TWO-STAGE METROPOLIS HASTINGS; BAYESIAN CONDITIONAL DENSITY ESTIMATION & SURVIVAL ANALYSIS VIA PARTITION MODELING, LAPLACE APPROXIMATIONS, AND EFFICIENT COMPUTATION

A Dissertation

by

RICHARD DANIEL PAYNE

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

May 2018

Major Subject: Statistics

ABSTRACT

Bayesian statistical methods are known for their flexibility in modeling. This flexibility is possible because parameters can often be estimated via Markov chain Monte Carlo methods. In large datasets or models with many parameters, Markov chain Monte Carlo methods are insufficient and inefficient. We introduce the two-stage Metropolis-Hastings algorithm which modifies the proposal distribution of the Metropolis-Hastings algorithm via a screening stage to reduce the computational cost. The screening stage requires a cheap estimate of the log-likelihood and speeds up computation even in complex models such as Bayesian multivariate adaptive regression splines. Next, a partition model, constructed from a Voronoi tessellation, is proposed for conditional density estimation using logistic Gaussian processes. A Laplace approximation is used to approximate the marginal likelihood providing a tractable Markov chain Monte Carlo algorithm. In simulations and an application to windmill power output, the model successfully provides interpretation and flexibly models the densities. Last, a Bayesian tree partition model is proposed to model the hazard function of survival & reliability models. The piecewise-constant hazard function in each partition element is modeled via a latent Gaussian process. The marginal likelihood is estimated using Laplace approximations to yield a tractable reversible jump Markov chain Monte Carlo algorithm. The method is successful in simulations and provides insight into lung cancer survival rates in relation to protein expression levels.

DEDICATION

To my wife, Estelle, whose support and encouragement made this dissertation possible.

To my son, Paxton, for his cheerful daily office visits.

ACKNOWLEDGMENTS

I would like to especially thank Dr. Mallick for his inspiration, encouragement, and trust. During the most difficult portions of this research, he remained positive, practical, and was always supportive. Often, he was more optimistic about my research than I was, which gave me the motivation to continue. I have learned much from his broad style of research and am grateful to have been exposed to such a broad range of topics under his tutelage.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a dissertation committee consisting of Professors Bani K. Mallick, Anirban Bhattacharya and Jianhua Huang of the Department of Statistics, and Professor Yu Ding of the Department of Industrial & Systems Engineering at Texas A&M University.

The Freddie Mac data in Chapter 2 was obtained from Freddie Mac's website and the bank dataset in Chapter 2 was obtained from the University of California Irvine machine learning repository. The windmill data analyzed in Chapter 3 was provided by Professor Yu Ding. The lung cancer data in Section 4 was provided by Dr. Arnab Maity, a post-doctoral researcher at Texas A&M, and was originally obtained from the Broad Institute TCGA Genome Data Analysis Center. Dr. Maity also identified the proteins which were used in the lung cancer analysis in Chapter 4. All other work conducted for the dissertation was completed by the student independently.

Funding Sources

Graduate study was supported by the Graduate Merit Fellowship from Texas A&M University and a Technology Teaching Assistantship from the Department of Statistics at Texas A&M University.

NOMENCLATURE

MCMC	Markov chain Monte Carlo
M	number of partition elements
MH	Metropolis-Hastings
n	number of observations
n_i	number of observations in partition element i
$p(\cdot \mid \cdot)$	conditional distribution
$p(\cdot)$	probability distribution
RMSE	root mean square error
T	partition structure; either a tessellation or a tree
$\mathbf{x}$	covariate vector
$\mathbf{x}_i$	covariate vector in partition element i
$\mathbf{y}$	vector of observed response variable
$\mathbf{y}_i$	vector of observed response variable in partition element i

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1 Statistics: A Melting Pot

The field of statistics is both vast and deep. It provides methodology for solving many problems in a wide variety of fields, and has become an integral part of the scientific process. There are even two separate philosophical mindsets (Bayesian and frequentist) which define how these statistics are developed! This unique combination of breadth and depth make the field of statistics a true melting pot of methodologies, theories, and applications.

In that same spirit, this dissertation is a melting pint – a smaller amalgamation of new developments in statistical methodology which vary in application and purpose. This dissertation has applications in finance, green energy, and health as new methodologies are developed in statistical computation, conditional density estimation, and survival analysis. The methods are tied together through their common Bayesian perspective and emphasis on efficient computation.

This dissertation contains an introduction, three main chapters, and a conclusion. This section provides a brief overview of Bayesian statistics and sets the stage for the three main chapters. The second chapter discusses the two-stage Metropolis-Hastings (MH) algorithm which is designed to speed up Markov chain Monte Carlo (MCMC) for large datasets. The third chapter introduces a partition model to perform conditional density estimation. The fourth chapter extends this partition model framework to model hazard functions in survival analysis. The fifth chapter gives some concluding comments. Owing to the fact that each of the three methodologies presented require very different literature reviews, a detailed literature review is contained within each of the three main chapters rather than in the first chapter.

1.2 Bayesian Statistics

In the field of statistics, we are typically interested in making inference about parameters, θ, which describe a population. In Bayesian methodologies, we model these parameters using a prior distribution, $p(\theta)$, which represents our belief about where the parameters lie prior to observing data, $\mathbf{y}$. We then utilize conditional probability to update our belief about θ based on the observed data $\mathbf{y}$. This updated understanding of θ is expressed in the posterior distribution

$$p(\theta \mid \mathbf{y}) = \frac{p(\mathbf{y} \mid \theta)p(\theta)}{p(\mathbf{y})} \tag{1.1}$$

which consists of the likelihood of the data, $p(\mathbf{y} \mid \theta)$, the prior, $p(\theta)$, and the marginal likelihood of the data $p(\mathbf{y}) = \int p(\mathbf{y} \mid \theta)p(\theta)d\theta$. Depending on the forms of the likelihood and the prior, an analytical solution for the posterior distribution $p(\theta \mid \mathbf{y})$ may exist. However, in more complicated models, including hierarchical models, it is often the case that there is no closed form solution for $p(\theta \mid \mathbf{y})$, usually due to the fact that the integral in the denominator of (1.1) is not tractable.

Fortunately there are algorithms which allow us to sample from the posterior distribution even when we do not have an analytical form. If enough samples are generated, we are able to obtain an approximation to the posterior via these samples. One common algorithm for sampling from the posterior distribution is the Metropolis-Hastings algorithm (Metropolis et al., 1953; Hastings, 1970) which largely came to the attention of Bayesian statisticians through the seminal paper of Gelfand and Smith (1990). The MH algorithm involves calculating ratios of posterior distributions with different parameters settings; therefore, the value of $p(\mathbf{y})$ is canceled out and does not need to be known.

1.2.1 Limitations of the Metropolis-Hastings Algorithm

There are, however, several limitations of the MH algorithm. When the dimension of θ is large, the computational cost of the algorithm is increased. Furthermore, the algorithm may not mix satisfactorily and therefore posterior inference may be limited. If the dataset is large, the computational cost of the algorithm can become too large to complete in a reasonable amount of time.

To overcome some of these issues, several general methods have been developed. One technique is to divide the data, run independent MCMC on each data subset, and then recombine the posterior draws in some way (Scott et al., 2016). Other techniques include parallelizing the likelihood and subsampling techniques (Quiroz et al., 2014). Each of these techniques have their limitations. For instance, the technique described by Scott et al. (2016) does not sample from the true posterior distribution and has extra computational costs when dealing even with simple hierarchical models. In our experience, subsampling methods can be computationally complex due to the need to interpolate surfaces using either splines or Gaussian processes which adds an extra layer of modeling.

At present, there is a need to reduce computational demand in the MH sampler while maintaining its relative simplicity. Chapter 2 accomplishes this by introducing a two-stage MH algorithm which is simple to implement while still retaining important theoretical properties.

1.2.2 Bayesian Partition Models

Partition models model data by partitioning the data and fitting separate models for each partition element. Consequently, these models can have many parameters due to both the number of partitions, and the complexity of the model in each data region. On the surface, partition models may appear to be difficult to search via MCMC due to the large number of parameters. However, they can often be searched efficiently by choosing priors such that the marginal likelihood, given a specific partition (denoted T), $p(\mathbf{y} \mid T)$ has an analytical form. Consequently, only the parameters defining the partition need to be searched via MCMC methods, typically using a reversible jump MCMC algorithm (Green, 1995) to traverse the variable dimension of the parameter space. These types of models have been used in a number of contexts including piecewise Gaussian processes (Kim et al., 2005), classification and regression trees (Chipman et al., 1998), treed Gaussian processes (Gramacy and Lee, 2008), and prediction using Voronoi tessellations (Holmes et al., 2005).

Partition models are appealing because they have a relatively simple form and can be very flexible, particularly when using model averaging. They are "nonparametric" in the sense that they are extremely flexible despite assuming a parametric model within each partition element. Although the curse of dimensionality may be mitigated by marginalizing partition-element specific parameters, care must be taken so that the reversible jump MCMC algorithm mixes across models effectively. Parallel tempering (Geyer, 1991) has been used in the commonly multi-modal tree models by Gramacy (2007) and Gramacy and Taddy (2010) which generally eliminates the need to restart the MCMC chain to allow it adequately explore various posterior modes (Chipman et al., 1998).

To date, the majority (if not all) partition models choose distributions in each partition element which provide an analytical solution for computing the marginal distribution of the data. This is often chosen for computational convenience so that an efficient MCMC algorithm can be constructed. Unfortunately, by specifying such models, flexibility is often lost. In this dissertation, we select a models which retain great flexibility in each partition element by specifying models which do not have an analytical form for the marginal of the data. To maintain a tractable MCMC algorithm, we approximate the marginal of the data by utilizing Laplace approximations, allowing us to have both simple interpretation and model flexibility.

1.3 Organization

Chapter 2 provides a solution to the difficulties in the MH algorithm by using an approximation of the likelihood to screen out "bad" proposals (i.e. proposals which are likely to be rejected) and save on computational cost. It focuses on cases where the dimension of θ is relatively small, but the number of observations is large, and develops methodology for reducing the computational cost of the MH algorithm using a screening stage to modify the proposal distribution of the MH algorithm. In Chapter 3 and Chapter 4 we shall encounter partition models with many parameters which cannot be marginalized. Without marginalization, reversible jump MCMC becomes extremely slow since parameters in each partition element need to be sampled on each iteration of the MCMC chain. Furthermore, reversible jump MCMC requires efficient proposal distributions when changing the partition structure, which are difficult to construct. To overcome these issues, the third and fourth chapters utilize Laplace approximations (Tierney and Kadane, 1986) to obtain an approximate marginal likelihood so that an efficient MCMC search can be performed over the partition space. Chapter 3 focuses on performing conditional density estimation in the partition model framework using Voronoi tessellations whereas Chapter 4 uses a tree partition model to perform survival analysis. Chapter 5 contains some concluding remarks and discussion.

1.4 Notation

While care has been taken to keep notation similar across all three chapters, the differences in the model formulations across the three chapters require some re-use of some symbols. Therefore, please refer to each individual chapter for each symbol's interpretation.

2. TWO-STAGE METROPOLIS-HASTINGS FOR TALL DATA

2.1 Introduction & Literature Review

In the past twenty-five years, Bayesian statistics have become increasingly popular as they are capable of analyzing data with complex structures. Consequently, Bayesian methods have been proven to be effective in a wide range of applications. The rise in popularity is largely attributed to simulation based algorithms which can approximate the complex posterior distributions of non-conjugate models, such as Markov chain Monte Carlo (MCMC) methods including the Metropolis-Hastings (MH) algorithm (Robert and Casella, 2013).

The term "tall data" generally describes data in which $n >> p$, that is, when the number of observations is much larger than the number of predictors. For MCMC methods, as n increases, so does the computational demand of the algorithm. Specifically, for MH, the increased computational demand is driven by the complete scan of the data through likelihood evaluations on each iteration of the algorithm. If n is large enough, MCMC methods (including MH) are computationally infeasible.

There are several general methods to overcome this issue. The simplest method involves parallelizing the likelihood to speed up computation. Another method divides the data across multiple machines and performs independent parallel MCMC on each machine to sample from the posterior distribution (consensus Monte Carlo). The results are then aggregated using weighting (Scott et al., 2016). A third approach is to use subsampling methods to provide a faster estimation of the likelihood (Quiroz et al., 2014; Korattikara et al., 2014; Bardenet et al., 2014). See Bardenet et al. (2015) for a review of MCMC approaches for tall data.

We propose a method based on a two-stage Metropolis algorithm which uses a cheap estimate of the likelihood to determine if a full estimation of the likelihood is necessary. Furthermore, we compare the strengths and weaknesses of the general tall data MH methods of consensus, sub-

*Reprinted in modified form with permission from "Two-Stage Metropolis-Hastings for Tall Data" by Payne, Richard D., and Mallick, Bani K. (2018). *Journal of Classification, 35(1).* Copyright 2018 Classification Society of North America.

sampling, and two-stage Metropolis, as well as briefly introduce the use of a combination of the consensus and two-stage methods. For definiteness, in the following, the focus of this chapter is on the classification problem. However, the developed methodology can be extended to any model which is suitable to analyze tall data. The methods are applied to three datasets: marketing data from a Portuguese bank, loan data from Freddie Mac, and a simulated dataset. Logistic regression is applied to both the Portuguese bank and Freddie Mac datasets and an additional logistic hierarchical model is fit to the Freddie Mac dataset. Modifications to the techniques described in the papers above have been made to accommodate the features of these datasets and are explained further. Lastly, the two-stage method is applied to a simulated binary classification problem using Bayesian multivariate adaptive regression splines (BMARS).

In Section 2.2, we describe the existing methods for handling tall data and present the two-stage methodology. Section 2.3 applies the methods to the marketing, Freddie Mac, and simulated datasets. Section 2.4 provides a brief discussion and Section 2.5 concludes.

2.2 Methodology

We begin by briefly describing existing techniques to speed up MCMC computation for tall data applications.

2.2.1 Likelihood Parallelization

Perhaps the simplest way to adapt the Metropolis-Hastings algorithm for tall data is to compute the likelihood in parallel. In this method, the data are partitioned into M partitions and each is assigned to a separate process/core. On each iteration of the MH algorithm, the master process draws parameters from the proposal distribution and sends the proposed values to the other processes. Each process then computes the likelihood for its partition and passes this information (i.e. the sum of the log-likelihood) to the master process which sums the log-likelihood contributions from each partition and determines whether or not to accept the proposal. As long as there is no significant communication overhead between the processes, the MH algorithm's speed will be increased while still sampling from the true posterior distribution.

2.2.2 Consensus Monte Carlo

In the consensus Monte Carlo method the data are randomly partitioned into M partitions. Subsequently, allow each partition to run a full MCMC simulation from a posterior distribution given its own data. Lastly, combine the posterior simulations from each partition to produce a set of global draws to reproduce the unified posterior distribution.

Suppose $\mathbf{y} = (y_1, \dots, y_n)$ denotes the full data and $\mathbf{y}_i$ is the data at the ith partition. We then represent the posterior distribution of $\boldsymbol{\beta}$ as

$$p(\boldsymbol{\beta}|\mathbf{y}) \propto \prod_{i=1}^{M} p(\mathbf{y}_i|\boldsymbol{\beta}) p(\boldsymbol{\beta})^{1/M}$$

where the prior distribution has been expressed as the product of the M components.

For each partition, a Metropolis sampler with a chain of length m is computed in parallel with the prior weight adjusted to M^{-1} its original weight. Once posterior samples are obtained from each of the partitions, the results are combined using a weighted average. The weight, W_i, for the ith partition is equal to the inverse of the posterior covariance matrix obtained from the Metropolis sampler. Let $\boldsymbol{\beta}_i$ be the posterior sample matrix from the ith partition. Thus, the final posterior sample, $\boldsymbol{\beta}$ is obtained using the following weighted average:

$$\boldsymbol{\beta} = \left(\sum_{i=1}^{M} \boldsymbol{\beta}_i W_i\right) \left(\sum_{i=1}^{M} W_i\right)^{-1}$$

For details see Scott et al. (2016).

2.2.3 Subsampling Based Methods

In subsampling methods, a small subset of the data is used to estimate the likelihood function which is then used to evaluate the acceptance probabilities of the MH algorithm. In principle, subsampling reduces the data size and therefore a faster MCMC algorithm can be developed. Using an unbiased likelihood estimate in the MCMC chain still provides the correct stationary distribution (Andrieu and Roberts, 2009), however the efficiency of the MCMC chain depends on the variance of the estimator. Usually complete random sampling does not work well in this situation (i.e. the chain gets stuck for many iterations), but some general guidelines for estimating the full likelihood from a subsample in a Bayesian setting have been developed. Quiroz et al. (2014) suggest using a portion of the data prior to MCMC and fitting Gaussian processes or splines to approximate the log-likelihood. On each iteration of the MCMC chain, the log-likelihood is estimated for each observation. The data are then sampled with probability proportional to its estimated log-likelihood value, which reduces the variance of the estimator and improves the efficiency of the chain.

2.2.4 Two-Stage Metropolis-Hastings

Consider the usual Bayesian model setup where the posterior distribution of the parameter $\boldsymbol{\beta}$ given data $\mathbf{y}$ is given by

$$p(\boldsymbol{\beta}|\mathbf{y}) \propto p(\mathbf{y}|\boldsymbol{\beta})p(\boldsymbol{\beta}) \tag{2.1}$$

where $p(\mathbf{y}|\boldsymbol{\beta})$ is the likelihood function and $p(\boldsymbol{\beta})$ is the prior distribution for the parameter vector $\boldsymbol{\beta}$. If a non-conjugate prior is selected, the posterior distribution $p(\boldsymbol{\beta}|\mathbf{y})$ often cannot be expressed in an explicit form and consequently MCMC methods must be used to simulate samples from this posterior distribution. More specifically, we use the Metropolis-Hastings (MH) algorithm to generate samples of $\boldsymbol{\beta}$s from $p(\boldsymbol{\beta}|\mathbf{y})$. The MH algorithm is described as follows.

A.1 MH Algorithm

1. At the tth iteration generate $\boldsymbol{\beta}$ from the proposal distribution $q(\boldsymbol{\beta}|\boldsymbol{\beta}_t)$ where $\boldsymbol{\beta}_t$ is the current state

2. Accept $\boldsymbol{\beta}$ as a posterior sample with probability

$$h(\boldsymbol{\beta}_t, \boldsymbol{\beta}) = \min\left\{1, \frac{q(\boldsymbol{\beta}_t|\boldsymbol{\beta})p(\boldsymbol{\beta}|\mathbf{y})}{q(\boldsymbol{\beta}|\boldsymbol{\beta}_t)p(\boldsymbol{\beta}_t|\mathbf{y})}\right\} \tag{2.2}$$

3. $\boldsymbol{\beta}_{t+1} = \boldsymbol{\beta}$ with probability $h(\boldsymbol{\beta}_t, \boldsymbol{\beta})$ and $\boldsymbol{\beta}_{t+1} = \boldsymbol{\beta}_t$ with probability $1 - h(\boldsymbol{\beta}_t, \boldsymbol{\beta})$.

At each iteration, the probability of moving from the state $\boldsymbol{\beta}_t$ to next state $\boldsymbol{\beta}$ is $q(\boldsymbol{\beta}|\boldsymbol{\beta}_t)h(\boldsymbol{\beta}_t, \boldsymbol{\beta})$, hence the transition kernel for the Markov Chain $\boldsymbol{\beta}_t$ is

$$T(\boldsymbol{\beta}_t, \boldsymbol{\beta}) = q(\boldsymbol{\beta}|\boldsymbol{\beta}_t)h(\boldsymbol{\beta}_t, \boldsymbol{\beta}) + \left\{1 - \int q(\boldsymbol{\beta}|\boldsymbol{\beta}_t)h(\boldsymbol{\beta}_t, \boldsymbol{\beta})d\boldsymbol{\beta}\right\} I(\boldsymbol{\beta} = \boldsymbol{\beta}_t)$$

where $I()$ is the indicator function. Due to the iterative nature of the algorithm, the likelihood function $p(\mathbf{y}|\boldsymbol{\beta})$ needs to be evaluated repeatedly which is expensive when n is large. Hence, we need to modify the MH algorithm to adapt it for tall data problems.

In the MH algorithm described in A.1, the evaluation of the likelihood is expensive in the tall data situation. Generally the MCMC chain requires thousands of iterations to converge. Furthermore, we need to generate a large number of samples to quantify the uncertainty in the parameters. We use the two-stage MH algorithm where the proposal distribution $q()$ is adapted to the target distribution using an approximate likelihood based model. These algorithms have been used previously (Christen and Fox, 2005; Higdon et al., 2002; Mondal et al., 2014), usually for solving expensive inverse problems. For our purposes, instead of testing each proposal by the exact likelihood based model directly, initially the algorithm tests the proposal by the approximate likelihood based model which is much cheaper to compute. If the proposal is accepted by the initial test, then an exact likelihood based computation will be conducted and the proposal will be further tested as in the MH algorithm method described in A.1. Otherwise, the proposal will be rejected by the approximate model and a new proposal will be generated from $q()$. The approximate likelihood based model filters the unacceptable proposals and avoids the expensive full likelihood computations.

A.2 Two-Stage MH Algorithm

Let $\hat{p}(\mathbf{y}|\boldsymbol{\beta})$ be an approximation of the full likelihood, and let the approximate posterior distribution be represented as $p^*(\boldsymbol{\beta}|\mathbf{y}) \propto \hat{p}(\mathbf{y}|\boldsymbol{\beta})p(\boldsymbol{\beta})$. Then the Two-Stage MH Algorithm proceeds as follows:

1. At the tth iteration generate $\boldsymbol{\beta}^{'}$ from the proposal distribution $q(\boldsymbol{\beta}^{'}|\boldsymbol{\beta}_t)$

2. Take a real proposal as

$$\boldsymbol{\beta} = \begin{cases} \boldsymbol{\beta}^{'} & \text{with probability } \delta(\boldsymbol{\beta}_t, \boldsymbol{\beta}^{'}) \\ \boldsymbol{\beta}_t & \text{with probability } 1 - \delta(\boldsymbol{\beta}_t, \boldsymbol{\beta}^{'}) \end{cases}$$

   where

$$\delta(\boldsymbol{\beta}_t, \boldsymbol{\beta}^{'}) = \min\left\{1, \frac{q(\boldsymbol{\beta}_t|\boldsymbol{\beta}^{'})p^*(\boldsymbol{\beta}^{'}|\mathbf{y})}{q(\boldsymbol{\beta}^{'}|\boldsymbol{\beta}_t)p^*(\boldsymbol{\beta}_t|\mathbf{y})}\right\}$$

3. Accept $\boldsymbol{\beta}$ as a posterior sample with probability

$$\rho(\boldsymbol{\beta}_t, \boldsymbol{\beta}) = \min\left\{1, \frac{Q(\boldsymbol{\beta}_t|\boldsymbol{\beta})p(\boldsymbol{\beta}|\mathbf{y})}{Q(\boldsymbol{\beta}|\boldsymbol{\beta}_t)p(\boldsymbol{\beta}_t|\mathbf{y})}\right\} \tag{2.3}$$

   where $Q(\boldsymbol{\beta}|\boldsymbol{\beta}_t) = \delta(\boldsymbol{\beta}_t, \boldsymbol{\beta})q(\boldsymbol{\beta}|\boldsymbol{\beta}_t) + \{1 - \int \delta(\boldsymbol{\beta}_t, \boldsymbol{\beta})q(\boldsymbol{\beta}|\boldsymbol{\beta}_t)d\boldsymbol{\beta}\}I(\boldsymbol{\beta} = \boldsymbol{\beta}_t)$

4. Hence take $\boldsymbol{\beta}_{t+1} = \boldsymbol{\beta}$ with probability $\rho(\boldsymbol{\beta}_t, \boldsymbol{\beta})$ and $\boldsymbol{\beta}_{t+1} = \boldsymbol{\beta}_t$ with probability $1-\rho(\boldsymbol{\beta}_t, \boldsymbol{\beta})$.

At each iteration, the probability of moving from the state $\boldsymbol{\beta}_t$ to next state $\boldsymbol{\beta}$ is $q(\boldsymbol{\beta}|\boldsymbol{\beta}_t)\rho(\boldsymbol{\beta}_t, \boldsymbol{\beta})$, hence the transition kernel for the Markov Chain $\boldsymbol{\beta}_t$ is

$$T(\boldsymbol{\beta}_t, \boldsymbol{\beta}) = q(\boldsymbol{\beta}|\boldsymbol{\beta}_t)\rho(\boldsymbol{\beta}_t, \boldsymbol{\beta}) + \left\{1 - \int q(\boldsymbol{\beta}|\boldsymbol{\beta}_t)\rho(\boldsymbol{\beta}_t, \boldsymbol{\beta})d\boldsymbol{\beta}\right\} I(\boldsymbol{\beta} = \boldsymbol{\beta}_t).$$

In the above algorithm, if the trial proposal $\boldsymbol{\beta}^{'}$ is rejected by the approximate posterior then no further computation is needed. Thus, the expensive exact posterior computation can be avoided for those proposals which are unlikely to be accepted. This is just an adaption of the proposal using the approximate posterior where the transition kernel can be written as $K(\boldsymbol{\beta}_t, \boldsymbol{\beta}) = \rho(\boldsymbol{\beta}_t, \boldsymbol{\beta})Q(\boldsymbol{\beta}|\boldsymbol{\beta}_t)$ for $\boldsymbol{\beta} \neq \boldsymbol{\beta}_t$ and $K(\boldsymbol{\beta}_t, \{\boldsymbol{\beta}_t\}) = 1 - \int_{\boldsymbol{\beta} \neq \beta_t} \rho(\boldsymbol{\beta}_t, \boldsymbol{\beta})Q(\boldsymbol{\beta}_t|\boldsymbol{\beta})d\boldsymbol{\beta}$ for $\boldsymbol{\beta} = \boldsymbol{\beta}_t$. It is simple to show that the detailed balance condition $p(\boldsymbol{\beta}_t|\mathbf{y})K(\boldsymbol{\beta}_t, \boldsymbol{\beta}) = p(\boldsymbol{\beta}|\mathbf{y})K(\boldsymbol{\beta}, \boldsymbol{\beta}_t)$ is always satisfied under some minor regularity conditions like the regular MH algorithm.

Result 1: The detailed balance condition is satisfied under the regularity conditions of the MH algorithm. That is, $p(\boldsymbol{\beta}_t|\mathbf{y})K(\boldsymbol{\beta}_t, \boldsymbol{\beta}) = p(\boldsymbol{\beta}|\mathbf{y})K(\boldsymbol{\beta}, \boldsymbol{\beta}_t)$.

Proof: When $\boldsymbol{\beta} = \boldsymbol{\beta}_t$, the result is trivial. When $\boldsymbol{\beta} \neq \boldsymbol{\beta}_t$ we have

$$\begin{aligned} p(\boldsymbol{\beta}_t|\mathbf{y})K(\boldsymbol{\beta}_t, \boldsymbol{\beta}) &= p(\boldsymbol{\beta}_t|\mathbf{y})\rho(\boldsymbol{\beta}_t, \boldsymbol{\beta})Q(\boldsymbol{\beta}|\boldsymbol{\beta}_t) \\ &= p(\boldsymbol{\beta}_t|\mathbf{y}) \min\left\{1, \frac{Q(\boldsymbol{\beta}_t|\boldsymbol{\beta})p(\boldsymbol{\beta}|\mathbf{y})}{Q(\boldsymbol{\beta}|\boldsymbol{\beta}_t)p(\boldsymbol{\beta}_t|\mathbf{y})}\right\} Q(\boldsymbol{\beta}|\boldsymbol{\beta}_t) \\ &= \min\left\{p(\boldsymbol{\beta}_t|\mathbf{y})Q(\boldsymbol{\beta}|\boldsymbol{\beta}_t), Q(\boldsymbol{\beta}_t|\boldsymbol{\beta})p(\boldsymbol{\beta}|\mathbf{y})\right\} \\ &= \min\left\{\frac{p(\boldsymbol{\beta}_t|\mathbf{y})Q(\boldsymbol{\beta}|\boldsymbol{\beta}_t)}{p(\boldsymbol{\beta}|\mathbf{y})Q(\boldsymbol{\beta}_t|\boldsymbol{\beta})}, 1\right\} p(\boldsymbol{\beta}|\mathbf{y})Q(\boldsymbol{\beta}_t|\boldsymbol{\beta}) \\ &= \rho(\boldsymbol{\beta}, \boldsymbol{\beta}_t)p(\boldsymbol{\beta}|\mathbf{y})Q(\boldsymbol{\beta}_t|\boldsymbol{\beta}) \\ &= p(\boldsymbol{\beta}|\mathbf{y})K(\boldsymbol{\beta}, \boldsymbol{\beta}_t) \end{aligned}$$

Result 2: The acceptance probability can be expressed as

$$\rho(\boldsymbol{\beta}_t, \boldsymbol{\beta}) = \min\left\{1, \frac{p^*(\boldsymbol{\beta}_t|\mathbf{y})p(\boldsymbol{\beta}|\mathbf{y})}{p^*(\boldsymbol{\beta}|\mathbf{y})p(\boldsymbol{\beta}_t|\mathbf{y})}\right\}$$

Proof: If $\boldsymbol{\beta} = \boldsymbol{\beta}_t$ then the result is trivial since $\rho(\boldsymbol{\beta}_t, \boldsymbol{\beta}) = 1$. For $\boldsymbol{\beta} \neq \boldsymbol{\beta}_t$

$$\begin{aligned} Q(\boldsymbol{\beta}_t|\boldsymbol{\beta}) &= \delta(\boldsymbol{\beta}, \boldsymbol{\beta}_t)q(\boldsymbol{\beta}_t|\boldsymbol{\beta}) \\ &= \frac{1}{p^*(\boldsymbol{\beta}|\mathbf{y})}\min\{q(\boldsymbol{\beta}_t|\boldsymbol{\beta})p^*(\boldsymbol{\beta}|\mathbf{y}), q(\boldsymbol{\beta}|\boldsymbol{\beta}_t)p^*(\boldsymbol{\beta}_t|\mathbf{y})\} \\ &= \frac{q(\boldsymbol{\beta}|\boldsymbol{\beta}_t)p^*(\boldsymbol{\beta}_t|\mathbf{y})}{p^*(\boldsymbol{\beta}|\mathbf{y})}\delta(\boldsymbol{\beta}_t, \boldsymbol{\beta}) \\ &= \frac{p^*(\boldsymbol{\beta}_t|\mathbf{y})}{p^*(\boldsymbol{\beta}|\mathbf{y})}Q(\boldsymbol{\beta}|\beta_t). \end{aligned}$$

Substituting this in the expression of $\rho(\boldsymbol{\beta}_t, \boldsymbol{\beta})$ we obtain the required expression.

It is important to note that the methodology above is general enough to be applied to any computationally expensive MH sampler. However, for definiteness in following, the method is applied to a few specific classification models. The success of the two-stage method in any given model depends on the construction of a computationally cheap and accurate estimate of the likelihood. The accuracy and speed of the likelihood estimator governs the efficiency of the MCMC chain. For instance, if the likelihood estimator $\hat{p}(\mathbf{y}|\boldsymbol{\beta}')$ severely underestimates $p(\mathbf{y}|\boldsymbol{\beta}')$, then $\delta(\boldsymbol{\beta}_t, \boldsymbol{\beta}')$

will be small and the proposal will be rejected (even if it might be a reasonable candidate). On the other hand, if $\hat{p}(\mathbf{y}|\boldsymbol{\beta}')$ severely overestimates $p(\mathbf{y}|\boldsymbol{\beta}')$, then it will likely pass the first stage and get rejected in the second stage since $\rho(\boldsymbol{\beta}_t, \boldsymbol{\beta})$ decreases as a function of $p^*(\boldsymbol{\beta}|\mathbf{y}) = \hat{p}(\mathbf{y}|\boldsymbol{\beta})p(\boldsymbol{\beta})$; thus the algorithm will compute the full likelihood for an unfavorable candidate. Consequently, it is important to select an accurate approximation to the likelihood. Specific likelihood approximations will be discussed for the examples in Section 2.3.

2.2.5 Combining Consensus with Two-Stage Metropolis-Hastings

For larger data sets which may not fit in RAM, we propose a combination of the consensus and the two-stage Metropolis methods. This is identical to the consensus method with the exception that each partition uses the two-stage Metropolis sampler rather than the usual Metropolis sampler. Since two-stage MH will draw from the same distribution as MH on each partition, the results of the consensus method will remain the same.

2.3 Applications

The methods introduced above were implemented on three datasets. For initial testing, the methods were implemented on a relatively small dataset with just over 40,000 observations from a phone marketing campaign conducted by a Portuguese bank. A larger dataset of approximately 2.3 million observations consisting of individual household loan data from Freddie Mac was used to test how the methods scale. In both, logistic regression was used to classify observations, the latter also employs a hierarchical model. Lastly, the two-stage method was implemented on a BMARS model with a large (10^6 observations) simulated dataset.

2.3.1 Logistic Regression Model

We are considering a binary classification problem where the response $\mathbf{y}$ takes the value 0 or 1 where $\mathbf{y} = (y_1, \dots, y_n)$ and we have a vector of covariates $\mathbf{x}$. We use a logit link function to link the jth response with the covariates as

$$\begin{aligned}
y_j \mid \boldsymbol{\beta}, \mathbf{x}_j &\sim \text{Bernoulli}\{\pi(\mathbf{x}_j)\} \\
\pi(\mathbf{x}_j) &= \{1 + \exp(-\mathbf{x}_j\boldsymbol{\beta})\}^{-1} \\
\boldsymbol{\beta} &\sim \text{Multivariate-Normal}(\mathbf{0}, \Sigma_0)
\end{aligned}$$

where $\boldsymbol{\beta}$ is the l dimensional vector of classification parameters, $\mathbf{x}_j$ is the jth row of the design matrix ($i = j, \ldots, n$), and a Gaussian prior is placed on $\boldsymbol{\beta}$. The model's posterior distribution can be expressed as $p(\boldsymbol{\beta}|\mathbf{y}, \mathbf{x}) \propto p(\mathbf{y}|\mathbf{x}, \boldsymbol{\beta})p(\boldsymbol{\beta})$.

In the logistic regression models for both the Portuguese bank and Freddie Mac datasets, we estimate the log-likelihood using a variant of the case-control approximate likelihood (Raftery et al., 2012). To understand the approximation, it is important to realize the log-likelihood for a logistic regression model can be written as two sums:

$$\log\{p(\mathbf{y}|\boldsymbol{\beta}, \mathbf{x})\} = \sum_{j:y_j=1} \left\{\theta_j - \log(1 + e^{\theta_j})\right\} + \sum_{j:y_j=0} -\log(1 + e^{\theta_j}) \tag{2.4}$$

where $\theta_j = \log\{\pi(\mathbf{x}_j)/[1 - \pi(\mathbf{x}_j)]\} = \mathbf{x}_j\boldsymbol{\beta}$.

If the data are sparse, then the computation of the first sum will be relatively cheap, and only the second summation needs to be estimated. We use a subsampling method where a random sample of a observations is taken from the failed outcomes (i.e. $y_j = 0$). The second sum in (2.4) is estimated by multiplying the average log-likelihood of the a sampled observations by $n_0 = \sum_{j=1}^{n} I(y_j = 0)$, the number of failures in the dataset. Let A be the index values of the subsample of size a. Thus, the original log-likelihood is estimated as

$$\widehat{\log}\{p(\mathbf{y}|\boldsymbol{\beta}, \mathbf{x})\} = \sum_{j:y_j=1} \left\{\theta_j - \log(1 + e^{\theta_j})\right\} + \frac{n_0}{a} \sum_{j\in A} -\log(1 + e^{\theta_j}). \tag{2.5}$$

We note $\widehat{\log}\{p(\mathbf{y}|\boldsymbol{\beta}, \mathbf{x})\}$ is an unbiased estimate of the log-likelihood as $E[\widehat{\log}\{p(\mathbf{y}|\boldsymbol{\beta}, \mathbf{x})\}] = \log\{p(\mathbf{y}|\boldsymbol{\beta}, \mathbf{X})\}$. We could obtain an unbiased estimate of the likelihood by making a bias correction (Quiroz et al., 2014) but that is not necessary for our method as we are doing further filtering of the proposal using the exact likelihood method.

The above approximation to the likelihood yields the following result:

Result 3: When the proposal β is promoted from the first stage, then for large a it can be shown that $\rho(\cdot,\cdot)$ goes towards 1. Thus, the two-stage MH algorithm only calculates the original full data likelihood when there is a high probability of acceptance of the proposal.

2.3.1.1 Portuguese Bank Data

The Portuguese bank dataset was obtained from the University of California Irvine Machine Learning Archive and were analyzed in a recent paper by Moro, Cortez, and Rita (2014). The binary dependent variable of interest was whether or not a client subscribed to a term deposit after contact through a telephone marketing campaign. The predictor variables of interest were the client's previous promotion outcome (non-existent, failure, success), age (years), and type of contact (telephone, cellular), education level (8 categories), and marital status (married, divorced, single, unknown). The categorical variables were treated as nominal values and the continuous variable age was logged and centered. A vague prior was placed on $\boldsymbol{\beta}$ resulting in the following model:

$$\begin{aligned} y_j \mid \boldsymbol{\beta}, \mathbf{x}_j &\sim \text{Bernoulli}\{\pi(\mathbf{x}_j)\} \\ \pi(\mathbf{x}_j) &= \{1+\exp(-\mathbf{x}_j\boldsymbol{\beta})\}^{-1} \\ \boldsymbol{\beta} &\sim \text{Multivariate-Normal}(\mathbf{0}, \Sigma_0 = I * 10^2) \end{aligned}$$

where $\boldsymbol{\beta}$ is the vector of coefficients, Σ_0 is the prior covariance matrix for $\boldsymbol{\beta}$, I is the identity matrix and $\mathbf{x}_j$ is the jth row of the design matrix, $j = 1, \ldots, n$.

The two-stage, consensus, and standard MH algorithms were coded in Fortran and run for 100,000 iterations with a burn in of 5,000 values. The subsampling method was considerably slower and was consequently run for only 10,000 iterations with the same burn in of 5,000. In the consensus method, the data were randomly split into 14 partitions. In the two-stage method, a single random subsample of 1,400 observations was taken prior to MCMC. This subsample was used to approximate the log-likelihood during the first stage on each iteration of the two-stage MH algorithm.

In the subsampling method, a thin-plate spline surface was fit to a subsample of the data (1,000 observations) prior to MCMC. For simplicity, in this smaller dataset, the thin-plate spline surface treated the categorical predictors as continuous. Although this resulted in a somewhat crude approximation to the log-likelihood surface, using a subsample of around 7,000 observations on each iteration allowed the MCMC chain to mix satisfactorily. To get a better idea of the speed of the subsampling method if a better spline surface was fit, it was also run subsampling 100 observations rather than 7,000. The number of likelihood evaluations per second for the MH and subsampling method (100 and 7,000 observations) were 355, 23.9, and 1.8 respectively.

Figures 2.1-2.3 compare the posterior densities of the two-stage, subsampling and consensus methods to the standard Metropolis sampler results. Figure 2.1 shows that the two-stage method matches the results obtained by the unmodified MH algorithm which is expected based on the theoretical results above. Figure 2.2 indicates that the subsampling method was effective in capturing the true posterior distribution since the subsampling method can be arbitrarily close to the true posterior based on the subsample size (Quiroz et al., 2014). Figure 2.3 shows that the consensus method matches the true posterior very well, with the exception of β_{10} and β_{15} which have a larger spreads and are slightly biased. Interestingly, β_{10} and β_{15} are the coefficients for education ('illiterate'), and marital status ('unknown') which have only 18 and 80 cases in the dataset respectively. Paradoxically, as Scott et al. (2016) points out, we are suffering from a case of small sample bias in a large dataset, which is a potential issue in consensus Monte Carlo applications. Since the Portuguese bank dataset is relatively small (approximately 40,000 observations), we refer the reader to the next section to better understand how these methods might scale to larger datasets, as well as a more detailed comparison of the speed and efficiency of the methods.

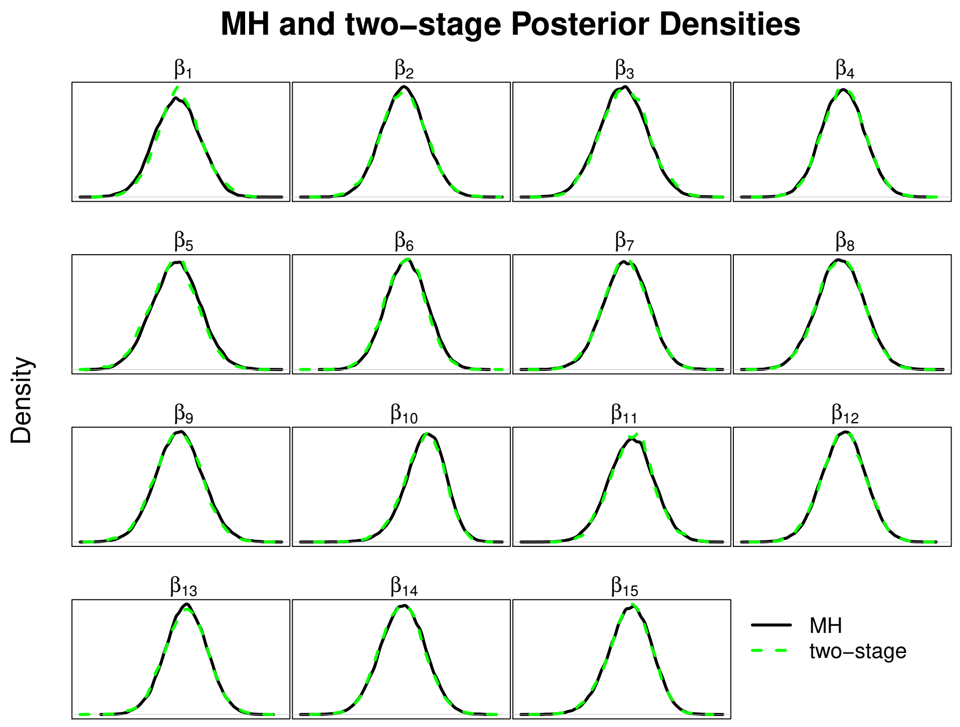

Figure 2.1: Posterior densities from the Portuguese bank data, two-stage Metropolis vs. Metropolis-Hastings. The MH and two-stage MH methods are represented by the solid and dashed lines, respectively.

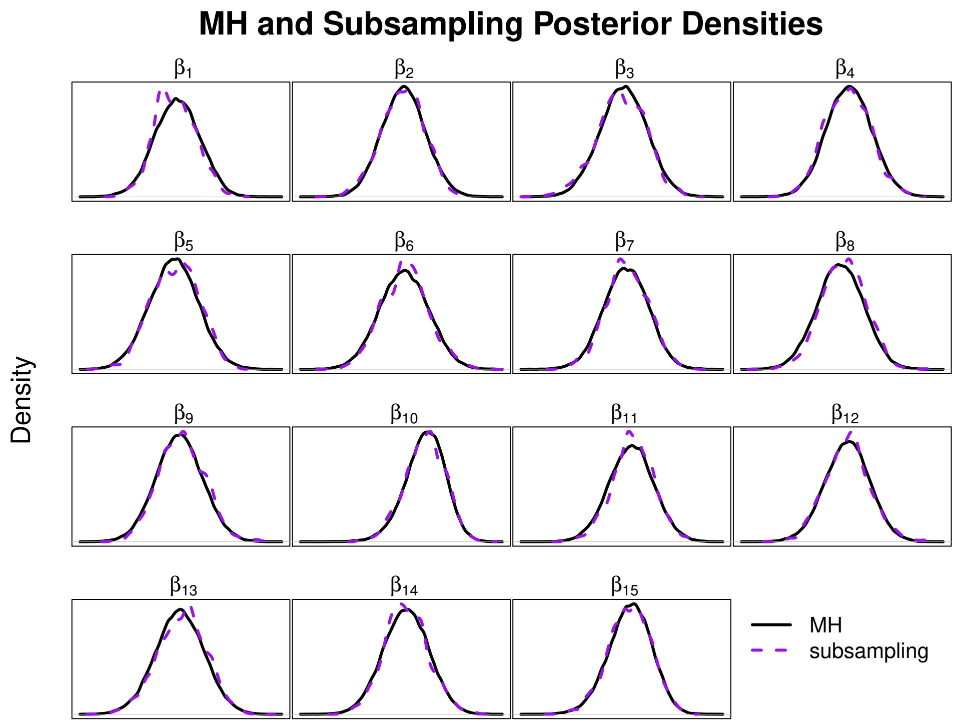

Figure 2.2: Posterior densities from the Portuguese bank data, subsampling vs. MH. The MH and subsampling methods are represented by the solid and dashed lines, respectively.

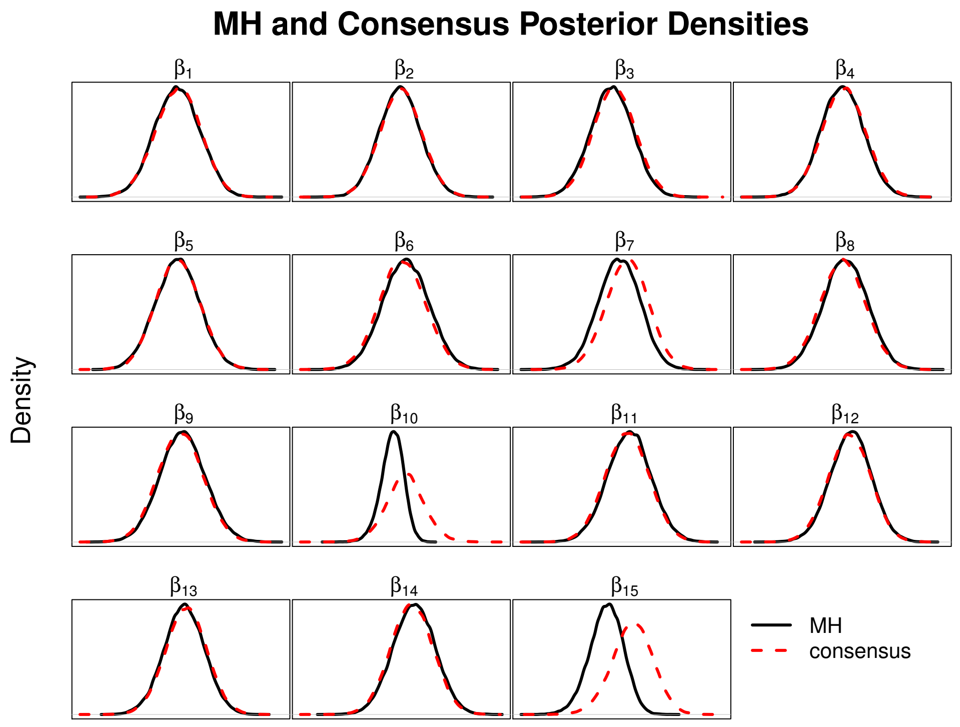

Figure 2.3: Posterior densities from the Portuguese bank data, consensus Monte Carlo vs. MH. The MH and consensus Monte Carlo methods are represented by the solid and dashed lines, respectively.

2.3.1.2 Freddie Mac Data, Logistic Regression

The loan data from Freddie Mac was obtained in September 2015 from Freddie Mac's website. The data consists of approximately 2.3 million loans which Freddie Mac acquired during 2009-2010 and contains monthly performance data on each loan. The binary dependent variable of interest is whether or not a loan was foreclosed by the end of September 2014.

In order to understand and quantify the effects of various covariates on foreclosure, a logistic model was used. Covariates of interest include the date of the first mortgage payment, FICO score, debt to income ratio, original principal balance of the loan, and first-time home-buyer status (yes, no, unknown). Each variable was transformed, centered, and scaled as appropriate. A vague prior was placed on $\boldsymbol{\beta}$ yielding the following logistic model:

$$\begin{aligned} y_j \mid \boldsymbol{\beta}, \mathbf{x}_j &\sim \text{Bernoulli}\{\pi(\mathbf{x}_j)\} \\ \pi(\mathbf{x}_j) &= \{1 + \exp(-\mathbf{x}_j \boldsymbol{\beta})\}^{-1} \\ \boldsymbol{\beta} &\sim \text{Multivariate-Normal}(\mathbf{0}, \Sigma_0 = I * 10^2) \end{aligned} \tag{2.6}$$

where $\boldsymbol{\beta}$ is the vector of coefficients, Σ_0 is the covariance matrix for $\boldsymbol{\beta}$, I is the identity matrix of appropriate dimension and $\mathbf{x}_j$ is the jth row of the design matrix, $j = 1, \ldots, n$.

The coefficients of the model were estimated using the usual MH, consensus MH, and two-stage MH algorithm, all of which were coded in Fortran. In order to provide a fair comparison with the consensus Monte Carlo algorithm, both the MH and two-stage MH algorithm were parallelized with $M = 14$ partitions as described in Section 2.2 using Open MPI for Fortran. In the two-stage MH algorithm, the s observations used to approximate the log-likelihood in the first stage were selected by randomly selecting s/M observations from each partition prior to the start of the MH algorithm.

The subsampling MH method was not employed on the Freddie Mac dataset since it was not likely to computationally competitive in this setting. Since the data are extremely sparse, the likelihood can be easily calculated for the cases when $y_j = 1$, so subsampling would only need

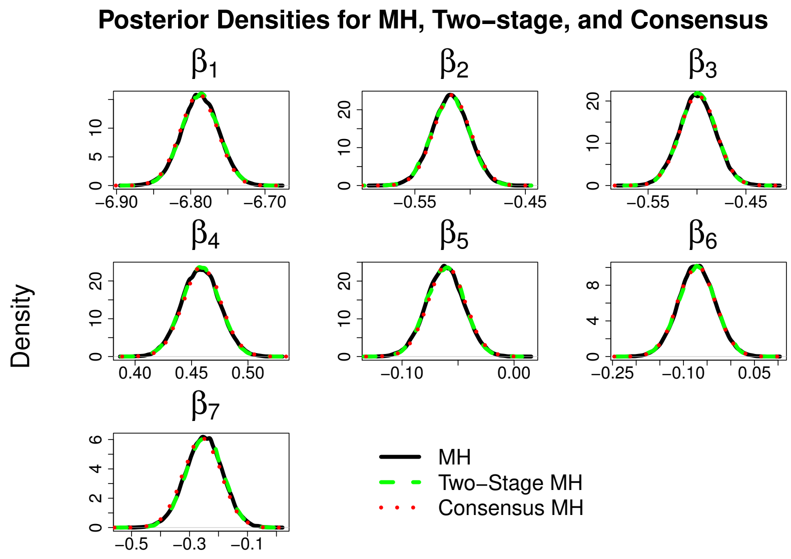

Figure 2.4: Posterior densities from Freddie Mac data. MH, two-stage MH, and consensus MH.

to be employed when $y_j = 0$. For full implementation, three separate spline surfaces would be required to be fit for each category of first-time home-buyer status (no, yes, unknown). Even if a relatively small sample was used for each spline surface approximation (e.g. several thousand observations), the corresponding matrices to calculate the spline fits would be in total far larger than the design matrix itself, and that computation is only the first step. Furthermore, the subsampling method is not likely to see the gains of parallelization that the MH and two-stage MH algorithms receive since more data will need to pass between processes and/or each process will have to calculate the same information in parallel (which defeats the purpose of parallelization).

The usual MH, consensus, and two-stage MH algorithms were run for 100,000 iterations with a burn-in period of 5,000. Parameters were updated sequentially and proposal variances were chosen such that the acceptance rate for each parameter was near 50%. Figure 2.4 plots the densities of the posterior distribution of $\beta_1, \ldots, \beta_7$. The densities are essentially indistinguishable between the MH, two-stage, and consensus methods. The execution times were 118, 115, and 81 minutes for the parallelized MH, consensus, and two-stage methods, respectively. In this particular application, the autocorrelation in the two-stage method was slightly more persistent than the regular MH algorithm. Consequently, it is of interest to evaluate the efficiency of the three MCMC chains, accounting for autocorrelation. This can be done by measuring the effective draws per minute (EDPM), which is a measure of the equivalent number of independent posterior draws per minute the MCMC chain represents. The EDPM diagnostic incorporates both the execution time and autocorrelation of the chain to measure its efficiency:

$$\text{EDPM} = t^{-1}\left(\frac{m}{1 + 2\sum_{k=1}^{\infty}\rho_k}\right)$$

where m is the number of MCMC iterations, t is the execution time of the MCMC chain in minutes, and ρ_k is the autocorrelation at the kth lag of the chain. EDPM can be calculated by estimating ρ_k with $\hat{\rho}_k$, the sample autocorrelation of the MCMC chain. In order to compare the efficiency of two MCMC chains, we can compute the relative effective draws per minute (REDPM) as

$$\text{REDPM} = \frac{\text{EDPM}_{\text{Algorithm 1}}}{\text{EDPM}_{\text{Algorithm 2}}}.$$

Figure 2.5 plots the REDPM of the two-stage and consensus methods relative to the MH method for each coefficient, $\beta_1, \ldots, \beta_7$. Also plotted are the REDPM values for the MCMC chain thinned by keeping every 10th and 20th values of the chain. We note that the two-stage method had REDPM values which were always above 1, and with the exception of one parameter, was always above the REDPM values of the consensus method. The median REDPMs for the two-stage method were 1.27, 1.44, and 1.47 as contrasted with 1.03, 1.07, and 1.02 for the consensus method (no thinning, keeping every 10th and 20th observations respectively). Thus in this application, the two-stage method appears to perform best in terms of speed, efficiency, and accuracy.

2.3.1.3 Freddie Mac, Hierarchical Logistic Regression

Bayesian statistics provide a simple way to fit hierarchical models, and with the help of MCMC, estimation of the parameters is generally straightforward. In addition to the covariates in the previous logistic regression model, the Freddie Mac dataset specifies which bank originally serviced

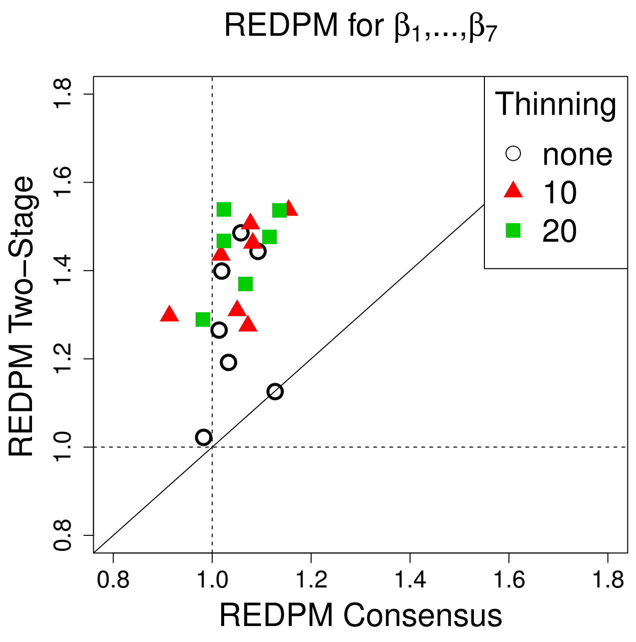

Figure 2.5: REDPM with respect to the MH algorithm for two-stage and consensus MH. REDPM is plotted for the original MCMC chain and the chain thinned every 10 and 20 values.

the loan. It is of particular interest to understand how delinquency rates vary between banks during this time period. In order to accomplish this, we specify the following hierarchical model:

$$
\begin{aligned}
y_{jk} \mid \theta_k, \boldsymbol{\beta}, \mathbf{x} &\sim \text{Bernoulli}\{\pi(\mathbf{x}_{jk})\} \\
\pi(\mathbf{x}_{jk}) &= [1 + \exp\{-(\theta_k + \mathbf{x}_{jk}\boldsymbol{\beta})\}]^{-1} \\
\boldsymbol{\beta} &\sim \text{Multivariate-Normal}(\mathbf{0}, \Sigma_0 = I * 10^2) \\
\theta_k &\overset{iid}{\sim} \text{Normal}(0, \tau^2) \\
p(\tau) &\propto \tau^{-2}
\end{aligned}
$$

where $\boldsymbol{\beta}$ is the vector of coefficients (the same covariates as in (2.6) with an intercept), Σ_0 is the covariance matrix for the vague prior on $\boldsymbol{\beta}$, I is the identity matrix of appropriate dimension, $\mathbf{x}_{jk}$ is the row of the design matrix corresponding to observation y_{jk}, the jth loan originating from the kth bank, and θ_k represents a random intercept term for the kth bank who serviced the loan,

$k = 1, \ldots, 16$, $j = 1, \ldots, n_k$. Lastly, Jeffrey's prior was placed on τ.

In this model, interest lies primarily in the posterior distribution of τ, which provides us with an understanding of the variability of loan foreclosure rates between banks after controlling for the other covariates. For datasets which have relatively small n, the MH algorithm is straightforward to implement on this simple hierarchical model. The two-stage method is also easily extended to this hierarchical model, however, the consensus and the subsampling methods are not as easily implemented.

The two-stage and the usual MH algorithms were successfully implemented. As before, the data was partitioned into 14 partitions and the likelihoods were computed in parallel on each iteration. Both chains were run for 100,000 iterations with a burn-in period of 5,000. Parameters were updated sequentially and proposal distributions were chosen such that the acceptance rates for each parameter was near 50%. The two-stage method was implemented twice with sample sizes of 224,000 and 22,400 observations which were sampled prior to running the algorithm (1000 and 100 data values from each bank on each partition, roughly 10% and 5% of data). The total run times for the parallelized MH and the two-stage MH (10% and 5% subsample) were 1106, 849, and 639 minutes, respectively. We note that the variance of the proposal distribution for the two-stage MH with 5% subsampling was reduced (compared to the MH and two-stage MH with a 10% sample) in order to obtain the desired acceptance rate of the MCMC chain.

As in the other two applications, the posterior densities for the MH and two-stage MH for the 24 parameters were within Monte Carlo error (since both the MH and two-stage MH algorithm attain the correct stationary distribution). For brevity, we plot only the posterior distribution of τ since it is the primary parameter of interest (Figure 2.6).

Figure 2.7 plots the REDPM values of the 24 parameters for the hierarchical model. We note that the two-stage algorithm using 10% of the data consistently had REDPM values greater than 1, whereas using 5% of the data yielded more variability in the REDPM values, including 4 values lower than 1 on the un-thinned MCMC chain. In the 5% sampling case, all the values of REDPM < 1 were elements of $\boldsymbol{\beta}$, not $\boldsymbol{\theta}$. This may be an artifact of the sampling design since sampling

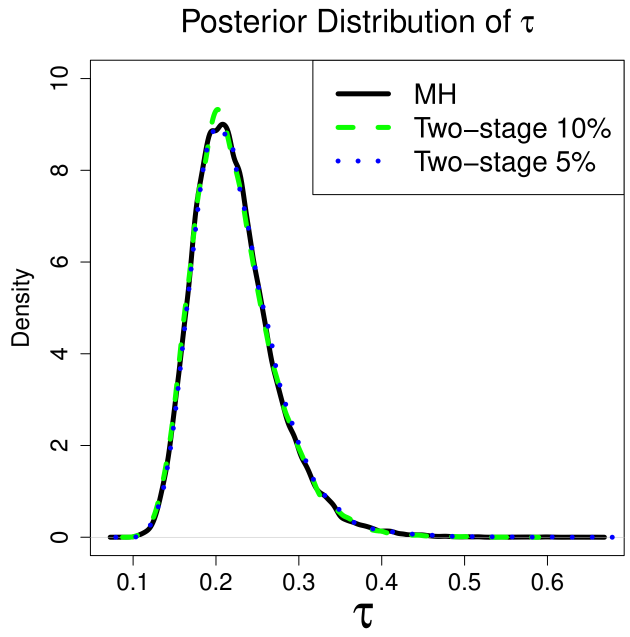

Figure 2.6: Posterior Distribution of τ for the MH and two-stage method subsampling 5% and 10% of the data.

was stratified by bank, and therefore some of the covariates may not have had adequate coverage in the smaller sample size. Thinning improves REDPM most dramatically for low REDPM values in the 5% sample, but otherwise doesn't seem to cause any major shifts in REDPM. Overall, the two-stage method showed increases in efficiency for the majority of the parameters.

The consensus method can be applied to this model as long as all the loans originating from a particular bank are in the same partition. The method proposed by Scott et al. (2016) requires running independent MCMC chains in parallel and then combining the draws of τ in the usual manner, and then discarding the values of $\boldsymbol{\theta}$ and $\boldsymbol{\beta}$. Once the draws of τ are combined, these values are sent to each partition which independently draw new values of $\boldsymbol{\beta}$ and $\boldsymbol{\theta}$ from $p(\boldsymbol{\beta}|\tau, \mathbf{X})$ and $p(\boldsymbol{\theta}|\tau, \mathbf{X})$. In our case, however, these distributions are not in standard form and are not easily sampled from. In implementation, the first consensus MCMC chain to obtain draws of τ was faster than the traditional MH algorithm with a parallelized likelihood but performed more slowly than the two-stage method (1106, 910, 849, 639 minutes for MH, consensus, and two-stage

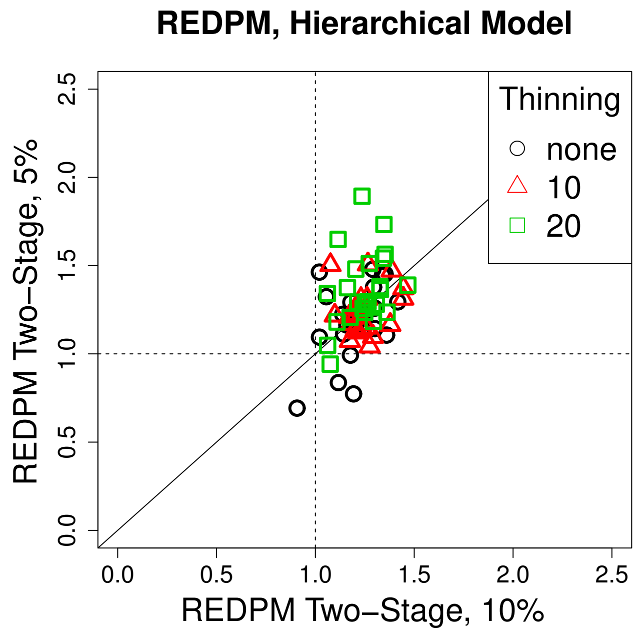

Figure 2.7: REDPM for the two-stage method with respect to the MH algorithm for subsample sizes of 5% and 10%.

MH (10% and 5% subsampling) respectively). Since the speed of the first run of the consensus method was slower than the two-stage method and the two-stage MH was more efficient than the consensus method in the previous model, drawing values from $p(\boldsymbol{\beta}|\tau, \mathbf{X})$ and $p(\boldsymbol{\theta}|\tau, \mathbf{X})$ was not implemented.

The subsampling method can also in theory be applied to this model. However, this requires fitting 48 spline surfaces prior to running the MCMC (16 banks, 3 levels of first-time home-buyer status). These spline surfaces were fit using the methodology provided by Ma, Racine, and Yang (2015) using a subsample of $s = 16,000$ observations (1,000 observations per group). However, on each iteration, approximating the log-likelihood surface for the entire dataset requires 48 matrix multiplications of dimension $z_i \times s, \sum_{i=1}^{48} z_i = n - s - n_1 = 2,297,813 - 3,711 - 16,000 = 2,278,102$ where $n_1 = \sum I(y_i = 1)$. These matrices were too large to fit into RAM, thus we were unable to implement the subsampling MH. Even if the data did fit into RAM, the computational cost of estimating the likelihood contribution with splines would likely be greater than evaluating the likelihood directly. Furthermore, implementing the subsampling method in parallel is not likely to produce significant gains in computation time since it will require either calculating the same quantities on each process (which defeats the purpose parallelizing) or passing vectors of information (rather than scalars) between processes.

2.3.2 Bayesian Multivariate Adaptive Regression Splines

The two-stage method also has applications in more complicated classification settings, including Bayesian multivariate adaptive regression splines (BMARS) (Friedman, 1991; Holmes and Denison, 2003). BMARS is a non-linear classification method which is extremely flexible for classification problems where the relationship between the response and covariates is complex, unknown, or otherwise difficult for the analyst to specify. It uses the data to adaptively choose splines and knots to flexibly model classification problems. Since the splines and knot locations are not known a priori, BMARS requires the use of reversible jump MCMC (Green, 1995) to explore a parameter space with varying dimension.

Even in this more complicated setting, implementing the two-stage MH requires only a few extra lines of code, but can still produce a faster MCMC chain. To quantify the effectiveness of the two-stage method using BMARS, one million observations were simulated from the following model:

$$y \sim \text{Bernoulli}\{[1+\exp(-\pi(\mu))]^{-1}\}$$

$$\mu = x_1 + x_2 - x_3 - x_4 + x_1x_2 - .5x_1x_3 - x_2x_3 + .2x_1x_2x_3;$$

$$x_1 \sim U(0,1),\ x_2 \sim N(0,1),\ x_3 \sim U(0,2),\ x_4 \sim N(0,2^2)$$

Where $U(a,b)$ denotes a uniform distribution on the interval (a,b). The BMARS method was implemented using the prior distributions outlined by Holmes and Denison (2003), to which we refer the reader to their paper for details. The two-stage method was implemented by choosing a random subsample prior to MCMC which was used in each iteration to approximate the likelihood. The log-likelihood approximation in the first stage was calculated as $\hat{l} = (n/a)l_{sub}$ where n and

Subsample Percentage	SD	Acceptance Rate	Time (sec)	Time Ratio
-	.0005	.31	41,594	-
15	.0005	.18	28,134	.68
10	.0005	.15	27,422	.66
5	.0005	.12	25,384	.61
1	.0005	.06	25,144	.60
1	.0001	.21	28,261	.68

Table 2.1: The results of the BMARS MCMC with the two-stage BMARS MCMC. The first row corresponds to the average of 5 runs of the usual BMARS algorithm to which the two-stage runs are compared. The last column is the ratio of the two-stage time to the regular MCMC time in the first row. Note that as the subsampling percentage decreased, the speed of the two-stage algorithm increased while the acceptance rate decreased for a fixed proposal standard deviation (SD).

a are the number of observations in the whole dataset and subsample, respectively, and l_{sub} is the log-likelihood contribution of the subsample.

The BMARS algorithm was run a total of 10 times on this simulated dataset. The usual BMARS algorithm was run 5 times and the average is summarized in the first row of Table 2.1. The two-stage method was implemented on the remaining five runs with various subsampling percentages. Once the priors are in place, there are only two parameters which need to be specified in the BMARS method: the maximum number of interactions allowed for the basis functions (which was chosen to be 3), and a tuning parameter (the proposal standard deviation of the spline coefficients).

From Table 2.1, it is clear that the two-stage method is faster than the usual BMARS MCMC, with all two-stage runs producing a 30%-40% reduction in time. As with the previous examples, as the subsampling percentage decreased, the acceptance rate of the two-stage MCMC chain decreased and the speed increased. When sampling only one percent of the data, the acceptance rate was very low, so it was re-run with a smaller proposal standard deviation. This led to an increase in the acceptance rate with a slight reduction in speed.

Due to the varying dimension of the parameter space during MCMC, comparing efficiency of the MCMC chain is not straightforward. Consequently, to determine the effectiveness of the two-stage method, one thousand observations were used as a test set, and the predictions based on both

MCMC chains were compared and are shown in Figure 2.8. The top-left panel of Figure 2.8 compares the predicted probabilities of two BMARS runs (neither implementing the two-stage method) to provide a visual of Monte Carlo error. The two-stage method with 15%, 10%, and 5% subsampling produced predictions which appear to be within Monte Carlo error of the usual BMARS algorithm. The two-stage 1% subsampling (sd = .0005) showed slightly more variability and 1% subsampling (sd = .0001) shows a fair amount of variability in the predictions. Interestingly, although the predictions between the BMARS and two-stage methods become more variable as the subsampling percentage decreased, the root-mean-square error (RMSE) of the predictions were essentially the same (RMSE for the 5 BMARS runs: .0806, .0805, .0806, .0806, .0807; RMSE for the 5 two-stage runs: .0807, .0806, .0808, .0804, .0806 for 15%, 10%, 5%, 1% (sd = .0005), 1% (sd = .0001), respectively). This gives further evidence that the two-stage method can still be highly effective even when using a very small percentage of the data as a subsample.

2.4 Discussion

Perhaps the most pressing question regarding two-stage MH is how to select the subsample size. From experience the authors note that for a fixed proposal distribution variance, decreasing the subsampling percentage will at some point decrease acceptance rates of the MCMC chain. This is due to the fact that the estimate of the likelihood is either overestimating or underestimating the likelihood ratio which causes proposed parameter values to be discarded by either the first stage (if the estimate of the likelihood ratio is too small) or the second stage (if the estimate of the likelihood ratio is too large). If too small of a subsample is used, the variance of the proposal distribution will need to be reduced to obtain the desired acceptance rate of the MCMC chain. A smaller subsample will increase the speed of the chain, but will likely increase the autocorrelation of the chain since the variance of the proposal distribution will need to be reduced. Even so, the hierarchical model for the Freddie Mac data still performed well with sampling only 5%-10% of the data, and the BMARS application performed well even with 1%-15% subsampling. This indicates that the speed and efficiency of the two-stage method may be somewhat robust to the subsample size used to approximate the log-likelihood.

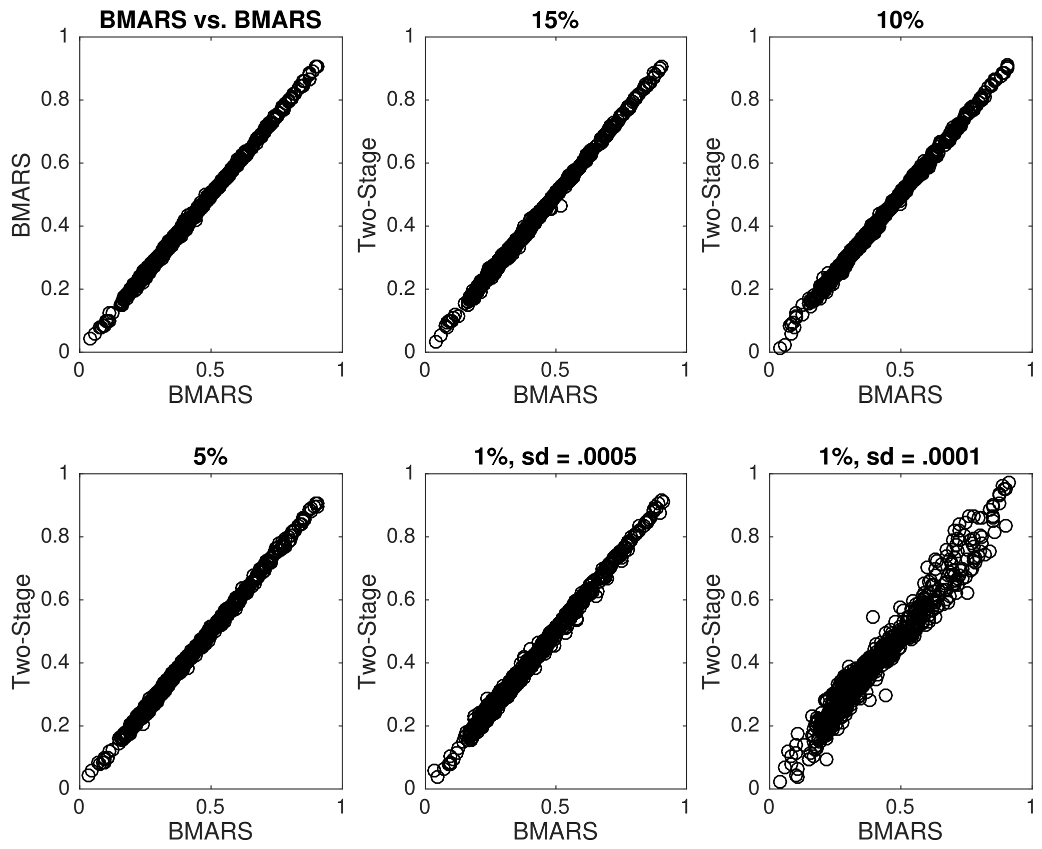

Figure 2.8: Comparison of the test-set predictions between the BMARS MCMC and the two-stage BMARS MCMC for various subsampling percentages. The top-left panel compares the predicted probabilities between two BMARS runs for a visual of Monte Carlo error.

One of the main advantages of the two-stage method is its simplicity and ease of implementation. It requires taking only one subsample prior to the MCMC algorithm and then adding a few lines of code to implement the first screening stage. Furthermore, it can be applied to any model in which a computationally cheap estimate of the likelihood can be obtained. Even using naive likelihood approximations, the two-stage method has performed well. If more precise likelihood estimates can be acquired for a particular model, the two-stage method may be even more effective at screening out bad proposals (although the speed will still depend on a computationally cheap likelihood estimate).

The consensus method is also generally straightforward in simple models, but even in hierarchical models it places restrictions on how the data can be partitioned and may require sampling from distributions which cannot be sampled from directly (which adds another potentially computationally demanding layer). The subsampling method requires the most effort to implement since it requires fitting spline surfaces to the data. Furthermore, these spline surfaces may require very large matrix multiplications to provide the approximation to the likelihood surface on each iteration of the MCMC.

The success of the two-stage method on the complex BMARS method indicates that it has potential in many other applications. Other potential non-linear classification methods include relevance vector machine (Tipping, 2001) and support vector machine models (Mallick, Ghosh, and Ghosh, 2005). This can also be extended in a multivariate responses framework (Holmes and Mallick, 2003). Perhaps most importantly, the two-stage method is not limited to classification problems. It can be applied to any model where a computationally cheap and accurate approximation of the likelihood can be constructed.

2.5 Conclusion

The results from this chapter indicate there are a number of tall data Bayesian methods which are effective in obtaining/approximating the posterior distribution more quickly than traditional methods. Two-stage MH is simple to implement, fast, and overall more efficient than consensus, subsampling, or unmodified MH algorithms in our applications. Combining two-stage MH with the consensus method shows promise for even larger datasets in which the data cannot fit in RAM. Future extensions to this work include applying the method to handle more complicated likelihoods, and finding better likelihood approximations which are still computationally cheap to evaluate.

3. A CONDITIONAL DENSITY ESTIMATION PARTITION MODEL USING LOGISTIC GAUSSIAN PROCESES

3.1 Introduction & Literature Review

Conditional density estimation, sometimes referred to as density regression, is a method used to estimate the conditional distribution of a response variable, y, given a vector of covariates, $\mathbf{x}$. Many common regression methods are special cases of conditional density estimation. For instance, the Gaussian regression model assumes that the mean of y changes with $\mathbf{x}$ with the variance of y being constant over the covariate space. Conditional density estimation is particularly useful when a parametric form linking the covariate space $\mathbf{x}$ with y is unknown or violates the assumptions of existing parametric methods. In its most flexible forms, conditional density estimation can be viewed as a general nonparametric regression method.

There are a number of existing frequentist approaches to perform density regression including kernel methods (Fan et al., 1996; Fu et al., 2011), spline methods (Kooperberg and Stone, 1991; Stone et al., 1997), and mixtures of experts (Jacobs et al., 1991). There are also several Bayesian approaches to conditional density estimation. One of the popular approaches is to use mixture models for the conditional distribution of $p(y \mid \mathbf{x})$ and allow the mixing weights as well as the parameters to depend on the covariates (Chung and Dunson, 2009; Dunson and Park, 2008; Dunson et al., 2007; Griffin and Steel, 2006). An alternative approach is to apply the logistic Gaussian process model in the conditional density estimation setting (Tokdar et al., 2010). Latent variable models have been utilized by Kundu and Dunson (2011) and Bhattacharya and Dunson (2010). A multivariate spline based method (Shen et al., 2016) and an optional Polya tree based method (Ma and Wong, 2011) have been recently proposed. Petralia et al. (2013) perform density regression using a convex combination of dictionary densities using a fixed tree decomposition which scales to accommodate a large number of features.

The method proposed in this chapter provides a novel partition model (Holmes et al., 2005;

Denison et al., 2002) framework to perform conditional density estimation using logistic Gaussian processes. The data are partitioned using a Voronoi tessellation on the covariate space $\mathbf{x}$ and the distribution of y within each partition is modeled using a univariate logistic Gaussian process. The primary goal of the partition model is to infer the partition structure. In other partition models, this is typically done through selecting priors such that the parameters in each partition element can be integrated out analytically, providing the marginal probability for the data y (Kim et al., 2005; Denison and Holmes, 2001). The logistic Gaussian process model does not have an analytical form for the marginal of y, but this can be estimated using a Laplace approximation (Riihimäki et al., 2014), allowing an effective search of the posterior tessellation structure.

By construction, the partition model assumes that the distribution of y is piecewise constant over the covariate space. This type of model is useful when it is known that the distribution of y changes sharply over $\mathbf{x}$. One notable motivating example is that of windmill power output where the distribution of power output changes sharply across at least two covariates. Wind direction, due to a wake effects, and wind speed, due to a limit on maximum power output both create sharp changes in the distribution of y over short distances in the covariate space. Conditional density estimation methods which assume that the density of y changes smoothly throughout the covariate space are not designed to handle these sharp boundaries.

Philosophically, partition models are very different from models which assume a smooth change in the distribution of y over $\mathbf{x}$. Since partition models assume that the data within each partition element are drawn from the same distribution, they provide marginal density estimates for *regions* of the covariate space, not specific points. Indeed, it would be ideal if a partition model were developed in which the model within each partition allowed the distribution to change smoothly, allowing the model to coherently capture both sharp and smooth changes in y over $\mathbf{x}$. The marginalization of such models, however, would likely prove difficult, and further research is needed in this area. Even so, we argue that one of the partition model's greatest virtues is that of providing an understanding of how the density of y changes over $\mathbf{x}$, i.e., providing inference on the tessellation structure.

The proposed partition model provides insight into how and where the density of y changes over $\mathbf{x}$. In other conditional density estimation models, the user must specify which points in the covariate space to view the density of y. It is possible when exploring the relationship between y and $\mathbf{x}$ that interesting relationships may be missed due to the volume of $\mathbf{x}$. Even in relatively small covariate dimensions, say 5, exploring a grid of 3 points in each dimension leads to $3^5 = 243$ different densities to view, leaving the researcher to determine and interpret how the density changes over $\mathbf{x}$, which may be difficult to understand even in relatively low dimensions. Further, in this example, using only 3 points in each direction is an extremely sparse grid in 5 dimensions, and it is possible that interesting relationships might be missed. On the other hand, the partition model is designed to determine where important changes in y occur throughout $\mathbf{x}$, providing greater interpretability. Indeed, the model is similar in spirit to the Bayesian classification and regression tree (CART) model (Chipman et al., 1998; Denison et al., 1998) as a decision tool to understand how and where the density of y changes in different regions of $\mathbf{x}$.

Although we are not the first to attempt to use a partition-type model structure for conditional density estimation, we are the first to use logistic Gaussian processes within a Voronoi tessellation partition to model conditional densities. Several tree methods have been proposed including Ma (2017) and Petralia et al. (2013). Tree models can provide nice logical interpretations of partition structures, however, they can become overly complex, even in low dimensions with few partition elements, if the true partition structure is not oriented to the block partitions of the tree model. The tessellation structure in this paper is more flexible than tree methods since it is constructed using a weighted Euclidean distance, which allows the model to partition the covariate space into regions of varying shapes, rather than sets of hyper-rectangles.

Logistic Gaussian processes have been used in other contexts to perform conditional density estimation. Tokdar et al. (2010) use the logistic Gaussian process to model the joint distribution of the response y and the covariates $\mathbf{x}$ and utilized a subspace projection method to reduce the dimension of the covariates. Conversely, we are fitting univariate logistic Gaussian processes within each region of $\mathbf{x}$, hence avoiding the curse of dimensionality. Furthermore, in joint modeling approaches, y and $\mathbf{x}$ are not identified as the response and covariates. Our experience is that the distribution of the response y is highly influenced by the distribution of $\mathbf{x}$, especially when the dimension of $\mathbf{x}$ is high.

Furthermore, this model does have important consistency properties which can be proved. These results, developed by a colleague, while not included in this dissertation will be published later. This will be, to the best of our knowledge, the first paper that considers the posterior consistency in estimating conditional distributions in the partition model framework. Indeed, there are a few papers which have considered theoretical properties of conditional density estimation models using other modeling frameworks (Tokdar and Ghosh, 2007; Pati et al., 2013; Norets and Pelenis, 2012; Bhattacharya and Dunson, 2010).

In Section 3.2 we present the conditional density estimation model in a partition framework and provide a reversible jump MCMC algorithm for estimation. Section 3.3 applies the method to both simulated and real data; Section 3.6 provides some additional discussion and Section 3.7 concludes.

3.2 Bayesian Hierarchical Conditional Density Estimation Partition Model

3.2.1 Modeling the Partition Structures Using a Voronoi Tessellation

The partition model divides the p-dimensional covariate space $\mathcal{D}$ into M distinct pieces where y is assumed to independently follow a different density $p_i(\cdot)$ within each partition. The partitioning of the covariate space is done through a Voronoi tessellation. The tessellation is defined by M centers $\mathbf{c}_1, \ldots, \mathbf{c}_M$ that divide the covariate space into M disjoint regions $R_1, \ldots, R_M$ where R_i consists of all the observed $\mathbf{x}$ that are closest to center $\mathbf{c}_i$. Formally, $R_i = \{\mathbf{x} \in \mathcal{D} : ||\mathbf{x} - \mathbf{c}_i|| < ||\mathbf{x} - \mathbf{c}_j|| \; \forall \; i \neq j\}$. Here, $||\mathbf{x}|| = ||(x_1, \ldots, x_p)|| = \sum_{i=1}^{p} w_i x_i^2$ where $\mathbf{w}$ is a normalized weighting vector, $\sum w_k = 1$, which places different weights on each of the covariates (Holmes et al., 2005). The weighting provides additional flexibility in the class of tessellations. Figure 3.1 shows an example of a Voronoi tessellation in two dimensions with $\mathbf{w} = (.5, .5)$.

For simplicity, we assume that the possible centers of the tessellation are restricted to the ob-

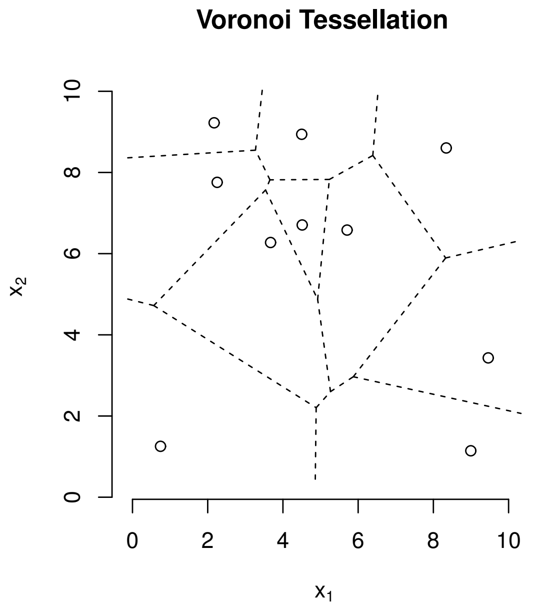

Figure 3.1: A Voronoi tessellation in two dimensions with weight vector $\mathbf{w} = (.5, .5)$. The points represent the 10 centers $\mathbf{c}_1, \ldots, \mathbf{c}_{10}$ and the dashed lines represent the partition borders.

served data points $\mathbf{x}$. Next, we assign prior distributions for both the number of centers as well as the center locations. An intuitive way to express the prior is $p(\mathbf{c}, M, \mathbf{w}) = p(\mathbf{c} \mid M)p(M)p(\mathbf{w})$ where

$$\begin{aligned} p(M) &= \text{DU}(M \mid 1, \ldots, M_{\text{max}}) \\ p(\mathbf{c} \mid M) &= \text{DU}\left(\mathbf{c} \mid 1, \ldots, \binom{n}{M}\right) \\ p(\mathbf{w}) &= \text{Di}(\mathbf{w} \mid 1, \ldots, 1) \end{aligned}$$

where DU$(x \mid 1, \ldots, n)$ means discrete uniform on $1, \ldots, n$ and M_{max} is the maximum number of allowable centers, a hyperparameter chosen by the user. The prior on $p(\mathbf{c} \mid M)$ gives equal weight to all possible combinations of M centers with possible center locations corresponding to the n observed values of the covariates $\mathbf{x}$. This prior penalizes larger models as long as $M < \lfloor n/2 \rfloor$, which should always be the case since one cannot adequately model the density in each partition element with so few points. The vector $\mathbf{w}$ has the Dirichlet prior which is uniform on the simplex.

3.2.2 Likelihood and Prior

Let $\mathbf{y}_i = \{y_j : \mathbf{x}_j \in R_i,\ j = 1, \ldots, n\}$ be the set of n_i observed response variables whose covariates are in the ith region of the tessellation. We assume that within the ith region of the partition the data, $\mathbf{y}_i$, independently follow a univariate logistic-Gaussian model (Lenk, 1988). The logistic Gaussian process models the density using an exponentiated Gaussian process over a bounded interval, $\mathcal{V}_i$. Therefore, the data $\mathbf{y}_i$ are assumed to be drawn independently from

$$p_i(y) = \frac{\exp(f_i(y))}{\int_{\mathcal{V}_i} \exp(f_i(s))ds} \tag{3.1}$$

where $f_i(\cdot)$ is modeled with a latent Gaussian process as

$$\begin{aligned} f_i(\cdot) &= \mu_i(\cdot) + g_i(\cdot), \\ g_i(\cdot) &\sim GP(0, \kappa_{\boldsymbol{\theta}_i}(\cdot,\cdot)), \\ \mu_i(\cdot) &= \mathbf{h}(\cdot)^T \boldsymbol{\beta}_i, \end{aligned} \tag{3.2}$$

where $\kappa_{\boldsymbol{\theta}_i}(\cdot,\cdot)$ is a covariance function which depends on the smoothing hyperparameter vector $\boldsymbol{\theta}_i$. For simplicity in exposition, assume $\boldsymbol{\theta}_i$ is known; its selection via empirical Bayes methods will be detailed later in the paper. The function $\mathbf{h}(z) = (z, z^2)^T$ is used to encourage decreasing tails in the density function $p_i(\cdot)$. By placing a Gaussian prior on $\boldsymbol{\beta}_i \sim N(\mathbf{b}, B)$, $\boldsymbol{\beta}_i$ can be integrated out to yield the marginal prior for $f_i(\cdot)$

$$f_i(\cdot) \sim GP\left(\mathbf{h}(\cdot)^T\mathbf{b},\ \kappa_{\boldsymbol{\theta}_i}(\cdot,\cdot) + \mathbf{h}(\cdot)^T B \mathbf{h}(\cdot)\right).$$

While a full Gaussian process prior is attractive due to its flexibility, it does present some computational challenges, particularly when trying to integrate out the latent function $f_i(\cdot)$ to obtain the marginal distribution of $\mathbf{y}_i$. To aid in providing a computationally feasible solution, Riihimäki et al. (2014) approximate the logistic-Gaussian density via discretization. The bounded region $\mathcal{V}_i$ is discretized into a regular grid of r_i subregions centered at $Z_i = (z_{i1}, \dots, z_{ir_i})^T$. Note that this discretization occurs over the range of y, not $\mathbf{x}$, since the data model within each partition is independent of the covariates, $\mathbf{x}$. The function $f_i(\cdot)$ is evaluated at r_i points in the vector $\mathbf{f}_i := (f_i(z_{i1}), \dots, f_i(z_{ir_i}))^T$ and the continuous density is approximated with the discrete version. That is, the likelihood of an observation falling into the jth subregion would be

$$\frac{w_j \exp(\mathbf{f}_{ij})}{\sum_{j=1}^{r_i} w_j \exp(\mathbf{f}_{ij})} \tag{3.3}$$

where w_j is the width of the region and $\mathbf{f}_{ij} = f_i(z_{ij})$, the jth element of $\mathbf{f}_i$. Note that since we are using a regular grid, the w_js cancel out in (3.3). This leads us to the joint likelihood of $\mathbf{y}_i$ as

$$p(\mathbf{y}_i \mid \mathbf{f}_i) = \exp\left\{ \mathbf{y}_i^{\star T}\mathbf{f}_i - n_i \log\left(\sum_{j=1}^{r_i} \exp(\mathbf{f}_{ij}) \right) \right\}$$

where n_i is the length of $\mathbf{y}_i$, and $\mathbf{y}_i^\star$ is a column vector of length r_i with the jth element as the number of elements of $\mathbf{y}_i$ that fall into the subregion centered at z_{ij}.

Since we are modeling $f_i(\cdot)$ as a Gaussian process it follows that $\mathbf{f}_i$ has a multivariate normal distribution

$$\mathbf{f}_i \sim N(H_i\mathbf{b},\ K_i + H_i B H_i^T)$$

where K_i is a $r_i \times r_i$ matrix with j, kth element $\kappa_{\boldsymbol{\theta}_i}(z_{ij}, z_{ik})$ and H_i is a $r_i \times 2$ matrix with jth row $\mathbf{h}(z_{ij})^T$.

The partition model assumes independence between data in different partitions, therefore, the posterior distribution can be expressed as a product

$$p(T, \mathbf{f}_1, \dots, \mathbf{f}_M \mid \mathbf{y}) \propto p(T) \prod_{i=1}^{M} p(\mathbf{y}_i \mid \mathbf{f}_i) p(\mathbf{f}_i)$$

where $T = \{M, \mathbf{c}, \mathbf{w}\}$ represents the tessellation parameters. In our case, as in many other partition models, interest lies in searching the posterior of the tessellation, $p(T \mid \mathbf{y})$. This is typically accomplished using reversible jump Markov chain Monte Carlo (Green, 1995) after integrating out the partition-specific parameters. In this framework we seek

$$\begin{aligned} p(T \mid \mathbf{y}) &\propto p(T) \prod_{i=1}^{M} \int p(\mathbf{y}_i \mid \mathbf{f}_i) p(\mathbf{f}_i) d\mathbf{f}_i \\ &= p(T) \prod_{i=1}^{M} p(\mathbf{y}_i) \end{aligned} \tag{3.4}$$

Unfortunately, there is no analytical solution for the integrals, but they can be estimated via a Laplace approximation. The Laplace approximation requires finding $\hat{\mathbf{f}}_i = \arg\max_{\mathbf{f}_i} p(\mathbf{y}_i \mid \mathbf{f}_i)p(\mathbf{f}_i)$ via Newton's method and using a Taylor expansion to construct the normal approximation. Indeed, the model presented is constructed primarily by embedding the univariate model of Riihimäki et al. (2014) into a partition model framework to infer the effect of covariates via the partition structure. Indeed, Riihimäki et al. (2014) provides the form of the Laplace approximation to obtain $p(\mathbf{y}_i)$ in Equation (16) of their paper, to which we refer the reader for the specific details.

Lastly, we discuss the form of $\kappa_{\boldsymbol{\theta}_i}(\cdot, \cdot)$ and the selection of $\boldsymbol{\theta}_i$ in each partition. For the purposes of this paper, we assume the covariance function $\kappa_{\boldsymbol{\theta}_i}(\cdot, \cdot)$, which depends on hyperparameters $\boldsymbol{\theta}_i = (\sigma_i, l_i)$, is the stationary squared-exponential covariance function

$$\kappa_{\boldsymbol{\theta}_i}(z, z') = \sigma_i^2 \exp\left(-\frac{1}{2l_i^2}(z - z')^2\right)$$

where σ_i is the magnitude hyperparameter and l_i is a length-scale hyperparameter which together govern the smoothness properties of $f_i(\cdot)$. The value of $\boldsymbol{\theta}_i$ is chosen via empirical Bayes by maximizing the posterior $p(\sigma,\ l \mid \mathbf{y}_i) = p(\sigma)p(l) \int p(\mathbf{y}_i \mid \mathbf{f}_i)p(\mathbf{f}_i)d\mathbf{f}_i$, which we shall denote $\hat{\boldsymbol{\theta}}_i = (\hat{\sigma}^2, \hat{l}^2)$ in each partition. We choose two temporary hyperpriors, $p(\sigma)$ and $p(l)$, to guide the selection of $\boldsymbol{\theta}_i$. We place a weakly informative half Student-t distribution with one degree of freedom and a variance equal to 10 for the magnitude parameter and the same prior with a variance of 1 for the length-scale hyperparameter (Riihimäki et al., 2014).

As a review, computing the marginal distribution of $\mathbf{y}$ in Equation (3.4) given a tessellation is done through the following steps:

- Given $T = \{M, \mathbf{c}, \mathbf{w}\}$, determine $\mathbf{y}_i$, $i = 1, \ldots, M$ based on covariate values.
- For each $\mathbf{y}_i$, find $\hat{\boldsymbol{\theta}}_i$ by maximizing $p(\mathbf{y}_i)$ subject to the temporary hyperprior on $\boldsymbol{\theta}_i$ and set $\boldsymbol{\theta}_i = \hat{\boldsymbol{\theta}}_i$.
- Compute $p(\mathbf{y}_i)$ in each partition via the Laplace approximation.

The next section describes the reversible jump Markov chain Monte Carlo algorithm for searching the posterior of the tessellation.

3.2.3 Markov Chain Monte Carlo Algorithm

The posterior distribution of the tessellation structure, T, does not have an explicit form, thus obtaining posterior draws of the tessellation proceeds using a reversible jump Markov chain Monte Carlo algorithm (Green, 1995). A reversible jump algorithm is necessary due to the varying dimension of the parameter space of the tessellation structure. The algorithm to explore the posterior of the tessellation has four possible moves, which are now briefly described.

- *Birth:* Add a new tessellation center randomly from $\mathbf{x}$ which is not currently a tessellation center.
- *Death:* Randomly remove an existing tessellation center.
- *Move:* Randomly move an existing tessellation center to another value of $\mathbf{x}$ which is not currently a tessellation center.
- *Change:* Randomly select an element of $\mathbf{w}$, modify it according to $q(\cdot, \cdot)$, and normalize the resulting vector.

In our implementation, we place an equal probability of 1/4 for each of the moves in the algorithm, but these can be changed, if desired. Algorithm 1 in Figure 3.2 shows the pseudo code for performing the reversible jump algorithm. After choosing an initial tessellation, the algorithm consists of iteratively choosing one of the four moves with probability 1/4, and accepting the proposed moves with probability

$$\alpha = \min\left(1, \frac{q(w \mid w^{(p)}) \prod_{i=1}^{M_p} p(\mathbf{y}_i)}{q(w^{(p)} \mid w) \prod_{i=1}^{M_c} p(\mathbf{y}_i)}\right) \tag{3.5}$$

where T_p is the proposed tessellation with M_p centers obtained from one of the four possible proposals, T_c is current state of the tessellation with M_c centers, and $q(\cdot \mid \cdot)$ is the proposal used to modify $\mathbf{w}$. In our implementation, we set $q(w \mid w')$ to be a truncated-normal distribution centered at w' and truncated to allow only positive values; it applies only to the single element of $\mathbf{w}$ which is modified in a change step. Note that the ratio involving $q(\cdot \mid \cdot)$ is 1 if a birth, death, or change step is proposed, since $\mathbf{w}$ remains unchanged. The only tuning parameters for the algorithm are the probabilities of proposing birth, death, move, and change steps and the proposal variance of $q(\cdot \mid \cdot)$.

It is important to note that for most non-boundary cases, the prior on the tessellation structure $P(T_p)$ and $P(T_c)$ does not appear in α due to cancellations with itself and the proposal distribution for the birth, death, move, and changing weight steps in the reversible jump algorithm. When M is at or near the boundary, however, adjustments to α need to be made in order to maintain the reversibility of the Markov chain. When $M = 1$ and a birth step is proposed, we must multiply the ratio in (3.5) by 3/4. When $M = 2$ and we propose a death step, the ratio must be multiplied by 4/3. The reverse must be applied when $M = M_{\text{max}},\ M_{\text{max}} - 1$.

3.2.4 Parallel Tempering

As is common in many partition models, such as classification and regression trees (Chipman et al., 1998), the mixing of the Markov chain may be poor due to multiple posterior nodes. In order to overcome this potential issue, parallel tempering is used to better explore the posterior distribution. Gramacy (2007) and Gramacy and Taddy (2010) use parallel tempering in tree models, and we employ a similar parallel tempering strategy.

Algorithm 1. Reversible Jump Markov Chain Monte Carlo Algorithm

```
Initialize Tessellation;
for $i \leftarrow 1$ to $N$ do
    Propose a birth, death, move, or change weight step with probability 1/4;
    if birth step then
        Propose a new tessellation center randomly from $\mathbf{x}$ which is not currently a center;
    else if death step then
        Randomly remove an existing tessellation center;
    else if move step then
        Move an existing tessellation center to a randomly chosen $\mathbf{x}$ which is not currently a
          tessellation center;
    else if change step then
        Randomly choose an element of the weight vector $\mathbf{w}$;
        Modify the element of the weight vector using proposal distribution $q(\cdot \mid \cdot)$;
        Normalize $\mathbf{w}$ such that $\sum w_i = 1$;
    end
    Accept the proposed change with probability $\alpha$;
end
Discard burn-in period;
```

Figure 3.2: Reversible jump MCMC algorithm for the conditional density estimation partition model.

To search the potentially multi-model posterior and to improve mixing of the Markov chain, parallel tempering (Geyer, 1991) is used. The parallel tempering algorithm starts multiple reversible jump Markov chain Monte Carlo algorithms in parallel, each with a slightly different target distribution, specifically

$$p^{(j)}(T \mid \mathbf{y}) \propto p(T) \left(\prod_{i=1}^{M} p_i(\mathbf{y}_i) \right)^{t_j}$$

where $1 = t_1 > t_2 > \ldots > t_d \geq 0$ are called the inverse temperatures, and are chosen using the sigmoidal ladder (Gramacy and Taddy, 2010). The algorithm proceeds by starting d parallel chains with target distributions $p^{(1)}(\cdot \mid \cdot), \ldots, p^{(d)}(\cdot \mid \cdot)$. Once every l iterations, two adjacent chains are randomly selected and switch partitions with a Metropolis-Hastings update with acceptance probability

$$\alpha_2 = \min\left\{1, \frac{p^{(j)}(T^{j-1} \mid \mathbf{y})p^{(j-1)}(T^j \mid \mathbf{y})}{p^{(j)}(T^j \mid \mathbf{y})p^{(j-1)}(T^{j-1} \mid \mathbf{y})}\right\}$$

where T^j is the current tree on the jth process and T^{j-1} is the current tree on process $j-1$. The resulting chain for $p^{(1)}(\cdot \mid \cdot)$ provides draws from the posterior distribution and explores the parameter space more efficiently than a single reversible jump Markov Monte Carlo chain.

The parallel tempering is designed to encourage smaller partitions for smaller inverse temperatures by tempering only the likelihood. This also has the advantage that we still have the cancellation of the prior with the proposal distribution for a specific tempered distribution.

Parallel tempering speeds up the search of the posterior and allows an easier transition than would be possible otherwise with a single chain. For instance, lower inverse temperatures generally have smaller partition sizes, allowing them to explore new weights and preliminary splits of the covariate space. As promising tessellations are found, they are passed up to higher inverse temperatures where additional refining is performed.

3.3 Simulations & Applications

3.3.1 Preliminaries

For our applications, we choose the maximum number of partitions to be $M_{\text{max}} = 10$ to maintain simplicity in interpretation as well as decreased computation. We set the hyperparameters for $\boldsymbol{\beta}$ as $\mathbf{b} = (0,0)^T$ and $B = 10^2 I$ where I is the 2×2 identity matrix. We note that each Z_i, $i = 1, \ldots, M$, the grid upon which the density of y is discretized upon, is a subset of a larger grid of 400 points $Z^\star = \{z_1^\star, \ldots, z_{400}^\star\}$ with $z_1^\star = \min\{\min(\mathbf{y}), \bar{\mathbf{y}} - 3\hat{\sigma}^2\}$ and $z_{400}^\star = \max\{\max(\mathbf{y}), \bar{\mathbf{y}} + 3\hat{\sigma}^2\}$ where $\bar{\mathbf{y}}$ and $\hat{\sigma}^2$ are the sample mean and variance of y. Specifically, $Z_i = \{z_{j'}^\star : z_{j'}^\star \in [\min\{\min(\mathbf{y}_i), \bar{\mathbf{y}}_i - 3\hat{\sigma_i}^2\}, \max\{\max(\mathbf{y}_i), \bar{\mathbf{y}}_i + 3\hat{\sigma_i}^2\}], j' = 1, \ldots, 400\}$ where $\bar{\mathbf{y}}_i$ and $\hat{\sigma}_i^2$ are the mean and standard deviation of the data $\mathbf{y}_i$. For our applications, we standardize the covariates and the response variable to have mean of 0 and standard deviation of 1. This standardization allows us to use the priors selected generally without needing to adjust the priors to account for the range of the data.

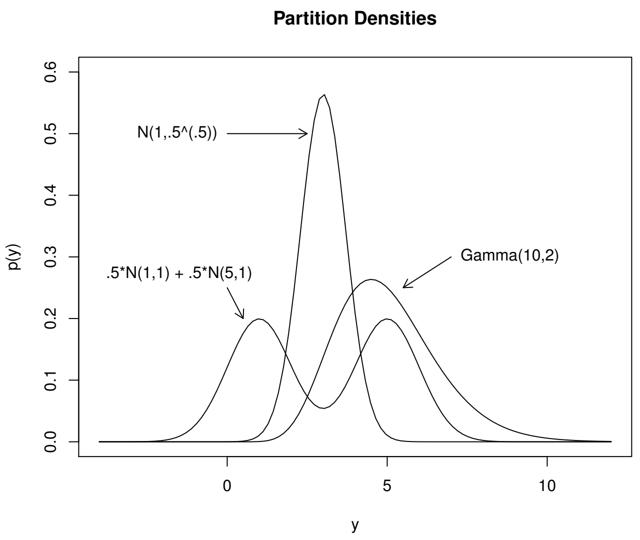

Figure 3.3: The three densities found in each partition element of the simulated dataset.

3.3.2 Simulation

A total of 500 observations were simulated from the following model: $X_1, X_2 \sim U(0,1)$ and

$$Y \mid X_1, X_2 \sim \begin{cases} \text{Gamma}(10,2) & \text{if } X_1 > X_2,\ X_2 < .75 \\ .5N(1,1) + .5N(5,1) & \text{if } X_1 < X_2,\ X_1 < .75 \\ N(1, \sqrt{.5}) & \text{if } X_1 > .75,\ X_2 > .75 \end{cases}$$

where Gamma(a,b) is a gamma distribution with shape parameter a and rate parameter b and $N(\mu, \sigma)$ denotes a normal distribution with mean μ and standard deviation σ. Figure 3.3 plots the three densities. In this simulation, there is considerable overlap in the distribution of y in each partition element, but the shape of the density of y is dramatically different. The Markov chain Monte Carlo algorithm was run for 10^5 iterations. The partition with the highest posterior probability is plotted in Figure 3.4 with the black lines indicating the true partition boundary. The method performed extremely well in this setting and correctly captured the partition. The model also does well in capturing the distribution within each partition element; see Figure 3.5.

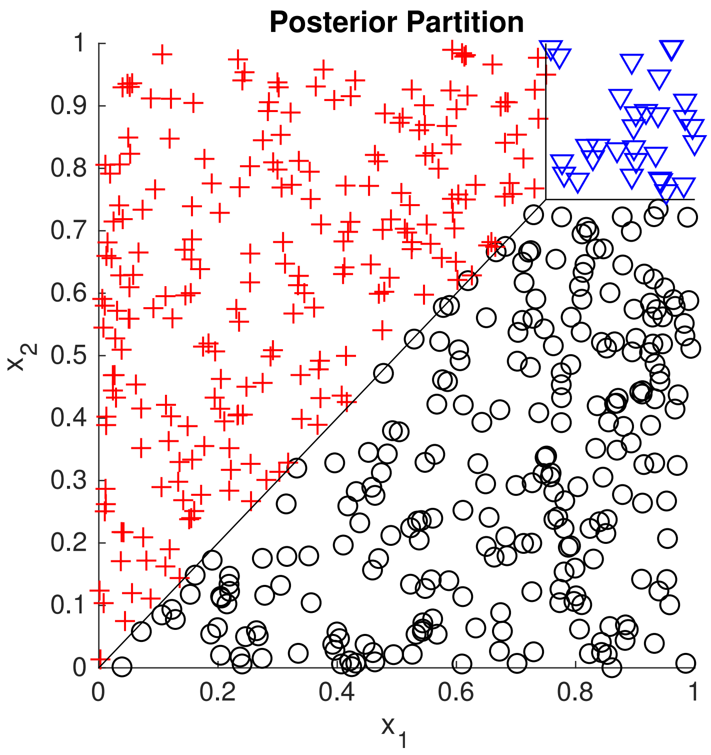

Figure 3.4: The partition with the highest posterior probability from the simulated dataset. The true partition boundaries are denoted by the black lines, and color and shape of the points designate their partition element.

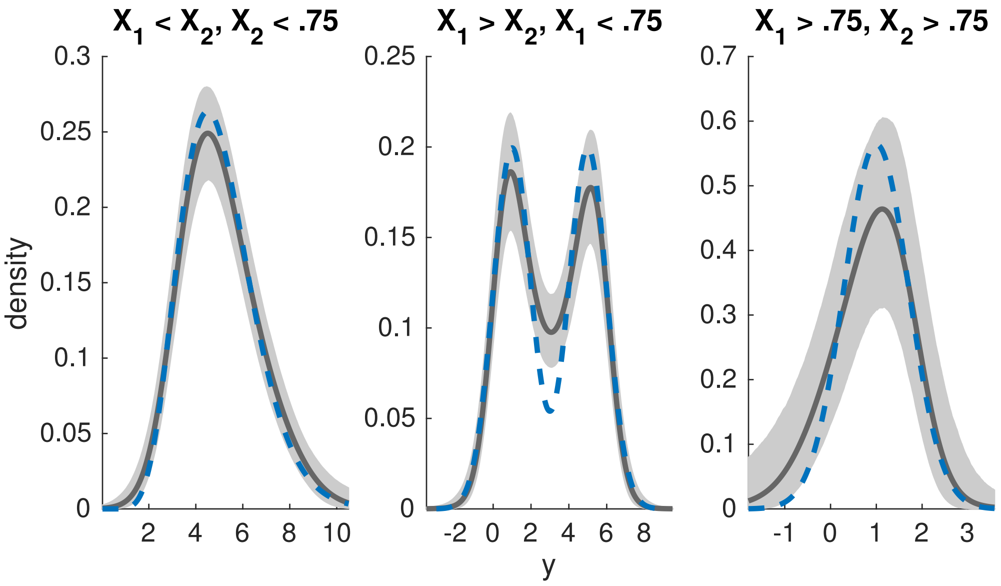

Figure 3.5: The posterior mean (solid) and 95% credible intervals (shaded) of the data in each partition element from the partition model. The true density is also shown (blue, dashed).

This dataset was analyzed by three other methods including dependent Bernstein polynomials (Barrientos et al., 2017), linear dependent tail-free processes (Jara and Hanson, 2011), and the Dirichlet process mixtures of normals method (Müller et al., 1996). All three methods are implemented in the `DPpackage` in `R`, and generally assume that the density of y changes smoothly as a function of the covariates – a very different assumption than the data generating model. The methods were run for a 10^4 iteration burn-in period and kept every other draw for the next 40,000 iterations for a total of 20,000 posterior draws. Details on the hyperparameters used are included in Section 3.5. Figures 3.6-3.8 compare the data generating and posterior densities at four different covariate locations, $\{(.76,.76),(.9,.9),(.1,.8),(.8,.1)\}$. The (.76,.76) location is located on the boundary of all 3 partition elements whereas the other 3 locations are away from it. At the (.76,.76) location, all three methods generally fail to capture the true density as seen in the top left panels. We found that the linear dependent tail-free process in Figure 3.6 generally does a poor job in capturing the density. The dependent Bernstein polynomial model in Figure 3.7 does better at the the (.1,.8) and (.8,.1) locations, but does not improve much at the (.9,.9) location as compared with (.76,.76). This is most likely due to the fact that there is less data in this partition element. Ma (2017) note that the performance of the Bernstein polynomial model in their piecewise examples worsened as sample size increased and attribute these results to the fact that the Bernstien polynomial model is designed to handle smooth changes in the density. The Dirichlet mixture model in Figure 3.8 does the best of these three “continuous" methods, generally capturing the data generating distribution in its credible intervals for the (.1,.8) and (.8,.1) covariate locations. It also performs noticeably better in the (.9,.9) case compared to the other two methods. Even so, this method still fails near the boundary, and has larger credible intervals.

This dataset was also analyzed with a Voronoi partition model which assumes normality within each partition element as described in Chapter 7 of Denison et al. (2002). The Markov chain ran for 80,000 iterations following a 20,000 iteration burn-in period. Since this partition model assumes normality within each partition element, model averaging was employed to obtain the posterior density at the same locations as the methods previously discussed. Figure 3.9 shows the results,

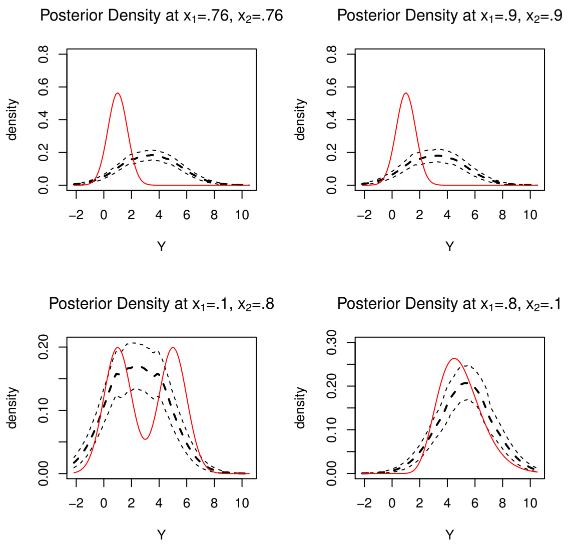

Figure 3.6: The posterior mean density (dashed, bold), 95% credible intervals (dashed), and the true density (solid, red) of the linear dependent tail-free process model.

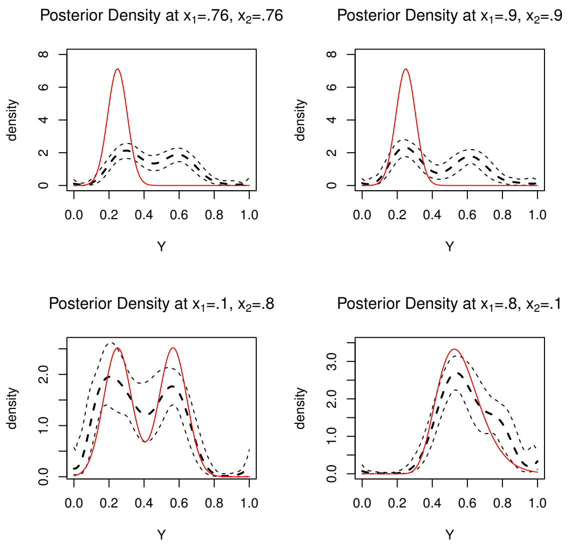

Figure 3.7: The posterior mean density (dashed, bold), 95% credible intervals (dashed), and the true density (solid, red) of the linear dependent Bernstein polynomial model.

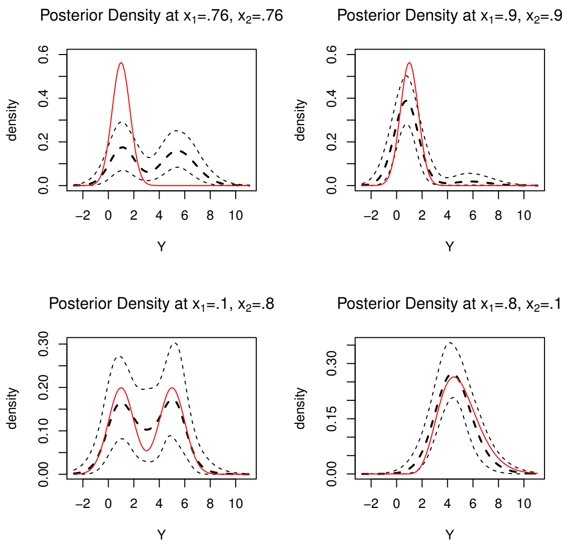

Figure 3.8: The posterior mean density (dashed, bold), 95% credible intervals (dashed), and the true density (solid, red) of the Bayesian Dirichlet process mixture of normals model.

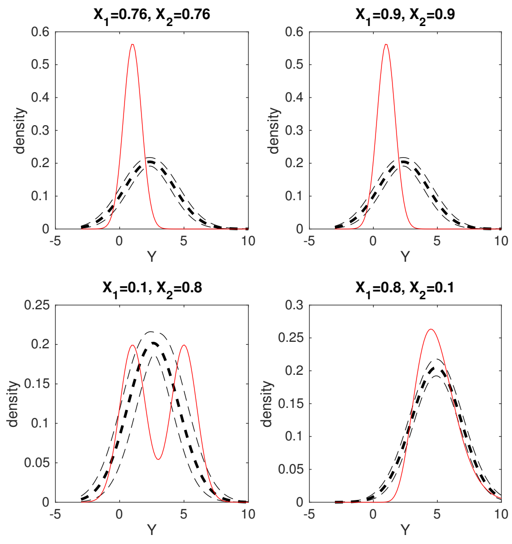

Figure 3.9: The posterior mean density (dashed, bold), 95% credible intervals (dashed), and the true density (solid, red) of the Voronoi partition model which assumes a normal density in each partition element.

which are similar to those of the linear dependent tail-free process in Figure 3.6. Details regarding the location of the code and implementation can be found in Section 3.5.

A direct comparison of this dataset to other partition models was not possible since Petralia et al. (2013) do not provide public code, and the code provided by Ma (2017) lacks adequate documentation at the time of this writing. However, we will replicate one of the scenarios in Ma (2017) next.

3.3.3 Comparison with Optional Polya Trees

Although we were unable to use the code from Ma (2017) due to inadequate documentation, we did replicate one of their examples and applied our method to it. The data are simulated from the following model:

$$\begin{aligned} X &\sim \text{Beta}(2,2) \\ Y \mid X < .25 &\sim \text{Beta}(30,20) \\ Y \mid .25 \le X \le .5 &\sim \text{Beta}(10,30) \\ Y \mid X > .5 &\sim \text{Beta}(0.5,0.5). \end{aligned}$$

Ma (2017) implement their model using sample sizes of 100, 500, and 2,500 observations. With 100 observations, their method found the correct partition via a tree partitioning structure, but they note that 100 data points is insufficient to adequately estimate the distribution in each partition element. We simulated 100 data points from the same model and implemented our method running the Markov chain for 10^4 iterations. The partition with the highest posterior probability correctly captured the true partition with 100% correct classification of the points. Furthermore, our method did a good job in modeling the density in each partition. Figure 3.10 shows the 95% credible intervals containing the true density a majority of the time, despite such a small sample size.

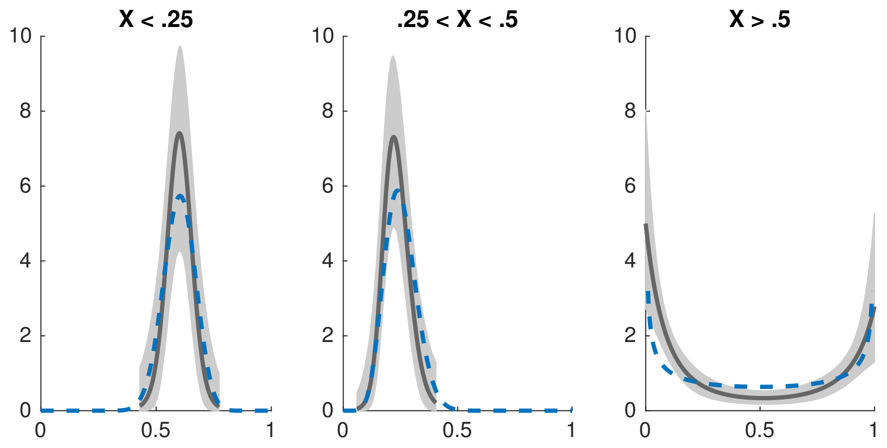

Figure 3.10: The posterior mean density (solid), 95% credible intervals (shaded), and the true density (dashed, blue) in the three posterior partition elements.

3.3.4 Smooth Changes

To demonstrate the partition model's performance on data where the density changes smoothly as a function of the covariates, 500 realizations of the vector (Y, X_1, X_2) were simulated from a trivariate normal distribution with mean $(1, 5, 7.5)$ and covariance matrix

$$\Sigma = \begin{bmatrix} 1 & .5 & .1 \\ .5 & 1 & .5 \\ .1 & .5 & 1 \end{bmatrix}.$$

The Markov chain was run for 10^5 iterations. Figure 3.11 shows the partition with the highest posterior probability in the top left panel, the other three panels plot the posterior mean, 95% credible intervals, and the true distribution at the tessellation center. Since we are estimating the marginal density of y in each region, there is no guarantee that the true distribution at the tessellation center will be contained in the credible intervals. Even so, the bottom two panels show that the credible intervals generally contain the distribution of the tessellation center whereas the top right panel performs slightly worse. Regardless, the partition model is performing as expected: it is identifying that both X_1 and X_2 influence the distribution of y and capturing general shifts in

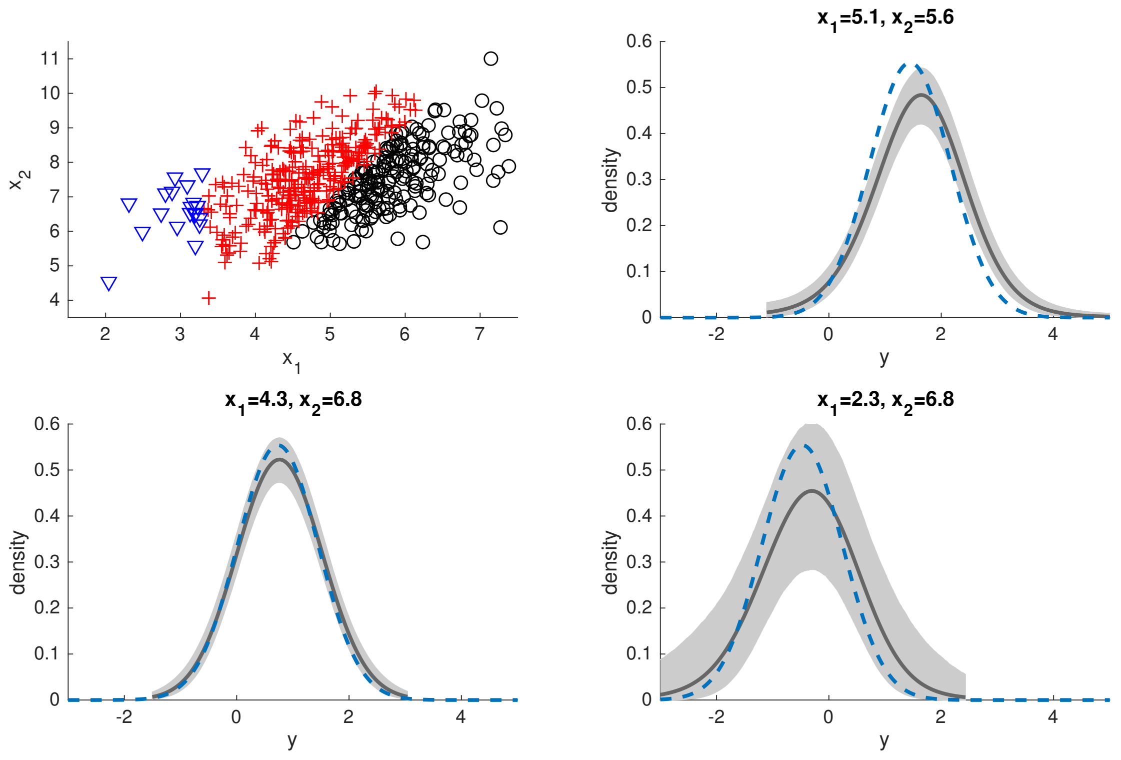

Figure 3.11: The top-left panel plots X_2 against X_1 with colors/symbols denoting the partition with the highest posterior probability. The remaining panels plot the posterior mean density (solid), 95% credible intervals (shaded), and the density (dotted, blue) at the tessellation centers in each partition element.

the distribution over the covariate space.

3.3.5 Windmill Data

Conditional density estimation is particularly useful when it is unclear how the density of y changes with respect to the predictors, $\mathbf{x}$. The exact relationship of electrical power output in windmills to covariates measuring wind speed, wind direction, air density, wind sheer, & turbulence intensity is unknown, but is believed to nonlinear. Furthermore, the conditional density of power output is known to have sharp changes as a function of wind speed and direction. For instance, modern wind turbines employ a pitch control mechanism to protect the generator under high wind. When the wind speed is high enough, the turbine blades start turning more parallel to the wind direction to reduce the energy absorption capability. Thus, for very high wind speeds, the power output is concentrated near a maximum power level. Sometimes, this dramatic change from

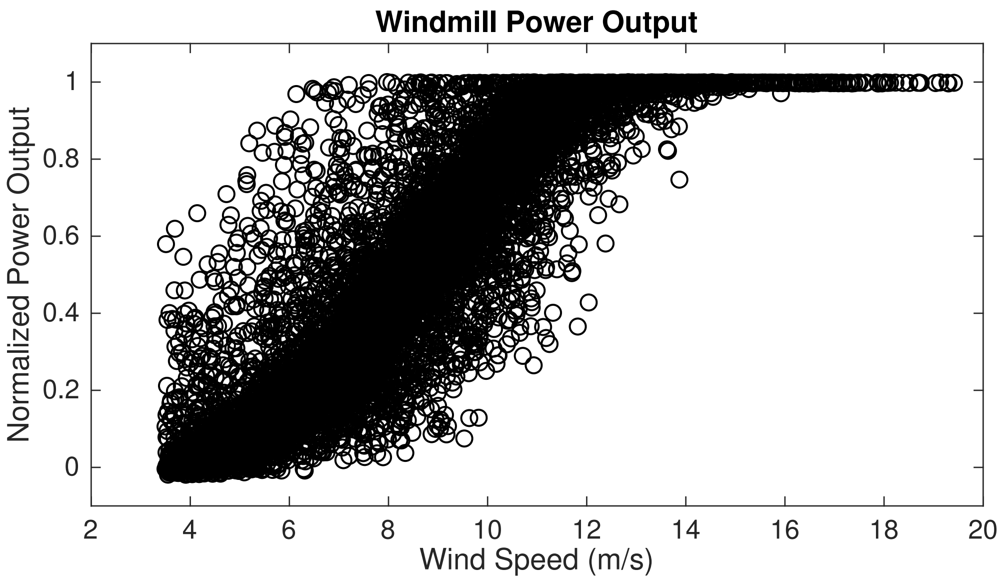

Figure 3.12: A plot of normalized power output against wind speed (m/s).

a very broad and often skewed distribution to a very narrow distribution can happen within a wind speed range of just 1 m/s. The dramatic change in the spread of power output as a function of wind speed can be seen in Figure 3.12.

Wind speed is not the only covariate which induces sharp changes in the distribution of power output. When terrain is not flat and smooth, there may also be sharp changes in power output as a function of wind direction, as the wind is affected by the terrain surrounding the turbine. When a wind turbine is downwind from another turbine, there is the potential for a wake effect which can also create sharp changes in power output over wind direction. Since the locations of turbines are fixed, changes in the wind direction will influence which turbines are downwind and thus influence the wake effect. Combining all of these effects together, we expect sharp changes in wind speed to occur in both the wind speed and wind direction. The wind turbine dataset consists of aggregated measurements of windmill power output and several covariates over 10 minute increments from a single turbine, consisting of 10,000 observations randomly sampled from a larger dataset. We analyze the wind turbine dataset in order to estimate the distribution of power output in various regions of the covariate space in order to better understand where the most dramatic changes occur. The partition model was fit to the data using 5 covariates: wind speed, wind direction, air density,

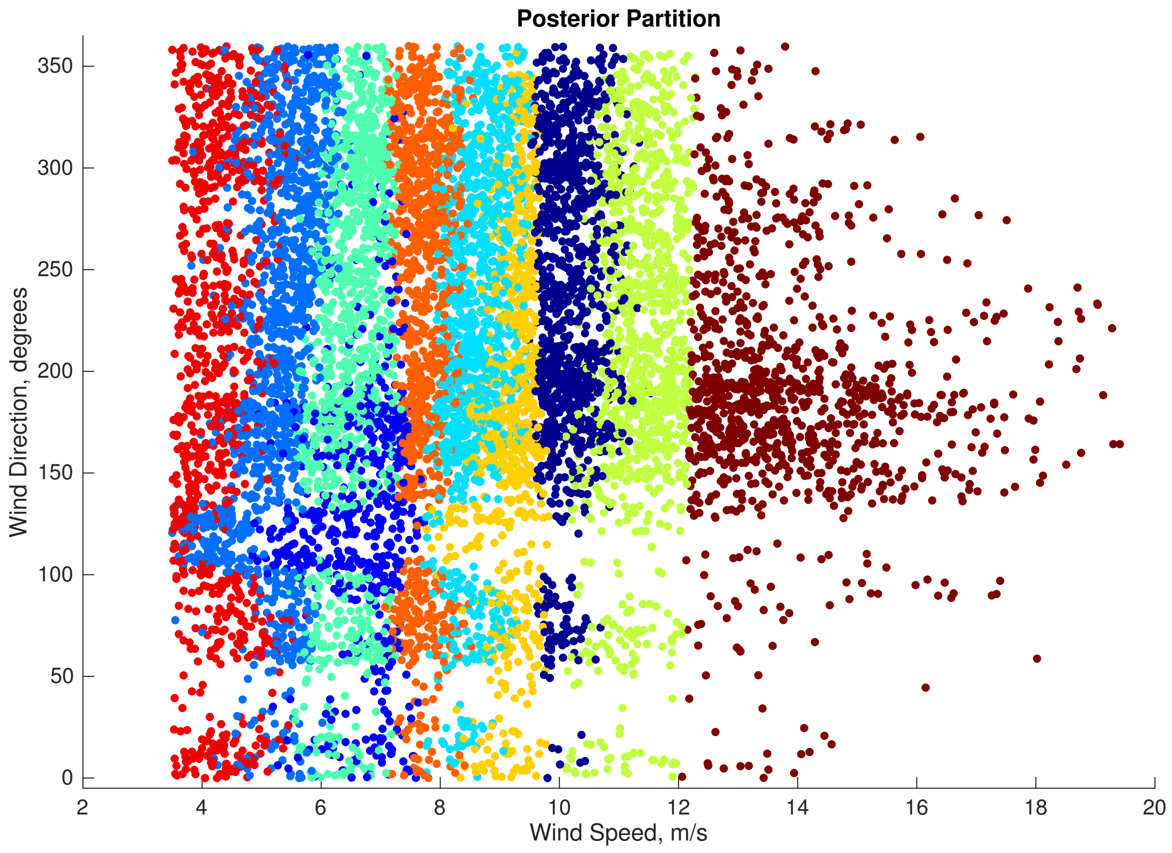

Figure 3.13: Wind direction plotted against wind speed with colors denoting the partition elements of the tessellation with the highest posterior probability. Wind speed describes the majority of the changes power output. Near a wind direction of 120 degrees, there is a sharp shift in the distribution of power output which is captured by the tessellation. Overlaps and blurred edges of the partitions indicate the role of other covariates in determining the partition structure.

wind sheer, & turbulence intensity. The reversible jump Markov chain Monte Carlo algorithm was run for 100,000 iterations.

Figure 3.13 shows the tessellation (in 2 dimensions) with the highest posterior probability. The partition structure identifies major changes across wind speed, and also identifies one interesting change across wind direction at about 120 degrees. This dramatic change in power output at about 120 degrees is believed to be caused by a wake effect from another turbine. Specifically, it is believed that the wake effect may have caused the instruments measuring wind speed to underestimate wind speed when the turbine is oriented at about 120 degrees. Figure 3.14 shows the posterior densities in four regions of the tessellation.

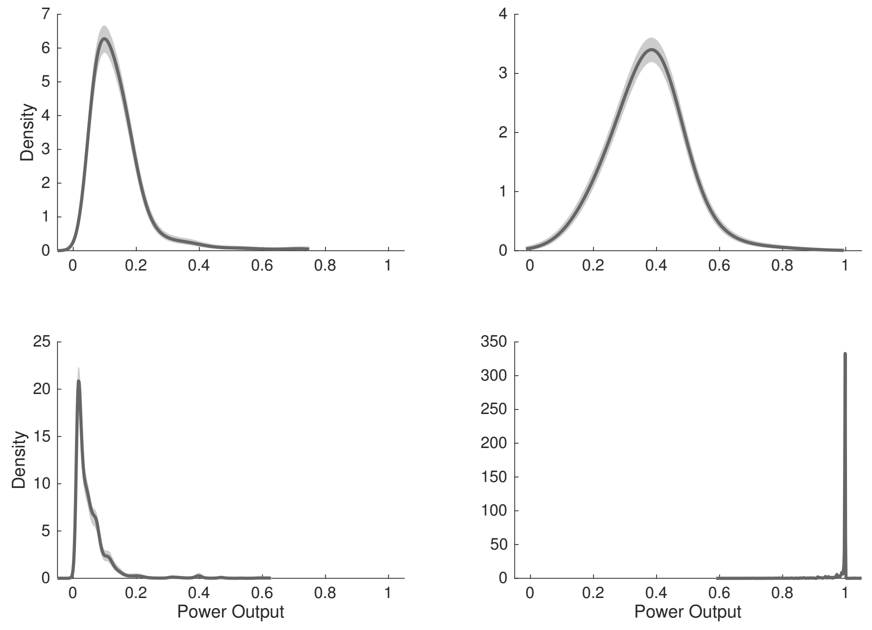

Figure 3.14: These four posterior densities with 90% credible intervals (shaded) show the dramatic changes in location, spread, and shape of the marginal density of normalized power output over 4 of the 10 partitions in the final posterior tessellation.

The code to run the partition model in the above examples was written in Matlab and utilizes portions of the excellent code developed by Vanhatalo et al. (2013).

3.4 Markov Chain Monte Carlo Mixing

Here we discuss the Markov chain performance using parallel tempering with 8 chains. We begin with a discussion of the first simulated example in Section 3.3.2. Figure 3.15 shows the traceplot for the log posterior for the untempered chain. It indicates that the posterior quickly attains the correct posterior node.

Our experience indicates that parallel tempering dramatically speeds up the discovery of true partitions in simulated examples. Even so, the difficulty of slow mixing of individual chains persists across individual temperatures. Even the chain with smallest inverse temperature of .1 only accepted 5% of birth steps, but accepted 86%, 96%, and 99% of death, move, and change steps, respectively. The high acceptance rates of the latter three moves is mostly due to the fact that the chain generally only had one partition, so the move and change steps acceptances are inflated

Inverse Temperature	Birth	Death	Move	Change
1	0.0001	0.0001	0.0007	0.2890
.98	0.0002	0.0000	0.0011	0.2912
.94	0.0001	0.0002	0.0010	0.3087
.85	0.0003	0.0002	0.0009	0.3471
.67	0.0009	0.0006	0.0038	0.4803
.43	0.0008	0.0008	0.0066	0.6386
.22	0.0054	0.0054	0.0384	0.7970
.1	0.0501	0.8591	0.9554	0.9872

Table 3.1: Proportion of accepted moves for each inverse temperature for the simulation in Section 3.3.2.

since they will always be accepted when the partition size is 1. Table 3.1 shows the proportion of acceptances for each inverse temperature. Swapping rates of the chains were generally higher for higher inverse temperatures and lower for lower inverse temperatures. This is likely due to the fact that the higher inverse temperatures are generally closer together which promotes more mixing. The swap percentages for each inverse temperature are shown in Table 3.2.

Inverse Temperature	1	.98	.94	.85	.67	.43	.22	.1
Proportion Accepted	0.9463	0.8998	0.6887	0.4512	0.3758	0.4316	0.2626	0.0355

Table 3.2: The acceptance proportion of swapping the tempered chains. Higher inverse temperatures typically had higher swapping rates.

Since our main aim is to find one suitable partition for interpretation, a good estimate of the maximum of the posterior will generally suffice. If we were to employ this model to perform some sort of model averaging, the slow mixing would certainly be a major concern.

In the simulated examples in the paper, multiple restarts of the chain resulted in the same partitions being obtained. For the windmill data, the Markov chain was run a second time with some slight changes. First, the minimum inverse temperature was set to .02 instead of .1. Second, the variance of the proposal distribution for the weights was decreased. In this second run of the chain,

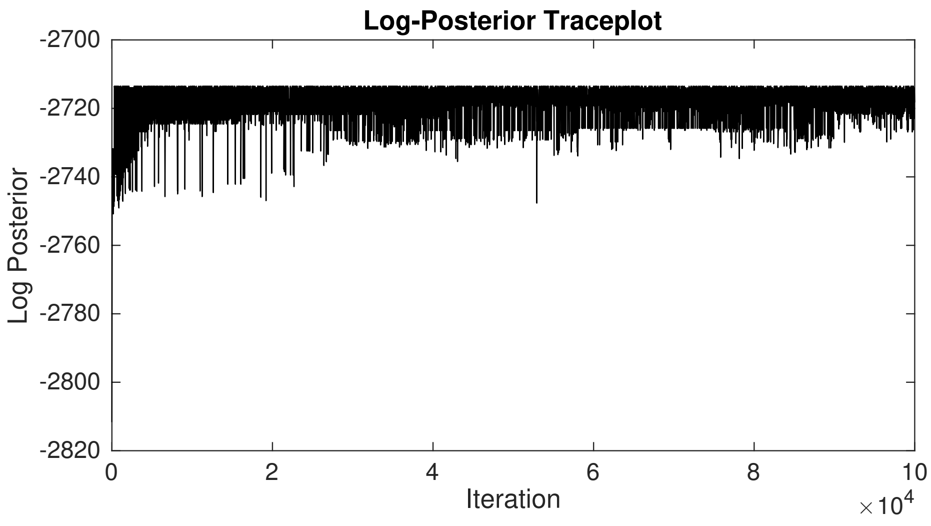

Figure 3.15: The traceplot of the log posterior for the untempered chain for the simulation in Section 3.3.2.

a slightly different partition was discovered with a higher maximum log posterior. Qualitatively, however, both runs indicated the same general result highlighting the shift in the distribution near a wind direction of 120 degrees.

For the smooth simulation in Section 3.3.4, the acceptance rates were generally better, but still poor overall. We believe these improved acceptance rates are due to the smoothness of the data generating model. Rates for birth and death steps range between 5-10% for the lower inverse-temperatures, indicating that these are doing the majority of the searching of the parameter space. Table 3.3 shows the acceptance proportions for each inverse-temperature. Figure 3.16 shows the traceplot for the untempered chain.

For the windmill data, we see similar results as the previous datasets in terms of mixing. The main difference, is that birth steps were accepted with a very high rate for all of the parallel chains. This high acceptance, with corresponding low acceptance for death steps, is due to the fact that the chains almost always had the maximum allowable number of partitions, in this case 10. Therefore, whenever a death step was accepted, the model quickly accepted a birth step to increase the number of partitions back to 10. The results are shown in Table 3.4. Note that if we look at the total number of acceptances over all the chains divided by the total number of iterations, we have an effective

Inverse Temperature	Birth	Death	Move	Change
1	0.0209	0.0214	0.0052	0.5681
.98	0.0208	0.0209	0.0051	0.5784
.94	0.0227	0.0223	0.0064	0.5866
.85	0.0216	0.0214	0.0055	0.6307
.67	0.0145	0.0147	0.0078	0.6995
.43	0.0065	0.0063	0.0156	0.7891
.22	0.0764	0.0980	0.3134	0.9197
.1	0.0303	1.0000	0.9850	0.9909

Table 3.3: Proportion of accepted moves for each inverse temperature for the simulation in Section 3.3.4

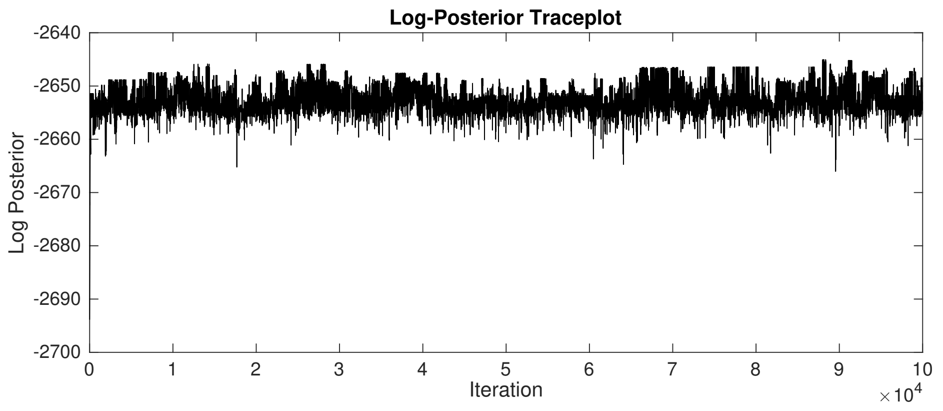

Figure 3.16: The traceplot of the log posterior for the untempered chain for the simulation in Section 3.3.4.

Inverse Temperature	Birth	Death	Move	Change
1	0.7059	0.0001	0.0008	0.0079
.98	0.4500	0.0001	0.0011	0.0063
.94	0.4412	0.0001	0.0014	0.0058
.85	0.3871	0.0001	0.0012	0.0083
.67	0.4000	0.0001	0.0014	0.0107
.43	0.3750	0.0001	0.0018	0.0171
.22	0.3750	0.0001	0.0022	0.0401
.1	0.1592	0.0015	0.0135	0.1180

Table 3.4: Proportion of accepted moves for each inverse temperature for the windmill data.

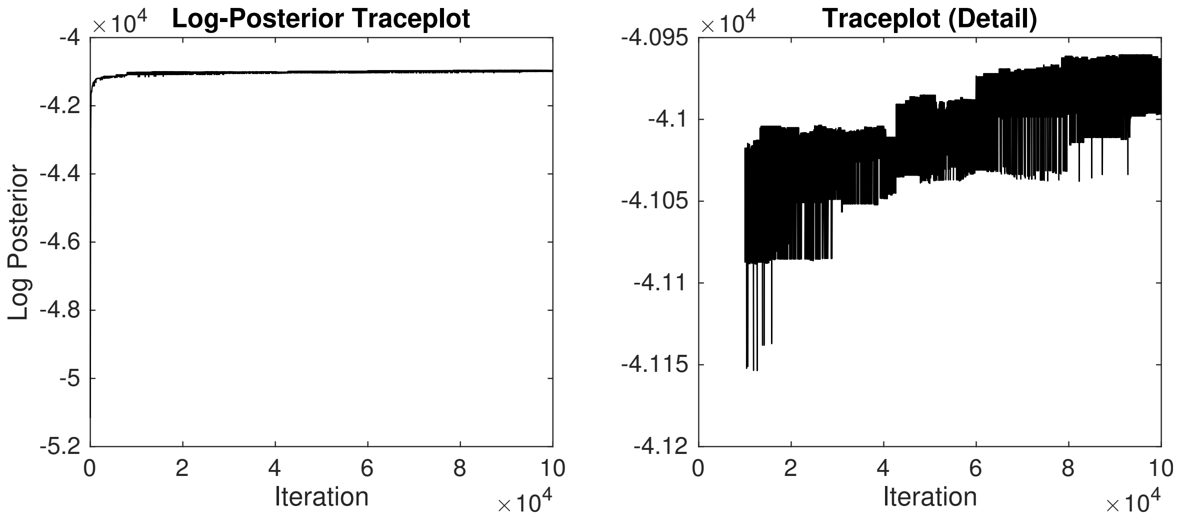

Figure 3.17: The traceplot of the log posterior for the untempered chain for the windmill data. The right panel shows the traceplot with the first 10,000 iterations omitted.

global search acceptance rate of 8%. Figure 3.17 shows the traceplot for the untempered chain.

In regard to mixing, the chain generally finds good posterior modes quickly, but then has a hard time mixing between modes. In our view, this is due to two reasons: the presence of multiple sharp modes, and the volume of the covariate space. Often when new modes are found, the new mode is substantially higher than the previous mode, making it unlikely that the previous mode will be returned to in the Metropolis-Hastings algorithm. Second, once at a new posterior mode, finding another mode with higher posterior probability can be difficult due to volume of the covariate space and a potential lack of posterior mass between the current tessellation and a tessellation with higher posterior probability.

A potential improvement to mixing may be possible adding or modifying the proposal distributions to add more recursive-type splits of existing tessellations. Indeed, something along the lines of Holmes et al. (2005) may yield better mixing. Despite the slow mixing, the partition model is still effective in finding partitions which describe how the density of y changes over the covariate space.

3.5 Competing Methods Detail

For the methods in the DPpackage in R, the following code shows the prior selections used in the code. Please see the documentation for the functions for more details regarding the specific parameterization.

For the linear dependent Bernstein polynomials (Barrientos et al., 2017), implemented in the LDBDPdensity() function, the prior structure was

```
prior <-  list(lambda=25,
          maxn=25,
          alpha=1,
          m0=rep(0,3),
          S0=diag(3)*10^3,
          nu=5,
          psiinv=diag(3)*10^3)
```

The linear dependent tail-free process method (Jara and Hanson, 2011) was implemented with the LDTFPdensity() function with prior

```
prior <-  list(maxm=6,
          a0=5,
          b0=1,
          mub=0,
          Sb=diag(3)*10ˆ 3,
          tau1=10ˆ (-4),
          tau2=10ˆ (-4))
```

The Dirichlet process mixtures of normals method (Müller et al., 1996) was implemented with the DPcdensity() function. Assume that dat is a data.frame with the realized Y in the first column and X_1 and X_2 in the second and third columns. The prior was selected as

```
w         = cov(dat)
wbar      = apply(dat,2,mean)
prior <-  list(a0=1,
          b0=.01,
          nu1=2 + ncol(dat),
          nu2 = 2 + ncol(dat),
          psiinv2=2*solve(w),
          tau1=6.01,
          tau2=2.01,
          m2=wbar,
          s2=.5*w)
```

The code for the Voronoi tessellation with a normal distribution in each partition element was obtained from the following website where the default options were used:

http://www.stat.tamu.edu/˜bmallick/wileybook/book_code.html

3.6 Discussion

Although the partition model is specifically designed for models with an abrupt change in the distribution of y over some portion of the covariate space, it is still useful in contexts where the density of y is believed to vary smoothly as a function of $\mathbf{x}$. For instance, most conditional density estimation techniques require manually exploring the conditional density by choosing various points throughout the covariate space. Even in low dimensions, an effective search of the entire covariate space is not practical due to the volume of the covariate space. Furthermore, the method of choosing the points in the covariate space is generally ad hoc and at the discretion of the modeler, allowing for the possibility of missing interesting features of the response-covariate relationship. The partition model framework searches the covariate space to determine where significant changes occur, creating a cohesive framework for understanding how the density of y is influenced by the covariates. Indeed, the partition model can be used to verify that interesting regions have not been overlooked.

An additional benefit to this partition model is the flexibility in modeling the density in each partition element. In more parametric models, including other partition models, even if the correct partition or other model structure is obtained, parameter estimates and/or densities may not be modeled well due to the rigid parametric assumptions. We saw this phenomenon demonstrated in the failure of the normal Voronoi partition model to capture bi-modality in the simulated example in Section 3.3.2. In our partition model, if for some reason the partition is incorrectly found, the flexibility in modeling the density will still reveal an accurate picture of the marginal density in that region.

Perhaps one of the greatest drawbacks of the present model is the computational burden. Although obtaining the Laplace approximation is relatively efficient using Newton's method, it still involves solving multiple systems of linear equations. This alone would not be overly prohibitive, however, this process needs to be repeated multiple times when optimizing σ and l in each partition element. Further, when making any change to the tessellation, potentially every partition element will have a different set of data, requiring the marginal to be recomputed. A more recursive partitioning strategy would limit this additional burden, and is an area for further research.

3.7 Conclusion

The combination of logistic Gaussian process density estimation with partition modeling provides an excellent framework for determining how and where the density of y changes over the covariate space. Furthermore, it has desirable consistency properties. The applications of this model will help analysts determine which variables are important in predicting the density of y, as well as better estimate the density of y over the covariate space. This flexible method will help give greater insight into datasets in which the relationship between the density of y and $\mathbf{x}$ is unknown or difficult to formulate parametrically. As part of our contribution, we provide publicly available software to perform Bayesian conditional density estimation. The code is available at `https://github.com/richardbayes/bayes-cde`.

4. A SURVIVAL TREE PARTITION MODEL USING LATENT GAUSSIAN PROCESSES AND LAPLACE APPROXIMATIONS

4.1 Introduction & Literature Review

In this chapter, we use the principles of partition modeling demonstrated in the previous chapter to construct another partition model to estimate survival functions. Latent Gaussian processes and Laplace approximations are used in a similar manner to obtain the marginal distribution so an efficient MCMC algorithm can be developed. A notable difference is the choice of partitioning, which is done through the construction of a tree model.

Consider survival times y_j, censoring indicators δ_j, and covariates x_j, $j = 1, \ldots, n$ where δ_j is 1 if y_j is observed and 0 if y_j is right censored. We assume that the data can be partitioned into M partition elements $\mathbf{y}_1, \ldots, \mathbf{y}_M$ such that $\bigcup_{i=1}^{M} \mathbf{y}_i = \mathbf{y}$ and $\mathbf{y}_i \cap \mathbf{y}_{i'} = \emptyset$ for all $i \neq i'$. We further assume that the data within a partition element are conditionally independent given the hazard function, $h_i(t)$, the hazard function of the ith partition element and assume that the M hazard functions are mutually independent. We construct the partition using a tree structure similar to that of the original Bayesian classification and regression tree papers (Chipman et al., 1998; Denison et al., 1998). The piecewise-constant hazard function at each terminal node is modeled flexibly using an exponentiated Gaussian process, and the trees are efficiently searched via MCMC by obtaining the marginal likelihood of the observed data via a Laplace approximation (Tierney and Kadane, 1986).

One of the main benefits of tree models is arguably their ease in interpretation. Indeed, there are a variety of other tree models which have been developed for use in survival and reliability models. Levine et al. (2014) present some methodology for modeling tree survival functions with correlated observations. Barrett and Coolen (2013) use an empirical Bayes approach by using Gaussian process regression on transformed survival times. Garg et al. (2011) utilize a frequentist approach to cluster hospital stays using survival trees and Loh et al. (2015) use regression trees to identify subgroups with differential treatment effects.

Tree models have also been used for prediction. Clarke and West (2008) use empirical Bayes priors and ensembles of Weibull tree models to model survival distributions. Other ensembles of trees used for prediction use various forms of Bayesian additive regression trees (BART) (Chipman et al., 2010). Bonato et al. (2010) utilize BART to estimate latent parameters in a gamma process (Kalbfleisch, 1978), Weibull, and accelerated failure time models; Henderson et al. (2017) use BART in conjunction with a mean-constrained Dirichlet process mixture model to analyze heterogeneous treatment effects, which is reminiscent of Kuo and Mallick (1997); Sparapani et al. (2016) model survival times using a BART probit regression approach. Survival, relative risk, and rotation forests have also been used to make mortality predictions (Ishwaran et al., 2004; Hothorn et al., 2005, 2006; Ishwaran et al., 2008; Zhou et al., 2015).

The majority of Bayesian tree approaches in survival analysis focus on prediction by using BART or ensembles. One purpose of the proposed methodology is to provide a Bayesian tree framework which is simple to interpret (i.e. obtaining one tree) and flexible, i.e., moves beyond the proportional hazards model (Cox, 1972). Indeed, although Gaussian processes have been utilized in modeling survival data, these methods do not move beyond the proportional hazards framework (Martino et al., 2011; Joensuu et al., 2012; Fernández et al., 2016). Our method provides the simple interpretation of a single tree while simultaneously modeling the survival function flexibly.

The paper is organized as follows: Section 4.2 describes the model, Section 4.3 details the MCMC algorithm, Section 4.4 shows some simulated examples and an application to a lung cancer dataset and Section 4.5 provides some discussion and concludes.

4.2 Model

4.2.1 Prior on Tree Partition

The tree partition is constructed through a series of recursive binary splits of the covariate space, which is explained in detail by Chipman et al. (1998). We adopt the same prior as Chipman et al. (1998) which is constructed by splitting the tree nodes with a designated probability and assigning a rule each time a node is split. Let $S_{i'}$ equal 1 if the i'th node is split, otherwise 0, $i' = 1, \ldots, 2M - 1$ where M is the number of terminal nodes on a tree. Let $R_{j'} = (v_{j'}, r_{j'})$, $j' = 1, \ldots, M - 1$ represent the rules of the $M - 1$ internal nodes. Here, $v_{j'} \in \{1, \ldots, p\}$ indicates which variable is to be split, and $r_{j'}$ is the splitting rule on the $v_{j'}$th variable. Let $T = (S_1, \ldots, S_{2M-1}, R_1, \ldots, R_{M-1})$ be the collection of split indicators and rules for a particular tree. Then the prior for a tree, T, can be evaluated as

$$p(T) = \prod_{i'=1}^{2M-1} p(S_{i'}) \prod_{j'=1}^{M-1} p(R_{j'}). \tag{4.1}$$

Following Chipman et al. (1998), we define $p(S_{i'} = 1) = 1 - P(S_{i'} = 0) = \gamma(1 + d_i)^{-\theta}$ where $d_{i'}$ represents the depth of the i'th node (e.g. the root node has a depth of 0). Here, γ and θ are hyperparameters chosen by the researcher and provide guidelines for the general shape of the tree (see Chipman et al. (1998) for details). Also, $p(R_{j'}) = 1/(m_{j'} m^{\star}_{j'})$ where $m_{j'}$ represents the number of variables on the corresponding node that are available for splitting and $m^{\star}_{j'}$ is the number of splitting rules available on the selected variable for the node to which the rule belongs.

In the present formulation, the prior is dependent upon the observed data values. That is, when assigning any node a rule, the available splitting values are taken from the data which reach that node from its ancestry. Take, for example, a simple tree that consists of the root node and two child nodes. For simplicity, suppose we observe 10 observations $\{(x_1, y_1), \ldots, (x_{10}, y_{10})\}$ where $(x_1, \ldots, x_{10}) = (1, \ldots, 10)$. If we restrict rules for continuous variables to the form $x \leq r_i$, the root node has 9 possible splitting rules which yield non-empty terminal nodes: $\{1, \ldots, 9\}$. Suppose the root node has the rule $R_0 = (1, 3)$, i.e. $x \leq 3$, then all the y_i whose x_i is less than 3 would descend to the left child node and all others would descend to the right child node. The left child node now has three data points and two possible splitting values, $\{1, 2\}$, and the right child has seven data points and six possible splitting rules $\{4, 5, 6, 7, 8, 9\}$. The tree structure easily generalizes to more than one covariate as well as categorical inputs (where the rule is a subset of available categories of the data reaching the node). To determine which rules are available on a given node, one simply looks at the subset of the data which reaches the node and determines which rules derived from the data will yield non-empty terminal nodes, or will yield terminal nodes with a minimum desired number of observations.

4.2.2 Likelihood & Prior

As before, let y_j, $j = 1, \ldots, n$ be the observed survival time where δ_j is 1 if y_j is observed and 0 if y_j is censored. Let $\mathbf{y} = \{y_1, \ldots, y_n\}$ and let $x_j \in \mathbb{R}^p$ be the observed covariate vector for observation j and $\mathbf{x} = \{x_1, \ldots, x_n\}$. Without loss of generality, we assume that $\mathbf{y}$ has been scaled to the interval $(0, 1]$. The partition model assumes that given a tree, T, the hazard functions at each terminal node are independent and the data within a terminal node, given the hazard function at the node, are independent and identically distributed. It is important to note that covariates are used to construct the tree, T, but are not used in the hazard functions. Consequently, the likelihood function can be expressed as a simple product of likelihoods

$$p(\mathbf{y} \mid T) = \prod_{i=1}^{M} p(\mathbf{y}_i \mid \cdot)$$

where $\mathbf{y}_i$ represents the n_i responses which belong to the ith terminal node, $p(\mathbf{y}_i \mid \cdot)$ represents the likelihood of $\mathbf{y}_i$ given the hazard function in the ith terminal node and M is the total number of terminal nodes.

At each terminal node, the hazard function is assumed to be piecewise constant. That is, the ith hazard function follows the form $h_i(t) = \lambda_{ik}$ for $t \in (s_{k-1}, s_k]$ with z_k denoting the midpoint of the interval. In all, there are K bins with endpoints $(s_{k-1}, s_k]$ for $k = 1, \ldots, K$ where $s_0 = 0$ and $s_K = 1$. For simplicity, we use the same grid for each hazard function and assume that the bins all have equal width, inducing a regular grid for $(z_1, \ldots, z_K)$. The likelihood for data at a single terminal node can be expressed as

$$
\begin{aligned}
p(\mathbf{y}_i \mid \lambda_{i1}, \quad \dots \quad, \lambda_{iK}) & \\
= & \prod_{j=1}^{n_i} \lambda_{ik_{ij}}^{\delta_{ij}} \exp\left\{ - \left[(y_{ij} - s_{k_{ij}-1})\lambda_{ik_{ij}} + I(k_{ij} > 1) \sum_{g=1}^{k_{ij}-1} (s_g - s_{g-1})\lambda_{ig} \right] \right\} \\
= & \exp\left\{ \sum_{j=1}^{n_i} \delta_{ij} \log(\lambda_{ik_j}) - \left[\sum_{j=1}^{n_i} (y_{ij} - s_{k_{ij}-1})\lambda_{k_{ij}} + \sum_{k=1}^{K} (s_k - s_{k-1})\lambda_{ik} m_{ik} \right] \right\} \\
= & \exp\left\{ \sum_{k=1}^{K} \left[n_{ik} \log(\lambda_{ik}) - \lambda_{ik} \left(\sum_{j:k_{ij}=k} (y_{ij} - s_{k-1}) + m_{ik}(s_k - s_{k-1}) \right) \right] \right\}
\end{aligned}
$$

where y_{ij} is the jth response from the ith terminal node, k_{ij} denotes the interval y_{ij} belongs to, m_{ik} is the number of observations of $\mathbf{y}_i$ which are greater than s_k, and n_{ik} is the number of uncensored observations of $\mathbf{y}_i$ which fall in interval k, and δ_{ij} is the censoring indicator for y_{ij}.

We model the hazard function $h_i(t) = \lambda_{ik} = \exp(f_i(z_k))$ for $t \in (s_{k-1}, s_k]$ where $f_i(t)$ is a Gaussian process with mean 0 and covariance function $C_i(t, t') = \tau_i^2 \exp\{-(t-t')/l_i\}$. The Gaussian process is evaluated at the K midpoints of the intervals $(s_{k-1}, s_k]$ and determines the value of $\lambda_{ik} = \exp\{f_i(z_k)\}$. Given a set of K bins, $\mathbf{f}_i = (f_i(z_1), \dots, f_i(z_K))^T$ follows a multivariate-normal distribution with mean 0 and covariance matrix Σ_i with the i', j'th element of Σ_i being $C_i(z_{i'}, z_{j'})$. The posterior distribution of $\mathbf{f}_i$, assuming the covariance function parameters are known, is

$$
p(\mathbf{f}_i \mid \mathbf{y}_i) \propto p(\mathbf{y}_i \mid \mathbf{f}_i) p(\mathbf{f}_i) = (2\pi)^{-K/2} |\Sigma_i|^{-.5} \exp\{g_0(\mathbf{f}_i)\}
$$

where

$$
g_0(\mathbf{f}_i) = \sum_{k=1}^{K} \left[n_{ik} f_{ik} - \exp(f_{ik}) \left(\sum_{j:k_{ij}=k} (y_{ij} - s_{k-1}) + m_k(s_k - s_{k-1}) \right) \right] - \frac{1}{2} \mathbf{f}_i^T \Sigma_i^{-1} \mathbf{f}_i
$$

and f_{ik} is the kth element of $\mathbf{f}_i$. Note that $g_0(\mathbf{f}_i)$ implicitly depends on the data, bins, and covariance function parameters. Due to the assumption of independence between partitions, the full joint posterior distribution of a tree, T, with M terminal nodes and $\mathbf{f}_1, \ldots, \mathbf{f}_M$ can be expressed as

$$p(T, \mathbf{f}_1, \ldots, \mathbf{f}_M \mid \mathbf{y}) \propto p(T) \prod_{i=1}^{M} p(\mathbf{y}_i \mid \mathbf{f}_i) p(\mathbf{f}_i)$$

where $p(T)$ is the prior on the tree as in (4.1).

The goal in many partition models is to find a suitable partition structure. In a Bayesian framework, this corresponds to searching the posterior of the tessellation, i.e.,

$$\begin{aligned} p(T \mid \mathbf{y}) &= \int \ldots \int p(T, \mathbf{f}_1, \ldots, \mathbf{f}_M \mid \mathbf{y}) d\mathbf{f}_1 \ldots d\mathbf{f}_M \\ &\propto p(T) \prod_{i=1}^{M} \int p(\mathbf{y}_i \mid \mathbf{f}_i) p(\mathbf{f}_i) d\mathbf{f}_i. \end{aligned} \tag{4.2}$$

To do this, often a model is chosen in which all node-specific parameters can be integrated out analytically. When the integrals have an analytic form, often an efficient reversible jump MCMC chain (Green, 1995) can be constructed to search the posterior partition space. Unfortunately there is no analytical solution for the integrals in (4.2) for the present model. To overcome this issue, a Laplace approximation is used to approximate each integral. Laplace approximations have been used previously in various survival model settings. Levine et al. (2014) utilize a Laplace approximation to search trees in a Bayesian frailty model and Martino et al. (2011) use integrated nested Laplace approximations via data augmentation to fit a Cox proportional hazards model to avoid MCMC. In our method, the Laplace approximation is similar to Levine et al. (2014), but we do not have random effects and we also optimize the covariance function hyperparameters via an empirical Bayes approach.

In usual form, the Laplace approximation approximates the integral with a normal distribution derived from a Taylor expansion of $\log[p(\mathbf{y}_i \mid \mathbf{f}_i)p(\mathbf{f}_i)]$. The Laplace approximation has a closed form given by

$$\int p(\mathbf{y}_i \mid \mathbf{f}_i) p(\mathbf{f}_i) d\mathbf{f}_i \approx |D_i + \Sigma_i^{-1}|^{\frac{1}{2}} |\Sigma_i|^{-\frac{1}{2}} \exp\left\{ g_0(\hat{\mathbf{f}}_i) \right\} \tag{4.3}$$

where D_i is a diagonal matrix with j'th element $\exp(f_{ij'}) \left[\sum_{j:k_{ij}=j'} (y_{ij} - s_{j'-1}) + (s_{j'} - s_{j'-1}) m_{j'} \right]$

and $\hat{\mathbf{f}}_i = \arg\max_{\mathbf{f}_i} p(\mathbf{y}_i \mid \mathbf{f}_i)p(\mathbf{f}_i)$, which can be obtained using Newton's method.

Thus far we have assumed that the covariance function parameters are fixed and known. In practice, we employ an empirical Bayes approach to set these values in each terminal node. Specifically, we choose τ_i and l_i to maximize $p(\mathbf{y}_i \mid \mathbf{f}_i)p(\mathbf{f}_i)\eta(l_i)\eta(\tau_i/10)$ where $\eta(\cdot)$ is the density of a t-distribution with one degree of freedom.

At this point, we have a way to approximate $p(T \mid \mathbf{y})$ by utilizing Laplace approximations. We now need to determine a way to search the approximate posterior. As is common in many tree models such as Chipman et al. (1998) and Gramacy and Lee (2008), we employ Markov chain Monte Carlo methods to search the posterior. Due to the varying number of parameters in the prior of the tree structure, a reversible jump MCMC algorithm (Green, 1995) needs to be constructed, and is discussed in the next section.

4.3 Markov Chain Monte Carlo

Since the dimension of the tree parameter space is variable, we use a reversible jump MCMC algorithm to search the posterior parameter space (Green, 1995). The algorithm is based on both Chipman et al. (1998) and Gramacy and Lee (2008) with a few modifications. The primary goal in constructing the algorithm is to obtain draws from $\tilde{p}(T \mid \mathbf{y})$, the approximate posterior obtained using the Laplace approximations. This is accomplished in the reversible jump MCMC algorithm by modifying an existing tree with one of four possible moves: grow, prune, change, and swap. These four steps are now described.

- **Grow:** The grow step consists of randomly selecting a terminal node and splitting it using a split rule. The split rule is selected by first randomly selecting a variable and then randomly selecting an available rule of that variable (i.e. the rule is drawn from the prior). This increases the number of terminal nodes of the tree by one, thereby “growing" the tree.

- **Prune:** The prune step is the opposite of the grow step. Randomly select a node whose children are both terminal nodes and delete the split rule and the children nodes. This reduces the number of terminal nodes of the tree by one, thereby “pruning" the tree.

```
Algorithm 2. Reversible Jump Markov Chain Monte Carlo Algorithm
Initialize Tree;
for $i \leftarrow 1$ to $N$ do
    Propose a grow, prune, change, or swap step to the current tree;
    if grow step then
        Split a random terminal node by randomly choosing a split rule;
    else if prune step then
        Combine two terminal nodes by randomly selecting an interior node whose children
          are both terminal nodes and removing the rule and child nodes;
    else if change step then
        Randomly change a rule from an existing interior node;
    else if swap step then
        Randomly choose a parent-child pair and swap their rules;
    end
    Accept the proposed change with probability $\alpha$;
end
Discard burn-in period;
```

Figure 4.1: Reversible jump Markov chain Monte Carlo algorithm for the survival tree model. The MCMC employs grow, prune, change, and swap steps.

- **Change:** The change step randomly selects an interior node and changes the existing split rule. If the current rule is based on a categorical variable, a new rule is drawn randomly from the prior by first randomly selecting a variable, and then randomly selecting a rule from that variable. If the current rule is based on a numeric variable, there are two possible changes that can be made: 1) Move the current rule to the next largest or smallest rule available for the current variable, 2) Draw a new rule from the prior. The first change is made with a certain probability chosen by the user (i.e. this is a tuning parameter), otherwise a new rule is drawn from the prior.

- **Swap:** The swap step involves randomly choosing a parent and child pair which are both internal nodes, and then swapping their rules. If, however, the parent and child's rules are based on the same continuous variable, a rotate step is employed (see Gramacy and Lee (2008)). If both children of the parent node have the same rule, both children swap their rules with the parent node.

Algorithm 2 displayed in Figure 4.1 provides pseudo-code for the MCMC algorithm. More specifically, the MCMC algorithm proceeds as follows:

1. Initialize a starting tree, T.
2. Propose a new tree, $T^\star$, from the current tree using a grow, prune, change, or swap step (as appropriate based on current tree).
3. Accept $T^\star$ with probability

   $$\alpha = \min\left\{1, \frac{q(T \mid T^\star)\tilde{p}(T^\star \mid \mathbf{y})}{q(T^\star \mid T)\tilde{p}(T \mid \mathbf{y})}\right\} \tag{4.4}$$

   where $q(\cdot \mid \cdot)$ is the proposal distribution and $\tilde{p}(T \mid \mathbf{y})$ is the approximate posterior of T obtained from the Laplace approximation; otherwise keep the current tree, T.
4. Repeat steps 2-3 until the stationary distribution is reached and the desired number of samples is obtained.

In practice we replace the approximate posterior of the tessellation in (4.4) with the likelihood-prior product in (4.3) since the normalizing constants cancel. It is important to note that the form of $q(\cdot, \cdot)$ is different with each of the four possible tree modifications. These details are discussed next.

4.3.1 Markov Chain Monte Carlo Details

We now proceed by describing the acceptance ratio of the MCMC algorithm in more detail. As shown in (4.4), the acceptance proposal for a proposed tree is

$$\alpha = \min\left\{1, \frac{q(T \mid T^\star)\tilde{p}(\mathbf{y} \mid T^\star)p(T^\star)}{q(T^\star \mid T)\tilde{p}(\mathbf{y} \mid T)p(T)}\right\}$$

where $q(\cdot \mid \cdot)$ is the proposal distribution from which $T^\star$ was generated and $\tilde{p}(\mathbf{y} \mid T)$ is the approximate marginal likelihood. The prior evaluation, $P(T)$, is given in (4.1). We now provide additional details regarding the evaluation of $q(\cdot \mid \cdot)$, which differ depending on which type of tree modification is proposed (i.e. grow, prune, change, swap).

4.3.1.1 Grow

The proposal probability for a grow step, $q_g(T^\star \mid T)$, is $p_g(n_g m_i m^\star_{ij})^{-1}$ where p_g is the probability of proposing a grow step, n_g is the number of nodes on the current tree T that are capable of a grow step, m_i is the number of variables that have valid split rules on the ith node which was selected for a grow step, and $m^\star_{ij}$ is the number of split rules available on the jth variable on the ith node selected as the split variable.

4.3.1.2 Prune

The proposal probability for a prune step, $q_p(T^\star \mid T)$, is $p_p(n_d)^{-1}$ where p_p is the probability of proposing a prune step and n_d are the number of interior nodes for which a prune step is possible (i.e. parent nodes which have both children as terminal nodes).

4.3.1.3 Change

The proposal probability of a change step, $q_c(T^\star \mid T)$ is $p_c n_c^{-1} p_{\text{prior}}(m_i m^\star_{ij})^{-1}$ where p_c is the probability of proposing a change step, n_c is the number of interior nodes eligible for a change step (which is all interior nodes if one includes the trivial change step of keeping the same rule if there is only one rule available to a particular node), and p_{prior} is the probability of drawing a rule from the prior. Note that if the current rule is based on a categorical variable, $p_{\text{prior}} = 1$ and m_i, $m^\star_{ij}$ are the same as those found in $q_g(\cdot, \cdot)$. If however, the current rule is a continuous variable, $p_{\text{prior}} \in [0, 1]$ is the value pre-specified by the user. If the new rule is proposed by the prior, m_i, $m^\star_{ij}$ are the same as those found in $q_g(\cdot, \cdot)$. If, however, the new rule is chosen by selecting the next largest or smallest rule at random from the current continuous variable on which the current rule is based, then $m_i = 1$ and $m^\star_{ij} = 2$ unless the current rule is on the boundary of available rules or is the only rule available, then $m^\star_{ij} = 1$.

4.3.1.4 Swap

The proposal probability of a swap step, $q_s(T^\star \mid T)$ is $p_s n_s^{-1}$ where p_s is the probability of proposing a swap step and n_s is the number of possible swap steps available on the current tree, T.

Prior to running the reversible jump MCMC algorithm, the user specifies $p_g,\ p_p,\ p_c$ and p_s. However, on a given MCMC iteration, these values may need to be adjusted. For instance, if one starts with the root node, the only possible proposal is a grow step, therefore $p_g = 1$ and $p_p = p_c = p_s = 0$. Other examples (not necessarily an exhaustive list) include the fact that a swap step is not possible with fewer than 3 terminal nodes, and birth steps may not be possible if the tree grows large enough that splitting any terminal node would result in a terminal node with fewer than the minimum number of required observations required by the user. Therefore, on each iteration of the MCMC chain, one must check to determine which proposals are actually available and adjust the probabilities accordingly. This is also the case when calculating the probability of returning to the current tree from the proposed tree, $q(T \mid T^\star)$.

4.3.2 Parallel Tempering

From their inception, Bayesian tree models have been notorious for their difficulty in mixing when using MCMC procedures. Various methods have been proposed to overcome this issue including restarting the MCMC chain multiple times (Chipman et al., 1998), parallel tempering (Gramacy and Taddy, 2010), and multiset sampling (Leman et al., 2009).

We implement parallel tempering (Geyer, 1991) modeled after Gramacy and Taddy (2010) to search the multi-modal posterior distribution and effectively search the parameter space. The parallel tempering algorithm was described in Section 3.2.4, and the same approach is used here.

4.3.3 Computational Considerations

Although the Laplace approximation reduces the dimension of the parameter space which needs to be explored via MCMC methods, there are still a number of computational challenges. The main computational cost in the algorithm comes from optimizing the covariance function parameters, l_i and τ_i, in each terminal node. Optimizing these parameters requires finding $\hat{\mathbf{f}}_i$ for each evaluated pair which involves solving multiple linear systems (Newton's method) and a determinant.

Fortunately, the choice of the covariance function for $f_i(\cdot)$ along with a regular grid imply that we have a sparse closed-form solution for

$$\Sigma_i^{-1} = \frac{1}{\tau_i^2(1-\rho_i^2)} \begin{pmatrix} 1 & -\rho_i & & & \\ -\rho_i & 1+\rho_i^2 & -\rho_i & & \\ & \ddots & \ddots & \ddots & \\ & & -\rho_i & 1+\rho_i^2 & -\rho_i \\ & & & -\rho_i & 1 \end{pmatrix}$$

where $\rho_i = \exp(-\Delta z/l_i)$ where Δz is the distance between points of the regular grid. This closed form implies that we never need to form the dense matrix Σ_i, and the matrices in (4.3) can be stored in a sparse matrix form, reducing memory and computation demands.

Lastly, although we use the same bins for each hazard function, we use a subset of the bins in each terminal node since the ranges of observed survival times will differ across partition elements. This reduces the computation burden as well as avoids extrapolating beyond the terminal node's data. This is important since the partition model does not borrow information from other partition elements. For each terminal node, we use the first $k_i^\star$ bins where $k_i^\star = \{k' : s_{k'-1} < \max(\mathbf{y}_i) \leq s_{k'}\}$.

4.4 Simulations & Applications

4.4.1 Preliminaries

For the following analyses, we use the same set of tuning parameters for the MCMC chain. We set the probability of grow, prune, change, and swap steps to be .25. For the change step for a continuous variable, we choose to move the rule to an adjacent rule with probability .75, otherwise a rule is drawn from the prior. We run each analysis with 8 parallel chains with the minimum inverse-temperature of .1, and propose a trade between the parallel chains every 10

MCMC iterations. Last, we require trees to have a minimum of 25 observations at a terminal node, and set the number of bins for the hazard function to be $K = 100$.

4.4.2 Non-Proportional Hazards

Our first simulation demonstrates the flexibility and resulting interpretability of the method. One thousand data points were simulated from the following model:

$$\begin{aligned} X_1 &\sim \text{Uniform}(0, 10) \\ X_2 &\sim \text{Multinomial(A,B,C,D; } .25, .25, .25, .25) \\ Y &\sim \begin{cases} \text{Weibull}(5, 2) & \text{if } X_1 > 5 \text{ and } X_2 \in \{\text{A,B}\} \\ \text{Weibull}(1, 5) & \text{if } X_1 \leq 5 \text{ and } X_2 \in \{\text{A}\} \\ \text{Weibull}(.5, .9) & \text{if } X_1 \leq 5 \text{ and } X_2 \in \{\text{B}\} \\ \text{Weibull}(5, 5) & \text{if } X_1 \leq 3 \text{ and } X_2 \in \{\text{C,D}\} \\ \text{Weibull}(.5, .5) & \text{if } 3 < X_1 \leq 7 \text{ and } X_2 \in \{\text{C,D}\} \\ G & \text{if } X_1 > 7 \text{ and } X_2 \in \{\text{C,D}\} \end{cases} \end{aligned} \tag{4.5}$$

where G is a random variable drawn from a piecewise linear survival function, shown in the bottom right panel in Figure 4.2. The data were also right censored with approximately a 5% censoring rate. The Markov chain Monte Carlo algorithm ran for 10^5 iterations. The tree with the highest posterior probability from the chain is shown in Figure 4.3, and shows that the desired tree structure was recovered. Note that since we are using observed data values to choose the rules at each node, the rules for numeric variables will not be exactly the same as those used to generate the tree. Figure 4.2 shows the posterior survival curves plotted with the true survival functions at each terminal node.

This example highlights the strength of this particular method, simultaneously producing easy interpretation and flexibility. Interpretation is simple since only one tree is needed, but the the various shapes of the survival curves can still be modeled flexibly at each terminal node.

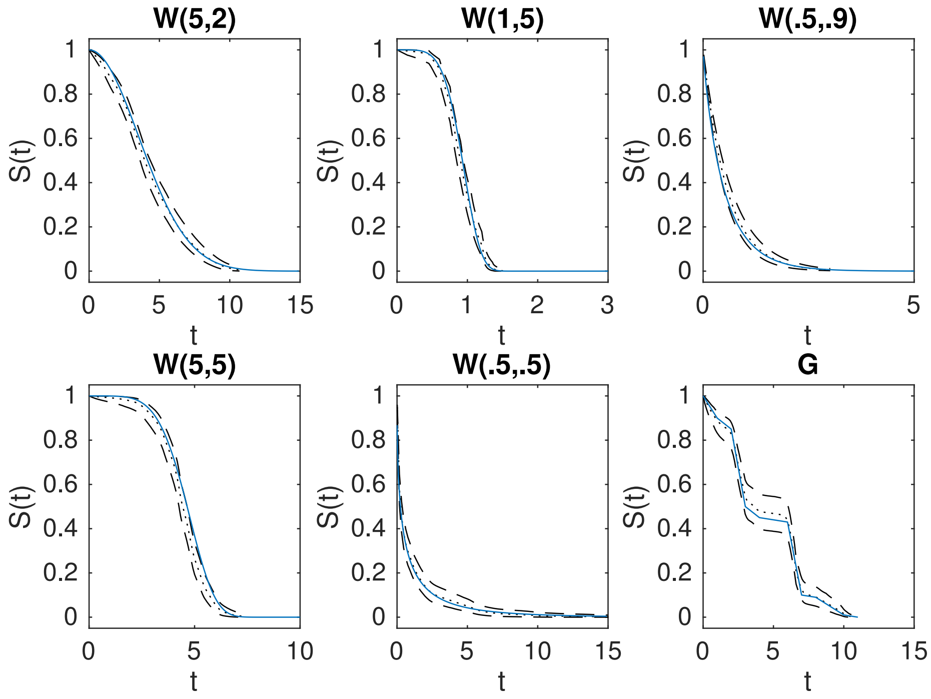

Figure 4.2: The posterior mean (dotted), 95% credible intervals (dashed), and the true survival curve (solid, blue) in each partition element. The titles indicate the true survival distribution with W denoting a Weibull distribution and G the piecewise linear survival curve in Simulation 1. The model is successful at modeling the varying shapes in this non-proportional hazards context.

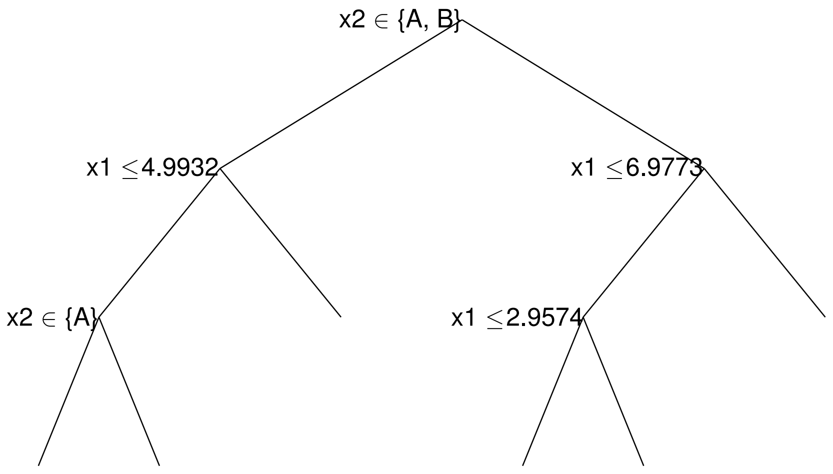

Figure 4.3: The tree with the highest posterior probability from the non-proportional hazards simulation. If data meet the criterion at a node, they descend to the left child node; otherwise, they descend to the right child node. The model successfully captures the partition structure.

4.4.3 Piecewise-Constant Proportional Hazards

Another simulation using a similar partition structure as (4.5) was performed with various exponential distributions. This is equivalent to a proportional hazards model where the hazard function is piecewise constant (rather than linear) in the covariate space. One thousand data points were generated from the following model with a censoring rate of 7.8%.

$$\begin{aligned}
X_1 &\sim \text{Uniform}(0, 10) \\
X_2 &\sim \text{Multinomial}(\text{A,B,C,D};\ .25, .25, .25, .25) \\
Y &\sim \begin{cases} \text{Exp}(1) & \text{if } X_1 > 5 \text{ and } X_2 \in \{\text{A,B}\} \\ \text{Exp}(10) & \text{if } X_1 \leq 5 \text{ and } X_2 \in \{\text{A,B}\} \\ \text{Exp}(.5) & \text{if } X_1 \leq 3 \text{ and } X_2 \in \{\text{C,D}\} \\ \text{Exp}(3) & \text{if } 3 < X_1 \leq 7 \text{ and } X_2 \in \{\text{C,D}\} \\ \text{Exp}(10) & \text{if } X_1 > 7 \text{ and } X_2 \in \{\text{C,D}\} \end{cases}
\end{aligned} \tag{4.6}$$

where Exp(μ) is an exponential distribution with mean μ. After 10^5 iterations of the MCMC chain, the tree with the highest posterior probability obtained the correct partition and the resulting survival curves fell within the 95% credible regions. For brevity, the tree and survival functions are not pictured.

Another proportional hazards model was constructed by simulating data from two hazard functions proportional to the hazard function of G in (4.5) with a 5.3% censoring rate. Only one covariate was used, $X_1 \sim \text{Uniform}(0, 1)$, and the covariate space was partitioned based on X_1 being either greater than or less than .5. The MCMC chain was run for 10^5 iterations, and the tree with the highest posterior probability was obtained. The tree misclassified only two points out of 1,000 simulated values. Figure 4.4 compares the posterior and true survival functions. The model successfully captures these two non-standard survival curves.

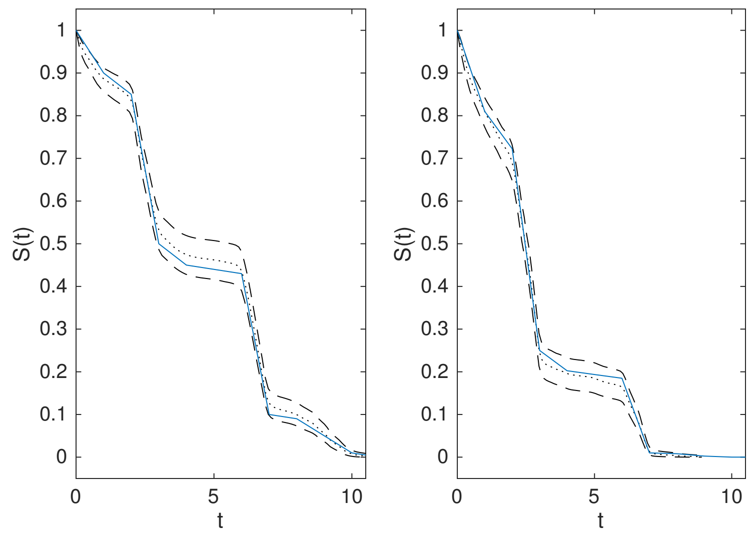

Figure 4.4: The posterior mean (dotted), 95% credible intervals (dashed), and the true survival curve (solid, blue) in each partition element. The model is successful at modeling the proportional-hazard non-parametric survival functions.

4.4.4 Linear Proportional Hazards

Although the present partition model assumes that survival functions are piecewise-constant in the covariate space, it still does a good job when this assumption does not hold, e.g., when dealing with the classic linear proportional hazards model (Cox, 1972). One thousand observations were simulated from an exponential proportional hazards model with approximately a 5% censoring rate. The model is

$$\begin{aligned} h(t \mid \mathbf{X}_j = \mathbf{x}_j) &= \frac{1}{2}\exp\{\mathbf{x}_j\beta\} \\ \mathbf{X}_j &= (X_1, \ldots, X_6) \\ X_1, \ldots, X_5 &\sim \text{Uniform(0,1)} \\ X_6 &\sim \text{Bernoulli(.5)} \\ \beta &= (-1, 1, 2, 0, 0, -2)^T. \end{aligned} \tag{4.7}$$

where $h(\cdot \mid \cdot)$ represents the hazard function.

Note that we have two spurious covariates (X_4, X_5) which do not influence the hazard function. The MCMC algorithm ran for 10^5 iterations and the tree with the highest posterior probability is

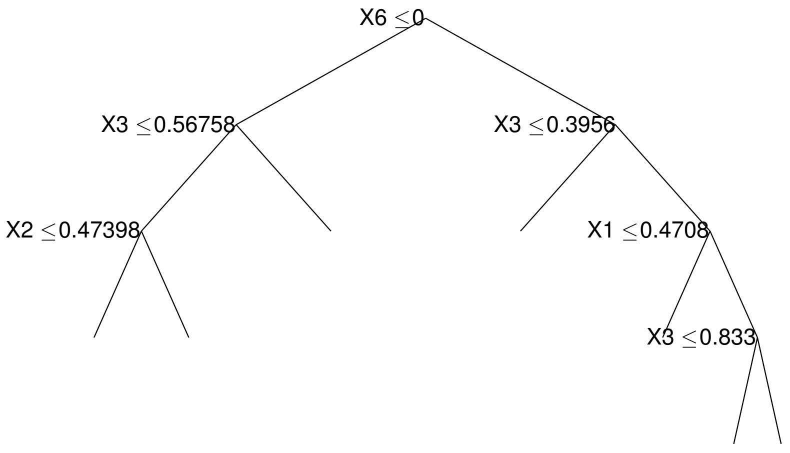

Figure 4.5: The tree with highest posterior probability from the linear proportional-hazards model.

shown in Figure 4.5. Note that neither of the two spurious covariates were included in the tree. To give a rough measure of performance, Figure 4.6 plots the posterior survival curve and the survival curve using the median hazard value of the data that reach the terminal node. Also plotted are the corresponding Kaplan-Meier curves, which indicate that the model does an excellent job at non-parametrically modeling the marginal survival functions in each partition element.

4.4.5 Sample Size and Censoring Rate Analysis

In all the above simulated examples, 9 different simulations were completed using a combination of 3 different sample sizes ($n = 1000, 500, 100$) and 3 different censoring rates (roughly $5\%, 10\%,$ and $40 - 70\%$ censoring). The results of the simulations presented previously correspond to the largest sample size of 1,000 with the lowest censoring rate.

Two methods for choosing trees include finding the tree with the highest posterior value, equivalent to finding the tree with the largest $p(\mathbf{y} \mid T)p(T)$, or the tree with the highest marginal likelihood value, $p(\mathbf{y} \mid T)$. As noted by Chipman et al. (1998), the present prior on the tree will generally favor categorical splits unfairly. One balanced approach is to plot the best marginal likelihood for each tree size and then determine at what point increasing the tree size no longer yields a notable increase in the likelihood. For simplicity in the simulations in this paper, we have chosen trees with the highest posterior likelihood, but we highly recommend looking at the best trees in

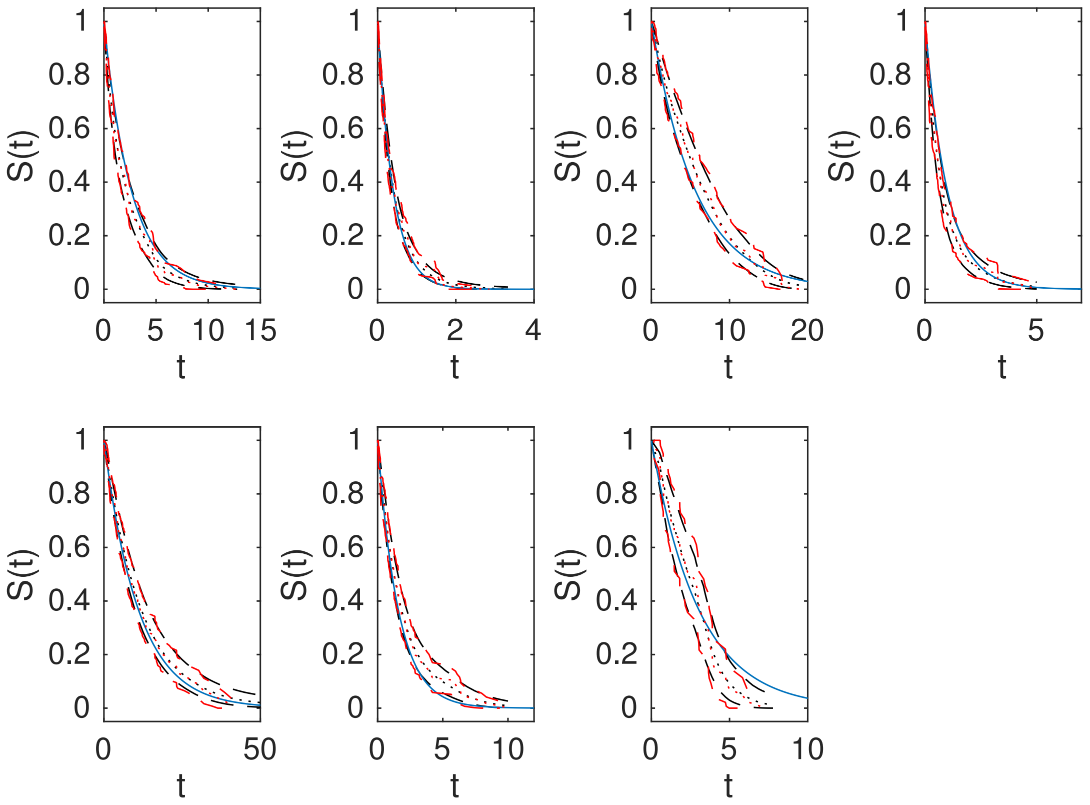

Figure 4.6: The posterior mean (dotted) and 95% credible intervals (dashed) as well as the Kaplan-Meier survival curve (red, dotted) and 95% confidence intervals (red,dashed). The survival curve corresponding to the median hazard function of the data in each terminal node is also plotted (solid, blue) to give a rough measure as to where the hazard functions in the covariate region lie.

several tree sizes.

Table 4.1 shows the number of terminal nodes found for the simulated models presented in sections 4.4.2-4.4.4 for various sample sizes and censoring rates. The table shows tree sizes based on the largest posterior values. In general, the method performs well with large sample sizes ($n = 1000$), even in the presence of moderate censoring. By "performing well", we mean that the model identifies appropriate partitions and models the data well in each partition element. With a moderate sample size ($n = 500$), the model continues to perform well when the survival functions follow Weibull or exponential distributions, but has more trouble identifying the partition when the data are drawn from an unusual survival function, as in Figure 4.4. This is likely due to the fact that the prior is centered around an exponential survival function, and the priors give slight preference to survival curves with smaller τ^2 and l values in the covariance function. For smaller sample sizes ($n = 100$), fewer partitions are obtained due in part to the fact that we required terminal nodes to have at least 25 data points so that survival curves can be modeled adequately.

It should be noted that when there is a high degree of censoring (i.e. $> 50\%$) that occasionally the estimated survival curve at a terminal node can become overly precise. This is due to the fact that the marginal likelihood is maximized when τ^2 is very small, forcing a very precise exponential survival curve. A careful researcher, however, will notice these issues, and choose a different posterior partition where this is not an issue. One way to diagnose this particular issue is to compare the posterior survival curves to the Kaplan-Meier curves (Kaplan and Meier, 1958) to determine if there are any major discrepancies.

4.4.6 Lung Cancer Survival

The partition model was applied to a lung cancer dataset which contains protein expression levels for 9 proteins which are believed to be associated with mortality. The dataset records the survival times, of which we have a high censoring rate of 72%. The partition model was run for 10^5 iterations. In this case, the log-posterior indicated that no partitioning should be done. However, our experience with the model leads us to believe that this is due to the high rate of censoring which decreases the likelihood contribution. The marginal likelihood, $p(\mathbf{y} \mid T)$, had

Model	n	Censoring	Terminal Nodes (log-posterior)	Terminal Nodes
Piecewise-Constant, Non-PH (Section 4.4.2)	1000	5.5%	6	6
	1000	11.2%	7	6
	1000	37.5%	6	6
	500	7.4%	6	6
	500	8.6%	6	6
	500	39.4%	7	6
	100	5.0%	3	6
	100	8.0%	3	6
	100	28.0%	4	6
Piecewise-Constant, PH, Exponential (Section 4.4.3)	1000	7.8%	5	5
	1000	16.2%	5	5
	1000	70.1%	5	5
	500	7.8%	5	5
	500	14.4%	5	5
	500	68.0%	6	5
	100	5.0%	3	5
	100	19.0%	3	5
	100	68.0%	3	5
Piecewise-Constant, PH, G-dist (Section 4.4.3)	1000	5.3%	2	2
	1000	13.8%	2	2
	1000	45.0%	2	2
	500	7.4%	1	2
	500	11.4%	2	2
	500	46.0%	1	2
	100	9.0%	1	2
	100	16.0%	1	2
	100	47.0%	1	2
Linear PH (Section 4.4.4)	1000	6.6%	7	N/A
	1000	13.7%	7	N/A
	1000	42.1%	6	N/A
	500	6.6%	6	N/A
	500	11.6%	5	N/A
	500	48.4%	5	N/A
	100	4.0%	2	N/A
	100	10.0%	3	N/A
	100	45.0%	3	N/A

Table 4.1: The number of terminal nodes found for the simulated models (third column) for various sample sizes n and censoring rates. The true number of terminal nodes is shown in the last column, where applicable.

consistent improvements until the tree reached 10 terminal nodes. However, in these larger trees, the optimized hyperparameters occasionally favored the prior too strongly by selecting very small values for τ^2. Therefore, a the tree with the highest marginal likelihood with 5 terminal nodes was selected since the optimization of the hyperparameters $(\tau,\ l)$ in each partition element was not suspect.

Figure 4.7 shows the selected tree, 95% credible intervals and posterior mean for the survival function at each terminal node as well as the Kaplan-Meier survival curves (Kaplan and Meier, 1958) for comparison. The tree suggests several different survival curves of various shapes including groups with relatively short survival times, and groups which tend to survive longer. For instance, the tree suggests that for subjects with low expression levels of X_6 and X_1 have relatively short survival times and higher levels of X_6 and X_7 generally lead to longer survival times, etc. The closeness of the model results with that of the Kaplan-Meier curves indicate that the model is sufficiently flexible to model the survival curves nonparameterically while simultaneously providing an interpretable partition of the covariate space. This closeness also indicates that there aren't any major concerns in regard to the optimization procedure.

4.4.7 Markov Chain Monte Carlo Mixing

Generally we find that mixing of the MCMC chain is satisfactory. In the partition models, typically there is a lower acceptance rate on the untempered chain since the model has sharp edges which make it difficult to make many modifications to the tree. In the models which vary smoothly through the covariate space, mixing is generally better, averaging about 20-40% acceptance rate overall across all moves. The parallel tempering generally swaps chains more frequently for the higher inverse-temperatures, and allows for better mixing of the overall chain.

4.5 Discussion & Conclusion

The main advantages of the partition model are its ease in interpretation coupled with its flexibility. Inference is easily obtained from inspecting trees and the resulting survival curves at the terminal nodes. In both simulations and applications, the model does well in obtaining the correct

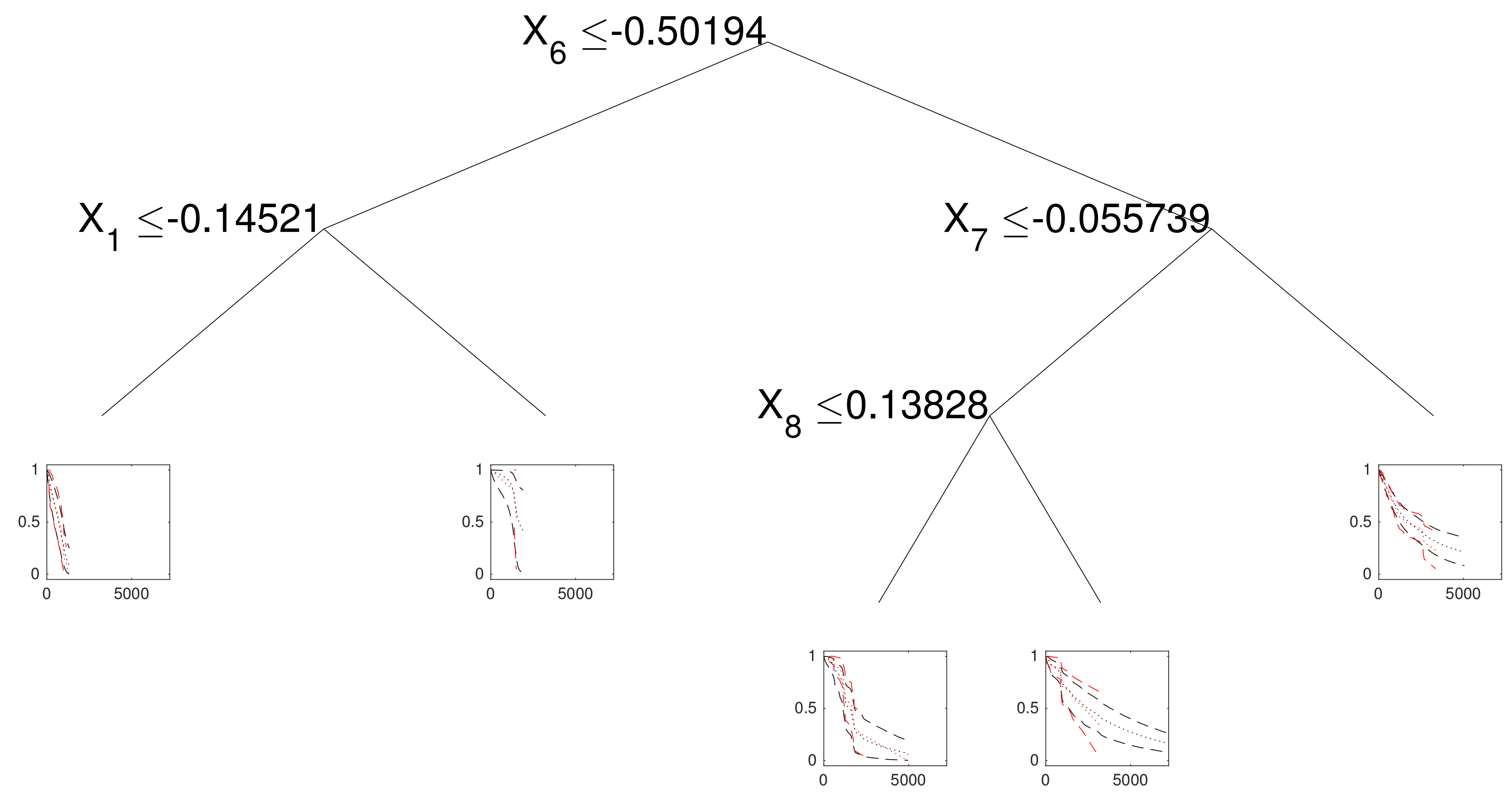

Figure 4.7: The selected posterior tree, posterior means (black, dotted), 95% credible intervals (black, dashed) and Kaplan-Meier curves with their 95% confidence intervals (red, dotted; red, dashed).

partition structure (when data are generated from a partition model) and still provides adequate modeling and interpretation in models where the survival function changes smoothly throughout the covariate space. Indeed, the model is flexible enough to be very close to the nonparametric Kaplan-Meier curves in the lung cancer data analyzed in this paper.

The main drawback of the present method is the computation time. For instance, the lung cancer data took approximately 12 hours to obtain 10^5 iterations. Even so, the tuning of the chain is extremely easy since node-specific parameters are marginalized through the Laplace approximation, and the only tuning parameters are related to modifying the tree and the parallel tempering. In our experience, the results are robust to these tuning parameters, therefore the method is almost fully automatic. To reduce computation time, one can reduce the number of bins, K, but this will lead to a coarser estimation of the hazard function.

The only other issue we've experienced is that of occasional issues when optimizing the covariance function parameters τ and l where τ is selected to be extremely small which does not allow the model to flexibly fit the survival function. This has usually happened only when there is a large amount of censored data, and the observed deaths happen early. Fortunately, as far as we can tell, these poor optimizations are quite obvious to detect since they result in extremely narrow credible intervals around an exponential survival function. Detecting potential optimization issues can be easily done by fitting Kaplan-Meier curves for comparison and looking for large differences in the estimated survival curves.

Overall, this method shows great potential for easy interpretation as well as flexible modeling. Extending this method to perform variable selection in high dimensions as well as potential tweaks to the prior to help the model perform better in high-censoring situations are areas for further research.

5. SUMMARY

In this dissertation, we have explored the construction of efficient Markov chains through modifying existing algorithms, such as the Metropolis-Hasting sampler, and searching approximate posteriors of partition models efficiently by marginalizing parameters using Laplace approximations. The methodologies are important to the field of Bayesian statistics since performing full Markov chain Monte Carlo methods on these increasingly complex models is often proving to be inadequate.

As the field of Bayesian statistics continues to move forward, researchers will need to allow computation considerations to play a larger role in the development of methodology. Such computational considerations formed the basis for many decisions in the present work, and without making decisions based largely on these computational considerations, the computational demand would have likely yielded intractable algorithms.

A common theme in this dissertation has been the balance between prediction and inference. There are many machine learning algorithms which can make predictions with a high degree of accuracy. However, these methods often lack interpretability. Conversely, there are many statistical methods which are simple to interpret, but are not flexible enough to model the data accurately enough. An overarching goal in this dissertation was to find a compromise between these two extremes, and we believe we have been successful.

No doubt there will be giant leaps forward in the world of computation and algorithm development which will allow the creation and application of statistical methods not yet imagined. Between these giant leaps, however, history has consistently shown that progress is made incrementally. This dissertation is one of many of these incremental steps which are necessary to move the field of statistics forward and we are excited to contribute and watch this ubiquitous field develop further in the coming years.

REFERENCES

Andrieu, C. and G. O. Roberts (2009). The pseudo-marginal approach for efficient Monte Carlo computations. *The Annals of Statistics 37*(2), 697–725.

Bardenet, R., A. Doucet, and C. Holmes (2014). Towards scaling up Markov chain Monte Carlo: an adaptive subsampling approach. In *Proceedings of the 31st International Conference on Machine Learning*, Volume 32, pp. 405–413.

Bardenet, R., A. Doucet, and C. Holmes (2015). On Markov chain Monte Carlo methods for tall data. *arXiv preprint arXiv:1505.02827*.

Barrett, J. E. and A. C. Coolen (2013). Gaussian process regression for survival data with competing risks. *arXiv preprint arXiv:1312.1591*.

Barrientos, A. F., A. Jara, and F. A. Quintana (2017). Fully nonparametric regression for bounded data using dependent Bernstein polynomials. *Journal of the American Statistical Association 112*(518), 806–825.

Bhattacharya, A. and D. B. Dunson (2010). Nonparametric Bayesian density estimation on manifolds with applications to planar shapes. *Biometrika 97*(4), 851–865.

Bonato, V., V. Baladandayuthapani, B. M. Broom, E. P. Sulman, K. D. Aldape, and K. Do (2010). Bayesian ensemble methods for survival prediction in gene expression data. *Bioinformatics 27*(3), 359–367.

Chipman, H. A., E. I. George, and R. E. McCulloch (1998). Bayesian CART model search. *Journal of the American Statistical Association 93*(443), 935–948.

Chipman, H. A., E. I. George, R. E. McCulloch, et al. (2010). BART: Bayesian additive regression trees. *The Annals of Applied Statistics 4*(1), 266–298.

Christen, J. A. and C. Fox (2005). Markov chain Monte Carlo using an approximation. *Journal of Computational and Graphical Statistics 14*(4), 795–810.

Chung, Y. and D. B. Dunson (2009). Nonparametric Bayes conditional distribution modeling with variable selection. *Journal of the American Statistical Association 104*(488), 1646–1660.

Clarke, J. and M. West (2008). Bayesian Weibull tree models for survival analysis of clinico-genomic data. *Statistical Methodology 5*(3), 238–262.

Cox, D. (1972). Regression models and life-tables. *Journal of the Royal Statistical Society. Series B (Methodological) 34*(2), 87–22.

Denison, D., N. Adams, C. Holmes, and D. Hand (2002). Bayesian partition modelling. *Computational Statistics & Data Analysis 38*(4), 475–485.

Denison, D. and C. Holmes (2001). Bayesian partitioning for estimating disease risk. *Biometrics 57*(1), 143–149.

Denison, D. G., C. Holmes, B. Mallick, and A. Smith (2002). *Bayesian methods for nonlinear classification and regression*. Chichester, United Kingdom: John Wiley & Sons.

Denison, D. G., B. K. Mallick, and A. F. Smith (1998). A Bayesian CART algorithm. *Biometrika 85*(2), 363–377.

Dunson, D. B. and J.-H. Park (2008). Kernel stick-breaking processes. *Biometrika 95*(2), 307–323.

Dunson, D. B., N. Pillai, and J. Park (2007). Bayesian density regression. *Journal of the Royal Statistical Society: Series B (Statistical Methodology) 69*(2), 163–183.

Fan, J., Q. Yao, and H. Tong (1996). Estimation of conditional densities and sensitivity measures in nonlinear dynamical systems. *Biometrika 83*(1), 189–206.

Fernández, T., N. Rivera, and Y. W. Teh (2016). Gaussian processes for survival analysis. In *Advances in Neural Information Processing Systems*, pp. 5021–5029.

Friedman, J. H. (1991). Multivariate adaptive regression splines. *The Annals of Statistics 19*(1), 1–67.

Fu, G., F. Y. Shih, and H. Wang (2011). A kernel-based parametric method for conditional density estimation. *Pattern Recognition 44*(2), 284–294.

Garg, L., S. McClean, B. J. Meenan, and P. Millard (2011). Phase-type survival trees and mixed distribution survival trees for clustering patients' hospital length of stay. *Informatica 22*(1), 57–72.

Gelfand, A. E. and A. F. Smith (1990). Sampling-based approaches to calculating marginal densities. *Journal of the American Statistical Association 85*(410), 398–409.

Geyer, C. J. (1991). Markov chain Monte Carlo maximum likelihood. In *Computing Science and Statistics, Proceedings of the 23rd Symposium on the Interface*, pp. 156–163.

Gramacy, R. B. (2007). tgp: an R package for Bayesian nonstationary, semiparametric nonlinear regression and design by treed Gaussian process models. *Journal of Statistical Software 19*(9), 1–46.

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

Gramacy, R. B. and M. Taddy (2010). Categorical inputs, sensitivity analysis, optimization and importance tempering with tgp version 2, an R package for treed Gaussian process models. *Journal of Statistical Software 33*(6), 1–48.

Green, P. J. (1995). Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika 82*(4), 711–732.

Griffin, J. E. and M. J. Steel (2006). Order-based dependent Dirichlet processes. *Journal of the American Statistical Association 101*(473), 179–194.

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

Henderson, N. C., T. A. Louis, G. L. Rosner, and R. Varadhan (2017). Individualized treatment effects with censored data via fully nonparametric Bayesian accelerated failure time models. *arXiv preprint arXiv:1706.06611*.

Higdon, D., H. Lee, and Z. Bi (2002). A Bayesian approach to characterizing uncertainty in inverse problems using coarse and fine-scale information. *Institute of Electrical and Electronics Engineers Transactions on Signal Processing 50*(2), 389–399.

Holmes, C. and D. Denison (2003). Classification with Bayesian MARS. *Machine Learning 50*(1), 159–173.

Holmes, C., D. T. Denison, S. Ray, and B. Mallick (2005). Bayesian prediction via partitioning.

Journal of Computational and Graphical Statistics 14(4), 811–830.

Holmes, C. and B. Mallick (2003). Generalized nonlinear modeling with multivariate free-knot regression splines. *Journal of the American Statistical Association 98*(462), 352–368.

Hothorn, T., P. Bühlmann, S. Dudoit, A. Molinaro, and M. J. Van Der Laan (2005). Survival ensembles. *Biostatistics 7*(3), 355–373.

Hothorn, T., K. Hornik, and A. Zeileis (2006). Unbiased recursive partitioning: A conditional inference framework. *Journal of Computational and Graphical statistics 15*(3), 651–674.

Ishwaran, H., E. H. Blackstone, C. E. Pothier, and M. S. Lauer (2004). Relative risk forests for exercise heart rate recovery as a predictor of mortality. *Journal of the American Statistical Association 99*(467), 591–600.

Ishwaran, H., U. B. Kogalur, E. H. Blackstone, and M. S. Lauer (2008). Random survival forests. *The Annals of Applied Statistics 2*(3), 841–860.

Jacobs, R. A., M. I. Jordan, S. J. Nowlan, and G. E. Hinton (1991). Adaptive mixtures of local experts. *Neural Computation 3*(1), 79–87.

Jara, A. and T. E. Hanson (2011). A class of mixtures of dependent tail-free processes. *Biometrika 98*(3), 553–566.

Joensuu, H., A. Vehtari, J. Riihimäki, T. Nishida, S. E. Steigen, P. Brabec, L. Plank, B. Nilsson, C. Cirilli, C. Braconi, et al. (2012). Risk of recurrence of gastrointestinal stromal tumour after surgery: an analysis of pooled population-based cohorts. *The Lancet Oncology 13*(3), 265–274.

Kalbfleisch, J. D. (1978). Non-parametric Bayesian analysis of survival time data. *Journal of the Royal Statistical Society. Series B (Methodological) 40*(2), 214–221.

Kaplan, E. L. and P. Meier (1958). Nonparametric estimation from incomplete observations. *Journal of the American Statistical Association 53*(282), 457–481.

Kim, H., B. K. Mallick, and C. Holmes (2005). Analyzing nonstationary spatial data using piecewise Gaussian processes. *Journal of the American Statistical Association 100*(470), 653–668.

Kooperberg, C. and C. J. Stone (1991). A study of logspline density estimation. *Computational Statistics & Data Analysis 12*(3), 327–347.

Korattikara, A., Y. Chen, and M. Welling (2014). Austerity in MCMC land: Cutting the Metropolis-Hastings budget. In *Proceedings of the 31st International Conference on Machine Learning*, Volume 32, pp. 181–189.

Kundu, S. and D. B. Dunson (2011). Single factor transformation priors for density regression. *arXiv preprint arXiv:1108.2720*.

Kuo, L. and B. Mallick (1997). Bayesian semiparametric inference for the accelerated failure-time model. *Canadian Journal of Statistics 25*(4), 457–472.

Leman, S. C., Y. Chen, and M. Lavine (2009). The multiset sampler. *Journal of the American Statistical Association 104*(487), 1029–1041.

Lenk, P. J. (1988). The logistic normal distribution for Bayesian, nonparametric, predictive densities. *Journal of the American Statistical Association 83*(402), 509–516.

Levine, R. A., J. Fan, X. Su, and M. E. Nunn (2014). Bayesian survival trees for clustered observations, applied to tooth prognosis. *Statistical Analysis and Data Mining: The ASA Data Science Journal 7*(2), 111–124.

Loh, W., X. He, and M. Man (2015). A regression tree approach to identifying subgroups with differential treatment effects. *Statistics in Medicine 34*(11), 1818–1833.

Ma, L. (2017). Recursive partitioning and multi-scale modeling on conditional densities. *Electronic Journal of Statistics 11*(1), 1297–1325.

Ma, L. and W. H. Wong (2011). Coupling optional Pólya trees and the two sample problem. *Journal of the American Statistical Association 106*(496), 1553–1565.

Ma, S., J. S. Racine, and L. Yang (2015). Spline regression in the presence of categorical predictors. *Journal of Applied Econometrics 30*(5), 705–717.

Mallick, B. K., D. Ghosh, and M. Ghosh (2005). Bayesian classification of tumours by using gene expression data. *Journal of the Royal Statistical Society: Series B (Statistical Methodology) 67*(2), 219–234.

Martino, S., R. Akerkar, and H. Rue (2011). Approximate Bayesian inference for survival models. *Scandinavian Journal of Statistics 38*(3), 514–528.

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

Mondal, A., B. Mallick, Y. Efendiev, and A. Datta-Gupta (2014). Bayesian uncertainty quantification for subsurface inversion using a multiscale hierarchical model. *Technometrics 56*(3), 381–392.

Moro, S., P. Cortez, and P. Rita (2014, June). A data-driven approach to predict the success of bank telemarketing. *Decision Support Systems 62*, 22–31.

Müller, P., A. Erkanli, and M. West (1996). Bayesian curve fitting using multivariate normal mixtures. *Biometrika 83*(1), 67–79.

Norets, A. and J. Pelenis (2012). Bayesian modeling of joint and conditional distributions. *Journal of Econometrics 168*(2), 332–346.

Pati, D., D. B. Dunson, and S. T. Tokdar (2013). Posterior consistency in conditional distribution estimation. *Journal of Multivariate Analysis 116*, 456–472.

Payne, R. D. and B. K. Mallick (2018). Two-stage Metropolis-Hastings for tall data. *Journal of Classification 35*(1).

Petralia, F., J. T. Vogelstein, and D. B. Dunson (2013). Multiscale dictionary learning for estimating conditional distributions. In *Advances in Neural Information Processing Systems*, pp. 1797–1805.

Quiroz, M., M. Villani, and R. Kohn (2014). Speeding up MCMC by efficient data subsampling. *arXiv preprint arXiv:1404.4178*.

Raftery, A. E., X. Niu, P. D. Hoff, and K. Y. Yeung (2012). Fast inference for the latent space network model using a case-control approximate likelihood. *Journal of Computational and Graphical Statistics 21*(4), 901–919.

Riihimäki, J., A. Vehtari, et al. (2014). Laplace approximation for logistic Gaussian process density estimation and regression. *Bayesian Analysis 9*(2), 425–448.

Robert, C. and G. Casella (2013). *Monte Carlo Statistical Methods*. New York: Springer Science

& Business Media.

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

Shen, W., S. Ghosal, et al. (2016). Adaptive Bayesian density regression for high-dimensional data. *Bernoulli 22*(1), 396–420.

Sparapani, R. A., B. R. Logan, R. E. McCulloch, and P. W. Laud (2016). Nonparametric survival analysis using Bayesian additive regression trees (BART). *Statistics in Medicine 35*(16), 2741–2753.

Stone, C. J., M. H. Hansen, C. Kooperberg, Y. K. Truong, et al. (1997). Polynomial splines and their tensor products in extended linear modeling: 1994 Wald memorial lecture. *The Annals of Statistics 25*(4), 1371–1470.

Tierney, L. and J. B. Kadane (1986). Accurate approximations for posterior moments and marginal densities. *Journal of the American Statistical Association 81*(393), 82–86.

Tipping, M. E. (2001). Sparse Bayesian learning and the relevance vector machine. *Journal of Machine Learning Research 1*(Jun), 211–244.

Tokdar, S. T. and J. K. Ghosh (2007). Posterior consistency of logistic Gaussian process priors in density estimation. *Journal of Statistical Planning and Inference 137*(1), 34–42.

Tokdar, S. T., Y. M. Zhu, J. K. Ghosh, et al. (2010). Bayesian density regression with logistic Gaussian process and subspace projection. *Bayesian Analysis 5*(2), 319–344.

Vanhatalo, J., J. Riihimäki, J. Hartikainen, P. Jylänki, V. Tolvanen, and A. Vehtari (2013). GPstuff: Bayesian modeling with Gaussian processes. *Journal of Machine Learning Research 14*, 1175–1179.

Zhou, L., Q. Xu, and H. Wang (2015). Rotation survival forest for right censored data. *PeerJ 3*, e1009.