As Figure 6.9 shows, the IGPS typically performs comparably to SVI at equal levels of computational complexity in predictive performances using

[5]While RBCM is designed to scale to large data, we were unable to run the available Python package `gptf` due to memory issues.

Table 6.5: Test set log likelihood and AUC on three classification datasets.

	Log Likelihood			AUC		
Data	Full GP	IGPS	FITC	Full GP	IGPS	FITC
Pima	-128.79	-135.09	-128.61	0.83	0.81	0.83
Parkinsons	-17.00	-22.76	-28.42	0.86	0.93	0.88
WDBC	-15.50	-12.62	-18.01	0.83	0.91	0.81

both metrics until approximately 90 clusters, in which our method performs notably worse due to the long range correlation present in this dataset.

6.3.2.3 Applications Beyond Regression

Finally, to highlight that our method is not limited to a specific GP model, we apply our method on a binary classification task, using a Laplace approximation with a squared exponential kernel. We compare with the full GP [101] and the sparse GP [40] on three classification datasets from the UCI repository: the Pima Indians diabetes dataset; the Parkinsons dataset; and the Wisconsin diagnostic breast cancer (WDBC) dataset.[6] As Table 6.5 shows, the IGPS can approximate the full GP results very well, with comparable area under the curve (AUC) scores and log likelihood to a full GP and a sparse approximation.

[6]All empirical classification datasets are available in the UCI repository at `http://archive.ics.uci.edu/ml/`.

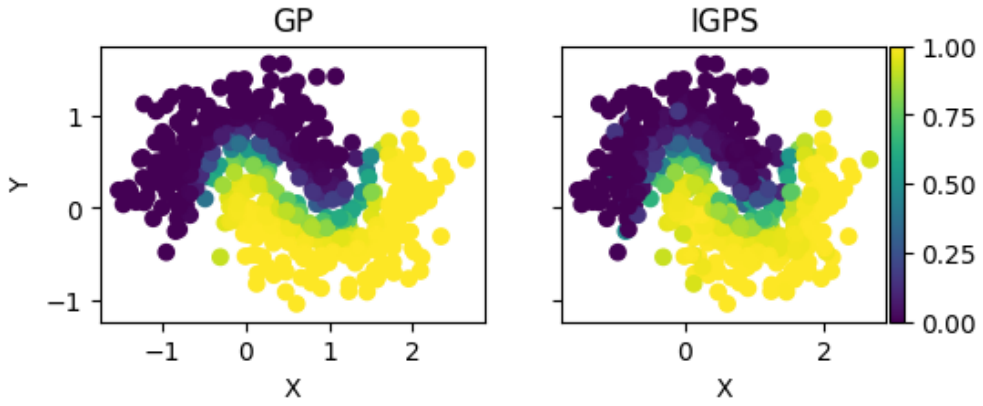

Figure 6.10: Binary classification task: label probabilities obtained using the full GP and the IGPS.

6.4 Discussion

While Gaussian processes provide a flexible framework for a wide variety of modeling scenarios, their use has been limited in the "big data" regime, since most implementations scale cubically with the number of data points. As we saw in Section 4.1, while a number of approximations have been proposed to reduce this cost, these approximations come with notable failure modes. The IGPS avoids these pitfalls, using parallelizable importance sampling to explore a mixture of block-diagonal, easily invertible matrices.

A potential avenue for future research is to explore whether we can achieve further speed-ups by using GPU-based computation [16, 35, 33]. In this paper, we have focused on regression models using a Gaussian mixture model on the covariates, but the scope of the IGPS is much broader. For example, we could use alternative distributions over partitions, or embed the

IGPS within a more complex model–particularly in deep Gaussian process models. We leave such explorations for future work.

Chapter 7

Conclusion

Bayesian non-parametrics, while attractive on an intuitive level, still faces difficult challenges especially under the regime of "big data" due to inferential problems especially for MCMC methods. In contrast, something like deep learning models are difficult to interpret; but because fitting deep learning models is possible for complicated and large datasets, we have seen immense interest in the continued research and use for such models. Hopefully, with the advent of better inference, the Bayesian non-parametric community can also enjoy the popularity and interest that the deep learning community has earned itself. This dissertation has introduced four ideas that can help promote the use of Bayesian non-parametric modeling in the "big data" scenario. As a future direction of research, Bayesian non-parametric models need to be able to handle data with more complex likelihoods than the simple parametric forms typically used in basic Bayesian models.

We have seen in Chapter 4 that, as a result of our novel method, we can learn the features of simple black and white image datasets like the MNIST or Yale face dataset where typical MCMC sampling will fail. Additionally, through the method introduced in Chapter 6 means that we can learn non-

stationary functions that, previously, we would have needed complicated kernels to learn. Now, we can instead use a composite of simpler covariance functions which may facilitate function learning when the latent function in question does not behave as nicely as a typical RBF kernel does. Over the long term, there will hopefully be sustained interest in developing such methods for learning deep Gaussian processes and fitting latent variable models to data that can only be modeled with more complicated likelihoods, like color images or video. Optimistically, the work presented in this dissertation can be extended to accomplish this goal so that people in statistics and machine learning can use the theoretically appealing properties of Bayesian non-parametrics in actual practice for more complicated problems than the settings for which current inference methods have already been developed.

Bibliography

[1] Akaike, H. (1974). A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716–723.

[2] Aldous, D. J. (1985). Exchangeability and related topics. In *École d'Été de probabilités de Saint-Flour XIII*.

[3] Angelino, E., Johnson, M. J., and Adams, R. P. (2016). Patterns of scalable Bayesian inference. *Foundations and Trends in Machine Learning*, 9(2-3):119–247.

[4] Antoniak, C. E. (1974). Mixtures of Dirichlet processes with applications to Bayesian nonparametric problems. *Ann. Statist.*, 2(6):1152–1174.

[5] Asuncion, A., Smyth, P., and Welling, M. (2008). Asynchronous distributed learning of topic models. In *NIPS*.

[6] Banerjee, S., Gelfand, A. E., Finley, A. O., and Sang, H. (2008). Gaussian predictive process models for large spatial data sets. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 70(4):825–848.

[7] Barbieri, M. M. and Berger, J. O. (2004). Optimal predictive model selection. *Annals of Statistics*, pages 870–897.

[8] Birgé, L. (1983). Approximation dans les espaces métriques et théorie de l'estimation. *Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete*, 65(2):181–237.

[9] Blackwell, D. and MacQueen, J. B. (1973). Ferguson distributions via Pólya urn schemes. *The Annals of Statistics*, pages 353–355.

[10] Blei, D. M., Jordan, M. I., et al. (2006). Variational inference for dirichlet process mixtures. *Bayesian analysis*, 1(1):121–143.

[11] Blei, D. M., Kucukelbir, A., and McAuliffe, J. D. (2017). Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877.

[12] Broderick, T., Jordan, M. I., Pitman, J., et al. (2012). Beta processes, stick-breaking and power laws. *Bayesian analysis*, 7(2):439–476.

[13] Brooks, S., Gelman, A., Jones, G., and Meng, X.-L. (2011). *Handbook of Markov chain Monte Carlo*. CRC Press.

[14] Cao, Y. and Fleet, D. J. (2014). Generalized product of experts for automatic and principled fusion of Gaussian process predictions. In *Modern Nonparametrics 3: Automating the Learning Pipeline workshop at NIPS*.

[15] Cattell, R. B. (1952). Factor analysis: an introduction and manual for the psychologist and social scientist.

[16] Dai, Z., Damianou, A., Hensman, J., and Lawrence, N. (2014). Gaussian process models with parallelization and GPU acceleration. *arXiv preprint arXiv:1410.4984*.

[17] Dalcn, L., Paz, R., and Storti, M. (2005). MPI for Python. *Journal of Parallel and Distributed Computing*, 65(9):1108 – 1115.

[18] Deisenroth, M. and Ng, J. W. (2015). Distributed Gaussian processes. In *International Conference on Machine Learning*, pages 1481–1490.

[19] Doshi, F., Miller, K., Van Gael, J., and Teh, Y. W. (2009). Variational inference for the indian buffet process. In *Artificial Intelligence and Statistics*, pages 137–144.

[20] Doshi-Velez, F., Mohamed, S., Ghahramani, Z., and Knowles, D. A. (2009). Large scale nonparametric Bayesian inference: Data parallelisation in the Indian buffet process. In *Advances in Neural Information Processing Systems*, pages 1294–1302.

[21] Efron, B., Hastie, T., Johnstone, I., Tibshirani, R., et al. (2004). Least angle regression. *The Annals of statistics*, 32(2):407–499.

[22] Ferguson, T. S. (1973). A Bayesian analysis of some nonparametric problems. *Ann. Statist.*, 1(2):209–230.

[23] Fox, E. B. (2009). *Bayesian nonparametric learning of complex dynamical phenomena*. PhD thesis, Massachusetts Institute of Technology.

[24] Gal, Y. and Ghahramani, Z. (2014). Pitfalls in the use of parallel inference for the Dirichlet process. In *Proceedings of the 31st International Conference on Machine Learning*, pages 208–216.

[25] Ge, H., Chen, Y., Wan, M., and Ghahramani, Z. (2015). Distributed inference for Dirichlet process mixture models. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 2276–2284.

[26] Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., and Rubin, D. B. (2014). *Bayesian data analysis*, volume 2. CRC Press.

[27] Gelman, A. and Shalizi, C. R. (2013). Philosophy and the practice of bayesian statistics. *British Journal of Mathematical and Statistical Psychology*, 66(1):8–38.

[28] George, E. I. and McCulloch, R. E. (1993). Variable selection via Gibbs sampling. *Journal of the American Statistical Association*, 88(423):881–889.

[29] Gershman, S. J. and Blei, D. M. (2012). A tutorial on Bayesian nonparametric models. *Journal of Mathematical Psychology*, 56(1):1–12.

[30] Ghosal, S., Ghosh, J. K., and van der Vaart, A. W. (2000). Convergence rates of posterior distributions. *The Annals of Statistics*, 28(2):500–531.

[31] Gibbs, M. (1998). *Bayesian Gaussian processes for regression and classification*. PhD thesis, University of Cambridge.

[32] Görür, D., Jäkel, F., and Rasmussen, C. E. (2006). A choice model with infinitely many latent features. In *Proceedings of the 23rd International Conference on Machine learning*, pages 361–368. ACM.

[33] Gramacy, R. B. and Apley, D. W. (2015). Local Gaussian process approximation for large computer experiments. *Journal of Computational and Graphical Statistics*, 24(2):561–578.

[34] Gramacy, R. B. and Lee, H. K. H. (2008). Bayesian treed Gaussian process models with an application to computer modeling. *Journal of the American Statistical Association*, 103(483):1119–1130.

[35] Gramacy, R. B., Niemi, J., and Weiss, R. M. (2014). Massively parallel approximate Gaussian process regression. *SIAM/ASA Journal on Uncertainty Quantification*, 2(1):564–584.

[36] Griffiths, T. and Ghahramani, Z. (2005). Infinite latent feature models and the Indian buffet process. In *Advances in Neural Information Processing Systems*.

[37] Griffiths, T. L. and Ghahramani, Z. (2011). The Indian buffet process: An introduction and review. *The Journal of Machine Learning Research*, 12:1185–1224.

[38] Hastings, W. K. (1970). Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109.

[39] Hensman, J., Fusi, N., and Lawrence, N. D. (2013). Gaussian processes for big data. In *Uncertainty in Artificial Intelligence*.

[40] Hernández-Lobato, D. and Hernández-Lobato, J. M. (2016). Scalable Gaussian process classification via expectation propagation. In *Artificial Intelligence and Statistics*, pages 168–176.

[41] Hjort, N. L. (1990). Nonparametric Bayes estimators based on beta processes in models for life history data. *The Annals of Statistics*, 18(3):1259–1294.

[42] Hoeting, J. A., Madigan, D., Raftery, A. E., and Volinsky, C. T. (1999). Bayesian model averaging: a tutorial. *Statistical science*, pages 382–401.

[43] Hoffman, M. D., Blei, D. M., Wang, C., and Paisley, J. (2013). Stochastic variational inference. *The Journal of Machine Learning Research*, 14(1):1303–1347.

[44] Hu, Y., Zhai, K., Williamson, S., and Boyd-Graber, J. (2012). Modeling images using transformed Indian buffet processes. In *International Conference of Machine Learning*.

[45] Huggins, J., Campbell, T., and Broderick, T. (2016). Coresets for scalable Bayesian logistic regression. In *Advances in Neural Information Processing Systems*, pages 4080–4088.

[46] Ishwaran, H. and James, L. F. (2001). Gibbs sampling methods for stick-breaking priors. *JASA*, 96(453):161–173.

[47] Ishwaran, H. and Rao, J. S. (2005). Spike and slab variable selection: Frequentist and Bayesian strategies. *Annals of Statistics*, pages 730–773.

[48] Ishwaran, H. and Zarepour, M. (2002). Exact and approximate sum representations for the Dirichlet process. *Canadian Journal of Statistics*, 30(2):269–283.

[49] Jordan, M. I. (2011). The era of big data. *ISBA Bulletin*, 18(2):1–3.

[50] Kalli, M., Griffin, J. E., and Walker, S. G. (2011). Slice sampling mixture models. *Statistics and computing*, 21(1):93–105.

[51] Kaufman, C. G., Schervish, M. J., and Nychka, D. W. (2008). Covariance tapering for likelihood-based estimation in large spatial data sets. *Journal of the American Statistical Association*, 103(484):1545–1555.

[52] Kim, B., Shah, J. A., and Doshi-Velez, F. (2015). Mind the gap: A generative approach to interpretable feature selection and extraction. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R., editors, *Advances in Neural Information Processing Systems 28*, pages 2251–2259. Curran Associates, Inc.

[53] Kingma, D. P. and Welling, M. (2014). Auto-encoding variational bayes. *ICLR 2014*.

[54] Kingman, J. F. C. (1967). Completely random measures. *Pacific Journal of Mathematics*, 21(1):59–78.

[55] Kong, A. (1992). A note on importance sampling using standardized weights. Technical Report 348, University of Chicago, Dept. of Statistics.

[56] Le Cam, L. (2012). *Asymptotic Methods in Statistical Decision Theory*. Springer Science & Business Media.

[57] LeCun, Y. and Cortes, C. (1998). The MNIST database of handwritten digits.

[58] Lee, K.-C., Ho, J., and Kriegman, D. J. (2005). Acquiring linear subspaces for face recognition under variable lighting. *IEEE Transactions on pattern analysis and machine intelligence*, 27(5):684–698.

[59] Ma, Y.-A., Chen, T., and Fox, E. (2015). A complete recipe for stochastic gradient MCMC. In *Advances in Neural Information Processing Systems*, pages 2917–2925.

[60] Meeds, E. and Osindero, S. (2006). An alternative infinite mixture of Gaussian process experts. In *Neural Information Processing Systems*, pages 883–890.

[61] Mescheder, L. M., Nowozin, S., and Geiger, A. (2017). Adversarial variational bayes: Unifying variational autoencoders and generative adversarial networks. *CoRR*, abs/1701.04722.

[62] Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953). Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092.

[63] Minsker, S. (2015). Geometric median and robust estimation in Banach spaces. *Bernoulli*, 21(4):2308–2335.

[64] Minsker, S., Srivastava, S., Lin, L., and Dunson, D. (2014a). Scalable and robust Bayesian inference via the median posterior. In *International Conference on Machine Learning*, pages 1656–1664.

[65] Minsker, S., Srivastava, S., Lin, L., and Dunson, D. B. (2014b). Scalable and robust Bayesian inference via the median posterior. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1656–1664.

[66] Neal, R. M. (2000). Markov chain sampling methods for Dirichlet process mixture models. *Journal of Computational and Graphical Statistics*, 9(2):249–265.

[67] Neiswanger, W., Wang, C., and Xing, E. (2013). Asymptotically Exact, Embarrassingly Parallel MCMC. *ArXiv e-prints*.

[68] Ng, J. W. and Deisenroth, M. P. (2014). Hierarchical mixture-of-experts model for large-scale Gaussian process regression. *arXiv preprint arXiv:1412.3078*.

[69] Orbanz, P. (2014). Lecture notes on Bayesian nonparametrics.

[70] Owen, A. B. (2013). *Monte Carlo theory, methods and examples*.

[71] Paisley, J. and Carin, L. (2009). Nonparametric factor analysis with beta process priors. In *Proceedings of the 26th Annual International Conference on Machine Learning*.

[72] Paisley, J. W., Zaas, A. K., Woods, C. W., Ginsburg, G. S., and Carin, L. (2010). A stick-breaking construction of the beta process. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 847–854.

[73] Park, C., Huang, J. Z., and Ding, Y. (2011). Domain decomposition approach for fast Gaussian process regression of large spatial data sets. *Journal of Machine Learning Research*, 12(May):1697–1728.

[74] Rasmussen, C. E. (2000). The infinite Gaussian mixture model. In *Advances in Neural Information Processing Systems*, pages 554–560.

[75] Rasmussen, C. E. and Ghahramani, Z. (2002). Infinite mixtures of Gaussian process experts. In *Neural Information Processing Systems*, pages 881–888.

[76] Rasmussen, C. E. and Williams, C. K. I. (2006). Gaussian processes for machine learning. *Gaussian Processes for Machine Learning*.

[77] Robert, C. P. and Casella, G. (2004). *Monte Carlo Statistical Methods*, volume 2. Springer-Verlag New York.

[78] Schwarz, G. (1978). Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464.

[79] Scott, S. L., Blocker, A. W., Bonassi, F. V., Chipman, H. A., George, E. I., and McCulloch, R. E. (2016). Bayes and big data: The consensus Monte Carlo algorithm. *International Journal of Management Science and Engineering Management*, 11(2):78–88.

[80] Seeger, M., Williams, C., and Lawrence, N. (2003). Fast forward selection to speed up sparse Gaussian process regression. In *Artificial Intelligence and Statistics*.

[81] Sethuraman, J. (1994). A constructive definition of Dirichlet priors. *Statistica Sinica*, 4:639–650.

[82] Sivasubramaniam, A., Singla, A., Ramachandran, U., and Venkateswaran, H. (1994). An approach to scalability study of shared memory parallel systems. *ACM SIGMETRICS Performance Evaluation Review*, 22(1):171–180.

[83] Smyth, P., Welling, M., and Asuncion, A. U. (2009). Asynchronous distributed learning of topic models. In *Advances in Neural Information Processing Systems*, pages 81–88.

[84] Snelson, E. and Ghahramani, Z. (2005). Sparse Gaussian processes using pseudo-inputs. In *Neural Information Processing Systems*, pages 1257–1264.

[85] Snoek, J., Larochelle, H., and Adams, R. P. (2012). Practical Bayesian optimization of machine learning algorithms. In *Neural Information Processing Systems*, pages 2951–2959.

[86] Sriperumbudur, B. K., Fukumizu, K., Gretton, A., Schölkopf, B., Lanckriet, G. R. G., et al. (2012). On the empirical estimation of integral probability metrics. *Electronic Journal of Statistics*, 6:1550–1599.

[87] Sriperumbudur, B. K., Gretton, A., Fukumizu, K., Schölkopf, B., and Lanckriet, G. R. G. (2010). Hilbert space embeddings and metrics on probability measures. *The Journal of Machine Learning Research*, 11:1517–1561.

[88] Srivastava, S., Cevher, V., Dinh, Q., and Dunson, D. (2015). WASP: Scalable Bayes via barycenters of subset posteriors. In *Artificial Intelligence and Statistics*, pages 912–920.

[89] Teh, Y. W., Görür, D., and Ghahramani, Z. (2007). Stick-breaking construction for the Indian buffet process. In *International Conference on Artificial Intelligence and Statistics*.

[90] Teh, Y. W., Jordan, M. I., Beal, M. J., and Blei, D. M. (2004). Hierarchical dirichlet processes. *Journal of the American Statistical Association*, 101.

[91] Thibaux, R. and Jordan, M. I. (2007). Hierarchical beta processes and the Indian buffet process. In *International Conference on Artificial Intelligence and Statistics*.

[92] Titsias, M. K. (2009). Variational learning of inducing variables in sparse Gaussian processes. In *Artificial Intelligence and Statistics*, volume 5, pages 567–574.

[93] Tran, D., Ranganath, R., and Blei, D. M. (2017). Deep and hierarchical implicit models. *CoRR*, abs/1702.08896.

[94] Tresp, V. (2000). A Bayesian committee machine. *Neural Computation*, 12(11):2719–2741.

[95] Wainwright, M. J., Jordan, M. I., et al. (2008). Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1–2):1–305.

[96] Walker, S. G. (2007). Sampling the Dirichlet mixture model with slices. *Communications in Statistics?Simulation and Computation®*, 36(1):45–54.

[97] Wang, J. M., Fleet, D. J., and Hertzmann, A. (2005). Gaussian process dynamical models. In *Neural Information Processing Systems*.

[98] Wang, X., Peng, P., and Dunson, D. B. (2014). Median selection subset aggregation for parallel inference. In *Advances in Neural Information Processing Systems*, pages 2195–2203.

[99] Weiszfeld, E. (1937). Sur le point pour lequel la somme des distances de n points donnes est minimum. *Tohoku Mathematical Journal*, 43:355–386.

[100] Welling, M. and Teh, Y. W. (2011). Bayesian learning via stochastic gradient Langevin dynamics. In *International Conference on Machine Learning*, pages 681–688.

[101] Williams, C. K. and Barber, D. (1998). Bayesian classification with Gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12):1342–1351.

[102] Williamson, S., Dubey, A., and Xing, E. (2013). Parallel Markov chain Monte Carlo for nonparametric mixture models. In *Proceedings of the 30th International Conference on Machine Learning*, pages 98–106.

[103] Zhou, M., Chen, H., Ren, L., Sapiro, G., Carin, L., and Paisley, J. W. (2009). Non-parametric Bayesian dictionary learning for sparse image representations. In *Advances in Neural Information Processing Systems*.