Optimization of Maximum Likelihood Function in the Presence of Missing Data

A Dissertation

Presented to the

Graduate Faculty of the

University of Louisiana at Lafayette

In Partial Fulfillment of the

Requirements for the Degree

Doctor of Philosophy

Thu Thi Nguyen

Summer 2021

Optimization of Maximum Likelihood Function in the Presence of Missing Data

Thu Thi Nguyen

APPROVED:

Dr. Bruce A. Wade, Chair C.B.I.T. TC/LEQSF Regents Professor and Department Head	Dr. Bruce A. Wade, Department Head C.B.I.T. TC/LEQSF Regents Professor and Department Head
Dr. Ralph Baker Kearfott Professor Emeritus, Mathematics	Dr. Xiang-Sheng Wang Assistant Professor, Mathematics
Dr. Longfei Li Assistant Professor, Mathematics	Dr. Gabriele Morra Associate Professor, Physics
Dr. Mary Farmer-Kaiser Dean of the Graduate School	

To my dear parents.

Acknowledgments

First and foremost, I would like to send my deepest gratitude to my advisor, Dr. Dr. Bruce A. Wade, for his guidance and supervision.

In addition, I would like to send my sincere appreciations to the other members of my dissertation committee, Dr. Ralph Baker Kearfott, Dr. Xiang-Sheng Wang, Dr. Longfei Li, and Dr. Gabriele Morra. Thanks to the Graduate Coordinators, Dr. Arturo Magidin and Dr. Ping Wong Ng for their guidance and help through my Ph.D. program.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

Missing data is a common problems in data analysis. For example, respondents in a survey may refuse to fill in the income, sensors may fail and need to be replaced, etc. These imply missing data challenges.

A special case of missing data problems is monotone missing data. This problem occurs in practical situations ubiquitously, especially in longitudinal studies where the information on a set of cases is collected repeatedly over time [McKnight et al.(2007)McKnight, McKnight, Sidani, and Figueredo]. For instance, a Parkinson's study may have multiple periods. All variables from the first block can be recorded at the beginning of the research and, therefore, fully observed. However, the second block of variables recorded two years later may not be fully observed as some of the patients have left the study. Similarly, the third block of variables, recorded three years later, can increase missing values due to attrition. Various other examples can be found in sensor or biological data when sensors or measurement devices, which are in data recording progress, encounter errors due to several external factors. Such situations usually imply a monotone missing data challenge.

The problem of monotone missing data has been broadly studied during the last two decades and has many applications in various fields such as bioinformatics or statistics. Yet, commonly used imputation techniques for missing data problems usually require multiple iterations through the data before yielding convergence. Moreover, those approaches may introduce extra noises and biases to the subsequent modeling. In this work, we derive formulas and propose a novel algorithm to compute the maximum likelihood estimators (MLEs) of a dataset directly. Specifically, in chapter 3, we disscuss

estimating parameters of a multiple class, monotone missing dataset when all the covariance matrices of all categories are assumed to be equal, namely, Efficient Parameter Estimation for Multiple Class Monotone Missing Data (EPEM). Next, in chapter 4, we extend the idea of chapter 3, with some modification to provide direct parameter estimation schemes for randomly missing data without the monotone missing assumption.

As the computation is direct via formulas, our algorithms do not require multiple iterations through the data as other imputation approaches, thus promising to be much less time-consuming than other methods. This effectiveness is validated by empirical results when our algorithms reduce the error rates significantly while requiring a shorter computation time compared to several imputation-based approaches.

The contributions of our work can be summarized as follows: (1) We derive MLE estimation for the mean and covariance matrix, and therefore, provide an approximation to the underlying distribution of the data; (2) We provide efficient algorithms to compute the MLEs from the data that is not only suitable for small data sets but also for the large-scale data; (3) We propose a linear discriminant procedure based on the MLEs with asymptotic properties; (4) The experimental results show that our algorithms can outperform popular imputation methods in terms of error rates, speed, memory requirements; (5) Our experiments on large datasets shows that while iterative methods may not handle big datasets with high missing rate, our approach can still find the parameter rapidly without requiring extensive memory/computation time; (6) We discuss the potential drawbacks of our methods and possible strategies in such cases.

Chapter 2: Related Works

So far, various methods have been developed to tackle the problem of missing data. They can be divided into several subcategories.

The first category includes all techniques that are based on matrix completion. For example, Polynomial Matrix Completion [Fan et al.(2020)Fan, Zhang, and Udell] tries to recover the missing entries of a matrix by assuming that the columns of a matrix are generated by polynomials acting on a low-dimensional intrinsic variable. Then, they minimize the rank of the matrix in a high dimensional feature space using the kernel trick and a rank objective relaxation. SOFT-IMPUTE
[Mazumder et al.(2010)Mazumder, Hastie, and Tibshirani], on the other hand, iteratively replaces the missing elements with those obtained from a soft-thresholded SVD. Next, Nuclear Norm Minimization [Candès and Recht(2009)] completes a matrix with missing data by sampling of its entries randomly.

The second category consists of regression-based and least-square techniques. For example, CBRL and CBRC
[M Mostafa et al.(2020)M Mostafa, S Eladimy, Hamad, and Amano] cumulate the imputed features; those imputed features will be incorporated within the Bayesian Ridge equation for predicting the missing values in the next incomplete selected feature. Another important technique is multiple imputation by chained equation
(MICE)[Buuren and Groothuis-Oudshoorn(2010)], which treat each feature with missing values as the dependent variable in a regression, with some or all of the remaining variables as its predictors. Then, the procedure cycles through these models, fitting each in turn, then it uses *predictive mean matching* to generate random draws from the

predictive distributions determined by the fitted models for imputing the missing values. In addition, some other methods are local least squares imputation [Kim et al.(2005)Kim, Golub, and Park], least trimmed squares imputation [Templ et al.(2011)Templ, Kowarik, and Filzmoser], etc.

The third one consists of tree-based techniques. MissForest [Stekhoven and Bühlmann(2012)] is an iterative imputation method based on a random forest which can cope with different types of variables simultaneously. Next, in [Burgette and Reiter(2010)], the authors present a nonparametric approach for implementing multiple imputation via chained equations by using sequential regression trees as the conditional models. On the other hand, DMI algorithm [Rahman and Islam(2013)], integrates decision trees and fuzzy clustering into an iterative learning approach.

Next, the fourth category contains clustering-based techniques. For example, [Razavi-Far et al.(2020)Razavi-Far, Cheng, Saif, and Ahmadi], proposes the $kEMI$ and $kEMI^+$ methods that are based on the k-Nearest Neighbours algorithm for pre-imputation and the EM algorithm for posterior-imputation. DIFC [Nikfalazar et al.(2020)Nikfalazar, Yeh, Bedingfield, and Khorshidi], on the other hand, integrates the merits of decision tress and fuzzy clustering into an iterative learning approach. Another hybrid approach is to utilize a Fuzzy c-means clustering [Aydilek and Arslan(2013)], which is a fuzzy c-means clustering hybrid approach that combines support vector regression and a genetic algorithm. Some other techniques in this category can be fuzzy clustering-based EM imputation [Rahman and Islam(2016)], imputation using fuzzy neighborhood density-based clustering [Razavi-Far and Saif(2016)],

imputation using Hybrid K-Means and Association Rules
[Chhabra et al.(2018)Chhabra, Vashisht, and Ranjan], and evolutionary clustering method [Gautam and Ravi(2015)].

Also, one can find techniques that are based on extreme learning, for example, AAELM [Gautam and Ravi(2015)], extreme learning machine multiple imputation [Sovilj et al.(2016)Sovilj, Eirola, Miche, Björk, Nian, Akusok, and Lendasse], or Bayesian approaches, which are Naive Bayesian imputation [Garcia and Hruschka(2005)], Bayesian network imputation [Hruschka et al.(2007)Hruschka, Hruschka, and Ebecken], and Bayesian principal component analysis-based
imputation[Audigier et al.(2016)Audigier, Husson, and Josse].

In addition, deep learning techniques also form another dominant class of imputation methods. Those techniques include multiple imputations using Deep Denoising Autoencoders [Gondara and Wang(2017)], imputation via Stacked Denoising Autoencoders [Costa et al.(2018)Costa, Santos, Soares, and Abreu]. In [Spinelli et al.(2020)Spinelli, Scardapane, and Uncini], the authors formulate the problem using a graph denoising autoencoder, where each edge of the graph encodes the similarity between two patterns. Next, Generative Adversarial Multiple Imputation Network [Yoon and Sull(2020)] is a generative adversarial multiple imputation network (GAMIN) based on generative adversarial network (GAN) for multiple imputation. Next, Swarm Intelligence-deep neural network [Leke and Marwala(2016)] combines Gravitational search algorithm with a deep-autoencoder
[Garg et al.(2018)Garg, Naryani, Aggarwal, and Aggarwal]. However, a significant drawback of deep learning methods is the need for a lot of data.

A final category consists of fuzzy imputation techniques such as the evolving granular fuzzy-rule-based model [Garcia et al.(2019a)Garcia, Esmin, Leite, and Škrjanc], [Garcia et al.(2019b)Garcia, Leite, and Skrjanc], fuzzy c-means clustering [Aydilek and Arslan(2013)], fuzzy clustering-based EM imputation [Rahman and Islam(2016)], and imputation using fuzzy neighborhood density-based clustering [Razavi-Far and Saif(2016)].

Up to now, there have been several results in estimating the parameters directly from the data [Anderson(1957), Fujisawa(1995), Nguyen et al.(2021a)Nguyen, Nguyen, Nguyen, Nguyen, and Wade] without iterative processes. Anderson [Anderson(1957)] derived the MLEs for the one-group two-step monotone case. Yu et al. [Yu et al.(2006)Yu, Krishnamoorthy, and Pannala] gave the MLEs for the two-group three-step monotone case. Fujisawa [Fujisawa(1995)] derived the MLEs in closed and understandable forms for the mean vector and the covariance matrices of the general one-group monotone case by using a conditional set-up. Kanda et al. [Kanda and Fujikoshi(1998)] studied different properties of the MLEs for a multivariate normal population based on k-step monotone data. However, to our knowledge, there has not been any study about the general case for multiple class monotone missing data, where the data from all classes are assumed to have the same covariance matrix.

In addition, to our knowledge, these approaches rely on the assumption that the data follows a multivariate normal distribution, and can only be used for monotone missing data. On the other hand, in the second part of our work (chapter 4), we only assume pairwise normality (i.e., each pair of features follows a bi-variate normal distribution), and works for randomly missing data in general.

Chapter 3: Parameter Estimation for Multiple Class Monotone Missing Data

3.1 Introduction

In various applications such as linear discriminant analysis, hypothesis testings, or when the sample size is small, the assumption of equal covariance matrix is usually utilized/needed. Therefore, in this chapter, we will describe our approaches to calculate the parameter estimation in multiple-class monotone missing data under the assumption of equal covariance matrix.

This chapter is organized as follows. In Section 3.3, we present the formulas and algorithms for computing the means and covariance matrix for the case that the data has two classes as well as for the general case of multiple categories. Section 3.4 is devoted to an application in linear discriminant analysis. We illustrate our approach's power via experiments in Section 3.5 and discuss the implications of the experimental results. The chapter ends with our conclusion.

3.2 Monotone Missing Data Patterns

A data set with ordered features $Z_1, Z_2, \ldots, Z_p$ is said to have a monotone missing pattern when the event that a feature Z_j is missing for a particular sample implies that all subsequent features $Z_k, k > j$, are missing for that sample. Alternatively, when a feature is observed for a particular sample, it is assumed that all previous features $Z_h, h < j$, are also observed for that individual.

For example, the following displays a data set of four features with a monotone missing pattern.

$$\begin{pmatrix} 3 & 5 & 1 & 9 & 7 \\ 2 & 1 & 0 & 8 & * \\ 1 & 2 & 6 & * & * \\ 4 & 3 & * & * & * \end{pmatrix},$$

and

$$\begin{pmatrix} 3 & 5 & 1 & 9 & 7 \\ 2 & 1 & 0 & 8 & * \\ 1 & 2 & 6 & * & * \\ 4 & 3 & * & * & * \\ 6 & * & * & * & * \end{pmatrix},$$

a data set of five features with a monotone missing pattern.

3.3 Methodology

In this section, we will describe our proposed approaches to calculate the maximum likelihood estimators in multiple-class monotone missing data under the assumption of equal covariance matrix.

3.3.1 MLEs for two-class monotone missing data.

3.3.1.1 Data partition. Let $\boldsymbol{D} = [\boldsymbol{y}, \boldsymbol{z}]$ be a two-class data set having missing values that can be partitioned into the following monotone patterns:

$$\boldsymbol{y} = \begin{pmatrix} \boldsymbol{y}_{11} & \cdots & \boldsymbol{y}_{1n_k} & \cdots & \boldsymbol{y}_{1n_2} & \cdots & \boldsymbol{y}_{1n_1} \\ \boldsymbol{y}_{21} & \cdots & \boldsymbol{y}_{2n_k} & \cdots & \boldsymbol{y}_{2n_2} & \cdots & * \\ \boldsymbol{y}_{31} & \cdots & \boldsymbol{y}_{3n_k} & \cdots & * & \cdots & * \\ \vdots & & \vdots & \ddots & \vdots & \vdots & \vdots \\ \boldsymbol{y}_{k1} & \cdots & \boldsymbol{y}_{kn_k} & \cdots & * & \cdots & * \end{pmatrix},$$

$$\boldsymbol{z} = \begin{pmatrix} \boldsymbol{z}_{11} & \cdots & \boldsymbol{z}_{1m_k} & \cdots & \boldsymbol{z}_{1m_2} & \cdots & \boldsymbol{z}_{1m_1} \\ \boldsymbol{z}_{21} & \cdots & \boldsymbol{z}_{2m_k} & \cdots & \boldsymbol{z}_{2m_2} & \cdots & * \\ \boldsymbol{z}_{31} & \cdots & \boldsymbol{z}_{3m_k} & \cdots & * & \cdots & * \\ \vdots & & \vdots & \ddots & \vdots & \vdots & \vdots \\ \boldsymbol{z}_{k1} & \cdots & \boldsymbol{z}_{km_k} & \cdots & * & \cdots & * \end{pmatrix},$$

Here, all samples in y belong to the first class, all samples in z belong to the second class, and each column corresponds to a sample. In addition, k is the number of blocks, and each

"$*$" denotes a missing block of values. $\boldsymbol{y}_{in_j}$ and the "$*$" at the (i, n_j) position have size $p_i \times 1$.

Note that the number of features in $y_{i1}, \ldots, y_{in_k}, \ldots, y_{in_i}$ is the same for each $i = 1, 2, \ldots, k$. The above data partitioning means there are n_1, m_1 observations available on the first p_1 variables, n_2, m_2 observations available on the first $p_1 + p_2$ variables, and so on, for the first and the second classes, respectively. Typically, one can partition $\boldsymbol{y}$ into:

$$
\begin{aligned}
\boldsymbol{y}_1 &= \begin{pmatrix} \boldsymbol{y}_{11} & \cdots & \boldsymbol{y}_{1n_k} & \cdots & \boldsymbol{y}_{1n_2} & \cdots & \boldsymbol{y}_{1n_1} \end{pmatrix}_{p_1 \times n_1}, \\
\boldsymbol{y}_2 &= \begin{pmatrix} \boldsymbol{y}_{11} & \cdots & \boldsymbol{y}_{1n_k} & \cdots & \boldsymbol{y}_{1n_2} \\ \boldsymbol{y}_{21} & \cdots & \boldsymbol{y}_{2n_k} & \cdots & \boldsymbol{y}_{2n_2} \end{pmatrix}_{(p_1+p_2) \times n_2}, \\
&\vdots \\
\boldsymbol{y}_k &= \begin{pmatrix} \boldsymbol{y}_{11} & \cdots & \boldsymbol{y}_{1n_k} \\ \boldsymbol{y}_{21} & \cdots & \boldsymbol{y}_{2n_k} \\ \vdots & \ddots & \vdots \\ \boldsymbol{y}_{k1} & \cdots & \boldsymbol{y}_{kn_k} \end{pmatrix}_{(\sum_{j=1}^{k} p_j) \times n_k}.
\end{aligned}
$$

Similarly, one can determine k partitions of $\boldsymbol{z}$, including $\boldsymbol{z}_1$, $\boldsymbol{z}_2$, …, $\boldsymbol{z}_k$.

A specific partitioning example. For the rest of this chapter, we denote by $*$ a missing entry (not to be confused with the notation $*$ of a missing block of values).

Given

$$
\begin{pmatrix} 3 & 5 & 1 & 9 & 7 \\ 2 & 1 & 0 & 8 & 4 \\ 1 & 2 & * & * & * \\ 4 & 3 & * & * & * \end{pmatrix},
$$

In which,

$$
y = \begin{pmatrix} 3 & 1 \\ 2 & 0 \\ 1 & * \\ 4 & * \end{pmatrix}, \quad z = \begin{pmatrix} 5 & 9 & 7 \\ 1 & 8 & 4 \\ 2 & * & * \\ 3 & * & * \end{pmatrix}.
$$

Then

$$
y_1 = \begin{pmatrix} 3 & 1 \\ 2 & 0 \end{pmatrix}, \quad z_1 = \begin{pmatrix} 5 & 9 & 7 \\ 1 & 8 & 4 \end{pmatrix},
$$

and

$$y_2 = \begin{pmatrix} 3 \\ 2 \\ 1 \\ 4 \end{pmatrix} \quad z_2 = \begin{pmatrix} 5 \\ 1 \\ 2 \\ 3 \end{pmatrix}.$$

3.3.1.2 Maximum likelihood estimators. For the rest of the article, we denote $\hat{\theta}$ as the maximum likelihood estimate of a parameter θ. Suppose that the data from the first class ($\boldsymbol{y}$) follow a multivariate normal distribution with mean $\boldsymbol{\mu}$, and the data from the second class ($\boldsymbol{z}$) follow a multivariate normal distribution with mean $\boldsymbol{\eta}$, and both classes have the same covariance matrix $\boldsymbol{\Sigma}$. Let $\bar{\boldsymbol{y}}_i$ and $\mathbf{S}_i$ be the mean and covariance matrices of $\boldsymbol{y}_i$, $\bar{\boldsymbol{z}}_i$ and $\mathbf{R}_i$ be the mean and covariance matrices of $\boldsymbol{z}_i$, respectively. Then, one can define

$$\mathbf{W}_i = (n_i - 1)\mathbf{S}_i + (m_i - 1)\mathbf{R}_i, \tag{3.1}$$

as the *sum of square and cross product matrix* of $\boldsymbol{y}_i$ and $\mathbf{z}_i$, and Σ_i as the leading principal submatrix of Σ of size $\left(\sum_{j=1}^{i} p_j\right) \times \left(\sum_{j=1}^{i} p_j\right)$, for any $i \in \{1, 2, \ldots, k\}$. We denote

$$\boldsymbol{\mu} = \begin{pmatrix} [\boldsymbol{\mu}]_1 \\ [\boldsymbol{\mu}]_2 \\ \vdots \\ [\boldsymbol{\mu}]_k \end{pmatrix}, \boldsymbol{\mu}_i = \begin{pmatrix} [\boldsymbol{\mu}]_1 \\ [\boldsymbol{\mu}]_2 \\ \vdots \\ [\boldsymbol{\mu}]_i \end{pmatrix}, \tag{3.2}$$

where $[\boldsymbol{\mu}]_j$ is a vector of order $p_j \times 1$ $(j = 1, \ldots, k)$. To represent $\widehat{\boldsymbol{\Sigma}}_i$ in terms of $\widehat{\boldsymbol{\Sigma}}_{i-1}$ as well as $\widehat{\boldsymbol{\mu}}_i$ in terms of $\widehat{\boldsymbol{\mu}}_{i-1}$, we partition $\bar{\boldsymbol{x}}_i$,Σ_i, and $\mathbf{W}_i$ as follows:

$$\bar{\boldsymbol{y}}_{\boldsymbol{i}} = \begin{pmatrix} [\bar{\boldsymbol{y}}_i]_1 \\ [\bar{\boldsymbol{y}}_i]_2 \end{pmatrix}, \boldsymbol{\Sigma_i} = \begin{pmatrix} [\boldsymbol{\Sigma_i}]_{\mathbf{11}} & [\boldsymbol{\Sigma_i}]_{\mathbf{12}} \\ [\boldsymbol{\Sigma_i}]_{\mathbf{21}} & [\boldsymbol{\Sigma_i}]_{\mathbf{22}} \end{pmatrix}, \mathbf{W_i} = \begin{pmatrix} [\mathbf{W_i}]_{\mathbf{11}} & [\mathbf{W_i}]_{\mathbf{12}} \\ [\mathbf{W_i}]_{\mathbf{21}} & [\mathbf{W_i}]_{\mathbf{22}} \end{pmatrix} \quad \boldsymbol{i = 2, \ldots, k}, \tag{3.3}$$

where $[\bar{\boldsymbol{y}}_i]_1$ and $[\bar{\boldsymbol{y}}_i]_2$ have the size of $(\sum_{j=1}^{i-1} p_j) \times 1$, $p_i \times 1$, respectively; $[\Sigma_i]_{11}$ and $[\mathbf{W}_i]_{11}$ have the size $\left(\sum_{j=1}^{i-1} p_j\right) \times \left(\sum_{j=1}^{i-1} p_j\right)$; $[\Sigma_i]_{12}$ and $[\mathbf{W}_i]_{12}$ are matrices of size $\left(\sum_{j=1}^{i-1} p_j\right) \times p_i$; $[\Sigma_i]_{22}$ and $[\mathbf{W}_i]_{22}$ have the size $p_i \times p_i$. Then, $[\Sigma_i]_{11} = \Sigma_{i-1}$. Next, we can create similar

partitions for $\bar{\boldsymbol{z}}_i$ and $\boldsymbol{\eta}$, and obtain the following theorem

[Nguyen et al.(2021a)Nguyen, Nguyen, Nguyen, Nguyen, and Wade]:

Theorem 3.1. *The MLEs for the two-class case can be explicitly expressed as:*

$$\widehat{\boldsymbol{\mu}}_1 = \bar{\boldsymbol{y}}_1, \quad [\widehat{\boldsymbol{\mu}}]_i = [\bar{\boldsymbol{y}}_i]_2 - \boldsymbol{P}_i([\bar{\boldsymbol{y}}_i]_1 - \widehat{\boldsymbol{\mu}}_{i-1}),$$

$$\widehat{\boldsymbol{\eta}}_1 = \bar{\boldsymbol{z}}_1, \quad [\widehat{\boldsymbol{\eta}}]_i = [\bar{\boldsymbol{z}}_i]_2 - \boldsymbol{P}_i([\bar{\boldsymbol{z}}_i]_1 - \widehat{\boldsymbol{\eta}}_{i-1}),$$

and for $i = 2, .., k$,

$$\widehat{\Sigma}_1 = \frac{\mathbf{W}_1}{n_1 + m_1}, \widehat{[\Sigma_i]}_{21} = \boldsymbol{P}_i \widehat{[\Sigma_i]}_{11}, \tag{3.4}$$

$$\widehat{[\Sigma_i]}_{12} = \widehat{[\Sigma_i]}_{21}', \widehat{[\Sigma_i]}_{22} = \boldsymbol{Q}_i + \boldsymbol{P}_i \widehat{[\Sigma_i]}_{12}, \tag{3.5}$$

where

$$\boldsymbol{P}_i = [\mathbf{W}_i]_{21}[\mathbf{W}_i]_{11}^{-1} \quad \textit{and} \quad \boldsymbol{Q}_i = \frac{1}{n_i + m_i}\left([\mathbf{W}_i]_{22} - \boldsymbol{P}_i[\mathbf{W}_i]_{12}\right). \tag{3.6}$$

All details of the proof of Theorem 3.1 can be found in the Appendix A.2. An example of the MLEs computation for this theorem is provided in A.1.2. Based on Theorem 3.1, one can construct the following algorithm, namely EPEM (*Efficient Parameter Estimation for Multiple class monotone missing data*)

[Nguyen et al.(2021a)Nguyen, Nguyen, Nguyen, Nguyen, and Wade], to calculate the MLEs for the two-class monotone missing data.

Algorithm 3.3.1. (EPEM algorithm for two-class monotone missing data)

1. Input: $\boldsymbol{y}, \mathbf{z}, k; n_i, m_i, p_i,$ for $i = 1, 2, .., k$.

2. Initiate:

$$\widehat{\boldsymbol{\mu}} \leftarrow \bar{\boldsymbol{y}}_1, \widehat{\boldsymbol{\eta}} \leftarrow \bar{\boldsymbol{z}}_1, \widehat{\Sigma} \leftarrow \mathbf{W}_1/\left(n_1 + m_1\right) \tag{3.1}$$

3. For $2 \leq i \leq k$:

- Compute:

$$\boldsymbol{P}_i \leftarrow [\mathbf{W}_i]_{21}[\mathbf{W}_i]_{11}^{-1} \tag{3.2}$$

$$\boldsymbol{Q}_i \leftarrow \frac{1}{n_i + m_i}\left([\mathbf{W}_i]_{22} - \boldsymbol{P}_i[\mathbf{W}_i]_{12}\right). \tag{3.3}$$

- Update:

$$\widehat{[\Sigma_i]}_{21} \leftarrow \boldsymbol{P}_i\widehat{[\Sigma_i]}_{11}, \tag{3.4}$$

$$\widehat{\boldsymbol{\mu}} \leftarrow \begin{pmatrix} \widehat{\boldsymbol{\mu}} \\ [\bar{\boldsymbol{y}}_i]_2 - \boldsymbol{P}_i([\bar{\boldsymbol{y}}_i]_1 - \widehat{\boldsymbol{\mu}}) \end{pmatrix} \tag{3.5}$$

$$\widehat{\boldsymbol{\eta}} \leftarrow \begin{pmatrix} \widehat{\boldsymbol{\eta}} \\ [\bar{\boldsymbol{z}}_i]_2 - \boldsymbol{P}_i([\bar{\boldsymbol{y}}_i]_1 - \widehat{\boldsymbol{\eta}}) \end{pmatrix}, \tag{3.6}$$

$$\widehat{\boldsymbol{\Sigma}} \leftarrow \begin{pmatrix} \widehat{\Sigma} & \widehat{[\Sigma_i]}_{21}' \\ \widehat{[\Sigma_i]}_{21} & \boldsymbol{Q}_i + \boldsymbol{P}_i\widehat{[\Sigma_i]}_{21}' \end{pmatrix}. \tag{3.7}$$

4. Output: $\widehat{\boldsymbol{\mu}}, \widehat{\boldsymbol{\eta}}, \widehat{\boldsymbol{\Sigma}}$.

3.3.2 MLEs for multiple-class monotone missing data. Assume that there is a dataset with G categories (classes), where each sample from class $g^{th}(1 \leq g \leq G)$ follows a $p-$ dimensional multivariate normal distribution with mean $\boldsymbol{\mu}^{(g)}$ and covariance Σ, denoted as $N_p(\boldsymbol{\mu}^{(g)}, \Sigma)$. In addition, let the samples from class g^{th} have the following monotone pattern:

$$\boldsymbol{x}^{(g)} = \begin{pmatrix} \boldsymbol{x}_{11}^{(g)} & \dots & \boldsymbol{x}_{1n_k^g}^{(g)} & \dots & \boldsymbol{x}_{1n_3^g}^{(g)} & \dots & \boldsymbol{x}_{1n_2^g}^{(g)} & \dots & \boldsymbol{x}_{1n_1^g}^{(g)} \\ \boldsymbol{x}_{21}^{(g)} & \dots & \boldsymbol{x}_{2n_k^g}^{(g)} & \dots & \boldsymbol{x}_{2n_3^g}^{(g)} & \dots & \boldsymbol{x}_{2n_2^g}^{(g)} & \dots & * \\ \boldsymbol{x}_{31}^{(g)} & \dots & \boldsymbol{x}_{3n_k^g}^{(g)} & \dots & \boldsymbol{x}_{3n_2^g}^{(g)} & \dots & * & \dots & * \\ \vdots & & \vdots & \ddots & \vdots & \dots & \vdots & \dots & \vdots \\ \boldsymbol{x}_{k1}^{(g)} & \dots & \boldsymbol{x}_{kn_k^g}^{(g)} & \dots & * & \dots & * & \dots & * \end{pmatrix},$$

where there are n_1^g observations available on the first p_1 variables, n_2^g observations available on the first $p_1 + p_2$ variables, and so on. For $1 \leq g \leq G$, one can partition the data in

group g^{th} into:

$$\begin{aligned}
\boldsymbol{x}_1^{(g)} &= \begin{pmatrix} \boldsymbol{x}_{11}^{(g)} & \dots & \boldsymbol{x}_{1n_k^g}^{(g)} & \dots & \boldsymbol{x}_{1n_2^g}^{(g)} & \dots & \boldsymbol{x}_{1n_1^g}^{(g)} \end{pmatrix}, \\
\boldsymbol{x}_2^{(g)} &= \begin{pmatrix} \boldsymbol{x}_{11}^{(g)} & \dots & \boldsymbol{x}_{1n_k^g}^{(g)} & \dots & \boldsymbol{x}_{1n_2^g}^{(g)} \\ \boldsymbol{x}_{21}^{(g)} & \dots & \boldsymbol{x}_{2n_k^g}^{(g)} & \dots & \boldsymbol{x}_{2n_2^g}^{(g)} \end{pmatrix}, \\
&\vdots \\
\boldsymbol{x}_k^{(g)} &= \begin{pmatrix} \boldsymbol{x}_{11}^{(g)} & \dots & \boldsymbol{x}_{1n_k^g}^{(g)} \\ \boldsymbol{x}_{21}^{(g)} & \dots & \boldsymbol{x}_{2n_k^g}^{(g)} \\ \vdots & \ddots & \vdots \\ \boldsymbol{x}_{k1}^{(g)} & \dots & \boldsymbol{x}_{kn_k^g}^{(g)} \end{pmatrix},
\end{aligned}$$

where $\boldsymbol{x}_1^{(g)}$ has the size $p_1 \times n_1^g$, $\boldsymbol{x}_2^{(g)}$ is of size $(p_1+p_2) \times n_2^g$, $\dots$, $\boldsymbol{x}_k^{(g)}$ has the size $(\sum_{j=1}^{k} p_j) \times n_k^g$. Let $\bar{\boldsymbol{x}}_i^{(g)}, \mathbf{S}_i^{(g)}$ be the mean and covariance matrix of $\boldsymbol{x}_i^{(g)}$, respectively. For $i = 1, \dots, k$, we define

$$\boldsymbol{W}_i = \sum_{g=1}^{G} (n_i^g - 1)\mathbf{S}_i^{(g)}, \boldsymbol{\mu} = (\boldsymbol{\mu}^{(1)}, \dots, \boldsymbol{\mu}^{(G)}), \quad \bar{\boldsymbol{x}}_i = (\bar{\boldsymbol{x}}_i^{(1)}, \dots, \bar{\boldsymbol{x}}_i^{(G)}) \tag{3.8}$$

as the sum of squares and cross products matrix, the matrix of all classes' means, and the matrix of all classes' sample means. Now, we regard Σ_i as the $(\sum_{j=1}^{i} p_j)^{th}$ order leading principal submatrix of Σ, for $i = 1, \dots, k$, and do the following partitions:

$$\bar{\boldsymbol{x}}_i = \begin{pmatrix} [\bar{\boldsymbol{x}}_i]_1 \\ [\bar{\boldsymbol{x}}_i]_2 \end{pmatrix}, \quad \boldsymbol{\mu} = \begin{pmatrix} [\boldsymbol{\mu}]_1 \\ [\boldsymbol{\mu}]_2 \\ \vdots \\ [\boldsymbol{\mu}]_k \end{pmatrix}, \tag{3.9}$$

We denote

$$\boldsymbol{\Sigma_i} = \begin{pmatrix} [\boldsymbol{\Sigma_i}]_{\mathbf{11}} & [\boldsymbol{\Sigma_i}]_{\mathbf{12}} \\ [\boldsymbol{\Sigma_i}]_{\mathbf{21}} & [\boldsymbol{\Sigma_i}]_{\mathbf{22}} \end{pmatrix}, \quad \boldsymbol{W_i} = \begin{pmatrix} [\boldsymbol{W_i}]_{\mathbf{11}} & [\boldsymbol{W_i}]_{\mathbf{12}} \\ [\boldsymbol{W_i}]_{\mathbf{21}} & [\boldsymbol{W_i}]_{\mathbf{22}} \end{pmatrix}, \quad \boldsymbol{(i = 2, \dots, k)}, \tag{3.10}$$

where $[\bar{\boldsymbol{x}}_i]_1$ contains the first $\sum_{j=1}^{i-1} p_j$ rows of $\bar{\boldsymbol{x}}_i$, and $[\bar{\boldsymbol{x}}_i]_2$ has all p_i remaining rows; $[\boldsymbol{\mu}^{(g)}]_j$ is a block of order $p_j \times \left(\sum_{j=1}^{k} p_j\right)$, $j = 1, \dots, k$; $[\Sigma_i]_{11}$ and $[\boldsymbol{W}_i]_{11}$ are of order

$\left(\sum_{j=1}^{i-1} p_j\right) \times \left(\sum_{j=1}^{i-1} p_j\right)$, $[\Sigma_i]_{12}$ and $[\boldsymbol{W}_i]_{12}$ are of order $\left(\sum_{j=1}^{i-1} p_j\right) \times p_i$, $[\Sigma_i]_{22}$ and $[\boldsymbol{W}_i]_{22}$ are of order $p_i \times p_i$.

Now, if

$$\boldsymbol{\mu}_i = \begin{pmatrix} [\boldsymbol{\mu}]_1 \\ \vdots \\ [\boldsymbol{\mu}]_{i.} \end{pmatrix}, i = 1, \ldots, k, \tag{3.11}$$

then we have the following theorem

[Nguyen et al.(2021a)Nguyen, Nguyen, Nguyen, Nguyen, and Wade]:

Theorem 3.2. *The MLEs are calculated by the following recurrence equations:*

$$\widehat{\boldsymbol{\mu}}_1 = \bar{\boldsymbol{x}}_1, \tag{3.12}$$

$$[\widehat{\boldsymbol{\mu}}]_i = [\bar{\boldsymbol{x}}_i]_2 - \boldsymbol{P}_i([\bar{\boldsymbol{x}}_i]_1 - \widehat{\boldsymbol{\mu}}_{i-1}), \tag{3.13}$$

$$\widehat{\Sigma}_1 = \frac{\boldsymbol{W}_1}{\sum_{g=1}^{G} n_1^g}, \tag{3.14}$$

$$\widehat{[\Sigma_i]}_{21} = \boldsymbol{P}_i \widehat{[\Sigma_i]}_{11}, \tag{3.15}$$

and

$$\widehat{[\Sigma_i]}_{12} = (\widehat{[\Sigma_i]}_{21})', \tag{3.16}$$

$$\widehat{[\Sigma_i]}_{22} = \boldsymbol{Q}_i + \boldsymbol{P}_i \widehat{[\Sigma_i]}_{12}, \tag{3.17}$$

for $i = 2, \ldots, k$, where

$$\boldsymbol{P}_i = [\boldsymbol{W}_i]_{21}[\boldsymbol{W}_i]_{11}^{-1} \quad \textit{and} \quad \boldsymbol{Q}_i = \frac{1}{\sum_{g=1}^{G} n_i^g} \left([\boldsymbol{W}_i]_{22} - \boldsymbol{P}_i[\boldsymbol{W}_i]_{12}\right). \tag{3.18}$$

We provide the proof of Theorem 3.2 in the Appendix A.3. From this theorem, one can design the following algorithm to compute MLEs for multiple-class monotone missing data [Nguyen et al.(2021a)Nguyen, Nguyen, Nguyen, Nguyen, and Wade]:

Algorithm 3.3.2. (EPEM algorithm for multiple-class monotone missing data)

1. Input: $\boldsymbol{x}^{(g)}, n_i^g, p_i$, for $g = 1, 2, .., G;\ i = 1, 2, \ldots, k$.

2. Initiate:

$$\widehat{\boldsymbol{\mu}} \leftarrow \bar{\boldsymbol{x}}_1, \widehat{\Sigma} \leftarrow \boldsymbol{W}_1 / \left(\sum_{g=1}^{G} n_1^g \right). \tag{3.1}$$

3. For $2 \leq i \leq k$:

   - Compute:

$$\boldsymbol{P}_i = [\boldsymbol{W}_i]_{21} [\boldsymbol{W}_i]_{11}^{-1}, \tag{3.2}$$

$$\boldsymbol{Q}_i \leftarrow \frac{1}{\sum_{g=1}^{G} n_i^g} \left([\boldsymbol{W}_i]_{22} - \boldsymbol{P}_i [\boldsymbol{W}_i]_{12} \right) \tag{3.3}$$

   - Update:

$$\hat{\boldsymbol{\mu}} \leftarrow \begin{pmatrix} \hat{\boldsymbol{\mu}} \\ [\bar{\boldsymbol{x}}_i]_2 - P_i([\bar{\boldsymbol{x}}_i]_1 - \hat{\boldsymbol{\mu}}) \end{pmatrix}, \tag{3.4}$$

$$\widehat{\Sigma} \leftarrow \begin{pmatrix} \widehat{\Sigma} & \widehat{[\Sigma_i]}_{21}' \\ \widehat{[\Sigma_i]}_{21} & \boldsymbol{Q}_i + \boldsymbol{P}_i \widehat{[\Sigma_i]}_{21}' \end{pmatrix}. \tag{3.5}$$

4. Output: $\widehat{\boldsymbol{\mu}}, \widehat{\boldsymbol{\Sigma}}$.

Importantly, we can ensure that

[Nguyen et al.(2021a)Nguyen, Nguyen, Nguyen, Nguyen, and Wade]:

Theorem 3.3. *The resulting estimates of the EPEM algorithm are asymptotic, i.e.,*

$$\hat{\boldsymbol{\mu}} \xrightarrow{p} \boldsymbol{\mu}, \textit{and} \quad \hat{\boldsymbol{\Sigma}} \xrightarrow{p} \boldsymbol{\Sigma}, \tag{3.6}$$

where $\xrightarrow{p}$ *denotes convergence in probability.*

One can see our proof for Theorem 3.3 in the Appendix A.4. An example on the MLEs computation for Theorem 3.2 is available in Appendix A.1.3.

3.4 Applications In Linear Discriminant Analysis With Monotone Missing Data

Suppose that there are given data of G classes, and data from each class follow a multivariate normal distribution with mean $\boldsymbol{\mu}_k$ and covariance $\boldsymbol{\Sigma}$, i.e., the data from all classes have the same covariance matrix. In LDA, the classification rule that minimizes the total probability of misclassification is to assign $\boldsymbol{x}$ to π_h if

$$d_h(\boldsymbol{x}) = \max\{d_1(\boldsymbol{x}), d_2(\boldsymbol{x}), \ldots, d_G(\boldsymbol{x})\}, \tag{3.1}$$

where

$$d_i(\boldsymbol{x}) = \boldsymbol{\mu}_i'\Sigma^{-1}\boldsymbol{x} - \frac{1}{2}\boldsymbol{\mu}_i'\Sigma^{-1}\boldsymbol{\mu}_i + \ln r_i, \quad i = 1, 2, \ldots, G. \tag{3.2}$$

Here, r_i is the proportion of the number of samples that belong to the i^{th} class. In our approach, the parameters $\boldsymbol{\mu}_i, \boldsymbol{\Sigma}$ are replaced by the corresponding maximum likelihood estimates. Therefore, the classification rule is to assign $\boldsymbol{x}$ to class π_h if

$$\hat{d}_h(\boldsymbol{x}) = \max\{\hat{d}_1(\boldsymbol{x}), \hat{d}_2(\boldsymbol{x}), \ldots, \hat{d}_G(\boldsymbol{x})\}, \tag{3.3}$$

where for $i = 1, 2, \ldots, G$,

$$\hat{d}_i(\boldsymbol{x}) = \hat{\boldsymbol{\mu}}_i'\widehat{\Sigma}^{-1}\boldsymbol{x} - \frac{1}{2}\hat{\boldsymbol{\mu}}_i'\widehat{\Sigma}^{-1}\hat{\boldsymbol{\mu}}_i + \ln \hat{r}_i. \tag{3.4}$$

We have the following important results

[Nguyen et al.(2021a)Nguyen, Nguyen, Nguyen, Nguyen, and Wade]:

Theorem 3.4. ***(Invariant under Non-degenerate Linear Transformations)*** *The classification rule (3.3) is invariant to non-degenerate linear transformations, i.e., if we transform the data by an invertible linear transformation A then the classification rule is the same as before transforming, i.e., to assign $\boldsymbol{x}$ to π_h if*

$$\hat{d}_h(\boldsymbol{x}) = \max\{\hat{d}_1(\boldsymbol{x}), \hat{d}_2(\boldsymbol{x}), \ldots, \hat{d}_G(\boldsymbol{x})\}, \tag{3.5}$$

where $\hat{d}_i(\boldsymbol{x})$ *is defined as in Eq. (3.4) for* $i = 1, 2, \ldots, G$.

Theorem 3.5. $\hat{d}_i(\boldsymbol{x})$ *converges in probability to* $d_i(\boldsymbol{x})$ *for all* $i = 1, 2, \ldots, G$.

All details of the proof for both Theorems 3.4 and 3.5 can be found in Appendix A.5 and Appendix A.6.

3.5 Experiments

In this section, we describe all datasets, the implementation of the proposed techniques, and experimental results in detail.

3.5.1 Settings. To illustrate the efficiency of our algorithm, we compare the results of the proposed method with three traditional, yet popular, and powerful methods: K-nearest neighbor (KNN) imputation
[García-Laencina et al.(2010)García-Laencina, Sancho-Gómez, and Figueiras-Vidal], and Multiple Imputation by Chained Equation (MICE)
[Buuren and Groothuis-Oudshoorn(2010)]. In addition, we compare EPEM with several state-of-art techniques such as SOFT-IMPUTE
[Mazumder et al.(2010)Mazumder, Hastie, and Tibshirani] and imputation by Nuclear Norm Minimization [Candès and Recht(2009)]. We implement these methods via packages *fancyimpute*[1] and *impyute*[2].

We perform the experiments on the following data sets from the Machine Learning Database Repository at the University of California, Irvine [Dua and Graff(2017)]: Iris, Wine, Seeds, and Ionosphere. Besides, we do experiments on two large datasets: MNIST[LeCun(1998)], a large dataset of handwritten digits, consisting of 60000 training

[1] `https://pypi.org/project/fancyimpute/`
[2] `https://pypi.org/project/impyute/`

Table 3.1. The description of data sets that are used in our experiments

Dataset	# Classes	# Features	# Samples
Iris	3	4	150
Wine	3	13	178
Seeds	3	7	210
Ionosphere	2	34 (32*)	351
MNIST	10	784(649*)	70000
Fashion MNIST	10	784	70000

images and 10000 testing images of size 28×28, and Fashion MNIST [Xiao et al.(2017)Xiao, Rasul, and Vollgraf], a data set of clothing images, also having 60000 training images and 10000 testing images of size 28×28. Each of these two datasets has ten labels. Table 3.1 shows a summary of all data sets used. In each dataset, we normalize by removing the mean and scaling to unit variance all features. For the Ionosphere data set, we delete the first column where the values are the same within one of the classes and the second columns where all entries are 0. Finally, for the MNIST data set, we delete 135 columns that have no more than ten nonzero values (Note that we have ten classes. If one class has a column of all zeros, then the sample covariance matrix for that class is a degenerate matrix.)

For the parameter estimation experiments, we separate each label by creating a list of arrays, where each array consists of the corresponding data from one category. Then, we make monotone patterns by specifying k, p, n that makes up $20\%, 30\%, 40\%, 50\%$ of the data. For two datasets MNIST and Fashion MNIST, we merge the training and testing sets and simulate missing data similarly.

In the experiments for the application in Linear Discriminant Analysis for MNIST and Fashion MNIST, the training and testing data are already split. Therefore, we shuffle the training data, separate each class, and make a monotone pattern on each category by

specifying k, p, n that makes up $20\%, 30\%, 40\%, 50\%$ of the training data. For other datasets, each dataset is shuffled and split into folds for cross-validation [Browne(2000)]. During cross-validation, we separate each class and create a monotone pattern on each category by specifying k, p, n that makes up $20\%, 30\%, 40\%, 50\%$ of the training data on average across folds (This is to avoid specifying a different set of k, p, n for each fold. The reason is if k, p, n consists of τ integers, then specifying a different set of k, p, n for each fold is equivalent to specifying 5τ integers for the experiment, which is a restriction on randomness).

When using iterative methods for small-size and middle-size datasets, we use the default options in the packages *fancyimpute* and *impyute*. Specifically, the maximum number of iterations is 100 for SOFT-IMPUTE, MICE, and Nuclear Norm Minimization, and $k = 3$ for KNN imputation. For big datasets (MNIST & Fashion MNIST), the maximum number of iterations for MICE and Nuclear Norm Minimization re reduced to 10 instead due to expensive memory and time requirements. The experiments on small datasets are ran directly on *Google Colaboratory*[3]. Meanwhile, the experiments on large datasets are run on a Linux machine with 128 GB RAM, 8 cores, Xeon 6244 processor 3.66 Hz. We terminate an experiment if no result is produced after twenty-four hours of running or when having the out-of-memory issue, and denote this as NA in the result tables (Tables 3.2 and 3.3).

For evaluation, we aim to have a metric that evaluates the performance on the means and the covariance. Yet, the means may have a very small number of parameters compared to the covariance matrix. Hence, one can expect the estimation of the covariance to result in more errors. Consequently, it is not reasonable to use Mean Square

[3] `https://colab.research.google.com/`

Error or Root Mean Square Error, which considers the roles of entries in the means and covariance matrix equally. Therefore, we use the following metric, which takes into account the number of entries in the means and the covariance matrix instead [Nguyen et al.(2021a)Nguyen, Nguyen, Nguyen, Nguyen, and Wade],

$$r = \frac{||\boldsymbol{\mu} - \hat{\boldsymbol{\mu}}||_F}{n_{\boldsymbol{\mu}}} + \frac{||\boldsymbol{\Sigma} - \hat{\boldsymbol{\Sigma}}||_F}{n_{\boldsymbol{\Sigma}}}, \tag{3.1}$$

where $||.||$ denotes the Frobenius norm; $\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\Sigma}}$ are estimated values derived from EPEM for $\boldsymbol{\mu}, \boldsymbol{\Sigma}$ respectively; $n_{\boldsymbol{\mu}}$, $n_{\boldsymbol{\Sigma}}$ are the corresponding number of entries in $\boldsymbol{\mu}, \boldsymbol{\Sigma}$ and the ground truth $\boldsymbol{\mu}, \boldsymbol{\Sigma}$ are calculated from the full data without missing values. For interpretation, note that this is the sum of the average difference of each entry in the mean/common covariance matrix.

3.5.2 Results and discussion. From Table 3.2, one can see that EPEM consistently outperforms other approaches by a significant margin. For example, with 40% of missing data in the Parkinson data set, EPEM gives the error rate of 0.022, which is less than 1/2 the error rate of the other approaches (including powerful methods such as, e.g., SOFT-IMPUTE, MICE, KNN, and NNMT). Note that the Royston test for testing multivariate normality gives $p-$value < 0.01 for Seeds, Iris, Wine, and Ionosphere, indicating these datasets do not follow a multivariate normal distribution. Therefore, even though multivariate normality is assumed in order to prove all the theorems in this paper, the results are robust relative to normality.

When our estimation scheme is utilized for LDA, one can see from Table 3.3 that the proposed technique also outperforms other methods over most of the data sets in the experiment again. For the Iris dataset with a 20% missing rate, our approach surpasses MICE imputation by 3.9% error rate reduction, SOFT-IMPUTE by 9.3%, and KNN by

Table 3.2. Parameter estimation errors with different missing rates. Bold implies the best results.

Dataset	Missing Rate(%)	EPEM	KNN Imputation	MICE	SOFT-IMPUTE	Nuclear Norm
Seeds	20%	**0.016**	0.195	0.197	0.196	0.196
	30%	**0.017**	0.194	0.196	0.195	0.195
	40%	**0.022**	0.188	0.194	0.191	0.192
	50%	**0.022**	0.185	0.192	0.187	0.188
Iris	20%	**0.026**	0.282	0.288	0.289	0.286
	30%	**0.034**	0.275	0.289	0.280	0.281
	40%	**0.053**	0.261	0.278	0.266	0.267
	50%	**0.053**	0.266	0.282	0.269	0.270
Wine	20%	**0.014**	0.110	0.109	0.110	0.110
	30%	**0.016**	0.112	0.109	0.110	0.110
	40%	**0.019**	0.113	0.109	0.112	0.112
	50%	**0.022**	0.112	0.110	0.111	0.111
Ionosphere	20%	**0.009**	0.033	0.033	0.033	0.033
	30%	**0.012**	0.033	0.033	0.033	0.033
	40%	**0.015**	0.033	0.033	0.033	0.033
	50%	**0.017**	0.034	0.033	0.033	0.033
MNIST	20%	**1.4e-4**	NA	NA	4.4e-3	NA
	30%	**1.6e-4**	NA	NA	4.4e-3	NA
	40%	**2.0e-4**	NA	NA	4.4e-3	NA
	50%	**2.5e-4**	NA	NA	4.4e-3	NA
Fashion MNIST	20%	**1.0e-4**	NA	NA	6.4e-3	NA
	30%	**1.2e-4**	NA	NA	6.4e-3	NA
	40%	**1.4e-4**	NA	NA	6.4e-3	NA
	50%	**1.6e-4**	NA	NA	6.4e-3	NA

Table 3.3. Cross-validation errors on datasets with different missing rates in LDA application. Bold implies the best results

Dataset	Missing Rate(%)	EPEM	KNN imputation	MICE	SOFT-IMPUTE	Nuclear Norm
Seeds	20%	**0.034**	0.051	0.042	0.052	0.052
	30%	**0.038**	0.051	0.062	0.062	0.062
	40%	**0.038**	0.071	0.078	0.074	0.068
	50%	**0.048**	0.085	0.077	0.086	0.090
Iris	20%	**0.024**	0.125	0.063	0.117	0.117
	30%	**0.032**	0.155	0.063	0.126	0.135
	40%	**0.037**	0.153	0.075	0.133	0.133
	50%	**0.046**	0.201	0.135	0.143	0.146
Wine	20%	**0.011**	0.025	0.018	0.013	0.013
	30%	**0.011**	0.041	0.018	0.016	0.016
	40%	**0.011**	0.051	0.017	0.029	0.029
	50%	**0.011**	0.056	0.017	0.033	0.034
Ionosphere	20%	**0.155**	0.159	0.161	0.159	0.159
	30%	0.139	0.151	**0.134**	0.146	0.148
	40%	**0.151**	0.153	0.164	0.170	0.165
	50%	0.164	0.166	0.161	**0.160**	0.168
MNIST	20%	**0.128**	NA	0.135	0.140	NA
	30%	**0.128**	NA	0.139	0.151	NA
	40%	**0.130**	NA	0.143	0.151	NA
	50%	**0.130**	NA	0.148	0.155	NA
Fashion MNIST	20%	**0.184**	NA	NA	0.189	NA
	30%	**0.184**	NA	NA	0.195	NA
	40%	**0.184**	NA	NA	0.202	NA
	50%	**0.185**	NA	NA	0.198	NA

10.1%. Another example is for the Parkinson dataset with a 30% missing rate, and our method exceeds KNN imputation and Nuclear norm minimization by reducing 3.5%, 4.9% the error rates, respectively.

For the parameter estimation part on MNIST and Fashion MNIST, EPEM has an error almost equal to zero on these two datasets. Recalling that the actual means and the covariance matrix are unknown, our evaluation is based on the corresponding estimates on the full data (i.e., the dataset before the missing data simulation). Therefore, this implies that using EPEM can give approximations to the means and the covariance matrix with almost excellent results as using the full data. This result is not surprising since Multivariate Central Limit Theorem [Van der Vaart(2000)] suggests that the maximum likelihood parameter approximation gets better as the sample size increases, and the sample sizes of both datasets are large.

It is worth noting that for these big datasets, only results for EPEM and SOFT-IMPUTE are available. Even though we reduced the maximum number of iterations for MICE and Nuclear Norm Minimization to 10, they could not produce results within twenty-four hours or require more than 128 GB to process. Among the available methods, EPEM is the fastest one with minimum parameter estimates running on MNIST at (0.53s) and SOFT-IMPUTE is the second fastest (737.32s). For other approaches, the corresponding outcomes are not available due to previously stated reasons. Therefore, we can deduce that EPEM and SOFT-IMPUTE are more suitable than MICE and Nuclear Norm Minimization for large datasets with a high rate of missing values.

Discussion: It is well-known that when the sample size is large enough, the multivariate normal distribution can be a good approximator to the proper underlying

distribution of the data. Yet, it may not be true when the sample size is small compared to the number of features. Therefore, in such cases, this method should only be used directly when the data come from a multivariate normal distribution.

It is worth noting that in the problems where the data distribution is highly skew [Mardia(1970)], the multivariate normal distribution might not be a good approximator. Therefore, other methods than EPEM should be utilized for these settings.

One possible drawback is when the dimension is too large, directly inverting a big covariance matrix is quite expensive. In this case, approximation methods should be used to calculate the inverse (see [Toutounian and Soleymani(2013)], [Li and Li(2010)] or similar materials).

3.6 Conclusion

In this chapter, we have derived the formulas and proposed an efficient algorithm for computing the parameters of a multiple class monotone missing data set. The covariance matrices of all the classes are assumed to be equal. Even though normality is considered, we have illustrated that the proposed algorithm can work well in practice and achieve better performance through different experiments than several other imputation-based approaches. While we only demonstrated two typical applications, our method is general enough to extend for other scenarios such as principal component analysis, Fisher's discriminant analysis, and hypothesis testing for multiple class monotone missing data.

Chapter 4: Parameter Estimation For Randomly Missing Data

In the previous chapter, we have discussed the MLEs for monotone missing data. This approach is a great improvement of speed and is able to reduce the estimation error significantly compared to iterative methods. Yet, it can only be applied to monotone missing data. Inspired by the idea of EPEM, in this chapter, we examine DPER (*Direct Parameter Estimation for Randomly Missing Data*) a different approach that can be applied to general randomly missing data.

4.1 Single Class DPER Algorithm And Multiple-class DPER Algorithm Without The Assumption Of Equal Covariance Matrix

This section will describe our proposed approaches to find the maximum likelihood estimators for a randomly missing dataset that only consists of a single class and for a multiple-class dataset without the assumption of equal covariance matrices.

4.1.1 Single class DPER algorithm. First, we have the following results [Nguyen et al.(2021b)Nguyen, Nguyen-Duy, Nguyen, Nguyen, and Wade],

Theorem 4.1. *Assume that we have a set of i.i.d observations from a bivariate normal distribution with mean*

$$\boldsymbol{\mu} = \begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix} \tag{4.1}$$

and covariance matrix

$$\boldsymbol{\Sigma} = \begin{pmatrix} \sigma_{11} & \sigma_{12} \\ \sigma_{21} & \sigma_{22} \end{pmatrix}. \tag{4.2}$$

Arrange the data into the following pattern

$$\boldsymbol{x} = \begin{pmatrix} x_{11} & \dots & x_{1m} & x_{1m+1} & \dots & x_{1n} & * & \dots & * \\ x_{21} & \dots & x_{2m} & * & \dots & * & x_{2n+1} & \dots & x_{2l} \end{pmatrix}. \tag{4.3}$$

Note that each column represents an observation, and $x_{ij} \in \mathbb{R}$ is an entry, i.e., each observation has two features.

Let L be the likelihood of the data and

$$s_{11} = \sum_{j=1}^{m} (x_{1j} - \hat{\mu}_1)^2, \tag{4.4}$$

$$s_{12} = \sum_{j=1}^{m} (x_{2j} - \hat{\mu}_2)(x_{1j} - \hat{\mu}_1), \tag{4.5}$$

$$s_{22} = \sum_{j=1}^{m} (x_{2j} - \hat{\mu}_2)^2. \tag{4.6}$$

Then, the resulting estimators obtained by maximizing L w.r.t $\mu_1, \sigma_{11}, \mu_2, \sigma_{22}$, and σ_{12} are:

$$\hat{\mu}_1 = \frac{1}{n} \sum_{j=1}^{n} x_{1j}, \tag{4.7}$$

$$\hat{\mu}_2 = \frac{\sum_{j=1}^{m} x_{2j} + \sum_{j=n+1}^{l} x_{2j}}{m + l - n}, \tag{4.8}$$

$$\hat{\sigma}_{11} = \frac{\sum_{j=1}^{n} (x_{1j} - \hat{\mu}_1)^2}{n}, \tag{4.9}$$

$$\hat{\sigma}_{22} = \frac{\sum_{j=1}^{m} (x_{2j} - \hat{\mu}_2)^2 + \sum_{j=n+1}^{l} (x_{2j} - \hat{\mu}_2)^2}{m + l - n}, \tag{4.10}$$

and $\hat{\sigma}_{12}$, where $\hat{\sigma}_{12}$ is the maximizer of:

$$\eta = C - \frac{1}{2} m \log \left(\sigma_{22} - \frac{\sigma_{12}^2}{\sigma_{11}} \right) - \frac{1}{2} \left(s_{22} - 2 \frac{\sigma_{12}}{\sigma_{11}} s_{12} + \frac{\sigma_{12}^2}{\sigma_{11}^2} s_{11} \right) \left(\sigma_{22} - \frac{\sigma_{12}^2}{\sigma_{11}} \right)^{-1}. \tag{4.11}$$

Thus, the estimates for $\mu_1, \mu_2, \sigma_{11}, \sigma_{22}$ are the corresponding sample means and uncorrected sample variance (where the denominator is divided by the sample size) of the first/second feature after dropping the missing entries. We have the following theorem related to the existence of $\hat{\sigma}_{12}$

[Nguyen et al.(2021b)Nguyen, Nguyen-Duy, Nguyen, Nguyen, and Wade],

Theorem 4.2. *In equation 4.11, solving*

$$\frac{d\eta}{d\sigma_{12}} = 0 \tag{4.12}$$

can be reduced to solving the following third-degree polynomial

$$s_{12}\sigma_{11}\sigma_{22} + \left(\sum_{g=1}^{G} m_g\sigma_{11}\sigma_{22} - s_{22}\sigma_{11} - s_{11}\sigma_{22}\right)\sigma_{12} + s_{12}\sigma_{12}^2 - (\sum_{g=1}^{G} m_g)\sigma_{12}^3 \tag{4.13}$$

which has at least one real root. In addition, the global maximum is a real solution to that equation, provided that

$$-\frac{s_{22}\sigma_{11} + s_{11}\sigma_{22}}{2\sqrt{\sigma_{11}\sigma_{22}}} \neq s_{12} \neq \frac{s_{22}\sigma_{11} + s_{11}\sigma_{22}}{2\sqrt{\sigma_{11}\sigma_{22}}}. \tag{4.14}$$

The proof of Theorem 4.1 and Theorem 4.2 follow directly as corollaries of Theorem 4.3 and Theorem 4.4, the proofs of which are given in B.3 and B.2, respectively. Based on the above theorem, we have the following algorithm [Nguyen et al.(2021b)Nguyen, Nguyen-Duy, Nguyen, Nguyen, and Wade],

Algorithm 4.1.1. (DPER algorithm for single class randomly missing data)

Input: a dataset of p features.

Ouput: $\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\Sigma}} = (\sigma_{ij})_{i,j=1}^{p}$

1. Estimate $\boldsymbol{\mu}$: $\hat{\mu}_i$ is the sample mean of all the available entries in the i^{th} feature.

2. Estimate the diagonal elements of $\boldsymbol{\Sigma}$: $\hat{\sigma}_{ii}$ is the uncorrected sample variance of all the available entries in the i^{th} feature.

3. For $1 \leq i \leq p$:

   For $1 \leq j \leq i$:

Compute $\hat{\sigma}_{ij}$ based on equation 4.12. If there are two solution that maximize the function, choose the one that is closest to the estimate based on case deletion method.

Recall that in the *case deletion method*, the sample that has one or more missing values are deleted. In this setting, during the estimation of $\hat{\sigma}_{ij}$, we delete any $(i,j)^{th}$ pair has at least one missing entry.

4.1.2 Multiple-class DPER algorithm without the assumption of equal covariance matrices.

When the covariance matrices of all classes are not assumed to be equal, the mean and covariance matrix for each class can be estimated separately by applying the *DPER algorithm for single class randomly missing data* to the data from each class.

4.2 Multiple-class DPER Algorithm Under The Assumption Of Equal Covariance Matrices

In the previous section, we have described our proposed approaches to find the maximum likelihood estimators for a randomly missing dataset that only consists of a single class and a multiple-class dataset without the assumption of equal covariance matrices. Yet, in many cases, for a multiple-class dataset, it may be more desirable to assume that the covariance matrices are equal. Those cases may happen when the number of samples per class is small or in linear discriminant analysis. Therefore, in this section, we tackle the multiple-class problem under the assumption of equal covariance matrices. First, we have the following result

[Nguyen et al.(2021b)Nguyen, Nguyen-Duy, Nguyen, Nguyen, and Wade],

Theorem 4.3. *Assume that there is a dataset with G classes, where each sample from class g^{th} $(1 \leq g \leq G)$ follows a bivariate normal distribution with mean*

$$\boldsymbol{\mu}^{(g)} = \begin{pmatrix} \mu_1^{(g)} \\ \mu_2^{(g)} \end{pmatrix} \tag{4.1}$$

and covariance matrix

$$\boldsymbol{\Sigma} = \begin{pmatrix} \sigma_{11} & \sigma_{12} \\ \sigma_{21} & \sigma_{22} \end{pmatrix} \tag{4.2}$$

Arrange the data from each class into the following pattern

$$\boldsymbol{x}^{(g)} = \begin{pmatrix} x_{11}^{(g)} & \dots & x_{1m_g}^{(g)} & x_{1m_g+1}^{(g)} & \dots & x_{1n_g}^{(g)} & * & \dots & * \\ x_{21}^{(g)} & \dots & x_{2m_g}^{(g)} & * & \dots & * & x_{2n_g+1}^{(g)} & \dots & x_{2l_g}^{(g)} \end{pmatrix}. \tag{4.3}$$

Note that each column represents an observation and $x_{ij} \in \mathbb{R}$, i.e., each observation has only two features.

Let L be the likelihood of the data and

$$A = \sum_{g=1}^{G} m_g, \tag{4.4}$$

$$s_{11} = \sum_{g=1}^{G} \sum_{j=1}^{m_g} (x_{1j}^{(g)} - \hat{\mu}_1^{(g)})^2, \tag{4.5}$$

$$s_{12} = \sum_{g=1}^{G} \sum_{j=1}^{m_g} (x_{2j}^{(g)} - \hat{\mu}_2^{(g)})(x_{1j}^{(g)} - \hat{\mu}_1^{(g)}), \tag{4.6}$$

$$s_{22} = \sum_{g=1}^{G} \sum_{j=1}^{m_g} (x_{2j}^{(g)} - \hat{\mu}_2^{(g)})^2. \tag{4.7}$$

Then, the resulting estimators obtained by maximizing L w.r.t $\mu_1^{(g)}, \sigma_{11}, \mu_2^{(g)}$, σ_{22}, σ_{12}

are:

$$\hat{\mu}_1^{(g)} = \frac{1}{n_g}\sum_{j=1}^{n_g} x_{1j}^{(g)},$$
$$\hat{\mu}_2^{(g)} = \frac{\sum_{j=1}^{m_g} x_{2j}^{(g)} + \sum_{j=n_g+1}^{l_g} x_{2j}^{(g)}}{m_g + l_g - n_g},$$
$$\hat{\sigma}_{11} = \frac{\sum_{g=1}^{G}\sum_{j=1}^{n_g}(x_{1j}^{(g)} - \hat{\mu}_1^{(g)})^2}{\sum_{g=1}^{G} n_g},$$
$$\hat{\sigma}_{22} = \frac{\sum_{g=1}^{G}[\sum_{j=1}^{m_g}(x_{2j}^{(g)} - \hat{\mu}_2^{(g)})^2 + \sum_{j=n_g+1}^{l_g}(x_{2j}^{(g)} - \hat{\mu}_2^{(g)})^2]}{\sum_{g=1}^{G}(m_g + l_g - n_g)},$$

$\hat{\sigma}_{12}$*, where* $\hat{\sigma}_{12}$ *is the maximizer of:*

$$\eta(\sigma_{12}) = C - \frac{A}{2}\log\left(\sigma_{22} - \frac{\sigma_{12}^2}{\sigma_{11}}\right) - \frac{s_{22} - 2\frac{\sigma_{12}}{\sigma_{11}}s_{12} + \frac{\sigma_{12}^2}{\sigma_{11}^2}s_{11}}{2\left(\sigma_{22} - \frac{\sigma_{12}^2}{\sigma_{11}}\right)}. \tag{4.8}$$

This means the estimate for $\mu_1^{(g)}, \mu_2^{(g)}, \sigma_{11}, \sigma_{22}$ are the corresponding sample means and uncorrected sample variance of the first/second feature after dropping the missing entries. We also have the following theorem regarding $\hat{\sigma}_{12}$

[Nguyen et al.(2021b)Nguyen, Nguyen-Duy, Nguyen, Nguyen, and Wade],

Theorem 4.4. *For equation 4.8, solving*

$$\frac{d\eta}{d\sigma_{12}} = 0 \tag{4.9}$$

can be reduced to finding the roots of the following third-degree polynomial

$$s_{12}\sigma_{11}\sigma_{22} + (\sigma_{11}\sigma_{22}A - s_{22}\sigma_{11} - s_{11}\sigma_{22})\sigma_{12} + s_{12}\sigma_{12}^2 - A\sigma_{12}^3, \tag{4.10}$$

which has at least one real root. Moreover, the global maxima is a real solution to that equation, provided that

$$-\frac{s_{22}\sigma_{11} + s_{11}\sigma_{22}}{2\sqrt{\sigma_{11}\sigma_{22}}} \neq s_{12} \neq \frac{s_{22}\sigma_{11} + s_{11}\sigma_{22}}{2\sqrt{\sigma_{11}\sigma_{22}}}. \tag{4.11}$$

The proof of this theorem is given in B.3. In addition, the proof of Theorem 4.3 is given in B.2. It makes use of the following lemma

[Nguyen et al.(2021b)Nguyen, Nguyen-Duy, Nguyen, Nguyen, and Wade], whose proof is available in B.1,

Lemma 4.5. *Suppose that we have a data set that comes from G classes, where the observations from the g^{th} class follows multivariate normal distribution with mean $\tau^{(g)}$ and covariance matrix Δ. Let the observations of the g^{th} class be $\boldsymbol{u}_1^{(g)}, \boldsymbol{u}_2^{(g)}, ..., \boldsymbol{u}_{n_g}^{(g)}$. Then, the MLEs of $\tau^{(g)}$ is the sample class mean $\bar{\mathbf{u}}^{(g)}$, and the MLE of Δ is*

$$\hat{\Delta} = \frac{1}{\sum_{g=1}^{G} n_g} \sum_{g=1}^{G} \sum_{j=1}^{n_g} (\boldsymbol{u}_j^{(g)} - \bar{\boldsymbol{u}}^{(g)})(\boldsymbol{u}_j^{(g)} - \bar{\boldsymbol{u}}^{(g)})'. \tag{4.1}$$

Based on Theorem 4.3, we have the following algorithm

[Nguyen et al.(2021b)Nguyen, Nguyen-Duy, Nguyen, Nguyen, and Wade],

Algorithm 4.2.1. (DPER algorithm for multiple -class randomly missing data)

Input: a dataset of G classes, p features.

Ouput: $\hat{\boldsymbol{\mu}}^{(g)}, \hat{\Sigma} = (\sigma_{ij})_{i,j=1}^{p}$

1. Estimate $\boldsymbol{\mu}^{(g)}$: $\hat{\mu}_i^{(g)}$ is the mean of all the available entries in the i^{th} feature of the g^{th} class.

2. Estimate the diagonal elements of $\boldsymbol{\Sigma}$: $\hat{\sigma}_{ii}$ is the uncorrected sample variance of all the available entries in the i^{th} feature.

3. For $1 \leq i \leq p$:

   For $1 \leq j \leq i$:

Compute $\hat{\sigma}_{ij} = \hat{\sigma}_{ji}$ based on equation 4.9. If there are two solution that maximize the function, choose the one that is closest to the estimate based on case deletion method.

Remark 4.2.1. We have the following remarks about our estimates [Nguyen et al.(2021b)Nguyen, Nguyen-Duy, Nguyen, Nguyen, and Wade].

- The computation is done on each pair of features. Therefore, we only need the assumption of bivariate normality on each pair of features instead of multivariate normality on each observation as in EPEM [Nguyen et al.(2021a)Nguyen, Nguyen, Nguyen, Nguyen, and Wade].
- The computation is done separately for each σ_{ij}. Therefore, our method is also applicable to the problems where the number of features is much higher than the sample size.
- Since maximum likelihood estimates are asymptotic (see [Casella and Berger(2002)]), we can ensure that the resulting estimates of the our algorithm are asymptotic, i.e.,

$$\hat{\boldsymbol{\mu}} \xrightarrow{p} \boldsymbol{\mu}, \text{and} \quad \hat{\boldsymbol{\Sigma}} \xrightarrow{p} \boldsymbol{\Sigma}, \tag{4.2}$$

where $\xrightarrow{p}$ denotes convergence in probability for each element of the matrix $\hat{\boldsymbol{\mu}}$ or $\hat{\boldsymbol{\Sigma}}$.

4.3 Experiments

In this section, we describe all datasets, the implementation of the proposed techniques, and experimental results in detail.

4.3.1 Settings. To illustrate the efficiency of our algorithm, we compare the results of the proposed method the following methods: SOFT-IMPUTE [Mazumder et al.(2010)Mazumder, Hastie, and Tibshirani], by Nuclear Norm Minimization [Candès and Recht(2009)], MissForest [Stekhoven and Bühlmann(2012)] and Multiple Imputation by Chained Equation (MICE) [Buuren and Groothuis-Oudshoorn(2010)]. We implement these methods via packages *fancyimpute*[1], *impyute*[2], *missingpy*[3], and *scikit-learn*[4].

Similar to the previous chapter, we perform experiments on the following data sets from the Machine Learning Database Repository at the University of California, Irvine [Dua and Graff(2017)]: Iris, Wine, Seeds, and Ionosphere. Besides, we do experiments on two large datasets: Fashion MNIST [Xiao et al.(2017)Xiao, Rasul, and Vollgraf], a large data set of clothing images, consisting of 60000 training images and 10000 testing images of size 28×28, and MNIST[LeCun(1998)], a dataset of handwritten digits, also having 60000 training images and 10000 testing images of size 28×28. Each of these two datasets has ten labels. Table 3.1 shows a summary of all data sets used in the experiment. Similar to the setting in the last chapter, for each dataset, we normalize by removing the mean and scaling to unit variance all features. We delete the first column for Ionosphere, where the values are the same within one of the classes, and the second column where all entries are 0. Lastly, for the MNIST, we delete 135 columns with no more than ten nonzero values (Note that we have ten classes. If one class has a column of all zeros, then the sample covariance matrix for that class is a degenerate matrix.)

[1] `https://pypi.org/project/fancyimpute/`
[2] `https://pypi.org/project/impyute/`
[3] `https://pypi.org/project/missingpy/`
[4] `https://scikit-learn.org/`

When using iterative methods for small datasets, we use the default options in the packages *fancyimpute, sklearn* and *impyute*. Specifically, the maximum number of iterations is 100 for SOFT-IMPUTE, MICE, Nuclear Norm Minimization, and 10 for MissForest. For big datasets (MNIST & Fashion MNIST), the maximum number of iterations for MICE and Nuclear Norm Minimization are reduced to 10 instead due to expensive memory and time requirements.

The experiments on small datasets are ran directly on *Google Colaboratory*[5]. Meanwhile, the experiments on large datasets are run on a Linux machine with 128 GB RAM, 8 cores, Xeon 6244 processor 3.66 Hz. We terminate an experiment if no result is produced after 5 hours of running or when having out-of-memory issues, and denote this as NA in the result tables (Tables 4.1 and 4.2).

For evaluation, as in the previous chapter, we use the following metric [Nguyen et al.(2021a)Nguyen, Nguyen, Nguyen, Nguyen, and Wade], which is the sum of the average difference of each entry in the mean/common covariance matrix.

$$r = \frac{||\boldsymbol{\mu} - \hat{\boldsymbol{\mu}}||_F}{n_{\boldsymbol{\mu}}} + \frac{||\boldsymbol{\Sigma} - \hat{\boldsymbol{\Sigma}}||_F}{n_{\boldsymbol{\Sigma}}}, \tag{4.3}$$

where $||.||$ denotes the Frobenius norm; $\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\Sigma}}$ are estimated values derived from DPER for $\boldsymbol{\mu}, \boldsymbol{\Sigma}$ respectively; $n_{\boldsymbol{\mu}}$, $n_{\boldsymbol{\Sigma}}$ are the corresponding number of entries in $\boldsymbol{\mu}, \boldsymbol{\Sigma}$ and the ground truth $\boldsymbol{\mu}, \boldsymbol{\Sigma}$ are calculated from the full data without missing values (because we do not know the true value of $\boldsymbol{\mu}, \boldsymbol{\Sigma}$). Note that unlike the mean square error, this evaluation measure takes into account the number of entries in the means and the covariance matrix instead.

[5] `https://colab.research.google.com/`

Table 4.1. Parameter estimation errors without equal covariance matrix assumption. Bold implies the best results.

Dataset	**Missing Rate(%)**	**DPER**	**MICE**	**SOFT-IMPUTE**	**Nuclear Norm**	**MissForest**
Seeds	20%	**0.004**	**0.004**	0.007	0.522	0.52
	35%	**0.008**	0.009	0.015	0.517	0.543
	50%	**0.007**	0.01	0.016	0.459	0.504
	65%	**0.009**	0.017	0.025	0.489	0.607
Iris	20%	**0.007**	0.009	0.013	0.693	0.71
	35%	**0.009**	0.015	0.025	0.665	0.857
	50%	**0.006**	0.017	0.026	0.597	0.812
	65%	**0.013**	0.033	0.049	0.672	0.998
Wine	20%	**0.005**	**0.005**	0.006	0.27	0.288
	35%	**0.009**	0.01	0.012	0.256	0.277
	50%	**0.008**	0.011	0.016	0.239	0.276
	65%	**0.012**	0.014	0.023	0.229	0.304
Ionosphere	20%	**0.003**	0.004	**0.003**	0.67	0.717
	35%	**0.004**	0.005	0.005	0.615	0.704
	50%	0.006	**0.005**	**0.005**	0.57	0.693
	65%	0.007	**0.006**	0.007	0.531	0.688
MNIST	20%	**0.4e-4**	NA	0.5e-4	NA	NA
	35%	**0.5e-4**	NA	0.9e-4	NA	NA
	50%	**0.7e-4**	NA	1.4e-4	NA	NA
	65%	**0.8e-4**	NA	2e-4	NA	NA
Fashion MNIST	20%	**0.3e-4**	NA	0.4e-4	NA	NA
	35%	**0.4e-4**	NA	0.7e-4	NA	NA
	50%	**0.5e-4**	NA	0.9e-4	NA	NA
	65%	**0.6e-4**	NA	1.3e-4	NA	NA

Table 4.2. Parameters estimation errors under the equal covariance matrix assumption. Bold implies the best results.

Dataset	Missing Rate(%)	DPER	MICE	SOFT-IMPUTE	Nuclear Norm	MissForest
Seeds	20%	0.004	**0.003**	0.005	0.005	0.004
	35%	0.007	**0.006**	0.01	0.01	0.008
	50%	**0.009**	0.013	0.02	0.019	0.011
	65%	**0.008**	0.014	0.024	0.023	0.016
Iris	20%	**0.008**	**0.008**	0.014	0.014	0.01
	35%	**0.01**	0.018	0.029	0.028	0.017
	50%	**0.01**	0.029	0.043	0.04	0.021
	65%	**0.01**	0.03	0.045	0.042	0.037
Wine	20%	**0.005**	0.006	0.007	0.007	**0.005**
	35%	**0.006**	0.009	0.011	0.011	0.007
	50%	**0.009**	0.01	0.016	0.015	0.01
	65%	**0.009**	0.014	0.021	0.02	0.013
Ionosphere	20%	0.004	**0.003**	**0.003**	**0.003**	**0.003**
	35%	0.005	**0.004**	**0.004**	**0.004**	**0.004**
	50%	0.006	0.006	0.006	0.006	**0.005**
	65%	0.008	**0.007**	0.008	**0.007**	**0.007**
MNIST	20%	0.4e-4	NA	**0.3e-4**	NA	NA
	35%	0.5e-4	NA	**0.4e-4**	NA	NA
	50%	**0.6e-4**	NA	**0.6e-4**	NA	NA
	65%	0.8e-4	NA	**0.7e-4**	NA	NA
Fashion MNIST	20%	**0.3e-4**	NA	0.4e-4	NA	NA
	35%	**0.4e-4**	NA	0.7e-4	NA	NA
	50%	**0.5e-4**	NA	1e-4	NA	NA
	65%	**0.6e-4**	NA	1.3e-4	NA	NA

4.3.2 Results And Discussion. Note that we do not conduct any experiment about single class randomly missing data. The reason is that the multiple class assuming equal covariance matrices problem boils down to one-class sub-problems as explained in section 4.1.2.

Table 4.1 shows that without the assumption of equal covariance matrices, DPER gives the best result on every data set except for the Ionosphere. There is little distinction between DPER with MICE and SOFT-IMPUTE at lower missing rates, but the higher the missing rate, the better DPER performs against its peers. Considering the Seeds data set, for example, at 20% missing rate, DPER gives the same best 0.004 error rate as MICE, but at 65% missing rate, the 0.008 error rate of DPER is almost less than half of the others.

From Table 4.2, we see that under the equal covariance assumption, DPER gives the best results on all of the missing rates on every data sets except for Ionosphere and MNIST. Still, for Ionosphere and MNIST, DPER's performance is competitive against other approaches, and when it does not perform best, the best method can only improve slightly than DPER (the maximum difference is 0.001 for Ionosphere and $0.1e-4$ for MNIST). Again, there is little distinction between DPER with other approaches at lower missing rates, but the higher the missing rate, the better DPER performs against its peers. For example, at 20% missing rate in the Iris data set, the 0.008 error rate of DPER is indistinguishable from the 0.008 error rate of MICE, but at 65% missing rate, the error rate is 0.01 compare to the second best of 0.03 of MICE.

Note that whether to assume the covariance matrices are equal or not depends on the application and the nature of datasets. Yet, from Table 4.1 and Table 4.2, we can also see how this assumption affects the algorithms. First, the assumption seems to benefit

DPER at high missing rates (65%) on small datasets (Seeds, Iris, Wine). For example, at 65% of missing values on Iris, there is some reduction in the 0.013 error rate of DPER to 0.01 when there is an equal covariance assumption. Yet, improvement may not happen in those datasets with lower missing rates. For example, for the Wine data set, at 50%, the experiment with the assumption gives an error rate of 0.009 while it is of 0.008 without the assumption. For MICE and SOFT-IMPUTE, there is no clear pattern on the effects of the assumption.

Meanwhile, Nuclear Norm and MissForest perform significantly better with the assumption. As an indication, with 20% missing rate in the Ionosphere data set, the 0.717 error rate of MissForest reduces over 99% to 0.005 in Table 4.2. The result suggests that for small datasets, the equal covariance matrix assumption benefits Nuclear Norm and MissForest significantly. This maybe due to the number of sample used for the covariance estimation is lower without the assumption.

The parameter estimation on large datasets such as MNIST and Fashion MNIST, DPER gives small error rates. Note that since the populational means and covariance matrix are unknown, the estimation is evaluated on the complete data sets (i.e., the data sets before the missing data simulation). This implies that with DPER, we can approximate means and covariance matrices with almost identical results with the complete data set. This result is reasonable since the Multivariate Central Limit Theorem suggests the maximum likelihood parameter approximation gets better with the increment of sample size, and these data sets are large. It should be remarked that only the results for DPER and SOFT-IMPUTE are available for these two data sets. Among the available methods, DPER is the fastest one with minimum run-time for parameter estimation on

MNIST at 69.54s, while SOFT-IMPUTE is the second fastest (165.15s), respectively. For other approaches, the corresponding outcomes are not available due to memory-bound or running-time bound. Therefore, we can deduce that DPER and SOFT-IMPUTE are more suitable than MICE, Nuclear Norm, and MissForest for large data sets.

4.4 Conclusion

In this chapter, we derived formulas and provided algorithms to compute the mean and covariance matrices from the data that are suitable for both small and large data sets, hence providing an approximation to the data's underlying distribution(s). We illustrate the power of our methods by experiments, which show that while iterative methods may not be able to handle the missing data problem for big data sets, our method can still find the parameter rapidly without requiring extensive memory and computation time. In addition, since the estimation is done for each pair of features, our method is also applicable to the problems where the number of features is much higher than the sample size.

Yet, our methods still have several drawbacks and room for improvement. First, when the distribution of the data is highly skew [Mardia(1970)], the bivariate normal distribution might not be a good approximator. Therefore, when the pair of features is highly skew, other methods than DPER should be utilized for that pair of features.

Second, the computation is done separately for each σ_{ij}, which only requires the data of the ith and jth features. This also means we have not been able to use the extra information from the other features of the dataset. Also, separate computation for each of σ_{ij} means we could parallelize the algorithm even further. These will be the topics for our future research.

Chapter 5: Conclusion And Remarks

In this work, we have discussed about the direct estimation of the mean and covariance matrices of a dataset.

First, we derived the exact formulas and an efficient algorithm for computing the parameters of a multiple class monotone missing data set where the covariance matrices of all the classes are assumed to be equal. Even though normality is considered, we have illustrated that the proposed algorithm can work well in practice and achieve better performance through different experiments than several other imputation-based approaches. Yet, this approach is only appropriate for monotone missing data.

Next, in order to find a direct estimation of the parameters that works on an arbitrary dataset, we estimate the parameters in pairs of features instead. This allows us to impose a milder assumption of bivariate normality on each pair of features instead of multivariate normality. This approach, like the previous ones on monotone missing data, is suitable for both small and large data sets. In addition, since the estimation is done for each pair of features, our method is also applicable to the problems where the number of features is much higher than the sample size. However, a drawback of this is we are not able to utilize the information available from other features.

For all of the approaches mentioned, we have illustrated their power by experiments, which show their state-of-the-art performance in terms of speed and error rate reduction. In addition, we can also conclude that while iterative methods may not be able to handle the missing data problem for big data sets, our method can still find the parameter rapidly without requiring extensive memory and computation time.

Yet, our methods still have some drawbacks. First, when the distribution of the

data is highly skew [Mardia(1970)], the multivariate normal distribution might not be a good approximator. Therefore, when the pair of features is highly skew, other methods than EPEM/DPER should be utilized for that pair of features.

Bibliography

[Anderson(1957)] Theodore W Anderson. Maximum likelihood estimates for a multivariate normal distribution when some observations are missing. *Journal of the American Statistical Association*, 52(278):200–203, 1957.

[Audigier et al.(2016)Audigier, Husson, and Josse] Vincent Audigier, François Husson, and Julie Josse. Multiple imputation for continuous variables using a bayesian principal component analysis. *Journal of Statistical Computation and Simulation*, 86 (11):2140–2156, 2016.

[Aydilek and Arslan(2013)] Ibrahim Berkan Aydilek and Ahmet Arslan. A hybrid method for imputation of missing values using optimized fuzzy c-means with support vector regression and a genetic algorithm. *Information Sciences*, 233:25–35, 2013.

[Browne(2000)] Michael W Browne. Cross-validation methods. *Journal of Mathematical Psychology*, 44(1):108–132, 2000.

[Burgette and Reiter(2010)] Lane F Burgette and Jerome P Reiter. Multiple imputation for missing data via sequential regression trees. *American journal of Epidemiology*, 172(9):1070–1076, 2010.

[Buuren and Groothuis-Oudshoorn(2010)] S van Buuren and Karin Groothuis-Oudshoorn. mice: Multivariate imputation by chained equations in r. *Journal of Statistical Software*, pages 1–68, 2010.

[Candès and Recht(2009)] Emmanuel J Candès and Benjamin Recht. Exact matrix completion via convex optimization. *Foundations of Computational Mathematics*, 9 (6):717, 2009.

[Casella and Berger(2002)] George Casella and Roger L Berger. *Statistical inference*, volume 2. Duxbury Pacific Grove, CA, 2002.

[Chhabra et al.(2018)Chhabra, Vashisht, and Ranjan] Geeta Chhabra, Vasudha Vashisht, and Jayanthi Ranjan. Missing value imputation using hybrid k-means and association rules. In *2018 International Conference on Advances in Computing, Communication Control and Networking (ICACCCN)*, pages 501–509. IEEE, 2018.

[Costa et al.(2018)Costa, Santos, Soares, and Abreu] Adriana Fonseca Costa, Miriam Seoane Santos, Jastin Pompeu Soares, and Pedro Henriques Abreu. Missing data imputation via denoising autoencoders: the untold story. In *International Symposium on Intelligent Data Analysis*, pages 87–98. Springer, 2018.

[Dua and Graff(2017)] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`.

[Fan et al.(2020)Fan, Zhang, and Udell] Jicong Fan, Yuqian Zhang, and Madeleine Udell. Polynomial matrix completion for missing data imputation and transductive learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3842–3849, 2020.

[Fujisawa(1995)] Hironori Fujisawa. A note on the maximum likelihood estimators for multivariate normal distribution with monotone data. *Communications in Statistics-Theory and Methods*, 24(6):1377–1382, 1995.

[Garcia and Hruschka(2005)] Antonio JT Garcia and Eduardo R Hruschka. Naive bayes as an imputation tool for classification problems. In *Fifth International Conference on Hybrid Intelligent Systems (HIS'05)*, pages 3–pp. IEEE, 2005.

[Garcia et al.(2019a)Garcia, Esmin, Leite, and Škrjanc] Cristiano Garcia, Ahmed Esmin, Daniel Leite, and Igor Škrjanc. Evolvable fuzzy systems from data streams with missing values: With application to temporal pattern recognition and cryptocurrency prediction. *Pattern Recognition Letters*, 128:278–282, 2019a.

[Garcia et al.(2019b)Garcia, Leite, and Skrjanc] Cristiano Garcia, Daniel Leite, and Igor Skrjanc. Incremental missing-data imputation for evolving fuzzy granular prediction. *IEEE Transactions on Fuzzy Systems*, 2019b.

[García-Laencina et al.(2010)García-Laencina, Sancho-Gómez, and Figueiras-Vidal] Pedro J García-Laencina, José-Luis Sancho-Gómez, and Aníbal R Figueiras-Vidal. Pattern classification with missing data: a review. *Neural Computing and Applications*, 19(2):263–282, 2010.

[Garg et al.(2018)Garg, Naryani, Aggarwal, and Aggarwal] Ayush Garg, Deepika Naryani, Garvit Aggarwal, and Swati Aggarwal. Dl-gsa: a deep learning metaheuristic approach to missing data imputation. In *International Conference on Sensing and Imaging*, pages 513–521. Springer, 2018.

[Gautam and Ravi(2015)] Chandan Gautam and Vadlamani Ravi. Data imputation via evolutionary computation, clustering and a neural network. *Neurocomputing*, 156: 134–142, 2015.

[Gondara and Wang(2017)] Lovedeep Gondara and Ke Wang. Multiple imputation using deep denoising autoencoders. *arXiv preprint arXiv:1705.02737*, 2017.

[Hruschka et al.(2007)Hruschka, Hruschka, and Ebecken] Estevam R Hruschka, Eduardo R Hruschka, and Nelson FF Ebecken. Bayesian networks for imputation in classification problems. *Journal of Intelligent Information Systems*, 29(3):231–252, 2007.

[Johnson et al.(2002)Johnson, Wichern, et al.] Richard Arnold Johnson, Dean W Wichern, et al. *Applied multivariate statistical analysis*, volume 5. Prentice hall Upper Saddle River, NJ, 2002.

[Kanda and Fujikoshi(1998)] Takashi Kanda and Yasunori Fujikoshi. Some basic properties of the mle's for a multivariate normal distribution with monotone missing data. *American Journal of Mathematical and Management Sciences*, 18(1-2):161–192, 1998.

[Kim et al.(2005)Kim, Golub, and Park] Hyunsoo Kim, Gene H Golub, and Haesun Park. Missing value estimation for DNA microarray gene expression data: local least squares imputation. *Bioinformatics*, 21(2):187–198, 2005.

[LeCun(1998)] Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

[Leke and Marwala(2016)] Collins Leke and Tshilidzi Marwala. Missing data estimation in high-dimensional datasets: A swarm intelligence-deep neural network approach. In *International Conference on Swarm Intelligence*, pages 259–270. Springer, 2016.

[Li and Li(2010)] Weiguo Li and Zhi Li. A family of iterative methods for computing the approximate inverse of a square matrix and inner inverse of a non-square matrix. *Applied Mathematics and Computation*, 215(9):3433–3442, 2010.

[M Mostafa et al.(2020)M Mostafa, S Eladimy, Hamad, and Amano] Samih M Mostafa, Abdelrahman S Eladimy, Safwat Hamad, and Hirofumi Amano. Cbrl and cbrc: Novel algorithms for improving missing value imputation accuracy based on bayesian ridge regression. *Symmetry*, 12(10):1594, 2020.

[Mardia(1970)] Kanti V Mardia. Measures of multivariate skewness and kurtosis with applications. *Biometrika*, 57(3):519–530, 1970.

[Mazumder et al.(2010)Mazumder, Hastie, and Tibshirani] Rahul Mazumder, Trevor Hastie, and Robert Tibshirani. Spectral regularization algorithms for learning large incomplete matrices. *Journal of Machine Learning Research*, 11(Aug):2287–2322, 2010.

[McKnight et al.(2007)McKnight, McKnight, Sidani, and Figueredo] Patrick E McKnight, Katherine M McKnight, Souraya Sidani, and Aurelio Jose Figueredo. *Missing data: A gentle introduction*. Guilford Press, 2007.

[Nguyen et al.(2021a)Nguyen, Nguyen, Nguyen, Nguyen, and Wade] Thu Nguyen, Duy H.M. Nguyen, Huy Nguyen, Binh T. Nguyen, and Bruce A. Wade. Epem: Efficient parameter estimation for multiple class monotone missing data. *Information Sciences*, 567:1–22, 2021a. ISSN 0020-0255. doi: https://doi.org/10.1016/j.ins.2021.02.077.

[Nguyen et al.(2021b)Nguyen, Nguyen-Duy, Nguyen, Nguyen, and Wade] Thu Nguyen, Khoi Minh Nguyen-Duy, Duy Ho Minh Nguyen, Binh T. Nguyen, and Bruce Alan Wade. Dper: Efficient parameter estimation for randomly missing data, 2021b.

[Nikfalazar et al.(2020)Nikfalazar, Yeh, Bedingfield, and Khorshidi] Sanaz Nikfalazar, Chung-Hsing Yeh, Susan Bedingfield, and Hadi A Khorshidi. Missing data imputation using decision trees and fuzzy clustering with iterative learning. *Knowledge and Information Systems*, 62(6):2419–2437, 2020.

[Petersen and Pedersen(2012)] Kaare Brandt Petersen and Michael Syskind Pedersen. The matrix cookbook, nov 2012. *URL http://www2. imm. dtu. dk/pubdb/p. php*, 3274, 2012.

[Rahman and Islam(2013)] Md Geaur Rahman and Md Zahidul Islam. Missing value imputation using decision trees and decision forests by splitting and merging records: Two novel techniques. *Knowledge-Based Systems*, 53:51–65, 2013.

[Rahman and Islam(2016)] Md Geaur Rahman and Md Zahidul Islam. Missing value imputation using a fuzzy clustering-based em approach. *Knowledge and Information Systems*, 46(2):389–422, 2016.

[Razavi-Far and Saif(2016)] Roozbeh Razavi-Far and Mehrdad Saif. Imputation of missing data using fuzzy neighborhood density-based clustering. In *2016 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 1834–1841. IEEE, 2016.

[Razavi-Far et al.(2020)Razavi-Far, Cheng, Saif, and Ahmadi] Roozbeh Razavi-Far, Boyuan Cheng, Mehrdad Saif, and Majid Ahmadi. Similarity-learning information-fusion schemes for missing data imputation. *Knowledge-Based Systems*, 187:104805, 2020.

[Sovilj et al.(2016)Sovilj, Eirola, Miche, Björk, Nian, Akusok, and Lendasse] Dušan Sovilj, Emil Eirola, Yoan Miche, Kaj-Mikael Björk, Rui Nian, Anton Akusok, and Amaury Lendasse. Extreme learning machine for missing data using multiple imputations. *Neurocomputing*, 174:220–231, 2016.

[Spinelli et al.(2020)Spinelli, Scardapane, and Uncini] Indro Spinelli, Simone Scardapane, and Aurelio Uncini. Missing data imputation with adversarially-trained graph convolutional networks. *Neural Networks*, 129:249 – 260, 2020. ISSN 0893-6080.

[Stekhoven and Bühlmann(2012)] Daniel J Stekhoven and Peter Bühlmann. Missforest—non-parametric missing value imputation for mixed-type data. *Bioinformatics*, 28(1):112–118, 2012.

[Templ et al.(2011)Templ, Kowarik, and Filzmoser] Matthias Templ, Alexander Kowarik, and Peter Filzmoser. Iterative stepwise regression imputation using standard and robust methods. *Computational Statistics & Data Analysis*, 55(10):2793–2806, 2011.

[Toutounian and Soleymani(2013)] Faezeh Toutounian and Fazlollah Soleymani. An iterative method for computing the approximate inverse of a square matrix and the moore–penrose inverse of a non-square matrix. *Applied Mathematics and Computation*, 224:671–680, 2013.

[Van der Vaart(2000)] Aad W Van der Vaart. *Asymptotic Statistics*, volume 3. Cambridge university press, 2000.

[Xiao et al.(2017)Xiao, Rasul, and Vollgraf] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[Yoon and Sull(2020)] Seongwook Yoon and Sanghoon Sull. Gamin: Generative adversarial multiple imputation network for highly missing data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8456–8464, 2020.

[Yu et al.(2006)Yu, Krishnamoorthy, and Pannala] Jianqi Yu, K Krishnamoorthy, and Maruthy K Pannala. Two-sample inference for normal mean vectors based on monotone missing data. *Journal of Multivariate Analysis*, 97(10):2162–2176, 2006.

Appendix A: Supplementary Materials For Chapter 3

A.1 An Illustration For Data Partition On Section 3.1

If one considers the following data set

$$\begin{pmatrix} m_{11} & m_{12} & m_{13} & m_{14} & m_{15} \\ m_{21} & m_{22} & m_{23} & m_{24} & m_{25} \\ m_{31} & m_{32} & * & * & * \\ m_{41} & m_{42} & * & * & * \end{pmatrix},$$

in which,

$$\boldsymbol{x} = \begin{pmatrix} m_{11} & m_{13} \\ m_{21} & m_{23} \\ m_{31} & * \\ m_{41} & * \end{pmatrix}, \quad \boldsymbol{y} = \begin{pmatrix} m_{12} & m_{14} & m_{15} \\ m_{22} & m_{24} & m_{25} \\ m_{32} & * & * \\ m_{42} & * & * \end{pmatrix}.$$

Then,

$$\boldsymbol{x}_1 = \begin{pmatrix} m_{11} & m_{13} \\ m_{21} & m_{23} \end{pmatrix}, \boldsymbol{x}_2 = \begin{pmatrix} m_{11} \\ m_{21} \\ m_{31} \\ m_{41} \end{pmatrix}, \quad \boldsymbol{y}_1 = \begin{pmatrix} m_{12} & m_{14} & m_{15} \\ m_{22} & m_{24} & m_{25} \end{pmatrix}, \boldsymbol{y}_2 = \begin{pmatrix} m_{12} \\ m_{22} \\ m_{32} \\ m_{42} \end{pmatrix}.$$

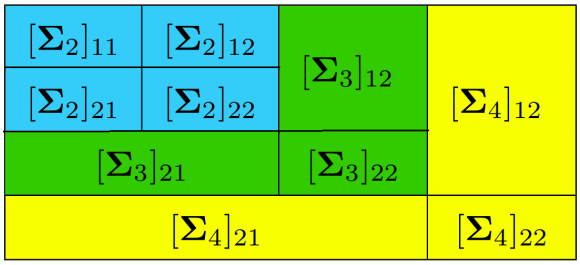
A.1.1 Our proposed computation diagram. The following computation diagram illustrates the ideas of data partitioning and the sequential computation process in Theorem 3.1. In this example, $k = 4$.

The mean of the first class is partitioned as (for simplicity, we remove some brackets):

$$\boldsymbol{\mu} = \boldsymbol{\mu}_4 \begin{cases} \boldsymbol{\mu}_3 \begin{cases} \boldsymbol{\mu}_2 \begin{cases} [\boldsymbol{\mu}]_1 \\ [\boldsymbol{\mu}]_2 \end{cases} \\ [\boldsymbol{\mu}]_3 \end{cases} \\ [\boldsymbol{\mu}]_4 \end{cases} \tag{A.1}$$

The sequential process for the computation of the mean of the first class is as follows. At first, we compute $\boldsymbol{\mu}_1 = [\boldsymbol{\mu}]_1$. Then, we compute $[\boldsymbol{\mu}]_2$ based on $[\boldsymbol{\mu}]_1$, which gives us $\boldsymbol{\mu}_2$. Next, we compute $[\boldsymbol{\mu}]_3$ based on $\boldsymbol{\mu}_2$, giving us $\boldsymbol{\mu}_3$. Finally, we compute $[\boldsymbol{\mu}]_4$ based on $[\boldsymbol{\mu}]_3$, providing us $\boldsymbol{\mu}_4$ as well as $\boldsymbol{\mu}$.

Figure A.1. The covariance computation process



For the computation of the covariance matrix, $\mathbf{\Sigma}$ is partitioned as in Table A.1. First, we calculate $[\mathbf{\Sigma}_2]_{11}$, which is also $\mathbf{\Sigma}_1$. Then, we estimate the remaining blue part based on that estimate, providing us $[\mathbf{\Sigma}_3]_{11}$. Next, we compute the green part based on the blue part. This gives us the estimate of $[\mathbf{\Sigma}_4]_{11}$. Finally, we compute the yellow part based on $[\mathbf{\Sigma}_4]_{11}$. This provides us $\mathbf{\Sigma}$.

A.1.2 An example for theorem 3.1: . Given

$$\begin{pmatrix} 3 & 5 & 1 & 3 & 2 & 8 & 1 & 0 \\ 2 & 1 & 7 & 8 & 4 & 7 & 2 & 1 \\ 1 & 2 & 8 & 9 & 0 & 5 & * & * \\ 4 & 3 & 5 & 4 & 1 & 3 & * & * \end{pmatrix} \tag{A.2}$$

in which

$$y = \begin{pmatrix} 3 & 5 & 1 & 1 \\ 2 & 1 & 7 & 2 \\ 1 & 2 & 8 & * \\ 4 & 3 & 5 & * \end{pmatrix}, \quad z = \begin{pmatrix} 3 & 2 & 8 & 0 \\ 8 & 4 & 7 & 1 \\ 9 & 0 & 5 & * \\ 4 & 1 & 3 & * \end{pmatrix}.$$

Then, one gets

$$y_1 = \begin{pmatrix} 3 & 5 & 1 & 1 \\ 2 & 1 & 7 & 2 \end{pmatrix}, \quad y_2 = \begin{pmatrix} 3 & 5 & 1 \\ 2 & 1 & 7 \\ 1 & 2 & 8 \\ 4 & 3 & 5 \end{pmatrix}, \quad z_1 = \begin{pmatrix} 3 & 2 & 8 & 0 \\ 8 & 4 & 7 & 1 \end{pmatrix}, \quad z_2 = \begin{pmatrix} 3 & 2 & 8 \\ 8 & 4 & 7 \\ 9 & 0 & 5 \\ 4 & 1 & 3 \end{pmatrix}.$$

Notice that $p_1 = 1, p_2 = 2, n_1 = m_1 = 4, n_2 = m_2 = 3, k = 2$. By calculation, one has

$$\overline{y}_1 = \begin{pmatrix} 2.5 \\ 3 \end{pmatrix}, \quad \overline{y}_2 = \begin{pmatrix} 3 \\ 3.33 \\ 3.67 \\ 4 \end{pmatrix}, \quad \overline{z}_1 = \begin{pmatrix} 3.25 \\ 5 \end{pmatrix}, \quad \overline{z}_2 = \begin{pmatrix} 4.33 \\ 6.33 \\ 4.67 \\ 2.67 \end{pmatrix},$$

$$S_1 = \begin{pmatrix} 3.67 & -3.33 \\ -3.33 & 7.33 \end{pmatrix}, \quad S_2 = \begin{pmatrix} 4 & -6 & -6 & 2 \\ -6 & 10.3 & 11.67 & 3 \\ -6 & 11.67 & 14.33 & 3 \\ -2 & 3 & 3 & 1 \end{pmatrix},$$

$$R_1 = \begin{pmatrix} 11.58 & 7.67 \\ 7.67 & 10 \end{pmatrix}, \quad R_2 = \begin{pmatrix} 10.33 & 2.83 & 3.17 & 1.67 \\ 2.83 & 4.33 & 9.17 & 3.17 \\ 3.17 & 9.17 & 20.33 & 6.83 \\ 1.67 & 3.17 & 6.83 & 2.33 \end{pmatrix}.$$

It turns out that

$$W_1 = (n_1 - 1)S_1 + (m_1 - 1)R_1 = \begin{pmatrix} 45.75 & 13 \\ 13 & 52 \end{pmatrix}$$

$$W_2 = (n_2 - 1)S_2 + (m_2 - 1)R_2 = \begin{pmatrix} 28.67 & -6.33 & -5.67 & -0.67 \\ -6.33 & 29.33 & 41.67 & 12.33 \\ -5.67 & 41.67 & 69.33 & 19.67 \\ -0.67 & 12.33 & 19.67 & 6.67 \end{pmatrix}.$$

Next, one can obtain

$$P_2 = [W_2]_{21}[W_2]_{11}^{-1} = \begin{pmatrix} -5.67 & 41.67 \\ -0.67 & 12.33 \end{pmatrix} \begin{pmatrix} 28.67 & -6.33 \\ -6.33 & 29.33 \end{pmatrix}^{-1} = \begin{pmatrix} 0.12 & 1.45 \\ 0.07 & 0.44 \end{pmatrix},$$

$$Q_2 = \frac{1}{n_2 + m_2}([W_2]_{22} - P_2[W_2]_{12}) = \begin{pmatrix} 1.62 & 0.32 \\ 0.32 & 0.22 \end{pmatrix}.$$

Finally, one can combine all of the above results to calculate the MLEs for the two classes case based on formulas in **Theorem 3.1**. Firstly, one has

$$[\hat{\boldsymbol{\mu}}]_1 = \hat{\boldsymbol{\mu}}_1 = \overline{y}_1 = \begin{pmatrix} 2.5 \\ 3 \end{pmatrix}, \quad [\hat{\boldsymbol{\mu}}]_2 = [\overline{y}_2]_2 - P_2([\overline{y}_2]_1 - \hat{\boldsymbol{\mu}}_1) = \begin{pmatrix} 3.67 \\ 4 \end{pmatrix},$$

$$[\hat{\boldsymbol{\eta}}]_1 = \hat{\boldsymbol{\eta}}_1 = \overline{z}_1 = \begin{pmatrix} 3.25 \\ 5 \end{pmatrix}, \quad [\hat{\boldsymbol{\eta}}]_2 = [\overline{z}_2]_2 - P_2([\overline{z}_2]_1 - \hat{\boldsymbol{\eta}}_1) = \begin{pmatrix} 2.61 \\ 2.01 \end{pmatrix}.$$

As a result,

$$\boldsymbol{\mu} = \begin{pmatrix} [\hat{\boldsymbol{\mu}}]_1 \\ [\hat{\boldsymbol{\mu}}]_2 \end{pmatrix} = \begin{pmatrix} 2.5 \\ 3 \\ 3.67 \\ 4 \end{pmatrix}, \quad \boldsymbol{\eta} = \begin{pmatrix} [\hat{\boldsymbol{\eta}}]_1 \\ [\hat{\boldsymbol{\eta}}]_2 \end{pmatrix} = \begin{pmatrix} 3.25 \\ 5 \\ 2.61 \\ 2.01 \end{pmatrix}.$$

Secondly,

$$\widehat{\Sigma}_1 = \frac{W_1}{n_1 + m_1} = \begin{pmatrix} 5.72 & 1.63 \\ 1.63 & 6.5 \end{pmatrix},$$

$$[\widehat{\Sigma}_2]_{21} = P_2[\widehat{\Sigma}_2]_{11} = P_2\widehat{\Sigma}_1 = \begin{pmatrix} 3.05 & 9.6 \\ 1.13 & 2.95 \end{pmatrix},$$

$$[\widehat{\Sigma}_2]_{22} = Q_2 + P_2[\widehat{\Sigma}_2]'_{21} = \begin{pmatrix} 3.63 & 5.76 \\ 1.03 & 2.21 \end{pmatrix}.$$

Therefore,

$$\boldsymbol{\Sigma} = \begin{pmatrix} \mathbf{5.72} & \mathbf{1.63} & \mathbf{3.05} & \mathbf{1.13} \\ \mathbf{1.63} & \mathbf{6.5} & \mathbf{9.6} & \mathbf{2.95} \\ \mathbf{3.05} & \mathbf{9.6} & \mathbf{3.63} & \mathbf{5.76} \\ \mathbf{1.13} & \mathbf{2.95} & \mathbf{1.03} & \mathbf{2.21} \end{pmatrix}.$$

A.1.3 An example for theorem 3.2:. If we use the same matrix A.2 to illustrate, then

$$\boldsymbol{x}^{(1)} = \boldsymbol{y}, \quad \boldsymbol{x}_1^{(1)} = \boldsymbol{y}_1, \quad \boldsymbol{x}_2^{(1)} = \boldsymbol{y}_2, \quad \bar{\boldsymbol{x}}_1^{(1)} = \bar{\boldsymbol{y}}_1, \quad \bar{\boldsymbol{x}}_2^{(1)} = \bar{\boldsymbol{y}}_2,$$

$$\boldsymbol{x}^{(2)} = \boldsymbol{z}, \quad \boldsymbol{x}_1^{(1)} = \boldsymbol{z}_1, \quad \boldsymbol{x}_2^{(1)} = \boldsymbol{z}_2, \quad \bar{\boldsymbol{x}}_1^{(1)} = \bar{\boldsymbol{z}}_1, \quad \bar{\boldsymbol{x}}_2^{(1)} = \bar{\boldsymbol{z}}_2.$$

and $p_1 = p_2 = 2, n_1^1 = n_1^2 = 4, n_2^1 = n_2^2 = 3, k = 2.$

In addition,

$$\bar{\boldsymbol{x}}_1 = (\bar{\boldsymbol{x}}_1^{(1)}, \bar{\boldsymbol{x}}_1^{(2)}) = \begin{pmatrix} 2.5 & 3.25 \\ 3 & 5 \end{pmatrix}, \bar{\boldsymbol{x}}_2 = (\bar{\boldsymbol{x}}_2^{(1)}, \bar{\boldsymbol{x}}_2^{(2)}) = \begin{pmatrix} 3 & 4.33 \\ 3.33 & 6.33 \\ 3.67 & 4.67 \\ 4 & 2.67 \end{pmatrix},$$

and

$$[\bar{\boldsymbol{x}}_2]_1 = \begin{pmatrix} 3 & 4.33 \\ 3.33 & 6.33 \end{pmatrix}, [\bar{\boldsymbol{x}}_2]_2 = \begin{pmatrix} 3.67 & 4.67 \\ 4 & 2.67 \end{pmatrix}.$$

Next, $\mathbf{W}_i, \mathbf{P}_i, \mathbf{Q}_i$ are identical to their corresponding $\mathbf{W}_i, \mathbf{P}_i, \mathbf{Q}_i$ in the previous example for i = 1,2. Moreover,

$$\hat{\boldsymbol{\mu}}_1 = \bar{\boldsymbol{x}}_1, \hat{\boldsymbol{\mu}}_2 = [\bar{\boldsymbol{x}}_2]_2 - P_2([\bar{\boldsymbol{x}}_2]_1 - \hat{\boldsymbol{\mu}}_1) = \begin{pmatrix} 3.67 & 2.61 \\ 4 & 2.01 \end{pmatrix}.$$

Therefore,

$$\hat{\boldsymbol{\mu}} = \begin{pmatrix} 2.5 & 3.25 \\ 3 & 5 \\ 3.67 & 2.61 \\ 4 & 2.01 \end{pmatrix}.$$

A.2 Proof Of Theorem 3.1

This is a special case of a general problem for multiple groups, of which the proof will be given in the next sections.

A.3 Proof Of Theorem 3.2

The proof makes use of the following identities from [Petersen and Pedersen(2012)]:

For a vector $\boldsymbol{x}$ *and matrices* $\mathbf{A}, \mathbf{B}, \mathbf{C}$ *of appropriate sizes*

$$\frac{\partial \boldsymbol{x}'\mathbf{B}\boldsymbol{x}}{\partial \boldsymbol{x}} = (\mathbf{B} + \mathbf{B}')\boldsymbol{x} \tag{A.1}$$

$$\frac{\partial tr(\mathbf{AXB})}{\partial \mathbf{X}} = \mathbf{A}'\mathbf{B}' \tag{A.2}$$

$$\frac{\partial tr(A\mathbf{XBX}'\mathbf{C})}{\partial \mathbf{X}} = \mathbf{A}'\mathbf{C}'\mathbf{X}\mathbf{B}' + \mathbf{CAXB}, \tag{A.3}$$

where A' *denotes the transpose of a matrix* A*.*

Now, we will prove this theorem. First, notice that maximizing the likelihood w.r.t $\boldsymbol{\mu}$ is equivalent to maximizing the likelihood w.r.t $\boldsymbol{\mu}^{(g)}, g = 1, ..., G$. We partition $\boldsymbol{\mu}^{(g)}$ as

$$\boldsymbol{\mu}^{(g)} = \begin{pmatrix} [\boldsymbol{\mu}^{(g)}]_1 \\ [\boldsymbol{\mu}^{(g)}]_2 \\ \vdots \\ [\boldsymbol{\mu}^{(g)}]_k \end{pmatrix} \text{ of sizes } \begin{pmatrix} p_1 \times 1 \\ p_2 \times 1 \\ \vdots \\ p_k \times 1 \end{pmatrix}, g = 1, ..., G. \tag{A.4}$$

Then, let

$$\boldsymbol{\mu}_i^{(g)} = \begin{pmatrix} [\boldsymbol{\mu}^{(g)}]_1 \\ [\boldsymbol{\mu}^{(g)}]_2 \\ \vdots \\ [\boldsymbol{\mu}^{(g)}]_i \end{pmatrix}, g = 1, ..., G, \ i = 1, ...k. \tag{A.5}$$

To represent $\widehat{[\boldsymbol{\mu}^{(g)}]}_i$ in terms of $\widehat{\boldsymbol{\mu}}_{i-1}^{(g)}$, one can also partition

$$\bar{\boldsymbol{x}}_i^{(g)} = \begin{pmatrix} [\bar{\boldsymbol{x}}_i^{(g)}]_1 \\ [\bar{\boldsymbol{x}}_i^{(g)}]_2 \end{pmatrix} \text{ of sizes } \begin{pmatrix} (\sum_{j=1}^{i-1} p_j) \times 1 \\ p_i \times 1 \end{pmatrix}. \tag{A.6}$$

We assume that $\boldsymbol{z}_{ij}^{(g)}$ be the j^{th} column of $\boldsymbol{x}_i^{(g)}$. For $h \geq 2$, we denote $[\boldsymbol{z}_{ij}^{(g)}]_{h-1}$ as a vector containing the first $\sum_{s=1}^{h-1} p_s$ elements of $\boldsymbol{z}_{ij}^{(g)}$ and $[\boldsymbol{z}_{ij}^{(g)}]_{h/(h-1)}$ includes the next p_h elements.

Let $n_{k+1}^g = 0$ for all $1 \leq g \leq G$, and use f to denote the density of the variable inside "$f(.)$" in general (for notation simplicity). In addition, denote by $N_d(\boldsymbol{\gamma}, \boldsymbol{x})$ the $d-$dimensional multivariate normal distribution with mean $\boldsymbol{\gamma}$ and covariance matrix $\boldsymbol{\Sigma}$. We apply the following Lemma A.1 to prove Theorem 2.

Lemma A.1. *The log-likelihood function can be formulated as*

$$\eta = \sum_{i=1}^{k} \eta_i, \tag{A.7}$$

where

$$\eta_1 = \sum_{g=1}^{G} \sum_{l=1}^{k} \sum_{j=n_{l+1}^g+1}^{n_l^g} f(\boldsymbol{z}_{1j}^{(g)}), \tag{A.8}$$

and

$$\eta_i = \sum_{g=1}^{G} \sum_{l=i}^{k} \sum_{j=n_{l+1}^g+1}^{n_l^g} f([\boldsymbol{z}_{lj}^{(g)}]_{i/(i-1)} | [\boldsymbol{z}_{lj}^{(g)}]_{i-1}), \quad 2 \leq i \leq k. \tag{A.9}$$

(a) To derive MLEs of $\boldsymbol{\mu}_1^{(g)}, \boldsymbol{\Sigma}_1$:

Deriving the MLEs of $\boldsymbol{\mu}_1^{(g)}$, $\boldsymbol{\Sigma}_1$ is straightforward by noting that $\boldsymbol{z}_{1g}^{(g)} \sim N_{p_1}(\boldsymbol{\mu}_1^{(g)}, \boldsymbol{\Sigma}_1)$ as well as

$$\frac{\partial \eta}{\partial \boldsymbol{\mu}_1^{(g)}} = \frac{\partial \eta_1}{\partial \boldsymbol{\mu}_1^{(g)}} \quad \text{and} \quad \frac{\partial \eta}{\partial \boldsymbol{\Sigma}_1} = \frac{\partial \eta_1}{\partial \boldsymbol{\Sigma}_1}, \tag{A.10}$$

and solving for $\boldsymbol{\mu}_1^{(g)}$ and $\boldsymbol{\Sigma}_1$ from

$$\frac{\partial \eta_1}{\partial \boldsymbol{\mu}_1^{(g)}} = 0 \quad \text{and} \quad \frac{\partial \eta_1}{\partial \boldsymbol{\Sigma}_1} = 0. \tag{A.11}$$

(b) Explicit expression for $\eta_i (2 \leq i \leq k)$: To find the MLEs of $[\boldsymbol{\mu}]_i^{(g)}$, $[\boldsymbol{\Sigma}_i]_{12}$, $[\boldsymbol{\Sigma}]_{22}$ by taking derivatives, we need an explicit expression for $\eta_i (2 \leq i \leq k)$. Note that the data are assumed to follow multivariate normal distribution and recall that for $i \geq 2$, $[\boldsymbol{z}_{lj}^{(g)}]_{i-1}$ contains the first $\sum_{s=1}^{i-1} p_s$ elements of $\boldsymbol{z}_{lj}^{(g)}$ and $[\boldsymbol{z}_{lj}^{(g)}]_{i/(i-1)}$ contains the next p_i elements. Hence

$$[\boldsymbol{z}_{lj}^{(g)}]_i = \begin{pmatrix} [\boldsymbol{z}_{lj}^{(g)}]_{i-1} \\ [\boldsymbol{z}_{lj}^{(g)}]_{i/(i-1)} \end{pmatrix}.$$

In addition,

$$[\boldsymbol{z}_{lj}^{(g)}]_i \sim N_{\sum_{j=1}^{i} p_j}(\boldsymbol{\mu}_i^{(g)}, \Sigma_i).$$

Therefore, to find an explicit expression for η_i, we use the following lemma, whose proof could be found in [Johnson et al.(2002)Johnson, Wichern, et al.]:

Lemma 2 *(conditional distribution for multivariate normal distribution) Suppose* $\mathbf{y} \sim N_p(\boldsymbol{\gamma}, \boldsymbol{\Gamma})$. *Let*

$$\mathbf{y} = \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \end{bmatrix} \text{ with sizes } \begin{bmatrix} q \times 1 \\ (p-q) \times 1 \end{bmatrix}$$

and accordingly we partition

$$\boldsymbol{\gamma} = \begin{bmatrix} \boldsymbol{\gamma}_1 \\ \boldsymbol{\gamma}_2 \end{bmatrix} \text{ with sizes } \begin{bmatrix} q \times 1 \\ (p-q) \times 1 \end{bmatrix}$$

and

$$\boldsymbol{\Gamma} = \begin{bmatrix} \boldsymbol{\Gamma}_{11} & \boldsymbol{\Gamma}_{12} \\ \boldsymbol{\Gamma}_{21} & \boldsymbol{\Gamma}_{22} \end{bmatrix} \text{ with sizes } \begin{bmatrix} q \times q & q \times (p-q) \\ (p-q) \times q & (p-q) \times (p-q) \end{bmatrix}.$$

Then the distribution of $\mathbf{y}_1$ *conditional on* $\mathbf{y}_2 = a$ *follows multivariate normal distribution with mean*

$$\bar{\boldsymbol{\gamma}} = \boldsymbol{\gamma}_1 + \boldsymbol{\Gamma}_{12}\boldsymbol{\Gamma}_{22}^{-1}(\mathbf{a} - \boldsymbol{\gamma}_2)$$

and covariance matrix

$$\overline{\boldsymbol{\Gamma}} = \boldsymbol{\Gamma}_{11} - \boldsymbol{\Gamma}_{12}\boldsymbol{\Gamma}_{22}^{-1}\boldsymbol{\Gamma}_{21}.$$

Therefore, we can prove similar to the lemma above that

$$[\boldsymbol{z}_{lj}^{(g)}]_{i/(i-1)}|[\boldsymbol{z}_{lj}^{(g)}]_{i-1} \sim N_{p_i}\left(\mathbf{m}, \mathbf{H}\right), \tag{A.12}$$

where

$$\mathbf{m} = [\boldsymbol{\mu}^{(g)}]_i + B_i([\boldsymbol{z}_{lj}^{(g)}]_{i-1} - \boldsymbol{\mu}_{i-1}^{(g)}), \tag{A.13}$$

$$\mathbf{H} = [\Sigma_i]_{22} - B_i[\Sigma_i]_{12}, \tag{A.14}$$

and

$$\boldsymbol{B_i = [\Sigma_i]_{21}([\Sigma_i]_{11})^{-1}.} \tag{A.15}$$

(We ignored the indices of $\mathbf{m}$ and $\mathbf{H}$ for notational simplicity.)

Therefore, maximizing the likelihood w.r.t. $[\boldsymbol{\mu}^{(g)}]_i$,$[\Sigma_i]_{12}$, $[\Sigma_i]_{11}$ is equivalent to maximizing it w.r.t $[\boldsymbol{\mu}^{(g)}]_i$, $B_i, \mathbf{H}$. From Eq. (A.12) we have

$$\eta_i = C - \frac{1}{2}\sum_{g=1}^{G}\sum_{l=i}^{k}(n_l - n_{l+1})\ln \mid \mathbf{H} \mid - \frac{1}{2}\sum_{g=1}^{G}\sum_{l=i}^{k}\sum_{j=n_{l+1}^{g}+1}^{n_l^g} u'_{iglj}\mathbf{H}^{-1}u_{iglj}, \tag{A.16}$$

where C is a constant and $2 \leq i \leq k$.

$$u_{iglj} = [\boldsymbol{z}_{lj}^{(g)}]_{i/(i-1)} - [\boldsymbol{\mu}^{(g)}]_i - B_i\left([\boldsymbol{z}_{lj}^{(g)}]_{i-1} - \boldsymbol{\mu}_{i-1}^{(g)}\right). \tag{A.17}$$

(c) To derive MLEs of $[\boldsymbol{\mu}^{(g)}]_i (2 \leq i \leq k)$:

Using Eq. (A.1), we have

$$\frac{\partial \eta_i}{\partial u_{iglj}} = -\frac{1}{2}2\mathbf{H}^{-1}u_{iglj} \text{ and } \partial u_{iglj} = -\partial[\boldsymbol{\mu}^{(g)}]_i.$$

Therefore,

$$\frac{\partial \eta}{\partial u_{iglj}} = \frac{\partial \eta_i}{\partial [\boldsymbol{\mu}^{(g)}]_i} = \mathbf{H}^{-1}\sum_{l=i}^{k}\sum_{j=n_{l+1}^{g}+1}^{n_l^g} u_{iglj},$$

which implies

$$\frac{\partial \eta}{\partial [\boldsymbol{\mu}^{(g)}]_i} = 0 \Longleftrightarrow \sum_{l=i}^{k} \sum_{j=n_{l+1}^g+1}^{n_l^g} u_{iglj} = 0 \tag{A.18}$$

$$\overset{Eq.(A.17)}{\Longleftrightarrow} \sum_{l=i}^{k} \sum_{j=n_{l+1}^g+1}^{n_l^g} \left[[\boldsymbol{z}_{lj}^{(g)}]_{i/(i-1)} - [\boldsymbol{\mu}^{(g)}]_i - B_i \left([\boldsymbol{z}_{lj}^{(g)}]_{i-1} - \boldsymbol{\mu}_{i-1}^{(g)} \right) \right] = 0 \tag{A.19}$$

In addition, note that $\bar{\boldsymbol{x}}_i^{(g)}$ is the mean of $\boldsymbol{x}_i^{(g)}$. Therefore,

$$\bar{\boldsymbol{x}}_i^{(g)} = \frac{\sum_{l=i}^{k} \sum_{j=n_{l+1}^g+1}^{n_l^g} \boldsymbol{z}_{lj}^{(g)}}{\sum_{l=i}^{k} (n_l^g - n_{l+1}^g)}. \tag{A.20}$$

Therefore, by simplifying Eq. (A.19), using Eq. (A.6) with the above fact yields an estimation for $[\boldsymbol{\mu}^{(g)}]_i$:

$$[\widehat{\boldsymbol{\mu}}^{(g)}]_i = [\bar{\boldsymbol{x}}_i^{(g)}]_2 - B_i([\bar{\boldsymbol{x}}_i^{(g)}]_1 - \widehat{\boldsymbol{\mu}}_{i-1}^{(g)}). \tag{A.21}$$

(d) To derive the MLEs of $\mathbf{B}_i (i = 2, \ldots, k)$:

Plugging Eq. (A.21) into η_i, we have

$$\eta_i = C - \frac{1}{2} \sum_{g=1}^{G} \sum_{l=i}^{k} (n_l^g - n_{l+1}^g) \ln \mid \mathbf{H} \mid - \frac{1}{2} tr \left(\mathbf{H}^{-1} D \right), \tag{A.22}$$

where $tr(A)$ is the trace of matrix A and

$$D = \sum_{g=1}^{G} \sum_{l=i}^{k} \sum_{j=n_{l+1}^g+1}^{n_l} \left[\left(([\boldsymbol{z}_{lj}^{(g)}]_{i/(i-1)} - [\bar{\boldsymbol{x}}_i^{(g)}]_2) - B_i \left([\boldsymbol{z}_{lj}^{(g)}]_{i-1} - [\bar{\boldsymbol{x}}_i^{(g)}]_1 \right) \right) \right.$$
$$\left. \times \left(([\boldsymbol{z}_{lj}^{(g)}]_{i/(i-1)} - [\bar{\boldsymbol{x}}_i^{(g)}]_2) - B_i \left([\boldsymbol{z}_{lj}^{(g)}]_{i-1} - [\bar{\boldsymbol{x}}_i^{(g)}]_1 \right) \right)' \right].$$

Moreover, recall that

$$\mathbf{W}_i = \sum_{g=1}^{G} (n_i^g - 1) S_i^{(g)}, \ i = 1, ..., k. \tag{A.23}$$

Therefore, by factoring, we can get

$$D = [\boldsymbol{W}_i]_{22} - B_i [\boldsymbol{W}_i]_{12} - (B_i [\boldsymbol{W}_i]_{12})' + B_i [\boldsymbol{W}_i]_{11} B_i'.$$

Hence, taking derivative of η_i in Eq. (A.22) w.r.t B_i, we have

$$\frac{\partial \eta_i}{\partial B_i} \propto \frac{\partial}{\partial B_i}\left[tr(\mathbf{H}^{-1}[\boldsymbol{W}_i]_{22}) - tr(\mathbf{H}^{-1}B_i[\boldsymbol{W}_i]_{12}) \right.$$
$$\left. - tr\left(\mathbf{H}^{-1}(B_i[\boldsymbol{W}_i]_{12})'\right) + tr(\mathbf{H}^{-1}B_i[\boldsymbol{W}_i]_{11}B_i')\right].$$

Moreover, since the trace of a matrix is equal to the trace of its transpose, it implies that

$$tr(\mathbf{H}^{-1}(B_i[\boldsymbol{W}_i]_{12})') = tr(B_i[\boldsymbol{W}_i]_{12}\mathbf{H}^{-1})$$
$$= tr(\mathbf{H}^{-1}B_i[\boldsymbol{W}_i]_{12}).$$

Therefore, using identities (A.2) and (A.3), we have

$$\frac{\partial \eta}{\partial B_i} = \frac{\partial \eta_i}{\partial B_i} \propto -2\mathbf{H}^{-1}[\boldsymbol{W}_i]_{12}' + \mathbf{H}^{-1}B_i[\mathbf{W}_i]_{11} + \mathbf{H}^{-1}B_i[\mathbf{W}_i]_{11}.$$

Hence,

$$\frac{\partial \eta}{\partial B_i} = 0 \Longleftrightarrow B_i[\mathbf{W}_i]_{11} = [\boldsymbol{W}_i]_{12}' = [\boldsymbol{W}_i]_{21},$$

which means the maximum likelihood estimate of B_i is

$$\boldsymbol{P}_i = [\boldsymbol{W}_i]_{21}[\boldsymbol{W}_i]_{11}^{-1}. \tag{A.24}$$

(e) To estimate H:

Computing the MLE for $\mathbf{H}$ is similar to finding the MLE for the covariance matrix of a multivariate normal distribution: Taking derivative of η_i in Eq. (A.22) w.r.t $(\mathbf{H})^{-1}$ and using the trace trick $\boldsymbol{b}'A\boldsymbol{b} = tr(A\boldsymbol{b}\boldsymbol{b}')$ for a matrix A and a vector $\boldsymbol{b}$, we can obtain the maximum likelihood estimate of $\mathbf{H}$ as follows:

$$\boldsymbol{Q}_i = \frac{1}{\sum_{g=1}^{G} n_i^g}\left([\boldsymbol{W}_i]_{22} - \boldsymbol{P}_i[\boldsymbol{W}_i]_{12}\right). \tag{A.25}$$

(f) To compute $\widehat{[\Sigma_i]}_{22}$ and $\widehat{[\Sigma_i]}_{12}$:

From Eq. (A.15) and Eq. (A.24), one can see that the estimate for $[\Sigma_i]_{12}$ can be given by:

$$\widehat{[\Sigma_i]}_{12} = P_i \widehat{[\Sigma_i]}_{11} \tag{A.26}$$

Next, from Eq. (A.14) and Eq. (A.25), we can calculate the estimate for $[\Sigma_i]_{22}$ as:

$$\widehat{[\Sigma_i]}_{22} = Q_i + P_i \widehat{[\Sigma_i]}_{12} \tag{A.27}$$

(g) To prove that $[\hat{\boldsymbol{\mu}}]_i = [\bar{\boldsymbol{x}}_i]_2 - \mathbf{P}_i([\bar{\boldsymbol{x}}_i^{(g)}]_1 - \hat{\boldsymbol{\mu}}_{i-1})$:

Recall that $\widehat{\boldsymbol{\mu}}_1 = \bar{\boldsymbol{x}}_1$ and plugging Eq. (A.15) into Eq. (A.21) gives

$$[\widehat{\boldsymbol{\mu}}^{(g)}]_i = [\bar{\boldsymbol{x}}_i^{(g)}]_2 - \mathbf{P}_i([\bar{\boldsymbol{x}}_i^{(g)}]_1 - \widehat{\boldsymbol{\mu}}_{i-1}^{(g)}). \tag{A.28}$$

In addition, recall that

$$\widehat{\boldsymbol{\mu}}_i = \begin{pmatrix} \widehat{\boldsymbol{\mu}}_{i-1} \\ [\widehat{\boldsymbol{\mu}}]_i \end{pmatrix} = \begin{pmatrix} \widehat{\boldsymbol{\mu}}_{i-1}^{(1)}, \widehat{\boldsymbol{\mu}}_{i-1}^{(2)}, ..., \widehat{\boldsymbol{\mu}}_{i-1}^{(G)} \\ [\widehat{\boldsymbol{\mu}}^{(1)}]_i, [\widehat{\boldsymbol{\mu}}^{(2)}]_i, ..., [\widehat{\boldsymbol{\mu}}^{(G)}]_i \end{pmatrix},$$
$$\bar{\boldsymbol{x}}_i = \begin{pmatrix} [\bar{\boldsymbol{x}}_i]_1 \\ [\bar{\boldsymbol{x}}_i]_2 \end{pmatrix} = \begin{pmatrix} [\bar{\boldsymbol{x}}_i^{(1)}]_1, [\bar{\boldsymbol{x}}_i^{(2)}]_1, ..., [\bar{\boldsymbol{x}}_i^{(G)}]_1 \\ [\bar{\boldsymbol{x}}_i^{(1)}]_2, [\bar{\boldsymbol{x}}_i^{(2)}]_2, ..., [\bar{\boldsymbol{x}}_i^{(G)}]_2 \end{pmatrix}.$$

Moreover, note that for matrices E, F of order $m \times n, n \times s$, respectively, and if $f_1, .., f_s$ are the $1^{st}, 2^{nd}, ..., s^{th}$ columns of F then $EF = (Ef_1, Ef_2, ..., Ef_s)$. Therefore,

$$[\hat{\boldsymbol{\mu}}]_i = [\bar{\boldsymbol{x}}_i]_2 - \mathbf{P}_i([\bar{\boldsymbol{x}}_i^{(g)}]_1 - \hat{\boldsymbol{\mu}}_{i-1}).$$

A.3.1 Proof of lemma A.1. The proof makes use of the following identity:

$$\prod_{l=2}^{k} \prod_{i=2}^{l} a_{li} = \prod_{i=2}^{k} \prod_{l=i}^{k} a_{li} \tag{A.29}$$

for any sequence $\{a_{li}\}_{l=2,i=2}^{k,l}$.

Proof of the identity The proof of the identity is straight forward by noting that

$$a_{22}(a_{32}a_{33})(a_{42}a_{43}a_{44})....(a_{k1}a_{k2}...a_{kk}) = (a_{22}a_{32}...a_{k2})(a_{33}a_{43}...a_{k3})....a_{kk} \tag{A.30}$$

Proof of Lemma A.1 Recall that $\boldsymbol{z}_{lj}^{(g)}$ is the j^{th} column of $\boldsymbol{x}_l^{(g)}$. Therefore, the likelihood is

$$L = \prod_{g=1}^{G} \prod_{l=1}^{k} \prod_{j=n_{l+1}^g+1}^{n_l^g} f(\boldsymbol{z}_{lj}^{(g)}) = \prod_{g=1}^{G} L^{(g)}, \tag{A.31}$$

where

$$L^{(g)} = \prod_{l=1}^{k} \prod_{j=n_{l+1}^g+1}^{n_l^g} f(\boldsymbol{z}_{lj}^{(g)}). \tag{A.32}$$

Recall that for $2 \leq h \leq k$, $[\boldsymbol{z}_{lj}^{(g)}]_{h-1}$ contains the first $\sum_{s=1}^{h-1} p_s$ elements of $\boldsymbol{z}_{lj}^{(g)}$ and $[\boldsymbol{z}_{lj}^{(g)}]_{h/(h-1)}$ contains the next p_h elements. Hence

$$[\boldsymbol{z}_{lj}^{(g)}]_h = \begin{pmatrix} [\boldsymbol{z}_{lj}^{(g)}]_{h-1} \\ [\boldsymbol{z}_{lj}^{(g)}]_{h/(h-1)} \end{pmatrix} \tag{A.33}$$

and

$$[\boldsymbol{z}_{lj}^{(g)}]_1 = \boldsymbol{z}_{1j}^{(g)} \ \ \forall i = 1, .., k \tag{A.34}$$

$$[\boldsymbol{z}_{lj}^{(g)}]_l = \boldsymbol{z}_{lj}^{(g)} \ \ \forall l = 2, .., k. \tag{A.35}$$

Therefore, for $2 \leq l \leq k$:

$$f(\boldsymbol{z}_{lj}^{(g)}) = f(\boldsymbol{z}_{1j}^{(g)}) \frac{f([\boldsymbol{z}_{lj}^{(g)}]_2)}{f(\boldsymbol{z}_{1j}^{(g)})} \frac{f([\boldsymbol{z}_{lj}^{(g)}]_3)}{f([\boldsymbol{z}_{lj}^{(g)}]_2)} ... \frac{f([\boldsymbol{z}_{lj}^{(g)}]_l)}{f([\boldsymbol{z}_{lj}^{(g)}]_{l-1})}.$$

Next, note that

$$\frac{f([\boldsymbol{z}_{lj}^{(g)}]_s)}{f([\boldsymbol{z}_{lj}^{(g)}]_{s-1})} = f([\boldsymbol{z}_{lj}^{(g)}]_{s/(s-1)} | [\boldsymbol{z}_{lj}^{(g)}]_{s-1}).$$

Hence, combining with Eq. (A.34) and Eq. (A.35) implies

$$\begin{aligned} f(\boldsymbol{z}_{lj}^{(g)}) &= f(\boldsymbol{z}_{1j}^{(g)}) f([\boldsymbol{z}_{lj}^{(g)}]_{2/1} | \boldsymbol{z}_{1j}^{(g)}) f([\boldsymbol{z}_{lj}^{(g)}]_{3/2} | [\boldsymbol{z}_{lj}^{(g)}]_2) ... f([\boldsymbol{z}_{lj}^{(g)}]_{l/(l-1)} | [\boldsymbol{z}_{lj}^{(g)}]_{l-1}) \\ &= f(\boldsymbol{z}_{1j}^{(g)}) \prod_{i=2}^{l} f([\boldsymbol{z}_{lj}^{(g)}]_{i/(i-1)} | [\boldsymbol{z}_{l,j}^{(g)}]_{i-1}). \end{aligned}$$

This deduces that

$$L^{(g)} = \left[\prod_{l=2}^{k}\prod_{j=n_{l+1}^g+1}^{n_l^g}\left(f(\boldsymbol{z}_{1j}^{(g)})\prod_{i=2}^{l}f([\boldsymbol{z}_{lj}^{(g)}]_{i/(i-1)}|[\boldsymbol{z}_{lj}^{(g)})]_{i-1}\right)\right] \times \prod_{j=n_2^g+1}^{n_1^g} f(\boldsymbol{z}_{1j}^{(g)})$$
$$= \left(\prod_{l=1}^{k}\prod_{j=n_{l+1}^g+1}^{n_l^g} f(\boldsymbol{z}_{1j}^{(g)})\right) \times \left(\prod_{l=2}^{k}\prod_{j=n_{l+1}^g+1}^{n_l^g}\prod_{i=2}^{l} f([\boldsymbol{z}_{lj}^{(g)}]_{i/(i-1)}|[\boldsymbol{z}_{lj}^{(g)})]_{i-1}\right).$$

Together with Eq. (A.29), this yields,

$$L = \left(\prod_{g=1}^{G}\prod_{l=1}^{k}\prod_{j=n_{l+1}^g+1}^{n_l^g} f(\boldsymbol{z}_{1j}^{(g)})\right) \times \left(\prod_{i=2}^{k}\prod_{g=1}^{G}\prod_{l=i}^{k}\prod_{j=n_{l+1}^g+1}^{n_l^g} f([\boldsymbol{z}_{lj}^{(g)}]_{i/(i-1)}|[\boldsymbol{z}_{lj}^{(g)}]_{i-1})\right)$$
$$= \prod_{i=1}^{k} L_i,$$

where

$$L_1 = \prod_{g=1}^{G}\prod_{l=1}^{k}\prod_{j=n_{l+1}^g+1}^{n_l^g} f(\boldsymbol{z}_{1j}^{(g)}), \tag{A.36}$$

and

$$L_i = \prod_{g=1}^{G}\prod_{l=i}^{k}\prod_{j=n_{l+1}^g+1}^{n_l^g} f([\boldsymbol{z}_{lj}^{(g)}]_{i/(i-1)}|[\boldsymbol{z}_{lj}^{(g)}]_{i-1}), \quad 2 \le i \le k. \tag{A.37}$$

Therefore, the log likelihood can be calculated as:

$$\eta = \sum_{i=1}^{k} \log L_i = \sum_{i=1}^{k} \eta_i.$$

A.4 Proof of Theorem 3.3

Maximum likelihood estimates are asymptotic (see [Van der Vaart(2000)]).

Therefore, the proof follows directly from the result of Theorem 3.2.

A.5 Proof Of Theorem 4.1

The proof is straightforward by using the well-known result that if $\widehat{\theta}$ is the MLE for θ, and if $g(\theta)$ is any transformation of θ , then the MLE for $g(\theta)$ is $g(\widehat{\theta}\,)$ (see [Van der Vaart(2000)]).

Let $\boldsymbol{y} = A\boldsymbol{x}$ and suppose that $\boldsymbol{x}$ belongs to the ith class, then the new data follow a normal distribution with mean $A\mu_i$ and covariance matrix $A\Sigma A'$. The MLEs for the mean and covariance matrix of the transformed data are $A\hat{\boldsymbol{\mu}}_i, A\widehat{\Sigma}A'$, respectively. The linear discriminant analysis classification rule after the transformation can be given as:

$$\hat{d}_i(\boldsymbol{y}) = (A\hat{\boldsymbol{\mu}}_i)'\widehat{\Sigma}^{-1}A^{-1}\hat{\boldsymbol{x}} - \frac{1}{2}(A\hat{\boldsymbol{\mu}}_i)'(A\widehat{\Sigma}A')^{-1}A\hat{\boldsymbol{\mu}}_i + \ln\hat{r}_i, \tag{A.1}$$

for $i = 1, 2, ..., G$.

As A is invertible, the above equation simplifies to

$$\hat{d}_i(\boldsymbol{x}) = \hat{\boldsymbol{\mu}}_i'\widehat{\Sigma}^{-1}\hat{\boldsymbol{x}} - \frac{1}{2}\hat{\boldsymbol{\mu}}_i'\widehat{\Sigma}^{-1}\hat{\boldsymbol{\mu}}_i + \ln\hat{r}_i, \tag{A.2}$$

and the proof follows.

A.6 Proof Of Theorem 4.2

Lemma A.2. *(Slutsky's theorem) Let X_n, X be random vectors and Y_n be random matrices. If $X_n \xrightarrow{d} X$ and $Y_n \xrightarrow{d} c$ where c is a constant matrix, then*

i) $X_nY_n \xrightarrow{d} Xc$;

ii) $X_nY_n^{-1} \xrightarrow{d} Xc^{-1}$, *provided Y_n and c are invertible matrices.*

The proof of this lemma can be found in [Van der Vaart(2000)].

Now, let us fix the value of $\boldsymbol{x}$. We will prove for the case $i = 1$, and then other cases can be done similarly.

First, $\hat{\boldsymbol{\mu}}_1$ and $\widehat{\Sigma}$ are the maximum likelihood estimators of $\boldsymbol{\mu}_1$ and Σ, respectively, which implies that

$$\begin{aligned} \hat{\boldsymbol{\mu}}_1 &\xrightarrow{p} \boldsymbol{\mu}_1, \\ \widehat{\Sigma} &\xrightarrow{p} \Sigma. \end{aligned} \tag{A.1}$$

Since Σ is a constant matrix, it can be inferred from Slutsky's theorem that

$$\boldsymbol{\hat{\mu}_1'\widehat{\Sigma}^{-1}} \xrightarrow{d} \boldsymbol{\mu_1'\Sigma^{-1}}. \tag{A.2}$$

Notice that $\boldsymbol{x}$ is a constant vector, then

$$\boldsymbol{\hat{\mu}_1'\widehat{\Sigma}^{-1}x} \xrightarrow{d} \boldsymbol{\mu_1'\Sigma^{-1}x}.$$

From Eq. (A.1) and Eq. (A.2), applying Slutsky's theorem again, we have

$$\boldsymbol{\hat{\mu}_1'\widehat{\Sigma}^{-1}\hat{\mu}_1} \xrightarrow{d} \boldsymbol{\mu_1'\Sigma^{-1}\mu_1}.$$

Moreover, both $\boldsymbol{\mu_1'\Sigma^{-1}x}$ and $\boldsymbol{\mu_1'\Sigma^{-1}\mu_1}$ are constants. Therefore,

$$\hat{\boldsymbol{\mu}}_1'\widehat{\Sigma}^{-1}\boldsymbol{x} \xrightarrow{p} \boldsymbol{\mu}_1'\Sigma^{-1}\boldsymbol{x}, \tag{A.3}$$

$$\hat{\boldsymbol{\mu}}_1'\widehat{\Sigma}^{-1}\hat{\boldsymbol{\mu}}_1 \xrightarrow{p} \boldsymbol{\mu}_1'\Sigma^{-1}\boldsymbol{\mu}_1. \tag{A.4}$$

Next, r_1 can be estimated by $\hat{r}_1$, the ratio of the number of samples belonging to the first class to the total number of samples. Hence, applying the Weak Law of Large Number, we have

$$\ln \hat{r}_1 \xrightarrow{p} \ln r_1. \tag{A.5}$$

Finally, from (A.3), (A.4) and (A.5), we conclude that

$$\boldsymbol{\hat{\mu}_1'\widehat{\Sigma}^{-1}x - \frac{1}{2}\hat{\mu}_1'\widehat{\Sigma}^{-1}\hat{\mu}_1 + \ln \hat{r}_1} \xrightarrow{p} \boldsymbol{\mu_1'\Sigma^{-1}x - \frac{1}{2}\mu_1'\Sigma^{-1}\mu_1 + \ln r_1},$$

or

$$\hat{d}_1(\boldsymbol{x}) \xrightarrow{p} d_1(\boldsymbol{x}).$$

Appendix B: Supplementary Materials For Chapter 4

B.1 Proof Of Theorem 4.5

Proof. The log-likelihood of the data is

$$l = \sum_{g=1}^{G}\sum_{j=1}^{n_g} \log f(\boldsymbol{u}_j^{(g)}) \tag{B.1}$$

$$= C - \frac{1}{2}\sum_{g=1}^{G} n_g \log|\Delta| - \frac{1}{2}\sum_{g=1}^{G}\sum_{j=1}^{n_g} \delta_j^{(g)}, \tag{B.2}$$

where

$$\delta_j^{(g)} = (\boldsymbol{u}_j^{(g)} - \boldsymbol{\tau}^{(g)})'\Delta^{-1}(\boldsymbol{u}_j^{(g)} - \boldsymbol{\tau}^{(g)}). \tag{B.3}$$

We have

$$\begin{aligned}\frac{\partial \delta_j^{(g)}}{\partial \boldsymbol{\tau}^{(g)}} &= \frac{\partial \delta_j^{(g)}}{\partial(\boldsymbol{u}_j^{(g)} - \boldsymbol{\tau}^{(g)})}\frac{\partial(\boldsymbol{u}_j^{(g)} - \boldsymbol{\tau}^{(g)})}{\partial \boldsymbol{\tau}^{(g)}} \\ &= \Delta^{-1}(\boldsymbol{u}_j^{(g)} - \boldsymbol{\tau}^{(g)}).\end{aligned}$$

Hence,

$$\begin{aligned}\frac{\partial l}{\partial \boldsymbol{\mu}_j^{(g)}} &\Leftrightarrow \sum_{g=1}^{G}\sum_{j=1}^{n_g}(\boldsymbol{u}_j^{(g)} - \boldsymbol{\tau}^{(g)}) = 0 \\ &\Leftrightarrow \hat{\boldsymbol{\tau}}^{(g)} = \bar{\mathbf{u}}^{(g)}.\end{aligned}$$

To find $\hat{\Delta}$, note that

$$\frac{\partial \log|\Delta|}{\partial \Delta^{-1}} = -\frac{\partial \log|\Delta^{-1}|}{\partial \Delta^{-1}} = -\Delta. \tag{B.4}$$

Next,

$$\delta_j^{(g)} = tr(\Delta^{-1}(\boldsymbol{u}_j^{(g)} - \boldsymbol{\tau}^{(g)})(\boldsymbol{u}_j^{(g)} - \boldsymbol{\tau}^{(g)})'). \tag{B.5}$$

Therefore,

$$\frac{\partial \delta_j^{(g)}}{\partial \Delta^{-1}} = (\boldsymbol{u}_j^{(g)} - \boldsymbol{\tau}^{(g)})(\boldsymbol{u}_j^{(g)} - \boldsymbol{\tau}^{(g)})'. \tag{B.6}$$

This implies

$$\frac{\partial l}{\partial \Delta^{-1}} = 0 \Leftrightarrow \sum_{g=1}^{G}\sum_{j=1}^{n_g}(\boldsymbol{u}_j^{(g)} - \boldsymbol{\tau}^{(g)})(\boldsymbol{u}_j^{(g)} - \boldsymbol{\tau}^{(g)})' = 0 \tag{B.7}$$

$$\Rightarrow \hat{\Delta} = \frac{1}{\sum_{g=1}^{G} n_g}\sum_{g=1}^{G}\sum_{j=1}^{n_g}(\boldsymbol{u}_j^{(g)} - \boldsymbol{\tau}^{(g)})(\boldsymbol{u}_j^{(g)} - \boldsymbol{\tau}^{(g)})'. \tag{B.8}$$

B.2 Proof Of Theorem 4.3

Let $\boldsymbol{u}_j^{(g)}$ be the j^{th} column of $\boldsymbol{x}^{(g)}$ and use $f(\boldsymbol{z})$ to denote the joint density of the available entries in a random vector $\boldsymbol{z}$.

Maximizing the likelihood w.r.t $\hat{\mu}_1^{(g)}, \sigma_{11}$

Note that for $j = 1, 2, ..., m_g$

$$f(\boldsymbol{u}_j^{(g)}) = f(\boldsymbol{x}_{1j}^{(g)}, \boldsymbol{x}_{2j}^{(g)}) = f(\boldsymbol{x}_{1j}^{(g)}) f(\boldsymbol{x}_{2j}^{(g)} | \boldsymbol{x}_{1j}^{(g)}), \tag{B.1}$$

Hence, the likelihood function is

$$L = \prod_{g=1}^{G}\left[\left(\prod_{j=1}^{m_g} f(\boldsymbol{x}_{1j}^{(g)}) f(\boldsymbol{x}_{2j}^{(g)} | \boldsymbol{x}_{1j}^{(g)})\right)\left(\prod_{j=m_g+1}^{n_g} f(\boldsymbol{x}_{1j}^{(g)})\right)\left(\prod_{j=n_g+1}^{l_g} f(\boldsymbol{x}_{2j}^{(g)})\right)\right], \tag{B.2}$$

which can be rewrite as

$$L = \prod_{g=1}^{G}\left[\left(\prod_{j=1}^{m_g} f(\boldsymbol{x}_{2j}^{(g)} | \boldsymbol{x}_{1j}^{(g)})\right)\left(\prod_{j=1}^{n_g} f(\boldsymbol{x}_{1j}^{(g)})\right)\left(\prod_{j=n_g+1}^{l_g} f(\boldsymbol{x}_{2j}^{(g)})\right)\right]. \tag{B.3}$$

Hence, maximizing L w.r.t $\mu_1^{(g)}, \sigma_{11}$ is equivalent to maximizing $\prod_{g=1}^{G}\prod_{j=1}^{n_g} f(\boldsymbol{x}_{1j}^{(g)})$ w.r.t $\mu_1^{(g)}, \sigma_{11}$. This problem is similar to the problem of finding the MLEs in lemma 4.5. Therefore, by applying 4.5, the MLEs for $\mu_1^{(g)}, \sigma_{11}$ are

$$\hat{\mu}_1^{(g)} = \frac{1}{n_g}\sum_{j=1}^{n_g} x_{1j}^{(g)}, \tag{B.4}$$

$$\hat{\sigma}_{11}^{(g)} = \frac{\sum_{j=1}^{n_g}\sum_{g=1}^{G}(x_{1j}^{(g)} - \hat{\mu}_1^{(g)})^2}{\sum_{g=1}^{G} n_g}. \tag{B.5}$$

Maximizing the likelihood w.r.t $\hat{\mu}_2^{(g)}, \sigma_{22}$

For $j = 1, 2, ..., m_g$,

$$f(\boldsymbol{u}_j^{(g)}) = f(\boldsymbol{x}_{1j}^{(g)}, \boldsymbol{x}_{2j}^{(g)}) = f(\boldsymbol{x}_{2j}^{(g)}) f(\boldsymbol{x}_{1j}^{(g)} | \boldsymbol{x}_{2j}^{(g)}). \tag{B.6}$$

Hence, the likelihood function is

$$L = \prod_{g=1}^{G} \left[\left(\prod_{j=1}^{m_g} f(\boldsymbol{x}_{2j}^{(g)}) f(\boldsymbol{x}_{1j}^{(g)} | \boldsymbol{x}_{2j}^{(g)}) \right) \left(\prod_{j=m_g+1}^{n_g} f(\boldsymbol{x}_{1j}^{(g)}) \right) \left(\prod_{j=n_g+1}^{l_g} f(\boldsymbol{x}_{2j}^{(g)}) \right) \right]. \tag{B.7}$$

Hence, maximizing L w.r.t $\mu_2^{(g)}, \sigma_{22}$ is equivalent to maximizing

$$\prod_{g=1}^{G} \left[\left(\prod_{j=1}^{m_g} f(\boldsymbol{x}_{2j}^{(g)}) \right) \left(\prod_{j=n_g+1}^{l_g} f(\boldsymbol{x}_{2j}^{(g)}) \right) \right] \tag{B.8}$$

w.r.t $\mu_2^{(g)}, \sigma_{22}$. This problem is similar to the problem of finding the MLEs in lemma 4.5.

Therefore, by applying 4.5, the MLEs for $\mu_2^{(g)}, \sigma_{22}$ are

$$\hat{\mu}_2^{(g)} = \frac{\sum_{j=1}^{m_g} x_{2j}^{(g)} + \sum_{j=n_g+1}^{l_g} x_{2j}^{(g)}}{m_g + l_g - n_g},$$
$$\hat{\sigma}_{22}^{(g)} = \frac{\sum_{g=1}^{G} [\sum_{j=1}^{m_g} (x_{2j}^{(g)} - \hat{\mu}_2^{(g)})^2 + \sum_{j=n_g+1}^{l_g} (x_{2j}^{(g)} - \hat{\mu}_2^{(g)})^2]}{\sum_{g=1}^{G} (m_g + l_g - n_g)}.$$

Maximizing the likelihood function with respect to σ_{12}**.**

Maximizing $\log L$ w.r.t σ_{12} is equivalent to maximizing

$$\eta = \sum_{g=1}^{G} \sum_{j=1}^{m_g} \log f(x_{2j}^{(g)} | x_{1j}^{(g)}) \tag{B.9}$$

w.r.t σ_{12}.

Note that $x_{2j}^{(g)} | x_{1j}^{(g)}$ follows normal distribution with mean

$$\hat{\mu}_2^{(g)} + \frac{\sigma_{12}}{\sigma_{11}} (x_{1j}^{(g)} - \hat{\mu}_1^{(g)}) \tag{B.10}$$

and variance

$$\sigma_{22} - \frac{\sigma_{12}^2}{\sigma_{11}}. \tag{B.11}$$

Therefore,

$$\eta = C - \frac{1}{2}\sum_{g=1}^{G} m_g \log\left(\sigma_{22} - \frac{\sigma_{12}^2}{\sigma_{11}}\right) - \frac{1}{2}\left(\sum_{g=1}^{G}\sum_{j=1}^{m_g}\delta_{gj}^2\right)\left(\sigma_{22} - \frac{\sigma_{12}^2}{\sigma_{11}}\right)^{-1}, \tag{B.12}$$

where

$$\delta_{gj} = x_{2j}^{(g)} - \hat{\mu}_2^{(g)} - \frac{\sigma_{12}}{\sigma_{11}}(x_{1j}^{(g)} - \hat{\mu}_1^{(g)}). \tag{B.13}$$

Note that

$$\delta_{gj}^2 = ((x_{2j}^{(g)} - \hat{\mu}_2^{(g)}) - \frac{\sigma_{12}}{\sigma_{11}}(x_{1j}^{(g)} - \hat{\mu}_1^{(g)}))^2 \tag{B.14}$$

$$= (x_{2j}^{(g)} - \hat{\mu}_2^{(g)})^2 - 2(x_{2j}^{(g)} - \hat{\mu}_2^{(g)})\frac{\sigma_{12}}{\sigma_{11}}(x_{1j}^{(g)} - \hat{\mu}_1^{(g)}) + \frac{\sigma_{12}^2}{\sigma_{11}^2}(x_{1j}^{(g)} - \hat{\mu}_1^{(g)})^2. \tag{B.15}$$

Therefore

$$\sum_{g=1}^{G}\sum_{j=1}^{m_g}\delta_{gj}^2 = s_{22} - 2\frac{\sigma_{12}}{\sigma_{11}}s_{12} + \frac{\sigma_{12}^2}{\sigma_{11}^2}s_{11}. \tag{B.16}$$

Hence, $\hat{\sigma}_{12}$ is the maximizer of

$$\eta(\sigma_{12}) = C - \frac{A}{2}\log\left(\sigma_{22} - \frac{\sigma_{12}^2}{\sigma_{11}}\right) - \frac{s_{22} - 2\frac{\sigma_{12}}{\sigma_{11}}s_{12} + \frac{\sigma_{12}^2}{\sigma_{11}^2}s_{11}}{2\left(\sigma_{22} - \frac{\sigma_{12}^2}{\sigma_{11}}\right)}. \tag{B.17}$$

B.3 Proof Of Theorem 4.4

In the following proof, when a real-valued function $h(x)$ has two one-sided limits $\lim_{x\to-\sqrt{a}^+} h(x)$, $\lim_{x\to\sqrt{a}^-} h(x)$ that coincide, we simultaneously refer to them as

$$\lim_{x^2\to a^-} h(x) = \lim_{x\to-\sqrt{a}^+} h(x) = \lim_{x\to\sqrt{a}^-} h(x).$$

Provided that the left-handed limits are equal to their corresponding right-handed limits, and handedness of the limits are preserved with our relevant calculation, a true result one one-sided limit is also true for another. The use of the notation $\lim_{x^2\to a^-} h(x)$ is then to avoid redundancy.

The proof attempts to show that the global maximum point of η lies inside the domain, hence is a solution to $\dfrac{d\eta}{d\sigma_{12}} = 0$. It should be reminded that in our problem, $\sigma_{12} \in (-\sqrt{\sigma_{11}\sigma_{22}}, \sqrt{\sigma_{11}\sigma_{22}})$, $A \geq 0$, $\sigma_{11} > 0$, $\sigma_{22} > 0$ and

$$-\frac{s_{22}\sigma_{11} + s_{11}\sigma_{22}}{2\sqrt{\sigma_{11}\sigma_{22}}} \neq s_{12} \neq \frac{s_{22}\sigma_{11} + s_{11}\sigma_{22}}{2\sqrt{\sigma_{11}\sigma_{22}}}. \tag{B.1}$$

For $1 \leq g \leq G$, $1 \leq j \leq m_g$, we define

$$\begin{aligned}
a_{g,j} &= \frac{\sigma_{12}}{\sigma_{11}}(x_{1j}^{(g)} - \hat{\mu}_1^{(g)}), \\
b_{g,j} &= (x_{2j}^{(g)} - \hat{\mu}_2^{(g)}), \\
q &= s_{22} - 2\frac{\sigma_{12}}{\sigma_{11}}s_{12} + \frac{\sigma_{12}^2}{\sigma_{11}^2}s_{11}, \\
\phi(\sigma_{12}) &= -\frac{A}{2}\log\left(\sigma_{22} - \frac{\sigma_{12}^2}{\sigma_{11}}\right), \\
\psi(\sigma_{12}) &= \frac{s_{22} - 2\frac{\sigma_{12}}{\sigma_{11}}s_{12} + \frac{\sigma_{12}^2}{\sigma_{11}^2}s_{11}}{2\left(\sigma_{22} - \frac{\sigma_{12}^2}{\sigma_{11}}\right)}.
\end{aligned}$$

Proof.

Note that $a_{g,j}^2 + b_{g,j}^2 - 2a_{g,j}b_{g,j} \geq 0$ for every pair of (g, j). By summing across all pair of indices, it follows that

$$q = \sum_{g=1}^{G}\sum_{j=1}^{m_g}(a_{g,j} - b_{g,j})^2 \geq 0$$

From condition B.1,

$$\sigma_{11}q(-\sqrt{\sigma_{11}\sigma_{22}}) = s_{22}\sigma_{11} + s_{11}\sigma_{22} + 2\sqrt{\sigma_{11}\sigma_{22}}s_{12} > 0,$$

$$\sigma_{11}q(\sqrt{\sigma_{11}\sigma_{22}}) = s_{22}\sigma_{11} + s_{11}\sigma_{22} - 2\sqrt{\sigma_{11}\sigma_{22}}s_{12} > 0.$$

As a consequence,

$$\begin{aligned}
\lim_{\sigma_{12}\to-\sqrt{\sigma_{11}\sigma_{22}}^+}\psi(\sigma_{12}) &= \frac{q(-\sqrt{\sigma_{11}\sigma_{22}})}{0^+} = \infty, \\
\lim_{\sigma_{12}\to\sqrt{\sigma_{11}\sigma_{22}}^-}\psi(\sigma_{12}) &= \frac{q(\sqrt{\sigma_{11}\sigma_{22}})}{0^+} = \infty.
\end{aligned}$$

In our notation, that is $\lim_{\sigma_{12}^2 \to \sigma_{11}\sigma_{22}^-} \psi(\sigma_{12}) = \infty$.

Case of $A > 0$:

It can be seen that

$$\lim_{\sigma_{12}^2 \to \sigma_{11}\sigma_{22}^-} \phi(\sigma_{12}) = \infty.$$

Moreover,

$$\begin{aligned}
\lim_{\sigma_{12} \to -\sqrt{\sigma_{11}\sigma_{22}}^+} \frac{-\psi(\sigma_{12})}{\phi(\sigma_{12})} &= \lim_{\sigma_{12} \to -\sqrt{\sigma_{11}\sigma_{22}}^+} \frac{s_{22}\sigma_{11}\sigma_{12} - s_{12}\sigma_{12}^2 - (s_{12}\sigma_{11} - s_{11}\sigma_{12})\sigma_{22}}{A\sigma_{12}(\sigma_{12}^2 - \sigma_{11}\sigma_{22})} \\
&= \frac{-\sqrt{\sigma_{11}\sigma_{22}}\left(2\, s_{12}\sqrt{\sigma_{11}\sigma_{22}} + s_{22}\sigma_{11} + s_{11}\sigma_{22}\right)}{0^+} = -\infty, \\
\lim_{\sigma_{12} \to \sqrt{\sigma_{11}\sigma_{22}}^-} \frac{-\psi(\sigma_{12})}{\phi(\sigma_{12})} &= \lim_{\sigma_{12} \to \sqrt{\sigma_{11}\sigma_{22}}^-} \frac{s_{22}\sigma_{11}\sigma_{12} - s_{12}\sigma_{12}^2 - (s_{12}\sigma_{11} - s_{11}\sigma_{12})\sigma_{22}}{A\sigma_{12}(\sigma_{12}^2 - \sigma_{11}\sigma_{22})} \\
&= \frac{\sqrt{\sigma_{11}\sigma_{22}}\left(-2\, s_{12}\sqrt{\sigma_{11}\sigma_{22}} + s_{22}\sigma_{11} + s_{11}\sigma_{22}\right)}{0^-} = -\infty.
\end{aligned}$$

It follows that

$$\begin{aligned}
\lim_{\sigma_{12}^2 \to \sigma_{11}\sigma_{22}^-} \eta(\sigma_{12}) &= C + \lim_{\sigma_{12}^2 \to \sigma_{11}\sigma_{22}^-} \phi(\sigma_{12}) - \psi(\sigma_{12}) \\
&= C + \lim_{\sigma_{12}^2 \to \sigma_{11}\sigma_{22}^-} \left(1 + \frac{-\psi(\sigma_{12})}{\phi(\sigma_{12})}\right) \lim_{\sigma_{12}^2 \to \sigma_{11}\sigma_{22}^-} \phi(\sigma_{12}) \\
&= C + (-\infty)\infty = -\infty.
\end{aligned}$$

It can be derived that

$$\eta'(\sigma_{12}) = \frac{s_{12}\sigma_{11}\sigma_{22} + (\sigma_{11}\sigma_{22}A - s_{22}\sigma_{11} - s_{11}\sigma_{22})\sigma_{12} + s_{12}\sigma_{12}^2 - A\sigma_{12}^3}{\left(\sigma_{12}^2 - \sigma_{11}\sigma_{22}\right)^2}.$$

Since $\lim_{\sigma_{12}^2 \to \sigma_{11}\sigma_{22}^-} \eta(\sigma_{12}) = -\infty$ and η is twice differentiable over its domain, it has a global maximum point in the domain, which is a real root of

$$s_{12}\sigma_{11}\sigma_{22} + (\sigma_{11}\sigma_{22}A - s_{22}\sigma_{11} - s_{11}\sigma_{22})\sigma_{12} + s_{12}\sigma_{12}^2 - A\sigma_{12}^3. \tag{B.2}$$

Case of $A = 0$:

The function becomes $\eta = C - \psi$. Consequently, $\lim_{\sigma_{12}^2 \to \sigma_{11}\sigma_{22}^-} \eta(\sigma_{12}) = -\infty$. The derivative is now

$$\eta' = \frac{s_{12}\sigma_{11}\sigma_{22} - (s_{22}\sigma_{11} + s_{11}\sigma_{22})\sigma_{12} + s_{12}\sigma_{12}^2}{(\sigma_{12}^2 - \sigma_{11}\sigma_{22})^2}.$$

With the same reasoning as the last paragraph, η has a global maximum point in the domain that is a root of B.2.

Appendix C: Python Codes

C.1 Python Codes For Algorithm 3.3.2

```
def EPEM(Xtrain, n, p, G):

'''

Xtrain: list of input. The ith element of the list contains the sample from

the ith class.

'''

if p[0]==1:

# the array that contains the means of each block for the 1st block

mus = [np.mean(Xtrain[g][:,0]) for g in np.arange(G)]

S = [(n[g,0]-1)*np.var(Xtrain[g][:,0]) for g in np.arange(G)]

else:

mus = [np.mean(Xtrain[g][:,0:p[0]], axis = 0) for g  in np.arange(G)]

S = [(n[g,0]-1)*np.cov(Xtrain[g][:,0:p[0]],rowvar =False)

for g in np.arange(G)]


mus = np.asarray(mus).T # so that each column is the mean of a class

S = sum(S)/(sum(n[:,0]))

S = S.reshape((p[0],-1))

for i in np.arange(1,len(p)):

W = [(n[g,i]-1)*np.cov(Xtrain[g][0:n[g,i],0:p[i]],

rowvar=False) for g in np.arange(G)]

W = sum(W)
```

```
P = np.matmul(W[(p[i-1]):p[i], 0:p[i-1]],
np.linalg.inv(W[0:p[i-1],0:p[i-1]]))
Q = (W[p[i-1]:p[i],p[i-1]:p[i]]-
np.matmul(P, W[0:p[i-1],p[i-1]:p[i]]))/sum(n[:,i])
xmeans = [np.mean(Xtrain[g][0:n[g,i],0:p[i]], axis = 0)
for g in np.arange(G)]


xmeans = np.asarray(xmeans)
xmeans = xmeans.T
mus = np.vstack((mus, xmeans[p[i-1]:p[i],:]
- np.matmul(P, xmeans[0:p[i-1]]-mus)))
S21 = np.matmul(P, S)
S = np.vstack((np.hstack((S, S21.T)),
np.hstack((S21, Q+np.matmul(P, S21.T)))))
return [mus, S]
```

C.2 Python Codes For Parameter Estimation Assuming Equal Covariance Matrix

```
def mle(X,y,G):
'''
X: input, should be a numpy array
y: label
G: number of classes
output:
- mus: each row is a class mean
- S: common covariance matrix of class 1,2,..., G
'''
epsilon = 1e-5 # define epsilon to put r down to 0 if r < epsilon
n,p = X.shape[0], X.shape[1]

# Estimating class means
# each column is the mean of a class
mus = np.array([np.nanmean(X[y==g,:],axis=0) for g in range (G)]).T

S = np.diag([diag_term(i,X,y) for i in range(p)])

for i in range(p):
for j in range(i):
mat = X[:,[i,j]]
```

```
# drop rows with NA

idx = ~np.isnan(mat).any(axis=1)

mat, y_arr = mat[idx], y[idx]


_, counts = np.unique(y_arr, return_counts=True)

ind = np.insert(np.cumsum(counts), 0, 0)


m_g = counts


A = len(y_arr)

scaled_mat = [mat[ind[g-1]:ind[g],:]-mus[[i,j],g-1] for g in range(1,G+1)]


q = lambda g: np.dot(scaled_mat[g][:,0],scaled_mat[g][:,0])

s11 = sum(map(q,range(G)))

q = lambda g: np.dot(scaled_mat[g][:,1],scaled_mat[g][:,1])

s22 = sum(map(q,range(G)))

d = lambda g: np.dot(scaled_mat[g][:,0],scaled_mat[g][:,1])

s12 = sum(map(d,range(G)))


start_solve = time.time()

B = S[i,i]*S[j,j]*A - s22 * S[i,i] - s11 * S[j,j]

coefficient = [-A, s12, B, s12*S[i,i]*S[j,j]]
```

```
r = np.roots(coefficient)


r = r[abs(np.imag(r)) < epsilon]

r = np.real(r)

r[abs(r) < epsilon] = 0


if len(r)>1:

condi_var = S[j,j] - r**2/S[i,i]

eta = -A*np.log(condi_var)-(S[j,j]-2*r/S[i,i]*s12 +

r**2/S[i,i]**2*s11)/condi_var

# if condi_var <0 then eta = NA.

# in practice, it's impossible for cov to be negative

#  therefore, we drop NA elements of eta

r = r[eta == max(eta[~np.isnan(eta)])]


if len(r) > 1:

w = [m_g[g-1]*np.cov(mat[ind[g-1]:ind[g],], rowvar=False) for

g in range(1,G+1)]

w = np.sum(w, axis = 0)

r = r[np.abs(r-w[0,1]).argmin()] # select r that is closet to w[0,1]


S[i,j] = S[j,i] = r

return [mus, S]
```

```
def diag_term(i,X,y):

arr0 = X[:,i]

nar2 = 0

arr = arr0[~np.isnan(arr0)]

y_arr = y[~np.isnan(arr0)]


_, counts = np.unique(y_arr, return_counts=True)

ind = np.insert(np.cumsum(counts), 0, 0)


return sum([(ind[g]-ind[g-1])*np.var(arr[ind[g-1]:ind[g]]) for

g in range(1,G+1)])/len(y_arr)
```

C.3 Python Codes For Algorithm Parameter Estimation Without Assuming Equal Covariance Matrix

```
def diag_term(X,i):

arr0 = X[:,i].flatten()

arr = arr0[~np.isnan(arr0)]

return np.var(arr)


def musMLE(X,y,G):

n,p = X.shape[0], X.shape[1]

f = lambda g: np.nanmean(X[y==g,:],axis=0)

musMLE = np.array([f(g) for g in range(G)])

return musMLE.T


def Smle(X,y,musMLE,g):

'''

function to compute the covariance matrix for the g-th class

X: input, should be a numpy array

y: label

G: number of classes

g: class index

output:

- mus: each row is a class mean

- S: common covariance matrix of class 1,2,..., G
```

```
,,,

epsilon = 1e-5 # define epsilon to put r down to 0 if r < epsilon

Xg, yg = X[y==g,:], y[y==g]

n,p = Xg.shape[0], Xg.shape[1]


S = np.diag([diag_term(Xg,i) for i in range(p)])


for i in range(p):

for j in range(i):

if ((S[i,i] == 0.) | (S[j,j] == 0.)):

S[i,j] = S[j,i] = 0.

continue


mat = Xg[:,[i,j]]


# drop rows with NA

idx = ~np.isnan(mat).any(axis=1)

mat, y_arr = mat[idx], yg[idx]

A = mg = len(y_arr)


s11 = mg*np.var(mat[:,0])

s22 = mg*np.var(mat[:,1])

s12 = sum((mat[:,0]-musMLE[i,g])*(mat[:,1]-musMLE[j,g]))
```

```
B = S[i,i]*S[j,j]*A - s22 * S[i,i] - s11 * S[j,j]

coefficient = [-A, s12, B, s12*S[i,i]*S[j,j]]

r = np.roots(coefficient)

r = r[abs(np.imag(r)) < epsilon]

r = np.real(r)

r[abs(r) < epsilon] = 0


if len(r)>1:

condi_var = S[j,j] - r**2/S[i,i]

eta = -A*np.log(condi_var)-(S[j,j]-2*r/S[i,i]*s12 + r**2/S[i,i]**2*s11)/condi_var

r = r[eta == max(eta[~np.isnan(eta)])]


if len(r) > 1:

if sum(r==0.0) == len(r):

r = 0.

else:

w = np.cov(mat, rowvar=False)

#r = r[w[0,1]*r>=0]

r = r[np.abs(r-w[0,1]).argmin()] # select r that is closet to w[0,1]


S[i,j] = S[j,i] = r

return S
```

Nguyen, Thu Thi. Bachelor of Science, University of Science, Ho Chi Minh city, 2015; Master of Science, University of Louisiana at Lafayette, 2018; Doctor of Philosophy, University of Louisiana at Lafayette, 2021.

Major: Mathematics

Title of Dissertation: Optimization of Maximum Likelihood Function in the Presence of Missing Data

Dissertation Director: Dr. Bruce A. Wade

Pages in Dissertation: 87; Words in Abstract: 201

Abstract

The problem of monotone missing data has been broadly studied during the last two decades and has many applications in various fields such as bioinformatics or statistics. Commonly used imputation techniques for missing data problems usually require multiple iterations through the data before yielding convergence. Moreover, those approaches may introduce extra noises and biases to the subsequent modeling. In this work, we derive exact formulas and propose a novel algorithms to compute the maximum likelihood estimators (MLEs) of a dataset directly. Specifically, in chapter 3, we disscuss about estimating parameters of a multiple class, monotone missing dataset when all the covariance matrices of all categories are assumed to be equal, namely Efficient Parameter Estimation for Multiple Class Monotone Missing Data (EPEM). Next, in chapter 4, we extend the idea of chapter 3, with some modification to provide direct parameter estimation schemes for single-/multiple- class missing data.

As the computation is exact, our algorithms does not require multiple iterations through the data as other imputation approaches, thus promising to handle much less time-consuming than other methods. This effectiveness was validated by empirical results when our algorithms reduced the error rates significantly and required a short computation time compared to several imputation-based approaches.

Biographical Sketch

Thu Thi Nguyen received her Bachelor degree in mathematics and computer science in 2015 from the University of Science, Ho Chi Minh city, Vietnam. She then pursued her graduate studies at University of Louisiana at Lafayette in fall 2016. Along the way, Thu Thi Nguyen was granted a Masters in mathematics in the fall of 2018. She completed the requirements for a Doctor of Philosophy in mathematics from the University of Louisiana at Lafayette in Summer, 2021.