SCALABLE MONTE CARLO INFERENCE IN REGRESSION MODELS WITH MISSING DATA

by

Didem Koçhan

B.S., Electronics Engineering, Sabancı University, 2015

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Industrial Engineering

Sabanci University

2018

SCALABLE MONTE CARLO INFERENCE IN
REGRESSION MODELS WITH MISSING DATA

APPROVED BY:

Assist. Prof. Dr. Sinan Yıldırım .....
(Thesis Supervisor)

Prof. Dr. Ş. İlker Birbil .....
(Thesis Co-supervisor)

Prof. Dr. Ali Taylan Cemgil .....

Assoc. Prof. Dr. Kemal Kılıç .....

Assist. Prof. Dr. Hüseyin Özkan .....

DATE OF APPROVAL: 31.07.2018

ACKNOWLEDGEMENTS

I would like to express my very profound gratitude to my thesis advisor Assist. Prof. Sinan Yıldırım for his enlightening technical knowledge and his patient guidance throughout this study.

I am grateful to my co-advisor Prof. İlker Birbil for his support, motivation and positive attitude which has always inspired me. He has been a great mentor and role model to me with his immense knowledge and enthusiasm.

A very special gratitude goes to my best friend, my non-academic life advisor Görkem who has always been my comfort zone. He helped me a lot with his understanding and companionship during this stressful period of my life. I am also grateful to Elif, my "Prozac" who provided me her endless patience. She was the one who has taken me off my emotional roller coaster and put me the correct path.

I also would like to thank my friends: Ahmet, who wisely leaded me with his experiences, Alihan, who has provided me any kind of motivation and introduced me lots of great songs that I listened when I am working on this thesis, Alper and Apo, for sharing all happiness and sorrow throughout years and opening the doors of electronics lab, Begüm, for understanding me without any words, my event mate Can, for buying our tickets and my roommate Yelda, for giving me fashion advices, and also Adil, Baran, Başak, Oytun and Selin for all their support.

Last but not the least, I am deeply grateful to my family, especially my mother, my first teacher whose compassionate upbringing and nurturing have made me what I am today and brought me where I am today.

This thesis is all about the methods of handling missing components, and I am 100% sure that in the literature, there is no way to replace one of you people. If one of you was missing, I would stay incomplete.

ABSTRACT

SCALABLE MONTE CARLO INFERENCE IN REGRESSION MODELS WITH MISSING DATA

Markov chain Monte Carlo (MCMC) and Stochastic Gradient Langevin Dynamics (SGLD) algorithms comprise a basis for this thesis. These methods are studied in detail and combined for handling incomplete and large datasets. Two algorithms, which are based on Metropolis-Hastings (MH) and SGLD, are proposed to improve the performance of regression with missing data.

We introduce an SGLD algorithm for large datasets with missing portions. The algorithm approximates the gradient of the log-likelihood of a subset of the data with respect to the unknown parameter by using samples for missing components obtained with MH moves.

We implemented these methods for a logistic regression model to obtain parameter estimations. We worked with two different datasets with missing features and compared their performances. The first dataset is artificially generated from a logistic regression model where the features are normally distributed, whereas the second dataset is a real categorical data.

ÖZET

EKSİK VERİ İÇEREN REGRESYON MODELLERİ İÇİN ÖLÇEKLENEBİLİR MONTE CARLO ÇIKARIMI

Markov zinciri Monte Carlo (MCMC) ve Stokastik Gradient Langevin Dinamikleri (SGLD) algoritmaları bu tez için bir temel oluşturmaktadır. Bu yöntemler, eksik veri içeren ve geniş ölçekli veri setlerinin ele alınması için ayrıntılı olarak incelenip, bir araya getirilmiştir. Büyük ölçekli veri setlerinde eksik verilerle regresyonun performansını iyileştirmek için Metropolis-Hastings ve SGLD temelli iki yeni algoritma geliştirilmiştir.

Eksik kısımlar içeren büyük veri setleri için SGLD algoritması geliştirilmiştir. Bu yöntemde, veri setinin rastgele seçilmiş bir alt kümesi kullanılarak, bilinmeyen parametrelerin logaritmik olasılık türevlerinin yaklaşık değerleri hesaplanmaktadır. Bu yaklaşımlar hesaplanırken, veri içerisindeki eksik bileşenler MH adımları ile tahmin edilmiştir.

Bu metotlar, parametre tahminleri üretebilmek için lojistik regresyon modelleri üzerine uygulanmıştır. Algoritmalar, eksik değişkenler içeren iki farklı veri seti üzerinde denenmiş ve performansları karşılaştırılmıştır. İlk veri seti yapay bir şekilde lojistik regresyon modelinden üretilmiş olup, değişkenler normal dağılımdan gelmektedir, öte yandan ikinci veri seti gerçek ve kategorik bir veridir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

LIST OF SYMBOLS

A	The masking matrix
exp	Power of the natural exponential constant e
log	Natural logarithm
N	Number of samples
$\mathcal{N}(\mu, \sigma^2)$	Gaussian distribution with mean μ and variance σ^2
$p(\cdot)$	Probability density function
t	Iteration index in algorithms
$q(\cdot\|\cdot)$	Conditional density of proposal kernel
Y	The output of models
ϵ	Step size of SGLD algorithm
μ	Mean vector of the probability distribution
η	The noise vector
$\pi_i(\cdot\|\cdot)$	Conditional distribution of i^{th} component
Σ	Co-variance matrix of probability distribution
θ	Parameter vector of logistic regression function

LIST OF ACRONYMS/ABBREVIATIONS

i.i.d	Independent and identically distributed
CCA	Complete Case Analysis
MCMC	Markov chain Monte Carlo
MH	Metropolis-Hastings
PA	Prediction accuracy
SGLD	Stochastic Gradient Langevin Dynamics

1. INTRODUCTION

Missing data [1] is a problem that occurs in almost all empirical research. The main concern is that, if the data were complete, would the results of the research be different [2, 3]? This question does not have an obvious answer, since incompleteness causes a decrease in the performance of parameter estimation and sensitivity of the method. If the missing data pattern is nonrandom, there is an additional concern that arises from bias. In this context, bias results in the failure of observed data to represent the incomplete parts [2]. In recent years, the interest in handling missing data mechanisms has increased and different techniques are introduced [4].

In the literature, there are various strategies to handle missing data problem [5]. The first approach, called listwise deletion (or complete case analysis), proposes to consider only observed variables, and do the calculations through the available parts of the data [6]. Although this idea is easy to implement, it loses the track of the unavailable parts as well as ignores the possible differences between the characteristics of the observed and unobserved parts of the data. The second approach proposes the single imputation idea in which the missing components are replaced by the mean of observed variables. Since single imputation cannot reflect the uncertainty of imputations, Rubin (1987) have introduced the multiple imputation idea, where all possible values for the latent variables are evaluated, and each one of them is used in parameter estimation [2, 5, 7].

Another well-known missing data handling technique is Expectation Maximization (EM). The foundations of EM framework is first laid down by Little and Rubin [8]. In this technique, maximum likelihood estimates are calculated for incomplete data [5]. As the name implies, the EM algorithm consists of two steps: Expectation and Maximization. The first (expectation) step calculates the conditional log-likelihood expectation of the observed data, when the observed data and the first parameter estimations are given, while the second step calculates the maximum log-likelihood of the expectation yielded by the first step, in order to obtain the parameter updates. The

cycle of expectation and maximization continues iteratively, until the convergence is attained [2].

1.1. Motivations and Contributions of the Thesis

This thesis investigates the Bayesian methods for incomplete data problems. Our motivation is to utilize the Metropolis-Hastings (MH) and Stochastic Gradient Langevin Dynamics (SGLD) methods in order to address incomplete data problems in large-scale datasets. The main contribution of this thesis is as follows: We have shown that using gradient estimates instead of the exact sampling methods [9, 10] provides an efficient way to handle missing data. In SGLD framework, a Bayesian learning is iteratively performed from large-scale datastes via small mini-batches. The mini-batches are used to approximate gradients and then the parameter updates are generated using the gradient information. Since the algorithm does not require computations over the whole dataset, a significant amount of time has saved, especially when the data size becomes larger. Another contribution of the thesis is to use MH sampling to replace latent variables in the dataset. MH [11, 12] is an efficient sampling method which enables us to draw samples for the cases where the full conditional distributions are not easy to sample from.

We introduce two approaches to the literature: The first approach uses MH idea for parameter estimation, whereas the second approach uses the SGLD idea. We have compared the performances of the algorithms, as well as we compare these methods with a method that proposes to consider only observed variables. The similar results are obtained by the implementations of three algorithms, and one can prefer one of them considering the advantages and disadvantages of the methods according to her preferences.

1.2. Scope of the Thesis

Chapter 2 provides a theoretical background for the readers. First the sampling problem is stated, then the Bayesian inference and the Markov chain Monte Carlo

methods are introduced. Chapter 3 introduces the incomplete data problem. The existing ways of handling this problem, especially in big datasets, are provided. This chapter concludes by proposing two MCMC-based methods to handle the incomplete data problem in big datasets. In Chapter 4, numerical results of the two proposed algorithms and a listwise deletion (or complete case analysis) algorithm on a Gaussian distributed synthetic incomplete dataset are shown with the methodology. Similarly, in Chapter 5, the proposed methods and complete case analysis method are applied on a real dataset, the methodology and results are also given. Finally, in Chapter 6, the results are discussed and possible future works are suggested.

2. MARKOV CHAIN MONTE CARLO METHODS

Markov chain Monte Carlo (MCMC) methods are mostly used to approximate the probability densities by a finite number of samples. Through this method, one can characterize a distribution even though the mathematical properties and the parameters of the distribution are not completely known. As the name implies, an MCMC method is the combination of Monte Carlo and Markov chain approaches. Monte Carlo is the part where properties of a distribution is estimated by drawing samples and examining them, while Markov chain part provides the memorylessness property in which each new sample depends only on the one before it [13].

An MCMC method depends on an ergodic Markov chain whose stationary distribution is π. If an ergodic Markov chain with stationary distribution π is simulated for long enough time, it will converge to π. This convergence property makes it possible to sample the estimated parameters from a Markov chain with the stationary distribution being the target distribution π [14].

2.1. The Sampling Problem

Let us suppose we have N random samples $X^{(1:N)} = X^{(1)}, X^{(2)}, ..., X^{(N)}$ from a set $\mathcal{X}$. The samples are independent and identically distributed with respect to some unknown probability distribution π. We can notate this as

$$X^{(1)}, X^{(2)}, ..., X^{(N)} \overset{\text{i.i.d.}}{\sim} \pi.$$

Although the probability distribution P is unknown, we can approximately calculate its mean value (the expectation of X) via these samples $X^{(1:N)}$. The expectation can be written as

$$\mathbb{E}_\pi(X) = \int_X x\pi(x)dx, \tag{2.1}$$

where π is also used as the probability density function of π. An approximation for this expression is given as

$$\mathbb{E}_\pi(X) \approx \frac{1}{N}\sum_{i=1}^{N} X^{(i)}. \tag{2.2}$$

We can modify (2.2) for a certain function φ of X with respect to π:

$$\mathbb{E}_p(\varphi(X)) \approx \frac{1}{N}\sum_{i=1}^{N} \varphi(X^{(i)}). \tag{2.3}$$

One can calculate (2.3) without knowing anything explicitly about π. The samples $X^{(1:N)}$ are sufficient to evaluate the integral.

However, if we know about P but we are not given any samples from it, then we cannot calculate the integral in (2.3). In order to handle this problem and estimate (2.3), we can generate our own samples from π. The main idea behind the Monte Carlo methods is that we can avoid the implementation of π if we generate samples from it.

2.2. Bayesian Estimation

Bayesian parameter estimation which gives weight to prior knowledge and reweights it with the available data, is an example where sampling methods are used. In Bayesian estimation the distribution of interest is $\pi(x) = p(x|y)$, the conditional distribution of the unknown parameter given the observed variable y.

$$p(x|y) = \frac{p(x)p(y|x)}{p(y)}, \tag{2.4}$$

Here, $p(x|y)$ is the posterior probability density, $p(x)$ is the prior probability density and $p(y|x)$ is the likelihood function. Since $p(y)$ does not depend on the variable of interest x, in Bayesian literature it is usually neglected and (2.4) can be written as

$$p(x|y) \propto p(x)p(y|x). \tag{2.5}$$

In simple words, the Bayesian idea is

$$\text{posterior distribution} \propto \text{prior distribution} \times \text{likelihood function}$$

The expression in (2.5) consists of the prior density $p(x)$, which is an estimated description of where the parameters are located before the data is analyzed, and the likelihood $p(y|x)$ which represents the modelling of the data [15]. The Bayesian framework states that a posterior distribution that contains all available knowledge about parameters can be constructed when prior information is shaped by the available data, *i.e.* the likelihood function [16]. In this thesis, the likelihood function is chosen from regression models to make parameter estimations from the observed data.

2.3. MCMC Methods

Markov chain Monte Carlo is a generic method to draw samples of x from approximate distributions and correcting these samples to obtain better approximations for the target distribution π. The samples come sequentially, in which the distribution of last sample depends on the previous one. Thus a Markov chain is formed by these samples.

The idea behind MCMC is to construct a Markov chain whose stationary distribution is π and simulate it long enough time that the distribution of current samples converge to the target distribution. The convergence of MCMC methods has been proven in the literature, even though the samples generated from the Markov chain are not independent and identically distributed. MCMC methods are mostly used in applications where there are intractable densities which are approximated by finite number of samples [12].

Let us suppose that the probability distribution of state is denoted by $\pi^{(t)}(x)$ at iteration t. The objective is to build a Markov chain such that $\pi^{(t)}(x)$ converges to the target distribution π, as t goes to infinity. We need to specify a transition probability $T(x';x)$ to define such a Markov chain. The probability distribution of the Markov

chain at iteration $t+1$ is calculated as

$$\pi^{(t+1)}(x') = \int_x \pi^{(t)}(x) T(x'; x) dx. \tag{2.6}$$

There are several requirements that we need to satisfy when designing an MCMC method. The first one is that the desired distribution should be an invariant distribution of the Markov chain. The second condition states that the Markov chain should be ergodic. The chain should be aperiodic and irreducible to provide ergodicity.

One way to ensure the invariance of the target distribution is to show that the detailed balance property holds for the transition kernel. The detailed balance property is given as [17]:

$$T(x_a; x_b)\pi(x_b) = T(x_b; x_a)\pi(x_a), \quad \forall x_a, x_b \in \mathcal{X}.$$

2.3.1. Metropolis-Hastings Method

The Metropolis-Hastings algorithm is a well-known MCMC method devised by Metropolis and Ulam [18] and improved by Hastings [19].

The algorithm uses a Markov transition kernel q on $\mathcal{X}$, in order to propose new values from the old ones. The proposal values, x', are chosen from these simpler proposal distributions, often in the neighborhood of current parameters, x.

The algorithm draws a starting point x_0 from a starting distribution which might be based on an approximation. Then for every iteration t, a new value x' is proposed by sampling from the proposal distribution q, i.e. $x' \sim q(\cdot|x^{(t-1)})$. The proposed value

is accepted as the new value of x, with the acceptance probability $\alpha(x, x')$ where

$$\alpha(x, x') = \min\left\{1, \frac{\pi(x')q(x|x')}{\pi(x)q(x'|x)}\right\}, \quad x, x' \in \mathcal{X}.$$

If the proposal is accepted, then the algorithm sets the value of $x^{(t)}$ as x', and if the proposal is rejected the algorithm sets the value of $x^{(t)}$ as $x^{(t-1)}$. Even if the proposal value is rejected at iteration t, it is still counted as an iteration. Given the current value $x^{(t-1)}$, the transition kernel $q_t(x^{(t)}|x^{(t-1)})$ of the Markov chain is a combination of a point mass at $x^{(t-1)} = x^{(t)}$, and a weighted version of the proposal distribution $q(x^{(t)}|x^{(t-1)})$, which adjusts the acceptance probability [12]. Algorithm 1 shows the steps of the Metropolis-Hastings method.

The ratio in the acceptance probability α is called the acceptance ratio or acceptance rate:

$$r(x, x') = \frac{\pi(x')q(x|x')}{\pi(x)q(x'|x)}.$$

If the proposal distributions are equal, *i.e.* $q(x|x') = q(x|x')$, then the algorithm is called Metropolis method.

Algorithm 1 Metropolis-Hastings

1: Begin with some $x_0 \in \mathcal{X}$

2: **for** $t = 1, 2, ..., N$ **do**

3: Sample $x' \sim q(x'|x^{(t-1)})$

4: Set $x^{(t)} = x'$ with probability

$$\alpha(x^{(t-1)}, x') = \min\left\{1, \frac{\pi(x')q(x^{(t-1)}|x')}{\pi(x^{(t-1)})q(x'|x^{(t-1)})}\right\},$$

5: Else set $x^{(t)} = x^{(t-1)}$.

6: **end for**

2.3.2. Gibbs Sampling

The Gibbs framework is one of the most well known MCMC sampling methods, which can be used when the the random variable X is multi-dimensional. The idea behind the method is drawing samples from complete (or full) conditional distributions sequentially, thereby producing a Markov chain by updating one parameter at a time with the posterior density as its stationary distribution [20]. The foundations of Gibbs sampling idea were laid down by Stuart Geman and Donald Geman, and its name was dedicated to physicist J. W. Gibbs due to the similarities between sampling algorithm and the statistical physics [17].

Let $X = (x_1, x_2, ..., x_d)$ be a random variable (vector) with density $\pi(X) = \pi(x_1, ..., x_d)$. Assume further that one can sample for $\pi(x)$ from the conditional densities $\pi_i(x_i|x_1, ..., x_{i-1}, x_{i+1}, ..., x_d)$, $i = 1, 2, ..., d$, but not from π itself. A successively implemented Gibbs method will provide samples from the conditional densities $\pi_1, ..., \pi_n$ by conditioning on the latest samples.

The sampling procedure of the Gibbs sampling at iteration number t is given by the following:

$$
\begin{aligned}
x_1^{(t)} &\sim \pi_1(x_1^{(t-1)}|x_2^{(t-1)}, x_3^{(t-1)}, ..., x_d^{(t-1)}) \\
x_2^{(t)} &\sim \pi_2(x_2^{(t-1)}|x_1^{(t-1)}, x_3^{(t-1)}, \ldots, x_d^{(t-1)}) \\
&\vdots \\
x_d^{(t)} &\sim \pi_d(x_d^{(t-1)}|x_1^{(t-1)}, x_2^{(t-1)}, ..., x_{d-1}^{(t-1)}).
\end{aligned}
$$

It is guaranteed that the samples come from the exact distribution $P(x)$ as the number of iterations goes to infinity. Algorithm 2 shows the steps of the Gibbs sampling method.

Algorithm 2 Gibbs Sampling

1: Begin with some $X_0 \in \mathcal{X}$
2: **for** $t = 1, 2, ..., N$ **do**
3: **for** $i = 1, 2, ..., d$ **do**
4: Sample $x_i^{(t)} \sim \pi_i(\cdot|x_1^{(t-1)}, ..., x_{i-1}^{(t-1)}, x_{i+1}^{(t-1)}, ..., x_d^{(t-1)})$.
5: **end for**
6: **end for**

Gibbs sampling can be considered as a special type of MH method [12], where the acceptance probability is always one.

2.3.3. Metropolis-Hastings within Gibbs

As we can see from the previous sections, Gibbs and Metropolis-Hastings algorithms can be used in various combinations in order to draw samples from the distributions that we cannot directly calculate. The simplest method is the Gibbs method in which direct samples are drawn from the conditional posterior distributions. On the other hand, MH algorithm is mostly used for the cases where the full conditional distributions are not tractable.

If, in a model, some of the conditional posterior distributions can be calculated directly, whereas some of them cannot be calculated, the Metropolis-Hastings within Gibbs idea is used to perform the sampling. The components are updated one at a time with Gibbs sampling method if possible, and with Metropolis-Hastings moves otherwise [12]. The Algorithm 3 provides the steps for Metropolis-Hastings within Gibbs method.

Algorithm 3 Metropolis-Hastings within Gibbs

1: **for** $i = 1, 2, ...N$ **do**

2: **for** $i = 1, 2, ..., d$ **do**

3: Update x_i by a Metropolis-Hastings move that targets

$$\pi_i(\cdot|x_1, ..., x_{i-1}, x_{i+1}, ..., x_d)$$

4: **end for**

5: **end for**

2.3.4. Stochastic Gradient Langevin Dynamics

Stochastic Gradient Langevin Dynamics (SGLD) algorithm is an iterative subsampling based technique for Bayesian learning from large-scale datasets, first proposed by Welling and Teh (2011). The main idea of the SGLD algorithm is combining stochastic optimization algorithms, in which small mini-batches are used to approximate gradients, with Langevin Dynamics approach, in which the updates for the parameters are generated using the gradient information. Langevin dynamics introduces noise to the parameter updates that make the parameters converge to the samples from the full posterior distribution, while stochastic optimization provides an optimized likelihood and approximation to the Markov chain [21]. In general, the algorithm is a transition from stochastic optimization to a Bayesian method that samples from the posterior distribution.

Stochastic Optimization in SGLD. Let θ be the vector of parameters, and Y be the random varible whose dimensions $n \times d$. We can define the posterior distribution of n data items, $Y = (y_1, y_2, ..., y_n)$, as

$$p(\theta|Y) \approx p(\theta) \prod_{i=1}^{n} p(y_i|\theta).$$

A simple procedure of stochastic optimization method is given below [22]:

- A subset of m data items, $Y^{(t)} = \{y_1^{(t)}, y_2^{(t)}, ..., y_m^{(t)}\}$ is provided for each iteration t,
- The parameters in the randomly selected subset are updated according to the following equation:

$$\theta^{(t+1)} = \theta^{(t)} + \Delta\theta^{(t)}, \tag{2.7}$$

$$\Delta\theta^{(t)} = \frac{\epsilon^{(t)}}{2}\left(\nabla \log p(\theta^{(t)}) + \frac{n}{m}\sum_{i=1}^{m} \nabla \log p(y_i^{(t)}|\theta^{(t)})\right). \tag{2.8}$$

where ϵ_t is the step size and its choice has an important role to ensure convergence. The constraints that step sizes are required to meet are:

$$\sum_{t=1}^{\infty} \epsilon^{(t)} = \infty, \quad \sum_{t=1}^{\infty} (\epsilon^{(t)})^2 < \infty. \tag{2.9}$$

The parameters can attain the high probability regions without considering their initialization through the first constraint, and it is guaranteed that they will converge to the point through second constraint [21].

Langevin Dynamics in SGLD. Langevin dynamics idea introduces a normally distributed noise term to the stochastic gradient optimization method. Adding appropriate amount of noise term and choosing step-sizes that satisfy the conditions in (2.9) will ensure that the parameters will converge to the samples of the posterior distribution. In order to obtain samples that come from the posterior, the gradient step-sizes and the variance of the noise term are matched. Injecting Gaussian noise into the parameter update given in (2.8) yields a new update:

$$\Delta\theta^{(t)} = \frac{\epsilon^{(t)}}{2}\left(\nabla \log p(\theta^{(t)}) + \sum_{i=1}^{n} \nabla \log p(y_i^{(t)}|\theta^{(t)})\right) + \eta^{(t)}, \tag{2.10}$$

where $\eta^{(t)} \sim \mathcal{N}(0, \epsilon^{(t)})$.

Since Langevin dynamics simply aim to discretize a stochastic differential equation with an equilibrium distribution coming from the posterior, (2.10) can be considered as a proposal distribution. Then Metropolis-Hastings decides whether this proposal is accepted or rejected to correct the error arisen from the discretization [21]. Another method that corrects the discretization error is Hamiltonian Monte Carlo where Metropolis-Hastings framework is still applied, but instead of coming from a random walk, the proposals are chosen such that they enable more efficient transitions between the states via the momentum variables [23].

Combining SG with LD. The stochastic gradient optimization provides estimates for the gradient information using a subset of the dataset, without considering the uncertainty, while Langevin dynamics approach handles with the parameter uncertainty and data over-fitting problems processing the whole dataset. Combining the expedient properties of these two techniques helps improve the performance of the parameter estimation, especially when we have a large-scale dataset. SGLD framework provides Bayesian learning from huge datasets using small mini-batches iteratively. The parameter update of SGLD algorithm which is a combination of (2.8) and (2.10), is the following:

$$\Delta\theta^{(t)} = \frac{\epsilon^{(t)}}{2}\left(\nabla \log p(\theta^{(t)}) + \frac{n}{m}\sum_{i=1}^{m} \nabla \log p(y_i^{(t)}|\theta^{(t)})\right) + \eta^{(t)}, \tag{2.11}$$

where $\eta^{(t)} \sim \mathcal{N}(0, \epsilon^{(t)})$ and the step-sizes satisfy (2.9), i.e. converge to zero.

The decay of step-sizes makes the rejection ratio of MH close to zero. Thus we do not need to go over the whole dataset and calculate the probabilities to obtain acceptance ratios of Metropolis-Hastings algorithm.

Algorithm 4 SGLD

1: Input: a, b, γ

2: **for** $t = 1, 2, ..., T$ **do**

3: Choose a random subset of m data items $y^{(t)} = (y_1^{(t)}, ..., y_m^{(t)}) \in Y$

4: Calculate the step size

$$\epsilon^{(t)} = a(b+t)^{-\gamma} \quad \text{and} \quad \eta^{(t)} \sim \mathcal{N}(0, \epsilon^{(t)}),$$

5: Set

$$\theta^{(t+1)} = \theta^{(t)} + \frac{\epsilon^{(t)}}{2}\left(\nabla \log p(\theta^{(t)} + \frac{n}{m}\sum_{i=1}^{m} \nabla \log p(y_i^{(t)}|\theta^{(t)}))\right) + \eta^{(t)}$$

6: **end for**

3. MISSING DATA PROBLEMS IN REGRESSION MODELS

3.1. Regression Models

Regression is a method of modelling a target value based on independent predictor values, mostly used for explaining the causal relationship between the variables. The number of independent and dependent random variables, and the type of relationship between them are the two factors that determine the regression technique [12]. A simple structure of a regression model is depicted in Figure 3.1.

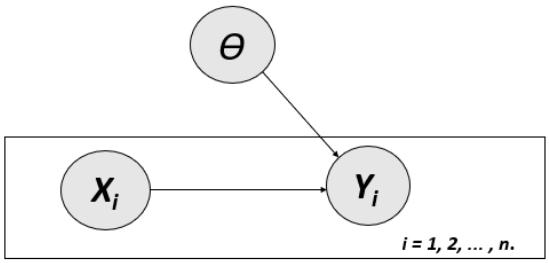

Figure 3.1: Schematic of a Regression Function.

3.1.1. Linear Regression

Linear regression is a simple regression analysis where there is one independent random variable which is linearly related to the dependent variable. In linear regression, input and output values are both numeric.

The linear equation assigns one scale factor to each input value (or column) called coefficient. An additional coefficient that gives one more degree of freedom is added. The added coefficient is often called the bias coefficient that helps move the model up

or down. A generalized linear regression model is defined as

$$y = X\theta + \epsilon,$$

where $y = (y_1, y_1, ..., y_n)^{\mathrm{T}}$ is an $n \times 1$ dependent variable (output) vector, X is the independent $n \times d$ random variable (input) matrix, $\theta = (\theta_1, \theta_2, ..., \theta_d)^{\mathrm{T}}$ are the corresponding coefficient values of X, and ϵ is an n-dimensional error term.

The model for predicting a single component, say i^{th} component, of X can be written as

$$y_i = x_i\theta + \epsilon_i, \tag{3.1}$$

where y_i is a scalar value, i^{th} element of y, $x_i = (x_{i,1}, x_{i,2}, ..., x_{i,d})$ is a $1 \times d$ vector (i^{th} row of X), and ϵ_i is the corresponding scalar value of error term [24].

The aim of learning a linear regression model is to estimate the values of coefficients used to represent the available data. There are various estimating methods based on the dimension of the variables, such as ordinary least squares and gradient descent.

3.1.2. Logistic Regression

Logistic regression is a binary classification method that measures the relationship between a binary dependent variable and one or more independent variables. The dependent variable is restricted to binary values, while independent variables can be continuous. Logistic regression model, which uses the sigmoid function to estimate the probability of occurrence of an event for a single variable, is defined as

$$p(y_i = 1|x_i, \theta) = \frac{1}{1 + e^{-x_i\theta}}, \tag{3.2}$$

where y_i is the binary response variable (or outcome of an event), and x_i is the i^{th} row of X given in Section 3.1.2, and $\theta = (\theta_1, \theta_2, ..., \theta_d)^{\mathrm{T}}$ is the parameter vector of the logistic regression model [25]. The graph of sigmoid function and visual representation of logistic regression model is depicted in Figure 3.2.

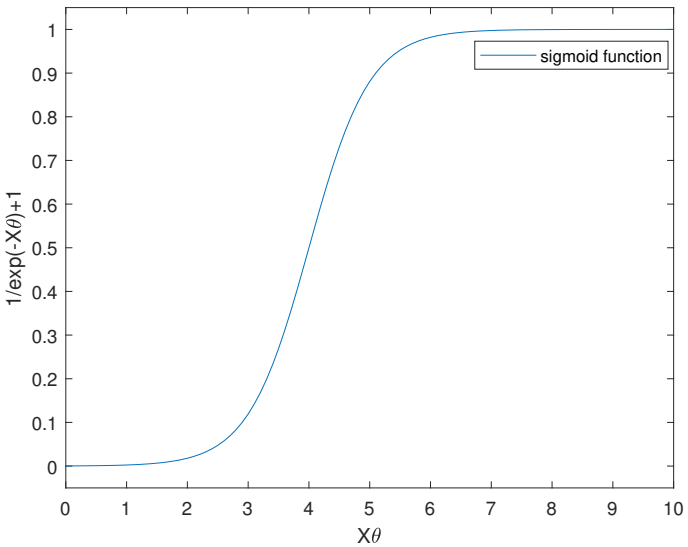

Figure 3.2: Graph of a Logistic Regression Function.

Logistic regression modelling is a widely-used and efficient technique since it does not require too many computations or input features to be scaled as well as it is easy to interpret. On the other hand, logistic regression can be applied only on the data which already has clearly identified independent variables. It is also vulnerable to over-fitting, which is the problem of failing to represent additional components of the data, due to the production of a rule that extremely corresponds to a particular set of data [26, 27].

3.2. Missing Data

Incomplete data problem may arise naturally or intentionally. The dataset can contain latent variables because of the design of the researcher or noncompliance of respondents [7]. As an example, portion of a census data, which aims to predict whether a person's income exceeds $50,000 per year or not, with some missing values is shown in Table 3.1. Since incompleteness increases the complexity of the data analysis, studies have been conducted to analyze the missing data patterns deeply, and understand

their effects on the results of regression models. Incompleteness patterns are basically categorized into two: missing at random and missing at nonrandom. The nonrandom pattern may cause the additional problem of bias, which is the failure of observed data to represent the unobserved parts of the data. Thus, it is more difficult to handle [2].

	workclass	education	marital-status	sex	native-country	income
1	State-gov	Bachelors	Never-married	Male	US	$\leq 50K$
2	Private	Doctorate	Married	Male	US	$\geq 50K$
3	Private	High school	Never-married	Female	?	$\geq 50K$
4	?	Some college	Divorced	Female	US	$\leq 50K$
5	Self-emp	Masters	Never-married	Male	China	$\leq 50K$
6	Private	Bachelors	Married-spouse-absent	Male	Cambodia	$\geq 50K$
7	?	High school	Divorced	Female	England	$\leq 50K$

Table 3.1: Adult data for income level, '?' represents the missing parts

3.3. Previous Methods in Literature

The first proposed procedure for handling missing data is excluding the missing components of the data and using only the observed ones. Although this procedure is easy to implement, it loses the information of incomplete cases. Since the approach ignores the possible differences between missing and observed parts of the data, the resulting inference may fail to reflect the complete dataset. Another approach proposes to substitute a predicted value for each missing component in dataset. For example, the mean of observed components can be used to replace the missing values. The algorithm is called single imputation that applies standard statistical procedure to newly predicted complete dataset. Since single imputation performs as if there is no missing value in dataset, it cannot reflect the uncertainty of predictions for missing values, and it can produce biased resulting variances of estimated variables which tend to converge to zero [5]. Rubin [8] has proposed a strategy called multiple imputation. Multiple imputation procedure imputes each missing component with a set of possible values. Through these multiple values offered, the uncertainty of missing variables can be represented. The algorithm performs the standard data analysis procedure to each

filled dataset. Then it combines and compares the results for the inference.

3.3.1. Non-MCMC Methods

The non-MCMC procedures for handling missing data generally can be assorted into two categories which are regression-based and neighbor-based approaches [28].

A regression-based approach uses linear or logistic regression model to obtain missing variables. The missing variables are assumed as response variables (dependent variables), whereas the observed ones are assumed as predictor variables (independent variables). The strategy produces imputations which are defined as samples from posterior predictive distributions specified by the regression model. The iterative imputing procedure continues by overwriting previously sampled values [29].

A neighbor-based approach uses a certain distance function to predict the missing values in the data. The distance function is supposed to determine the closest vector in order to impute the missing vector. The closest vector is the one whose characteristics are similar to the incomplete vector [28, 30].

3.3.2. MCMC Methods

The main idea of MCMC approach to handling missing data is that the values are generated via a statistical model that describes the distribution of the complete data. An improved version of this approach uses Bayesian networks. A Bayesian network provides a natural way to encode the relations between and within the variables. In order to comprehend the theory behind the MCMC methods with Bayesian inference for handling missing data, it will be helpful to clearly provide the definition of posterior distribution using the prior and joint densities.

Suppose that we have an $n \times d$ matrix X, that consists of observed and unobserved variables, such that $X = (X_{\text{miss}}, X_{\text{obs}})$, its corresponding d-dimensional parameter vector for the given model is $\theta = (\theta_1, \theta_2, ..., \theta_d)^{\text{T}}$, and the n-dimensional output vector

of the model is $Y = (y_1, y_2, ..., y_n)^T$. Using the Bayesian idea described in Section 2.1, we can write the conditional probability of of X_{miss}:

$$\begin{aligned}\pi(\theta, X_{\text{miss}}) = p(\theta, X_{\text{miss}}|X_{\text{obs}}, Y) &\propto p(\theta, X_{\text{miss}}, X_{\text{obs}}, Y) \\ &= p(\theta)\prod_{i=1}^{n} p(x_{\text{miss},i}, x_{\text{obs},i}|\theta)p(y_i|x_{\text{miss},i}, x_{\text{obs},i}, \theta),\end{aligned}$$

where we assume X is independent of θ, *i.e.*

$$p(x_{\text{miss},i}, x_{\text{obs},i}|\theta) = p(x_{\text{miss},i}, x_{\text{obs},i}) = p(x_i),$$

$$p(y_i|x_{\text{miss},i}, x_{\text{obs},i}, \theta) = p(y_i|x_i).$$

The procedure starts with Bayesian learning for variables, and then an MCMC technique is employed to draw samples from the probability distributions learned by the Bayesian network. The algorithm continues iteratively, until the convergence is reached. This method provides unbiased estimates, as well as the confidence levels of the imputation results [31].

Data Augmentation. Data augmentation is another traditional multiple imputation algorithm based on MCMC technique [32]. Parameter estimates are produced by repeating substitution conditionally on the prior value, constructing a stochastic process called Markov chain [33]. Suppose we have a dataset $X = (X_{\text{miss}}, X_{\text{obs}})$, its parameter vector $\theta = (\theta_1, \theta_2, ..., \theta_d)^{\text{T}}$ and $Y = (y_1, y_2, ..., y_n)^{\text{T}}$, like in the previous sections. Then, the procedure of data augmentation algorithm can be written as

$$X_{\text{miss}}^{(t+1)} \sim \pi(X_{\text{miss}}|\theta) = p(X_{\text{miss}}|X_{\text{obs}}, \theta^{(t)}, Y), \tag{3.3}$$

$$\theta^{(t+1)} \sim \pi(\theta|X_{\text{miss}}) = p(\theta|X_{\text{obs}}, X_{\text{miss}}^{(t+1)}, Y). \tag{3.4}$$

The imputation step is given in relation (3.3), where predicted values are generated from prior conditional distribution of missing values given the observed values and the parameter values at t^{th} iteration. Relation (3.4) is the parameter update part, where parameter values are generated from the posterior distribution, given the observed values and replaced values of X at the $(t+1)^{th}$ iteration. This procedure continues iteratively until its convergence is attained. The multiple imputation can be performed in two different ways. The first method runs a single chain for all iterations and takes every t^{th} prediction for X_{miss}, whereas the second method runs for parallel chains and takes the last values for replacing from these chains X_{miss} [32].

Metropolis-Hastings for Imputation and Parameter Estimation. Here, we note a contribution to the approach described in the previous section for the cases, where the full conditional distributions are not easy to sample from. We propose to use Metropolis-Hastings idea for imputation of missing variables. Especially for the cases where drawing samples is not possible or computationally efficient, making MH moves and updating X_{miss} will improve the performance of regression with missing data.

Suppose that X is the data matrix whose dimensions are $n \times d$, which contains missing variables in it, and its parameter and outcome vectors are the same with the ones defined in Section 3.2. The proposed algorithm for imputing missing data and estimating parameters is as follows: At every iteration t, an inner loop performs to impute the missing values in i^{th} row of X, where $i = 1, 2, ..., n$, by making MH moves for latent variables, while θ is supposed to be fixed. The proposal values for missing variables are updated according to the acceptance ratio of MH algorithm. In the experiments we have conducted, we choose the kernel proposal distribution q as symmetric random walk. So, the q ratio for X_{miss} is simply one. After the imputation period is completed in the inner loop, the provided X value is fixed and used to estimate parameters of the model. The algorithm estimates θ by MH moves again, using imputed version of $X^{(t)}$. The proposal kernel distribution for θ is also chosen as symmetric random walk giving the q ratio is one. This iterative process continues until convergence is obtained. Algorithm 5 shows the steps of the introduced method

for data augmentation and parameter estimation.

Algorithm 5 Metropolis-Hastings with Missing Data

Begin with some $X^{(0)} \in \mathcal{X}$

for $t = 0, 1, ..., T$ **do**

  for $i = 1, 2, ..., n$ **do**

    Sample $x'_{\text{miss},i} \sim q(x'_{\text{miss},i}|x^{(t)}_{\text{miss},i})$

    Set $x^{(t+1)}_{\text{miss},i} = x'_{\text{miss},i}$ with probability

$$\alpha(x^{(t)}_{\text{miss},i}, x'_{\text{miss},i}) = \min\left\{1, \frac{\pi(x_{\text{miss},i}|\theta)q(x^{(t)}_{\text{miss},i}|x'_{\text{miss},i})}{\pi(x^{(t)}_{\text{miss},i}|\theta)q(x'_{\text{miss},i}|x^{(t)}_{\text{miss},i})}\right\}.$$

    Else set $x^{(t+1)}_{\text{miss},i} = x^{(t)}_{\text{miss},i}$.

  end for

  Sample $\theta' \sim q(\theta'|\theta^{(t)})$

  Set $\theta^{(t+1)} = \theta'$ with probability

$$\alpha(\theta^{(t)}, \theta') = \min\left\{1, \frac{\pi(\theta'|X^{(t)}_{\text{miss}})q(\theta^{(t)}|\theta')}{\pi(\theta^{(t)}|X^{(t)}_{\text{miss}})q(\theta'|\theta^{(t)})}\right\}.$$

  Else set $\theta^{(t+1)} = \theta^{(t)}$.

end for

3.4. SGLD Method for Missing Data in Big Datasets

Unlike typical optimization-based methods where point-wise estimations for parameters are aimed to obtain, Bayesian methods attempt to provide the full posterior distribution of parameters based on the available data and the prior distribution. In this way, better characterizations are obtained, and the uncertainty is captured. Even though Bayesian methods provide these advantages, they are not popular in large-scale machine learning problems. The main reason for that is that a typical Markov chain Monte Carlo algorithm requires computation over the entire dataset at each iteration.

Previous studies conducted with the big datasets including missing components

such as internet traffic and network data or survey results have shown that Bayesian estimations is a feasible approach to handle the missing data problem. Ni and Leonard (2005) introduced the idea of using Markov chain Monte Carlo methods for sampling and imputing missing data [31], and then using the imputed data to estimate parameters. Their experiments have shown that MCMC methods are successful to estimate parameters when the data is not complete.

We aim to utilize the advantages of MCMC methods for handling the incomplete data problem by introducing the SGLD framework. As it is mentioned before, the SGLD method provides learning from large-scale datasets based on iterative learning from small mini-batches. We propose a hybrid algorithm that combines the Metropolis-Hastings idea with SGLD. The hybrid algorithm makes an SGLD move for θ using an estimate of gradient log-likelihood of a randomly selected subsample, while taking MH steps for unobserved elements of X. The imputation of X is performed by an inner loop, where the latent variables in the subsampled matrix are imputed line by line. After the imputation period, the parameter estimation is performed via averaging the gradients calculated with completed X whose missing components are sampled with MH moves. The procedure of the MH-SGLD algorithm is outlined in the Algorithm 6.

Here we have a small notation change. In this context, let u denote the missing variables in i^{th} row of X, *i.e.* $u = x_{\text{miss},i}$, and z denote the combination of the observed variables in i^{th} row of X with the corresponding output, *i.e.* $z = (x_{\text{obs},i}, y_i)$. Then the derivations for the gradient of log-likelihood for incomplete data is calculated by the following:

$$\nabla \log p_\theta(z) = \int \nabla \log p_\theta(u, z) p(u|z) du,$$

where the gradient of $p_\theta(u, z)$ is calculated as

$$\begin{aligned}\nabla \log p_\theta(u, z) &= \nabla \log(p(u)p_\theta(z|u)) \\ &= \nabla \log p(u) + \nabla \log p_\theta(z|u) \\ &= \nabla \log p_\theta(z|u).\end{aligned}$$

Since prior probability distribution of u is independent of θ, its derivative with respect to θ is zero. Thus, the second line is followed by the third line.

When exact sampling is not possible, MH can be employed to draw samples

$$u^{(1)}, u^{(2)}, ..., u^{(N)}$$

from $p_\theta(u|z)$ so that the gradient of log-likelihood is approximated as

$$\nabla \log p_\theta(z) \approx \frac{1}{N} \sum_{i=1}^{N} \nabla \log p_\theta(u, z).$$

We have conducted some experiments in order to observe and compare the performances of two approaches that we propose. The imputation part given in (3.3) is the same for both MH and SGLD algorithms, whereas the parameter updates are different. The main question is: Can we obtain similar results with MH sampling, if we use gradient approximations of a small subset of the data for parameter estimations? In this way, we aim to enhance the performance of regression with missing data according to time and efficiency measures.

Algorithm 6 SGLD with Missing Data

Input: a, b, γ

Start with an initial value $X^{(0)} \in \mathcal{X}$

for $t = 1, 2, ..., T$ **do**

  Choose a random subsample $U \subset \{1, 2, ..., n\}$ such that the size of U is m.

  for $u = 1, 2, ..., m$ **do**

    $x_u^{(0)} = x_u^{(t-1)}$

    for $j = 1, 2, ..., J$ **do**

      Update $x_{u,\text{miss}}^{(t,j)} \to x_{u,\text{miss}}^{(t,j-1)}$ given $(y_u, x_{u,\text{obs}}, \theta)$

      Set $x_u^{(t,j)} = (x_{u,\text{miss}}^{(t,j)}, x_{u,\text{obs}}^{(t)})$

    end for

    Calculate

$$\nabla \log p(y_u | x_{u,\text{miss}}^{(t)}, \theta) = \frac{1}{J} \sum_{j=1}^{J} \nabla \log p(y_u | x_u^{(t)}, \theta)$$

  end for

  Calculate

$$\epsilon^{(t)} = a(b+t)^{-\gamma} \quad \text{and} \quad \eta^{(t)} \sim \mathcal{N}(0, \epsilon^{(t)}),$$

  Set

$$\theta^{(t+1)} = \theta^{(t)} + \frac{\epsilon^{(t)}}{2} \left(\nabla \log p(\theta^{(t)}) + \frac{n}{m} \sum_{i=1}^{m} \nabla \log p(y_u | x_u^{(t)}, \theta^{(t)}) \right) + \eta^{(t)}$$

end for

4. EXPERIMENTS WITH SYNTHETIC DATA

In this chapter, the applications of two proposed methods on an artifical dataset are presented. In addition to this, another missing data algorithm mentioned in Section 3.3 is implemented in order to compare the performances of proposed algorithms with an existing algorithm. Methodology and the derivations are also provided.

4.1. Data Description

We generate a data matrix X whose dimensions are $n \times d$. The variables in X has Gaussian distribution with mean and covariance parameters denoted by μ_x and Σ_x, respectively. Since the aim is to measure the performances of algorithms for large-scale datasets, we choose $n = 500,000$ and $d = 5$. In order to obtain missing variables in X, we produce a response indicator matrix A with the same dimensions of X. We mask some random variables by pointwise multiplication of X by A, whenever X is observed, A is equal to one; otherwise A is equal to zero. The sparsity of A can be adjusted so that one can obtain different ratio of unobserved values to observed ones and observe the effect of missingness on parameter estimation. The n-dimensional output vector Y is obtained by applying the logistic regression function to X and its given initial parameter vector. The parameter vector for logistic regression function is denoted by $\theta = (\theta_1, \theta_2, ..., \theta_d)^{\mathrm{T}}$. It is supposed that the initial parameter vector is distributed from normal distribution with mean and covariance denoted by μ_θ and Σ_θ accordingly.

4.2. Methodology

As it is stated in the previous chapter, the proposed algorithm completes unobserved parts of the data by making MH moves. Since we aim to introduce a framework that utilizes the Metropolis-Hastings and SGLD ideas, instead of the whole dataset, we work with the randomly selected subset of X. The algorithm chooses a random subset at each iteration and completes the unobserved parts of only this subset. The algorithm assumes that the variables are independent and identically distributed. Therefore, we

can work with each element in a row separately, and we can suppose that replacing one element with a proposed value does not affect the probability of others.

4.2.1. Imputation of Missing Components

The imputation is necessary when the subsampled predictor matrix has missing values. The location of the missing variables in X is indicated by zeroes in a binary matrix, say A, whose variables are 0 to represent missing components in X. We provide the prior parameters for X, and the algorithm performs Metropolis-Hastings move for the missing variables in the predictors of a logistic regression model.

We use the logarithmic values of prior conditional probabilities and proposal distributions in order to make calculations conveniently. For our case, ratio $r(x_{\text{miss},i}, x'_{\text{miss},i})$ in the acceptance probability $\alpha(x_{\text{miss},i}, x'_{\text{miss},i})$ is defined as

$$r(x_{\text{miss},i}, x'_{\text{miss},i}) = \frac{\pi(x'_{\text{miss},i}|\theta)q(x_{\text{miss},i}|x'_{\text{miss},i})}{\pi(x_{\text{miss},i}|\theta)q(x'_{\text{miss},i}|x_{\text{miss},i})}.$$

The q ratio is one since proposal kernel is chosen as a symmetric random walk and the conditional probabilities are calculated by the Bayesian approximation. That is

$$\pi(x'_{\text{miss},i}|\theta) = p(x'_{\text{miss},i}|\theta, y_i, x_{\text{obs},i}) \propto p(\theta, x'_{\text{miss},i}, y_i, x_{\text{obs},i}) = p(x'_i)p(y_i|x'_i, \theta),$$
$$\pi(x_{\text{miss},i}|\theta) = p(x_{\text{miss},i}|\theta, y_i, x_{\text{obs},i}) \propto p(\theta, x_{\text{miss},i}, y_i, x_{\text{obs},i}) = p(x'_i)p(y_i|x'_i, \theta),$$

where $x'_i = (x'_{\text{miss},i}, x_{\text{obs},i})$. So we obtain the acceptance ratio as

$$r(x_{\text{miss},i}, x'_{\text{miss},i}) = \frac{p(x'_i)p(y_i|x'_i, \theta)}{p(x_i)p(y_i|x_i, \theta)}.$$

We consider the ratio of priors and the ratio of likelihood terms separately.

Calculation of Prior Probability Ratio. The logarithmic ratio of prior probabilities of $x_{\text{miss},i}$ and $x'_{\text{miss},i}$ can be defined as

$$\begin{aligned}\log \frac{p(x'_{\text{miss},i})}{p(x_{\text{miss},i})} &= \log p(x'_{\text{miss},i}) - \log p(x_{\text{miss},i}) \\ &= -\frac{1}{2}\left((x'_{\text{miss},i} - \mu_x)\Sigma^{-1}(x'_{\text{miss},i} - \mu_x)^{\mathrm{T}} - (x_{\text{miss},i} - \mu_x)\Sigma^{-1}(x_{\text{miss},i} - \mu_x)^{\mathrm{T}}\right).\end{aligned} \tag{4.1}$$

The evaluation of the expression in (4.1) requires calculations for each row of X (x_i) separately, which means that we need to perform n operations in one iteration. In order to avoid deceleration in the algorithm caused by these row operations, we apply the Cholesky decomposition to the covariance matrix Σ_x. We can write $\Sigma_x = C^{\mathrm{T}}C$ where C is the Cholesky factor of Σ_x. If we represent $(x_{\text{miss},i} - \mu_x)$ by Z, then the logarithm of the prior probability ratio can be expressed as

$$\log \frac{p(x'_{\text{miss},i})}{p(x_{\text{miss},i})} \stackrel{+}{=} -\frac{1}{2}(ZC^{-1})(ZC^{-1})^{\mathrm{T}}, \tag{4.2}$$

which can be evaluated by taking the square of product of two matrices, instead of evaluating each row of X_{miss} individually.

Calculation of Likelihood Ratio. The second part of the acceptance rate is the ratio of logistic regression functions evaluated at the current and the proposed values for X. Again we take the logarithm, and define the likelihood ratio vector as

$$\log \frac{p(y_i|x'_i,\theta)}{p(y_i|x'_i,\theta)} = \log p(y_i|x'_i,\theta) - \log(p(y_i|x_i,\theta). \tag{4.3}$$

We plug the logistic regression function, and after simple calculations, obtain the likelihood vector as

$$\log \frac{p(y_i|x'_i,\theta)}{p(y_i|x_i,\theta)} = -(x'_i - x_i)\theta(1-y_i) - \log(1+e^{-x'_i\theta}) + \log(1+e^{-x_i\theta}). \tag{4.4}$$

Obtaining the ratio vectors defined in (4.1) and (4.4) and taking the summation of them yield the logarithm of the acceptance rate. The algorithm updates missing variables of X according to this acceptance rate in an inner iteration that we call burn-in. Thus the first imputations for X_{miss} are discarded and more reliable replacements are presented. The sampling of missing variables in X_{miss} is completed at this point.

4.2.2. Parameter Update Using MH

After sampling and replacing missing X components for each iteration, we estimate the parameter θ when we suppose that X and Y are given. We propose a value for next value of θ that also comes from random walk and calculate the ratio of conditional densities as

$$\frac{\pi(\theta'|X)}{\pi(\theta|X)} = \frac{p(\theta')\prod_{i=1}^{n} p(y_i|x_i,\theta')}{p(\theta)\prod_{i=1}^{n} p(y_i|x_i,\theta)},$$

where $\theta' = \theta + z$ is the proposed value of θ, and $z \sim N(0, \Sigma_q)$ is the random walk step with zero mean and variance Σ_θ. Since proposal kernel is symmetric random walk, q ratio is one again. We apply the same procedure with X, which is taking the logarithms and calculating prior probability ratio vector and likelihood ratio vector separately. Then, the combination of these vectors provide acceptance probability vector of MH algorithm for parameter update.

Since θ is supposed to be normally distributed with mean μ_θ and covariance Σ_θ, we can calculate the logarithm of its prior probabilities as

$$\begin{aligned}
\log\frac{p(\theta')}{p(\theta')} &= \log p(\theta') - \log p(\theta) \\
&= \log\left(e^{-0.5(\theta'-\mu_\theta)\Sigma_\theta^{-1}(\theta'-\mu_\theta)^{\mathrm{T}}}\right) - \log\left(e^{-0.5(\theta-\mu_\theta)\Sigma_\theta^{-1}(\theta-\mu_\theta)^{\mathrm{T}}}\right) \\
&= -0.5(\theta'-\mu_\theta)\Sigma_\theta^{-1}(\theta'-\mu_\theta)^{\mathrm{T}} + 0.5(\theta-\mu_\theta)\Sigma_\theta^{-1}(\theta-\mu_\theta)^{\mathrm{T}}. \qquad (4.5)
\end{aligned}$$

The log-likelihood ratios for parameters of the logistic regression function is

$$\begin{aligned}\log \frac{p(y_i|x_i,\theta')}{p(y_i|x_i,\theta)} &= \log p(y_i|x_i,\theta') - \log p(y_i|x_i,\theta) \\ &= -(\theta'-\theta)x_i(1-y_i) - \log(1+e^{-x_i\theta'}) + \log(1+e^{-x_i\theta}), \end{aligned} \tag{4.6}$$

4.2.3. Parameter Update Using SGLD

The first part of the algorithm provides a complete subset of the data that we use to estimate the parameters for the logistic regression model. In this second part of the method, incompleteness is not considered and the algorithm makes an SGLD move for θ using an approximate value of gradient log-likelihood of a randomly selected subsample. The gradient estimation is performed via averaging the gradients calculated with complete X, whose missing components are sampled with MH moves.

The algorithm first calculates the estimates for gradient of prior distribution, which is the first part of (2.11). In order to make calculations more conveniently, we prefer to use the logarithmic values of priors. Since we have normally distributed parameter vector, we define its gradient of log-prior as

$$\begin{aligned}\nabla_\theta \log p(\theta) &= \nabla_\theta \log\left(e^{-\frac{1}{2}(\theta-\mu_\theta)^{\mathrm{T}}\Sigma^{-1}(\theta-\mu_\theta)}\right) \\ &= \nabla_\theta\left(-\frac{1}{2}(\theta-\mu_\theta)^{\mathrm{T}}\Sigma^{-1}(\theta-\mu_\theta)\right) \\ &= -\Sigma_\theta^{-1}(\theta-\mu_\theta). \end{aligned} \tag{4.7}$$

The second part of SGLD update for theta in (2.11) is the summation of posterior

probabilities. The sum-log-posterior is derived by the following:

$$\begin{aligned}\sum_{i=1}^{m} \nabla_\theta \log p(y_i|x_i,\theta) &= \sum_{i=1}^{m} \nabla_\theta \log \frac{e^{-x_i\theta(1-y_i)}}{1+e^{-x_i\theta}} \\ &= \sum_{i=1}^{m} \nabla_\theta \left(-x_i\theta(1-y_i) - \log(1+e^{-x_i\theta})\right) \\ &= \sum_{i=1}^{m} \left(-x_i(1-y_i) + \frac{e^{-x_i\theta}}{1+e^{-x_i\theta}}\right), \end{aligned} \tag{4.8}$$

where m denotes the sub-sample size.

Now, we need to introduce an appropriate amount of noise term to guarantee that the obtained θ values will converge to samples from true posterior distribution. The stepsize parameters are adjusted so that the noise term satisfies the required condition. The stepsize also needs to obey the convergence itself, and it is restrained by the constraints given in (2.9). The definitions of stepsize and noise term for each iteration t are

$$\epsilon_t = a(b+t)^{-\gamma} \quad \text{and} \quad \eta_t \sim \mathcal{N}(0,\epsilon_t),$$

where a, b and γ are the stepsize parameters. Combining the gradient of log-priors (4.5), and the summation of log-posteriors (4.8), and adding the Gaussian noise term we obtain θ update within (3.4) as

$$\Delta\theta_t = \frac{\epsilon_t}{2}\left(-\Sigma_\theta^{-1}(\theta-\mu_\theta) + \frac{n}{m}\left(\sum_{i=1}^{m} -x_i(1-y_i) + \frac{e^{-x_i\theta}}{1+e^{-x_i\theta}}\right)\right) + \eta_t. \tag{4.9}$$

4.3. Experiments and Results

The Gaussian data matrix is synthetically generated in the MATLAB environment to test the algorithm on a normally distributed dataset with latent variables. The prior parameters, mean and covariance of the data are also provided. Then an-

other binary matrix whose dimensions are the same with the data matrix is generated in order to mask some variables in the dataset randomly. The zero values represent the latent variables in X, while one represents the observed variables. The sparsity of mask matrix can be adjusted so that the portion of incompleteness is determined. We also suppose that, for starting values, the parameter θ of the logistic regression model comes from normal distribution with specified mean and covariance values. The subsample size is denoted by m. The specified dimensions and the parameters are as follows:

- Data matrix $X_{n\times d}$, with $n = 500,000$ and $d = 5$,
- Mask matrix $A_{n\times d}$, with $n = 500,000$ and $d = 5$,
- 20% of A is 0, meaning that 20% of the data is missing,

As we mentioned before, the data is fitted to the logistic regression function which is provided in (3.1), and the obtained results are compared by the prediction analysis. The steps of our prediction analysis are as follows:

- Separate the data into two sets: training and test,
- Use the training set to impute missing values and estimate parameters and observed components of test set,
- Determine the observed variables in the test set,
- Calculate the expected value of the likelihood function applying the expression given in (2.3) to the observed variables in the test set:

$$\mathbb{E}_p(y_i) \approx \frac{1}{N}\sum_{i=1}^{N}\frac{1}{1+e^{-x_i\theta^{(i)}}},$$

  where N is the number of iterations after burn-in period, and $\mathbb{E}_p(y_i)$ is the prediction probability that determines the predicted value for y_i.
- If the prediction probability is greater than threshold, which is chosen as 0.5 in our experiments, then assign the predicted value of the component to one, otherwise assign it to zero.

- Obtain the number of components where predicted value and actual value match, and calculate the total prediction accuracy by taking the ratio of the number of matching components to total number of components.

We have simulated MH and SGLD algorithms with different iteration numbers and subsample sizes. In order to reduce the computational costs in the MH algorithm, we use the same subsample m when imputing latent variables in X, instead of using the whole dataset. On the other hand, when estimating parameters we sampled from the entire but partially-imputed dataset X. The results are compared according to their prediction successes and run times of the algorithms, when the iteration numbers are the same. However, we should point out that SGLD algorithm performs an additional inner loop to take the average of the estimated gradients. We choose the iteration number of inner loop as five and the burn-in period as three, in order to discard the first imputed missing values.

Table 4.1 shows the prediction accuracy for MH and SGLD results. It can be clearly seen that the iteration number does not have a great impact on the results when the subset size is 100, whereas the runtimes for these iteration numbers differ significantly for the algorithms. Approximately 92% of the obtained results for both MH and SGLD algorithms are consistent with the output which comes from the logistic regression model and its initial parameters. As the iteration number is increased, the prediction accuracy presented by the SGLD method starts to be greater than the prediction accuracy presented by the MH method.

Since the generated data has $500,000$ samples, a subset of size 100 might not be enough to represent the entire data. So we have also run the algorithms with random subset of size 500 (0.1% of the data). It can be observed from Table 4.3 that the prediction accuracy has increased to 94% for the SGLD algorithm. The difference between the execution times of the algorithms is also increased, so one can prefer to apply SGLD algorithm instead of MH, especially when the data size becomes larger.

Table 4.1: Comparison of MH and SGLD algorithms when subsample size is 100 (synthetic data).

Number of Iterations	Inference Method	Prediction Accuracy	Elapsed Time
1×10^4	MH sampling	0.930	223.40 sec
	MH-SGLD sampling	0.918	15.05 sec
5×10^4	MH sampling	0.931	1121.88 sec
	MH-SGLD sampling	0.929	76.93 sec
1×10^5	MH sampling	0.899	3408.13 sec
	SGLD sampling	0.933	149.94 sec

Table 4.2: Comparison of MH and SGLD algorithms when subsample size is 500 (synthetic data).

Number of Iterations	Inference Method	Prediction Accuracy	Elapsed Time
1×10^4	MH sampling	0.930	227.30 sec
	SGLD sampling	0.929	22.86 sec
5×10^4	MH sampling	0.929	1103.32 sec
	SGLD sampling	0.929	114.61 sec
1×10^5	MH sampling	0.870	3420.21 sec
	SGLD sampling	0.940	243.77 sec

Table 4.3: Comparison of MH and SGLD algorithms when subsample size is $10,000$ (synthetic data).

Number of Iterations	Inference Method	Prediction Accuracy	Elapsed Time
1×10^4	MH sampling	0.954	396.35 sec
	SGLD sampling	0.954	75.88 sec
5×10^4	MH sampling	0.955	1948.72 sec
	SGLD sampling	0.955	355.60 sec
1×10^5	MH sampling	0.961	6003.84 sec
	SGLD sampling	0.955	712.61 sec

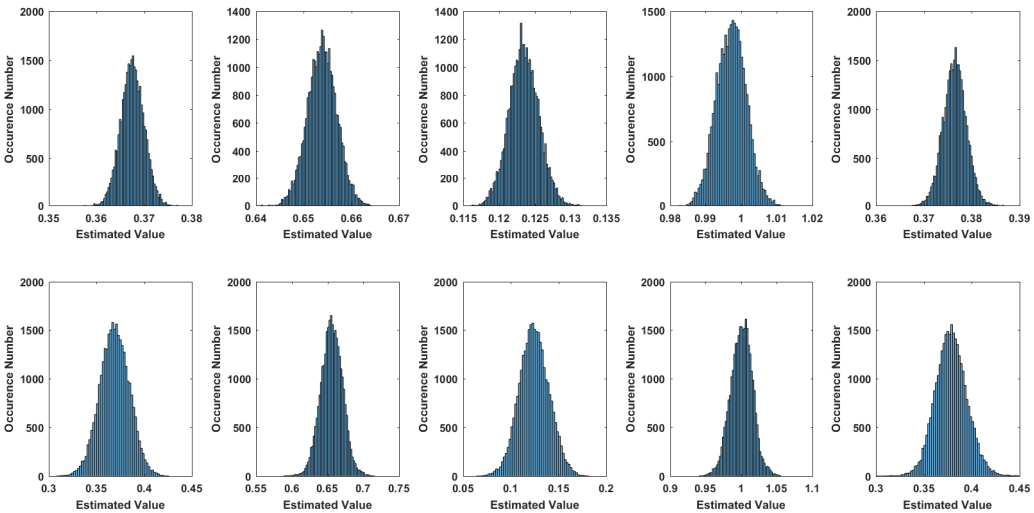

Figure 4.1: Histograms of parameter components of θ estimated by MH (above) and SGLD (bottom) with $m = 500$ and number of iterations 5×10^4.

Figure 4.1 shows the histograms of five components of θ, when the subsample size is 500 and the number of iterations is 5×10^4. The histograms lying in the above row are obtained by the MH algorithm, and the histograms lying in the bottom row are obtained by the SGLD algorithm. It can be observed from the figure that, the histograms obtained by the SGLD algorithm are narrower than the histograms obtained by the MH algorithm. Considering the variances of components provided in Table 4.6, where the components obtained by SGLD have higher variances, the difference between the histograms is as expected. Increasing the number of iterations might decrease the variances yielded by SGLD, and broader histograms similar to MH can be obtained.

In order to evaluate the performances of algorithms not only compared to each other, but also compared to the existing algorithms in the literature, we conducted experiments with the complete case analysis method in which the latent variables are excluded (mentioned in Section 3.3). We have observed the change in the prediction successes with the percentage of missing variables in the dataset. The simulations are run for 5×10^4 iterations, and the subsample size for SGLD algorithm is chosen as 500. In this third method, the MH algorithm is simulated using only with the observed components, and the resulting prediction successes are shown in Table 4.4. Clearly,

Table 4.4: Prediction Value for algorithms, with $m = 500$, number of iterations 5×10^4.

Missingness ratio	PA for MH	PA for SGLD	PA for CCA
10%	0.946	0.941	0.895
20%	0.930	0.929	0.845
30%	0.930	0.874	0.814
40%	0.923	0.870	0.777
50%	0.892	0.868	0.731

Table 4.5: Prediction Value for algorithms, with $m = 500$, number of iterations 10^5.

Missingness ratio	PA for MH	PA for SGLD	PA for CCA
10%	0.946	0.943	0.926
20%	0.932	0.940	0.900
30%	0.930	0.905	0.871
40%	0.922	0.872	0.820
50%	0.892	0.868	0.787

one can interpret that the three approaches have similar prediction values for small amount of latent variables. However, the success rate of the complete case analysis algorithm has significantly decreased when the ratio of missing variables gets higher. This might be caused by the lack of information of incomplete cases, as mentioned in Section 3.3. It can be stated that MH and SGLD algorithms have more tolerance to change in the ratio of incompleteness.

Table 4.5 provides the experiment results yielded by the three algorithms, with 10^5 iterations and a subsample size of 500. The aim of the tests with higher number of iterations is to see the effect of iteration number on the success of complete case analysis approach. Even though the iteration number has raised the prediction accuracy, the algorithm is still outperformed by MH and SGLD algorithms, especially for higher rates of latent variables.

Table 4.6: Posterior mean and variance values of components obtained by the three algorithms, when subsample size is 500, and number of iterations is 5×10^4 (synthetic data).

Parameter component	Posterior Values	MH	SGLD	CCA
1	mean	-0.610	-0.610	-0.503
	variance ($\times 10^{-5}$)	1.2	4.9	0.66
2	mean	0.241	0.243	0.116
	variance ($\times 10^{-5}$)	0.5	4.4	0.67
3	mean	-1.199	-1.196	-1.091
	variance ($\times 10^{-5}$)	3	5.0	1.77
4	mean	-0.051	-0.050	0.063
	variance ($\times 10^{-5}$)	0.2	5.1	0.35
5	mean	0.100	0.100	0.276
	variance ($\times 10^{-5}$)	0.3	4.2	0.97

5. EXPERIMENTS WITH REAL DATA

In this chapter, the proposed methods are applied to a real dataset, and the results are presented. In addition to this, another missing data approach is implemented in order to compare the performances of proposed algorithms with an existing algorithm. Methodology and the derivations are also provided.

5.1. Data Description

We analyze a real adult dataset that predicts whether the annual income of a person exceeds $\$50K$ or not based on the census data. The census data has various categorical variables that might affect the income level of adults such as education level, sex, marital status, current occupation and native country. Each row of the data matrix contains a person's attributes for these categories. Corresponding binary labels represent that the income of an adult exceeds $\$50K$ per year if it is one, and zero otherwise. Some of the attributes are missing in the rows and we indicate the unobserved variables with the response indicator matrix A. The categorical data is fit with the logistic regression model so that we make our derivations using logistic regression function. Like in the synthetic data, $X = (X_{\text{miss}}, X_{\text{obs}})$ denotes the $n \times d$ data matrix, Y denotes the corresponding binary outcome, and $\theta = (\theta_1, \theta_2, ..., \theta_d)^{\text{T}}$ is the parameter vector for the logistic regression model.

5.2. Methodology

5.2.1. Imputation of Missing Components

Since MH algorithm requires simulating samples based on the present data for missing components of X, we need to determine the prior distribution from observed data. This prior distribution is used to calculate the probability of X that we need to determine the acceptance ratio of the MH algorithm. We also need to determine the proposal distribution $q(x)$, which cannot be chosen as a symmetric random walk for

the real dataset. Recall that the general form of the acceptance probability $\alpha(x, x')$ of MH move for $x, x' \in X$ is defined as

$$\alpha(x, x') = \min\left\{1, \frac{\pi(x')q(x|x')}{\pi(x)q(x'|x)}\right\}. \tag{5.1}$$

where $\pi(x)$ is the conditional probability distribution of x and $q(x|x^{'})$ is the proposal kernels.

There are two important and different points that we need to consider for categorical dataset. The first issue is the calculation of the prior probability distribution of X and the second issue is determining the proposal kernel.

We can approximately calculate the probability distribution of X using the prior availability of each element. In other words, we calculate the probability of each element of X by counting the present number of the specific values for each category and dividing this to total number of observed variables.

Prior Probability Distribution of X_{miss}. Unlike the synthetic dataset in which we already have normally distributed random variables and parameter values of the distribution, we do not have a specific probability distribution to calculate $\pi(X_{\text{miss}}|\theta)$ for the categorical dataset. What we have is the number of observed elements belonging to each category. We can use this information to approximately calculate the conditional probability of X_{miss} when we are given X_{obs}. Counting the occurrence number of every category and dividing this number to the total number of occurrences provide an estimation for the next (missing) variable. Thus, having the frequencies of each category, we can extract the conditional probability distribution $\pi(X_{\text{miss}}|\theta)$.

Proposal Kernel Distribution. Since $q(x|x^{'})$ is the probability of moving from present point to another point in the neighborhood of the current location, choosing the proposal values depending on the current ones is a reasonable approach. So, the

proposal distribution is determined such that the algorithm takes an MH step directly proportional to the frequencies of the elements in each category. In other words, for the missing variables, the probability distribution of kernel proposal is equal to the probability distribution of X:

$$q(x_{\text{miss},i}|x'_{\text{miss},i}) = p(x_{\text{miss},i}) \quad \text{and} \quad q(x'_{\text{miss},i}|x_{\text{miss},i}) = p(x'_{\text{miss},i}). \tag{5.2}$$

Now we plug the proposal kernels which are given in (5.2) into the acceptance ratio given in (5.1). This results in the acceptance ratio

$$\alpha(x_{\text{miss},i}, x'_{\text{miss},i}) = \min\left\{1, \frac{\pi(x'_{\text{miss},i}|\theta)p(x_{\text{miss},i})}{\pi(x_{\text{miss},i}|\theta)p(x'_{\text{miss},i})}\right\}.$$

where we can calculate $\pi(X'_{\text{miss},i}|\theta)$ and $\pi(x_{\text{miss},i}|\theta)$ by

$$\begin{aligned}
\pi(x'_{\text{miss},i}|\theta) &= p(x'_{\text{miss},i}|\theta, y_i, x_{\text{obs},i}) \propto p(\theta, x'_{\text{miss},i}, y_i, x_{\text{obs},i}) = p(x'_i)p(y_i|x'_i, \theta), \\
\pi(x_{\text{miss},i}|\theta) &= p(x_{\text{miss},i}|\theta, y_i, x_{\text{obs},i}) \propto p(\theta, x_{\text{miss},i}, y_i, x_{\text{obs},i}) = p(x_i)p(y_i|x_i, \theta)
\end{aligned}$$

where $x'_i = (x'_{\text{miss},i}, x_{\text{obs},i})$. This yields the final version of acceptance ratio of MH algorithm for the adult dataset:

$$\begin{aligned}
\alpha(x_{\text{miss},i}, x'_{\text{miss},i}) &= \min\left\{1, \frac{p(x'_i)p(y_i|x'_i, \theta)p(x_i)}{p(x_i)p_\theta(y_i|x_i, \theta)p(x_i)}\right\} \\
&= \min\left\{1, \frac{p(y_i|x'_i, \theta)}{p(y_i|x_i, \theta)}\right\} \\
&= \min\left\{1, \frac{e^{-x_i\theta(1-y_i)}(1+e^{-x'_i\theta})}{e^{-x'_i\theta(1-y_i)}(1+e^{-x_i\theta})}\right\}.
\end{aligned}$$

5.2.2. Parameter Update Using MH

For a categorical dataset, we cannot easily recognize the relationship between the variable types belonging in specific categories and their corresponding parameters. In order to consider the effect of each variable type on the response separately, we need to expand the parameters as well as the data X. The procedure that we apply to expand the parameters is as follows:

- If a category, say k^{th} category, has more than two different types, define a new parameter vector for θ_k, such that $\theta_k = (\theta_{k,1}, \theta_{k,2}, ..., \theta_{k,l})$, where l is the number of types in the category,
- Determine the active parameter according to the available element type in X,
- Extend the row vector of X as much as the length of the parameter vector,
- Fill the extended vector of X with zeroes for the inactive types and one for the active (current) type.

After obtaining the expanded versions of X and θ, the algorithm performs the sampling procedure with these new matrix and vector. The remaining calculations are similar to those for the synthetic dataset. In each iteration, the first part of the algorithm provides a completed random subsample to make estimations for θ. The extended and imputed version of X, say X_{ext} and the expanded parameter vector, say θ_{ext} coming from the last iteration are used to make MH moves, and calculate its acceptance rate. Replacing X and θ with X_{ext} and θ_{ext} in equations (4.5) and (4.6) provides the acceptance rate vector and the parameter update is performed according to this vector.

5.2.3. Parameter Update Using SGLD

The data and parameter vectors we use in the calculations of SGLD move are also the expanded vectors of X and θ. Algorithm performs subsampling, data imputation and parameter estimation via X_{ext} and θ_{ext}. Completed random subsample of the data is provided by the first part of the algorithm, and this imputed subset is used to calculate the gradient log-likelihood. The average value of the gradient log-likelihood

serves the SGLD move. An appropriate amount of Gaussian noise term is injected into the gradient steps in order to avoid the collapse of the parameters. The appropriate amount is determined by considering the balance of the gradient step-sizes and the variance of samples [21].

The gradient estimates for prior probability distribution $p(\theta)$ can be calculated by assuming that the probability distribution of expanded parameter is Gaussian. Again we use the logarithm of the likelihoods. Prior mean and covariance values are provided according to the expanded parameter, and the rest of the calculations are the same as in (4.7):

$$\begin{aligned}\nabla_\theta \log p(\theta) &= \nabla_\theta \log \left(e^{-\frac{1}{2}(\theta-\mu_\theta)^{\mathrm{T}}\Sigma^{-1}(\theta-\mu_\theta)} \right) \\ &= \nabla_\theta \left(-\frac{1}{2}(\theta-\mu_\theta)^{\mathrm{T}}\Sigma^{-1}(\theta-\mu_\theta) \right) \\ &= -\Sigma_\theta^{-1}(\theta-\mu_\theta). \end{aligned} \tag{5.3}$$

The derivation of the gradient estimates for posterior distribution is also same as in (4.8). Using the extended parameter vector instead of the original one, we can calculate the sum-log-posterior by

$$\begin{aligned}\sum_{i=1}^{m} \nabla_\theta \log p(y_i|x_i,\theta) &= \sum_{i=1}^{m} \nabla_\theta \log \frac{e^{-x_i\theta(1-y_i)}}{1+e^{-x_i\theta}} \\ &= \sum_{i=1}^{m} \nabla_\theta \left(-x_i\theta(1-y_i) - \log(1+e^{-x_i\theta}) \right) \\ &= \sum_{i=1}^{m} -x_i(1-y_i) + \frac{e^{-x_i\theta}}{1+e^{-x_i\theta}}, \end{aligned} \tag{5.4}$$

where, again, m denotes the sub-sample size and x_i stands for the i^{th} row of the expanded version of X, and y_i denotes the corresponding i^{th} component of Y.

Additionally, we use a diagonal preconditioning matrix D in order to avoid divergence of θ values. Using this matrix with appropriate values for the step-size parameters a, b and γ helps θ converge. Combining the gradient log-likehood estimations with the noise term and the diagonal preconditioning matrix D, we obtain the update for θ

$$\Delta\theta^{(t)} = \frac{\epsilon^{(t)}}{2} \times D \times \left(-\Sigma_\theta^{-1}(\theta^{(t)} - \mu_\theta) + \frac{n}{m} \left(\sum_{i=1}^{m} -x_i^{(t)}(1 - y_i) + \frac{e^{-x_i^{(t)}\theta^{(t)}}}{1 + e^{-x_i^{(t)}\theta^{(t)}}} \right) \right) + \eta^{(t)},$$

where i represents the i^{th} row of the expanded matrix of X, y_i represents the i^{th} component of Y, t represents the iteration number, m is the subsample size, $\eta^{(t)} \sim \mathcal{N}(0, \epsilon^{(t)})$ represents the Gaussian noise term, and $\epsilon^{(t)} = a(b + t)^\gamma$ represents the step-size which is required to satisfy the convergence properties given in (2.9). In order to obtain step-sizes that decay polynomially (satisfy the convergence property) [21], we adjust the parameters a, b and γ in the calculation of step-sizes.

We add a burn-in period to the SGLD algorithm in order to eliminate the first replacements for the missing components of X. In the outer iteration, we select a random subset, but the replacement of its unobserved elements takes place in an inner iteration. After this burn-in period, the first burn-in number of them are discarded, and the obtained last version of subset X is used to estimate the parameter vector.

5.3. Experiments and Results

The categorical dataset has information about $32,561$ people in five different categories. There are missing variables in two of these categories and they are indicated by the response indicator matrix A. Since the data is not from a certain distribution, the simulations are run under some assumptions. The first assumption is that the probability distribution of a variable in X is the ratio of occurrence number of that specific variable to total number of occurrences. Under this distribution, the second assumption is that all the variables are independent so imputing the latent variables does not affect the probability of others. We have also assumed that the education level is proportional to the income level, *i.e.* there is a linear relationship with education

and income. The parameter components whose corresponding categories have only two different choices, like sex, is also scalar, since the effect can be only positive or negative. On the other hand for other categories such as work class or marital status, it is not easy to construct a linear relationship between income level and his/her response to that category. This is because we extend the component to multiple components such that each label in the category has its own sub-component. To be more rigorous, suppose that the second category, which matches the second component of θ has l different choices. Then θ_2 is extended to l components, $\theta_2 = (\theta_{2,1}, \theta_{2,2}, ..., \theta_{2,l})$ and so $\theta = (\theta_1, \theta_{21}, \theta_{22}, ..., \theta_{2l}, \theta_3, ..., \theta_d)$.

We have simulated MH and SGLD algorithms for different subsample sizes and iteration numbers. The way we compare the results are the same as the comparison for the normally distributed dataset (Chapter 4). We measure the predicted accuracy and computation time of each method to determine which one is more effective or preferable. Table 5.1 and Table 5.2 show the results of the simulations for subsample sizes 100 and 500, respectively. The prediction accuracy is approximately 0.71 for MH algorithm and it can be observed that increasing the number of iterations does not have a significant impact on the success rate of the prediction, while it greatly affects the computation time. The similar interpretation is valid for SGLD algorithm as well, except for 10^6 iterations with subsample of size 500, where the prediction accuracy rose up to 76%. In Table 5.3, comparison of prediction accuracy and elapsed times with a subsample of size $1,000$ are presented. The prediction accuracy is still around 70% for both algorithms. Unlike the results of artificial data, we have observed that increasing subsample size does not increase the success rate of prediction. It should also be noted that the inner iteration number of SGLD method is chosen as five, while three of them are discarded by the burn-in period, like in the experiments conducted for artificial data.

Table 5.1: Comparison of MH and SGLD algorithms when subsample size is 100 (categorical dataset).

Number of Iterations	Inference Method	Prediction Accuracy	Elapsed Time
5×10^5	MH sampling	0.704	247.50 sec
	SGLD sampling	0.715	234.52 sec
1×10^6	MH sampling	0.717	550.90 sec
	SGLD sampling	0.711	515.54 sec
2×10^6	MH sampling	0.720	1844.55 sec
	SGLD sampling	0.714	1019.00 sec

Table 5.2: Comparison of MH and SGLD algorithms when subsample size is 500 (categorical dataset).

Number of Iterations	Inference Method	Prediction Accuracy	Elapsed Time
1×10^5	MH sampling	0.700	649.04 sec
	SGLD sampling	0.716	289.85 sec
5×10^5	MH sampling	0.709	3230.17 sec
	SGLD sampling	0.722	1491.40 sec
1×10^6	MH sampling	0.715	6557.66 sec
	SGLD sampling	0.763	3011.58 sec

Since the parameter vector θ has components whose characteristics are different from each other, it might be expected that the correctness of the estimations will not be the same for every component. In order to observe the components separately, Figure 5.1 depicts some of the components of θ, when the subsample size is 500, and iteration number is 10^6. The graph shows that both algorithms converge for these selected components. It also can be observed that linear components, such as developed-underdeveloped countries and education, attain the convergence in a shorter time than other components do. Figures 5.2, 5.3 and 5.4 show the comparisons of the methods with various subsample sizes and the numbers of iterations. Figure 5.5 is the histogram of all components obtained by MH and SGLD algorithms when subsample size is $10,000$. The histograms also illustrate that the linear components like education, are more successful to converge.

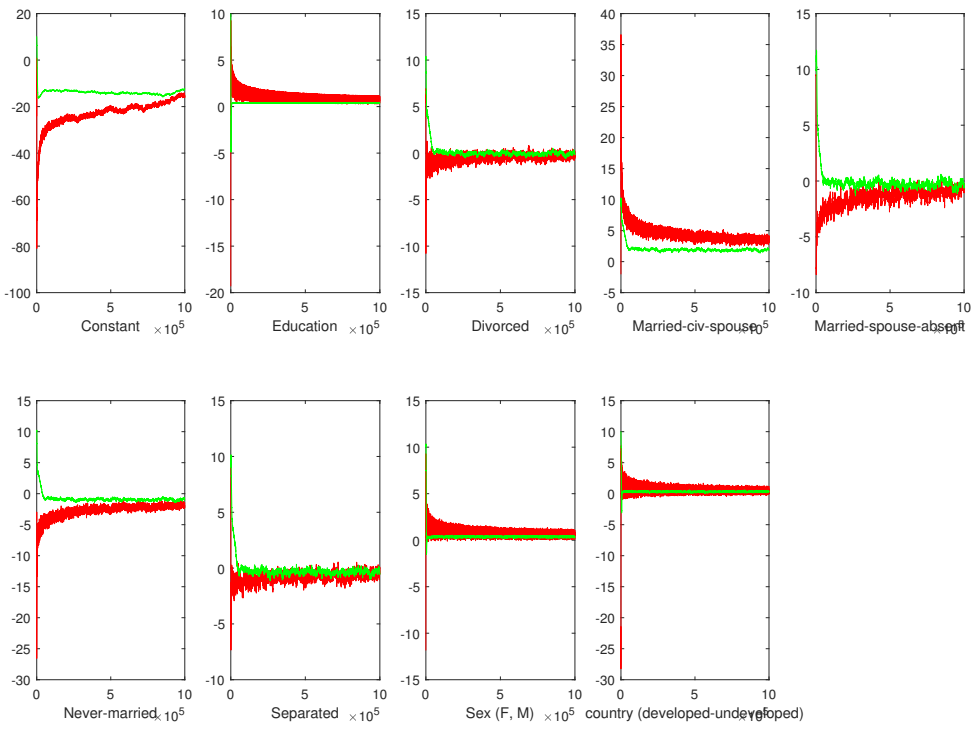

Figure 5.1: The estimated values obtained by MH (green lines) and SGLD (red lines) methods for selected components of θ, with $m = 500$, number of iterations 10^6.

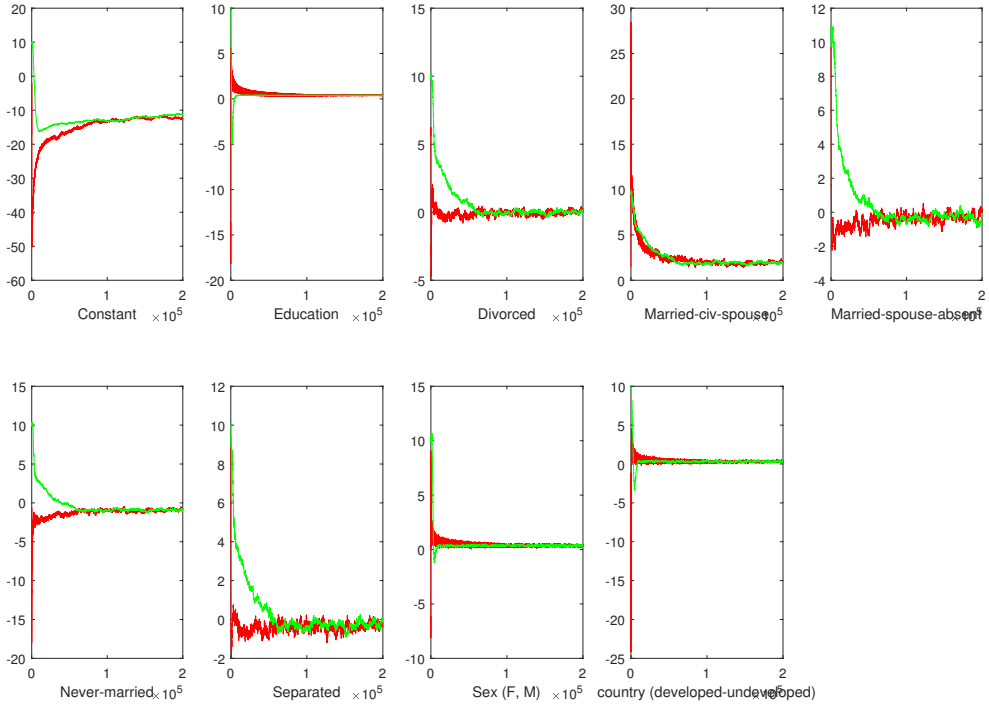

Figure 5.2: The estimated values obtained by MH (green lines) and SGLD (red lines) methods for selected components of θ, with $m = 100$, number of iterations 2×10^5.

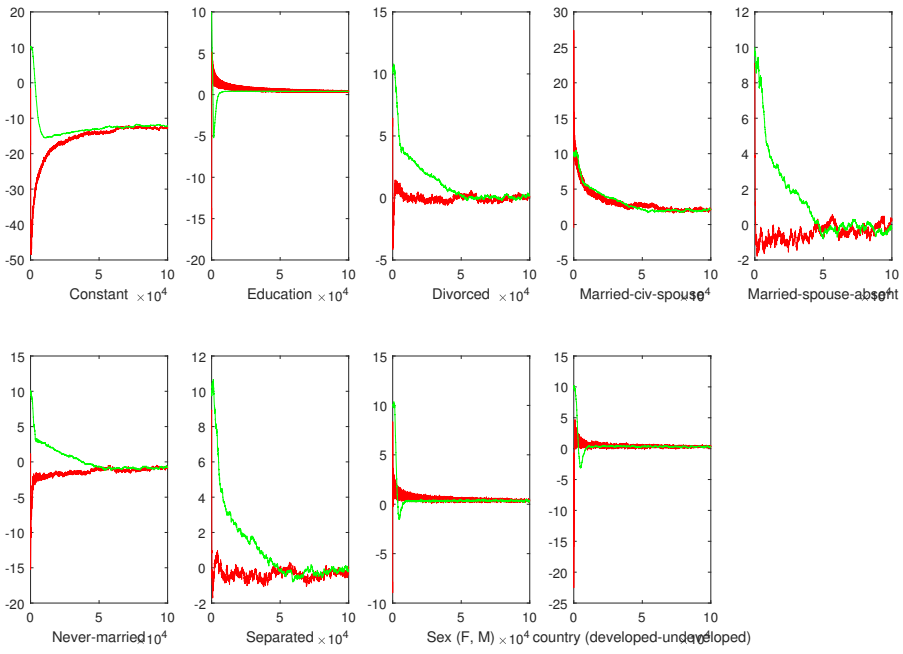

Figure 5.3: The estimated values obtained by MH (green lines) and SGLD (red lines) methods for selected components of θ, with $m = 1,000$, number of iterations 10^5.

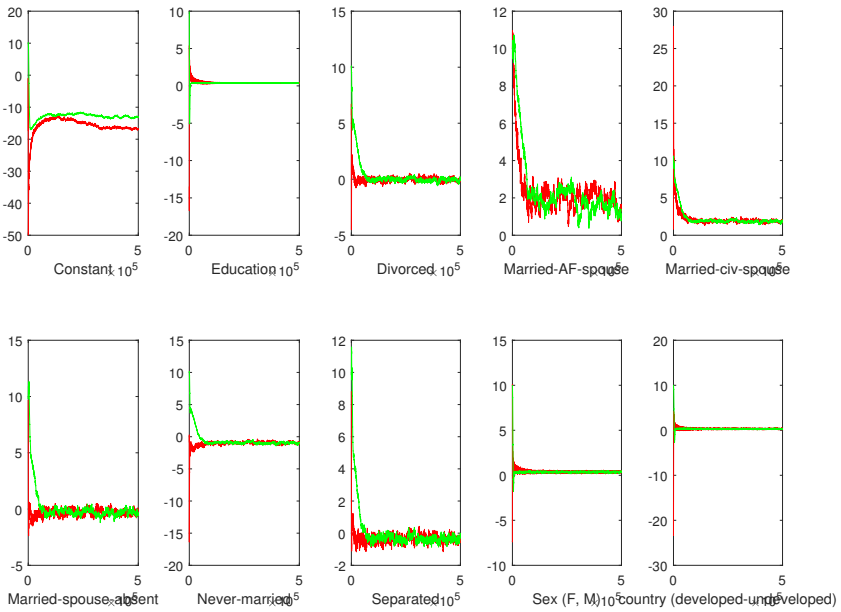

Figure 5.4: The estimated values obtained by MH (green lines) and SGLD (red lines) methods for selected components of θ, when $m = 1,000$, number of iterations 5×10^5.

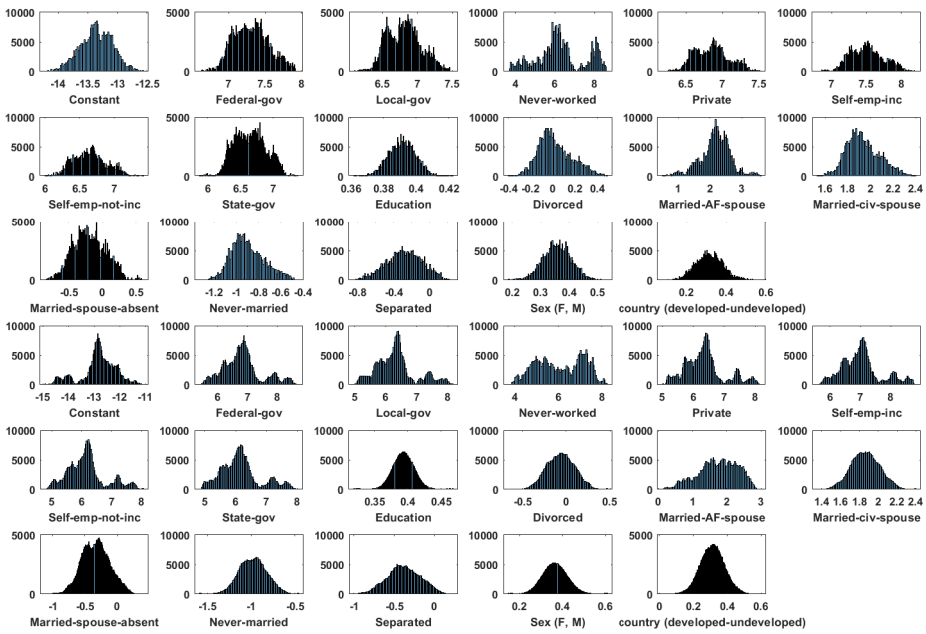

Figure 5.5: Histograms of selected components of θ, obtained by MH (first three lines) and SGLD methods (last three lines) with $m = 10,000$, number of iterations 10^5.

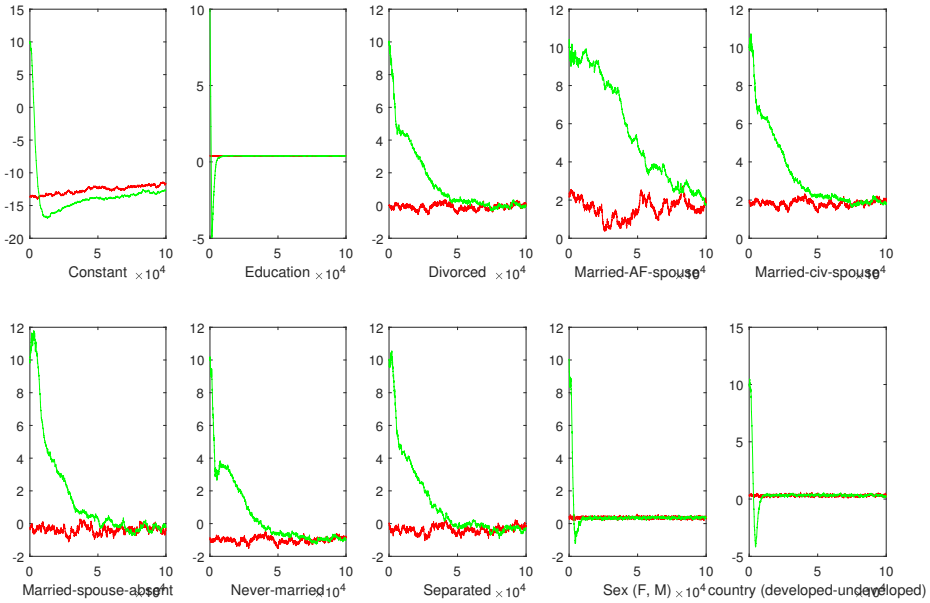

Figure 5.6: The estimated values obtained by MH (green lines) and SGLD (red lines) methods for selected components of θ, with $m = 10,000$, number of iterations 10^5.

Like in the previous chapter, we also implemented complete case analysis algorithm where the idea of excluding latent variables is applied [7]. The comparison of three methods is shown in Table 5.3 and Table 5.4. The tables provide the prediction successes and calculation times of the algorithms for different numbers of iterations. Table 5.3 shows the simulation results for a subsample size of $1,000$, whereas Table 5.3 shows the results for a subsample size of $10,000$. Using $10,000$ lines means that we make computations through 30% of the data approximately. For MH and SGLD methods, we can observe that although the portion of the data that we use and impute is relatively high, the prediction accuracy rates are not remarkably different than the rates obtained by using smaller subsets. Considering the time performances of the simulations with different sizes of subsets, it might be wiser to choose small subsamples and increase the number of iterations for MH and SGLD methods, since their prediction successes are close to the values presented in tables 5.1 and 5.2. In addition to this, an obvious interpretation of the results shown in the tables is that, the complete case analysis algorithm has reached a success rate of 71% in a smaller amount of time than the MH and the SGLD algorithms. This might be because of the small proportion

(approximately 5%) of latent variables in the real dataset.

Figure 5.7 shows the posterior mean and Figure 5.8 shows the posterior variance of each component of θ, for a subsample size of 500 and number of iterations is 10^6. For MH and SGLD algorithms, the results shown in both figures are the same simulation results as given in Table 5.2, where the prediction value is approximately 71% for MH, and 72% for SGLD. Since the prediction accuracy rates are close to each other, the posterior mean values are as expected, even though there are differences between some of the components. It is also clear that the posterior means obtained by the complete case analysis algorithm are fairly close to the results of the SGLD algorithm. One can observe from Figure 5.8 that the complete case analysis has the largest variance for most components.

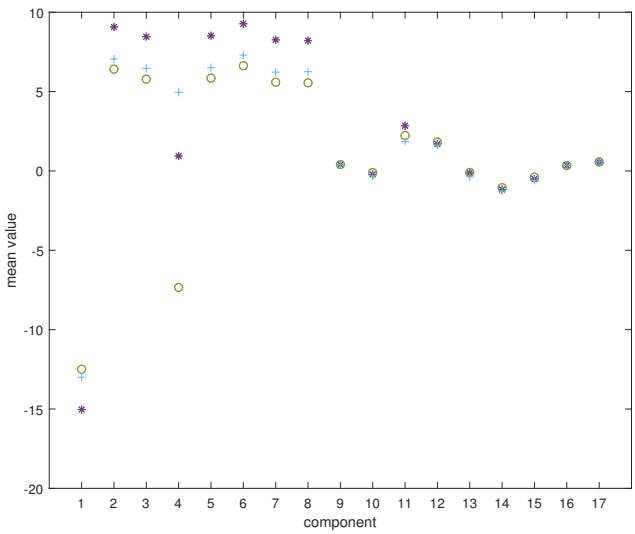

Figure 5.7: The posterior means of the components, obtained by MH (*) and SGLD (o) and complete case (+) algorithms, with $m = 500$, number of iterations 10^6.

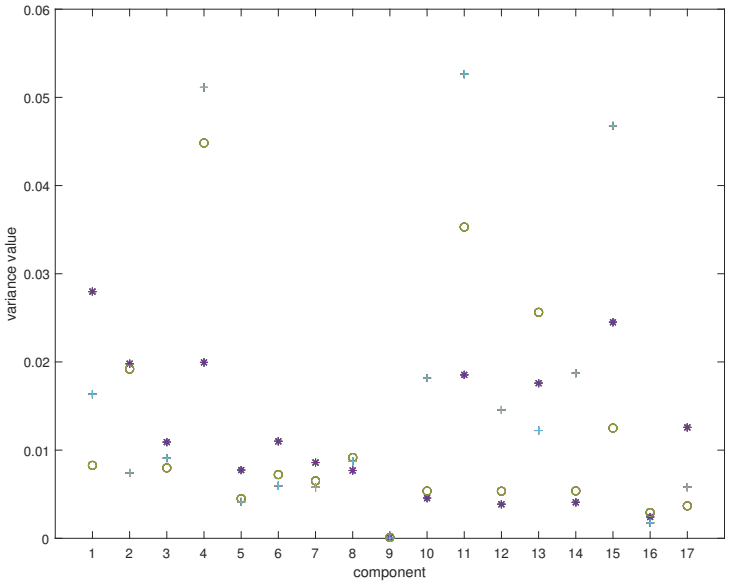

Figure 5.8: The posterior variances of the components, obtained by MH (*) and SGLD (o) and complete case (+) algorithms, with $m = 500$, number of iterations 10^6.

Table 5.3: Comparison of MH, SGLD and complete case analysis algorithms when subsample size is $1,000$ (categorical dataset).

Number of Iterations	Inference Method	Prediction Accuracy	Elapsed Time
1×10^5	MH sampling	0.710	719.29 sec
	SGLD sampling	0.709	521.37 sec
	Complete case	0.710	568.23 sec
2×10^5	MH sampling	0.713	1436.21 sec
	SGLD sampling	0.712	1113.39 sec
	Complete case	0.712	957.23 sec
5×10^5	MH sampling	0.731	3637.63 sec
	SGLD sampling	0.728	2668.20 sec
	Complete case	0.715	2349.67 sec

Table 5.4: Comparison of MH, SGLD and complete case algorithms when subsample size is $10,000$ (categorical dataset).

Number of Iterations	Inference Method	Prediction Accuracy	Elapsed Time
5×10^4	MH sampling	0.709	1027.85 sec
	SGLD sampling	0.711	897.12 sec
	Complete case	0.712	578.14 sec
1×10^5	MH sampling	0.721	2035.92 sec
	SGLD sampling	0.715	1518.85 sec
	Complete case	0.713	1254.35 sec

6. CONCLUSION AND DISCUSSION

The main goal of this thesis is to investigate the missing data problem in large-scale datasets. We have developed two different strategies in order to address this problem. The proposed methods are based on the Bayesian inference and Markov chain Monte Carlo methods, where prior knowledge and the likelihood function provide an insight for posterior knowledge. Both of the proposed methods use the Metropolis-Hastings idea for the imputation of unobserved parts, where the latent variables are imputed by proposing new values based on a proposal distribution. The first introduced technique uses the MH sampling for estimating parameters, while the second technique employs the SGLD idea in which the calculations are made over the small subset of the dataset and updates for parameters are generated via the gradient information. Since we have utilized the MCMC and SGLD techniques to sample the missing variables, the subsampling idea is applied on the imputation part of both algorithms.

The proposed methods and complete case analysis (or listwise deletion) method are tested and compared on synthetic data having Gaussian distribution as well as real-categorical data. The methods are compared according to their time and error performances. Our experiments have shown that their inferences are fairly close to each other. A significant difference has occurred when we compare the execution time of the algorithms. We have observed that the SGLD-based algorithm has outperformed the MH and complete case algorithms. Even though the prediction accuracy rates are close, choosing SGLD algorithm is a wiser approach, since it is not fast to make calculations over the entire dataset. Especially when the dataset becomes larger, the advantage of SGLD algorithm comes into prominence. On the other hand, complete case analysis algorithm has provided a successful prediction accuracy with shorter execution time than MH for 10% incompleteness. However, when the sparsity of the data becomes higher, the algorithm is outperformed by SGLD and MH, even though its computation time is relatively small.

The first contribution of the thesis is that, introducing the SGLD approach to

missing data problems in big datasets has resulted in the parameter estimations as good as the MH approach. Since MH is a sampling method that provides exact solutions, it is considered as a benchmark of the comparisons, and the success of SGLD is measured by its close results to MH. Considering the prediction success rates of the algorithms, it can be stated that using gradient estimations instead of exact sampling methods might be more efficient, especially for the cases where the data size is extremely large. The SGLD algorithm has attained its convergence in a relatively small amount of time when it is compared to the MH algorithm for parameter estimations.

The second contribution of the thesis to MCMC methods for incomplete data is that, we have proposed an alternative approach through MH sampling, for the cases in which sampling from full conditional distributions is not possible or easy. MH sampling not only provides a solution to sampling problem, it also provides an improvement in the performance of the method due to its computational efficiency. In addition to this, applying the idea of subsampling on imputation part provides a remarkable decrease in execution time of the algorithm.

Considering the correctness of parameter estimations that we tested with both synthetic and real datasets, the results of MH and SGLD algorithms are quite promising. Especially for the Gaussian distributed synthetic data, the prediction accuracy rates have attained almost 95%. This high success rate of prediction is a strong motivation for further research on regression with missing data by utilizing MH and SGLD frameworks.

In the scope of this thesis, we have used a limited number of datasets to test the methods. Although it has been proven that the methods that we propose have performed quite well for our datasets, it would be wiser to apply these algorithms on the various real datasets. For most the real-life applications, it is not always easy to specify the prior probability distribution of given data. This is because one may need to make some assumptions at the beginning of the calculations. Testing the assumptions which are made for different distributions, and proposing better initial assumptions would improve the performance of the methods. Moreover, since it will be more costly

to make calculations when the dataset becomes large, the applications on even larger datasets would help show the differences between two algorithms.

For the experiments with real dataset, we have determined the prior probability distribution of variables by the frequency of the observed variables, and we have not updated the initial distribution after we replaced the unobserved variables. However, since the frequency of the variables will change once the new variables are replaced, updating the prior probability of the variables would be a wiser assumption for the prior probability distribution. The algorithms can be performed better, and they might provide higher rates of prediction accuracy under the updated prior probability distributions.

In the experiments that we have conducted, we have used symmetric random walk as the proposal distribution for the synthetic data, while we have used an independent proposal distribution for the categorical data. These proposal distributionos are two simple proposal kernels in Metropolis-Hastings sampling. Since the choice of proposal kernel plays a crucial role in the performance of MH algorithm, one future work might be to apply the algorithms with more sophisticated proposal kernel functions.

REFERENCES

1. Little, R. and D. Rubin, *Statistical Analysis with Missing Data, Second Edition*, Wiley, Hoboken, NJ, 2002.

2. Gold, M. S. and P. M. Bentler, "Treatments of Missing Data: A Monte Carlo Comparison of RBHDI, Iterative Stochastic Regression Imputation, and Expectation-Maximization", *Lawrence Erlbaum Associates, Inc.*, pp. 319–355, 2000.

3. Sung, Y. J. and C. J. Geyer, "Monte Carlo likelihood inference for missing data models", *Ann. Statist.*, Vol. 35, No. 3, pp. 990–1011, 07 2007.

4. K Enders, C., "Using the Expectation Maximization Algorithm to Estimate Coefficient Alpha for Scales With Item-Level Missing Data", Vol. 8, pp. 322–37, 10 2003.

5. C Yuan, Y., "Multiple Imputation for Missing Data: Concepts and New Development", *SAS Institute Inc.*, January 2005.

6. Soley-Bori, M., *Dealing with missing data: Key assumptions and methods for applied analysis*, Tech. rep., Boston University School of Public Health Department of Health and Policy Management, May 2010.

7. Rubin, D. B., *EM and beyond*, Vol. 56, Psychometrica, 1991.

8. Rubin, D. B., *Multiple Imputation for Nonresponse in Surveys*, John Wiley Sons, Inc., New York, 1987.

9. Propp, J. G. and D. B. Wilson, "Exact Sampling with Coupled Markov Chains and Applications to Statistical Mechanics", *Random Struct. Algorithms*, Vol. 9, No. 1-2, pp. 223–252, Aug. 1996.

10. Lemieux, C. and P. Sidorsky, "Exact sampling with highly uniform point sets", *Mathematical and Computer Modelling*, Vol. 43, No. 3, pp. 339 – 349, 2006.

11. K., G. J., "Introduction to Applied Bayesian Statistics and Estimation for Social Scientists by Scott M. Lynch", *International Statistical Review*, Vol. 76, No. 2, pp. 311–312.

12. Gelman, A., J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari and D. B. Rubin, *Bayesian Data Analysis*, Tyler & Francis Group, Inc., 2014.

13. van Ravenzwaaij, D., P. Cassey and S. D. Brown, "A simple introduction to Markov Chain Monte Carlo sampling", *Psychonomic Bulletin & Review*, Vol. 25, No. 1, pp. 143–154, Feb 2018.

14. Yıldırım, S., "Simulation Methods for Statistical Inference, Sabanci University Lecture Notes", , January 2017, last accessed on April 2018.

15. O'hara, R. B., E. Arjas, H. Toivonen and I. Hanski, "Bayesian Analysis of Metapopulation Data", *Wiley 2002, on behalf of the Ecological Society of America*, pp. 1–4, 2002.

16. Acquah, H. D.-G., "Bayesian Logistic Regression Modelling via Markov Chain Monte Carlo Algorithm", *Journal of Social and Development Sciences*, Vol. 4, pp. 193–197, 2013.

17. Yıldırım, S., *Bayesian Methods for Deconvolution of Sparse Processes*, Master's Thesis, Boğaziçi University, 2009.

18. Metropolis, N. and S. Ulam, "The Monte Carlo Method", *Journal of the American Statistical Association*, Vol. 44, No. 247, 1949.

19. Hastings, W. K., "Monte Carlo Sampling Methods Using Markov Chains and Their Applications", *Biometrika*, Vol. 57, No. 1, pp. 97–109, 1970.

20. Gilks, W., S. Richardson and D. Spiegelhalter, *Markov Chain Monte Carlo in Practice*, Chapman & Hall/CRC Interdisciplinary Statistics, Taylor & Francis, 1995.

21. Welling, M. and Y. W. Teh, "Bayesian Learning via Stochastic Gradient Langevin Dynamics", *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, pp. 681–688, Omnipress, USA, 2011.

22. Robbins, H. and S. Monro, "A Stochastic Approximation Method", *The Annals of Mathematical Statistics*, Vol. 22, pp. 400–407, 1951.

23. Chen, T., E. B. Fox and C. Guestrin, "Stochastic Gradient Hamiltonian Monte Carlo", *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, pp. II–1683–II–1691, 2014.

24. Goldberger, A. S., "Best Linear Unbiased Prediction in the Generalized Linear Regression Model", *Journal of the American Statistical Association*, Vol. 57, No. 298, pp. 369–375, 1962.

25. García, M., C. Valverde, M. I. López, J. Poza and R. Hornero, "Comparison of logistic regression and neural network classifiers in the detection of hard exudates in retinal images", *2013 35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pp. 5891–5894, July 2013.

26. "The Logistic Regression Algorithm", `https://towardsdatascience.com/the-\logistic-regression-algorithm-75fe48e21cfa`, accessed: May 2018.

27. Dietterich, T. G., "Overfitting and Undercomputing in Machine Learning", *ACM Comput. Surv.*, Vol. 27, pp. 326–327, 1995.

28. Ding, Y. and A. Ross, "A comparison of imputation methods for handling missing scores in biometric fusion", *Pattern Recognition*, 2011.

29. E. Raghunathan, T., J. Lepkowski, J. H. Van Hoewyk and P. W. Solenberger, "A Multivariate Technique for Multiply Imputing Missing Values Using a Sequence of Regression Models", Vol. 27, November 2000.

30. Dixon, J. K., "Pattern Recognition with Partly Missing Data", *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 9, No. 10, pp. 617–621, 1979.

31. Ni, D. and J. D. L. II, "Markov Chain Monte Carlo Multiple Imputation Using Bayesian Networks for Incomplete Intelligent Transportation Systems Data", *Transportation Research Record: Journal of the Transportation Research Board, (1935)*, pp. 57–67, 2005.

32. Takahashi, M., "Statistical Inference in Missing Data by MCMC and Non-MCMC Multiple Imputation Algorithms: Assessing the Effects of Between-Imputation Iterations", *Data Science*, Vol. 16, pp. 1–17, 2017.

33. K. Ghosh, J., "Bayesian Methods: A Social and Behavioral Sciences Approach, Second Edition by Jeff Gill", Vol. 77, pp. 301–302, August 2009.