Algorithms for Bayesian Conditional Density Estimation on a Large Dataset

by

Palak Jain

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved April 2024 by the
Graduate Supervisory Committee:

P. Richard Hahn, Chair
Jingyu He
Shuang Zhou
Ming-Hung Kao
Shiwei Lan

ARIZONA STATE UNIVERSITY

August 2024

ABSTRACT

This dissertation considers algorithmic techniques for non- or semi-parametric Bayesian estimation of density functions or conditional density functions. Specifically, computational methods are developed for performing Markov chain Monte Carlo (MCMC) simulation from posterior distributions over density functions when the dataset being analyzed is quite large, say, millions of observations on dozens or hundreds of covariates. The motivating scientific problem is the relationship between low birth weight and various factors such as maternal attributes and prenatal circumstances.

Low birth weight is a critical public health concern, contributing to infant mortality and childhood disabilities. The dissertation utilizes birth records to investigate the impact of maternal attributes and prenatal circumstances on birth weight through a statistical method called density regression. However, the challenges arise from the estimation of the density function, the presence of outliers, irregular structures in the data, and the complexity of selecting an appropriate method.

In the absence of other possible effects that cause birth weight to be low, how does birthweight, directly or indirectly causing health problems.

To address these challenges, the study employs a Bayesian Gaussian mixture model inspired by kernel density methods. Additionally, it develops a fast MCMC algorithm tailored to handle the computational demands of large datasets. A targeted sample selection procedure is introduced to overcome difficulties in analyzing weakly informative data.

To further enhance the study's approach to addressing challenges, a sophisticated clustering methodology is incorporated. The study leverages the creation of clusters based on different sizes, emphasizing the scalability and complexity of cluster formation within a dichotomous state space. Valid clusters, representing unique combinations of data points from distinct feature states, offer a granular understanding of patterns in the dataset.

ACKNOWLEDGMENTS

Completing this dissertation has been a journey enriched by the support and encouragement of many individuals whom I would like to express my deepest gratitude to.

First and foremost, I extend my sincere appreciation to my advisor, Dr. Hahn, for their unwavering guidance, insightful feedback, and dedicated mentorship throughout this research endeavor. Their expertise, encouragement, and commitment have been instrumental in shaping the trajectory of this dissertation.

I am thankful to the members of my dissertation committee, Dr. Jingyu He, Dr. Ming-Hung Kao, Dr. Shiwei Lan and Dr. Shuang Zhou, for their valuable contributions, constructive feedback, and scholarly insights that have significantly enhanced the quality of this work. Their collective expertise has been a source of inspiration and guidance.

I want to express my gratitude to my colleagues and peers who have been a constant source of support, fostering an intellectually stimulating environment. The collaborative spirit and exchange of ideas have played a crucial role in refining the concepts explored in this dissertation.

Special thanks are due to my family and friends, especially my mother, for their unwavering support, understanding, and encouragement throughout this academic journey. Their belief in my abilities has been a driving force, and I am truly grateful for their presence in my life.

Finally, I would like to acknowledge the institutions and organizations that have provided financial assistance and resources, enabling the successful completion of this dissertation.

This work is dedicated to all those who have been part of this academic expedition, contributing to its growth and fruition.

TABLE OF CONTENTS

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

In the field of statistical modeling and data analysis, the search for robust and adaptable methods has led us to explore various techniques that can capture the underlying structures of complex datasets. One of the widely used approaches to understanding these data sets is *nonparametric density estimation* [33] [16] [9]. This approach provides a way to disclose intricate patterns without the constraints of predefined parametric models. We will delve into techniques like Kernel Density Estimation and Bayesian Gaussian Mixture Models as we go deeper into each aspect of this method. Each of these approaches has its own advantages when it comes to identifying patterns in many different kinds of datasets. Furthermore, computational methods such as the Metropolis-Hastings Algorithm and Reversible Jump Monte Carlo Markov Chain (RJMCMC) enhance adaptability to varying complexities. This preface lays the groundwork for a comprehensive review that will offer additional modifications and comparing with existing models in later chapters.

1.1 Nonparametric Density Estimation

1.1.1 Kernel Density Estimation

Kernel density technique brief

Kernel Density Estimation is a non-parametric technique that is used to estimate the probability density function of a random variable. This technique convolves a kernel function with the sample data in order to get a smoothed and continuous estimate of the underlying density.

Consider a set of data points y_i, where i can be any number from 1 to N, the

density estimate at a point y with Gaussian kernel is given by:

$$\text{KDE}(y, h) = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{h\sqrt{2\pi}} \exp\left\{-\frac{1}{2}\left(\frac{y - {y_i}^2}{h}\right)\right\}$$

The bandwidth, h, is a parameter that controls the "spread" of the smoothing. This bandwidth determines whether the estimate appears smooth or bumpy.

Kernel Density Estimation is a flexible technique since it doesn't assume a specific shape for the data and can adapt to whatever information is available. The selection of kernel function and bandwidth parameter is important because they define how exact and precise the density estimate will be. We can adjust these parameters to better fit the unique characteristics of the data, which makes Kernel Density Estimation an effective method.

Sliverman's 1986 book [33] gives a complete overview of density estimation methods. The articles [27] and [26] by B. U. Park and J. S. Marron compare various methods for selecting bandwidths. The authors emphasize on "data-driven" bandwidth selectors, which implies the bandwidth is decided by the data itself rather than being fixed, that also balances bias and variation.

1.1.2 Bayesian Gaussian Mixture Models (BGMMs)

Distinguishes Bayesian GMM vs. regular GMM

A Gaussian Mixture Model assumes data is generated from a discrete mixture of Gaussian distributions. A Bayesian Gaussian Mixture Model (BGMM) is a probabilistic model that treats the unknown parameters of the mixture components (the means and variances) as random variables with specified prior distributions. A Gaussian mixture model may have a fixed number of components or an unknown number of components. In the Bayesian framework it is common to use latent variables to assign individual data points to distinct components and Markov Chain Monte Carlo techniques to estimate the posterior distribution of model parameters.

In more detail, the Gaussian mixture model for random variable Y may be written in terms of a density function f as

$$\begin{aligned} f(y \mid \theta) &= \sum_{j=1}^{K} w_j \phi(y \mid \mu_j, \sigma_j^2) \\ &= \sum_{j=1}^{K} w_j \frac{1}{\sqrt{2\pi}\sigma_j} \exp\left\{ -\frac{1}{2} \frac{(y-\mu_j)^2}{\sigma_j^2} \right\} \end{aligned} \tag{1.1}$$

for K components with parameters (μ_j, σ_j^2, w_j) and $\sum_{j=1}^{K} w_j = 1$ and $w_j > 0$ for all j. Alternatively, this model can be written in a stochastic form using latent variables γ_i such that

$$\begin{aligned} \Pr(\gamma_i = j) &= w_j \\ f(y_i \mid \gamma_i) &= \phi(y_i \mid \mu_{\gamma_i}, \sigma_{\gamma_i}^2). \end{aligned} \tag{1.2}$$

The latent variable γ_i can be explicitly modeled using a categorical distribution and priors can be placed over the subgroup means and variances, which we will collect into a generic parameter $\theta = (\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$ where $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_K)$ and $\boldsymbol{\sigma}^2 = (\sigma_1^2, \ldots, \sigma_K^2)$. In the context of our motivating application, Y represents a measurement of infant birthweight, discussed more in section 1.4.

By incorporating a probabilistic approach to subgroup assignments, BGMMs offer a more flexible and realistic representation of the inherent complexity in real-world datasets. This flexibility makes them particularly well-suited for applications where data points may belong to multiple subgroups or when the true underlying structure is ambiguous. BGMM has computational limitations despite its strength, particularly when dealing with large or high-dimensional datasets. Due to its intrinsic complexity, the model is not as scalable and presents challenges for quick calculations, particularly when working with large or high-dimensional datasets.

The paper by Ferguson [10] contributes to the foundational literature to Bayesian approaches to address nonparametric problems, discussing techniques for incorporating prior information and updating beliefs based on observed data. Escobar and

West's 1995 article [8] and Roberts's 1998 paper [30] discusses Bayesian density estimation and inference utilizing mixtures. The authors especially address difficulties relating to hierarchical prior specification, inference about parameter layers, and hyperparameters, with a particular emphasis on variance and precision factors that determine degrees of local smoothing.

1.1.3 Conditional Density Estimation

In *conditional* density estimation, the goal is to learn/estimate $f(y \mid x)$ where $Y \mid X \sim F_X$ denotes a conditional random variable with distribution F_X that has conditional density function $f(y \mid x)$. In a sense, this is like estimating an infinite number of density functions, one for each value of $X = x$. In our motivating application, X represents a vector of medical and demographic attributes relevant to infant birthweight and Y is the birthweight.

Conditional density estimation is also referred to as *density regression*; we will use these terms interchangeably. The expression "density regression" emphasizes the problem of conditional density estimation as one of modeling the relationship between an independent variable and a dependent variable in terms of probability density functions. It estimates the density function of the dependent variable given the independent variable. By comparison, standard multiple linear regression models a conditional mean as a linear function of observed covariates (features or predictors). One generalization of linear regression is nonlinear mean regression, which models the mean of a random variable as a nonlinear function of covariates; i.e. $\mathrm{E}(Y \mid X = x) = g(x)$ for a possibly nonlinear function g. Density regression refers to a further generalization which is concerned not merely with $\mathrm{E}(Y \mid X = x)$ but rather the entire conditional density function $f(y \mid x)$. Density regression therefore also generalizes quantile regression, which is the problem of estimating conditional quantiles. More

specifically, with a conditional density in hand one can obtain a condition cumulative distribution function and from there determine a conditional quantile function for any particular quantile: $F(y \mid x) = \Pr(Y < y \mid x)$ and $F^{-1}(q \mid x) = y$.

From the other side, density regression is a generalization of density estimation, which estimates a single density function $f(y)$ rather than a distinct density for each value of x. This method is useful when the relationship between the variables is complex, non-linear, or the dependent variable is multi-modal. It is also useful when the data is noisy or contains outliers, as it provides a more robust estimate of the relationship compared to traditional linear regression.

Density regression is challenging for several reasons, as discussed in [35] [20]. One of the main challenges is simply the estimation of the density function itself, which is quite difficult already. It is challenging to determine the correct form of the density function and to estimate its parameters accurately, and it can be difficult to select appropriate tuning parameters (such as the bandwidth parameter) in nonparametric methods. Additionally, density regression is often carried out in high-dimensional spaces, such as when the number of regression covariates is large. Further, the presence of outliers, noisy data, and irregular structures – which pose problems in standard linear regression – make density regression extremely challenging.

A number of previous papers have treated conditional density estimation from a Bayesian perspective [6] [13] presents a Bayesian method for regression problems where the response variable is a draw from a particular distribution, with density function that varies nonparametrically as a function of covariates. These earlier papers model the density of the response variable as a mixture of normal distributions, where the weights and means are functions of the predictor variables. The model parameters are estimated using Markov chain Monte Carlo (MCMC) methods. This dissertation is a direct extension and refinement of these approaches.

1.2 Computational techniques

Within the Bayesian statistical framework, Monte Carlo techniques are the go-to method for producing posterior inferences. This dissertation builds on several well-known Markov chain Monte Carlo methods, including Metropolis-Hastings, Reversible Jump, and Sampling Importance-Resampling.

1.2.1 Metropolis-Hastings Algorithm

The Metropolis-Hastings algorithm represents a cornerstone Monte Carlo Markov Chain technique for sampling from probability distributions where direct sampling is infeasible. Initially introduced by N. Metropolis et al. in 1953 [24] and subsequently generalized by W.K. Hastings in 1970 [17], this algorithm has become an indispensable tool in the field of statistical computation.

Within the Bayesian framework, the Metropolis-Hastings algorithm plays a pivotal role in the facilitation of posterior distributions computation, as highlighted by S. Chib in 1995 [2] and elucidated further by C.P. Robert in 2004 [29]. It enables the estimation of a Bayesian model's parameters by generating a Markov chain of random samples that asymptotically approximates the posterior distribution.

The methodological essence of the algorithm is to iteratively propose new samples from a specified proposal distribution. These proposed samples are then subjected to an acceptance-rejection mechanism in the Metropolis-Hastings criterion, which incorporates both the model's likelihood and the specification of the proposal distribution. Through iterative application, the algorithm amasses a substantial collection of samples which serve as an empirical representation of the posterior distribution, thus facilitating subsequent inferential and predictive statistical analyses.

The importance of the Metropolis-Hastings algorithm lies in its ability to explore

the high-dimensional parameter space efficiently, allowing for accurate inference even in complex models.

The Metropolis-Hastings algorithm for sampling from a target distribution π for a random variable Y, using proposal distribution q, consists of the following steps:

- Sample y' from $q(y' \mid y)$.
- Compute
$$\alpha(y, y') = \min\left(1, \frac{\pi(y')q(y \mid y')}{\pi(y)q(y' \mid y)}\right). \tag{1.3}$$
$\alpha(y, y')$ is known as 'acceptance probability'.
- With probability $\alpha(y, y')$ accept the proposed value.

Stopping the algorithm after the sufficient amount of Monte Carlo simulations ensures that the current state will be an approximate draw from π such that we can construct an irregular and irreducible Markov transition kernel $P(y' \mid y)$ in Markov Chain Monte Carlo computation which satisfies the detailed balance condition as represented in equation (1.4).

$$\int \int \pi(y)P(y' \mid y)dy' = \int \int \pi(y')P(y \mid y')dy \tag{1.4}$$

This chain can be simulated to obtain a dependent, estimated sample from the target distribution of interest, $\pi(y)$. To have a stationary distribution π, a chain running according to a transition density $P(y, y')$ must satisfy this condition.

N.B. In Bayesian inference applications, the target distribution of interest is typically the posterior distribution of a random variable, often denoted θ, having observed data, often denoted Y. In that context, the target distribution would be denoted $\pi(\theta \mid Y)$. Above, we have explained the algorithm in terms of sampling a random variable Y because that is often how it is described in previous probability

literature (or X). Later in this dissertation we will focus on target distributions over parameters and our samplers will be written for parameters θ and Y will denote our fixed, observed data.

1.2.2 Gibbs Sampler

Gibbs sampling is a technique used in statistics and machine learning to generate samples from a complex probability distribution [19]. Imagine having a multi-dimensional space where each dimension represents a different variable. Directly sampling from this high-dimensional space can be really hard. Gibbs sampling simplifies this by focusing on one dimension at a time while keeping the others fixed.

In Gaussian Mixture Model, random variable Y is thought of as coming from one of several Gaussian distributions, each with its own mean μ_j and variance σ_j, and a certain weight w_j representing how likely it is to pick that Gaussian. The Gibbs sampler iteratively samples from the conditional posterior distributions of the mixture weights, means, and covariances, given the current values of the other parameters.

The model can be written using the latent indicator variable γ_i where latent variable $\gamma_i = j$ means that observation y_i came from the jth component which follows Gaussian distribution with mean μ_j and variance σ_j^2.

The process starts with initial guesses for which gaussian distribution each data point belongs to. With these initial assignments, Gibbs sampling proceeds in two main steps: updating the distribution assignments for each data point, and then updating the parameters i.e.,mean and variance of each gaussian distribution based on these assignments .

For each data point y_i, the algorithm calculates the probability that y_i belong to each of the K gaussian distributions, given the current parameters. Using Bayes' theorem, this probability can be computed as:

$$\Pr(\gamma_i = j \mid y_i, \mu_j, \sigma_j^2) = \frac{w_j \phi(y_i \mid \mu_j, \sigma_j^2)}{\sum_{k=1}^{K} w_j \phi(y_i \mid \mu_k, \sigma_k^2)} \tag{1.5}$$

Based on the computed probabilities, the algorithm assigns each data point y_i to j^{th} gaussian distribution. Then, the parameters are updated using Normal-inverse-gamma distribution, which assumes exchangeability between parameters [5].

This iterative process of updating latent variables and updating parameters continues until the assignments and parameters stabilize.

1.2.3 Reversible Jump MCMC

In this dissertation, extensive use will be made of models where the dimensionality of the parameters can vary. For this type of model, a generalization of the Metropolis Hastings algorithm called Reversible Jump Monte Carlo Markov Chain (RJMCMC) is the standard approach. RJMCMC is a Bayesian computational algorithm that is developed to handle models with varying dimensions and structures. RJMCMC adaptively updates the model space as the algorithm proceeds, in contrast to conventional methods that explore a fixed-dimension parameter space. Because of its ability to include and remove parameters at any point in the algorithm, it is particularly useful in scenarios where the dimensionality of the model is unknown or changing. This method was introduced by P. J. Green (1995) [14] [28].

The main idea behind RJMCMC is to suggest transitions between models with different dimensions, resulting in a Markov chain that explores a wide range of complexities. These transitions allow the algorithm to "jump" between various model configurations by adding or removing parameters. For the Markov chain to converge to the suitable target distribution, the detailed balance requirement must be maintained during the transition between models, and this is ensured by the reversibility property. In order to accomplish this, the algorithm introduces a collection of aux-

iliary variables that represent the dimensionality of the current model.The proposal mechanism takes into account transitions between models with varying amounts of parameters, and the acceptance probability is changed to reflect these changes. To guarantee that the algorithm effectively searches the model space, RJMCMC implementation in practice requires careful design of the proposal distributions and acceptance probability. Since, it extends the capabilities of traditional MCMC methods, this method is useful for a wide range of applications, including Bayesian model comparison [28] and Bayesian variable selection in regression models [15] [12] [3][23] [21].

We defer a technical description of RJMCMC until Chapter 2, where it will be applied in the context of fitting a Gaussian mixture model where the number of mixture components, K, is not pre-specified.

1.3 Sampling Importance-Resampling (SIR)

The sampling importance-resampling (SIR) algorithm was proposed by Rubin in 1987 [32] in the context of missing data. See [22] for more details and a literature review. In this dissertation we will use it to construct a targeted subsampling to improve computational efficiency in the context of nonparametric Bayesian density estimation. In the SIR algorithm, the goal is to obtain (approximate) samples of a random variable $\tilde{Y}$ from distribution $F_{\tilde{Y}}$ with density function f by first sampling a large number of variates Y_i from a convenient distribution G_Y and then resampling from that sample with specially chosen weights. In more detail, let $Y_1, \ldots, Y_m$ be drawn i.i.d. from distribution G_Y with density g and define a set of resampling weights $w_i \propto f(Y_i)/g(Y_i)$. Now, consider the variable $\tilde{Y}$ drawn from the set $Y_1, \ldots, Y_n$ with weights $w = \{w_1, \ldots w_n\}$. The distribution function of $\tilde{Y}$ can be simplified by applying iterated expectation

$$\Pr(\tilde{Y} \le y \mid Y_{1:n}) = \mathrm{E}(\mathbb{I}\{\tilde{Y} \le y\} \mid Y_{1:n}) = \mathrm{E}(\mathrm{E}(\mathbb{I}\{\tilde{Y} \le y\} \mid \tilde{Y} = Y_i, Y_{1:n})) \tag{1.6}$$

$$= \sum_{i=1}^{n} \mathbb{I}\{Y_i \le y\} w_i \tag{1.7}$$

$$= \frac{\frac{1}{n}\sum_{i=1}^{n} \frac{f(Y_i)}{g(Y_i)} \mathbb{I}\{Y_i \le y\}}{\frac{1}{n}\sum_{j=1}^{n} \frac{f(Y_j)}{g(Y_j)}}. \tag{1.8}$$

As $n \to \infty$, the Weak Law of Large Numbers ([7], Section 2.2) assures that this expression converges to

$$\frac{\mathrm{E}_g\left(\frac{f(Y)}{g(Y)}\mathbb{I}\{Y \le y\}\right)}{\mathrm{E}_g(\frac{f(Y)}{g(Y)})},$$

where

$$\mathrm{E}_g\left(\frac{f(Y)}{g(Y)}\mathbb{I}\{Y \le y\}\right) = \int_{-\infty}^{y} \frac{f(a)}{g(a)} g(a)da = \int_{-\infty}^{y} f(a)da = F(y)$$

and

$$\mathrm{E}_g\left(\frac{f(Y)}{g(Y)}\right) = \int_{-\infty}^{\infty} \frac{f(y)}{g(y)} g(y)dy = 1.$$

That is, we have shown that

$$\Pr(\tilde{Y} \le y \mid Y_{1:n}) \approx F(y),$$

or $\tilde{Y} \sim F$, approximately. While the distribution of $\tilde{Y}$ is not an exact sample from F_Y, for large n it should be a close approximation.

In Chapter 3 we will modify this idea to create an intentionally biased sample from which inferences on the original population are constructed.

1.4 Motivating application: Determinants of infant birth weight

The methodological work in this dissertation was all directly motivated by a specific scientific problem of inferring the determinants of low infant birth weight.

For instance, research has shown that mothers who are older or younger than a certain age range may be at a higher risk for delivering low birth weight infants.

This may be due to factors such as pre-existing health conditions or prenatal care. Mothers with higher levels of education may have better access to healthcare and resources during pregnancy, which can impact the infant's birth weight. Adequate prenatal care is important for the health and development of the fetus. Mothers who receive regular prenatal care throughout their pregnancy may have a lower risk of delivering a low birth weight infant. Smoking during pregnancy has been shown to have a significant impact on fetal growth and development, which can ultimately affect birth weight. Nicotine and other harmful chemicals found in cigarettes can restrict blood flow to the fetus and decrease the amount of oxygen and nutrients the fetus receives, which can then impact birth weight. Likewise gender, marital status, race, age, and education may impact birth weight. Research has shown that male infants tend to weigh more than female infants on average, which could be due to differences in fetal growth patterns and hormone levels. Unmarried mothers may be at a higher risk for certain complications during pregnancy, such as low birth weight, which can impact the infant's birth weight. There are documented differences in birth weight across racial groups, with some groups having higher rates of low birth weight infants. These differences may be due to factors such as socioeconomic status, access to healthcare, and genetic factors.

Fortunately, investigating these hypotheses and comparing the degree of impact on birth weight can be studied empirically because large databases record all of these factors and more. The National Center for Health Statistics maintains a database, which records detailed information on births in American hospitals starting in 1968. Specifically the database includes information on "live" births. A birth is considered live if it shows any sign of life after delivery, regardless of the duration of pregnancy, as defined by the World Health Organization in 1950 and outlined in a United Nations Handbook [25]. The birth registration system in the United States encompasses 50

states, the District of Columbia, New York City, Puerto Rico, the U.S. Virgin Islands, Guam, American Samoa, and the Northern Marianian Islands. For statistical purposes, the term "United States" only refers to the combined 50 states (including New York City) and the District of Columbia. The National Center for Health Statistics (NCHS) receives birth records from all states, two cities, and four territories through the Vital Statistics Cooperative Program. NCHS obtains the birth data for Guam in 2021 from original birth certificate images, which they code and enter into their system.

To take a specific example, the year 2021 consists of some 3,669,928 observations. In addition to birth weight (and other features) we have

- **Boy:** male baby
- **Married:** marital status of the mother
- **Black:** mother's race is black
- **Over33:** mother is over 33 years of age
- **HighSchool:** mother has high school diploma
- **FullPrenatal:** mother had prenatal care for the full duration of her pregnancy
- **Smoker:** if the mother is a smoker

All variables are coded as $\{0, 1\}$. This is only a subset of the recorded factors, but we will focus on these attributes in developing our methods.

While having access to large, rich data is beneficial, analyzing the birth weight data using nonlinear, nonparametric methods poses two basic computational challenges. The first challenge is that density estimation itself can be computationally expensive when sample sizes are large. Our first contribution is to develop a novel Bayesian Gaussian mixture model inspired by kernel density methods and to develop a fast Markov chain Monte Carlo algorithm for fitting the model.

The second challenge is that although the natality data is extremely large (millions of observations per year) most of that data is only weakly informative about particular regions of the density function. In this application, our primary focus is the left tail of the distribution, which, by definition, constitutes only a small fraction of the data. This imbalance in the data poses statistical difficulties for the regression aspect of the problem because small differences in the mean (for example) of the conditional densities can overshadow similarities in other regions of the density (particularly the left tail). Intuitively, this means that the data will be disaggregated because the means are distinct, which prevents any statistical aggregation of the data as it pertains to inferences about the left tail. In other words, we want to ask the question "What features impact increased probability of low birth weight" whereas standard nonparametric methods focusing on the entire density function implicitly answer the question "What features impact any difference in the density function"? Our second contribution is a targeted sample selection procedure which allows our algorithm to utilize a small fraction of the available data, concentrated in the left tail.

In addition to these computational innovations to nonparametric density estimation, we also develop a novel nonlinear regression model and algorithm for the case of dichotomous covariates. We posit a covariate-dependent clustering of the 2^d unique feature combinations, where observations in each cluster are assumed to be independent observations from a shared density.

All of this work will eventually be synthesized to do a thorough analysis of the determinants of infant birth weight using several decades of data from the U.S. natality database. In this dissertation, we describe the individual novel components that were developed towards this goal.

Chapter 2

A REVERSIBLE JUMP ALGORITHM FOR A UNIFORM-WEIGHTS GAUSSIAN MIXTURE MODEL

In the introduction, kernel density estimators (KDEs) and Gaussian mixture models (GMMs) were described. Both represent a density function as a convex combination of "component" densities with possibly differing weights, means, and variances. The KDE uses fixed weights and variances, while the general GMM allows all three component parameters to vary. In this chapter, we investigate a hybrid of these two approaches, which uses fixed uniform weights, but allows the variance and mean to both vary. This small simplification allows a novel algorithm to be developed which does not rely on latent indicator variables, saving computational time when the sample size is large. In this uniform-weights Gaussian mixture model, varying the number of components by adding or dropping component, leads to a simple deterministic change of weights: weights of $1/K$ go to $1/(K+1)$ or $1/(K-1)$, respectively.

2.1 Model

Consider a univariate dataset Y, represented by n independent scalar observations, denoted as y_i for $i = 1, 2, \ldots, n$. The fundamental idea behind a Gaussian mixture model is to represent the probability density function of the individual observations y_i mathematically. This density function can be expressed as:

$$f(y_i \mid \boldsymbol{\mu}, \boldsymbol{\sigma^2}) = \sum_{k=1}^{K} \left(\frac{1}{K}\right) \phi(y_i \mid \mu_k, \sigma_k) \quad \text{for } i = 1, 2, \ldots, n, \tag{2.1}$$

where length of $\boldsymbol{\mu}$ and $\boldsymbol{\sigma^2}$ equal K, the number of components in the mixture. In our model K is unknown a priori and a subject of determination during the model fitting process. In Equation 2.1, $\phi(y_i \mid \mu_k, \sigma_k)$ denotes the Gaussian distribution for the k^{th} component characterized by its mean μ_k and standard deviation σ_k. The mean μ_k identifies the central peak of the distribution, while the standard deviation σ_k quantifies its width, indicating the extent of variation around the mean. Each k iterates over the mixture components, and crucially, this equation applies to each observation y_i independently. This model is different from many Gaussian mixture models in that the weights of each component are considered uniform (and are therefore $1/K$). This simplification is not especially restrictive because we will K to vary, able to adapt the number of components (K) to meet the specific requirements of the data. In other words, extra components with (nearly) identical means and variances essentially amount to a model with fewer number of components but varying weights. To take a simple example, a two component model with parameters $(w_1 = 2/3, \mu_1 = 0, \sigma_1 = 1)$ and $(w_2 = 1/3, \mu_2 = 1, \sigma_2 = 1)$ is identical to a three component model with parameters $(w_1 = 1/3, \mu_1 = 0, \sigma_1 = 1)$ and $(w_2 = 1/3, \mu_2 = 0, \sigma_2 = 1)$ and $(w_3 = 1/3, \mu_3 = 0, \sigma_3 = 1)$. Large values of K, along with (nearly) identical parameters for several components, can provide approximately the same fits using rational approximations to arbitrary weights.

2.2 A Reversible Jump Algorithm

This section introduces an efficient sampling algorithm for estimating the Gaussian mixture normal model described above, based on Reversible Jump Markov Chain Monte Carlo (RJMCMC).

When the number of components K can vary, it's essential to allow the lengths of the parameter vectors $\boldsymbol{\mu}$ and $\boldsymbol{\sigma^2}$ to vary accordingly. In such cases, reversible jump

MCMC (RJMCMC) is the standard tool. We will now outline an algorithm to fit this Gaussian mixture model with fixed uniform weights using an RJMCMC approach. Our algorithm will involve local moves, starting from an initial parameter vector. These proposed changes will encompass three possibilities: we may suggest either an increase or a decrease in the number of components within the Gaussian mixture by one, or we might maintain the number of components at its current value. Each of these options will have a pre-specified probability denoted as q.

The model likelihood for $Y = \{y_1, \ldots, y_n\}$ sampled i.i.d from a GMM is

$$\mathcal{L}(\mu^K, \sigma^K, K) = \prod_{i=1}^{n} f(y_i \mid \mu^K, \sigma^K, K) \tag{2.2}$$

where $\mu^K = (\mu_1, \mu_2, \ldots, \mu_K)$ and $\sigma^K = (\sigma_1, \sigma_2, \ldots, \sigma_K)$ define the means and standard deviations for each of the K components in the mixture, respectively. For notational convenience, in the remainder of the dissertation we refrain from boldfacing the parameters to depict vectors and we refer to σ (the standard deviation) rather than σ^2 the variance, using the superscript to indicate the length of the parameter vector. Therefore, a move from current state (K, μ^K, σ^K) to new state $(K', \mu^{K'}, \sigma^{K'})$ involves a change in dimension from K to K'.

When $K > 1$, our proposal distribution $q(K' \mid K)$ is

$$q(K' \mid K) = \begin{cases} q_+ & \text{if } K' = K + 1 \\ q_- & \text{if } K' = K - 1 \\ q_\varnothing & \text{if } K' = K \end{cases} \tag{2.3}$$

where $q_+ + q_- + q_\varnothing = 1$ and $q_+ = q_-$. (In our specific algorithm we use $q_+ = q_- = 0.3$.) When $K = 1$,

$$q(K' \mid K) = \begin{cases} 1 - q_\varnothing & \text{if } K' = K + 1 \\ q_\varnothing & \text{if } K' = K. \end{cases} \tag{2.4}$$

Consider first the case where $K' = K + 1$, so that a component is being added.

1. Draw $\tilde{\mu} \sim \text{KDE}(y_{1:n}, h)$ and $\tilde{\sigma}/\beta \sim \text{Beta}(a, b)$ where y is the uniformly chosen data point from Y, and h, a, b and β are pre-specified hyper-parameters. Here $\text{KDE}(y_{1:n}, h)$ denotes a kernel density estimate based on the data $y_{1:n}$ and a bandwidth of h.

2. Set $\mu^{K'} = (\mu', \tilde{\mu})$ and $\sigma^{K'} = (\sigma', \tilde{\sigma})$ such that $\mu' = \mu^K$ and $\sigma' = \sigma^K$.

3. Compute the jacobian:

$$\frac{\partial(\mu', \tilde{\mu}, \sigma', \tilde{\sigma})}{\partial(\mu^K, \tilde{\mu}, \sigma^K, \tilde{\sigma})} = \begin{vmatrix} \frac{\partial \mu'}{\partial \mu^K} & \frac{\partial \mu'}{\partial \tilde{\mu}} & \frac{\partial \mu'}{\partial \sigma^K} & \frac{\partial \mu'}{\partial \tilde{\sigma}} \\ \frac{\partial \tilde{\mu}}{\partial \mu^K} & \frac{\partial \tilde{\mu}}{\partial \tilde{\mu}} & \frac{\partial \tilde{\mu}}{\partial \sigma^K} & \frac{\partial \tilde{\mu}}{\partial \tilde{\sigma}} \\ \frac{\partial \sigma'}{\partial \mu^K} & \frac{\partial \sigma'}{\partial \tilde{\mu}} & \frac{\partial \sigma'}{\partial \sigma^K} & \frac{\partial \sigma'}{\partial \tilde{\sigma}} \\ \frac{\partial \tilde{\sigma}}{\partial \mu^K} & \frac{\partial \tilde{\sigma}}{\partial \tilde{\mu}} & \frac{\partial \tilde{\sigma}}{\partial \sigma^K} & \frac{\partial \tilde{\sigma}}{\partial \tilde{\sigma}} \end{vmatrix} = \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} = 1$$

4. Accept the move from (K, μ, σ) to (K', μ', σ') with acceptance probability

$$\alpha((K, \mu^K, \sigma^K), (K', \mu^{K'}, \sigma^{K'})) = \min\left\{1, \frac{\mathcal{L}(\mu^{K'}, \sigma^{K'}, K')\pi(K')}{\mathcal{L}(\mu^K, \sigma^K, K)\pi(K)g_\mu(\tilde{\mu})g_\sigma(\tilde{\sigma})}\right\}$$

   where $\pi(k)$ is the density of a $\text{Gamma}(\beta \log(n), \beta)$ and β is the pre-specified hyper parameter. $g_\mu(\cdot)$ is KDE is the kernel density estimate at y with Gaussian kernel and bandwidth h, and $g_\sigma(\cdot)$ is the beta density with shape parameters a and b.

Note that the Jacobian term is here because of its standard use in reversible jump MCMC; it becomes 1 here because we are doing local changes that operate componentwise and do not "blend" different elements of the parameters vectors so there is no change in volume necessary.

Next, consider a move from (K, μ, σ) to (K', μ', σ') such that $K' = K - 1$. That is, let us consider a move by defining

1. Set $\mu^K = (\mu', \tilde{\mu})$ and $\sigma^K = (\sigma', \tilde{\sigma})$ such that $\mu^{K'} = \mu'$ and $\sigma^{K'} = \sigma'$.

2. Compute the jacobian:

$$\frac{\partial(\mu^{K'}, \tilde{\mu}, \sigma^{K'}, \tilde{\sigma})}{\partial(\mu', \tilde{\mu}, \sigma', \tilde{\sigma})} = \begin{vmatrix} \frac{\partial \mu^{K'}}{\partial \mu'} & \frac{\partial \mu^{K'}}{\partial \tilde{\mu}} & \frac{\partial \mu^{K'}}{\partial \sigma'} & \frac{\partial \mu^{K'}}{\partial \tilde{\sigma}} \\ \frac{\partial \tilde{\mu}}{\partial \mu'} & \frac{\partial \tilde{\mu}}{\partial \tilde{\mu}} & \frac{\partial \tilde{\mu}}{\partial \sigma'} & \frac{\partial \tilde{\mu}}{\partial \tilde{\sigma}} \\ \frac{\partial \sigma^{K'}}{\partial \mu'} & \frac{\partial \sigma^{K'}}{\partial \tilde{\mu}} & \frac{\partial \sigma^{K'}}{\partial \sigma'} & \frac{\partial \sigma^{K'}}{\partial \tilde{\sigma}} \\ \frac{\partial \tilde{\sigma}}{\partial \mu'} & \frac{\partial \tilde{\sigma}}{\partial \tilde{\mu}} & \frac{\partial \tilde{\sigma}}{\partial \sigma'} & \frac{\partial \tilde{\sigma}}{\partial \tilde{\sigma}} \end{vmatrix} = \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} = 1$$

3. Accept the move from (K, μ, σ) to (K', μ', σ') with acceptance probability

$$\alpha((K, \mu^K, \sigma^K), (K', \mu^{K'}, \sigma^{K'})) = \min\left\{1, \frac{\mathcal{L}(\mu^{K'}, \sigma^{K'}, K')\pi(K')g_\mu(\tilde{\mu})g_\sigma(\tilde{\sigma})}{\mathcal{L}(\mu^K, \sigma^K, K)\pi(K)}\right\}$$

   where $g_\mu(\cdot) = \text{KDE}(y_{1:n}, h)$ and $g_\sigma(\cdot) = \text{Beta}(a, b)$.

Lastly, consider a move from (K, μ, σ) to (K', μ', σ') such that $K' = K$. In this case, we randomly choose a component and conduct a random walk in the direction determined by a standard normal random variable. Therefore, the acceptance porbability will be

$$\alpha((K, \mu^K, \sigma^K), (K', \mu^{K'}, \sigma^{K'})) = \min\left\{1, \frac{\mathcal{L}(\mu^{K'}, \sigma^{K'}, K')\pi(K')}{\mathcal{L}(\mu^K, \sigma^K, K)\pi(K)}\right\}.$$

In summary, our methodology leverages Markov Chain Monte Carlo to navigate through diverse Gaussian mixture models, suggesting modifications to both the number of components and their associated parameters. These adjustments enhance the model's fit to the data, leading to more accurate posterior parameter estimations. Through these proposed changes and the corresponding acceptance criteria, our MCMC algorithm adeptly explores the parameter space, establishing itself as a valuable instrument for statistical inference.

2.3 Computational considerations

A common way to fit Gaussian mixture models, as mentioned in the introduction, is to impute latent variables that indicate which mixture component each observation arose from. This approach requires introducing and sampling a length n indicator variables, which involves keeping track of, and updating at each iteration, n K-dimensional probability vectors. When K is large, as might be necessary in the case of a skewed distribution, this computational burden can be high. When K is allowed to vary, the necessary data structures for this can also be difficult to deal with.

In our algorithm, with its fixed weights, we do not rely on sampling latent indicator variables. In addition to avoiding creating an addition n-dimensional parameter, we also avoid having to sample these variables and so do not have to evaluate the component likelihoods for every component on every data point at each iteration (or, equivalently, store them). Instead, our algorithm evaluates the likelihoods by taking advantage of the additive structure of a mixture model. Specifically, we maintain a likelihood vector, which can be written as:

$$\begin{bmatrix} \mathcal{L}_1 \\ \mathcal{L}_2 \\ \vdots \\ \mathcal{L}_n \end{bmatrix} = \begin{bmatrix} f_{1,1} & f_{1,2} & \dots & f_{1,K} \\ f_{2,1} & f_{2,2} & \dots & f_{2,K} \\ \vdots & \vdots & \vdots & \\ f_{n,1} & f_{n,2} & \dots & f_{n,K} \end{bmatrix} \begin{bmatrix} 1/K \\ 1/K \\ \vdots \\ 1/K \end{bmatrix}. \tag{2.5}$$

Accordingly, computing the likelihood of a proposed parameter vector can be accomplished in a single sweep of the n data points, using the following updates, depending on whether a component is being modified, added, or dropped. In the case of modifying a single component's parameters, the likelihood update is

$$\mathcal{L}_i' = \frac{K\mathcal{L}_i - f_{i,j} + f_{i,j}'}{K} \tag{2.6}$$

for updating the jth component. For adding a new component, the update is

$$\mathcal{L}_i' = \frac{K\mathcal{L}_i + f_{i,K+1}}{K+1}. \tag{2.7}$$

For removing the jth component the update is

$$\mathcal{L}_i' = \frac{K\mathcal{L}_i - f_{i,j}}{K-1}. \tag{2.8}$$

These updates are simple and readily parallelizable. In future work we plan to investigate how they can be efficiently implemented on graphical processing units (GPUs) for massive parallelization. Pseudocode is given below in Algorithm 1; our code is implemented in `C++` to be run in `R` using the `Rcpp` package.

2.4 Small illustration

In this example, we generate a sample comprising 1000 observations, drawn from a six-component mixture of Gaussians, and demonstrate that our algorithm produces reliable point estimates and inferences. A more thorough simulation study is deferred until the next chapter where this algorithm is combined with a novel data subsampling strategy; there it is compared with existing packages fitting more traditional Bayesian Gaussian mixture models.

Figure (2.1) shows our estimated density curve, depicted by a dotted green line and the true mixture density is shown in purple. The 95% credible interval is illustrated by a dotted red area within the figure, providing a quantifiable measure of our estimates' uncertainty. This interval demarcates the likely range of the true data density, thus serving as an indicator of our analysis's precision and reliability.

2.5 Empirical illustration

In this illustration, we have used birthweight dataset from the year 2021 as described in section 1.4. Figure (2.2) represents our estimated density curve, depicted

Algorithm 1 Reversible Jump Kernel Density Estimation (RJKDE)

procedure RJKDE

Input: Data Y of length n, hyperparameters a, b, β, h.

Output: samples from a posterior distribution over mixture component parameters.

Initialize to K components with parameters μ_k and σ_k, $k \in \{1, \ldots, K\}$.

  for M iterations **do**

    Draw Δ from the set $\{-1, 0, 1\}$ with probabilities $q_+ = q_- = 0.3$ and $q_\varnothing = 0.4$. $K' \leftarrow |K - 1 + \Delta|$.

    if $K' > K$ **then**

    Draw $\tilde{\mu} \sim \mathrm{KDE}(y_{1:n}, h)$ and $\tilde{\sigma} \sim \beta\mathrm{Beta}(a, b)$

    else if $K' < K$ **then**

      Drop a μ and σ component uniformly at random.

    else if $K' = K$ **then**

      Propose an update to a single parameter pair (μ_j, σ_j) using a random walk proposal distribution.

    end if

    Compute the likelihoods $\mathcal{L}(\mu^{K'}, \sigma^{K'}, K')$ and $\mathcal{L}(\mu^K, \sigma^K, K)$.

    Compute the prior evaluations of K' and K, $\pi(K')$ and $\pi(K)$.

    Compute $g_\mu(\tilde{\mu})$, $g_\sigma(\tilde{\sigma})$.

    Compute α, the acceptance probability as described in the main text.

    Sample u as $U \sim \mathrm{Uniform}(0, 1)$.

    if $u < \alpha$ **then**

      Update the state as $(K, \mu^K, \sigma^K) \leftarrow (K', \mu^{K'}, \sigma^{K'})$.

    end if

    Return (K, μ^K, σ^K).

  end for

end procedure

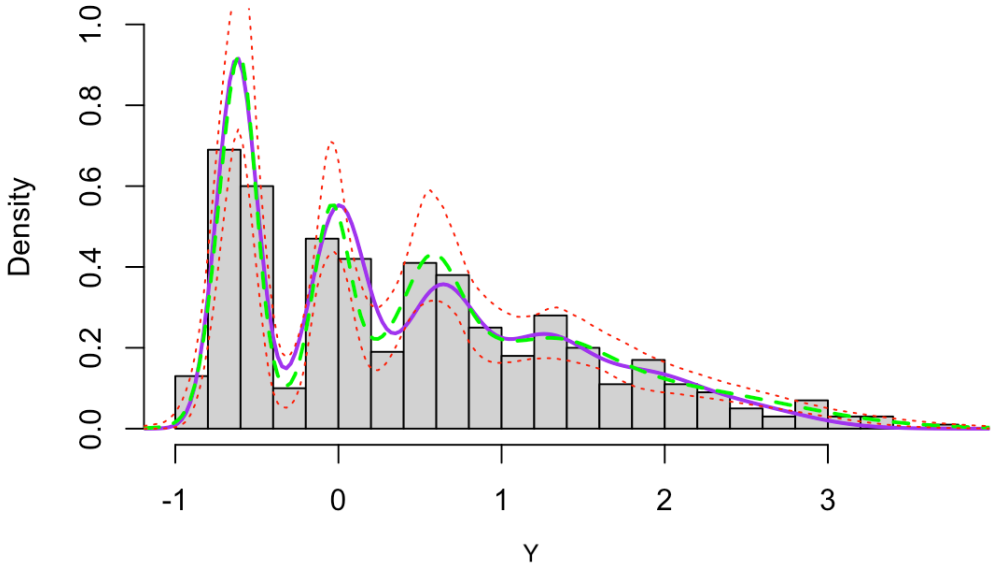

Figure 2.1: Illustration of RJKDE Method

The histogram illustrates the distribution of dataset Y, while the solid purple curve represents the true data density. The dotted green curve is the density estimated using RJKDE. The dotted red region indicates the 95% credible interval.

by a dotted green line. The histogram represent complete dataset (in Kgs). The 95% credible interval is illustrated by a dotted red area within the figure.

As we see in the figure (2.2), the density curve is very precise, closely matching the actual data distribution. Also, the credible intervals are narrow which indicates high level of certainty in our estimates.

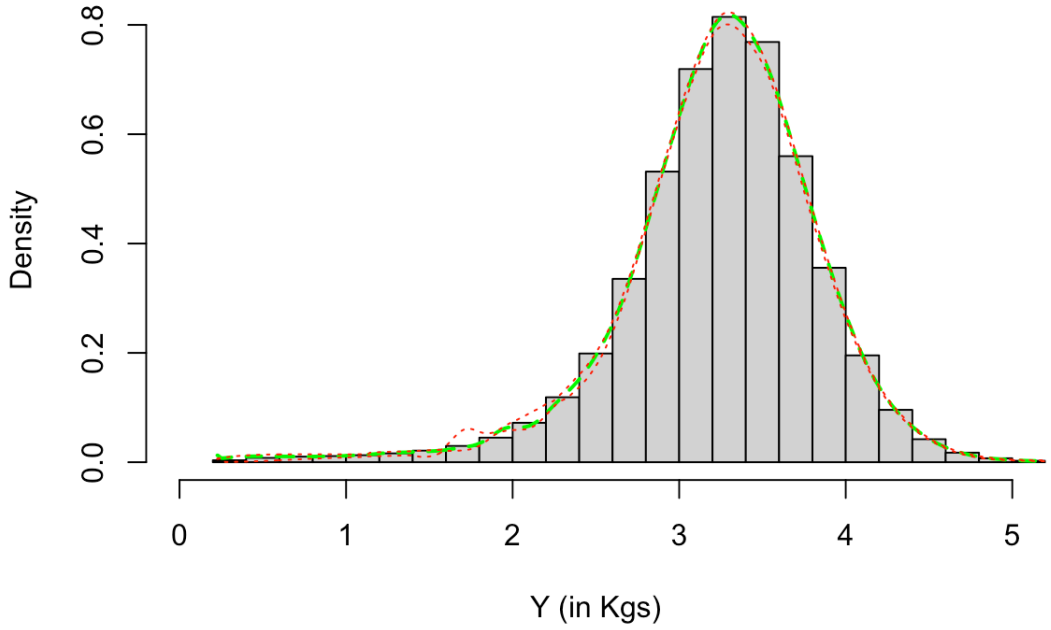

Figure 2.2: Illustration of RJKDE Method on 2021 Birthweight dataset

The histogram illustrates the distribution of birthweight dataset Y. The dotted green curve is the density estimated using RJKDE. The dotted red region indicates the 95% credible interval.

Chapter 3

TARGETED SUBSAMPLING FOR GAUSSIAN MIXTURE MODELING OF LARGE DATASETS

Even with a fast RJMCMC algorithm for fitting Bayesian GMMs, some data sets are simply too big. In our motivating application, the US Natality dataset is approximately a quarter of a million observations per month dating back to the mid-60s. If we wanted to use all of that data to estimate a density function, it would be several hundred million observations.

In this chapter, a subsampling technique is presented, aiming to enable better estimation of particular, pre-specified regions of the density function without having to use the entire data set. The way that this is achieved is through intentionally biased subsampling of the data, the effects of which – because the bias is known – can be adjusted for in the posterior.

This approach has certain advantages over simpler approaches. First, one could analyze a uniform random sample from the entire dataset. This approach has the flaw that it under-samples from the region of interest; in extreme cases you may end up with almost no observations that inform about the estimand of interest, for instance, a tail density. At the opposite extreme, another approach would be to subsample only from a fixed target region; this approach has the flaw that it does not take into account any of the data from outside of that region, which may be statistically valuable in the case of a continuous density function. The new approach strikes a compromise between these two extremes, focusing the computational resources on data that is likely to be most informative about key estimands but without having to pre-specify hard thresholds and so still able to borrow information from adjacent

regions of sample space. In practical terms, targeted subsampling means that instead of analyzing the entire massive dataset, researchers can focus their computational resources and statistical efforts on a reduced subsample from select regions of the observation space.

3.1 Sampling Importance-resampling, revisited

Our biased sampling scheme is inspired by the SIR algorithm of [32]. Instead of using weights $w_i \propto f(Y_i)/g(Y_i)$ we instead apply weights $w_i \propto h(Y_i)$ for a *targeting* density h that is supplied by the analyst. We assume that the observed data comes from a distribution F_Y with density f. As a result, the resampled data $\tilde{Y}$ is approximately drawn from a distribution $\tilde{F}$ which has density $\tilde{f}(y) \propto h(y)f(y)$ where h is the analysts biasing density and f is nature's distribution over the (un-resampled) data. In more detail,

$$\Pr(\tilde{Y} \le y \mid Y_{1:n}) = \mathrm{E}(\mathbb{I}\{\tilde{Y} \le y\} \mid Y_{1:n}) = \mathrm{E}(\mathrm{E}(\mathbb{I}\{\tilde{Y} \le y\} \mid \tilde{Y} = Y_i, Y_{1:n})) \tag{3.1}$$

$$= \sum_{i=1}^{n} \mathbb{I}\{Y_i \le y\} w_i \tag{3.2}$$

$$= \frac{\frac{1}{n}\sum_{i=1}^{n} h(Y_i)\mathbb{I}\{Y_i \le y\}}{\frac{1}{n}\sum_{j=1}^{n} h(Y_j)}. \tag{3.3}$$

As $n \to \infty$, the Weak Law of Large Numbers ([7], Section 2.2) assures that this expression converges to

$$\frac{\mathrm{E}_f\left(h(Y)\mathbb{I}\{Y \le y\}\right)}{\mathrm{E}_f(h(Y))},$$

where

$$\mathrm{E}_f\left(h(Y)\mathbb{I}\{Y \le y\}\right) = \int_{-\infty}^{y} h(a)f(a)da$$

and

$$\mathrm{E}_f\left(h(Y)\right) = \int_{-\infty}^{\infty} h(y)f(y)dy.$$

Note that the denominator is the normalizing constant for the density $\tilde{f}$.

Our strategy will be to perform estimation concerning $\tilde{f}$ using a Bayesian kernel density model on the resampled data and then deduce the implied inferences on the underlying complete population density f. This is facilitated by using a biasing density that is Gaussian, which allows inferences on f to be obtained in closed form.

3.1.1 Debiasing Subsampled Gaussian Mixtures

If $f(y)$ is presumed to arise from a finite Gaussian mixture model, then the biasing distribution can be understood as affecting the distribution component-wise, in the sense that

$$\tilde{f}(y) \propto h(y)f(y) = h(y)\sum_{k=1}^{K}\pi_k\phi(y \mid \mu_k, \sigma_k^2) = \sum_{k=1}^{K}\pi_k h(y)\phi(y \mid \mu_k, \sigma_k^2),$$

where $\sum_{k=1}^{K}\pi_k = 1$. Moreover, when h is Gaussian, $\tilde{f}$ can be computed explicitly using well-known facts about convolutions of Gaussian random variables. Here, we explain the derivation in terms of a conjugate Bayesian update.

Specifically, when $h(y) = \phi(y \mid m, \lambda^2)$ and $f_k(y) = \phi(y \mid \mu_k, \sigma_k^2)$ they may be thought of as a "prior" and a "likelihood" respectively. First, note that $\phi(y \mid m, \lambda^2) = \phi(m \mid y, \lambda^2)$; the latter expression allows us to think of m as the "data", $\phi(m \mid y, \lambda^2)$ as a "prior", and y as the "parameter". Accordingly, $f_k(y) = \phi(y \mid \mu_k, \sigma_k^2)$ can be thought of as a "likelihood". This allows us to apply the conjugate update formulas to determine, for each mixture component, what the normalizing constant will be. Specifically, each component in $\tilde{f}$ is going to be the "posterior" for y, having seen data μ_k and having a prior centered at m. Therefore using conjugate prior, the posterior parameters can be written as

$$\tilde{\sigma}_k^2 = \left(\frac{1}{\frac{1}{\sigma_k^2} + \frac{1}{\lambda^2}}\right), \tag{3.4}$$

$$\tilde{\mu}_k = \left(\frac{\mu_k}{\sigma_k^2} + \frac{m}{\lambda^2}\right)\tilde{\sigma}_k^2. \tag{3.5}$$

Solving equations (3.4) and (3.5) provides us with estimates for μ_k and σ_k^2:

$$\sigma_k^2 = \left(\frac{1}{\frac{1}{\tilde{\sigma}_k^2} - \frac{1}{\lambda^2}} \right), \tag{3.6}$$

$$\mu_k = \left(\frac{\tilde{\mu}_k}{\tilde{\sigma}_k^2} - \frac{m}{\lambda^2} \right) \sigma_k^2. \tag{3.7}$$

It is important to note that σ_k^2 must be greater than 0 in equation (3.6), implying:

$$\frac{1}{\tilde{\sigma}_k^2} > \frac{1}{\lambda^2}. \tag{3.8}$$

That is, the prior's standard deviation must be sufficiently large, surpassing the "posterior" standard deviation of the targeted sub-sampling for the low-probability region.

With these relationships established, we can know deduce the relationship between $\tilde{\pi}_k$ and π_k. Begin by considering the denominator of $\tilde{f}$, which is

$$\int h(y) f(y) dy = \int h(y) \sum_{k=1}^{K} \pi_k \phi(y \mid \mu_k, \sigma_k^2) dy = \sum_{k=1}^{K} \pi_k \int h(y) \phi(y \mid \mu_k, \sigma_k^2) dy.$$

This term is relatively easy to deal with, because $\int h(y)\phi(y \mid \mu_k, \sigma_k^2)dy$ is the marginal likelihood of a Gaussian model with a Gaussian prior. Specifically, it is $\phi(m \mid \mu_k, \sigma_k^2 + \lambda^2)$. Accordingly, we have the relationship

$$\tilde{\pi}_k \propto \pi_k \phi(m \mid \mu_k, \sigma_k^2 + \lambda^2).$$

When fitting a KDE Mixture model, as in the last chapter, we assume that the mixture weights are equal to $1/K$ for all components. Here, making this same assumption for $\tilde{\pi}_k$ yields the relationship $\pi_k^{-1} \propto \phi(m \mid \mu_k, \sigma_k^2 + \lambda^2)$. Note that the equal-weights assumption for $\tilde{f}$ does not imply an equal weights assumption for f; although with a variable number of components, the restriction has few practical ramifications on the ability of the model to characterize arbitrary density functions.

3.2 Small illustration

To illustrate the effectiveness of targeted subsampling in the context of density estimation, we present a simulation study with synthetic data. The dataset, as depicted in Figure (3.1), was simulated for a total of $N = 10,000$ observations, generated from a mixture model with 8 components. The true density of the data is outlined in purple, revealing a significant difference between the left and right tails of the distribution.

The simulation intentionally mimics a scenario where certain regions of the sample space have low density, specifically focusing on the left-hand side tail. This characteristic is crucial for assessing the performance of the targeted subsampling technique. The simulation employs a mixture model with 8 components, creating a dataset that exhibits varying densities across different regions. A simulation loop with 10,000 iterations was implemented.

To emphasize the benefits of targeted subsampling, a subset of the data, denoted as $\tilde{n} = 400$, is sampled using targeting function parameters $\mu_\tau = -1.5$ and $s_\tau = 1$. These values bias the subsample towards the low-density region of interest on the left-hand side. The RJKDE method is then applied to estimate the density of the subsampled dataset and posterior inferences are subsequently "inverted" to obtain inferences about the underlying density. This method is compared to an approach that uses data subsampled uniformly (and so no inversion is necessary). The top left of Figure (3.1) displays the histogram of the targeted subsampled dataset, while the top right shows the histogram of a randomly subsampled dataset. The green dotted curve represents the estimated density using RJKDE for the targeted subsample, with a 95% credible interval in red dots.

The results in this single example are striking, showcasing the precision of the

density estimation achieved through targeted subsampling. The bottom left of Figure (3.1) zooms in on the left-hand side tail, displaying the estimated density curve and the associated credible interval for the targeted subsampled dataset. In contrast, the bottom right provides the same information for the randomly subsampled dataset. The true density curve is consistently outlined in purple for reference.

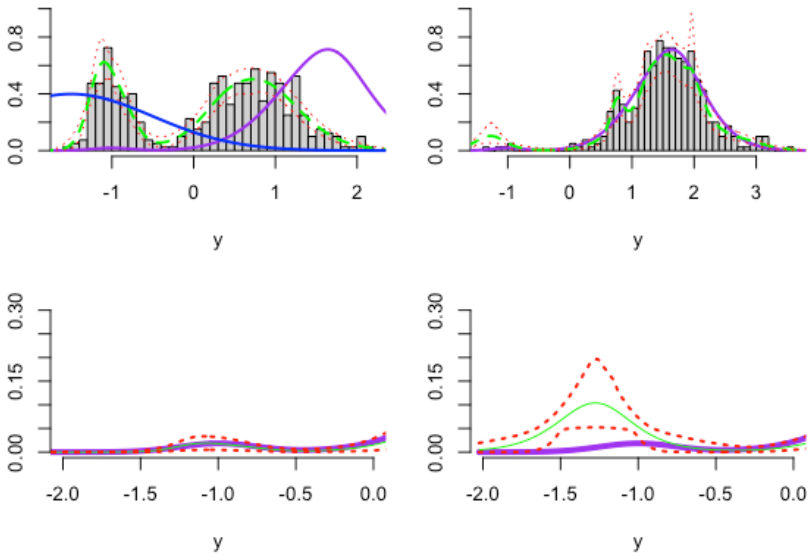

Figure 3.1: Illustration of Targeted subsampling along with RJKDE Method

On the top left, histogram of the subsampled dataset using weights w, whereas on the top right, histogram of the randomly subsampled dataset. The estimated density curve using RJKDE in dotted green and 95% credible interval in dotted red of the subsampled dataset. On the bottom left, the estimated density curve in dotted green and 95% credible interval in dotted red of the original dataset just on the left hand side tail for weighted subsampled dataset and on the bottom right for the random subsampled dataset. The true density curve is represented in purple.

This small example illlustrates the superior performance of RJKDE when combined with targeted subsampling. The method efficiently captures and estimates

the density of specific regions of interest, outperforming traditional random subsampling. The slender credible intervals associated with the model highlight the accuracy achieved by this innovative approach, providing a more efficient and computationally efficient method for density estimation in the context of large datasets.

3.3 Empirical illustration

In this illustration, we have used birthweight dataset from the year 2021 as described in section 1.4.

A subset of the data,$\tilde{n}$, consisting of 1000 datapoints, is sampled using targeting function parameters $\mu_\tau = 0.6$ and $s_\tau = 1$. These values bias the subsample towards the low-density region of interest on the left-hand side which is a low birthweight region in this case. The RJKDE method is applied to estimate the density of the subsampled dataset. This method is compared to an approach that uses data subsampled uniformly.

The top left of Figure (3.2) displays the red histogram of the targeted subsampled dataset whereas the grey histogram represent complete dataset (in Kgs). The top right shows the red histogram of a randomly subsampled dataset. The green dotted curve represents the estimated density using RJKDE for the targeted subsample, with a 95% credible interval in red dotted line. The bottom left of Figure (3.2) zooms in on the left-hand side tail, displaying the estimated density curve and the associated credible interval for the targeted subsampled dataset. The bottom right provides the same information for the randomly subsampled dataset.

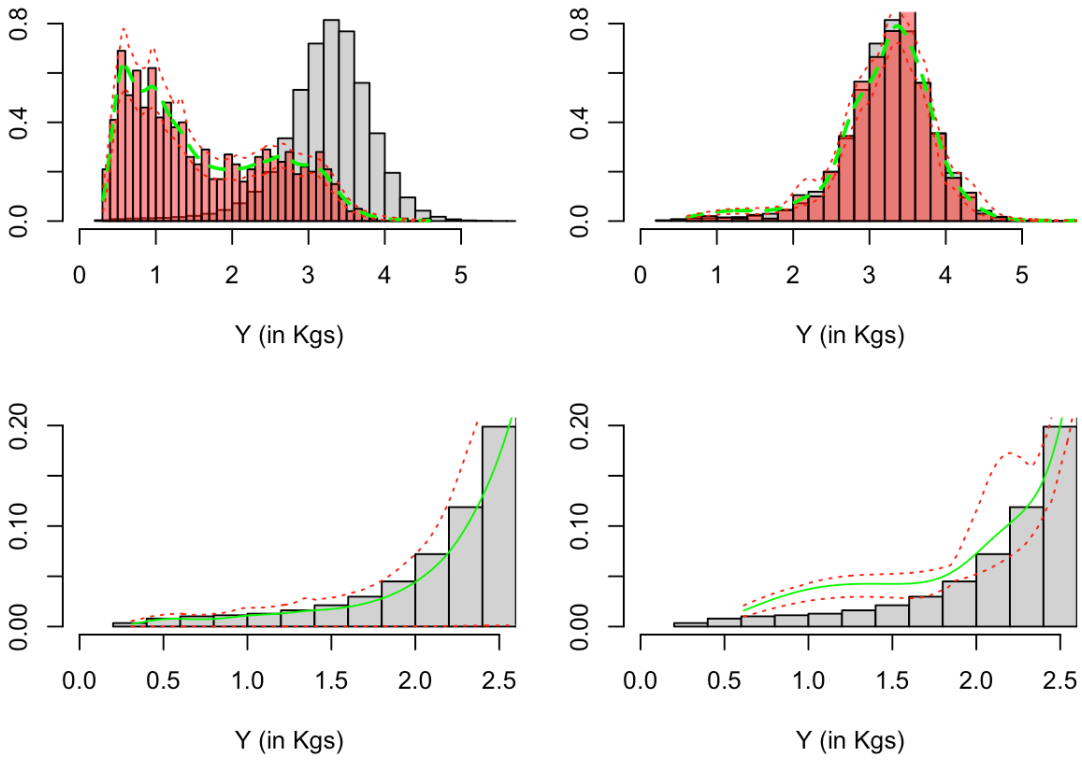

Figure 3.2: Illustration of Targeted subsampling along with RJKDE Method on Birthweight dataset

On the top left, histogram (colored in red) of the subsampled dataset using weights w, whereas on the top right, histogram of the randomly subsampled dataset. The estimated density curve using RJKDE in dotted green and 95% credible interval in dotted red of the subsampled dataset. On the bottom left, the estimated density curve in dotted green and 95% credible interval in dotted red of the original dataset just on the left hand side tail for weighted subsampled dataset and on the bottom right for the random subsampled dataset.

The density curve estimated using targeted subsampling aligns more closely with the histogram of the complete dataset compared to the density curve obtained through random subsampling. This means that the targeted subsampling method provides a more accurate representation of the true underlying data distribution, particularly in

the region of low birthweights.

3.4 Simulation and Comparative Analysis of Density Estimation Methods

In the process of simulating data, our objective is to generate a dataset that imitates the characteristics of a birthweight dataset. The dataset, comprising $N = 5000$ observations, is constructed through a mixture of normal distributions involving four distinct groups. Each group within dataset is distinguished by its unique mean, standard deviation, and weight parameters such that $\boldsymbol{\mu} = (-1.25, -1, -0.5, 1)$, $\boldsymbol{\sigma} = (0.1, 0.1, 0.1, 0.5)$ and $\boldsymbol{\pi} = (0.01, 0.02, 0.01, 0.96)$. Subsequently, three distinct datasets are derived from this mixture. The first employs Targeted subsampling of size $\tilde{n} = 500$. The second dataset is created without targeted subsampling, i.e., using uniform subsampling, again of size $\tilde{n} = 500$. Finally, the third dataset is the complete dataset.

We implemented a simulation loop of 100 iterations, each loop generating a distinct complete dataset. For each generated dataset we estimate the density with four separate methods, three of them using the RJKDE from the last chapter, and one using the `bayesm` package. Performance evaluation metrics, including Root Mean Squared Error (RMSE), coverage of credible intervals, complete coverage, and interval width, where complete coverage indicating the inclusion of the entire true density within the estimated credible interval, are employed. These metrics provide insights into the accuracy, precision, and reliability of each method.

In our density estimation approach, specifically using RJKDE, we set some initial values to guide the process. Initially, we assume there are 5 components in the mixture. As opposed to this, for alternative methods, including the Bayesian approach using the `bayesm` package [31], we initialize the number of components to be 5 times the number used in the Targeted subsampling. We employ a scaled version (10 times) of the bandwidth acquired from the density function of our sampled data to compute

the bandwidth, which is an important hyperparameter in our algorithm. This aids in our ability to adjust to the unique properties of the dataset that we are using. For this simulation study $\mu_\tau = -0.75$, and $s_\tau = 0.75$.

From table 3.1, we see that Targeted subsampling proved to be the fastest method, with an average time of 4.03 seconds. Bayesian modeling using the `bayesm` package took significantly longer at 23.21 seconds. Targeted subsampling and No subsampling achieved the lowest RMSE, indicating high accuracy in density estimation. Uniform subsampling had higher RMSE values. Targeted subsampling and Uniform Subsampling with RJKDE method exhibited good coverage of the true density, whereas Uniform subsampling with bayesian method showed lower coverage rates. Interestingly, the Uniform subsampling with RJKDE method achieved higher complete coverage in all iterations. Uniform subsampling with RJKDE method had larger interval widths, while Targeted subsampling and No subsampling had the smallest. The Bayesian method, despite its longer computation time, achieved a reasonable balance between accuracy and interval width.

Method	**Time (sec.)**	**RMSE**	**Coverage**	**Complete Coverage**	**Interval Width**
Targeted subsampling (RJKDE)	4.03	0.08	0.91	0.24	0.04
Uniform subsampling (RJKDE)	4.39	0.17	0.92	0.27	0.06
No subsampling (RJKDE)	16.11	0.08	0.89	0.08	0.04
Uniform subsampling (Bayesm)	23.21	0.14	0.43	0.07	0.05

Table 3.1: Comparison of Density Estimation Methods in Simulated Data Analysis

The dataset, comprising $N = 5000$ observations, is constructed through a mixture of normal distributions involving four distinct groups. The first row employs Targeted subsampling of size $\tilde{n} = 500$. The second and fourth row are created utilizing Uniform subsampling, again of size $\tilde{n} = 500$. Lastly, the third represents the entire dataset. We implemented a simulation loop with 50,000 iteration.

Chapter 4

MODEL-BASED CONTIGUOUS COVARIATE PARTITIONING

So far, this dissertation has been concerned with estimating density functions nonparametrically using Gaussian mixture models. Often, it is of interest to examine conditional density functions in an attempt to understand the factors that drive changes in the density. In our motivating application, we are interested in understanding what factors drive changes in the birthweight distribution, for example, smoking or mother's age.

In the case of categorical features, one could simply perform conditional density estimation separately for all the distinct feature combinations. However, some feature combinations are quite rare relative to others and the question naturally arises whether or not two or more feature combinations can be "collapsed" into a single grouping that shares a common conditional density. This chapter develops a Bayesian Markov chain Monte Carlo algorithm for automatically estimating the conditional densities while allowing for distinct covariate combinations to arise from a common conditional density function. Such an approach is in between a completely disaggregated analysis and an entirely marginal (aggregated) analysis and the degree of partial aggregation is data-driven.

Note that this idea of partial aggregation is also the driving idea behind the widely-used classification and regression tree (CART) method [1], or just "decision trees" for short. In terms of previous work, the method presented here is most similar to a classification tree model that partitions the data according to decision trees using the covariates. Decision trees represent a heuristic that limits the clusterings that can be used. The clustering approach described here is more flexible. In par-

ticular, the clustering method presented here should be capable of discovering "deep interactions" that would be difficult for a decision tree to discover. Decision trees are constructed by recursively splitting data based on feature values, leading to a hierarchical tree structure. In other words, the search for a good-fitting partition is conducted top-down in the sense that the data is initially fully aggregated and is gradually (recursively) partitioned into smaller and smaller subpartitions by the addition of decision rules using a single covariate feature at a time. The inherent statistical question posed by decision trees is therefore "when to stop disaggregating?" In particular, the folk-wisdom surround fitting decision trees holds that one should grow the trees "too deep" and then "prune" them back to a smaller size.

By comparison, the proposed contiguous partition algorithm developed here proceeds in a bottom-up fashion, starting from fully disaggregated partition (a maximal partition) and merging clusters together if the data suggests that they arose from similar distributions. The main technical obstacle to such an approach is determining which clusters can be merged, because not all feature combinations can be merged sensibly. For interpretability, it is desirable that the partition of the covariate space not aggregate feature combinations that are too dissimilar. In other words, we want to only collapse feature combinations that are "contiguous" in a certain mathematical sense which we describe in detail below.

4.1 Contiguous covariate partitioning

To illustrate the idea with a concrete example, consider a dataset with two binary features. We may depict the feature state space as a square with four nodes, each representing a combination of feature states. If we label these nodes as **1**, **2**, **3** and **4**, as shown in Figure (4.1), we can picture different clustering possibilities within this space. As we consider larger clusters, the number of valid clustering options increases,

demonstrating the scalable and intricate nature of cluster formation for dichotomous features. A valid cluster in this space is made up of connected nodes—combinations of data points that share an edge on the graph.

Let's look at an example where our dataset contains two dichotomously-valued features, thus creating a state space for clustering. In this example, there are two binary features, therefore, each combination of the features can be represented as the node of the square with 4 nodes and 4 edges since there are 2^p number of unique combinations of the features. Lets name each node as **1**, **2**, **3** and **4** as depicted in the Figure (4.1).

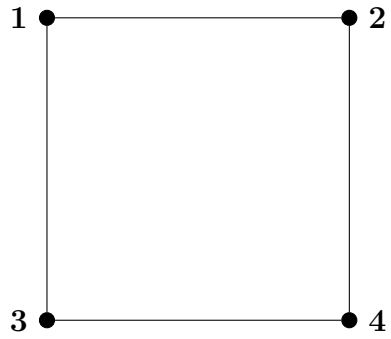

Figure 4.1: State Space Representation of 2 Dichotomous Valued Features Example

Hence, our state space for valid clustering configurations is as follows:

Size of Cluster	**Valid Clustering**
1	{**1**,**2**,**3**,**4**}
2	{{**1**,**2**}, {**3**,**4**}}, {{**1**,**3**}, {**2**,**4**}} , {{**1**}, {**2**,**3**,**4**}}, {{**2**}, {**1**,**3**,**4**}} , {{**3**}, {**1**,**2**,**4**}} , {{**4**}, {**1**,**2**,**3**}}
3	{{**1**}, {**2**}, {**3**,**4**}}, {{**1**}, {**3**}, {**2**,**4**}} {{**2**}, {**4**}, {**1**,**3**}}, {{**3**}, {**4**}, {**1**,**2**}}
4	{{**1**}, {**2**}, {**3**}, {**4**}}

Table 4.1: Valid Clustering Configurations for 2 Dichotomous Valued Features Example

As we consider larger clusters, the number of valid clustering options increases, demonstrating the scalable and intricate nature of cluster formation for dichotomous features. A valid cluster in this space is made up of connected nodes—combinations of data points that share an edge on the graph.

Not every potential cluster is included in our state space. For instance, a valid cluster must have at least one data point; empty clusters don't provide useful insights. Some groups may be excluded if they don't consist of all the elements in the dataset, as our aim in clustering is to ensure every data point belongs to a cluster. Additionally, configurations like $\mathbf{1}, \mathbf{4}, \mathbf{2}, \mathbf{3}$ aren't seen as valid because they break the principle of connectedness—every node within a cluster should be reachable from any other through the edges connecting them. The decision to limit the state space in this way focuses our analysis on the most likely insightful clusters, enhancing computational efficiency. Defining these boundaries ensures that our clustering process is purposeful and reflects the specific objectives of our research.

In the motivating application, seven dichotomous features are considered, leading to a total of 128 unique combinations of these features, and so 128×2 = 256 parameters. This model can be thought of as a "saturated" interaction model where there are 128 "cells" each with distinct parameters. Although the birthweight dataset is massive, some of those 128 cells do not have many data points. Therefore, the statistical question addressed in this chapter is how data from the other cells can help to estimate the parameters in the low-sample cells. Specifically, the clustering approach that will be developed assumes that some of those 128 cells can be collapsed into a single cell in the sense of having the same underlying parameters.

The computational challenge with this model is that the number of ways to possibly collapse the cells is enormous. With only seven features there are 128 unique cells

and the number of ways to cluster those 128 unique cells is overwhelming. Therefore exhaustive search is infeasible and a stochastic search procedure is preferred.

Consider the following example. Suppose a model's training set has 3 dichotomous valued features. Lets consider gender of the baby, martial status of the mother and if mother is a smoker or not. Therefore, **Boy (B)**, **Married (M)** and **Smoker (S)** are three features in the training set. Since, there are 3 features i.e., $p = 3$, since, there will be 8 number of unique combinations of the features in this setting. These 8 unique combinations will define our state space for clustering, allowing us to explore different groupings of data based on these binary feature values, as illustrated in Figure (4.2).

When we're looking to combine clusters, we focus on grouping those that are similar or related. The principle is straightforward: we can only merge clusters if they directly connect, meaning they share common features or characteristics. For instance, a cluster that represents married mothers who smoke and have a baby boy **(M, S, B)** logically aligns with the cluster for mothers who are unmarried **(M')**smokers with a baby boy because they share two out of three features. However, merging the cluster of married mothers who smoke with a cluster of unmarried non-smokers **(S')** with baby girls **(B')** would not be suitable as they have no direct connection or shared edge, and thus, represent starkly different combinations.

This method ensures that we merge clusters that make sense together, reflecting more accurate and meaningful relationships within our data. The use of this clustering technique allows for a nuanced analysis that takes into account the direct connections between data points based on their shared attributes.

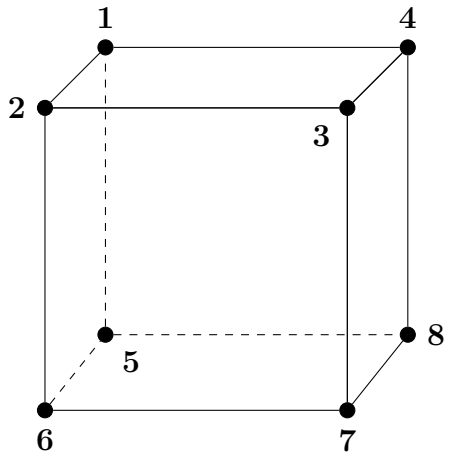

Figure 4.2: State Space Representation of 3 Dichotomous Valued Features Example

Node	y	Node	y
1	(B, M, S)	**5**	(B', M, S)
2	(B, M, S')	**6**	(B', M, S')
3	(B, M', S)	**7**	(B', M', S)
4	(B, M', S')	**8**	(B', M', S')

Table 4.2: Representation of Each Node for 3 Dichotomous Valued Features Example

Since we have an undirected graph without loops and multiple edges such that each node of the graph represents the unique combination of the features. Let's name the unique combination of the feature as "nodes". Now we sort out the nodes in a order such that we can construct the adjacency matrix. Adjacency matrix allows for lookups of whether two nodes are connected especially in case of graphs with many edges. Construct the list of the subsets of nodes named "clusters" which contains information about its members, neighbors, and size. Initialize at each variable at one cluster and store it as a "cluster membership". Therefore, each cluster has only one member where neighbors are those nodes which share the edge with its members.

Now, each cluster is merged with another one using Markov Chain Monte Carlo

(MCMC) sample over the state space of the valid clustering which is explained further in the next section.

4.2 Algorithm

We will present our algorithm as a two-step Gibbs sampler, alternating between updating the partition P and the parameters within each partition Θ. Our algorithm is similar to the Markov chain methods for redistricting discussed by in [4]; however, those authors consider partitions of a fixed size whereas we consider partitions of any size.

Our data will be a length n vector $Y = (y_1, y_2, \ldots, y_n)$. The covariate information will be recorded generically as a graph. Let V be a set indicating each unique combination of features. In the example above, with three dichotomous features, $V = \{1, \ldots, 2^3\}$. As in that example, a graph will be used to describe which feature combinations are considered similar. Specifically, a graph $G = (V, E)$, where V are vertices, or *nodes* and E is the set of all the edges that connect any two vertices in the graph. In our earlier example, E consisted of all pairs (v, v') for $v \in V$ and $v' \in V$ where the Hamming distance between v and v' in their bit-string representation was 1. The edge set could be defined in other ways depending on the applied problem context.

A partition P is a grouping of V into (non-empty) subsets, where every element (vertex) is included in exactly one subset. In symbols, $P = \{V_1, \ldots, V_d\}$ where $V = \bigcup_{j=1}^{d} V_j$ and $d = |P|$. Each V_j is referred as a block, cell, or cluster in different papers; we will refer to them as clusters.

At the heart of our algorithm is a random walk over possible partitions. Accordingly, we need to define a process for moving from one partition P to another P'. To do this, we will take a node-centric approach, meaning that P and P' will differ by

the allocation of a single-node, v. Then, selecting P' is a two step process. In the following notation, let $v = v(P, P')$ convey the identity of the node by which P and P' differ.

First, we select a node from V that is "moveable", starting from the partition P. Not all nodes can be relocated starting from partition P because doing so would render the cluster from which it exited invalid in the sense that the vertices in the cluster do not comprise a single connected component, violating our desired contiguity property.

When evaluating a specific vertex $v \in V$ for "moveability" we therefore look at the subgraph that is formed by that vertex and others in the same cluster according to P, which we denote as $[v]^P$. Let $c(G)$ denote the number of connected components, within any given graph. So, when we talk about $c([v]^P \backslash v) = 1$, what we mean is that if we were to remove the vertex v from this induced subgraph, we would still have a single connected component. This suggests that v is not a critical connection point in this subgraph; despite its removal, all vertices remain reachable from any other by traversing edges.

Therefore, let the quantity $m(P)$ reflects the number of such moveable points across the graph G for the given partition P:

$$m(P) = |v \in V \mid c([v]^P \backslash v) = 1|. \tag{4.1}$$

A node v is randomly selected from the "moveable nodes" allowed by partition P, each one having probability $1/m(P)$ of being selected. Observe that knowing P and P' uniquely identifies the node v by which they differ, but knowing P and v does not yet identify P' because we have yet to specify where v was moved to in the new partition.

In the second step, a list is created by finding nodes in the graph that are not

already part of $[v]^P$, but adjacent to it. This list is a"destination list" which defines the clusters to which v can be validly relocated. One of the valid destinations for "moved node" can be an empty cluster (unless that node was alone to begin with in P). A destination is randomly drawn from the "destination list" with probability $1/a(P, v)$ where

$$a(P, v) = \left|v' \in V \mid (v, v') \in E \bigcap v' \notin [v]^P\right|. \tag{4.2}$$

We will use the notation $a(P, P')$ to mean $a(P, v(P, P'))$ where it is convenient. This equation counts the number of vertices v' that fulfill both of the following conditions: v' is connected to v by an edge, and v' is not in the same partition block as v within the graph structure prescribed by partition P.

This two-step process yields a new partition, P' according to a straightforward probability, the so-called "forward jump" probability. To compute this probability requires one more term. Given v, the number of possible ways $w(P, P')$ to reach P' from P can be written as

$$w(P, P') = \left|\tilde{v} \in V \mid (\tilde{v}, v) \in E \bigcap \tilde{v} \in [v]^{P'}\right|. \tag{4.3}$$

Putting everything together, the probability of moving from current partitioning P to new partition P' is

$$\Pr(P \to P') \equiv \Pr(P' \mid P) = \Pr(P' \mid v, P)\Pr(v \mid P) = \frac{w(P, P')}{a(P, P')}\frac{1}{m(P)} \tag{4.4}$$

where v is the "moved node". Note that $\Pr(P' \mid P) = \Pr(P' \mid v, P)\Pr(v \mid P)$ because $v = v(P, P')$ is uniquely determined by P and P', so $\Pr(P', v \mid P) = \Pr(P' \mid P)$.

The reverse jump probability involves inverting the process, considering P' as the starting point. That is, for the "moved node" v, the reverse jump probability can be written as

$$\Pr(P' \to P) = \frac{w(P', P)}{a(P', P)}\frac{1}{m(P')}. \tag{4.5}$$

4.2.1 Computing graph connectivity

A key step in the algorithm above is determining if a (sub)graph has a single connected component [11][34]. To determine this, we compute a quantity from spectral graph theory called the Fiedler value. This section defines the Fiedler value and explains how it is informative about graph connectivity.

First, let D be the degree matrix, which is a d-by-d diagonal matrix with each element on the diagonal recording the degree of the associated vertex (where $d = |V|$). Recall that the degree of a vertex is the number of edges incident upon it. The adjacency matrix, A, is a symmetric zero-one matrix such that element $a_{v,v'} = 1$ if and only if $(v, v') \in E$. Finally, the Laplacian matrix is defined as the difference between the degree matrix and the adjacency matrix, $L = D - A$.

Notice that by definition $DJ = AJ$, where J is the column vector of all ones. This is because the degree is, by definition, the sum of the corresponding row of the adjacency matrix. Accordingly, J is an eigenvector of L with eigenvalue 0: $LJ = DJ - AJ = 0 \cdot J$. More importantly, it is easy to show that

$$h^t L h = \frac{1}{2} \sum_{i=1}^{d} \sum_{j=1}^{d} a_{i,j} (h_i - h_j)^2,$$

for any function $h : V \to \mathbb{R}$ that assigns values to nodes; that is $h_i = h(v_i)$, where we have numbered the nodes from $i = 1, \ldots, d$ for convenience. Consequently, if a particular function h has eigenvalue 0, then $h^t L h = 0$, because $h^t(0 \cdot h)$, which can only happen when connected vertices (which have $a_{i,j} = 1 \neq 0$) have identical function evaluations. In other words, for eigenvector h with eigenvalue 0, $a_{i,j} = 1$ implies $h_i = h_j$. By transitivity of equality, that means that any two nodes that are reachable by a path must have the same evaluation.

In a graph consisting of a single connected component, there is only one connected component indicator vector, represented by the constant one vector. For graphs with

multiple connected components, the adjacency matrix and Laplacian matrix L take a block diagonal form, each block having its own Laplacian matrix. These sub-matrices essentially serve as proper graph Laplacians for their respective components. Since L is structured as a block diagonal matrix, its set of eigenvalues becomes a union of the spectra of its individual blocks. Furthermore, the eigenvectors of L are constructed by amalgamating the eigenvectors of its blocks, with zeros in the positions corresponding to other blocks. Consequently, the matrix L exhibits as many eigenvalues equal to 0 as there are connected components in the graph, and the corresponding eigenvectors serve as indicator vectors for these components.

The Fielder value is defined as the second eigenvector of the Laplacian. Therefore, if the Fiedler value is non-zero, then there is only a single connected component. In our application, we consider the induced graph obtained from a partition P by removing vertex v, (denoted $[v]^P/v$) and compute its Fiedler value using singular-value decomposition. If that value is non-zero, then there is only a single connected component and it is safe to remove v. Conversely, if it is zero, then removing v violates our contiguity requirement and v is not able to be moved. In practice, exact zeros are sometimes not obtained when they ought to be, and so instead we determine connectivity using a small non-zero threshold (denoted ϵ in our pseudocode and set to 0.001 in our implementation).

4.2.2 Likelihood evaluation

With the basic random walk reversible jump moves established, what remains is to consider how the data informs the inferred partition. For that, all this is necessary is to weight the acceptance probabilities by the appropriate likelihood evaluations.

Denote the likelihood of the partition P, given fixed parameter values Θ, as

$$\mathcal{L}(Y, P) = \prod_{j=1}^{|P|} \prod_{i|v_i \in V_j} \mathcal{L}(y_i; \theta_j). \tag{4.6}$$

For the present purposes, to illustrate the contiguous partition algorithm, we will use a *parametric* model. The eventual goal is to combine this algorithm with the algorithms from the previous two chapters to do nonparametric clustering. For now, we consider a Gaussian likelihood, which permits us to integrate out an unknown (but shared) mean parameter and determine the acceptance probabilities using an integrated likelihood. Specifically, we use the integrated likelihood from the BART literature [18](shown here on the log-scale):

$$\log\left(\prod_{i|v_i \in V_j} \mathcal{L}(y_i; \theta_j)\right) = -n_j \log(\sigma_j) - \frac{1}{2}\log(1 + n_j) - \frac{1}{2}\frac{\tilde{s}_j}{\sigma_j^2} + \frac{1}{2}\frac{s_j^2}{(\sigma_j^2(1 + n_j))} \tag{4.7}$$

where n_j denotes the number of terms in the product (the number of observations in cluster j), $s_j = \left(\sum_{i|v_i \in V_j} y_i\right)$ and $\tilde{s}_j = \left(\sum_{i|v_i \in V_j} y_i^2\right)$.

Finally, the acceptance ratio from current cluster P to proposed cluster P' can be written as

$$\alpha(P, P') = \frac{\Pr(P' \to P)\mathcal{L}(Y, P')}{\Pr(P \to P')\mathcal{L}(Y, P)}. \tag{4.8}$$

Based on probabilities and a random draw, decide whether to accept or reject the proposed move.

4.2.3 Updating cluster parameters

If the proposed move is accepted, the cluster assignments are updated accordingly and update the standard deviation for each cluster j using

$$\sigma_j^{-2} \mid P \sim \text{Gamma}\left(\alpha + \frac{n_j}{2}, \beta + \frac{\sum_{i|v_i \in V_j}(y_i - \bar{y}_j)^2}{2} + \frac{\nu_0 n_j (\bar{y}_j - \nu_1)^2}{2(\nu_0 + n_j)}\right), \tag{4.9}$$

where $\bar{y}_j = \frac{1}{n_j} \sum_{i|v_i \in V_j} y_i$ and n_j is the number of observations in cluster j, which is the conjugate update for a precision parameter in a normal model with an unknown mean and hyperparameters ν_0 and ν_1.

It is important to note that in practice, these probabilities are often converted into logarithms in order to achieve more numerically consistent calculations, particularly when working with large data sets and intricate models. This Metropolis-Hastings algorithm, combined with these probability calculations, allows us to efficiently explore the state space of valid clusterings and make informed decisions about cluster assignments.

4.2.4 Dirichlet-Multinomial Likelihood

Instead of likelihood from the BART, we can use the Dirichlet-multinomial likelihood for modeling count data.

The Dirichlet-multinomial likelihood combines two key concepts: the Dirichlet distribution and the multinomial distribution. The multinomial distribution extends the binomial distribution to handle multiple outcomes, such as predicting the probabilities of different colors of balls drawn from a bag. However, in our birthweight dataset, the exact probabilities of these outcomes are unknown and can vary. The Dirichlet distribution models this uncertainty, representing a range of possible probabilities. By combining these two distributions, the Dirichlet-multinomial model allows us to update our prior beliefs with observed data.

Consider the vector of bin counts as $\mathbf{n}$ and the vector of Dirichlet parameters as $\boldsymbol{\alpha}$ such that the total number of trials can be written as N. The dirichlet part can be represented as (on the log-scale)

$$\log\Gamma\left(\sum_{i|v_i\in V_j}\alpha_i\right)-\log\Gamma\left(N+\sum_{i|v_i\in V_j}\alpha_i\right),$$

whereas the multinomial part is

$$\sum_{i|v_i\in V_j}\log\Gamma(n_i+\alpha_i)-\sum_{i|v_i\in V_j}\log\Gamma(\alpha_i).$$

By combining these two parts, the Dirichlet-multinomial log-likelihood can be written as:

$$\log\left(\prod_{i|v_i\in V_j}\mathcal{L}(y_i;\theta_j)\right)=\log\Gamma\left(\sum_{i|v_i\in V_j}\alpha_i\right)-\log\Gamma\left(N+\sum_{i|v_i\in V_j}\alpha_i\right)+\sum_{i|v_i\in V_j}\log\Gamma(n_i+\alpha_i)-\sum_{i|v_i\in V_j}\log\Gamma(\alpha_i).$$

4.3 Small illustration

To illustrate the new clustering algorithm we give a small example using synthetic data. The synthetic data is generated using a single Gaussian within each cluster. The dataset includes two categorical control variables, x_1 and x_2. These variables represent categorical features with three levels (0, 1, and 2), with different probabilities assigned to each level. We assign an index to each data point based on the combination of x_1 and x_2 values, for a total of 9 unique combinations. However, the response variable is then generated in just 3 clusters. Specifically, if both x_1 and x_2 are 0, the index is set to 1; if they are both 2, the index is set to 3; otherwise, it is set to 2. The total number of observations N is 1500. One thousand iterations of the clustering MCMC are then performed. One hopes to see that the most common cluster configuration puts $x_1 = x_2 = 0$ together, $x_1 = x_2 = 2$ together, and any other covariate combination all together in a common cluster.

Figure (4.3a) shows a 9-by-9 heatmap depicting how often each of the nine nodes is included in the same cluster as each of the other eight nodes. As we hoped, the matrix has high probability in the center 7-by-7 block, indicating that those 7 nodes

Algorithm 2 A reversible jump MH update on graph partitions

procedure UPDATECLUSTER(Y, A, P, Θ)

Input: Data Y; $m \times m$ adjacency matrix A; partition P; parameters Θ.

Output: New partition P'

`Movers` $\leftarrow$ MOVEABLE(A, P)

$m(P) \leftarrow |$`Movers`$|$.

Draw v uniformly at random from `Movers`.

`Inviters` $\leftarrow \{v' \in V \mid A_{v,v'} = 1 \cap v' \notin [v]^P\}$.

$a(P, v) \leftarrow |$`Inviters`$|$.

Complete the draw of P' by drawing v' uniformly at random from `Inviters`.

Note that (P, v, v') together fully specify P'.

`SharedNeighbors` $\leftarrow \{\tilde{v} \in V \mid A_{\tilde{v},v} = 1 \cap \tilde{v} \in [v]^{P'}\}$.

$w(P, P') \leftarrow |$`SharedNeighbors`$|$.

Compute the likelihoods $\mathcal{L}(Y, P, \Theta)$ and $\mathcal{L}(Y, P', \Theta)$.

Compute the acceptance ratio: $\alpha(P, P') = \frac{\Pr(P' \to P)\mathcal{L}(Y,P',\Theta)}{\Pr(P \to P')\mathcal{L}(Y,P,\Theta)}$.

where $\Pr(P \to P') = \frac{w(P,P')}{a(P,P')}\frac{1}{m(P)}$ and $\Pr(P' \to P) = \frac{w(P',P)}{a(P',P)}\frac{1}{m(P')}$.

Draw $U \sim$ Uniform(0,1).

if $U < \alpha(P, P')$ **then** $P \leftarrow P'$.

end if

Return P.

end procedure

Algorithm 3 Determine which nodes can be relocated in partition P.

function MOVEABLE(A, P) Input: $m \times m$ adjacency matrix A; partition P.

Output: $S \subset \{1, \ldots, m\}$ indicating moveable nodes.

    $S \leftarrow \varnothing$.

    for j from 1 to m **do**

    $\kappa_j \leftarrow$ the Fiedler value of $[v_j]^P / v_j$.

        if $\kappa_j > \varepsilon$ **then**

    $S \leftarrow S \cup v_j$.

        end if

    end for

end function

Algorithm 4 Outer Monte Carlo loop for contiguous covariate partitioning.

procedure MAINMCMC(Y, A)

    Input: Data Y, $m \times m$ adjacency matrix A

    Output: M posterior samples of partitions P and parameters Θ.

    Initialize P and Θ.

    for M iterations **do**

        UPDATECLUSTER(Y, A, P, Θ).

        UPDATEPARAMETERS(Y, P, Θ).

    end for

end procedure

are often combined. Meanwhile, the top left and bottom right components are usually not associated either with the other seven nor with each other, indicating that we successfully recover the true clustering of the data generating process.

Similarly, Figure (4.3b) depicts the posterior mode cluster assignment, with the same shade indicating common assignment. Clearly there are three clusters, again corresponding to the $x_1 = x_2 = 0$ node, the $x_1 = x_2 = 2$ node, and the remainder.

(a) (b)

Figure 4.3: Illustration of Clustering Method

The figure (4.3a) illustrates the 9-by-9 heatmap depicting how often each of the nine nodes is included in the same cluster as each of the other eight nodes. The prominent probabilities within the central 7-by-7 block suggest these seven nodes are frequently clustered together, while the top-left and bottom-right nodes are typically excluded from this grouping, reflecting an accurate reconstruction of the original data's clustering pattern. The figure (4.3b) depicts the posterior mode cluster assignment, with the same shade indicating common assignment.

4.4 Empirical illustration

Within birthweight dataset, initially we combine the data based on each unique combination of features and calculates various statistics for the logarithm of the birth weights, such as the total, the sum of squares, and the variance. Then, we defined weight categories using a series of cut points from 0 to 2.5 kgs, in increments of 0.25 kgs. For each weight range, count how many data points fall into that specific range. These counts are done separately for each unique combination of features. The clustering method is applied to these weight categories using the Dirichlet-multinomial likelihood.

In Figure (4.4), we plot the estimated density curves of birth weight categories for each identified cluster, with each cluster represented by a uniquely colored line.

Since, in the birthweight dataset, we have 7 features, there are 128 unique combination of features. From 128 variables, we estimated to group them into 28 clusters after cluster analysis. The birth weight dataset includes seven features, which results in 128 possible unique combinations of these features. After performing cluster analysis on these combinations, we grouped them into 28 distinct clusters. Each cluster combines different unique combinations of the original features.

Upon examining these clusters, we found that each cluster groups together various combinations of features, indicating the diversity of variable interactions within the clusters. However, understanding and representing these complex interactions fully remains an ongoing task. Future work will focus on extracting more detailed information from these clusters to better understand their implications and relationships within the dataset.

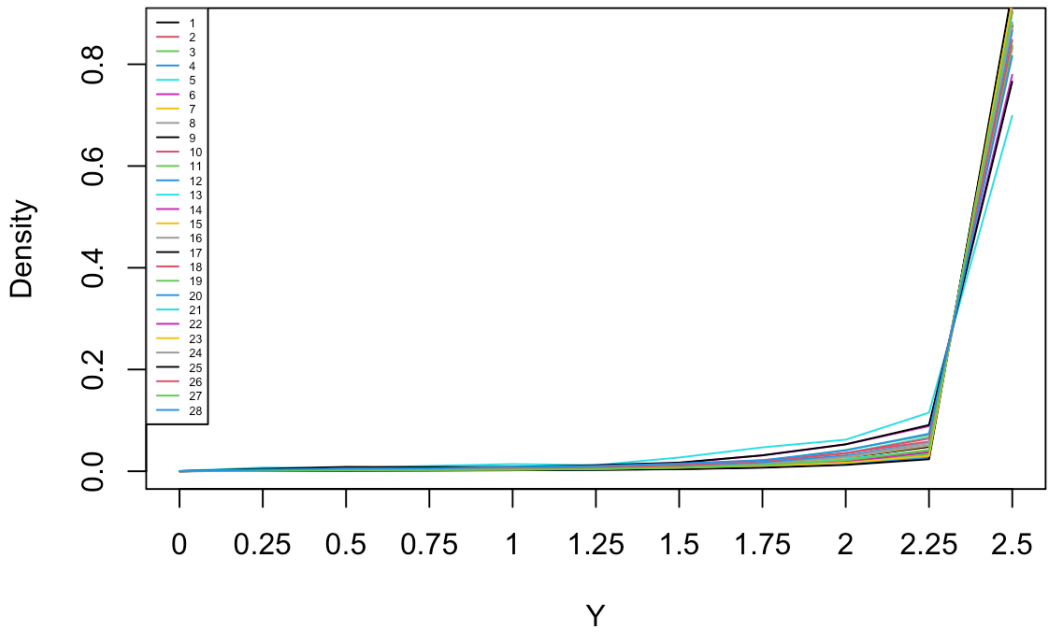

Figure 4.4: Illustration of Clustering method on Birthweight dataset

The estimated density curve using cluster method for each cluster after cluster analysis.

REFERENCES

[1] Breiman, L., J. H. Friedman, R. A. Olshen and C. J. Stone, *Classification and Regression Trees* (Chapman & Hall, 1984).

[2] Chib, S. and E. Greenberg, "Understanding the metropolis-hastings algorithm", The American Statistician **49**, 4, 327–335 (1995).

[3] Chib, S. and I. Jeliazkov, "Marginal likelihood from the metropolis–hastings output", Journal of the American Statistical Association **96**, 453, 270–281 (2001).

[4] DeFord, D., M. Duchin and J. Solomon, "Recombination: A family of markov chains for redistricting", Harvard Data Science Review **3**, 1, 3 (2021).

[5] Diebolt, J. and C. P. Robert, "Estimation of finite mixture distributions through bayesian sampling", Journal of the Royal Statistical Society: Series B (Methodological) **56**, 2, 363–375 (1994).

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

[7] Durrett, R., *Probability: theory and examples*, vol. 49 (Cambridge university press, 2019).

[8] Escobar, M. D. and M. West, "Bayesian density estimation and inference using mixtures", Journal of the american statistical association **90**, 430, 577–588 (1995).

[9] Fan, J., "Design-adaptive nonparametric regression", Journal of the American statistical Association **87**, 420, 998–1004 (1992).

[10] Ferguson, T. S., "A bayesian analysis of some nonparametric problems", The annals of statistics pp. 209–230 (1973).

[11] Fiedler, M., "Algebraic connectivity of graphs", Czechoslovak mathematical journal **23**, 2, 298–305 (1973).

[12] George, E. I. and R. E. McCulloch, "Approaches for bayesian variable selection", Statistica Sinica pp. 339–373 (1997).

[13] Ghosh, J. K. and D. B. Dunson, "Adaptive bayesian density regression", Journal of the American Statistical Association **110**, 511, 1622–1633 (2015).

[14] Green, P. J., "Reversible jump markov chain monte carlo computation and bayesian model determination", Biometrika **82**, 4, 711–732 (1995).

[15] Green, P. J., "Trans-dimensional markov chain monte carlo", Methods of Statistical Analysis and Simulation **83**, 179–197 (2003).

[16] Härdle, W., *Applied nonparametric regression*, no. 19 (Cambridge university press, 1990).

[17] Hastings, W. K., "Monte carlo sampling methods using markov chains and their applications", (1970).

[18] He, J., S. Yalov and P. R. Hahn, "Xbart: Accelerated bayesian additive regression trees", in "The 22nd International Conference on Artificial Intelligence and Statistics", pp. 1130–1138 (PMLR, 2019).

[19] Hinton, G. E., "The gibbs sampler", in "Readings in Machine Learning", edited by J. W. Shavlik and T. G. Dietterich, pp. 431–438 (Morgan Kaufmann, 1992).

[20] Kulp, C. and R. Carroll, "Bayesian density regression: an overview and selected novel extensions", Journal of Computational and Graphical Statistics **26**, 2, 297–310 (2017).

[21] Li, B. and M. A. Clyde, "Bayesian variable selection in quantile regression", Journal of Computational and Graphical Statistics **27**, 3, 540–549 (2018).

[22] Li, K.-H., "Pool size selection for the sampling/importance resampling algorithm", Statistica Sinica pp. 895–907 (2007).

[23] Lopes, H. F. and M. West, "Bayesian model assessment in factor analysis", Statistica Sinica pp. 41–67 (2004).

[24] Metropolis, N., A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller and E. Teller, "Equation of state calculations by fast computing machines", The journal of chemical physics **21**, 6, 1087–1092 (1953).

[25] Nations, U., *United Nations Handbook 1950* (United Nations, 1950).

[26] PARK, B. and B. TURLACH, "Practical performance of several data driven bandwidth selectors", , 1992005, URL `https://ideas.repec.org/p/cor/louvco/1992005.html` (1992).

[27] Park, B. U. and J. S. Marron, "Comparison of data-driven bandwidth selectors", Journal of the American Statistical Association **85**, 409, 66–72, URL `https://www.jstor.org/stable/2289526` (1990).

[28] Richardson, S. and P. J. Green, "On bayesian analysis of mixtures with an unknown number of components (with discussion)", Journal of the Royal Statistical Society Series B: Statistical Methodology **59**, 4, 731–792 (1997).

[29] Robert, C. P. and G. Casella, *Monte Carlo Statistical Methods*, vol. 5 (Springer, 2004).

[30] Roberts, S. J., D. Husmeier, I. Rezek and W. Penny, "Bayesian approaches to gaussian mixture modeling", IEEE Transactions on Pattern Analysis and Machine Intelligence **20**, 11, 1133–1142 (1998).

[31] Rossi, P., M. P. Rossi, I. Rcpp, L. Rcpp, B. Regression, B. S. U. R. SUR, M. L. MNL, M. P. MNP and N. B. P. Regression, "Package 'bayesm'", (2015).

[32] Rubin, D. B., "Comment: a noniterative sampling/importance resampling alternative to the data augmentation algorithm for creating a few imputations when fractions of missing information are modest: the sir algorithm", Journal of the American Statistical Association **82**, 398, 542–543 (1987).

[33] Silverman, B. W., *Density Estimation for Statistics and Data Analysis* (Chapman & Hall/CRC, 1986).

[34] Von Luxburg, U., "A tutorial on spectral clustering", Statistics and computing **17**, 395–416 (2007).

[35] Wu, T. L. and Y. Chen, "Semiparametric density regression with nonparametric mixing", Journal of the American Statistical Association **105**, 491, 940–954 (2010).