ABSTRACT

HAN, XU. Majorization-Minimization Techniques and Applications in Optimization and Statistical Learning. (Under the direction of Dr. Eric Chi).

In this dissertation, we discuss the Majorization-Minimization algorithm and explain how it improves the current optimization methods by showing three examples. Optimization has broad applications in nowaday's Statistical Learning. Many famous algorithms are indeed solving an optimization problem, or some part of them consists of an optimization step. However, the traditional optimization methods usually work on the true objective function and develop some gradient-based methods, which leads to complicated iterative procedures and high computational costs, especially when the objective function is complex. The Majorization-Minimization algorithm is a clever idea that simplifies the problem by converting the complex objective function into a sequence of more straightforward problems. People hence iteratively solve the simpler problems to approach the optimum of the original objective function. We present three applications in different fields where the Majorization-Minimization technique helps simplify the most challenging part of the objective functions. In clustering, we develop a new method for clustering histogram valued data. We majorize the L_1 penalty by a quadratic approximation, which produces an inverse problem during the iterations. In the constrained binomial regression problem, we also construct a quadratic majorization of the log-likelihood and convert the problem into a sequence of inverse problems. In the 1-bit matrix completion problem, we construct the majorization of the loss function and use the Gauss-Newton method to search for a descending direction. We hence develop new methods for solving these problems. The synthetic studies and real data applications show that the newly proposed methods based on the Majorization-Minimization achieve better performances and faster efficiency.

Majorization-Minimization Techniques and Applications in Optimization and Statistical Learning

by
Xu Han

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Statistics

Raleigh, North Carolina

2021

APPROVED BY:

Arvind Saibaba	Minh Tang
Jessie Jeng	Eric Chi Chair of Advisory Committee

BIOGRAPHY

The author was born and grew up in Tianjin, China. He earned a Bachelor of Science degree from the Department of Mathematics, Nankai University, in 2016 and a Master of Science degree from the Department of Statistics, the University of Wisconsin-Madison, in 2017. He then entered the Ph.D. program of Statistics at North Carolina State University and worked under the direction of Dr. Eric Chi.

ACKNOWLEDGEMENTS

First of all, I would like to extend my sincere gratitude to my supervisor, Dr. Eric Chi, for his instructive advice and valuable suggestions on my thesis. I am deeply grateful for his help with this thesis. I still remember the first day I went to his office asked if he was open to having more students. He said yes, and that started the four years of collaborations between us. He keeps being energetic, inspired and providing a lot of assistance to me every time we meet or discuss our research. My thesis cannot be completed without his help. No one will say a Ph.D. is easy, but only after taking one can I realize how on earth it feels like. He put so much confidence and patience in me. I gradually learned how to think independently, study actively and develop the abilities to become a good researcher.

Second, I would like to express my heartfelt gratitude to Dr. Saibaba, Dr. Jeng, Dr. Tang. I am so grateful to have them in my committee. They provided helpful suggestions on my research work from different perspectives and made me see my deficiency. I learned a lot from their lectures and also personal meetings. I believe these precious suggestions will benefit me for a long time, even after I graduate. I also would like to thank all my friends and colleagues I met at NCSU, especially Weilian Zhou, Min Zhang, Xiaoqian Liu, Bowen Liu, Yukun Song, and Alex Chen. They offered me a lot of help and company.

Last, my thanks would go to my beloved family for their loving considerations and great confidence in me all through these years. I may not go through those difficult moments without their encourages.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER

1

INTRODUCTIONS: OPTIMIZATION, MAJORIZATION AND NOTATIONS

Optimization is always a popular scientific topic since it has broad applications, including estimation, clustering, classification, etc. In general, there is an objective function representing some properties of the problem with user-defined constraints. The corresponding optimum gives a result of the problem. Many well-known algorithms are indeed doing an optimization, or some part consists of the optimization step. Take the most straightforward simple linear regression as an example. We search for the best estimator that produces the least mean square error. The generalized linear model does a similar thing by replacing the standard mean square error with the negative log-likelihood as the loss function. In the clustering field, the very famous k-means [Mac67] iteratively minimizes the distances of the data points to their assumed centers. Another algorithm, convex clustering [Pel05], minimizes the discrepancy of the estimation and observation with a convex regularization penalizes the pairwise difference among the observations to generate the clustering assignment. As for classification, Linear Discriminant Analysis (LDA) [Fis36; McL04] maximizes the distance of means among different groups and minimizes the within-group variances simultaneously. The decision tree [Qui86; Qui14], another typical classification method, selects the feature that maximizes the information gain (ratio) after splitting on each level of the tree. In some more modern areas, such as deep learning and reinforcement learning, the parameters of the model are also decided by maximizing the likelihood or the expected gains.

We see the critical role that optimization played in many areas. So whether or not we can solve the optimization step correctly and efficiently becomes a vital issue for a successful algorithm. In

some cases where the problem is simple, we can get the exact solution from mathematical derivations. For example, in simple linear regression, we have the expression of the best estimation, even though it does not have a closed-form to get (pseudo) inverse of the matrix. Furthermore, with L_1 or L_2 penalties, the linear regression becomes either the Lasso [Tib96] or the ridge regression [Hoe70], where we also get the format of solutions. The Karush-Kuhn-Tucker (KKT) conditions [Kar39; Kuh51; Kje00] are common first-order necessary conditions to test optimality in the more complex case. Typical examples are LDA and Support Vector Machine (SVM) [Dru97; Guy93; Bos92], where we derive the solution through analyzing the KKT conditions. It is also a typical application of KKT contributes in the machine learning area. However, in most cases, the objective function will be complex with various constraints. We do not have the closed expression of the optimal solution and have to turn out to some numerical methods. The most conventional way should be the gradient-based methods if the objective function is differentiable, or at least some part is differentiable. The negative gradient will be in a descending direction; hence we can iteratively use it to approach the minimum point. With some other assumptions, such as convexity, the gradient descent [Lem12; Had08; Cou94; Cur44] works well and promises a global minimizer if there exists one. Many other algorithms are developed based on gradient descent, including proximal (projected) gradient descent [Com11; Com05; Mos10]; conditional gradient descent [Fra56; Lev66; Dun78]; stochastic gradient descent [Saa98]. Many methods also include the second-order derivative to fix the potential drawback of gradients such as the Newton's method [Mor82] and Quasi-Newton method [Wri99; Bon06]. Another popular strategy is the Alternative Directional Method of Multipliers (ADMM) [Glo75; Gab76; Boy11], which splits the optimization concerning joint variables into different parts, and only optimizes under a specific variable.

However, all these techniques may be hard to use since they may produce a very complicated updating formula without closed expression, especially when the objective function is complex. In gradient-based methods, the gradient is also hard to compute if the objective function itself is complicated. The user usually spends an amount of time computing the gradient or even the Hessian in second-order methods. Sometimes, we only work on the part of the objective function to simplify the problem. Typical examples are the ADMM and the block-wise coordinate descent, where we still may not get a closed expression along a single direction (coordinate). Therefore, we need to solve a sub-problem using numerical methods, leading to another level of iterations and high computational costs. Here we introduce the principal concept, the Majorization-Minimization (**MM**) technique, of this dissertation. Hunter & Lange [Hun04] proposed the MM technique. The key idea is to replace the original objective function with a simple majorized function and iteratively minimize the majorizations to approach the minimizer of the original problem. Usually, we can manually construct the majorization. So the problem can become very simple and easy to update in each iteration. This dissertation will introduce three applications of the MM technique in clustering (Chapter 2), shape restricted binomial regression (Chapter 3), and 1-bit matrix completion (Chapter 4). We will show how the MM technique helps simplify the corresponding optimization problem and achieves better computational results than the existing methods. In the rest of this chapter, we

first introduce the notations used throughout the dissertation. We then briefly introduce the MM technique and talk about a common strategy to construct the majorization.

1.1 Notations

The notations listed in this section are the common principles throughout the dissertation. We will give additional illustrations if there are special notations that are only used under a specific chapter.

- a: Little non-bold letter, a scalar, real or integer.
- $\mathbb{R}^p$: The p-dimensional vector space. Real line when $p = 1$, i.e., $(-\infty, +\infty)$.
- $[a, b]$: A closed interval from a to b on the real line.
- (a, b): An open interval from a to b on the real line.
- $[m]$: A discrete set $\{1, 2, \ldots, m\}$ with a positive integer m.
- $|a|$ or $|\Omega|$: The absolute value of a real number a. Or the cardinality of a set Ω.
- $\boldsymbol{x}$: Little bold letter, a real-valued vector with specific dimension.
- $\boldsymbol{X}$: Capital bold letter, a real-valued matrix with specific size.
- $\boldsymbol{X}^{-1}$: The inverse of matrix $\boldsymbol{X}$ when $\boldsymbol{X}$ is non-singular.
- $\boldsymbol{X}^{-}$: Moore-Penrose pseudo inverse of matrix $\boldsymbol{X}$.
- vec: Column-wise major vectorization.
- $\otimes$: The Kronecker product.
- $\circ$: The Hadamard product, i.e., element-wise product for two matrices.
- $f : \mathscr{A} \to \mathscr{B}$ The function f from domain $\mathscr{A}$ to range $\mathscr{B}$. In this dissertation, we usually use $f : \mathbb{R}^p \to \mathbb{R}$, a real-valued function from a vector space $\mathbb{R}^p$.
- $\nabla f(\boldsymbol{x})$: Gradient of function f evaluate at a vector $\boldsymbol{x}$ when f is differentiable.
- $f'(x)$: Derivative of function f evaluate at a scalar x when f is differentiable. This is a special case of $\nabla f(\boldsymbol{x})$ when f is a univariate function $\mathbb{R} \to \mathbb{R}$.
- $f'(\boldsymbol{x}\,; \boldsymbol{y})$: The directional derivative of function f at $\boldsymbol{x}$ in the direction of $\boldsymbol{y}$, i.e., $f'(\boldsymbol{x}\,; \boldsymbol{y}) = \lim_{\alpha \downarrow 0} \frac{f(\boldsymbol{x}+\alpha \boldsymbol{y}) - f(\boldsymbol{x})}{\alpha}$.
- $\nabla^2 f(\boldsymbol{x})$: Hessian matrix of function f evaluate at $\boldsymbol{x}$ when f is twice differentiable.
- $\iota_{\texttt{condition}}$: The indicator function, which equals to 0 if the `condition` is satisfied, $+\infty$ otherwise.

- $\|\boldsymbol{x}\|$: L_2 norm of a vector, i.e., $\|\boldsymbol{x}\|^2 = \sum_i x_i^2$.
- $\|\boldsymbol{x}\|_1$: L_1 norm of a vector, i.e., $\|\boldsymbol{x}\|_1 = \sum_i |x_i|$.
- $\|\boldsymbol{X}\|_F$: Frobenius norm of a matrix, i.e., $\|\boldsymbol{X}\|_F^2 = \sum_{i,j} X_{ij}^2$.
- $\|\boldsymbol{X}\|_*$: Nuclear norm of a matrix, i.e., $\|\boldsymbol{X}\|_* = \sum_i \sigma_i(\boldsymbol{X})$, where $\sigma_i(\boldsymbol{X})$ is the singular values of $\boldsymbol{X}$.
- $\|\boldsymbol{X}\|_\infty$: Infinity norm of a matrix, i.e., $\|\boldsymbol{X}\|_\infty = \max_{i,j} |X_{ij}|$.
- $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$: The scalar product of two vectors $\boldsymbol{x}, \boldsymbol{y}$.
- $\mathrm{P}_G(\boldsymbol{x})$: The projection of vector $\boldsymbol{x}$ onto a set G. $\mathrm{P}_G(\boldsymbol{x}) = \arg\min_{\boldsymbol{y} \in G} \frac{1}{2}\|\boldsymbol{x}-\boldsymbol{y}\|^2$.
- $\mathrm{Prox}_f(\boldsymbol{x})$: The proximal operator of a function f at point $\boldsymbol{x}$. $\mathrm{Prox}_f(\boldsymbol{x}) = \arg\min_{\boldsymbol{y}} \frac{1}{2}\|\boldsymbol{x}-\boldsymbol{y}\|^2 + f(\boldsymbol{y})$.

1.2 Majorization-Minimization Method

We briefly review the MM algorithm [Hun04] in this section. The idea behind MM algorithm is to solve a hard optimization problem by converting it into a sequence of simpler problems. Suppose that we want to minimize a complex function f. We first construct a majorization function $g(\boldsymbol{x}|\tilde{\boldsymbol{x}})$ of the objective function $f(\boldsymbol{x})$ at an anchor point $\tilde{\boldsymbol{x}}$. The function $g(\boldsymbol{x}|\tilde{\boldsymbol{x}})$ is called the **majorization** of f if it satisfies the following two conditions:

1. Tangency condition: $g(\tilde{\boldsymbol{x}}|\tilde{\boldsymbol{x}}) = f(\tilde{\boldsymbol{x}})$.
2. Domination condition: $g(\boldsymbol{x}|\tilde{\boldsymbol{x}}) \geq f(\boldsymbol{x})$ for all $\boldsymbol{x}$.

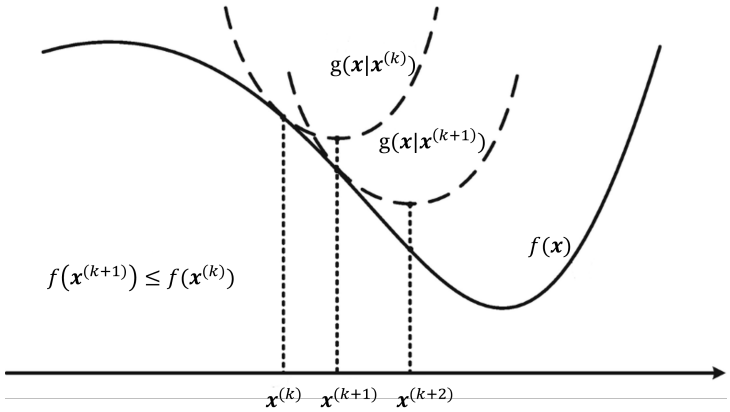


Figure 1.1 General working procedure of the MM algorithm [Scu18].

We show a simple example given by Scutari & Sun [Scu18] in Figure 1.1. Function f is our objective function, which could be quite complex and hard to minimize. Function $g(\boldsymbol{x}|\boldsymbol{x}^{(k)})$ is a majorization of

f at $\boldsymbol{x}^{(k)}$. It is easy to check that $g(\boldsymbol{x}|\boldsymbol{x}^{(k)})$ is always greater or equal than f, and they are equal if and only if $\boldsymbol{x}=\boldsymbol{x}^{(k)}$. Instead of directly minimizing f, in a specific step, we minimize the majorization $g(\boldsymbol{x}|\boldsymbol{x}^{(k)})$ and get its minimum point $\boldsymbol{x}^{(k+1)}$. Then we construct the same majorization but at the new position $\boldsymbol{x}^{(k+1)}$, minimize this new majorization and updated point $\boldsymbol{x}^{(k+2)}$. In summary, the MM algorithm generates a sequence of iterates or new anchor points using the rule $\boldsymbol{x}^{(k+1)} = \arg\min_{\boldsymbol{x}} g(\boldsymbol{x}|\boldsymbol{x}^{(k)})$. By the tangency and domination conditions, the iterates possess the following descent property:

$$f(\boldsymbol{x}^{(k+1)}) \;\leq\; g(\boldsymbol{x}^{(k+1)}|\boldsymbol{x}^{(k)}) \;\leq\; g(\boldsymbol{x}^{(k)}|\boldsymbol{x}^{(k)}) \;=\; f(\boldsymbol{x}^{(k)}).$$

So the sequence $\{\boldsymbol{x}^{(k)}\}_{k=1}^{\infty}$ will always decrease and iteratively approach the minimum point of the original function. In practice, the bottleneck is how do we construct a simple enough majorization of the objective function. We talk about a common strategy under the convex optimization, which is also used in Chapter 4 and Chapter 3.

1.2.1 Convex and Lipschitz Differentiable Function

There is a common way to construct majorization if the function is convex and Lipschitz differentiable. In practice, we usually design a convex objective function to simplify the optimization procedure. Furthermore, if it is also Lipschitz differentiable, we could construct the majorization from the following proposition. We start by giving definitions.

Definition 1 (Convexity)**.** *A function $f:\mathbb{R}^p\to\mathbb{R}$ is called convex if for any $\boldsymbol{x},\boldsymbol{y}\in\mathbb{R}^p$ and a scalar $\alpha\in(0,1)$:*

$$f\left(\alpha\boldsymbol{x}+(1-\alpha)\boldsymbol{y}\right) \;\leq\; \alpha f(\boldsymbol{x})+(1-\alpha)f(\boldsymbol{y}).$$

Definition 2 (Lipschitz differentiable)**.** *A function $f:\mathbb{R}^p\to\mathbb{R}$ is Lipschitz differentiable with Lipschitz constant L if for any $\boldsymbol{x},\boldsymbol{y}\in\mathbb{R}^p$:*

$$\|\nabla f(\boldsymbol{x})-\nabla f(\boldsymbol{y})\| \;\leq\; L\|\boldsymbol{x}-\boldsymbol{y}\|.$$

With these two concepts well-defined, we give the following proposition.

Proposition 1. *Let $f:\mathbb{R}^p\to\mathbb{R}$ be convex and Lipschitz differentiable with Lipschitz constant L. Then for any $\boldsymbol{x},\boldsymbol{y}\in\mathbb{R}^p$:*

$$f(\boldsymbol{y}) \;\leq\; f(\boldsymbol{x})+\langle\boldsymbol{y}-\boldsymbol{x},\nabla f(\boldsymbol{x})\rangle+\frac{L}{2}\|\boldsymbol{y}-\boldsymbol{x}\|^2.$$

Proof. Define a new univariate function $\phi:\mathbb{R}\to\mathbb{R}$

$$\phi(t) \;=\; f\left(\boldsymbol{x}+t(\boldsymbol{y}-\boldsymbol{x})\right) \quad \forall\, t.$$

Then $\forall s \in (0,1)$:

$$\frac{\phi(t+s)-\phi(t)}{s} = \frac{f\left(\boldsymbol{x}+t(\boldsymbol{y}-\boldsymbol{x})+s(\boldsymbol{y}-\boldsymbol{x})\right)-f\left((\boldsymbol{x}+t(\boldsymbol{y}-\boldsymbol{x})\right)}{s}.$$

It is easy to check function ϕ is differentiable. Take limits with $s \downarrow 0$, i.e., take the limit from right hand side, we have:

$$\begin{aligned}
\phi'(t) &= \lim_{s\downarrow 0} \frac{\phi(t+s)-\phi(t)}{s} \\
&= \lim_{s\downarrow 0} \frac{f\left(\boldsymbol{x}+t(\boldsymbol{y}-\boldsymbol{x})+s(\boldsymbol{y}-\boldsymbol{x})\right)-f\left((\boldsymbol{x}+t(\boldsymbol{y}-\boldsymbol{x})\right)}{s} \\
&= f'(\boldsymbol{x}+t(\boldsymbol{y}-\boldsymbol{x})\,;\boldsymbol{y}-\boldsymbol{x}) \qquad (1.1) \\
&= \left\langle \boldsymbol{y}-\boldsymbol{x}, \nabla f\left(\boldsymbol{x}+t(\boldsymbol{y}-\boldsymbol{x})\right)\right\rangle. \qquad (1.2)
\end{aligned}$$

Equation (1.1) is the definition of directional derivative. Equation (1.2) holds following that f is differentiable. Therefore:

$$\begin{aligned}
f(\boldsymbol{y})-f(\boldsymbol{x}) &= \phi(1)-\phi(0) \\
&= \int_0^1 \phi'(t)\,\mathrm{d}t \\
&= \int_0^1 \left\langle \boldsymbol{y}-\boldsymbol{x}, \nabla f\left(\boldsymbol{x}+t(\boldsymbol{y}-\boldsymbol{x})\right)\right\rangle \mathrm{d}t.
\end{aligned}$$

Consequently,

$$\begin{aligned}
f(\boldsymbol{y})-f(\boldsymbol{x})-\langle \boldsymbol{y}-\boldsymbol{x}, \nabla f(\boldsymbol{x})\rangle &= \int_0^1 \left\langle \boldsymbol{y}-\boldsymbol{x}, \nabla f\left(\boldsymbol{x}+t(\boldsymbol{y}-\boldsymbol{x})\right)\right\rangle \mathrm{d}t - \langle \boldsymbol{y}-\boldsymbol{x}, \nabla f(\boldsymbol{x})\rangle \\
&= \int_0^1 \left\langle \boldsymbol{y}-\boldsymbol{x}, \nabla f\left(\boldsymbol{x}+t(\boldsymbol{y}-\boldsymbol{x})\right)\right\rangle \mathrm{d}t - \int_0^1 \langle \boldsymbol{y}-\boldsymbol{x}, \nabla f(\boldsymbol{x})\rangle\,\mathrm{d}t \\
&= \int_0^1 \left\langle \boldsymbol{y}-\boldsymbol{x}, \nabla f\left(\boldsymbol{x}+t(\boldsymbol{y}-\boldsymbol{x})\right)\right\rangle - \langle \boldsymbol{y}-\boldsymbol{x}, \nabla f(\boldsymbol{x})\rangle\,\mathrm{d}t \\
&= \int_0^1 \left\langle \boldsymbol{y}-\boldsymbol{x}, \nabla f\left(\boldsymbol{x}+t(\boldsymbol{y}-\boldsymbol{x})\right)-\nabla f(\boldsymbol{x})\right\rangle - \mathrm{d}t \\
&= \int_0^1 \frac{1}{t}\left\langle \boldsymbol{x}+t(\boldsymbol{y}-\boldsymbol{x})-\boldsymbol{x}, \nabla f\left(\boldsymbol{x}+t(\boldsymbol{y}-\boldsymbol{x})\right)-\nabla f(\boldsymbol{x})\right\rangle \mathrm{d}t \\
&\le \int_0^1 \frac{1}{t}\|\boldsymbol{x}+t(\boldsymbol{y}-\boldsymbol{x})-\boldsymbol{x}\| \cdot \|\nabla f\left(\boldsymbol{x}+t(\boldsymbol{y}-\boldsymbol{x})\right)-\nabla f(\boldsymbol{x})\|\,\mathrm{d}t \qquad (1.3) \\
&\le \int_0^1 \|\boldsymbol{y}-\boldsymbol{x}\| \cdot L\|t(\boldsymbol{y}-\boldsymbol{x})\|\,\mathrm{d}t \qquad (1.4)
\end{aligned}$$

$$
\begin{aligned}
&= L\|\boldsymbol{y}-\boldsymbol{x}\|^2 \int_0^1 t \,\mathrm{d}t \\
&= \frac{L}{2}\|\boldsymbol{y}-\boldsymbol{x}\|^2.
\end{aligned}
$$

Inequality (1.3) holds from the Cauchy-Schwarz Inequality. Inequality (1.4) holds because f is L-Lipschitz differentiable. □

For more details of the proof, please refer to Chapter 18 of Bauschke, Combettes, et al. [Bau11]. Note that in Proposition 1, if we set $\boldsymbol{x} = \tilde{\boldsymbol{y}}$, we get a quadratic majorization of $f(\boldsymbol{y})$ at $\tilde{\boldsymbol{y}}$. A typical example that satisfies convexity and Lipschitz differentiability is $f(x) = -\log \frac{e^x}{1+e^x}$, which is the negative logarithm of occurrence probability under the logistic regression.

We will see corresponding applications of MM technique and this proposition in the following chapters. In Chapter 2, we use convex clustering [Pel05] formulation for clustering histogram data, which produces L_1 norm in the penalty part under an optimization problem. MM gets rid of the non-smoothness of L_1 terms by constructing quadratic majorized functions. In Chapter 3, we investigate a faster solution for shape restricted binomial regression, whose loss function enjoys the similar properties as the logistic regression. So the MM constructs the majorization by using Proposition 1. Chapter 4 discusses the 1-bit matrix completion problem, where we can directly apply the MM technique by using Proposition 1 since its objective function is same as the logistic regression.

CHAPTER

2

ALGORITHMIC REGULARIZATION MAJORIZATION-MINIMIZATION METHOD FOR CLUSTERING HISTOGRAM DATA

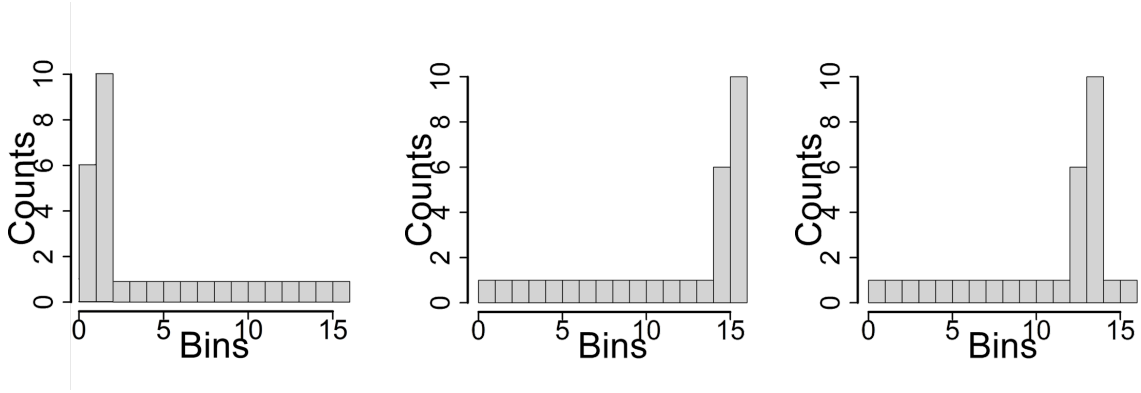
In this chapter, we introduce a new method for hierarchically clustering histogram valued data. Traditional clustering approaches to hierarchical clustering rely on Euclidean distances which are not appropriate distances for assessing differences in histogram valued observations. Instead, we build a hierarchical clustering approach that combines an approximate Earth Mover's Distance with a convex formulation of hierarchical clustering. Our formulation comes with stability guarantees that ensure that small perturbations in the data cannot lead to disproportionate fluctuations in the resulting hierarchical clustering. Additionally, we present a Majorization-Minimization algorithmic regularization path to efficiently compute hierarchical clustering. Numerical examples show that the new method outperforms existing methods.

2.1 Introduction and Background

Histogram valued data is a special type of data whose values come with particular order. Common examples of data with values subject to orderings include time-series data (chronological order) and images (spatial order). In this chapter, we investigate how to account for this ordering in hierarchical

clustering.

Clustering methods aim to partition data into groups so that objects within the same group are more similar than objects from different groups. Consequently, how we quantify similarity between histogram objects will be critical to the clustering task. Many commonly used clustering methods, however, rely on the Euclidean distance, which fails to account for ordering or location information. Figure 2.1 shows an example of three histograms to illustrate the issue with the Euclidean distance: the peaks have identical heights in the three histograms, but the peak locations are different. Histograms $\boldsymbol{x}_2$ and $\boldsymbol{x}_3$ can be considered minor perturbations of each other because their peaks are so close to each other; both of these histograms in contrast can be considered very dissimilar to $\boldsymbol{x}_1$. But the Euclidean distance is exactly the same between any two pairs of histograms, namely $\|\boldsymbol{x}_1-\boldsymbol{x}_2\|_2 = \|\boldsymbol{x}_1-\boldsymbol{x}_3\|_2 = \|\boldsymbol{x}_2-\boldsymbol{x}_3\|_2$. The issue is that the Euclidean distance ignores the ordering structure within the histograms. A more appropriate alternative is the Earth Mover's Distance (EMD) [Rub00]. Intuitively, if we think of probability mass as "earth," we see that $\boldsymbol{x}_2$ requires "moving earth" a shorter distance to reshape $\boldsymbol{x}_2$ into $\boldsymbol{x}_3$ compared to reshaping $\boldsymbol{x}_2$ into $\boldsymbol{x}_1$. Another measure that is more suitable for quantifying the distances between histograms is the Wasserstein-Kantorovich (WK) metric [Irp06]. Kim & Billard [Kim13] proposed yet another similarity metric for histogram observations, and employed it to perform hierarchical clustering [Bil17]. More recently, Park et al. [Par19] incorporated the WK metric with a convex formulation of the clustering problem [Pel05; Lin11; Hoc11]. Despite the improvements in clustering histogram objects by modifying the distance metric used, these prior methods have two drawbacks that we seek to address: i) low clustering accuracy due to insufficient use of the location information and ii) lack of stability, in the sense that small perturbations in the data may dramatically change the clustering assignments.



(a) Histogram $\boldsymbol{x}_1$ **(b)** Histogram $\boldsymbol{x}_2$ **(c)** Histogram $\boldsymbol{x}_3$

Figure 2.1 Three histograms with peaks of the same height at different locations. We have $\|\boldsymbol{x}_1-\boldsymbol{x}_2\|_2 = \|\boldsymbol{x}_1-\boldsymbol{x}_3\|_2 = \|\boldsymbol{x}_2-\boldsymbol{x}_3\|_2$, $d_{\text{EMD}}(\boldsymbol{x}_1,\boldsymbol{x}_3) \approx d_{\text{EMD}}(\boldsymbol{x}_1,\boldsymbol{x}_2) > d_{\text{EMD}}(\boldsymbol{x}_2,\boldsymbol{x}_3)$. The EMD captures differences in shapes across histograms, but the Euclidean distance does not.

To address the low accuracy, we introduce an approximate EMD, which is easy to compute and

can capture the location information just like the true EMD does. We combine this approximate EMD with a convex formulation of the hierarchical clustering task. We prove that the resulting clustering tree possesses attractive stability guarantees. Finally, we propose an efficient algorithm called Algorithmic Regularization Fusion Majorization-Minimization (**ARFMM**) method to solve the problem. ARFMM applies the Majorization-Minimization (**MM**) principle [Hun04] with a fusion scheme and algorithmic regularization [Hu16; Wey19].

The rest of the chapter is organized as follows. In Section 2.2, we review the convex clustering problem and set up our hierarchical clustering procedure as solving a sequence of convex optimization problems. In Section 2.3, we discuss important properties of the solutions to the convex optimization problems described in Section 2.2. In Section 2.4 we describe our algorithmic framework for computing our hierarchical clustering. In Section 2.5, we explore the behavior of ARFMM and compare it to existing methods.We end with a summary and discussion of future work.

2.2 Background and Problem Set Up

Let $\boldsymbol{X}$ be a matrix in $\mathbb{R}^{n\times p}$ with n histogram observations. The ith row $\boldsymbol{X}_{i.} \in \mathbb{R}^p$ is the ith histogram observation with p bins or dimensions. Let $\boldsymbol{x} \in \mathbb{R}^{np}$ represent the column-major vectorization of the transpose of $\boldsymbol{X}$, i.e. $\boldsymbol{x} = \text{vec}(\boldsymbol{X}^T)$; we denote by $\boldsymbol{x}_i$ the ith row of the matrix $\boldsymbol{X}$ arranged as a column vector, i.e. $\boldsymbol{x}_i = (\boldsymbol{X}_{i.})^T$. Scalars such as λ and w_{ij} are real-valued parameters, whose meaning will be described shortly.

Pelckmans et al. [Pel05] posed the clustering task as the following convex optimization problem:

$$\hat{\boldsymbol{V}}(\lambda) \quad = \quad \underset{\boldsymbol{V}\in\mathbb{R}^{n\times p}}{\arg\min} \frac{1}{2}\|\boldsymbol{X}-\boldsymbol{V}\|_{\text{F}}^2 + \lambda\left(\sum_{i<j} w_{ij}\left\|\boldsymbol{V}_{i.}-\boldsymbol{V}_{j.}\right\|_2\right). \tag{2.1}$$

The objective function is the sum of a quadratic data-fidelity term, that quantifies the difference between the data matrix $\boldsymbol{X}$ and a smooth estimate of it $\boldsymbol{V}$, and a sum-of-norms regularization term that penalizes the pairwise distances among the rows of $\boldsymbol{V}$. The penalties on the pairwise differences can be made non-uniform through the use of non-negative weights w_{ij}. The non-negative tuning parameter λ trades off between how well $\boldsymbol{V}$ agrees with $\boldsymbol{X}$ with how smooth $\boldsymbol{V}$ is. On the one hand, for sufficiently large λ, all the rows of $\boldsymbol{V}$ are identical, i.e. $\hat{\boldsymbol{V}}_{i.} = \hat{\boldsymbol{V}}_{j.}$ for all i, j. In this case, all observations have been assigned to the same cluster. On the other hand, when $\lambda = 0$, $\hat{\boldsymbol{V}} = \boldsymbol{X}$. In this case, each observation is assigned to its own a singleton cluster. For appropriately chosen weights w_{ij}, as λ varies the smooth estimate $\hat{\boldsymbol{V}}(\lambda)$ will encode a hierarchical clustering tree [Chi19].

We now describe how we adapt the convex clustering formulation in (2.1) for histogram valued data. As noted earlier, the Euclidean distance fails to account for ordering structure in histograms. Consequently, an intuitive strategy is to replace the Euclidean distance between pairs of rows in the

regularization term in (2.1) with the EMD between pairs of rows:

$$\hat{\boldsymbol{V}}(\lambda) \quad = \quad \underset{\boldsymbol{V}\in\mathbb{R}^{n\times p}}{\arg\min}\, \frac{1}{2}\|\boldsymbol{X}-\boldsymbol{V}\|_{\mathrm{F}}^2 + \lambda\left(\sum_{i<j} w_{ij}\, \mathrm{d}_{\mathrm{EMD}}\left(\boldsymbol{V}_{i.}, \boldsymbol{V}_{j.}\right)\right). \tag{2.2}$$

The optimization problem in (2.2) is convex and admits a unique global minimizer. In fact, Problem (2.2) can be expressed as a Quadratic Program (QP) with $np+n(n-1)p^2/2$ variables (details are given in Appendix A.1), which can be solved with interior point methods in polynomial computational complexity of $\mathcal{O}((n^2p^2)^\alpha)$ where $\alpha \geq 3$.

This complexity is prohibitive in practice, however. Therefore, we define an approximate EMD that yields a computationally tractable problem. The approximate EMD accounts for the ordering structure by cutting and smoothing a histogram over coarser bins and taking a weighted sum of the L_1 distance between the resulting coarsened histograms that have been smoothed at multiple scales. For simplicity, assume that the number of cuttings $N = \log_2 p$ is a positive integer and define the kth cutting mapping $\boldsymbol{T}_k = \boldsymbol{I}_{2^k} \otimes \boldsymbol{1}_{p/2^k}^T : \mathbb{R}^p \to \mathbb{R}^{2^k}$ for $k = 1,\ldots,N$, where $\boldsymbol{1}_{p/2^k}$ is the column vector of 1 with length $p/2^k$ and $\otimes$ denotes the Kronecker product. The approximate EMD for any two histograms $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^p$ is $\tilde{\mathrm{d}}_{\mathrm{EMD}}(\boldsymbol{a},\boldsymbol{b}) = \sum_{k=1}^N 2^{-k}\|\boldsymbol{T}_k\boldsymbol{a} - \boldsymbol{T}_k\boldsymbol{b}\|_1$. This approximate EMD is a special case of a metric over the space of probability distributions that is equivalent to the EMD [Coi13]. We provide additional discussion on the approximate EMD and how it accounts for the ordering structure in Section 2.2.1.

Substituting the approximate EMD for the EMD in (2.2) gives us the following convex optimization problem:

$$\hat{\boldsymbol{v}}(\boldsymbol{x}, \boldsymbol{W}, \lambda) \quad = \quad \underset{\boldsymbol{v}\in\mathbb{R}^{np}}{\arg\min}\, \frac{1}{2}\|\boldsymbol{x}-\boldsymbol{v}\|_2^2 + \lambda \sum_{i<j} w_{ij} \sum_{k=1}^N 2^{-k} \left\|\boldsymbol{T}_k\boldsymbol{v}_i - \boldsymbol{T}_k\boldsymbol{v}_j\right\|_1, \tag{2.3}$$

where $\boldsymbol{W}$ is a matrix whose (i,j)th entry is the weight w_{ij}. To recover a hierarchical clustering structure, we Problem (2.3) for a sequence of tuning parameters $\boldsymbol{\lambda} = \{\lambda_1,\ldots,\lambda_M\}$. The problem formalization in (2.3) enjoys two advantages that address the two drawbacks of existing methods described in Section 2.1.

- The approximate EMD accounts for the ordering structure just as the true EMD does. Table 2.1 shows the comparisons of of the EMD, the approximate EMD, and Euclidean distance for histograms shown in Figure 2.1 after normalization. Even though the approximate EMD does not recover true EMD, it effectively quantifies the work needed to reshape one histogram into another like the EMD and unlike the Euclidean distance.

- The estimated $\hat{\boldsymbol{v}}(\boldsymbol{x}, \boldsymbol{W}, \lambda)$ is stable against the perturbations in the data. We will show that it is actually Lipschitz continuous with respect to $\boldsymbol{x}$.

Table 2.1 Comparison of different types of distances for three histograms in Figure 2.1. Histograms $\boldsymbol{x}_2$ and $\boldsymbol{x}_3$ should be close together and both of them are far away from $\boldsymbol{x}_1$, but the Euclidean distance ignores ordering structure and three pairs have the same distances. By contrast, both the EMD and approximate EMD account for the ordering structure.

PAIR OF DISTANCE	$(\boldsymbol{x}_1, \boldsymbol{x}_2)$	$(\boldsymbol{x}_1, \boldsymbol{x}_3)$	$(\boldsymbol{x}_2, \boldsymbol{x}_3)$
EMD	6.533	5.600	0.933
APPROXIMATE EMD	0.875	0.875	0.175
EUCLIDEAN **DISTANCE**	0.236	0.236	0.236

2.2.1 Intuition behind the Approximate EMD

We construct the approximate EMD by cutting the histograms into coarser bin widths. For simplicity, recall for two histograms $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^p$, we assume that the number of cuttings $N = \log_2 p$ is a positive integer and define the kth cutting mapping $\boldsymbol{T}_k = \boldsymbol{I}_{2^k} \otimes \boldsymbol{1}^T_{p/2^k} : \mathbb{R}^p \to \mathbb{R}^{2^k}$ for $k = 1, \ldots, N$, where $\boldsymbol{I}_{2^k}$ is the identity matrix with size 2^k, $\boldsymbol{1}_{p/2^k}$ is the column vector of 1 with length $p/2^k$ and $\otimes$ denotes the Kronecker product. The approximate EMD for any two histograms $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^p$ is $\tilde{\mathrm{d}}_{\mathrm{EMD}}(\boldsymbol{a}, \boldsymbol{b}) = \sum_{k=1}^{N} 2^{-k} \|\boldsymbol{T}_k \boldsymbol{a} - \boldsymbol{T}_k \boldsymbol{b}\|_1$.

We illustrate this procedure with a simple example of three histograms from Figure 2.1. Here we have $p = 16$ bins; the number of cuttings is $N = \log_2 p = 4$. Consider histograms $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$. The first cutting $\boldsymbol{T}_1 \in \mathbb{R}^{2\times 16}$, which can be written as a linear transformation, cuts the histogram into 2 pieces: the first eight bins and the second eight bins. We then sum the bins under the first and second piece respectively and get a new histogram with 2 bins. We then compute the L_1 distance between the two coarser or smoothed histograms $D_1 = \|\boldsymbol{T}_1 \boldsymbol{x}_1 - \boldsymbol{T}_1 \boldsymbol{x}_2\|_1$. This first cutting procedure is shown in Figure 2.2.

Similarly, we compute $D_k = \|\boldsymbol{T}_k \boldsymbol{x}_1 - \boldsymbol{T}_k \boldsymbol{x}_2\|_1$ at kth cutting for $k = 2, 3, 4$. Note that $p = 16$ in this example, so $\boldsymbol{T}_4$ is the identity matrix and $\boldsymbol{T}_4 \boldsymbol{x}_1 = \boldsymbol{x}_1$. We then define the approximate EMD of $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ as $\tilde{d}_{\mathrm{EMD}}(\boldsymbol{x}_1, \boldsymbol{x}_2) = \sum_{k=1}^{4} 2^{-k} D_k$. This procedure is briefly illustrated by Figure 2.3.

We can see how the approximate EMD accounts for ordering structure (location information) from this example. For histograms $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, the peaks are far away from each other, a scenario in which a good metric between the histograms should be large. The approximate EMD amplifies this discrepancy by adding the L_1 distances of two peaks multiple times. Note that two peaks fall into different pieces after each cutting. So, the L_1 distances between the smoothed copies of $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are all large. On the other hand, if we compute the approximate EMD for $\boldsymbol{x}_2$ and $\boldsymbol{x}_3$, $D_1 = D_2 = 0$, only the L_1 distances between the least and second least smoothed copies of $\boldsymbol{x}_2$ and $\boldsymbol{x}_3$ are not zero. However, these two L_1 distances are also the least and second least emphasized terms in the weighted sum of L_1 distances. Consequently, as desired, the approximate EMD is small, i.e., $\tilde{d}_{\mathrm{EMD}}(\boldsymbol{x}_2, \boldsymbol{x}_3) = 2^{-3} D_3 + 2^{-4} D_4$.

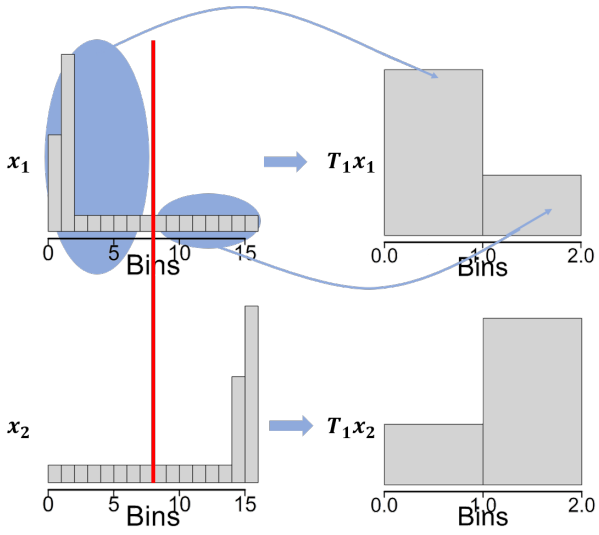
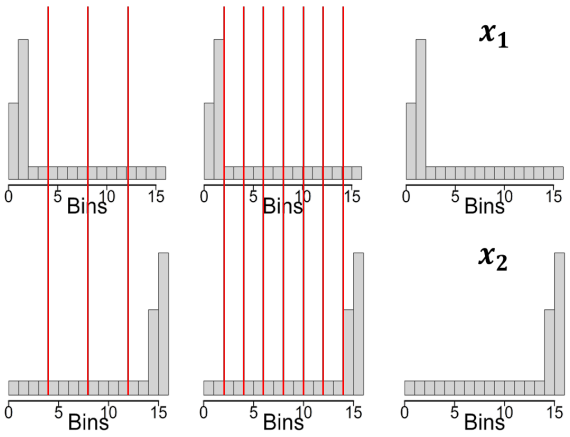


Figure 2.2 The first cutting to construct the approximate EMD between histograms $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$. The red line cuts $\boldsymbol{x}_1$ ($\boldsymbol{x}_2$) into two equally sized coarse bins to get the histogram $\boldsymbol{T}_1\boldsymbol{x}_1$ ($\boldsymbol{T}_1\boldsymbol{x}_2$). We then compute the L_1 distance between the two coarser or smoothed histograms $D_1 = \|\boldsymbol{T}_1\boldsymbol{x}_1 - \boldsymbol{T}_1\boldsymbol{x}_2\|_1$



(a) Second cutting. **(b)** Third cutting **(c)** Fourth cutting

Figure 2.3 Cuttings 2, 3, and 4 for histograms $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$. The red lines in panel (a) and (b) represents the 2nd and 3rd cuttings. Note that the 4th cutting recovers the original histograms $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ so we omit the red lines in panel (c).

2.3 Properties of the Solution to Problem (2.3)

In this section, we describe two important properties of the solution to Problem (2.3). We emphasize that these results are inherent to the minimization of the objective function in (2.3) and hold regardless of the algorithm used to find the minimizer, since they are a consequence of formulating the clustering problem as a convex program.

The first result states that the optimal solution $\hat{\boldsymbol{v}}(\boldsymbol{x}, \boldsymbol{W}, \lambda)$ of Problem (2.3) is a jointly continuous function of the data, weights, and tuning parameter.

Proposition 2 (Continuity)**.** *The optimal solution $\hat{\boldsymbol{v}}(\boldsymbol{x}, \boldsymbol{W}, \lambda)$ to Problem (2.3) is jointly continuous with respect to $(\boldsymbol{x}, \boldsymbol{W}, \lambda)$.*

Proof. First note that tuning parameter λ can be absorbed in each w_{ij}, so we define a new variable $\boldsymbol{\xi} = (\boldsymbol{x}, \boldsymbol{W})$, and we only need to show that the solution $\hat{\boldsymbol{v}}(\boldsymbol{\xi}) = \operatorname{argmin}_{\boldsymbol{v}} f(\boldsymbol{v}, \boldsymbol{\xi})$ is continuous with respect to $\boldsymbol{\xi}$ where the function f is defined as:

$$f(\boldsymbol{v}, \boldsymbol{\xi}) \quad = \quad \frac{1}{2}\|\boldsymbol{x} - \boldsymbol{v}\|_2^2 + \sum_{i<j} w_{ij} \sum_{k=1}^{N} 2^{-k} \left\| \boldsymbol{T}_k \boldsymbol{v}_i - \boldsymbol{T}_k \boldsymbol{v}_j \right\|_1 . \tag{2.4}$$

We proceed with a proof by contradiction. Suppose $\hat{\boldsymbol{v}}(\boldsymbol{\xi})$ is not continuous at some point $\boldsymbol{\xi}^*$, then there exists an $\epsilon > 0$ and a sequence of $\{\boldsymbol{\xi}^{(m)}\}$ converging to $\boldsymbol{\xi}^*$ such that $\|\hat{\boldsymbol{v}}^{(m)} - \hat{\boldsymbol{v}}(\boldsymbol{\xi}^*)\|_2 \geq \epsilon$ for all m where $\hat{\boldsymbol{v}}^{(m)} = \operatorname{argmin}_{\boldsymbol{v}} f(\boldsymbol{v}, \boldsymbol{\xi}^{(m)})$.

Observe that $f(\boldsymbol{v}, \boldsymbol{\xi})$ is a strongly convex function with respect to variable $\boldsymbol{v}$ because of the presence of square of two norm. So both $\hat{\boldsymbol{v}}^{(m)}$ and $\hat{\boldsymbol{v}}(\boldsymbol{\xi}^*)$ exist and are unique. In addition, $f(\boldsymbol{v}, \boldsymbol{\xi})$ is a continuous function jointly with respect to both $\boldsymbol{v}$ and $\boldsymbol{\xi}$. Without loss of generality, we assume $\|\boldsymbol{\xi}^{(m)} - \boldsymbol{\xi}^*\|_2 \leq 1$. Fix an arbitrary point $\tilde{\boldsymbol{v}}$. If $\hat{\boldsymbol{v}}^{(m)}$ is bounded, then there exists a subsequence $\hat{\boldsymbol{v}}_{(k)}^{(m)}$ that converges to some point $\bar{\boldsymbol{v}}$. We replace the original sequence $\{\hat{\boldsymbol{v}}^{(m)}\}$ with this subsequence $\{\hat{\boldsymbol{v}}_{(k)}^{(m)}\}$, then we have:

$$f(\hat{\boldsymbol{v}}^{(m)}, \boldsymbol{\xi}^{(m)}) \quad \leq \quad f(\tilde{\boldsymbol{v}}, \boldsymbol{\xi}^{(m)}).$$

By the continuity of the function $f(\boldsymbol{v}, \boldsymbol{\xi})$, when we take limit with respect to m on both sides of the above inequality we arrive at the following inequality.

$$f(\bar{\boldsymbol{v}}, \boldsymbol{\xi}^*) \quad \leq \quad f(\tilde{\boldsymbol{v}}, \boldsymbol{\xi}^*).$$

Since $\tilde{\boldsymbol{v}}$ is arbitrary, the above inequality implies that $\bar{\boldsymbol{v}} = \hat{\boldsymbol{v}}(\boldsymbol{\xi}^*)$. So, we have that $\hat{\boldsymbol{v}}(\boldsymbol{\xi}^{(m)}) = \hat{\boldsymbol{v}}^{(m)} \to \bar{\boldsymbol{v}} = \hat{\boldsymbol{v}}(\boldsymbol{\xi}^*)$, which is a contradiction.

The remaining task is to show that $\hat{\boldsymbol{v}}^{(m)}$ is bounded. We define a new function g:

$$
\begin{aligned}
g(\boldsymbol{v}) &= \sup_{\tilde{\boldsymbol{\xi}}:\|\tilde{\boldsymbol{\xi}}-\boldsymbol{\xi}^*\|_2\le 1} \frac{1}{2}\|\tilde{\boldsymbol{x}}-\boldsymbol{v}\|_2^2 + \sum_{i<j}\tilde{w}_{ij}\sum_{k=1}^{N} 2^{-k}\left\|\boldsymbol{T}_k\boldsymbol{v}_i - \boldsymbol{T}_k\boldsymbol{v}_j\right\|_1 \\
&= \sup_{\tilde{\boldsymbol{\xi}}:\|\tilde{\boldsymbol{\xi}}-\boldsymbol{\xi}^*\|_2\le 1} f(\boldsymbol{v},\tilde{\boldsymbol{\xi}}).
\end{aligned}
$$

Note that for a fixed $\tilde{\boldsymbol{\xi}}$, the function $f(\boldsymbol{v},\boldsymbol{\xi})$ is a strongly convex with respect to $\boldsymbol{v}$. By taking the supremum of $f(\boldsymbol{v},\boldsymbol{\xi})$ over the unit ball around $\tilde{\boldsymbol{\xi}}^*$, we will still get a strongly convex function. So $g(\boldsymbol{v})$ is also a strongly convex function, and obviously it is proper, so there exist a unique global minimizer $\boldsymbol{v}^*$ with $g(\boldsymbol{v}^*)<\infty$.

Since we assume $\|\boldsymbol{\xi}^{(m)}-\boldsymbol{\xi}^*\|_2\le 1$, we have $f(\boldsymbol{v},\boldsymbol{\xi}^{(m)})\le g(\boldsymbol{v})$ for all m, therefore:

$$
f(\hat{\boldsymbol{v}}^{(m)},\boldsymbol{\xi}^{(m)}) \;\le\; f(\boldsymbol{v}^*,\boldsymbol{\xi}^{(m)}) \;\le\; g(\boldsymbol{v}^*) \quad \forall m. \tag{2.5}
$$

By the reverse triangle inequality, we have

$$
\begin{aligned}
\frac{1}{2}\left(\|\hat{\boldsymbol{v}}^{(m)}\|_2 - \|\boldsymbol{x}^{(m)}\|_2\right)^2 &\le \frac{1}{2}\|\hat{\boldsymbol{v}}^{(m)}-\boldsymbol{x}^{(m)}\|_2^2 \\
&\le f(\hat{\boldsymbol{v}}^{(m)},\boldsymbol{\xi}^{(m)}) \quad \forall m.
\end{aligned} \tag{2.6}
$$

Combining the inequalities in (2.5) and (2.6) together implies

$$
\frac{1}{2}\left(\|\hat{\boldsymbol{v}}^{(m)}\|_2 - \|\boldsymbol{x}^{(m)}\|_2\right)^2 \;\le\; g(\boldsymbol{v}^*) \;<\; \infty \quad \forall m.
$$

Therefore $\hat{\boldsymbol{v}}^{(m)}$ must be a bounded sequence. □

The continuity of $\hat{\boldsymbol{v}}$ suggests a homotopy strategy for efficiently computing $\hat{\boldsymbol{v}}(\lambda_i)$ for $i=1,\dots,M$ and $\lambda_1\le\lambda_2\le\lambda_3\le\cdots\le\lambda_M$. Suppose we have already computed $\hat{\boldsymbol{v}}(\lambda_1),\dots,\hat{\boldsymbol{v}}(\lambda_i)$ and wish to compute $\hat{\boldsymbol{v}}(\lambda_{i+1})$. Since $\hat{\boldsymbol{v}}(\lambda)$ is continuous in λ, we can initialize, or warm start, our iterative algorithm at $\hat{\boldsymbol{v}}(\lambda_i)$ which will likely be closer to the solution to Problem (2.3) when $\lambda=\lambda_{i+1}$ than a random initialization. We will employ this warm start strategy in Section 2.4.2.

The second result has important consequences to the stabiity of our clustering procedure.

Proposition 3 (Lipschitz Continuity)**.** *The optimal solution $\hat{\boldsymbol{v}}(\boldsymbol{x},\boldsymbol{W},\lambda)$ to Problem (2.3) is Lipschitz continuous with respect to $\boldsymbol{x}$ with Lipschitz constant* 1*, namely for all $\boldsymbol{x}$, $\boldsymbol{W}$, and λ,*

$$
\|\hat{\boldsymbol{v}}(\boldsymbol{x}+\Delta\boldsymbol{x},\boldsymbol{W},\lambda)-\hat{\boldsymbol{v}}(\boldsymbol{x},\boldsymbol{W},\lambda)\|_2 \quad\le\quad \|\Delta\boldsymbol{x}\|_2.
$$

Proof. For fixed $\boldsymbol{W}$ and λ, the optimal solution $\hat{\boldsymbol{v}}(\boldsymbol{x},\boldsymbol{W},\lambda)$ is the proximal mapping of a proper, closed, convex function. By Proposition 12.28 in [Bau11], the proximal mapping of a proper, closed, convex function is 1-Lipschitz continuous with respect to $\boldsymbol{x}$. □

Lipschitz continuity with respect to $\boldsymbol{x}$ implies that the clustering result is stable under perturbations in the data. In other words, adding a small perturbation $\Delta\boldsymbol{x}$ to the data $\boldsymbol{x}$ *cannot* cause a disproportionately large change in $\hat{\boldsymbol{v}}$. We will demonstrate this stability in a real data example in Section 2.5.2.

2.4 ARFMM

We solve a smoothed approximation to Problem (2.3) by approximating the L_1 norm with a differentiable surrogate. The smoothed approximation will enable better computational efficiency without any essential qualitative changes to the solution. We next describe the three components of our algorithm.

2.4.1 Majorization-Minimization

We minimize the smoothed approximation with an MM algorithm [Hun04]. Please check the idea behind MM in Section 1.2. MM algorithms are particularly useful when the MM update admits efficient computation. In our case, our choice of majorization will lead to MM updates that correspond to solving sparse linear systems.

We construct a quadratic majorization for a smooth approximation to the regularization term in (2.3) using a standard smooth approximation to the absolute value function, namely

$$|a| \;\leq\; \sqrt{a^2+\epsilon} \;\leq\; \sqrt{\tilde{a}^2+\epsilon}+\frac{a^2-\tilde{a}^2}{2\sqrt{\tilde{a}^2+\epsilon}}, \tag{2.7}$$

where ϵ is a small positive number. Note that the inequality in (2.7) becomes an equality when $a=\tilde{a}$. So the right hand side of the inequality in (2.7) is a quadratic majorization of $\sqrt{a^2+\epsilon}$ at $\tilde{a}$. Note that the regularization term in (2.3) is the sum of L_1 distances, each of which can be written as the sum of absolute values. Thus, it is straightforward to apply the majorization in (2.7) to construct a majorization of a smooth approximation to the regularization term in (2.3) to arrive at the following majorization at $\tilde{\boldsymbol{v}}$ of a smoothed approximation to the objection function in (2.3):

$$g_\epsilon(\boldsymbol{v}|\tilde{\boldsymbol{v}}) \;=\; \frac{1}{2}\|\boldsymbol{x}-\boldsymbol{v}\|_2^2+\lambda\sum_{i<j}w_{ij}\sum_{k=1}^{N}2^{-k}\sum_{l=1}^{2^k}\left\{\frac{1}{2\sqrt{[\boldsymbol{D}_{ij}^{(k)}\tilde{\boldsymbol{v}}]_l^2+\epsilon}}[\boldsymbol{D}_{ij}^{(k)}\boldsymbol{v}]_l^2\right\}, \tag{2.8}$$

with $\boldsymbol{D}_{ij}^{(k)}=(\boldsymbol{e}_i-\boldsymbol{e}_j)^T\otimes\boldsymbol{T}_k\in\mathbb{R}^{2^k\times np}$. Define a diagonal matrix $\boldsymbol{A}_{ij}^{(k)}\in\mathbb{R}^{2^k\times 2^k}$ as $\boldsymbol{A}_{ij}^{(k)}=\text{diag}[a_1,\ldots,a_{2^k}]$ with $a_l=\frac{1}{2\sqrt{[\boldsymbol{D}_{ij}^{(k)}\tilde{\boldsymbol{v}}]_l^2+\epsilon}}$ for $l=1,\ldots,2^k$ as diagonal entries. After some simplification, the majorization $g_\epsilon(\boldsymbol{v}|\tilde{\boldsymbol{v}})$ can be written as the following convex quadratic function:

$$g_\epsilon(\boldsymbol{v}|\tilde{\boldsymbol{v}}) \;=\; \boldsymbol{v}^T\left(\frac{1}{2}\boldsymbol{I}+\boldsymbol{S}\right)\boldsymbol{v}-\boldsymbol{x}^T\boldsymbol{v}+c, \tag{2.9}$$

where $\boldsymbol{I}$ is the identity matrix and $\boldsymbol{S}=\boldsymbol{S}(\lambda,\boldsymbol{W},\tilde{\boldsymbol{v}})=\lambda\sum_{ij}w_{ij}\sum_{k=1}^{N}2^{-k}(\boldsymbol{D}_{ij}^{(k)})^T\boldsymbol{A}_{ij}^{(k)}\boldsymbol{D}_{ij}^{(k)}$. The constant c

does not depend on $\boldsymbol{v}$. The minimizer of $g_\epsilon(\boldsymbol{v}|\tilde{\boldsymbol{v}})$ is given by the MM update formula:

$$\boldsymbol{v}^+ \quad = \quad (\boldsymbol{I}+2\boldsymbol{S})^{-1}\boldsymbol{x}. \tag{2.10}$$

Solving this linear system of size np requires $\mathcal{O}(n^3p^3)$ in general. We use this complexity in the analysis in Section 2.4.4, but in practice, we get a much faster procedure since the matrix $\boldsymbol{S}$ is quite sparse.

For a given λ, we iteratively approach the optimal solution by applying MM updates. The MM procedure to solve Problem (2.3) is summarized in Algorithm 1.

Algorithm 1 MM algorithm for Problem (2.3)

Require: $\boldsymbol{v}, \boldsymbol{W}, \{\lambda_1, \ldots, \lambda_M\}$.
1: **for** $i = 1, \ldots, M$ **do**
2: Initialize $\boldsymbol{v}^-$.
3: **repeat**
4: Update $\boldsymbol{v}^+ = (\boldsymbol{I}+2\boldsymbol{S}(\lambda_i, \boldsymbol{W}, \boldsymbol{v}^-))^{-1}\boldsymbol{x}$.
5: $\boldsymbol{v}^- = \boldsymbol{v}^+$.
6: **until** convergence
7: $\boldsymbol{v}^{(i)} = \boldsymbol{v}^-$
8: **end for**
9: Output: $\{\boldsymbol{v}: \boldsymbol{v}^{(1)}, \ldots, \boldsymbol{v}^{(M)}\}$.

2.4.2 Algorithmic Regularized Path

Algorithm 1 introduces two levels of iterations. The first level is with respect to the tuning parameter λ. The second inner level is a loop of MM updates. We next describe how to improve the efficiency using the algorithmic regularization path strategy [Hu16] which can be considered an inexact form of the alternating direction method of multipliers (ADMM) [Boy11]. Instead of taking several iterations in the inner loop to get a highly accurate solution for a fixed λ, we take only a single inner loop iteration before increasing λ by a small amount. We then compute the next inner iteration using the updated λ with a warm start. To be specific, we define an inflation factor $t > 1$ and replace the inner level loop in Algorithm 1 with the following two steps:

1. Update $\boldsymbol{v}$ via the MM update (2.10) with matrix $\boldsymbol{S}(\lambda)$.

2. Update λ via $\lambda = t\lambda$.

Remarkably very little is lost in the quality of the solution using this extremely inexact but computationally cheap framework, provided that the increase in λ is not too large. In fact, Weylandt et al. [Wey19] proved that for appropriately chosen t, the algorithmic regularization path can approximate to arbitrary accuracy the more computationally demanding exact solutions to the

convex clustering problem over a grid of λs. Since the MM algorithm has similar properties in terms of the convergence speed and Lipschitz continuity with its ADMM counterpart, we expect it shares the same benefits by using the algorithmic regularization path.

2.4.3 Fusions

In each MM update in Algorithm 1, the updated $\boldsymbol{v}^+$ can be written as:

$$\boldsymbol{v}^+ \quad = \quad \begin{pmatrix} \boldsymbol{v}_1^+ \\ \vdots \\ \boldsymbol{v}_n^+ \end{pmatrix} \in \mathbb{R}^{np}.$$

We fuse $\boldsymbol{v}_m$ and $\boldsymbol{v}_q$ if they are close enough to each other, replacing two observations with a single new pseudo observation. For example, if $\|\boldsymbol{v}_m^+ - \boldsymbol{v}_q^+\|_2 < \delta$, where δ is a user-defined tolerance, for some $m, q \in \{1, \ldots, n\}$. we calculate a weighted mean $\boldsymbol{v}_i^* = \frac{1}{2}(\boldsymbol{v}_m^+ + \boldsymbol{v}_q^+)$, and replace $\boldsymbol{v}_m^+, \boldsymbol{v}_q^+$ with this $\boldsymbol{v}_i^*$ to reduce the sample size by 1. We then use the representation with pseudo observations in next update:

$$\boldsymbol{v}^+ \quad = \quad (\boldsymbol{v}_1^{*^T}, \ldots, \boldsymbol{v}_{n^*}^{*^T})^T, \tag{2.11}$$

where $n^* \leq n$ denotes the pseudo sample size after fusions. Fusing observations together enables us to pose a smaller optimization problem to solve. As λ increases, each subsequent optimization problem involves fewer decision variables. This reduction in problem size, however, requires some care in book-keeping which we describe next.

In practice, $\boldsymbol{v}_m^+, \boldsymbol{v}_q^+$ themselves may already be fused versions of some other observations in previous iterations. To compute $\boldsymbol{v}_i^*$, we need to record a history of the fusions in a data structure. Denote this structure as `fusion.information`, which is updated whenever a fusion occurs. For example, `fusion.information` keeps track of information that tells us if the current $\boldsymbol{v}_m^+$ and $\boldsymbol{v}_q^+$ contain 10 and 2 original observations respectively. We use this information to construct a pseudo observation that sets $\boldsymbol{v}_i^* = \frac{1}{12}(10\boldsymbol{v}_m^+ + 2\boldsymbol{v}_q^+)$.

For simplicity of coding, we also assume that only two (pseudo) observations will be fused together in a single fusion. Also, if one observation has multiple possible fusion partners, we select the one with the smallest difference. For example, if $\|\boldsymbol{v}_q^+ - \boldsymbol{v}_m^+\|_2 < \|\boldsymbol{v}_l^+ - \boldsymbol{v}_m^+\|_2 < \delta$, we fuse $\boldsymbol{v}_q^+$ and $\boldsymbol{v}_m^+$ into $\boldsymbol{v}_i^*$, and keep $\boldsymbol{v}_l$ unchanged. These assumptions greatly simplify the task of identifying points to be fused. Not much is lost in making these assumptions as breaking down a single fusion event involving three (pseudo) observations being merged together into two back-to-back fusion events does not drastically change the resulting hierarchical clustering.

Finally, our fusion framework provides a trivial stopping criterion: we stop the algorithm when $n^* = 1$, i.e. when all original observations have been fused into a single cluster, and the final hierarchical clustering result can be extracted from the `fusion.information`.

Algorithm 2 summarizes our final procedure that incorporates several elements. Based on the

three techniques applied, we call our estimation procedure the **A**lgorithmic **R**egularization **F**usion **M**ajorization-**M**inimization (**ARFMM**) method.

Algorithm 2 ARFMM

Require: $\boldsymbol{x}, \boldsymbol{W}, \lambda, t, \delta$.
1: Initialize $\boldsymbol{v}^{(0)} = \boldsymbol{x}$, $n^{(0)*} = n$, $i = 0$.
2: Initialize `fusion.information`.
3: **while** $n^{(i)*} > 1$ **do**
4: $i = i + 1$.
5: Update $\boldsymbol{v}^{(i)+} = \left(\boldsymbol{I} + 2\boldsymbol{S}(\lambda, \boldsymbol{W}, \boldsymbol{v}^{(i-1)})\right)^{-1} \boldsymbol{v}^{(i-1)}$.
6: Do fusion on $\boldsymbol{v}^{(i)+}$, denote the fused observation $\boldsymbol{v}^{(i)}$ with fused sample size $n^{(i)*}$.
7: Update `fusion.information`.
8: $\lambda = t\lambda$
9: **end while**
10: Output: $\{\boldsymbol{v}: \boldsymbol{v}^{(1)}, \boldsymbol{v}^{(2)}, \dots\}$ and `fusion.information`.

2.4.4 Complexity Analysis

In this section we analyze the computational complexity of ARFMM. Assume that we use a sequence of tuning parameters $\{\lambda_1, \dots, \lambda_M\}$ and observations fuse together under λ_M. We only compute a single MM update for each tuning parameter value, which requires $\mathcal{O}(n^3 p^3)$ flops. Then the complexity of ARFMM has an upper bound $\mathcal{O}(M n^3 p^3)$ flops in a worst case. Note that our analysis does not account for intermediate fusion events. We consider the worst case where there are no fusions under the first $(M-1)$ tuning parameter values and all observations fuse together under λ_M. In practice, many observations fuse together at early steps and leading to much lower computational costs.

2.4.5 Setting Weights

We now address the question of how to set the weight parameters w_{ij}. We employ sparse Gaussian kernel weights [Chi15] and define $w_{ij} = \iota^k_{\{ij\}} \exp(-\mathrm{d}_{\mathrm{EMD}}(\boldsymbol{x}_i, \boldsymbol{x}_j))$. The indicator $\iota^k_{\{ij\}}$ is 1 if and only if $\boldsymbol{x}_i$ is among the k-nearest neighbors of $\boldsymbol{x}_j$ or $\boldsymbol{x}_j$ is among the k-nearest neighbors of $\boldsymbol{x}_i$. Here distance is measured by the EMD. We set w_{ij} in this way since it ensures that $\hat{\boldsymbol{v}}(\lambda)$ will recover a tree or global organization that is consistent with the local geometry encoded in the pairwise EMDs of the observations [Chi19]. To set the number of nearest neighbors k, we turn to the following graphical interpretation of clustering where each histogram observation represents a node in a graph and an edge connects node i and node j whenever $w_{ij} > 0$. We set k to be the smallest integer such that the graph is connected. As k gets smaller, the linear system in the MM update (2.10) gets sparser. At the same time if k so small that the graph becomes disconnected, the algorithm will return disconnected trees, one for each connected component of the graph.

2.5 Numerical Experiments

In this section, we evaluate ARFMM on simulated and real data. We visualize the results with a dendrogram representing a hierarchical clustering tree, using `fusion.information`. Details on how we construct dendrograms is given in Appendix A.2.

2.5.1 Simulated Bivariate Normal

We first revisit a simulation study from Billard & Kim [Bil17]. We draw 5 bivariate distributions from a 3-component mixture of bivariate normals; we then sample 100 observations from the selected bivariate normal. Specifically, set:

$$s_1 \sim N(\mu_1, \boldsymbol{C}_1) \quad s_2, s_3 \sim N(\mu_2, \boldsymbol{C}_1) \quad s_4, s_5 \sim N(\mu_2, \boldsymbol{C}_2)$$

with $\mu_1 = (6,1)^T, \mu_2 = (1,1)^T, \boldsymbol{C}_1 = \begin{pmatrix} 0.2 & 0 \\ 0 & 0.2 \end{pmatrix}$ and $\boldsymbol{C}_2 = \begin{pmatrix} 1 & 0.2 \\ 0.2 & 0.8 \end{pmatrix}$. The distribution patterns of the 500 points are shown on x-y plane in Figure 2.4. We then generate histograms from this raw data by cutting the plane into 64 bins (8 bins along the x-axis and 8 bins along the y-axis). For each cluster and each bin, we compute how many points out of 100 fall into that bin and record the frequency bin-by-bin, then we generate a histogram for each underlying cluster. In this problem, $n = 5$ and $p = 64$.

Figure 2.4 Distribution of 500 points from bivariate normal sample on a true x-y plane. The underlying distributions s_2, s_3, s_4, and s_5 share a same mean vector but the covariance matrix of s_2, s_3 are different from s_4, s_5, and s_1 has a different mean vector.

The clustering tree generated by ARFMM is shown in Figure 2.5. Clusters (s_2, s_3) and (s_4, s_5) first

converge together and finally merge with cluster s_1. The recovered dendrogram agrees with how we defined the underlying clusters and generated the observations.

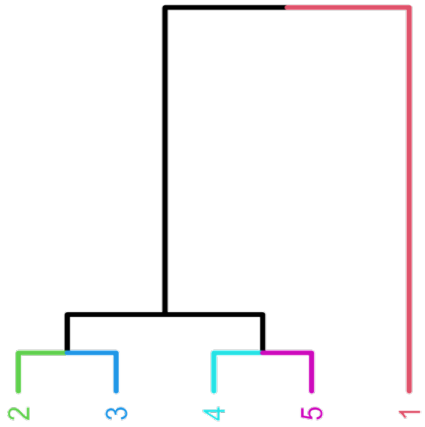


Figure 2.5 Dendrogram of bivariate normal example given by the ARFMM, s_2 (s_4) and s_3 (s_5) converge together, and finally merge with s_1.

2.5.2 NMR Wine Data Set

Nuclear Magnetic Resonance (**NMR**) spectroscopy provides a representative example of histogram data. NMR signals are produced by excitation of the nuclei sample with different radio waves into nuclear magnetic resonance. By treating the signals at different wavelengths as masses distributed among bins corresponding to ranges of wavelengths, we can normalize an NMR spectrum into a one-way histogram. We use this NMR data to investigate ARFMM's stability to perturbations.

We use a data set of NMR spectra of table wines given by Larsen et al. [Lar06]; it records NMR spectra of 40 samples of 3 different wine types (white, rosé, red). Each spectra has 8712 responses among different indices or wavelengths. Figure 2.6 displays two spectra: the black one shows the original spectrum of observation No.1 in this data set. We created the red spectrum by adding Gaussian noise to 100 out of 8712 responses and shifting the entire spectrum to left by 50 indices. The differences between the black and red spectra are minor. Consequently, a clustering method that is insensitive to noise and minor shift perturbations, in short stable, should return clustering results that are relatively unchanged if the black spectrum is substituted by the red one in the collection of spectra.

We evaluate the stability of **ARFMM** and two other hierarchical clustering alternatives: **hclust**, which is an instance of classic hierarchical clustering using the Lance-Williams method [Lan67] and **cvxclustrWK**, which is an alternative convex clustering algorithm designed for histogram data [Par19]. We created histogram valued observations from the raw spectra data using 128 bins, leaving

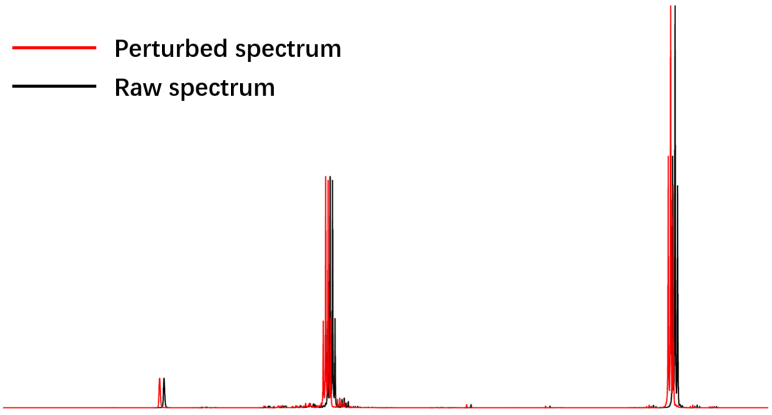


Figure 2.6 NMR spectra of observation No.1 before and after adding perturbations, the black spectrum is the original one and the red spectrum is the perturbed one. The differences between the original and perturbed spectra are very minor.

us with a data set of $n = 40$ observations of histograms in $p = 128$ dimensions.

The 40 observations consist of 7 white wines (W), 2 rosé wines (Ro) and 31 red wines (R). According to Larsen et al. [Lar06], white and rosé wines have similar spectra. Consequently, we should anticipate that a good clustering procedure should group white and rosé wines together. We first check whether each of the methods gives reasonable cluster assignments in the original data set (unperturbed data set) with this domain knowledge. We then check how sensitive the clustering results are for each method to perturbing a single observation in the data set (perturbed data set).

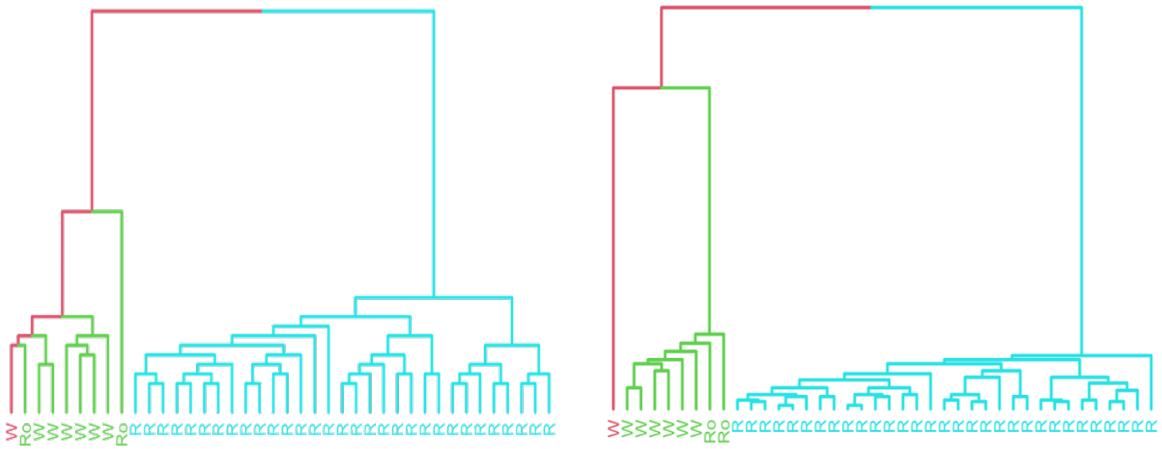
2.5.2.1 ARFMM

Figure 2.7a shows the dendrogram recovered by ARFMM on the unperturbed data. We label each leaf as W, Ro or R according to the observation's wine type. We see that there are two main clusters as anticipated: white/rosé wines (green) and red wines (cyan). The observation No.1 is colored red so that we can easily compare where it has been clustered in the unperturbed and perturbed data sets.

Figure 2.7b shows the dendrogram recovered by ARFMM on the perturbed data. We see that the perturbed No.1 (red) merges with the green cluster before all observations fuse together. Figure 2.7a and Figure 2.7b together show that the added perturbation does not drastically change the hierarchical clustering results of ARFMM, illustrating ARFMM's stability to noise and shift perturbations.

2.5.2.2 hclust

Figure 2.8a shows the dendrogram recovered by hclust on the unperturbed data. We see that hclust also recovers the same two main clusters as anticipated. Figure 2.8b shows the dendrogram recovered by hclust on the perturbed data. We use a black branch to represent the fusion of observations from different clusters. We see that on the perturbed data, hclust puts observation No.1 into its own single



(a) ARFMM on unperturbed data set.

(b) ARFMM on perturbed data set.

Figure 2.7 Dendrograms given by ARFMM under two data sets. Observation No.1 is colored red so that we can easily compare where it has been clustered in the unperturbed and perturbed data sets: (a) ARFMM recovers two main clusters: the first cluster in green consists of white (W) and rosé (Ro) wines and the second cluster in cyan consists of red (R) wines on the unperturbed data set, (b) ARFMM recovers very similar clusters on the perturbed data, illustrating ARFMM's stability to noise and shift perturbations.

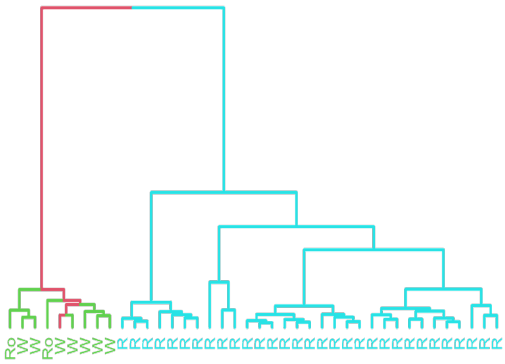
cluster, treating the subtly perturbed observation as an outlier. Based on the illustrative examples from Figure 2.1b and Table 2.1, this is not surprising as hclust employs the Euclidean distance to quantify the similarity between observations.

2.5.2.3 cvxclustrWK

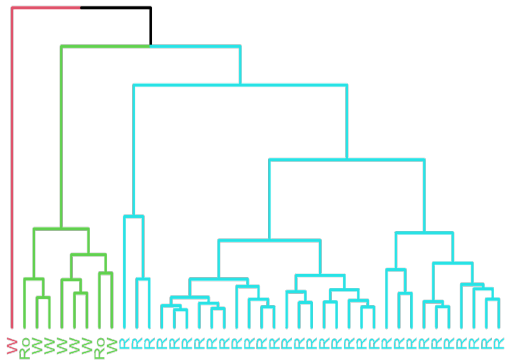
CvxclustrWK clusters histogram valued data by minimizing a convex objective whose data-fidelity term is the Euclidean distance between sample quantiles from pairs of histograms and the regularization term is the sum of WK distances [Par19]. Figure 2.9 shows the dendrograms recovered by cvxclustrWK on the unperturbed data. We see that even for unperturbed data, cvxclustrWK does not identify the anticipated clusters, as there is no clear separation between the white / rosé group and red group. Figure 2.9b shows the dendrogram recovered by cvxlustrWK on the perturbed data. We see that on the perturbed data, the recovered dendrogram is visually quite different from the dendrogram computed from the unperturbed data, illustrating that cvxclustrWK appears to be sensitive to the minor perturbations introduced in observation No.1 . We provide some discussion on why the WK metric does not adequately capture enough location information in cvxclustrWK in Appendix A.3.

2.6 Discussion and Future Work

In this chapter, we proposed a new stable procedure for hierarchically clustering histogram valued data as well as a practical algorithm, ARFMM, for computing the dendrograms. We compared

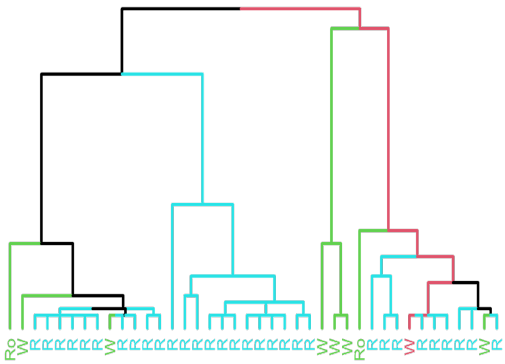


(a) hclust on unperturbed data set.

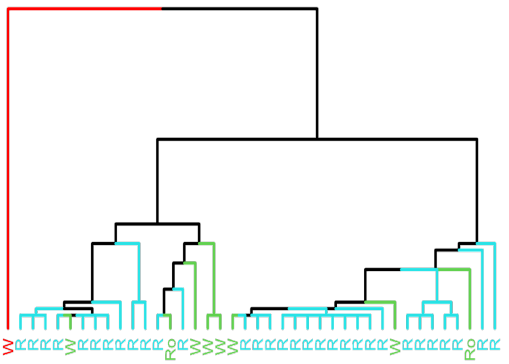


(b) hclust on perturbed data set.

Figure 2.8 Dendrograms given by hclust under two data sets. Observation No.1 is colored red so that we can easily compare where it has been clustered in the unperturbed and perturbed data sets: hclust recovers two main clusters: the first cluster in green consists of white (W) and rosé (Ro) wines and the second cluster in cyan consists of red (R) wines on the unperturbed data set, (b) observation No.1 is treated as an outlier and does not merge until all other observations fuse together under the perturbed data set, illustrating that the hclust method is not as stable as the ARFMM.



(a) cvxclustrWK on unperturbed data set.



(b) cvxclustrWK on perturbed data set.

Figure 2.9 Dendrograms given by cvxclustrWK under two data sets. Observation No.1 is colored red so that we can easily compare where it has been clustered in the unperturbed and perturbed data sets: (a) cvxclustrWK does not recover the anticipated two clusters of white(W) / rosé(Ro) and red(R) wines in the unperturbed data set, (b) observation No.1 is treated as an outlier, and white, rosé, and red wines are also mixed together in the major subgroups found, illustrating the cvxclustrWK appears to be sensitive to the minor perturbations introduced in observation No.1 .

ARFMM to two existing hierarchical clustering methods, hclust and cvxclustrWK. Compared to these methods, ARFMM shows noticeably less sensitivity to perturbations.

The ARFMM framework shows promising utility for hierarchically clustering histogram valued observations, but there are directions that warrant future work. Some directions for further investigation include: (i) further improvements in the computational speed to solve the optimization problem, especially in the fusion step, and (ii) a proof for convergence in using the algorithmic regularization path for the MM algorithm.

CHAPTER

3

A SHARP-MAJORIZED ADMM ALGORITHM TO SOLVE SHAPE RESTRICTED BINOMIAL REGRESSION PROBLEM

This chapter discusses a majorized Alternative Direction Method of Multipliers (ADMM) algorithm for solving the binomial regression problem with linear constraints. This type of problem can be solved via the standard ADMM algorithm, an iterated strategy that usually contains a complex updating formula. Different modifications are developed based on majorization to simplify the standard ADMM. In general, people construct a quadratic majorization and convert the complex update into a simpler problem. Previous studies also build corresponding convergence results. We develop a sharp majorization method that achieves the deepest descending. Numerical experiments show that the sharp majorization can save many iteration numbers and run time under the same convergent tolerance.

3.1 Introduction

We first investigate the following convex composite optimization problem:

$$\begin{aligned} &\underset{\boldsymbol{y}\in \mathrm{Y},\boldsymbol{z}\in \mathrm{Z}}{\text{minimize}}\ p(\boldsymbol{y})+f(\boldsymbol{y})+q(\boldsymbol{z})+g(\boldsymbol{z}) \\ &\text{subject to: } \boldsymbol{A}^*\boldsymbol{y}+\boldsymbol{B}^*\boldsymbol{z}=\boldsymbol{c}, \end{aligned} \tag{3.1}$$

where X, Y, Z are given finite dimensional Euclidean spaces each equipped with an inner product $\langle\cdot\,,\cdot\rangle$; $f:\mathrm{Y}\to\mathbb{R}$ and $g:\mathrm{Z}\to\mathbb{R}$ are two convex and Lipschitz differentiable functions; $p:\mathrm{Y}\to(-\infty,+\infty]$ and $q:\mathrm{Z}\to(-\infty,+\infty]$ are two closed proper convex functions. Two linear operators $\boldsymbol{A}^*:\mathrm{Y}\to\mathrm{X}$ and $\boldsymbol{B}^*:\mathrm{Z}\to\mathrm{X}$ are adjoints of the linear operators $\boldsymbol{A}:\mathrm{X}\to\mathrm{Y}$ and $\boldsymbol{B}:\mathrm{X}\to\mathrm{Z}$ respectively; and $\boldsymbol{c}\in\mathrm{X}$. Problem (3.1) is with a very general definition. In practice, we usually work under a finite dimensional vector space $\mathbb{R}^m$ with a specific dimension m. For example, $\mathrm{Y}=\mathrm{Z}=\mathbb{R}^m$ and $\mathrm{X}=\mathbb{R}^n$ with a regular inner product $\langle\boldsymbol{a},\boldsymbol{b}\rangle=\boldsymbol{a}^T\boldsymbol{b}$. Then $\boldsymbol{A},\boldsymbol{B}\in\mathbb{R}^{m\times n}$ and $\boldsymbol{A}^*=\boldsymbol{A}^T,\boldsymbol{B}^*=\boldsymbol{B}^T\in\mathbb{R}^{n\times m}$ satisfy $\langle\boldsymbol{y}\,,\boldsymbol{A}\boldsymbol{c}\rangle=\langle\boldsymbol{A}^*\boldsymbol{y}\,,\boldsymbol{c}\rangle$ and $\langle\boldsymbol{z}\,,\boldsymbol{B}\boldsymbol{c}\rangle=\langle\boldsymbol{B}^*\boldsymbol{z}\,,\boldsymbol{c}\rangle$ following the definition of adjoint operator.

3.1.1 Preliminaries and Existing Methods

Many numerical methods have been established for solving Problem (3.1) from many years ago. The traditional and typical one is the method of multipliers [Ber14], which uses the following augmented Lagrangian function:

$$\mathscr{L}_\sigma(\boldsymbol{y},\boldsymbol{z};\boldsymbol{x}) \;=\; p(\boldsymbol{y})+f(\boldsymbol{y})+q(\boldsymbol{z})+g(\boldsymbol{z})+\langle\boldsymbol{A}^*\boldsymbol{y}+\boldsymbol{B}^*\boldsymbol{z}-\boldsymbol{c}\,,\boldsymbol{x}\rangle+\frac{\sigma}{2}\|\boldsymbol{A}^*\boldsymbol{y}+\boldsymbol{B}^*\boldsymbol{z}-\boldsymbol{c}\|^2, \tag{3.2}$$

and iteratively updates:

$$\begin{aligned} (\boldsymbol{y}^{(k+1)},\boldsymbol{z}^{(k+1)}) &= \arg\min\ \mathscr{L}_\sigma(\boldsymbol{y},\boldsymbol{z};\boldsymbol{x}^{(k)}), \\ \boldsymbol{x}^{(k+1)} &= \boldsymbol{x}^{(k)}+\tau\sigma(\mathrm{A}^*\boldsymbol{y}^{(k+1)}+\mathrm{B}^*\boldsymbol{z}^{(k+1)}-\boldsymbol{c}), \end{aligned} \tag{3.3}$$

with the augmented Lagrangian parameter σ, a proper step-length τ, and iteration number k. Actually, this method is a particular case of the proximal point algorithm [Roc76] with only equality constraints applied onto the dual problem of (3.1). The proximal point algorithm uses the non-expansive property of the proximal operator to construct a convergence sequence to the optimal solution, if there any exists. Please refer to [Lem89; Ius99; Com11] for more details of the proximal point algorithm and its connections to the augmented Lagrangian. Note that step (3.3) is often complex without any closed updating formula, where another level of numerical method has to be introduced to get the inexact minimization. So one key point of the method of multipliers is a convenient stopping criterion for the inexact minimization shown in (3.3) that still guarantees the global convergence of the algorithm. Several studies investigate this issue under the assumptions such as strong convexity, which is not always presented by the objective function. Rockafellar [Roc78] and Rockafellar [Roc76] hence obtained the proximal method of multipliers by replacing f and g with their proximal mappings:

$$(\boldsymbol{y}^{(k+1)},\boldsymbol{z}^{(k+1)}) \;=\; \arg\min\ \mathscr{L}_\sigma(\boldsymbol{y},\boldsymbol{z};\boldsymbol{x}^{(k)})+\frac{\mu}{2}(\|\boldsymbol{y}-\boldsymbol{y}^{(k)}\|^2+\|\boldsymbol{z}-\boldsymbol{z}^{(k)}\|^2), \tag{3.4}$$

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \tau\sigma(\boldsymbol{A}^*\boldsymbol{y}^{(k+1)} + \boldsymbol{B}^*\boldsymbol{z}^{(k+1)} - \boldsymbol{c}),$$

where $\mu \in \mathbb{R}$ is another user-defined value for the proximal terms. This strategy is applying the proximal point algorithm to a saddle point formulation of Problem (3.1) [Eck94]. But in either (3.3) or (3.4), the norm square $\|\boldsymbol{A}^*\boldsymbol{y}+\boldsymbol{B}^*\boldsymbol{z}-\boldsymbol{c}\|^2$ has a non-separable cross term containing both $\boldsymbol{y}$ and $\boldsymbol{z}$. It prevents the separable strategy from minimizing $\boldsymbol{y}$ and $\boldsymbol{z}$ respectively and increases the difficulty of the problem.

Many approaches have been proposed for eliminating this issue. The typical one is the Alternative Direction Method of Multipliers (ADMM) [Glo75; Gab76; Boy11], which splits the primal update in (3.3) into two separable parts:

$$\begin{aligned}
\boldsymbol{y}^{(k+1)} &= \arg\min\ \mathscr{L}_\sigma(\boldsymbol{y}, \boldsymbol{z}^{(k)}; \boldsymbol{x}^{(k)}), && (3.5)\\
\boldsymbol{z}^{(k+1)} &= \arg\min\ \mathscr{L}_\sigma(\boldsymbol{y}^{(k)}, \boldsymbol{z}; \boldsymbol{x}^{(k)}), && (3.6)\\
\boldsymbol{x}^{(k+1)} &= \boldsymbol{x}^{(k)} + \tau\sigma(\boldsymbol{A}^*\boldsymbol{y}^{(k+1)} + \boldsymbol{B}^*\boldsymbol{z}^{(k+1)} - \boldsymbol{c}).
\end{aligned}$$

The augmented Lagrangian shown in (3.2) is minimized first over $\boldsymbol{y}$ and then over $\boldsymbol{z}$. It does an alternating direction instead of truly minimizing the joint variable $(\boldsymbol{y}, \boldsymbol{z})$. The ADMM method is also a realization of the proximal point algorithm with alternating the directions iteration by iteration, see more details in [Par14; Boy11]. Now that the formulation in (3.4) improves the convergence result and the formulations in (3.5) and (3.6) simplify the minimization, the natural idea is to construct an algorithm both having proximal terms and alternating directions. Eckstein [Eck94] proposed the **proximal ADMM** method having both features:

$$\begin{aligned}
\boldsymbol{y}^{(k+1)} &= \arg\min\ \mathscr{L}_\sigma(\boldsymbol{y}, \boldsymbol{z}^{(k)}; \boldsymbol{x}^{(k)}) + \frac{\mu}{2}\|\boldsymbol{y}-\boldsymbol{y}^{(k)}\|^2, && (3.7)\\
\boldsymbol{z}^{(k+1)} &= \arg\min\ \mathscr{L}_\sigma(\boldsymbol{y}^{(k)}, \boldsymbol{z}; \boldsymbol{x}^{(k)}) + \frac{\mu}{2}\|\boldsymbol{z}-\boldsymbol{z}^{(k)}\|^2, && (3.8)\\
\boldsymbol{x}^{(k+1)} &= \boldsymbol{x}^{(k)} + \tau\sigma(\boldsymbol{A}^*\boldsymbol{y}^{(k+1)} + \boldsymbol{B}^*\boldsymbol{z}^{(k+1)} - \boldsymbol{c}).
\end{aligned}$$

The convergence results and numerical performances of the proximal ADMM are discussed in [Eck94].

3.1.2 Majorized iPADMM and Sharp-Majorized ADMM

More recently, people further improve the proximal ADMM algorithm by replacing the proximal terms in (3.7) and (3.8) with general positive semi-definite linear operators or even indefinite linear operators. Specifically, note that $\|\boldsymbol{y}-\boldsymbol{y}^{(k)}\|^2 = \|\boldsymbol{y}-\boldsymbol{y}^{(k)}\|_{\boldsymbol{I}}^2$ is with a special linear operator $\boldsymbol{I}: \mathrm{Y} \to \mathrm{Y}$. It is self-adjoint and positive definite. Fazel et al. [Faz13] proposed using a positive semi-definite operator instead of the trivial $\boldsymbol{I}$. In other words, substitute $\|\boldsymbol{y}-\boldsymbol{y}^{(k)}\|^2$, $\|\boldsymbol{z}-\boldsymbol{z}^{(k)}\|^2$with $\|\boldsymbol{y}-\boldsymbol{y}^{(k)}\|_{\boldsymbol{S}}^2$, $\|\boldsymbol{z}-\boldsymbol{z}^{(k)}\|_{\boldsymbol{T}}^2$ respectively where $\boldsymbol{S}$ and $\boldsymbol{T}$ are positive semi-definite and self-adjoint operators on space Y and Z. With some other conditions on $\boldsymbol{S}$ and $\boldsymbol{T}$, the authors established the convergence theory and showed this relaxation of positive definiteness improves the efficiency of the alternating direction

method. The corresponding **semi-proximal ADMM** iterates:

$$\begin{aligned} \boldsymbol{y}^{(k+1)} &= \arg\min \mathscr{L}_\sigma(\boldsymbol{y}, \boldsymbol{z}^{(k)}; \boldsymbol{x}^{(k)}) + \frac{\mu}{2}\|\boldsymbol{y}-\boldsymbol{y}^{(k)}\|_{\boldsymbol{S}}^2, \qquad (3.9)\\ \boldsymbol{z}^{(k+1)} &= \arg\min \mathscr{L}_\sigma(\boldsymbol{y}^{(k)}, \boldsymbol{z}; \boldsymbol{x}^{(k)}) + \frac{\mu}{2}\|\boldsymbol{z}-\boldsymbol{z}^{(k)}\|_{\boldsymbol{T}}^2, \qquad (3.10)\\ \boldsymbol{x}^{(k+1)} &= \boldsymbol{x}^{(k)} + \tau\sigma(\boldsymbol{A}^*\boldsymbol{y}^{(k+1)} + \boldsymbol{B}^*\boldsymbol{z}^{(k+1)} - \boldsymbol{c}). \end{aligned}$$

Note that the above formulation reduces to the classic ADMM when $\boldsymbol{S} = \boldsymbol{T} = \boldsymbol{0}$; if $\boldsymbol{S}$ and $\boldsymbol{T}$ are both identity in each space, we get the proximal ADMM; if both $\boldsymbol{S}$ and $\boldsymbol{T}$ are self-adjoint positive semi-definite linear operators, we have the semi-proximal ADMM. Nevertheless, there still remains two drawbacks:

- Updates (3.9) and (3.10) usually need another level of iterations to numerically approach the solution, especially when f and g have complex formats.
- The conditions for $\boldsymbol{S}$ and $\boldsymbol{T}$ are hard to satisfy, especially the positive semi-definiteness.

Many studies focus on these two drawbacks with further modification of the semi-proximal ADMM. Li et al. [Li16] found that even an **I**ndefinite **P**roximal term can do better. Also, majorization techniques are introduced to simplify the minimization of the augmented Lagrangian. The authors hence proposed the **majorized iPADMM** method. Zhang et al. [Zha20] proposed an improved version of majorized iPADMM to get a Q-linear convergence rate. There are many other works related to this topic, including augmented ADMM [Zhu17], majorized ADMM for coupled objective function [Cui15] and different types of inexact ADMM [Hag19].

We follow the majorized iPADMM method in [Li16] and construct the majorization based on the Lipschitz differentiabilities of f and g. There exists self-adjoint positive semi-definite linear operators $\hat{\boldsymbol{\Sigma}}_f$ and $\hat{\boldsymbol{\Sigma}}_g$ such that for any $\boldsymbol{y}, \boldsymbol{y}' \in \mathrm{Y}$ and $\boldsymbol{z}, \boldsymbol{z}' \in \mathrm{Z}$:

$$\begin{aligned} f(\boldsymbol{y}) \le \hat{f}(\boldsymbol{y}, \boldsymbol{y}') &= f(\boldsymbol{y}') + \langle \nabla f(\boldsymbol{y}'), \boldsymbol{y}-\boldsymbol{y}' \rangle + \tfrac{1}{2}\|\boldsymbol{y}-\boldsymbol{y}'\|_{\hat{\boldsymbol{\Sigma}}_f}^2,\\ g(\boldsymbol{z}) \le \hat{g}(\boldsymbol{z}, \boldsymbol{z}') &= g(\boldsymbol{z}') + \langle \nabla g(\boldsymbol{z}'), \boldsymbol{z}-\boldsymbol{z}' \rangle + \tfrac{1}{2}\|\boldsymbol{z}-\boldsymbol{z}'\|_{\hat{\boldsymbol{\Sigma}}_g}^2. \end{aligned} \qquad (3.11)$$

Majorizations $\hat{f}$ and $\hat{g}$ replace complicated f and g with quadratic functions, which will simplify the updates in (3.9) and (3.10). A common choice of $\hat{\boldsymbol{\Sigma}}_f$ and $\hat{\boldsymbol{\Sigma}}_g$ can be induced by Proposition 1 since f and g are Lipschitz differentiable. Specifically, suppose that ∇f and ∇g have Lipschitz constant L_f and L_g respectively. We can easily define $\hat{\boldsymbol{\Sigma}}_f = L_f \boldsymbol{I}$ and $\hat{\boldsymbol{\Sigma}}_g = L_g \boldsymbol{I}$. However, in practice, there exists $\hat{\boldsymbol{\Sigma}}_f \preceq L_f \boldsymbol{I}$ and $\hat{\boldsymbol{\Sigma}}_g \preceq L_g \boldsymbol{I}$ that also satisfy the inequalities in (3.11). This implies there exists majorization which is "sharper" than the L-majorization generated by Proposition 1. So we expect minimizing this "sharper" majorization to achieve much more descending in each iteration, which will improve the efficiency of the whole algorithm. We then construct the majorized augmented Lagrangian function at $\boldsymbol{y}', \boldsymbol{z}'$ based on (3.11):

$$\hat{\mathscr{L}}_\sigma(\boldsymbol{y}, \boldsymbol{z}, \boldsymbol{x}; \boldsymbol{y}', \boldsymbol{z}') = \hat{f}(\boldsymbol{y}, \boldsymbol{y}') + p(\boldsymbol{y}) + \hat{g}(\boldsymbol{z}, \boldsymbol{z}') + q(\boldsymbol{z}) + \langle \boldsymbol{A}^*\boldsymbol{y} + \boldsymbol{B}^*\boldsymbol{z} - \boldsymbol{c}, \boldsymbol{x} \rangle + \frac{\sigma}{2}\|\boldsymbol{A}^*\boldsymbol{y} + \boldsymbol{B}^*\boldsymbol{z} - \boldsymbol{c}\|^2 \qquad (3.12)$$

The majorized iPADMM method selects two self-adjoint, but may not be positive semi-definite linear operators $\boldsymbol{S}$ and $\boldsymbol{T}$ that satisfy the following conditions:

$$\begin{aligned} \tfrac{1}{2}\hat{\boldsymbol{\Sigma}}_f + \boldsymbol{S} &\succcurlyeq \boldsymbol{0}, \\ \tfrac{1}{2}\hat{\boldsymbol{\Sigma}}_g + \boldsymbol{T} &\succcurlyeq \boldsymbol{0}, \\ \tfrac{1}{2}\hat{\boldsymbol{\Sigma}}_f + \boldsymbol{S} + \sigma \boldsymbol{A}\boldsymbol{A}^* &\succ \boldsymbol{0}, \\ \tfrac{1}{2}\hat{\boldsymbol{\Sigma}}_g + \boldsymbol{T} + \sigma \boldsymbol{B}\boldsymbol{B}^* &\succ \boldsymbol{0}. \end{aligned} \tag{3.13}$$

With these conditions, the Algorithm 3 describes the procedure of the majorized iPADMM for solving Problem (3.1).

Algorithm 3 Majorized iPADMM for Problem (3.1)

Require: Let $\sigma > 0$ and $\tau \in (0, (1+\sqrt{5})/2)$.
Require: Let $\boldsymbol{S}$ and $\boldsymbol{T}$ be self-adjoint linear operators satisfy the conditions in (3.13).
1: Initialize $\boldsymbol{y}^{(0)}, \boldsymbol{z}^{(0)}, \boldsymbol{x}^{(0)}$.
2: **while** not converge **do**
3: $\quad \boldsymbol{y}^{(k+1)} = \arg\min \hat{\mathscr{L}}_\sigma(\boldsymbol{y}, \boldsymbol{z}^{(k)}, \boldsymbol{x}^{(k)}; \boldsymbol{y}^{(k)}, \boldsymbol{z}^{(k)}) + \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{y}^{(k)}\|^2_{\boldsymbol{S}}$.
4: $\quad \boldsymbol{z}^{(k+1)} = \arg\min \hat{\mathscr{L}}_\sigma(\boldsymbol{y}^{(k+1)}, \boldsymbol{z}, \boldsymbol{x}^{(k)}; \boldsymbol{y}^{(k)}, \boldsymbol{z}^{(k)}) + \frac{1}{2}\|\boldsymbol{z} - \boldsymbol{z}^{(k)}\|^2_{\boldsymbol{T}}$.
5: $\quad \boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \tau\sigma(\boldsymbol{A}^*\boldsymbol{y}^{(k+1)} + \boldsymbol{B}^*\boldsymbol{z}^{(k+1)} - \boldsymbol{c})$.
6: **end while**

Intuitively, the "sharper" majorization we find, the faster efficiency the algorithm will achieve. A consequent question is how further we can improve the majorizations. Unfortunately, there is no general answer since the majorization highly relies on the format of f and g. In this chapter, we only focus on the binomial regression problem with some shape constraints. We claim that a sharp majorization can be constructed for the binomial loss function, which produces the deepest descending among all quadratic type majorizations in ADMM iterations. We give the details of the derivation of sharp majorization in Section 3.2 and compare it with the regular L-majorization under a univariate case. We hence propose a **sharp-majorized ADMM** algorithm and talk about the stopping criteria in Section 3.3. We apply the sharp-majorized ADMM to a constrained logistic problem and a real data application in Section 3.4 and Section 3.5 respectively, and then present the numerical results. Finally, we give some conclusions and future works in Section 3.6.

3.2 Problem Set Up and Sharp Majorization

Consider general convex regression problem with a binomial loss function. Specifically, suppose that we have data $\{(X_i, Y_i)\}_{i=1}^n$ with X_i are scalars with ascending order and that Y_i follows a binomial distribution with size N_i and probability of success $\phi(\theta_i)$. Note that when $N_i = 1$, the problem reduces

to a standard logistic regression problem. The log-likelihood of binomial (logistic) regression is:

$$\mathscr{L}(\boldsymbol{\theta}) \quad = \quad \sum_{i=1}^{n} Y_i \log \phi(\theta_i) + (N_i - Y_i) \log(1 - \phi(\theta_i)), \tag{3.14}$$

with ϕ is the inverse of logit function $\phi(\theta) = \frac{e^{\theta}}{(1+e^{\theta})}$. Our goal is to get the maximum likelihood estimate of incidence θ_i subject to some linear constraints:

$$\begin{aligned} &\min_{\boldsymbol{\theta}} -\mathscr{L}(\boldsymbol{\theta}) \\ &\text{subject to: } \boldsymbol{M}_1\boldsymbol{\theta} \leq \boldsymbol{0} \text{ and } \boldsymbol{M}_2\boldsymbol{\theta} = \boldsymbol{0}, \end{aligned} \tag{3.15}$$

with linear equality constraint $\boldsymbol{M}_1 \in \mathbb{R}^{m_1 \times n}$ and inequality constraint $\boldsymbol{M}_2 \in \mathbb{R}^{m_2 \times n}$. Set $f(\boldsymbol{\theta}) = -\mathscr{L}(\boldsymbol{\theta})$, $\boldsymbol{M}_1\boldsymbol{\theta} = \gamma$. Problem (3.15) can be written as:

$$\begin{aligned} &\min_{\boldsymbol{\theta},\gamma} f(\boldsymbol{\theta}) + \iota_C(\gamma) \\ &\text{subject to: } \begin{bmatrix} \boldsymbol{M}_1 \\ \boldsymbol{M}_2 \end{bmatrix} \boldsymbol{\theta} + \begin{bmatrix} -\boldsymbol{I} \\ \boldsymbol{0} \end{bmatrix} \gamma = \boldsymbol{0}, \end{aligned} \tag{3.16}$$

where ι_C is the indicator function of set $C = \mathbb{R}_{-}^{m_1} = \{\gamma \in \mathbb{R}^{m_1} \mid \gamma \leq \boldsymbol{0}\}$:

$$\iota_C(\gamma) = \begin{cases} 0 & \gamma \in C, \\ \infty & \text{otherwise.} \end{cases}$$

Problem (3.16) matches the setting introduced in (3.1) with $q = \iota_C$ and g, p are zero functions. Note that f is the negative log-likelihood of binomial regression, which is convex and Lipschitz differentiable; ι_C is a proper closed convex function simply because the set C is a non-empty closed and convex set; two matrices in constraints are $\boldsymbol{A}^* = \begin{bmatrix} \boldsymbol{M}_1 \\ \boldsymbol{M}_2 \end{bmatrix}$ and $\boldsymbol{B}^* = \begin{bmatrix} -\boldsymbol{I} \\ \boldsymbol{0} \end{bmatrix}$. So we can apply the majorized iPADMM described in Algorithm 3 to solve the problem with a proper choice of $\hat{\boldsymbol{\Sigma}}_f$ and $\boldsymbol{S}$. The following of this section focuses on the choice of $\hat{\boldsymbol{\Sigma}}_f$ to construct the sharp majorization.

3.2.1 Sharp Majorization

Based on [Lee09], it is possible to search for the sharp majorization for the binomial regression loss function f. We start from the negative log-likelihood $f(\boldsymbol{\theta})$:

$$\begin{aligned} f(\boldsymbol{\theta}) \quad &= \quad \sum_{i=1}^{n} (-Y_i \log \phi(\theta_i)) - \left[(N_i - Y_i) \log(1 - \phi(\theta_i))\right] \\ &= \quad \sum_{i=1}^{n} -Y_i \log \frac{e^{\theta_i}}{1+e^{\theta_i}} - \left[N_i \log \frac{1}{1+e^{\theta_i}} - Y_i \log \frac{1}{1+e^{\theta_i}}\right] \end{aligned}$$

$$
\begin{aligned}
&= \sum_{i=1}^{n} -Y_i\left[\theta_i - \log(1+e^{\theta_i})\right] + N_i \log(1+e\,\theta_i) - Y_i \log(1+e^{\theta_i}) \\
&= \sum_{i=1}^{n} -Y_i\theta_i + N_i \log(1+e^{\theta_i}).
\end{aligned}
$$

Without loss of generality, we consider a univariate case since the objective function is the summation of n separable variables. Suppose that a single pair of observation is given by (N, Y), then the loss function is:

$$
l(\theta) = -Y\theta + N\log(1+e^{\theta}). \tag{3.17}
$$

We have shown that $l(\theta)$ is convex and Lipschitz differentiable by analyzing the second-order derivative in Chapter 4 Section 4.2.1.1. Actually, it is easy to check that $0 \le l''(\theta) = N\frac{e^{\theta}}{(1+e^{\theta})^2} \le \frac{1}{4}N$, which gives a regular L-majorization: $l(\theta) \le l(\tilde{\theta}) + l'(\tilde{\theta})(\theta-\tilde{\theta}) + \frac{L}{2}(\theta-\tilde{\theta})^2$ with $L = \frac{1}{4}N$ at some $\tilde{\theta}$.

Now we aim to find a "sharper" quadratic majorization. Consider all $a(\tilde{\theta}) > 0$ which satisfy:

$$
l(\theta) \le l(\tilde{\theta}) + l'(\tilde{\theta})(\theta-\tilde{\theta}) + \frac{1}{2}a(\tilde{\theta})(\theta-\tilde{\theta})^2, \quad \forall\theta\in\mathbb{R}, \tag{3.18}
$$

for a fixed $\tilde{\theta}$. Equivalently, we must have:

$$
a(\tilde{\theta}) \ge \frac{l(\theta) - l(\tilde{\theta}) - l'(\tilde{\theta})(\theta-\tilde{\theta})}{\frac{1}{2}(\theta-\tilde{\theta})^2}, \quad \forall\theta\neq\tilde{\theta}.
$$

Define a new function δ:

$$
\delta(\theta,\tilde{\theta}) = \frac{l(\theta) - l(\tilde{\theta}) - l'(\tilde{\theta})(\theta-\tilde{\theta})}{\frac{1}{2}(\theta-\tilde{\theta})^2}, \tag{3.19}
$$

for all $\theta \neq \tilde{\theta}$. An equivalent condition for the real value $a(\tilde{\theta})$ to exist is $\sup_{\theta} \delta(\theta,\tilde{\theta}) < \infty$. And if this is the case, we prefer to select $a(\tilde{\theta}) = \sup_{\theta} \delta(\theta,\tilde{\theta})$ since we want it to be as small as possible. We first show that the supreme is achieved over $\theta \neq \tilde{\theta}$ except for $\tilde{\theta} = 0$ in Proposition 4, and then analyze the case where $\tilde{\theta} = 0$. We also prove that the $a(\tilde{\theta}) \le L$ in Proposition 5, which suggests $a(\tilde{\theta})$ indeed gives a "sharper" majorization.

Proposition 4 (Optimality)**.** *Suppose that $\tilde{\theta} \neq 0$, the maximum value of $\delta(\theta,\tilde{\theta})$ is achieved at $s = -\tilde{\theta}$, i.e., $a(\tilde{\theta}) = \max_{\theta\neq\tilde{\theta}} \delta(\theta,\tilde{\theta}) = \delta(s,\tilde{\theta}) = \delta(-\tilde{\theta},\tilde{\theta})$.*

Please see the details in Appendix B.1. We prove this by showing that the $s = -\tilde{\theta}$ is a stationary point and the second-order condition, $\delta''(s,\tilde{\theta}) \le 0$, is satisfied. Actually, this only implies s is a local maximum. Empirical studies show that the function $\delta(\theta,\tilde{\theta})$ first increases until $s = \tilde{\theta}$ and then decreases, which produces a global maximum. We cannot give a complete proof from the perspective of monotonicity and leave this as a conjecture. For more discussions of the optimality,

please refer to [Lee09].

We hence compute the maximum value $a(\tilde{\theta})$ for $\tilde{\theta} \neq 0$:

$$\begin{aligned} a(\tilde{\theta}) &= \delta(-\tilde{\theta}, \tilde{\theta}) \\ &= \frac{l(-\tilde{\theta}) - l(\tilde{\theta}) - l'(\tilde{\theta})(\theta - \tilde{\theta})}{\frac{1}{2}(-\tilde{\theta} - \tilde{\theta})^2} \\ &= \frac{-Y(-\tilde{\theta}) + N\log(1+e^{-\tilde{\theta}}) + Y\tilde{\theta} - N\log(1+e^{\tilde{\theta}}) - l'(\tilde{\theta})(-2\tilde{\theta})}{2\tilde{\theta}^2} \\ &= \frac{2Y\tilde{\theta} + N\log\frac{1+e^{-\tilde{\theta}}}{1+e^{\tilde{\theta}}} + 2\tilde{\theta}(-Y + N\frac{e^{\tilde{\theta}}}{1+e^{\tilde{\theta}}})}{2\tilde{\theta}^2} \\ &= \frac{N}{2\tilde{\theta}^2}\left[\log\frac{1+e^{-\tilde{\theta}}}{1+e^{\tilde{\theta}}} + 2\tilde{\theta}\frac{e^{\tilde{\theta}}}{1+e^{\tilde{\theta}}}\right]. \end{aligned} \tag{3.20}$$

One problem could happen when $\tilde{\theta} = 0$, i.e., we want to construct the majorization of l at 0. The theoretical maximum can not be achieved because $-\tilde{\theta} = \tilde{\theta} = 0$ gives a 0 in the denominator in Equation (3.19). So we need to define $a(0) = \sup_{\theta} \delta(\theta, 0)$ instead. Similarly, $\delta(\theta, 0)$ approaches the supremum when θ goes to 0. And the supremum can be derived from taking the limit: the function is continuous so we compute the limit point from only one side. Specifically:

$$\begin{aligned} \lim_{\theta\to 0}\delta(\theta,0) &= \lim_{\theta\to 0}\frac{l(\theta) - l(0) - l'(0)(\theta)}{\frac{1}{2}(\theta)^2} \\ &= \lim_{\theta\to 0}\frac{N\left[\log(1+e^{\theta}) - \log 2 - \frac{\theta}{2}\right]}{\frac{1}{2}\theta^2} \\ &\overset{\text{L'Hopital}}{=} \lim_{\theta\to 0}\frac{N\left[\frac{e^{\theta}}{1+e^{\theta}} - \frac{1}{2}\right]}{\theta} \\ &\overset{\text{L'Hopital}}{=} \lim_{\theta\to 0} N\frac{e^{\theta}}{(1+e^{\theta})^2} = \frac{1}{4}N. \end{aligned} \tag{3.21}$$

It reduces to the Lipschitz constant of l'. Combine (3.20) and (3.21), we get the expression of $a(\tilde{\theta})$, which is the smallest coefficient we can achieve in front of the second-order term in Equation (3.18):

$$a(\tilde{\theta}) = \begin{cases} \frac{N}{2\tilde{\theta}^2}\left[\log\frac{1+e^{-\tilde{\theta}}}{1+e^{\tilde{\theta}}} + 2\tilde{\theta}\frac{e^{\tilde{\theta}}}{1+e^{\tilde{\theta}}}\right] & \text{if } \tilde{\theta} \neq 0, \\ \frac{1}{4}N & \text{otherwise.} \end{cases} \tag{3.22}$$

We then illustrate $a(\tilde{\theta})$ outperforms $L = \frac{1}{4}N$ by showing that $a(\tilde{\theta}) \leq \frac{1}{4}N$. Actually, the $a(\tilde{\theta})$ should be the smallest value that satisfies the inequality in (3.18) following its definition. We claim the following proposition to make it more clear.

Proposition 5. *The value $a(\tilde{\theta})$ defined in (3.22) satisfies $a(\tilde{\theta}) \leq L = \frac{1}{4}N$ for all $\tilde{\theta} \in \mathbb{R}$.*

We put the proof in Appendix B.2. The proof utilizes the first-order condition and the Mean Value Theorem.

Propositions 4 and 5 suggest that the sharp quadratic majorization of l at a fixed position $\tilde{\theta}$ is given by:

$$\hat{l}_S(\theta,\tilde{\theta}) \quad = \quad l(\tilde{\theta})+l'(\tilde{\theta})(\theta-\tilde{\theta})+\frac{1}{2}a(\tilde{\theta})(\theta-\tilde{\theta})^2. \tag{3.23}$$

Leeuw & Lange [Lee09] showed that the sharp majorization given in (3.23) satisfies $\hat{l}_S(s,\tilde{\theta})=l(s)$, i.e., $s=-\tilde{\theta}$ is a second anchor point other than $\tilde{\theta}$. Based on earlier works in [Gro03; VR05], a quadratic function majorizing a differentiable function l at two points must be a sharp majorization. We refer more details in [Lee09] and will show a simple example in Section 3.2.2.

We first visually illustrate how much we improve the regular L-majorization by replacing it with the sharp majorization from a simple example. L-majorization uses the Lipschitz constant $L=\frac{1}{4}N$ instead of $a(\tilde{\theta})$ in (3.23), namely:

$$\hat{l}_L(\theta,\tilde{\theta}) \quad = \quad l(\tilde{\theta})+l'(\tilde{\theta})(\theta-\tilde{\theta})+\frac{1}{2}L(\theta-\tilde{\theta})^2. \tag{3.24}$$

This majorization has a uniform upper bound L no matter where we construct it (no matter what value $\tilde{\theta}$ is). Suppose that $(Y,N)=(8,20)$ (actually Y does no affect the majorization property), we plot $\delta(\theta,\tilde{\theta})$ as a function of θ under different choices of $\tilde{\theta}$. We check the maximum value of $\delta(\theta,\tilde{\theta})$ among all θ, which is $a(\tilde{\theta})$, and compare it with L. With this parameter setting, the Lipschitz constant $L=\frac{1}{4}N=5$.

Figure 3.1b, 3.1c and 3.1d show the curve of $\delta(\theta,\tilde{\theta})$ with $\tilde{\theta}=1,4,8$ respectively. We see $\delta(\theta,\tilde{\theta})$ achieves the maximum $a(\tilde{\theta})$ at $\theta=-\tilde{\theta}$. The maximum values are less than L, and they become smaller when $\tilde{\theta}$ goes further away from 0. Figure 3.1a shows the particular case with $\tilde{\theta}=0$. The maximum value is not achieved. So we take the supremum as $a(\tilde{\theta})$, which coincides with L. In practice, the algorithm makes more descending as long as the majorization is not constructed at 0: the bigger $|\tilde{\theta}|$ is, the deeper descending we achieve by using the sharp majorization.

3.2.2 Comparison of Different Types of Majorizations

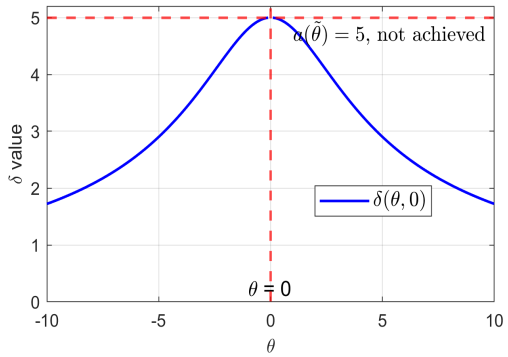
We then investigate the multivariate case. Construct the majorization of

$$f(\boldsymbol{\theta}) \quad = \quad \sum_{i=1}^{n} -Y_i\theta_i+N_i\log(1+e^{\theta_i})$$

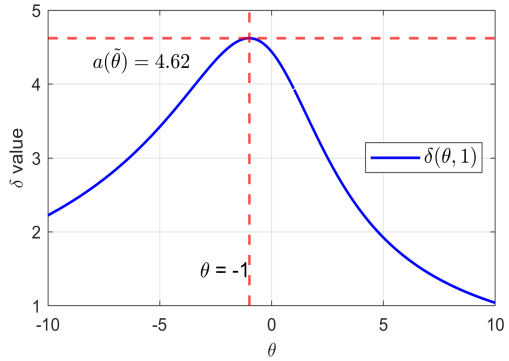
at $\tilde{\boldsymbol{\theta}}=(\tilde{\theta}_1,\tilde{\theta}_2,\ldots,\tilde{\theta}_n)^T\in\mathbb{R}^n$. There are three types of majorizations:

1. (**L-Majorization**). Use the overall uniform upper bound $L=\frac{1}{4}\max_i N_i$ of the second-order derivative for all $i=1,\ldots,n$. Specifically:

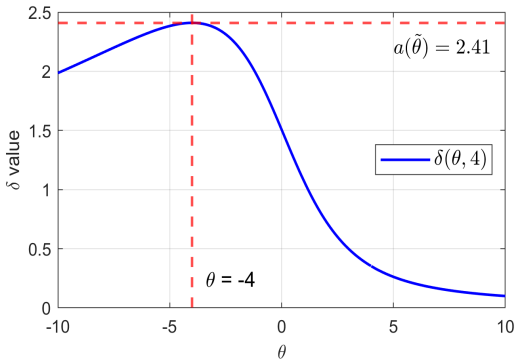
$$\nabla^2 f(\boldsymbol{\theta}) \quad = \quad \begin{pmatrix} N_1\frac{e^{\theta_1}}{(1+e^{\theta_1})^2} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & N_n\frac{e^{\theta_n}}{(1+e^{\theta_n})^2} \end{pmatrix} \quad \preceq \quad L\boldsymbol{I}. \tag{3.25}$$



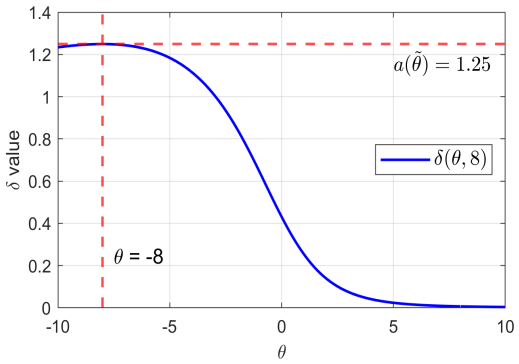
(a) Curve of $\delta(\theta, 0)$.



(b) Curve of $\delta(\theta, 1)$.



(c) Curve of $\delta(\theta, 4)$.



(d) Curve of $\delta(\theta, 8)$.

Figure 3.1 Plots of $\delta(\theta, \tilde{\theta})$ as a function of θ under different choices of $\tilde{\theta}$, which is the position where we construct the majorization. The maximum(supremum) value $a(\tilde{\theta})$ is achieved at $\theta = -\tilde{\theta}$. Panel (a): $\tilde{\theta} = 0$, maximum does not achieve, but we get $\sup \delta(\theta, 0) = \frac{1}{4}N$, which coincides with the Lipschitz constant of $L = 5$. Panel (b): $\tilde{\theta} = 1$, the maximum is achieved under $\theta = -1$. Panel (c): $\tilde{\theta} = 4$, the maximum is achieved under $\theta = -4$. Panel (d): $\tilde{\theta} = 8$, the maximum is achieved under $\theta = 1$. Sharp majorization helps the algorithm achieve deeper descending than L-majorization as long as the majorization is not constructed at 0, especially when $|\tilde{\theta}|$ is big.

So $f(\boldsymbol{\theta}) \le \hat{f}_L(\boldsymbol{\theta}, \tilde{\boldsymbol{\theta}}) = f(\tilde{\boldsymbol{\theta}}) + \langle \nabla f(\tilde{\boldsymbol{\theta}}), \boldsymbol{\theta} - \tilde{\boldsymbol{\theta}} \rangle + \frac{1}{2}\|\boldsymbol{\theta} - \tilde{\boldsymbol{\theta}}\|^2_{\hat{\boldsymbol{\Sigma}}_0}$ with

$$\hat{\boldsymbol{\Sigma}}_0 \quad := \quad L\boldsymbol{I}. \tag{3.26}$$

2. (**U-Majorization**). Use the uniform upper bound for each of the variable. We have $f(\boldsymbol{\theta}) \le \hat{f}_U(\boldsymbol{\theta}, \tilde{\boldsymbol{\theta}}) = f(\tilde{\boldsymbol{\theta}}) + \langle \nabla f(\tilde{\boldsymbol{\theta}}), \boldsymbol{\theta} - \tilde{\boldsymbol{\theta}} \rangle + \frac{1}{2}\|\boldsymbol{\theta} - \tilde{\boldsymbol{\theta}}\|^2_{\hat{\boldsymbol{\Sigma}}_1}$ with

$$\hat{\boldsymbol{\Sigma}}_1 \quad := \quad \frac{1}{4}\begin{pmatrix} N_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & N_n \end{pmatrix}. \tag{3.27}$$

3. (**Sharp Majorization**). Compute the $a(\tilde{\theta}_i)$ for each $i = 1, \dots, n$ following (3.22):

$$\boldsymbol{a}(\tilde{\boldsymbol{\theta}}) \quad = \quad \begin{pmatrix} \boldsymbol{a}(\tilde{\theta}_1) \\ \boldsymbol{a}(\tilde{\theta}_2) \\ \vdots \\ \boldsymbol{a}(\tilde{\theta}_n) \end{pmatrix}.$$

We have $f(\boldsymbol{\theta}) \le \hat{f}_S(\boldsymbol{\theta}, \tilde{\boldsymbol{\theta}}) = f(\tilde{\boldsymbol{\theta}}) + \langle \nabla f(\tilde{\boldsymbol{\theta}}), \boldsymbol{\theta} - \tilde{\boldsymbol{\theta}} \rangle + \frac{1}{2}\|\boldsymbol{\theta} - \tilde{\boldsymbol{\theta}}\|^2_{\hat{\boldsymbol{\Sigma}}_2}$.

$$\hat{\boldsymbol{\Sigma}}_2 \quad := \quad \mathrm{Diag}(\boldsymbol{a}(\tilde{\boldsymbol{\theta}})) \quad = \quad \begin{pmatrix} a(\tilde{\theta}_1) & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & a(\tilde{\theta}_n) \end{pmatrix}. \tag{3.28}$$

Matrices $\hat{\boldsymbol{\Sigma}}_0$ and $\hat{\boldsymbol{\Sigma}}_1$ are more common in majorized ADMM studies for the binomial regression problem. We claim that the sharp majorization with $\hat{\boldsymbol{\Sigma}}_2$ performs more efficiently in ADMM iterations. Figure 3.2 illustrates how sharp majorization outperforms regular L-majorization under a univariate case by plotting the f function under $(Y, N) = (8, 20)$ and corresponding majorizations at $\theta = 4$. Note that the L-majorization and the U-majorization coincide under the univariate case. Compared to the L-majorization curve in cyan, the sharp majorization (red) goes much deeper. Also, the minimizer of the sharp majorization is more closed to the minimum point of f. Furthermore, the sharp majorization has another anchor point other than $\theta = 4$, which implies it achieves the deepest descending among all quadratic type majorizations. Any "sharper" quadratic curve will violate the majorization's definition given in Section 1.2.

3.3 Sharp-Majorized ADMM

We derive the sharp-majorized ADMM algorithm based on the sharp majorization and the majorized iPADMM algorithm introduced in Algorithm 3. To apply the majorized iPADMM algorithm to

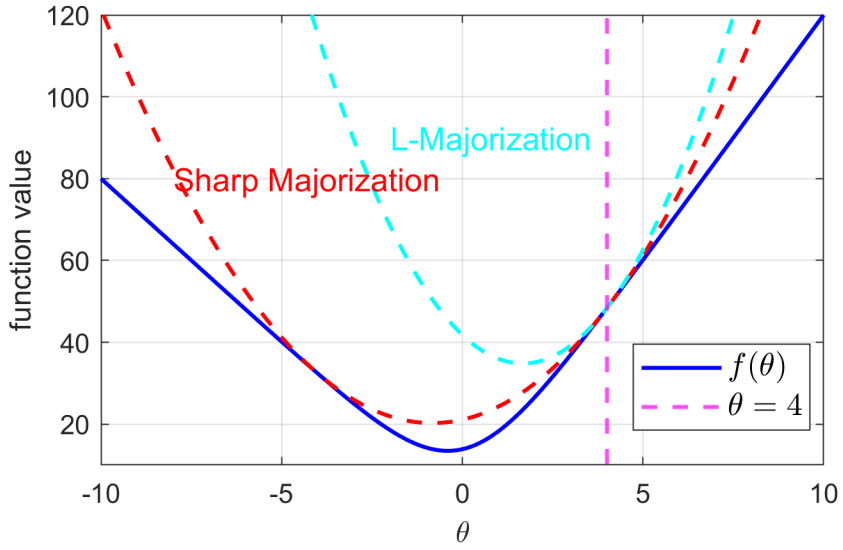


Figure 3.2 Comparison between L-majorization and sharp majorization of binomial loss function f at $\theta = 4$ under $(Y, N) = (8, 20)$. We see the sharp majorization (red) outperforms regular L-majorization (cyan) by achieving bigger descending. It it obvious that the minimizer of sharp majorization is more closed to the true minimum point of f. This suggests that we can use fewer iterations to get same convergence result by using sharp strategy.

Problem (3.16), we still need to find a linear operator $\boldsymbol{S}$ that satisfies the conditions listed in (3.13). There are multiple options without a specific rule to determine. So we will first derive the algorithm to see if there is any condition that can help us find a proper $\boldsymbol{S}$.

3.3.1 Algorithm Derivation

Denote $\hat{\boldsymbol{\Sigma}}_f$ as anyone from $\hat{\boldsymbol{\Sigma}}_0$(3.26), $\hat{\boldsymbol{\Sigma}}_1$(3.27) and $\hat{\boldsymbol{\Sigma}}_2$(3.28). We first write the augmented Lagrangian function:

$$\hat{\mathscr{L}}_\sigma(\boldsymbol{\theta}, \boldsymbol{\gamma}, \boldsymbol{x}) \quad = \quad f(\boldsymbol{\theta}) + \iota_C(\boldsymbol{\gamma}) + \langle \boldsymbol{A}^*\boldsymbol{\theta} + \boldsymbol{B}^*\boldsymbol{\gamma}\,, \boldsymbol{x}\rangle + \frac{\sigma}{2}\|\boldsymbol{A}^*\boldsymbol{\theta} + \boldsymbol{B}^*\boldsymbol{\gamma}\|^2. \tag{3.29}$$

Consequently, the majorized augmented Lagrangian function at $\tilde{\boldsymbol{\theta}}$ in (3.12):

$$\hat{\mathscr{L}}_\sigma(\boldsymbol{\theta}, \boldsymbol{\gamma}, \boldsymbol{x}; \tilde{\boldsymbol{\theta}}) = f(\tilde{\boldsymbol{\theta}}) + \langle \nabla f(\tilde{\boldsymbol{\theta}})\,, \boldsymbol{\theta} - \tilde{\boldsymbol{\theta}}\rangle + \frac{1}{2}\|\boldsymbol{\theta} - \tilde{\boldsymbol{\theta}}\|^2_{\hat{\boldsymbol{\Sigma}}_f} + \iota_C(\boldsymbol{\gamma}) + \langle \boldsymbol{A}^*\boldsymbol{\theta} + \boldsymbol{B}^*\boldsymbol{\gamma}\,, \boldsymbol{x}\rangle + \frac{\sigma}{2}\|\boldsymbol{A}^*\boldsymbol{\theta} + \boldsymbol{B}^*\boldsymbol{\gamma}\|^2. \tag{3.30}$$

Then we iterate the variables:

$$\begin{aligned}
\boldsymbol{\theta}^{(k+1)} &= \operatorname{argmin}\, \hat{\mathscr{L}}_\sigma(\boldsymbol{\theta}, \boldsymbol{\gamma}^{(k)}, \boldsymbol{x}^{(k)}; \boldsymbol{\theta}^{(k)}) + \frac{1}{2}\|\boldsymbol{\theta} - \boldsymbol{\theta}^{(k)}\|^2_{\boldsymbol{S}} \\
&= \operatorname{argmin}\, \langle \nabla f(\boldsymbol{\theta}^{(k)}), \boldsymbol{\theta}\rangle + \frac{1}{2}\|\boldsymbol{\theta} - \boldsymbol{\theta}^{(k)}\|^2_{\hat{\boldsymbol{\Sigma}}_f} + \langle \boldsymbol{x}^{(k)}, \boldsymbol{A}^*\boldsymbol{\theta}\rangle + \frac{\sigma}{2}\|\boldsymbol{A}^*\boldsymbol{\theta} + \boldsymbol{B}^*\boldsymbol{\gamma}^{(k)}\|^2 + \frac{1}{2}\|\boldsymbol{\theta} - \boldsymbol{\theta}^{(k)}\|^2_{\boldsymbol{S}} \\
&= \operatorname{argmin}\, \frac{1}{2}\left[\boldsymbol{\theta}^T(\hat{\boldsymbol{\Sigma}}_f + \boldsymbol{S} + \sigma\boldsymbol{A}\boldsymbol{A}^*)\boldsymbol{\theta}\right] - \boldsymbol{\theta}^T\left[(\hat{\boldsymbol{\Sigma}}_f + \boldsymbol{S})\boldsymbol{\theta}^{(k)} - \sigma\boldsymbol{A}\boldsymbol{B}^*\boldsymbol{\gamma}^{(k)} - \nabla f(\boldsymbol{\theta}^{(k)}) - \boldsymbol{A}\boldsymbol{x}^{(k)}\right]
\end{aligned}$$

$$= \quad (\hat{\boldsymbol{\Sigma}}_f + \boldsymbol{S} + \sigma \boldsymbol{A}\boldsymbol{A}^*)^{-1} \boldsymbol{v}^{(k)}, \tag{3.31}$$

where $\boldsymbol{v}^{(k)} = (\hat{\boldsymbol{\Sigma}}_f + \boldsymbol{S})\boldsymbol{\theta}^{(k)} - \sigma \boldsymbol{A}\boldsymbol{B}^* \boldsymbol{\gamma}^{(k)} - \nabla f(\boldsymbol{\theta}^{(k)}) - \boldsymbol{A}\boldsymbol{x}^{(k)}$ is a function of $\boldsymbol{\gamma}^{(k)}, \boldsymbol{\theta}^{(k)}, \boldsymbol{x}^{(k)}, \sigma$. Note that this problem with matrix $\hat{\boldsymbol{\Sigma}}_f + \boldsymbol{S} + \sigma \boldsymbol{A}\boldsymbol{A}^*$ in (3.31) may not be easy to solve. So a natural question is whether or not it is possible to have a diagonal matrix to get the simplest update formula for $\boldsymbol{\theta}$. We propose a simple way to construct linear operator $\boldsymbol{S}$ that satisfies conditions in (3.13) and makes the $\boldsymbol{\theta}$ update in (3.31) efficiently in Section 3.3.2.

For the γ update:

$$\begin{aligned}
\boldsymbol{\gamma}^{(k+1)} &= \arg\min \hat{\mathscr{L}}_\sigma(\boldsymbol{\theta}^{(k+1)}, \boldsymbol{\gamma}, \boldsymbol{x}^{(k)}; \boldsymbol{\theta}^{(k)}) \\
&= \arg\min \iota_C(\boldsymbol{\gamma}) + \langle \boldsymbol{x}^{(k)}, \boldsymbol{B}^* \boldsymbol{\gamma} \rangle + \frac{\sigma}{2} \|\boldsymbol{A}^* \boldsymbol{\theta}^{(k+1)} + \boldsymbol{B}^* \boldsymbol{\gamma}\|^2 \\
&= \arg\min \iota_C(\boldsymbol{\gamma}) + \boldsymbol{\gamma}^T \boldsymbol{B}\boldsymbol{x}^{(k)} + \frac{\sigma}{2} \boldsymbol{\gamma}^T \boldsymbol{B}\boldsymbol{B}^* \boldsymbol{\gamma} - \sigma \boldsymbol{\gamma}^T \boldsymbol{B}\boldsymbol{A}^* \boldsymbol{\theta}^{(k+1)} \\
&= \arg\min \iota_C(\boldsymbol{\gamma}) + \frac{\sigma}{2} \boldsymbol{\gamma}^T \boldsymbol{\gamma} - \boldsymbol{\gamma}^T (-\boldsymbol{B}\boldsymbol{x}^{(k)} + \sigma \boldsymbol{M}_1 \boldsymbol{\theta}^{(k+1)}) \qquad (3.32) \\
&= \arg\min \iota_C(\boldsymbol{\gamma}) + \frac{\sigma}{2} \|\boldsymbol{\gamma} - (-\sigma^{-1} \boldsymbol{B}\boldsymbol{x}^{(k)} + \boldsymbol{M}_1 \boldsymbol{\theta}^{(k+1)})\|_2 \\
&= \mathrm{P}_C(\boldsymbol{M}_1 \boldsymbol{\theta}^{(k+1)} - \sigma^{-1} \boldsymbol{B}\boldsymbol{x}^{(k)}). \qquad (3.33)
\end{aligned}$$

Equation (3.32) holds by simplifying the matrices multiplication. It is easy to check $\boldsymbol{B}\boldsymbol{B}^* = \boldsymbol{I}$ and $\boldsymbol{B}\boldsymbol{A}^* = -\boldsymbol{M}_1$. Actually, the γ update is only related to the equality constraint $\boldsymbol{M}_1$. Also, note that the projection in (3.33) is onto set C, which is a hyperoctant. We just need to truncate all coordinates of the projector at 0.

And the final $\boldsymbol{x}^{(k)}$ update is straightforward:

$$\boldsymbol{x}^{(k+1)} \quad = \quad \boldsymbol{x}^{(k)} + \tau\sigma(\boldsymbol{A}^* \boldsymbol{\theta}^{(k+1)} + \boldsymbol{B}^* \boldsymbol{\gamma}^{(k+1)}). \tag{3.34}$$

We see both γ and $\boldsymbol{x}$ updates are simple. To improve the efficiency of the iterations, the bottleneck is solving Problem (3.31) for updating $\boldsymbol{\theta}$.

3.3.2 Selection of Linear Operator $\boldsymbol{S}$

To apply the majorized iPADMM algorithm, we need to find the linear operator $\boldsymbol{S}$ that satisfies the conditions listed in (3.13):

$$\begin{aligned}
\tfrac{1}{2}\hat{\boldsymbol{\Sigma}}_f + \boldsymbol{S} &\succcurlyeq 0, \\
\tfrac{1}{2}\hat{\boldsymbol{\Sigma}}_f + \boldsymbol{S} + \sigma \boldsymbol{A}\boldsymbol{A}^* &\succ 0.
\end{aligned}$$

Under the binomial regression problem in (3.16), we have $\boldsymbol{A}^* = \begin{bmatrix} \boldsymbol{M}_1 \\ \boldsymbol{M}_2 \end{bmatrix}$ and $\boldsymbol{A} = [\boldsymbol{M}_1^T \ \boldsymbol{M}_2^T]$. We can select $\hat{\boldsymbol{\Sigma}}_f$ from one of its candidates in Section 3.2.2. Note that $\sigma \boldsymbol{A}\boldsymbol{A}^*$ is always positive semi-definite. For simplicity, we hope that $\hat{\boldsymbol{\Sigma}}_f + \boldsymbol{S} + \sigma \boldsymbol{A}\boldsymbol{A}^*$ is diagonal to reduce the complexity of updating $\boldsymbol{\theta}$ in

(3.31). So we can set $\boldsymbol{S}+\frac{1}{2}\hat{\boldsymbol{\Sigma}}_f$ also have off-diagonal elements that eliminate the off-diagonal entries of $\sigma\boldsymbol{A}\boldsymbol{A}^*$. To be specific:

- Get diagonal elements of $\sigma\boldsymbol{A}\boldsymbol{A}^*$, denote this vector as $\boldsymbol{c}$.
- Define matrix $\boldsymbol{Q}_1 = \text{Diag}(\boldsymbol{c})$ - $\sigma\boldsymbol{A}\boldsymbol{A}^*$. In this way, we get the negative off-diagonal entries.
- Define matrix $\boldsymbol{Q}_2 = \text{Diag}(l,\ldots,l)$ with some $l > 0$.
- Then define:

$$\boldsymbol{S}+\frac{1}{2}\hat{\boldsymbol{\Sigma}}_f = \boldsymbol{Q}_1 + \boldsymbol{Q}_2. \tag{3.35}$$

Note that we can set l large enough to make definition in (3.35) always positive semi-definite. In practice, this can be easily satisfied in most cases by setting $l = \min \boldsymbol{c}$. Also, this definition makes matrix $\hat{\boldsymbol{\Sigma}}_f + \boldsymbol{S} + \sigma\boldsymbol{A}\boldsymbol{A}^*$ diagonal, which accelerates the $\boldsymbol{\theta}$ update in (3.31).

3.3.3 Stopping Criteria

We use the KKT conditions of Problem (3.16) to develop a proper stopping criteria for the algorithm. Note that the Lagrangian function is given by:

$$f(\boldsymbol{\theta}) + \iota_C(\gamma) + \langle \boldsymbol{A}^*\boldsymbol{\theta} + \boldsymbol{B}^*\gamma\,, \boldsymbol{x}\rangle. \tag{3.36}$$

Taking derivative with respect to $\boldsymbol{\theta}$, γ and $\boldsymbol{x}$, we get KKT conditions:

$$\begin{aligned} \nabla f(\boldsymbol{\theta}) + \boldsymbol{A}\boldsymbol{x} &= \boldsymbol{0}, \\ \partial \iota_C(\gamma) + \boldsymbol{B}\boldsymbol{x} &= \boldsymbol{0}, \\ \boldsymbol{A}^*\boldsymbol{\theta} + \boldsymbol{B}^*\gamma &= \boldsymbol{0}. \end{aligned}$$

Based on these optimality conditions, we measure the accuracy of an approximate optimal point $(\boldsymbol{\theta}, \gamma, \boldsymbol{x})$ via:

$$\epsilon = \max\{\epsilon_P, \epsilon_D, \epsilon_C\}, \tag{3.37}$$

where

$$\begin{aligned} \epsilon_P &= \|\boldsymbol{A}^*\boldsymbol{\theta} + \boldsymbol{B}^*\gamma\|, \\ \epsilon_D &= \|\nabla f(\boldsymbol{\theta}) + \boldsymbol{A}\boldsymbol{x}\|, \\ \epsilon_C &= \|\gamma - \mathrm{P}_C(-\boldsymbol{B}\boldsymbol{x} + \gamma)\|, \end{aligned}$$

give the primal, dual and central residual respectively. This implies that terminating the algorithm when $\epsilon <$ `tol` with a user-defined tolerance `tol` gives a nearly optimal point.

3.3.4 Sharp-Majorized ADMM

We summarize the majorized ADMM for constrained binomial regression problem with different choices of $\hat{\boldsymbol{\Sigma}}_f$ in Algorithm 4. It reduces to the regular **L-majorized ADMM** if we use the L-majorization, i.e., set $\hat{\boldsymbol{\Sigma}}_f = \hat{\boldsymbol{\Sigma}}_0$ in (3.26); it is the majorized iPADMM algorithm introduced in [Li16; Zha20] if we select $\hat{\boldsymbol{\Sigma}}_f = \hat{\boldsymbol{\Sigma}}_1$ in (3.27), we call it the **U-majorized ADMM** since it uses the U-majorization; it becomes the **sharp-majorized ADMM** if we select $\hat{\boldsymbol{\Sigma}}_f = \hat{\boldsymbol{\Sigma}}_2$ in (3.28). Note that the sharp-majorized ADMM requires update $\hat{\boldsymbol{\Sigma}}_f$ in each iteration at the new position $\boldsymbol{\theta}^{(k)}$, which adds additional computations into the algorithm. Computing $\hat{\boldsymbol{\Sigma}}_f$ with new $\boldsymbol{\theta}^{(k)}$ only takes $\mathcal{O}(n)$, which can be easily compromised by the fewer iteration numbers of the sharp-majorized ADMM. In general, the sharp-majorized ADMM usually runs faster than the L-majorized and U-majorized ADMM even though it does extra computations if we have a simple update formula.

Algorithm 4 Majorized ADMM for shape restricted binomial regression

Require: Constraint matrix $\boldsymbol{A}^*$, $\boldsymbol{B}^*$; observation $\{(Y_i, N_i)\}_{i=1}^n$; maximum number of iterations K, tolerance `tol`; $\sigma > 0$; $\tau \in (0, (1+\sqrt{5})/2)$.

Initialize $\boldsymbol{\theta}^{(0)}, \boldsymbol{\gamma}^{(0)}, \boldsymbol{x}^{(0)}, k = 0$.
Select one version of $\hat{\boldsymbol{\Sigma}}_f$ from the candidates in (3.26), (3.27) or (3.28).
Compute $\boldsymbol{Q}_1 + \boldsymbol{Q}_2$, then compute $\boldsymbol{S}$ based on $\hat{\boldsymbol{\Sigma}}_f$ from (3.35)
while $k \le K$ **do**
    Compute $\boldsymbol{v}^{(k)} = (\hat{\boldsymbol{\Sigma}}_f + \boldsymbol{S})\boldsymbol{\theta}^{(k)} - \sigma \boldsymbol{A}\boldsymbol{B}^* \boldsymbol{\gamma}^{(k)} - \nabla f(\boldsymbol{\theta}^{(k)}) - \boldsymbol{A}\boldsymbol{x}^{(k)}$.
    Update $\boldsymbol{\theta}^{(k+1)} = (\hat{\boldsymbol{\Sigma}}_f + \boldsymbol{S} + \sigma \boldsymbol{A}\boldsymbol{A}^*)^{-1} \boldsymbol{v}^{(k)}$ by (3.31).
    Update $\boldsymbol{\gamma}^{(k+1)} = \mathrm{P}_C(\boldsymbol{M}_1 \boldsymbol{\theta}^{(k+1)} - \sigma^{-1} \boldsymbol{B}\boldsymbol{x}^{(k)})$ by (3.33).
    Update $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \tau\sigma(\boldsymbol{A}^* \boldsymbol{\theta}^{(k+1)} + \boldsymbol{B}^* \boldsymbol{\gamma}^{(k+1)})$ by (3.34).
    Compute the KKT residual ϵ by (3.37).
    if $\epsilon <$ `tol` **then**
        break
    end if
    if $\hat{\boldsymbol{\Sigma}}_f = \hat{\boldsymbol{\Sigma}}_2$ **then**
        Update $\hat{\boldsymbol{\Sigma}}_f$ at new $\boldsymbol{\theta}^{(k+1)}$, then update $\boldsymbol{S}$ by (3.35).
    end if
    Update $k = k + 1$
end while

3.4 Application I: Constrained Logistic Regression

In this section, we apply the sharp-majorized ADMM onto a constrained logistic regression problem in the following form:

$$\begin{aligned} &\min_{\boldsymbol{\zeta},\zeta_0} f(\boldsymbol{\zeta},\zeta_0)+\lambda\|\boldsymbol{\zeta}\|_1 \\ &\text{subject to: } \boldsymbol{D}\boldsymbol{\zeta}\geq \boldsymbol{d}, \end{aligned} \tag{3.38}$$

where $\boldsymbol{\zeta}\in\mathbb{R}^p, \zeta_0\in\mathbb{R}$ are variables and $\boldsymbol{D}\in\mathbb{R}^{m\times p}$, $\boldsymbol{d}\in\mathbb{R}^m$ give the linear constraints. Function $f:\mathbb{R}^{p+1}\to\mathbb{R}$ is the logistic loss, specifically:

$$f(\boldsymbol{\zeta},\zeta_0) \;=\; \frac{1}{n}\sum_{i=1}^{n}\log\big[1+\exp(-Y_i(\boldsymbol{B}_i^T\boldsymbol{\zeta}+\zeta_0))\big], \tag{3.39}$$

where $\boldsymbol{B}_i\in\mathbb{R}^p$ are the predictors and $Y_i\in\{-1,+1\}$ are the binary responses for $i=1,\ldots,n$. To simplify the notations, we set $\boldsymbol{\beta}=[\boldsymbol{\zeta}^T,\zeta_0]^T\in\mathbb{R}^{p+1}$, $\boldsymbol{P}_i=[-Y_i\boldsymbol{B}_i^T,-Y_i]^T\in\mathbb{R}^{p+1}$. Define a new design matrix $\boldsymbol{P}=[\boldsymbol{P}_1,\ldots,\boldsymbol{P}_n]\in\mathbb{R}^{(p+1)\times n}$ and rewrite the objective function $f(\boldsymbol{\beta})=\frac{1}{n}\sum_{i=1}^n\log(1+\exp(\boldsymbol{P}_i^T\boldsymbol{\beta}))$. Problem (3.38) has an equivalent format:

$$\begin{aligned} &\min_{\boldsymbol{\beta},\boldsymbol{v},\boldsymbol{w}} f(\boldsymbol{\beta})+\lambda\|\boldsymbol{v}\|_1+\iota_{\mathbb{R}^m_+}(\boldsymbol{w}) \\ &\text{subject to: } \boldsymbol{D}\boldsymbol{\zeta}-\boldsymbol{w}=\boldsymbol{d}, \\ &\qquad\qquad \boldsymbol{\zeta}-\boldsymbol{v}=0. \end{aligned} \tag{3.40}$$

Simple computations show that:

$$\nabla f(\boldsymbol{\beta}) \;=\; \frac{1}{n}\sum_{i=1}^{n}\frac{\exp(\boldsymbol{P}_i^T\boldsymbol{\beta})}{1+\exp(\boldsymbol{P}_i^T\boldsymbol{\beta})}\boldsymbol{P}_i, \tag{3.41}$$

$$\nabla^2 f(\boldsymbol{\beta}) \;=\; \frac{1}{n}\sum_{i=1}^{n}\frac{\exp(\boldsymbol{P}_i^T\boldsymbol{\beta})}{(1+\exp(\boldsymbol{P}_i^T\boldsymbol{\beta}))^2}\boldsymbol{P}_i\boldsymbol{P}_i^T \preceq \frac{1}{4n}\sum_{i=1}^{n}\boldsymbol{P}_i\boldsymbol{P}_i^T. \tag{3.42}$$

The majorized augmented Lagrangian function $\hat{\mathscr{L}}_\sigma$ in (3.12) can be written as:

$$\begin{aligned} \hat{\mathscr{L}}_\sigma(\boldsymbol{\zeta},\zeta_0,\boldsymbol{w},\boldsymbol{v},\boldsymbol{\xi},\boldsymbol{\eta};\tilde{\boldsymbol{\beta}}) \;=\;& \hat{f}(\boldsymbol{\beta},\tilde{\boldsymbol{\beta}})+\lambda\|\boldsymbol{v}\|_1+\iota_{\mathbb{R}^m_+}(\boldsymbol{w})+\langle\boldsymbol{D}\boldsymbol{\zeta}-\boldsymbol{w}-\boldsymbol{d},\boldsymbol{\xi}\rangle+\langle\boldsymbol{\zeta}-\boldsymbol{v},\boldsymbol{\eta}\rangle \\ &+\;\frac{\sigma}{2}\|\boldsymbol{D}\boldsymbol{\zeta}-\boldsymbol{w}-\boldsymbol{d}\|^2+\frac{\sigma}{2}\|\boldsymbol{\zeta}-\boldsymbol{v}\|^2, \end{aligned} \tag{3.43}$$

where $\hat{f}(\boldsymbol{\beta},\tilde{\boldsymbol{\beta}})$ is the majorization of $f(\boldsymbol{\beta})$ at $\tilde{\boldsymbol{\beta}}$:

$$\hat{f}(\boldsymbol{\beta},\tilde{\boldsymbol{\beta}}) \;=\; f(\tilde{\boldsymbol{\beta}})+\langle\nabla f(\tilde{\boldsymbol{\beta}}),\boldsymbol{\beta}-\tilde{\boldsymbol{\beta}}\rangle+\frac{1}{2}\|\boldsymbol{\beta}-\tilde{\boldsymbol{\beta}}\|^2_{\hat{\boldsymbol{\Sigma}}_f}. \tag{3.44}$$

We discuss the selections of $\hat{\boldsymbol{\Sigma}}_f$ in the next section. Based on (3.43), we solve Problem (3.40) via the

following iterative strategy:

$$\begin{aligned}
(\boldsymbol{\zeta}^{(k+1)}, \zeta_0^{(k+1)}) &= \arg\min_{\boldsymbol{\zeta}, \zeta_0} \hat{\mathscr{L}}_\sigma(\boldsymbol{\zeta}, \zeta_0, \boldsymbol{w}^{(k)}, \boldsymbol{v}^{(k)}, \boldsymbol{\xi}^{(k)}, \boldsymbol{\eta}^{(k)}; \boldsymbol{\beta}^{(k)}) + \tfrac{1}{2}\|\boldsymbol{\beta} - \boldsymbol{\beta}^{(k)}\|_{\boldsymbol{S}}^2, \\
\boldsymbol{w}^{(k+1)} &= \max\{\boldsymbol{D}^T \boldsymbol{\zeta}^{(k+1)} - \boldsymbol{d} + \boldsymbol{\xi}^{(k)}, \boldsymbol{0}\}, \\
\boldsymbol{v}^{(k+1)} &= \arg\min_{\boldsymbol{v}} \lambda\|\boldsymbol{v}\|_1 + \tfrac{\sigma}{2}\|\boldsymbol{v} - (\boldsymbol{\zeta}^{(k+1)} + \boldsymbol{\eta}^{(k)}/\sigma)\|^2, \\
\boldsymbol{\xi}^{(k+1)} &= \boldsymbol{\xi}^{(k)} + \tau\sigma(\boldsymbol{D}\boldsymbol{\zeta}^{(k+1)} - \boldsymbol{w}^{(k+1)} - \boldsymbol{d}), \\
\boldsymbol{\eta}^{(k+1)} &= \boldsymbol{\eta}^{(k)} + \tau\sigma(\boldsymbol{\zeta}^{(k+1)} - \boldsymbol{v}^{(k+1)}).
\end{aligned} \tag{3.45}$$

where $\tau \in (0, (\sqrt{5}+1)/2)$ and $\boldsymbol{S} = -\frac{1}{2}\hat{\boldsymbol{\Sigma}}_f + \mathrm{Diag}(\boldsymbol{0}, \sigma r)$ with $r > 0$ being a given real value. It is easy to check that the matrix $\hat{\boldsymbol{\Sigma}}_f$ and the linear operator $\boldsymbol{S}$ satisfy the conditions in (3.13).

And the KKT conditions are given by:

$$\begin{aligned}
\nabla f(\boldsymbol{\beta}) + [\boldsymbol{D}^T \boldsymbol{\xi} + \boldsymbol{\eta}, 0] = \boldsymbol{0}, \boldsymbol{v} - \mathrm{Prox}_{\lambda\|\cdot\|_1}(\boldsymbol{\eta} + \boldsymbol{v}) = \boldsymbol{0}, \\
\boldsymbol{w} - \mathrm{P}_{\mathbb{R}^+_m}(\boldsymbol{\xi} + \boldsymbol{w}) = \boldsymbol{0}, \boldsymbol{D}\boldsymbol{\zeta} - \boldsymbol{w} = \boldsymbol{d}, \boldsymbol{\zeta} - \boldsymbol{v} = \boldsymbol{0}.
\end{aligned}$$

Similarly, we measure the accuracy of an approximate KKT point $(\boldsymbol{\zeta}, \zeta_0, \boldsymbol{u}, \boldsymbol{v}, \boldsymbol{\xi}, \boldsymbol{\eta})$ via:

$$\epsilon_{\mathrm{KKT}} = \max\{\epsilon_P, \epsilon_D, \epsilon_C\}, \tag{3.46}$$

where

$$\begin{aligned}
\epsilon_P &= \max\big\{\|\boldsymbol{D}\boldsymbol{\zeta} - \boldsymbol{w} - \boldsymbol{d}\|, \|\boldsymbol{\zeta} - \boldsymbol{v}\|\big\}, \\
\epsilon_D &= \|\nabla f(\boldsymbol{\beta}) + [\boldsymbol{D}^T \boldsymbol{\xi} + \boldsymbol{\eta}, 0]\|, \\
\epsilon_C &= \max\big\{\|\boldsymbol{v} - \mathrm{Prox}_{\lambda\|\cdot\|_1}(\boldsymbol{\eta} + \boldsymbol{v})\|, \|\boldsymbol{w} - \mathrm{P}_{\mathbb{R}^+_m}(\boldsymbol{\xi} + \boldsymbol{w})\|\big\}.
\end{aligned}$$

3.4.1 Selection of $\hat{\boldsymbol{\Sigma}}_f$

For a majorization of $f(\boldsymbol{\beta})$ at $\tilde{\boldsymbol{\beta}}$ shown in (3.44), we construct following three candidates of $\hat{\boldsymbol{\Sigma}}_f$:

1. (**L-Majorization**). Use the largest eigenvalue of the upper bound of Hessian shown in (3.42). Specifically, compute $\lambda_{\max} = \lambda_{\max}(\frac{1}{4}\sum_{i=1}^n \boldsymbol{P}_i \boldsymbol{P}_i^T)$ where $\lambda_{\max}$ is the largest eigenvalue of a symmetric matrix. Then:

$$\hat{\boldsymbol{\Sigma}}_0 = \lambda_{\max} \boldsymbol{I}. \tag{3.47}$$

2. (**U-Majorization**). Use the upper bound of Hessian shown in (3.42):

$$\hat{\boldsymbol{\Sigma}}_1 = \frac{1}{4}\sum_{i=1}^n \boldsymbol{P}_i \boldsymbol{P}_i^T. \tag{3.48}$$

3. (**Sharp-Majorization**). Use the sharp majorization:

$$\hat{\boldsymbol{\Sigma}}_2 = \sum_{i=1}^{n} a(-\boldsymbol{P}_i^T \tilde{\boldsymbol{\beta}}) \boldsymbol{P}_i \boldsymbol{P}_i^T, \tag{3.49}$$

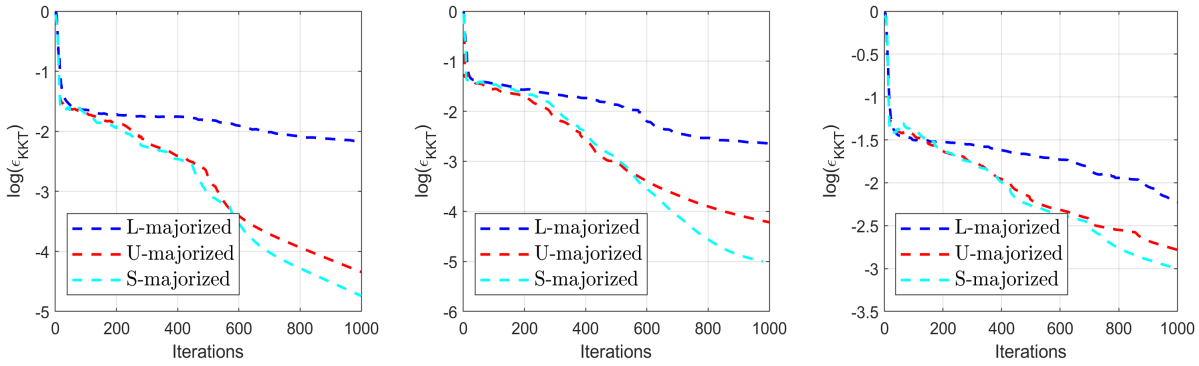
where $a(t) = \frac{2\frac{e^t}{1+e^t}-1}{2t}$ if $t \neq 0$ and $\frac{1}{4}$ otherwise, which is a smaller upper bound than $\frac{1}{4}$. It follows from [Lee09] that the sharp quadratic majorization of $\mu(x) = \log(1+\exp(-x))$ at $\tilde{x}$ is given by $\mu(\tilde{x}) + \mu'(\tilde{x}) + \frac{1}{2}a(\tilde{x})(x-\tilde{x})^2$ with $a(t)$ defined above. The derivation is a similar procedure as what we showed in Section 3.2. Please refer to [Lee09] for more details.

3.4.2 Numerical Experiments

This part tests the performances of the **L-majorized** ADMM, **U-majorized** ADMM and **S-majorized** (sharp-majorized) ADMM for the constrained logistic regression by using synthetic data. We generate matrices $\boldsymbol{B} \in \mathbb{R}^{n\times p}$, $\boldsymbol{D} \in \mathbb{R}^{m\times p}$, and $\boldsymbol{d} \in \mathbb{R}^m$ from the standard normal distribution. All three methods will be terminated when $\epsilon_{\text{KKT}} \leq 10^{-5}$ or the maximum number of iterations $(50,000)$ achieved. We select the regularization parameter $\lambda = \kappa \|\boldsymbol{B}^T \boldsymbol{b}\|_\infty / n$ with a scaling parameter $\kappa \in (0,1)$. During the simulation studies, we find that the sharp-majorized ADMM does not always converge. If this is the case, we generate a new set of matrices and re-run the algorithms. The divergent case could happen one or two times among 50 replicates. We leave this strange phenomenon as a future work.

Table 3.1 reports the number of iterations required, run time consumed of three methods to converge under different choices of σ and τ, which are also listed in the table. All values are an average of 50 replicates. We see that the sharp-majorized ADMM outperforms the other two in terms of the number of iterations: it takes the fewest iterations to achieve convergence under the same tolerance level. It shows that the sharp majorization makes the objective function achieve the biggest descending among all quadratic majorizations. Even though the problem size is small, we see some experiments almost achieve the maximum number of iterations, especially for the L-majorized ADMM under $\kappa = 10^{-3}$. Compared to the U-majorized method, sharp majorization does not give drastic improvement. Nevertheless, we still see the iteration number decrease a little bit. In terms of the run time, the L-majorized method is the slowest one. U-majorized and sharp-majorized ADMM give competitive results. The sharp-majorized method requires updating $\hat{\boldsymbol{\Sigma}}_f$ at a new position in each iteration, which suppresses the time saved from fewer iteration numbers. In this constrained logistic regression problem, the run time can not be improved due to the additional computations. The key reason is the updating $\boldsymbol{\zeta}$ in Equation (3.45) solves the problem that does not have a diagonal matrix, which results from the multivariate predictors $\boldsymbol{B}_i$ we assumed. If we can do further simplification in updating $\hat{\boldsymbol{\Sigma}}_f$ or other variables, sharp-majorized ADMM can be the most efficient one in both senses. We will see another real data application in Section 3.5.

Figure 3.3 plots the sequence of $\log(\epsilon_{\text{KKT}})$ as a function of number of iterations. We see the sharp majorization has the fastest descending among three methods. This coincides with the result shown in Table 3.1.



(a) $(n,p,m,\kappa)=(30,50,20,10^{-2})$. **(b)** $(n,p,m,\kappa)=(40,80,30,10^{-2})$. **(c)** $(n,p,m,\kappa)=(50,100,60,10^{-2})$.

Figure 3.3 Comparison between the performances of three types of majorized ADMM on synthetic data under the constrained logistic regression problem. The y-axis is the $\log(\epsilon_{\text{KKT}})$ defined in (3.46), changing with the number of iterations. We see the sharp majorization has the fastest descending in all scenarios.

Table 3.1 Comparison of the performances between L-majorized, U-majorized and S-majorized ADMM. "# Iterations" denotes the number of iterations until convergence; "Run time" denotes the run time in seconds until convergence. All results are averaged over 50 replicates.

Version	Parameters			# Iterations			Run time (sec)		
(n,p,m)	κ	σ	τ	L-majorized	U-majorized	S-majorized	L-majorized	U-majorized	S-majorized
(30,50,20)	10^{-2}	0.01	1.618	4955.00	1346.58	**1280.04**	0.73	0.20	0.21
	10^{-3}	0.01	1.618	38598.58	12394.02	**11874.30**	5.52	1.76	1.85
(40,80,30)	10^{-2}	0.01	1.2	6257.80	1947.54	**1868.36**	1.90	0.59	0.63
	10^{-3}	0.01	1.2	41819.92	18429.14	**16784.96**	10.85	4.79	4.71
(50,100,60)	10^{-2}	0.015	1.618	7619.54	3855.90	**3727.76**	3.72	1.88	1.82
	10^{-3}	0.01	1.2	46482.88	23652.98	**23502.46**	19.86	10.10	10.49

3.5 Application II: Down's Syndrome Data Set

In this section, we test the majorized ADMM algorithms by analyzing a data set of the incidence of Down's syndrome in single-year maternal age intervals [Gey91]. Down's syndrome is a generic disorder caused by trisomy of chromosome 21. It is well-known that the incidence of Down's syndrome highly depends on the age of the mother, rising rapidly after thirty, which is a natural setting to add shape constraints such as isotonicity or convexity. We use four large scale studies taking data from different locations including British Columbia (B.C.) [Tri78], Massachusetts (Mass.) [Hoo78a], New York (N.Y.) [Hoo77] and Sweden [Hoo78b].

The data are given in Table 3.2. We use the average age in the second column $\{X_i^{(t)}\}_{i=1}^{n_t}$ as the predictor, which is with ascending order. The superscript $t = 1,2,3,4$ represents the data set from B.C., Mass., N.Y. and Sweden respectively. Sequence $\{Y_i^{(t)}\}_{i=1}^{n_t}$ are the observations from the binomial regression $Y_i^{(t)} \sim \text{Binomial}(N_i^{(t)}, \phi(\theta_i^{(t)}))$. Our goal is to estimate the incidence parameter $\theta_i^{(t)}$ with assumption that the curve is convex and non-decreasing. Namely, we want to solve:

$$\begin{aligned} &\min -\mathscr{L}(\boldsymbol{\theta}^{(t)}) \\ \text{subject to: (c1) } &\frac{\theta_i^{(t)}-\theta_{i-1}^{(t)}}{X_i^{(t)}-X_{i-1}^{(t)}} \le \frac{\theta_{i+1}^{(t)}-\theta_i^{(t)}}{X_{i+1}^{(t)}-X_i^{(t)}}, \quad \forall i = 2,\dots,n_t-1, \\ \text{(c2) } &\theta_i^{(t)} \le \theta_{i+1}^{(t)}, \quad \forall i = 1,\dots,n_t-1. \end{aligned} \tag{3.50}$$

for $t = 1,2,3,4$. We express the constraints of convexity (c1) and the isotonicity (c2) in matrix notations. Problem (3.50) can be simplified as:

$$\begin{aligned} &\min_{\boldsymbol{\theta}\in\mathbb{R}^n} -\mathscr{L}(\boldsymbol{\theta}^{(t)}) \\ \text{subject to: } &\boldsymbol{C}_1^{(t)}\boldsymbol{\theta}^{(t)} \le 0, \\ &\boldsymbol{C}_2^{(t)}\boldsymbol{\theta}^{(t)} \le \boldsymbol{0}, \end{aligned} \tag{3.51}$$

with two matrices $\boldsymbol{C}_1^{(t)}$ and $\boldsymbol{C}_2^{(t)}$:

$$\boldsymbol{C}_1^{(t)} = \begin{bmatrix} -(X_3^{(t)}-X_2^{(t)}) & (X_3^{(t)}-X_1^{(t)}) & -(X_2^{(t)}-X_1^{(t)}) & 0 & \dots & 0 \\ 0 & -(X_4^{(t)}-X_3^{(t)}) & (X_4^{(t)}-X_2^{(t)}) & -(X_3^{(t)}-X_2^{(t)}) & \dots & 0 \\ 0 & 0 & \dots & \dots & \dots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & -(X_{n_t}^{(t)}-X_{n_t-1}^{(t)}) & (X_{n_t}^{(t)}-X_{n_t-2}^{(t)}) & -(X_{n_t-1}^{(t)}-X_{n_t-2}^{(t)}) \end{bmatrix},$$

Table 3.2 Incidence of Down's Syndrome by Maternal Age

Maternal	**Average**		**B.C.**		**Mass.**		**N.Y.**		**Sweden**
age	**age**(X_i)	$Y_i^{(1)}$	$N_i^{(1)}$	$Y_i^{(2)}$	$N_i^{(2)}$	$Y_i^{(3)}$	$N_i^{(3)}$	$Y_i^{(4)}$	$N_i^{(4)}$
15	15.562			1	1,364	1	5,142	0	383
16	16.543			2	3,959	4	12,524	1	1,979
15-17	17.018	16	13,555						
17	17.527			10	9,848	7	27,701	3	5,265
18	18.514	15	13,675	9	19,632	14	51,057	10	9,212
19	19.502	16	18,752	24	32,687	16	80,075	4	13,433
20	20.493	22	22,005	26	44,376	16	100,524	10	17,267
21	21.486	16	23,896	30	31,875	24	115,428	16	21,133
22	22.480	12	24,667	37	54,748	32	123,769	19	24,584
23	23.475	17	24,807	46	55,757	29	129,108	23	26,862
24	24.472	22	23,986	34	54,335	42	128,648	19	27,747
25	25.474	15	22,860	31	52,898	40	124,718	22	27,525
26	26.475	14	21,450	30	50,181	42	115,869	23	25,016
27	27.477	27	19,202	38	47,562	45	105,611	16	21,694
28	28.478	14	17,450	46	44,739	33	92,588	18	18,623
29	29.479	9	15,685	42	41,901	27	80,967	17	15,888
30	30.480	12	13,954	30	38,106	26	70,899	11	13,835
31	31.480	12	11,987	33	34,408	25	60,403	21	11,541
32	32.479	18	10,983	39	32,116	33	52,938	10	9,578
33	33.478	13	9,825	47	28,767	26	45,447	16	7,861
34	34.476	11	8,483	52	25,867	30	40,212	14	6,672
35	35.474	23	7,448	54	22,947	42	35,225	11	5,413
36	36.472	13	6,628	70	19,605	37	29,887	12	4,648
37	37.469	17	5,780	72	16,707	39	25,235	18	3,917
38	38.466	15	4,834	68	14,006	45	21,280	16	3,129
39	39.463	30	3,961	50	10,986	48	17,172	16	2,415
40	40.459	31	2,952	96	8,586	50	13,119	22	1,805
41	41.454	33	2,276	74	5,729	45	9,162	17	1,343
42	42.449	20	1,589	51	3,961	48	6,636	15	845
43	43.443	16	1,108	44	2,357	24	4,095	6	527
44	44.438	22	596	23	1,248	10	2,083	13	360
45	45.431	11	327	20	638	15	1,111	9	161
46	46.425			9	258	10	514	7	82
46-49	47.149	7	249						
47	47.419			7	103	3	183	1	35
48	48.411			2	41	1	65	2	19
49	49.410			0	13	1	22	0	7

and

$$\boldsymbol{C}_2^{(t)} = \begin{bmatrix} 1 & -1 & 0 & \dots & 0 \\ 0 & 1 & -1 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \dots & \dots & 1 & -1 \end{bmatrix}.$$

Note that $\boldsymbol{C}_1^{(t)} \in \mathbb{R}^{(n_t-2)\times n_t}$ and $\boldsymbol{C}_2^{(t)} \in \mathbb{R}^{(n_t-1)\times n_t}$. So we finally work on:

$$\begin{aligned} \min_{\boldsymbol{\theta}^{(t)}\in\mathbb{R}^{n_t}} \; & -\mathscr{L}(\boldsymbol{\theta}^{(t)}) + \iota_C(\boldsymbol{\gamma}) \\ \text{subject to: } & \begin{bmatrix} \boldsymbol{C}_1^{(t)} \\ \boldsymbol{C}_2^{(t)} \end{bmatrix} \boldsymbol{\theta}^{(t)} = \boldsymbol{\gamma}. \end{aligned} \tag{3.52}$$

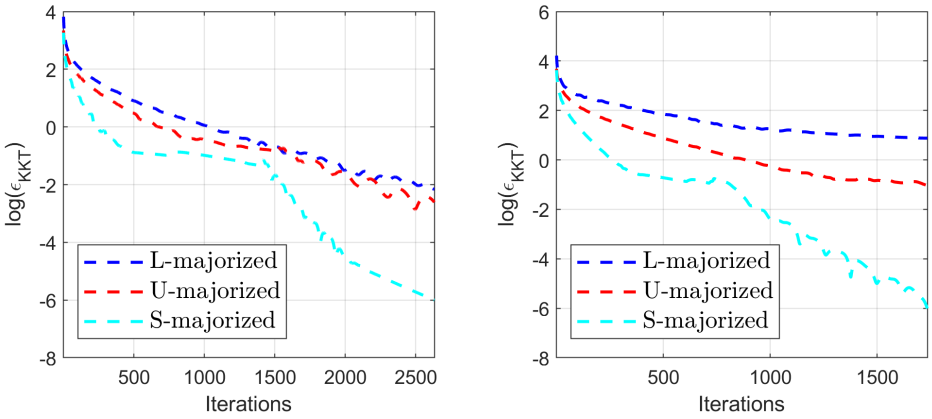
To match the general format of binomial regression in (3.16), note that $\boldsymbol{M}_1 = \begin{bmatrix} \boldsymbol{C}_1^{(t)} \\ \boldsymbol{C}_2^{(t)} \end{bmatrix}$ and $\boldsymbol{M}_2 = \boldsymbol{0}$. Hence $\boldsymbol{A} = \boldsymbol{M}_1^T$ and $\boldsymbol{A}\boldsymbol{A}^* = \boldsymbol{A}\boldsymbol{A}^T = \boldsymbol{M}_1^T\boldsymbol{M}_1$. We compare the performances of three versions of majorized ADMM algorithms: **L-majorized** ADMM with $\hat{\boldsymbol{\Sigma}}_{\boldsymbol{f}} = \hat{\boldsymbol{\Sigma}}_{\mathbf{0}}$ (3.26), **U-majorized** ADMM with $\hat{\boldsymbol{\Sigma}}_{\boldsymbol{f}} = \hat{\boldsymbol{\Sigma}}_{\mathbf{1}}$ (3.27), and **S-majorized** ADMM with $\hat{\boldsymbol{\Sigma}}_{\boldsymbol{f}} = \hat{\boldsymbol{\Sigma}}_{\mathbf{2}}$ (3.28). We report the number of iterations and corresponding run time for three methods under each of the data set with multiple choices of step size $\tau \in \{\frac{\sqrt{1}+1}{2}, \frac{\sqrt{2}+1}{2}, \frac{\sqrt{3}+1}{2}, \frac{\sqrt{4}+1}{2}\}$. We set same initial guess, which are randomly normal vectors, and same tolerance $\texttt{tol} = 10^{-6}$ for all methods. The augmented Lagrangian parameter is $\sigma = 1$ under N.Y. and Sweden data set, $\sigma = 3$ under Mass. and B.C. data set.

Table 3.3 reports the comparisons of iteration numbers and corresponding run time between the **L-majorized** ADMM, **U-majorized** ADMM and **S-majorized** (sharp-majorized) ADMM under Down's syndrome data set in different locations. Each value in the table is an average of 50 replicates with a random selection of initial values. The sharp-majorized ADMM algorithm always takes the fewest iterations to converge under the same tolerance level. It can bring at least 20% reduction in the number of iterations compared to the L-majorized or U-majorized ADMM, and sometimes can give more than 50% reduction. In terms of the run time, the sharp-majorized ADMM is also the fastest one. It is the most efficient one even though there is an additional step to update the $\hat{\boldsymbol{\Sigma}}_{\boldsymbol{f}}$ matrix in each iteration. Recall that we generates a diagonal matrix matrix for finding its inverse, which drastically simplifies the updating formula. Under this case, the time of extra computations is compensated by taking a deeper descending during the iteration, which also shows the benefits of manually designing the indefinite proximal linear operator $\boldsymbol{S}$ from Section 3.3.2.

Another insight from Table 3.2 is the trend under a specific method with increasing of step size τ. A method should take fewer iteration numbers to converge with a bigger step size starting from a fixed initial value. However, this fact is not achieved except under the sharp-majorized ADMM. The iteration numbers and run time of L-majorized and U-majorized ADMM fluctuate around a specific value. A possible reason could be the fluctuation near the optimum by using a more significant step size. The KKT residual is not constantly decreasing during the iterations. The updated variables

could go further beyond the optimum because of the big step size, which makes the algorithm harder to converge in the final stage.

Figure 3.4 plots the sequence of $\log(\epsilon_{\text{KKT}})$ as a function of number of iterations under $\tau = \frac{\sqrt{1}+1}{2}$. We see the sharp-majorized ADMM produces the fastest descending errors and first touches the threshold $\texttt{tol} = 10^{-6}$.



(a) Log error curve from B.C.

(b) Log error curve from Mass.

(c) Log error curve from N.Y.

(d) Log error curve from Sweden.

Figure 3.4 Comparison between the performances of three types of majorized ADMM on Down's syndrome data set. The y-axis is the $\log(\epsilon_{\text{KKT}})$ defined in (3.37), changing with the number of iterations. Sharp majorization has the fastest descending at all locations; it first achieves $\texttt{tol} = 10^{-6}$.

Table 3.3 Comparison of the performances between L-majorized, U-majorized and S-majorized ADMM under Down's syndrome data set. "# Iterations" denotes the number of iterations until convergence; "Run time" denotes the run time in seconds until convergence. All results are averaged over 50 replicates.

Version	# Iterations				Run time (sec)			
B.C.	$\tau=\frac{\sqrt{1}+1}{2}$	$\tau=\frac{\sqrt{2}+1}{2}$	$\tau=\frac{\sqrt{3}+1}{2}$	$\tau=\frac{\sqrt{4}+1}{2}$	$\tau=\frac{\sqrt{1}+1}{2}$	$\tau=\frac{\sqrt{2}+1}{2}$	$\tau=\frac{\sqrt{3}+1}{2}$	$\tau=\frac{\sqrt{4}+1}{2}$
L-majorized	5769.50	5729.84	5535.54	5618.24	0.5904	0.5890	0.5672	0.5746
U-majorized	4420.60	4593.88	4454.34	4419.26	0.4538	0.4738	0.4598	0.4562
S-majorized	**2829.74**	**2719.58**	**2326.70**	**2201.02**	**0.3344**	**0.3260**	**0.2802**	**0.2672**
Mass.	$\tau=\frac{\sqrt{1}+1}{2}$	$\tau=\frac{\sqrt{2}+1}{2}$	$\tau=\frac{\sqrt{3}+1}{2}$	$\tau=\frac{\sqrt{4}+1}{2}$	$\tau=\frac{\sqrt{1}+1}{2}$	$\tau=\frac{\sqrt{2}+1}{2}$	$\tau=\frac{\sqrt{3}+1}{2}$	$\tau=\frac{\sqrt{4}+1}{2}$
L-majorized	16838.32	17586.10	18006.98	18247.90	2.3114	2.4232	2.4840	2.5186
U-majorized	6247.68	6143.72	5920.18	5799.90	0.8634	0.8514	0.8143	0.8016
S-majorized	**3350.00**	**3177.28**	**2881.52**	**2872.68**	**0.5268**	**0.4992**	**0.4526**	**0.4556**
N.Y.	$\tau=\frac{\sqrt{1}+1}{2}$	$\tau=\frac{\sqrt{2}+1}{2}$	$\tau=\frac{\sqrt{3}+1}{2}$	$\tau=\frac{\sqrt{4}+1}{2}$	$\tau=\frac{\sqrt{1}+1}{2}$	$\tau=\frac{\sqrt{2}+1}{2}$	$\tau=\frac{\sqrt{3}+1}{2}$	$\tau=\frac{\sqrt{4}+1}{2}$
L-majorized	64327.38	64343.36	64397.42	64362.34	8.7664	8.6310	8.5566	8.5806
U-majorized	7046.14	6870.90	6747.30	6924.28	0.9700	0.9252	0.9032	0.9232
S-majorized	**5474.74**	**4980.96**	**4455.24**	**4434.92**	**0.8524**	**0.7680**	**0.6800**	**0.6792**
Sweden	$\tau=\frac{\sqrt{1}+1}{2}$	$\tau=\frac{\sqrt{2}+1}{2}$	$\tau=\frac{\sqrt{3}+1}{2}$	$\tau=\frac{\sqrt{4}+1}{2}$	$\tau=\frac{\sqrt{1}+1}{2}$	$\tau=\frac{\sqrt{2}+1}{2}$	$\tau=\frac{\sqrt{3}+1}{2}$	$\tau=\frac{\sqrt{4}+1}{2}$
L-majorized	32949.64	33000.80	32997.28	32983.92	4.5022	4.5054	4.5224	4.5284
U-majorized	4996.56	5040.16	6773.94	8862.94	0.6894	0.6934	0.9364	1.2184
S-majorized	**4071.64**	**3444.38**	**3088.12**	**2830.68**	**0.6422**	**0.5462**	**0.4934**	**0.4446**

3.6 Discussion and Future Work

In this chapter, we proposed a sharp-majorized ADMM algorithm for solving binomial regression problems with linear constraints. Compared to the other two existing majorized ADMM methods, the sharp-majorized technique gives a deeper descending and takes fewer iteration numbers to converge. In some cases, it also has the least run time.

However, several directions warrant further work and discussions. The critical issue is the proof for convergence in using the sharp majorization or any illustration of why the sharp-majorized ADMM sometimes does not converge in the simulation studies. Some other directions fur further investigation include: (i) a complete proof of Proposition 4, and (ii) the generalization of the sharp-majorization to other types of regression problems.

CHAPTER

4

A MAJORIZATION-MINIMIZATION GAUSS-NEWTON METHOD FOR 1-BIT MATRIX COMPLETION

We discuss another problem that can be simplified by using the Majorization-Minimization algorithm in this chapter: the problem of 1-bit matrix completion. The 1-bit matrix completion problem aims to recover the underlying low-rank matrix from an incomplete sample of binary observations, assuming a parametric probability model. The existing methods usually maximize the log-likelihood function with some relaxation of the low-rank constraint, leading to high computational cost and not getting an exact low-rank matrix. This chapter proposes a new method using the majorization to simplify the likelihood and Gauss-Newton algorithm to search for a descending direction. We also apply the factorization strategy to ensure a low-rank estimate. Numerical studies suggest that the newly proposed method beats the existing ones with better estimate accuracy and faster efficiency.

4.1 Introduction and Preliminaries

Matrix completion, which aims to recover a low-rank matrix from only part of its observations, has recently been a popular research area. The problem usually becomes more challenged in practice because the entries are always highly *quantized*, or even to a single bit ($-1/1$). This quantized (binary) observation is generated according to a probability distribution which is parameterized by the corresponding entry of the unknown low-rank matrix $\boldsymbol{M}$. We seek to recover this underlying

matrix $\boldsymbol{M}$. A typical example here is the discretized recommendation system for movies [Gol92], only a single bit rating representing "like" or "dislike" is recorded. We aim to predict the users' preference for the new movies, i.e., the underlying probability for "like" or "dislike" the missing entries. The problem of 1-bit matrix completion also arises in many other applications, including analysis of incomplete survey data, distance matrix recovery and multidimensional scaling, and quantum state tomography. Please refer to [Dav14] for more illustrations and examples. The previous methods that focus on the real-valued matrix completion are not applicable, and we need to turn out some new strategies.

4.1.1 Challenges

Note that several new challenges arise when developing the theorem and algorithm for the 1-bit matrix completion. We need some assumptions and conventions to make this problem feasible. First, the **low-rank** condition is necessary. It builds the bridge to link the information between the missing and observed entries. Secondly, even though they may be low-rank, some matrices cannot be recovered from a subset of its entries because of the spikiness. Candès & Recht [Can09] and Candes & Plan [Can10] discussed this issue under the matrix completion problem with real-valued observation (instead of binary observation). A trivial example is a matrix that consists of a single non-zero entry. For most sampling sets, we only observe zeros and have no reason to guess the matrix is non-zero. The 1-bit matrix completion has a similar issue. Suppose that a binary matrix has only one positive entry (which we might never observe) on (i, j)th position. We cannot recover the corresponding underlying matrix with $M_{ij} > 0$ since we only observe the negative entries. Davenport et al. [Dav14] proposed a typical way to deal with this scenario. We can:

> consider a reduced set of low-rank matrices by placing restrictions on the entry-wise maximum of the matrix or its singular vectors.

In other words, the matrix should be **non-spiky**. In [Dav14; Cai13; Bha15; Ni16] the authors designed the simulation studies following this principle by adding a normalization step such that $\|\boldsymbol{M}\|_\infty = 1$. Please refer to [Gro11] for more details. Thirdly, the **noise** is necessary. Consider an $m \times n$ matrix $\boldsymbol{M}$. Suppose we observe a subset $\Omega \subset [m] \times [n]$ of entries from a binary output $\boldsymbol{Y}$ following:

$$Y_{ij} = \begin{cases} +1 & \text{if } M_{ij} \geq 0, \\ -1 & \text{if } M_{ij} < 0 \end{cases} \quad \text{for } (i, j) \in \Omega. \tag{4.1}$$

This is a deterministic model without any noise. Any two matrices that have the same sign pattern will yield the same $\boldsymbol{Y}$ output. For example, assume that rank $r = 1$ and $\boldsymbol{M}$ has bounded entry-wise maximum, which satisfies the previous two conventions. It is still possible that recovering $\boldsymbol{M}$ is ill-posed even Ω is the set of all entries. Specifically, let $\boldsymbol{M} = \boldsymbol{u}\boldsymbol{v}^T$ for any vectors $\boldsymbol{u} \in \mathbb{R}^m$ and $\boldsymbol{v} \in \mathbb{R}^n$. Also, assume that there are no zero entries in $\boldsymbol{u}$ or $\boldsymbol{v}$. Let $\tilde{\boldsymbol{u}}$ and $\tilde{\boldsymbol{v}}$ be any other vectors with the same sign pattern as $\boldsymbol{u}$ and $\boldsymbol{v}$. Then both $\boldsymbol{M}$ and $\tilde{\boldsymbol{M}} = \tilde{\boldsymbol{u}}\tilde{\boldsymbol{v}}^T$ are rank 1 and give the same binary output $\boldsymbol{Y}$.

Surprisingly if we add some proper noise, the problem will become well-posed. We can only expect a good estimate of $\boldsymbol{M}$ when the noise level is in a specific range, which is an unusual characteristic under the 1-bit matrix completion. Either too big or too small noise will lead to a bad estimate. We will also see this fact from numerical experiments in Section 4.3.

4.1.2 Problem Set Up

Now that the non-noisy model is not feasible, we need to add some proper noise based on the model in (4.1). So the model becomes:

$$Y_{ij} = \begin{cases} +1 & \text{if } M_{ij}+Z_{ij} \geq 0, \\ -1 & \text{if } M_{ij}+Z_{ij} < 0 \end{cases} \quad \text{for } (i,j) \in \Omega, \tag{4.2}$$

where $\boldsymbol{Z}$ with i.i.d. entries is a matrix containing noise. The observed indices Ω are uniformly selected from $[m] \times [n]$ without replacement. To incorporate the likelihood, we get a more general way to write this observation model. Suppose that a differentiable function $f : \mathbb{R} \to [0,1]$ is given, we have binary observation $\boldsymbol{Y}$ with entry Y_{ij}:

$$Y_{ij} = \begin{cases} +1 & \text{with probability } f(M_{ij}), \\ -1 & \text{with probability } 1-f(M_{ij}) \end{cases} \quad \text{for } (i,j) \in \Omega. \tag{4.3}$$

Definitions (4.2) and (4.3) are equivalent. To see this, model in (4.2) reduces to that in (4.3) by setting $f(x) = P(Z_{1,1} \geq -x)$. Conversely, for any choice of f in (4.3), we can define $\boldsymbol{Z}$ as having i.i.d. entries drawn from a distribution whose cumulative distribution function is given by $F_Z(x) = P(z \leq x) = 1-f(-x)$. For simplicity, we will use the model defined in (4.3) in the following of this chapter. There are multiple choices of f, but we focus mainly on two possibilities:

- (Logistic model). $f(x) = \frac{e^x}{1+e^x}$.
- (Probit model). $f(x) = \Phi(x/\sigma)$ where Φ is the cumulative distribution function of a standard Gaussian distribution, i.e., $\Phi(x/\sigma) = \int_{-\infty}^{x/\sigma} \frac{1}{\sqrt{2\pi}} e^{-u^2/2} \, \mathrm{d}u$ with a noise level σ.

Set $\boldsymbol{Z} = (1+\boldsymbol{Y})/2$ where the plus and division are both entry-wise calculations, i.e., $Z_{ij} = (1+M_{ij})/2$ for each entry (i,j). It is easy to write the negative log-likelihood of this problem:

$$\ell(\boldsymbol{M}) = -\left[\sum_{(i,j)\in\Omega} Z_{ij} \log f(M_{ij}) + (1-Z_{ij}) \log\big[1-f(M_{ij})\big] \right]. \tag{4.4}$$

The traditional idea for solving the 1-bit matrix completion is to minimize the negative log-likelihood in (4.4) with some convex norm constraints to control the rank and the magnitude of the estimated matrix. Davenport et al. [Dav14] minimized (4.4) under the infinity norm constraint $\|\boldsymbol{M}\|_\infty \leq \alpha$ and the nuclear norm constraint $\|\boldsymbol{M}\|_* \leq r$ with user-defined scalar α and targeted rank

r. The authors used the non-monotone spectral projected-gradient method [Bir00] and showed that the estimate is near-optimal. Cai & Zhou [Cai13] proposed a max-norm constrained maximum likelihood estimate by using the so-called max-norm $\|\boldsymbol{M}\|_{\max} = \min_{\boldsymbol{M}=\boldsymbol{U}\boldsymbol{V}^T} \left\{\|\boldsymbol{U}\|_{2,\infty}\|\boldsymbol{V}\|_{2,\infty}\right\}$ [Lin07]. They used a special formulation of a projected gradient algorithm [Lee10] for solving large-scale convex programs involving the max-norm. Both strategies use iterative methods with projections to approach a minimizer, which yields a high computational complexity. Also, because the norm constraints are a kind of "relaxation" of the rank constraint, there is always a truncated SVD step to ensure the final estimated matrix is up to rank r. Several studies tried to get an exact low-rank estimate by using the factorization, i.e., minimize (4.4) with respect to matrices $\boldsymbol{U}, \boldsymbol{V}$ assuming that $\boldsymbol{M} = \boldsymbol{U}\boldsymbol{V}^T$. Bhaskar & Javanmard [Bha15] and Ni & Gu [Ni16] gave two typical examples. However, they still can not get rid of the high computational cost. Some other methods analyze the problem from the Bayesian perspective. Cottet & Alquier [Cot18] treated this as a classification problem and used the hinge loss to define a pseudo-posterior distribution on a set of matrices and selected one following some rule. Ravanbakhsh et al. [Rav16] searched for a maximum of the posterior inference problem that constructed based on a graphical model. One type of generalization of the 1-bit matrix completion is Positive Unlabeled (PU) Learning, in which case we only see the positive entries. The unobserved ones are either missing ones or negative values. Some PU Learning methods can also cope with the 1-bit problem under our setting, but they are not explicitly designed for it. So we do not consider such methods in this chapter. For more details, please refer to [Hsi15; Sin10; Nat15; Hay18].

All methods mentioned above have three potential drawbacks:

1. **High computational cost.** Any method based on projected gradient gives high computational complexity due to the projection and computing the gradient, especially when the problem size is huge.

2. **Sensitive to the "spikiness" of the underlying matrix.** It is valuable to test the methods when the underlying matrix has some "spikiness" since we have no way to control it practically. But the existing methods are pretty sensitive for even tiny "spikiness".

3. **Lack of testing under logistic model.** Even though almost all papers argued that the logistic model is a common choice for f, none of these studies investigated the methods' performances under the logistic model.

To focus on these drawbacks, we propose a **M**ajorization-**M**inimization **G**auss-**N**ewton (**MMGN**) method. We first construct a quadratic majorization of the negative log-likelihood. Minimizing the majorization is equivalent to finding a descending direction by using a Gauss-Newton (GN) algorithm. Numerical Experiments show that the MMGN achieves faster convergence speed and better recovery accuracy under both probit and logistic model. Also, the MMGN is not as sensitive as the previous methods to the spikiness of the matrix. We organize this chapter as follows: we restate the problem and develop the MMGN in Section 4.2; we conduct the numerical experiments in Section 4.3; we end with discussion and future work in Section 4.4.

4.2 Majorization Minimization Gauss-Newton Method

We first restate the model then introduce how we apply MM and GN methods in this section. Suppose that there is an underlying matrix $\boldsymbol{M} \in \mathbb{R}^{m\times n}$ with target rank r and a cumulative distribution function f. We have binary observation $\boldsymbol{Y}$ on the observed entries Ω, namely:

$$Y_{ij} = \begin{cases} +1 & \text{with probability } f(M_{ij}), \\ -1 & \text{with probability } 1-f(M_{ij}) \end{cases} \quad \text{for } (i,j)\in\Omega. \tag{4.5}$$

The set of the observed indices Ω is uniformly selected from the full set $[m]\times[n]$ without replacement. We focus mainly on the logistic and probit model, i.e., $f(x) = \frac{e^x}{1+e^x}$ or $f(x) = \Phi(x/\sigma)$ with Φ the cumulative distribution function of a standard Gaussian distribution. Set $\boldsymbol{Z} = (1+\boldsymbol{Y})/2$, then the negative log-likelihood of this problem is given by:

$$\ell(\boldsymbol{M}) = -\left[\sum_{(i,j)\in\Omega} Z_{ij}\log f(M_{ij}) + (1-Z_{ij})\log\big[1-f(M_{ij})\big]\right]. \tag{4.6}$$

We aim to get an estimate of the underlying matrix $\hat{\boldsymbol{M}}$ by minimizing $\ell(\boldsymbol{M})$ in Equation (4.6). MM generates a quadratic majorization $g(\boldsymbol{M}|\tilde{\boldsymbol{M}})$, which simplifies the problem by solving a sequence of least square problems. To ensure the estimates always with exact low rank, we apply the factorization technique during the iteration, i.e., assume that $\tilde{\boldsymbol{M}} = \boldsymbol{U}\boldsymbol{V}^T$ and $\boldsymbol{M} = (\boldsymbol{U}+\Delta\boldsymbol{U})(\boldsymbol{V}+\Delta\boldsymbol{V})^T$ with $\boldsymbol{U},\Delta\boldsymbol{U}\in\mathbb{R}^{m\times r}$ and $\boldsymbol{V},\Delta\boldsymbol{V}\in\mathbb{R}^{n\times r}$. We then inexactly minimize the $g(\Delta\boldsymbol{U},\Delta\boldsymbol{V}|\boldsymbol{U},\boldsymbol{V})$ with respect to $\Delta\boldsymbol{U}$ and $\Delta\boldsymbol{V}$ ignoring the crossing term. It is equivalent to a one-step GN update, which is a standard way to cope with the non-linear least square problem. We will briefly review the standard Gauss-Newton procedure before applying it to our problem.

4.2.1 Majorization-Minimization

In this section, we construct the majorization of the negative log-likelihood function $\ell(\boldsymbol{M})$ in (4.6). It is sufficient to analyze the univariate case, i.e., $-\log f(m)$ and $-\log\big[1-f(m)\big]$ for a real value $m\in\mathbb{R}$ since the objective function in (4.6) is a separable summation of all observed entries. The majorization has different formats depending on the different models we investigate. We first talk about the logistic model, which is an easier setting, and repeat the analogous analysis under the probit model.

4.2.1.1 Logistic Model

Note that the logistic probability of success and its first two order derivatives are given by:

$$f(m) = \frac{e^m}{1+e^m}, \tag{4.7}$$

$$f'(m) = \frac{e^m}{(1+e^m)^2}, \tag{4.8}$$

$$f''(m) = \frac{e^m}{(1+e^m)^3}(1-e^m). \tag{4.9}$$

Proposition 6. *Functions* $-\log f(m)$ *and* $-\log\big[1-f(m)\big]$ *are convex and Lipschitz differentiable with Lipschitz constant* $L = \frac{1}{4}$.

Proof. It is easy to show that $-\log f(m)$ is convex by checking the second-order derivative, namely:

$$\begin{aligned}
\big(-\log f(m)\big)'' &= \big(-\frac{f'(m)}{f(m)}\big)' \\
&= -\frac{f''(m)f(m)-f'(m)^2}{f(m)^2} \\
&= -\frac{(\frac{e^{\tilde{m}}}{1+e^{\tilde{m}}})(\frac{e^{\tilde{m}}}{(1+e^{\tilde{m}})^3})(1-e^{\tilde{m}})-\frac{e^{2\tilde{m}}}{(1+e^{\tilde{m}})^4}}{(\frac{e^{\tilde{m}}}{1+e^{\tilde{m}}})^2} \\
&= \frac{e^m}{(1+e^m)^2} \\
&\geq 0.
\end{aligned} \tag{4.10}$$

Another fact is that the second-derivative shown in (4.10) is bounded above by $\frac{1}{4}$. This indicates that $-\log f(m)$ is Lipschitz differentiable with Lipschitz constant $L = \frac{1}{4}$. Actually, a bounded derivative is equivalent to the Lipschitz continuity of the function. Specifically, for any $x, y \in \mathbb{R}$:

$$\begin{aligned}
\Big|\big(-\log f(x)\big)'-\big(-\log f(y)\big)'\Big| &= \Big|\frac{f'(x)}{f(x)}-\frac{f'(y)}{f(y)}\Big| \\
&= \Big|(-\frac{f'(m^*)}{f(m^*)})'\Big||x-y| \\
&= \frac{e^{\tilde{m}^*}}{(1+e^{\tilde{m}^*})^2}|x-y| \\
&\leq \frac{1}{4}|x-y|.
\end{aligned} \tag{4.11}$$

where $m^* \in \mathbb{R}$ is a scalar on the line segment between x and y. Equation (4.11) holds from the Mean Value Theorem.

We can repeat the same sequence of steps for the other term $-\log\big[1-f(m)\big]$ in the negative log-likelihood (4.6). First take the second-order derivative:

$$\begin{aligned}
\big(-\log\big[1-f(m)\big]\big)'' &= \frac{\big[1-f(m)\big]f''(m)+f'(m)^2}{\big[1-f(m)\big]^2} \\
&= \frac{e^m}{(1+e^m)^2},
\end{aligned} \tag{4.12}$$

which is the same value as the one shown in (4.10). So it has the same upper bound $L = \frac{1}{4}$. Conse-

quently, $-\log\big[1-f(m)\big]$ is also convex and Lipschitz differentiable. □

Based on the conditions provided by Proposition 6, we construct the majorization of $-\log f(m)$ at the point $\tilde{m}$ using Proposition 1:

$$-\log f(m) \;\leq\; -\log f(\tilde{m}) - \frac{f'(\tilde{m})}{f(\tilde{m})}(m-\tilde{m}) + \frac{1}{8}(m-\tilde{m})^2.$$

Note that we can simplify the coefficient in front of the first-order term:

$$\frac{f'(\tilde{m})}{f(\tilde{m})} \;=\; \frac{1}{1+e^{\tilde{m}}} \;=\; \frac{e^{-\tilde{m}}}{1+e^{-\tilde{m}}} \;=\; f(-\tilde{m}).$$

Consequently, we have:

$$\begin{aligned} -\log f(m) &\leq -\log f(\tilde{m}) - f(-\tilde{m})(m-\tilde{m}) + \frac{1}{8}(m-\tilde{m})^2 \\ &= \frac{1}{8}\big[m-(\tilde{m}+4f(-\tilde{m}))\big]^2 + c_1, \end{aligned} \tag{4.13}$$

where c_1 is a constant that does not depend on the variable m. Similarly, we also construct the majorization of $-\log\big[1-f(m)\big]$ at $\tilde{m}$:

$$\begin{aligned} -\log\big[1-f(m)\big] &\leq -\log\big[1-f(\tilde{m})\big] + \frac{f'(\tilde{m})}{1-f(\tilde{m})}(m-\tilde{m}) + \frac{1}{8}(m-\tilde{m})^2 \\ &= -\log\big[1-f(\tilde{m})\big] + f(\tilde{m})(m-\tilde{m}) + \frac{1}{8}(m-\tilde{m})^2 \\ &= \frac{1}{8}\big[m-(\tilde{m}-4f(\tilde{m}))\big]^2 + c_2, \end{aligned} \tag{4.14}$$

with another constant c_2 that does not depend on m. Combine majorizations (4.13) and (4.14) together into the negative log-likelihood (4.6). We show that the following function majorizes the negative log-likelihood under the logit model at the point $\tilde{\boldsymbol{M}}$:

$$\begin{aligned} g_l(\boldsymbol{M}|\tilde{\boldsymbol{M}}) &= \frac{1}{8}\left[\sum_{(i,j)\in\Omega} Z_{ij}\Big[M_{ij}-\big(\tilde{M}_{ij}+4f(-\tilde{M}_{ij})\big)\Big]^2 + (1-Z_{ij})\Big[M_{ij}-\big(\tilde{M}_{ij}-4f(\tilde{M}_{ij})\big)\Big]^2\right] + c \\ &= \frac{1}{8}\left[\sum_{(i,j)\in\Omega}\Big[M_{ij}-\big(\tilde{M}_{ij}+4Y_{ij}f(-Y_{ij}\tilde{\boldsymbol{M}}_{ij})\big)\Big]^2\right] + c \\ &= \frac{1}{8}\Big\|\boldsymbol{M}-\big[\tilde{\boldsymbol{M}}+4\boldsymbol{Y}\circ f(-\boldsymbol{Y}\circ\tilde{\boldsymbol{M}})\big]\Big\|_{F(\Omega)}^2 + c, \end{aligned} \tag{4.15}$$

with another constant $c = c(\tilde{\boldsymbol{M}})$ that does not depend on the variable $\boldsymbol{M}$. It changes with the point where we do the majorization $\tilde{\boldsymbol{M}}$ during MM iterations. The norm $\|\boldsymbol{A}\|_{F(\Omega)}$ represents taking the Frobenius norm of matrix $\boldsymbol{A}$ only over the observed indices, i.e., $\|\boldsymbol{A}\|_{F(\Omega)}^2 = \sum_{(i,j)\in\Omega} A_{ij}^2$. And the

notation $\circ$ is the Hammond product. We substitute the likelihood problem in (4.6) by a quadratic majorization, which is much easier to work on.

4.2.1.2 Probit Model

Under the probit model, f and its first two order derivatives with the noise level σ are:

$$f(m) = \Phi(m/\sigma) = \int_{-\infty}^{m/\sigma} \frac{1}{\sqrt{2\pi}} e^{-\frac{u^2}{2}} \,\mathrm{d}u. \tag{4.16}$$

$$f'(m) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{m^2}{2\sigma^2}}. \tag{4.17}$$

$$f''(m) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{m^2}{2\sigma^2}}(-\frac{m}{\sigma^2}). \tag{4.18}$$

Again, we repeat the analogous procedures. First, find the second-order derivative of $(-\log f(m))''$:

$$\begin{aligned}
\big(-\log f(m)\big)'' &= \big(-\frac{f'(m)}{f(m)}\big)' \\
&= -\frac{f''(m)f(m)-f'(m)^2}{f(m)^2} \\
&= \frac{f'(m)^2-f''(m)f(m)}{f(m)^2} \qquad (4.19) \\
&= \frac{\frac{1}{2\pi\sigma^2}e^{-\frac{\tilde{m}^2}{\sigma^2}} - \Phi(\tilde{m}/\sigma)\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{\tilde{m}^2}{2\sigma^2}}(-\frac{\tilde{m}}{\sigma^2})}{\Phi(\tilde{m}/\sigma)^2}. \qquad (4.20)
\end{aligned}$$

Similarly, to show $-\log f(m)$ is Lipschitz differentiable, we need to clarify its second-order derivative shown in Equation (4.19) is: 1) non-negative and 2) bounded above by a constant value. Even though it is true, this part is much harder to prove than the case under the logistic model because of the existence of the integral and exponential. Also, the upper bound is a function of the noise level σ. In this section, we assume this is the case and denote the upper bound as $t(\sigma)$. We will explain the proof procedure and how to pick $t(\sigma)$ in Section 4.2.1.3. Suppose that:

$$0 \le \frac{f'(m)^2-f(m)f''(m)}{f(m)^2} \le t(\sigma), \quad \forall m. \tag{4.21}$$

This implies that $-\log f(m)$ is convex and Lipschitz differentiable with Lipschitz constant $t(\sigma)$. Proposition 1 gives its majorization at point $\tilde{m}$:

$$\begin{aligned}
-\log f(m) &\le -\log f(\tilde{m}) + \big(-\log f(\tilde{m})\big)'(m-\tilde{m}) + \frac{t(\sigma)}{2}(m-\tilde{m})^2 \\
&= -\log f(\tilde{m}) - \frac{f'(\tilde{m})}{f(\tilde{m})}(m-\tilde{m}) + \frac{t(\sigma)}{2}(m-\tilde{m})^2 \\
&= \frac{t(\sigma)}{2}m^2 - t(\sigma)m\tilde{m} - \frac{f'(\tilde{m})}{f(\tilde{m})}m + \log f(\tilde{m}) + \frac{f'(\tilde{m})}{f(\tilde{m})}\tilde{m} + \frac{t(\sigma)}{2}\tilde{m}^2
\end{aligned}$$

$$= \frac{t(\sigma)}{2}\left[m-\left(\tilde{m}+\frac{f'(\tilde{m})}{f(\tilde{m})t(\sigma)}\right)\right]^2 + c_3(\sigma). \tag{4.22}$$

with $c_3(\sigma)$ is a constant that does not depend on m.

We repeat this procedure on $-\log\big[1-f(m)\big]$. Note that under the probit model, $f(m)=\Phi(m/\sigma)$. It is a cumulative distribution function whose density is symmetric around 0. So $1-f(m)$ produces $f(-m)$:

$$1-f(m) \;=\; 1-\Phi(\frac{m}{\sigma}) \;=\; \Phi(-\frac{m}{\sigma}) \;=\; f(-m).$$

Thus, the 2nd-order derivative of $-\log\big[1-f(m)\big]$ equals to the 2nd-order derivative of $-\log f(-m)$, specifically:

$$\begin{aligned}
\big(-\log\big[1-f(m)\big]\big)'' &= \big(\frac{f'(m)}{1-f(m)}\big)' \\
&= \frac{f''(m)\big[1-f(m)\big]+f'(m)^2}{\big[1-f(m)\big]^2} \qquad (4.23)\\
&= \frac{f''(m)f(-m)+f'(m)^2}{f(-m)^2} \\
&= \frac{-f''(-m)f(-m)+f'(-m)^2}{f(-m)^2} \qquad (4.24)\\
&= \big(-\log f(-m)\big)''.
\end{aligned}$$

Equation (4.24) holds because f' in Equation (4.17) is an even function and f'' in Equation (4.18) is an odd function. The second-order derivative of $-\log(1-f)$ evaluate at m is equivalent to the second-order derivative of $-\log f$ evaluate at $-m$. So we apply the same lower bound and upper bound defined in Equation (4.21). Thus, $-\log\big[1-f(m)\big]$ is also convex and Lipschitz differentiable. Then we can construct its majorization at $\tilde{m}$:

$$\begin{aligned}
-\log\big[1-f(m)\big] &\leq -\log\big[1-f(\tilde{m})\big]+\frac{f'(\tilde{m})}{1-f(\tilde{m})}(m-\tilde{m})+\frac{t(\sigma)}{2}(m-\tilde{m})^2 \\
&= \frac{t(\sigma)}{2}m^2-t(\sigma)m\tilde{m}+\frac{f'(\tilde{m})}{1-f(\tilde{m})}m-\log\big[1-f(\tilde{m})\big]-\frac{f'(\tilde{m})}{1-f(\tilde{m})}\tilde{m}+\frac{t(\sigma)}{2}\tilde{m}^2 \\
&= \frac{t(\sigma)}{2}\left[m-\left(\tilde{m}-\frac{f'(\tilde{m})}{\big(1-f(\tilde{m})\big)\,t(\sigma)}\right)\right]^2 + c_4(\sigma), \qquad (4.25)
\end{aligned}$$

with another constant $c_4(\sigma)$ that does not depend on m.

Combine expressions in (4.22) and (4.25) together into Equation (4.6). We show that the following function majorizes the negative log-likelihood under the probit model at the point $\tilde{\boldsymbol{M}}$:

$$
\begin{aligned}
g_p(\boldsymbol{M}|\tilde{\boldsymbol{M}}) &= \frac{t(\sigma)}{2}\left[\sum_{(i,j)\in\Omega} Z_{ij}\left(M_{ij}-\big(\tilde{M}_{ij}+\frac{f'(\tilde{M}_{ij})}{f(\tilde{M}_{ij})t(\sigma)}\big)\right)^2\right. \\
&\qquad \left. +(1-Z_{ij})\left(M_{ij}-\big(\tilde{M}_{ij}-\frac{f'(\tilde{M}_{ij})}{(1-f(\tilde{M}_{ij}))t(\sigma)}\big)\right)^2\right]+c(\sigma) \\
&= \frac{t(\sigma)}{2}\left[\sum_{(i,j)\in\Omega} Z_{ij}\left(M_{ij}-\big(\tilde{M}_{ij}+\frac{f'(\tilde{M}_{ij})}{f(\tilde{M}_{ij})t(\sigma)}\big)\right)^2\right. \\
&\qquad \left. +(1-Z_{ij})\left(M_{ij}-\big(\tilde{M}_{ij}-\frac{f'(\tilde{M}_{ij})}{f(-\tilde{M}_{ij})t(\sigma)}\big)\right)^2\right]+c(\sigma) \\
&= \frac{t(\sigma)}{2}\left[\sum_{(i,j)\in\Omega}\left(M_{ij}-\big(\tilde{M}_{ij}+\frac{Y_{ij}f'(\tilde{M}_{ij})}{f(Y_{ij}\tilde{M}_{ij})t(\sigma)}\big)\right)^2\right]+c(\sigma) \\
&= \frac{t(\sigma)}{2}\left\|\boldsymbol{M}-(\tilde{\boldsymbol{M}}+\boldsymbol{Y}\circ f'(\tilde{\boldsymbol{M}})/(t(\sigma)f(\boldsymbol{Y}\circ\tilde{\boldsymbol{M}})))\right\|^2_{F(\Omega)}+c(\sigma), \qquad (4.26)
\end{aligned}
$$

where $c(\sigma)=c(\sigma,\tilde{\boldsymbol{M}})$ is a constant that does not depend on $\boldsymbol{M}$ and the division inside the Frobenius norm in Equation (4.26) represents the element-wise division, i.e., $\boldsymbol{A}=f(\boldsymbol{B})/f(\boldsymbol{C})$ with $A_{ij}=f(B_{ij})/f(C_{ij})$.

4.2.1.3 Upper Bound under Probit Model

This section will show that the second-order derivative of $-\log\big[1-f(m)\big]$ in (4.19) is non-negative with an upper bound $t(\sigma)$ under the probit model. Namely:

$$
0 \;\le\; \frac{f'(m)^2-f(m)f''(m)}{f(m)^2} \;\le\; t(\sigma), \quad \forall m.
$$

For simplicity to investigate its property, we denote the negative of second-order derivative as a new function ϕ_σ. It is a univariate function with a fixed choice of noise level $\sigma>0$. To avoid the abuse of the variable name, we replace m with x in the following analysis in this section. Then:

$$
\begin{aligned}
\phi_\sigma(x) &= -\left(\frac{f'(x)^2-f(x)f''(x)}{f(x)^2}\right) \\
&= \frac{\Phi(x/\sigma)\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{x^2}{2\sigma^2}}(-\frac{x}{\sigma^2})-\frac{1}{2\pi\sigma^2}e^{-\frac{x^2}{\sigma^2}}}{\Phi(x/\sigma)^2} \\
&= \frac{\Phi(x/\sigma)f''(x)-f'(x)^2}{\Phi(x/\sigma)^2} \\
&= \frac{f''(x)}{\Phi(x/\sigma)}-\left[\frac{f'(x)}{\Phi(x/\sigma)}\right]^2. \qquad (4.27)
\end{aligned}
$$

It is sufficient to show that the function $\phi_\sigma(x)$ is non-positive with a lower bound. We claim this by proving the following proposition.

Proposition 7. *With a fixed $\sigma > 0$, the function $\phi_\sigma(x)$ satisfies the following conditions:*

1. *non-decreasing,*
2. $\lim_{x\to+\infty} \phi_\sigma(x) = 0$,
3. $\lim_{x\to-\infty} \phi_\sigma(x) = -\frac{1}{\sigma^2}$.

The second result is straightforward. Taking the limit under $x \to \infty$, we can easily detect that the numerators in Equation (4.27) go to 0 and the denominators go to 1. We put the discussions of the other two conclusions in Appendix C.1. Note that $\phi_\sigma(x)$ is a continuous function with any $\sigma > 0$. So if it goes from $-\frac{1}{\sigma^2}$ to 0 without any decreasing, the function value is bounded between $(-\frac{1}{\sigma^2}, 0)$. Consequently, the second-order derivative-which equals to the negative of $\phi_\sigma(x)$-is non-negative and bounded above by $\frac{1}{\sigma^2}$. Therefore, we show that $-\log f(m)$ and $-\log\big[1-f(m)\big]$ is convex and Lipschitz differentiable under the probit model; and find the upper bound $t(\sigma) = \frac{1}{\sigma^2}$ in Equation (4.26).

4.2.2 Gauss-Newton Step

We first implement the GN step in our setting and then discuss its connection to the standard GN algorithm. For simplicity, we define a new matrix $\boldsymbol{X}$ based on different majorizations shown in (4.15) and (4.26). Specifically, set:

$$\boldsymbol{X} = \begin{cases} 4\boldsymbol{Y} \circ f(-\boldsymbol{Y} \circ \tilde{\boldsymbol{M}}) & \text{under the logistic model,} \\ \boldsymbol{Y} \circ f'(\tilde{\boldsymbol{M}}) / (t(\sigma) f(\boldsymbol{Y} \circ \tilde{\boldsymbol{M}})) & \text{under the probit model.} \end{cases} \tag{4.28}$$

Note that matrix $\boldsymbol{X}$ is known since it is a function of the binary observation $\boldsymbol{Y}$ and the anchor point $\tilde{\boldsymbol{M}}$. Denote $g(\boldsymbol{M}|\tilde{\boldsymbol{M}})$ as $g_l(\boldsymbol{M}|\tilde{\boldsymbol{M}})$ in (4.15) or $g_p(\boldsymbol{M}|\tilde{\boldsymbol{M}})$ in (4.26) depending on the model we use. Then minimizing the majorization $g(\boldsymbol{M}|\tilde{\boldsymbol{M}})$ with low-rank constraint up to r is equivalent to solving:

$$\underset{\text{rank}(\boldsymbol{M})\le r}{\arg\min}\ g(\boldsymbol{M}|\tilde{\boldsymbol{M}}) \quad = \quad \underset{\text{rank}(\boldsymbol{M})\le r}{\arg\min}\ \|\boldsymbol{M} - \tilde{\boldsymbol{M}} - \boldsymbol{X}\|^2_{F(\Omega)}. \tag{4.29}$$

This is a least square problem. Instead of using the relaxation of the rank constraint by adding norm penalties [Cai13; Dav14], we apply the factorization method. Recall that $\tilde{\boldsymbol{M}} = \boldsymbol{U}\boldsymbol{V}^T$ and $\boldsymbol{M} = (\boldsymbol{U} + \Delta\boldsymbol{U})(\boldsymbol{V} + \Delta\boldsymbol{V})^T$ for $\boldsymbol{U}, \Delta\boldsymbol{U} \in \mathbb{R}^{m\times r}$ and $\boldsymbol{V}, \Delta\boldsymbol{V} \in \mathbb{R}^{n\times r}$. Matrices $\boldsymbol{U}$ and $\boldsymbol{V}$ are known. We aim to get optimum $\Delta\boldsymbol{U}^+$ and $\Delta\boldsymbol{V}^+$ to update the variable from $\tilde{\boldsymbol{M}}$ to the next position $\boldsymbol{M}^+ = (\boldsymbol{U} + \Delta\boldsymbol{U}^+)(\boldsymbol{V} + \Delta\boldsymbol{V}^+)^T$. With factorization, the rank of $\tilde{\boldsymbol{M}}$ and $\boldsymbol{M}^+$ will not exceed r during the iterations. In practice, the rank is seldom strictly less than r, and we usually get an estimate with the exact rank r. We do not keep the cross term to get rid of the non-linearity:

$$g(\Delta\boldsymbol{U}, \Delta\boldsymbol{V}|\boldsymbol{U}, \boldsymbol{V}) \quad \approx \quad \|\boldsymbol{U}\boldsymbol{V}^T + \boldsymbol{U}\Delta\boldsymbol{V}^T + \Delta\boldsymbol{U}\boldsymbol{V}^T - \boldsymbol{U}\boldsymbol{V}^T - \boldsymbol{X}\|^2_{F(\Omega)}$$

$$
\begin{aligned}
&= \|\boldsymbol{X}-\boldsymbol{U}\Delta\boldsymbol{V}^T-\Delta\boldsymbol{U}\boldsymbol{V}^T\|^2_{F(\Omega)}\\
&= \|\boldsymbol{X}-\boldsymbol{U}\Delta\boldsymbol{V}^T-\Delta\boldsymbol{U}\boldsymbol{V}^T\|^2_{F(\Omega)}.
\end{aligned}
$$

We can express this as a standard least square problem by doing vectorization. Specifically, set $\boldsymbol{x}=\text{vec}(\boldsymbol{X})$, $\Delta\boldsymbol{u}=\text{vec}(\Delta\boldsymbol{U})$ and $\Delta\tilde{\boldsymbol{v}}=\text{vec}(\Delta\boldsymbol{V}^T)$. Then the optimization problem in (4.29) can be written as:

$$
\begin{aligned}
\underset{\text{rank}(\boldsymbol{M})\le r}{\arg\min}\ \|\boldsymbol{M}-\tilde{\boldsymbol{M}}-\boldsymbol{X}\|^2_{F(\Omega)} &\approx \underset{\Delta\boldsymbol{U},\Delta\boldsymbol{V}}{\arg\min}\ \|\boldsymbol{X}-\boldsymbol{U}\Delta\boldsymbol{V}^T-\Delta\boldsymbol{U}\boldsymbol{V}^T\|^2_{F(\Omega)}\\
&= \underset{\Delta\boldsymbol{u},\Delta\tilde{\boldsymbol{v}}}{\arg\min}\ \|\text{vec}(\boldsymbol{X})-\text{vec}(\boldsymbol{U}\Delta\boldsymbol{V}^T)-\text{vec}(\Delta\boldsymbol{U}\boldsymbol{V}^T)\|^2_{\tilde{\Omega}}\\
&= \underset{\Delta\boldsymbol{u},\Delta\tilde{\boldsymbol{v}}}{\arg\min}\ \|\text{vec}(\boldsymbol{X})-\text{vec}(\boldsymbol{U}\Delta\boldsymbol{V}^T\boldsymbol{I})-\text{vec}(\boldsymbol{I}\Delta\boldsymbol{U}\boldsymbol{V}^T)\|^2_{\tilde{\Omega}}\\
&= \underset{\Delta\boldsymbol{u},\Delta\tilde{\boldsymbol{v}}}{\arg\min}\ \|\boldsymbol{x}-(\boldsymbol{I}\otimes\boldsymbol{U})\text{vec}(\Delta\boldsymbol{V}^T)-(\boldsymbol{V}\otimes\boldsymbol{I})\text{vec}(\Delta\boldsymbol{U})\|^2_{\tilde{\Omega}} \qquad (4.30)\\
&= \underset{\Delta\boldsymbol{u},\Delta\tilde{\boldsymbol{v}}}{\arg\min}\ \|\boldsymbol{x}-(\boldsymbol{I}\otimes\boldsymbol{U})\Delta\tilde{\boldsymbol{v}}-(\boldsymbol{V}\otimes\boldsymbol{I})\Delta\boldsymbol{u}\|^2_{\tilde{\Omega}}\\
&= \underset{\Delta\boldsymbol{u},\Delta\tilde{\boldsymbol{v}}}{\arg\min}\ \frac{1}{2}\|\boldsymbol{x}-(\boldsymbol{I}\otimes\boldsymbol{U})\Delta\tilde{\boldsymbol{v}}-(\boldsymbol{V}\otimes\boldsymbol{I})\Delta\boldsymbol{u}\|^2_{\tilde{\Omega}}. \qquad (4.31)
\end{aligned}
$$

The equality in (4.30) holds from a property of the Kronecker product: $\text{vec}(\boldsymbol{ABC})=(\boldsymbol{C}^T\otimes\boldsymbol{A})\text{vec}(\boldsymbol{B})$ for any matrices $\boldsymbol{A},\boldsymbol{B},\boldsymbol{C}$ with suitable sizes. The vector $\boldsymbol{x}$ is with length mn, but only $|\Omega|$ of them will be used to exclude the missing part. Consequently, the norm $\|\ \|_{\tilde{\Omega}}$ represents the two norm of a vector over the set $\tilde{\Omega}$, which is the set with the one-way expression of observed indices in Ω after taking the column-wise major vectorization. Specifically, for a pair of entry $(i,j)\in\Omega$, we have corresponding index $(m-1)j+i\in\tilde{\Omega}$. In other words, (i,j)th entry of matrix $\boldsymbol{X}$ shows up in the $\big((m-1)j+i\big)$th position in $\boldsymbol{x}=\text{vec}(X)$. Define $\boldsymbol{A}=\boldsymbol{I}\otimes\boldsymbol{U}$ and $\boldsymbol{B}=\boldsymbol{V}\otimes\boldsymbol{I}$, Problem (4.31) is:

$$
\underset{\Delta\boldsymbol{u},\Delta\tilde{\boldsymbol{v}}}{\arg\min}\ \frac{1}{2}\|\boldsymbol{x}-\boldsymbol{B}\Delta\boldsymbol{u}-\boldsymbol{A}\Delta\tilde{\boldsymbol{v}}\|^2_{\tilde{\Omega}} = \underset{\boldsymbol{\theta}}{\arg\min}\ \frac{1}{2}\|\boldsymbol{x}-\boldsymbol{J}\boldsymbol{\theta}\|^2_{\tilde{\Omega}}, \qquad (4.32)
$$

where

$$
\boldsymbol{J} = [\boldsymbol{B}\,|\,\boldsymbol{A}],\quad \boldsymbol{\theta} = \begin{pmatrix}\Delta\boldsymbol{u}\\ \Delta\tilde{\boldsymbol{v}}\end{pmatrix} = \begin{pmatrix}\text{vec}(\Delta\boldsymbol{U})\\ \text{vec}(\Delta\boldsymbol{V}^T)\end{pmatrix}. \qquad (4.33)
$$

We can further simplify the notations in Problem (4.32) by focusing on the part of $\boldsymbol{x}$ and matrix $\boldsymbol{J}$ whose (row) indices are in $\tilde{\Omega}$. For simplicity, define: $\boldsymbol{J}=\boldsymbol{J}[\tilde{\Omega},]$ and $\boldsymbol{x}=\boldsymbol{x}[\tilde{\Omega}]$ where $\boldsymbol{J}[\tilde{\Omega},]$ represents the rows of matrix $\boldsymbol{J}$ whose indices are in $\tilde{\Omega}$; and $\boldsymbol{x}[\tilde{\Omega}]$ represents the elements of vector $\boldsymbol{x}$ whose indices are in $\tilde{\Omega}$. Thus, the inexact MM update direction is given by the least-norm minimizer of the following least square problem:

$$
\frac{1}{2}\|\boldsymbol{x}-\boldsymbol{J}\boldsymbol{\theta}\|^2. \qquad (4.34)
$$

Simple regression knowledge tells us the solution is: $\boldsymbol{\theta}^+ = (\boldsymbol{J}^T\boldsymbol{J})^-\boldsymbol{J}^T\boldsymbol{x} = \begin{pmatrix}\Delta\boldsymbol{u}^+\\ \Delta\tilde{\boldsymbol{v}}^+\end{pmatrix}$. We then update the variables:

$$\begin{pmatrix}\text{vec}(\boldsymbol{U}^+)\\ \text{vec}(\boldsymbol{V}^{+T})\end{pmatrix} = \begin{pmatrix}\text{vec}(\boldsymbol{U})\\ \text{vec}(\boldsymbol{V}^T)\end{pmatrix} + (\boldsymbol{J}^T\boldsymbol{J})^-\boldsymbol{J}^T\boldsymbol{x}. \tag{4.35}$$

Alternatively, we can recover the estimated $\Delta\boldsymbol{U}^+$ and $\Delta\boldsymbol{V}^+$ by reshaping $\Delta\boldsymbol{u}^+$ and $\Delta\tilde{\boldsymbol{v}}^+$ into their matrix expressions, respectively. The update in terms of the matrices provides:

$$\boldsymbol{U}^+ = \boldsymbol{U} + \Delta\boldsymbol{U}^+, \quad \boldsymbol{V}^+ = \boldsymbol{V} + \Delta\boldsymbol{V}^+. \tag{4.36}$$

Then $\boldsymbol{M}^+ = \boldsymbol{U}^+(\boldsymbol{V}^+)^T$. We then treat $\boldsymbol{M}^+$ as the new $\tilde{\boldsymbol{M}}$ and repeat the same procedure. We use the pseudo-inverse because $\boldsymbol{J}^T\boldsymbol{J}$ in Equation (4.35) could be rank-deficient. There might be multiple solutions here, but we select the one with the least norm. Please refer to [Bau21] for more details of the advantages of using the least-norm solution.

4.2.2.1 Connection to Standard Gauss-Newton Algorithm

As a modification of Newton's method, the Gauss-Newton algorithm is a common strategy to solve a non-linear least square problem. It minimizes the sum of squares:

$$s(\boldsymbol{p}) = \sum_i r_i(\boldsymbol{p})^2, \tag{4.37}$$

where r_i are usually called residuals, and $\boldsymbol{p}$ is the variable. With an initialization $\boldsymbol{p}^{(0)}$, GN algorithm proceeds by the iterations:

$$\boldsymbol{p}^{(k+1)} = \boldsymbol{p}^{(k)} - \left[\boldsymbol{J}(\boldsymbol{p}^{(k)})^T\boldsymbol{J}(\boldsymbol{p}^{(k)})\right]^{-1}\boldsymbol{J}(\boldsymbol{p}^{(k)})^T r(\boldsymbol{p}^{(k)}), \tag{4.38}$$

where $\boldsymbol{r}(\boldsymbol{p}^{(k)})$ is the vector consisting of all residuals $r_i(\boldsymbol{p}^{(k)})$; $\boldsymbol{J}(\boldsymbol{p}^{(k)})$ is the Jacobian matrix evaluate at $\boldsymbol{p}^{(k)}$ with entries $\boldsymbol{J}_{ij} = \frac{\partial r_i(\boldsymbol{p}^{(k)})}{\partial p_j}$; and p_j is the jth coordinate of the variable $\boldsymbol{p}$. Note that matrix $\left[\boldsymbol{J}(\boldsymbol{p}^{(k)})^T\boldsymbol{J}(\boldsymbol{p}^{(k)})\right]^{-1}\boldsymbol{J}(\boldsymbol{p}^{(k)})$ approximates the inverse of Hessian to scale the descending direction. Go back to our setting in (4.29); we aim to solve the following problem:

$$\begin{aligned}\underset{\text{rank}(\boldsymbol{M})\le r}{\arg\min}\ \|\boldsymbol{M}-\tilde{\boldsymbol{M}}-\boldsymbol{X}\|_{F(\Omega)}^2 &= \underset{\boldsymbol{U}\in\mathbb{R}^{m\times r},\boldsymbol{V}\in\mathbb{R}^{n\times r}}{\arg\min}\ \|\boldsymbol{U}\boldsymbol{V}^T-(\tilde{\boldsymbol{M}}+\boldsymbol{X})\|_{F(\Omega)}^2\\ &= \underset{\boldsymbol{U}\in\mathbb{R}^{m\times r},\boldsymbol{V}\in\mathbb{R}^{n\times r}}{\arg\min}\ \|\text{vec}(\boldsymbol{U}\boldsymbol{V}^T)-\text{vec}(\tilde{\boldsymbol{M}}+\boldsymbol{X})\|_{\tilde{\Omega}}^2.\end{aligned} \tag{4.39}$$

This generates a non-linear least square problem with respect to matrices $\boldsymbol{U}$ and $\boldsymbol{V}$. Set $\boldsymbol{C} = (\tilde{\boldsymbol{M}} + \boldsymbol{X})$, we write the sum of squares in (4.37) as:

$$s(\boldsymbol{p}) = \sum_{i=1}^{|\tilde{\Omega}|} \text{vec}(\boldsymbol{U}\boldsymbol{V}^T)_i - \text{vec}(\boldsymbol{C})_i \tag{4.40}$$

$$:= \sum_{i=1}^{|\tilde{\Omega}|} r_i(\boldsymbol{p}), \tag{4.41}$$

where $\boldsymbol{p}$ is the variable defined as:

$$\boldsymbol{p} = \begin{pmatrix} \boldsymbol{u} \\ \tilde{\boldsymbol{v}} \end{pmatrix} = \begin{pmatrix} \text{vec}(\boldsymbol{U}) \\ \text{vec}(\boldsymbol{V}^T) \end{pmatrix}.$$

Simple calculations show that the Jacobian matrix $\boldsymbol{J}(\boldsymbol{p})$ with entries $\boldsymbol{J}_{ij} = \frac{\partial r_i(\boldsymbol{p})}{\partial p_j}$ in Equation (4.38) coincides with the design matrix shown in (4.33),i.e., $\boldsymbol{J} = [\boldsymbol{V} \otimes \boldsymbol{I} \mid \boldsymbol{I} \otimes \boldsymbol{U}]$. So if we assume current variables $\boldsymbol{U}$ and $\boldsymbol{V}$, or equivalently, $\boldsymbol{p}$ is known, then one-step GN produces:

$$\boldsymbol{p}^+ = \boldsymbol{p} - (\boldsymbol{J}^T\boldsymbol{J})^{-1}\boldsymbol{J}^T\boldsymbol{r}(\boldsymbol{p}). \tag{4.42}$$

Note that $\tilde{\boldsymbol{M}} = \boldsymbol{U}\boldsymbol{V}^T$ is also known in our setting, then $\boldsymbol{r}(\boldsymbol{p}) = \text{vec}(\boldsymbol{U}\boldsymbol{V}^T) - \text{vec}(\boldsymbol{C}) = -\text{vec}(\boldsymbol{X}) = -\boldsymbol{x}$ over the set of observed entries $\tilde{\Omega}$. Then the update in (4.42) becomes:

$$\boldsymbol{p}^+ = \boldsymbol{p} + (\boldsymbol{J}^T\boldsymbol{J})^{-1}\boldsymbol{J}^T\boldsymbol{x}, \tag{4.43}$$

which is the same update formula as the one shown in Equation (4.35). In summary, instead of exactly solving Problem (4.29) or (4.39), we only do a one-step update by using the GN algorithm. Then we do the majorization at this new position and repeat the same procedures. We provide more details of the equivalence of the Jacobian matrices in Appendix C.2.

4.2.3 Backtracking and Step Size

We use the Armijo condition [Arm66] to ensure the monotonicity of the MM procedure. This is a prerequisite for ensuring the convergence of the MM algorithm to a stationary point, specifically identifying cluster points. It picks the step size that gives an adequate descending of the objective function ℓ. Suppose that current iteration gives $\tilde{M} = \boldsymbol{U}\boldsymbol{V}^T$; the objective $\ell(\boldsymbol{M}) = \ell(\boldsymbol{U}, \boldsymbol{V})$; the descending direction $\boldsymbol{\theta}^+ = \begin{pmatrix} \Delta\boldsymbol{u}^+ \\ \Delta\tilde{\boldsymbol{v}}^+ \end{pmatrix}$ with $\Delta\boldsymbol{u}^+ = \text{vec}(\Delta\boldsymbol{U}^+)$ and $\Delta\tilde{\boldsymbol{v}}^+ = \text{vec}(\Delta\boldsymbol{V}^{+T})$. We define $\boldsymbol{\theta} = \begin{pmatrix} \Delta\boldsymbol{u} \\ \Delta\tilde{\boldsymbol{v}} \end{pmatrix} = \begin{pmatrix} \text{vec}(\boldsymbol{U}) \\ \text{vec}(\boldsymbol{V}^T) \end{pmatrix}$ and write $\ell(\boldsymbol{U}, \boldsymbol{V}) = \ell(\boldsymbol{\theta})$. Take $\beta \in (0,1)$ and take α to be a step-length parameter. We accept α if:

$$\ell(\boldsymbol{U} + \alpha\Delta\boldsymbol{U}^+, \boldsymbol{V} + \alpha\Delta\boldsymbol{V}^+) \leq \ell(\boldsymbol{U}, \boldsymbol{V}) + \beta\alpha\langle\nabla\ell(\boldsymbol{\theta}), \boldsymbol{\theta}^+\rangle, \tag{4.44}$$

Simple computation shows that the scalar product on the right hand side finally equals to:

$$\langle \nabla \ell(\boldsymbol{\theta}), \boldsymbol{\theta}^+ \rangle \quad = \quad -\Big\langle \boldsymbol{Y} \circ \frac{f'(\tilde{\boldsymbol{M}})}{f(\boldsymbol{Y} \circ \tilde{\boldsymbol{M}})}, \Delta \boldsymbol{U}^+ \boldsymbol{V}^T + \boldsymbol{U}(\Delta \boldsymbol{V}^+)^T \Big\rangle_\Omega. \tag{4.45}$$

where $\langle \boldsymbol{A}, \boldsymbol{B} \rangle_\Omega = \sum_{(i,j)\in\Omega} A_{ij} B_{ij}$ represents the scalar product of two matrices over observed entries Ω. We implement the computations of Equation (4.45) in Appendix C.3. Thus, the Armijo condition accepts a step-length α if

$$\ell(\boldsymbol{U} + \alpha \Delta \boldsymbol{U}^+, \boldsymbol{V} + \alpha \Delta \boldsymbol{V}^+) \quad \leq \quad \ell(\boldsymbol{U}, \boldsymbol{V}) - \beta \alpha \Big\langle \boldsymbol{Y} \circ \frac{f'(\tilde{\boldsymbol{M}})}{f(\boldsymbol{Y} \circ \tilde{\boldsymbol{M}})}, \Delta \boldsymbol{U}^+ \boldsymbol{V}^T + \boldsymbol{U}(\Delta \boldsymbol{V}^+)^T \Big\rangle_\Omega. \tag{4.46}$$

With the MM step in Section 4.2.1 and the GN step in Section 4.2.2, we summarize the MMGN method for solving the 1-bit matrix completion problem in Algorithm 5.

Algorithm 5 MMGN for 1-bit matrix completion

Require: $\boldsymbol{Y}$, Ω, target rank r, tolerance `tol`, index number $k = 0$.

1: Initialize $\boldsymbol{U}^{(0)}$, $\boldsymbol{V}^{(0)}$, relative error of ℓ rel.
2: **while** rel $>$ `tol` **do**
3: Construct $\boldsymbol{X}^{(k)} = \begin{cases} 4\boldsymbol{Y} \circ f(-\boldsymbol{Y} \circ (\boldsymbol{U}^{(k)} \boldsymbol{V}^{(k)^T})) & \text{under the logistic model,} \\ \boldsymbol{Y} \circ f'((\boldsymbol{U}^{(k)} \boldsymbol{V}^{(k)^T}))/(t(\sigma) f(\boldsymbol{Y} \circ (\boldsymbol{U}^{(k)} \boldsymbol{V}^{(k)^T}))) & \text{under the probit model.} \end{cases}$
4: Construct $\boldsymbol{x}^{(k)}$ from $\boldsymbol{X}^{(k)}$ over observed entries Ω.
5: Construct $\boldsymbol{J}^{(k)}$ in (4.33).
6: Solve the least square problem $\frac{1}{2}||\boldsymbol{x}^{(k)} - \boldsymbol{J}^{(k)}\boldsymbol{\theta}||^2$ and get the least norm solution $\boldsymbol{\theta}^+$.
7: Get $\Delta \boldsymbol{u}^+$ and $\Delta \tilde{\boldsymbol{v}}^+$ from $\boldsymbol{\theta}^+$.
8: Get $\Delta \boldsymbol{U}^+$ and $\Delta \boldsymbol{V}^+$ by reshaping $\Delta \boldsymbol{u}^+$ and $\Delta \tilde{\boldsymbol{v}}^+$.
9: Determine the step size α following the Armijo condition in (4.46).
10: Update $\boldsymbol{U}^{(k+1)} = \boldsymbol{U}^{(k)} + \alpha \Delta \boldsymbol{U}^+$ and $\boldsymbol{V}^{(k+1)} = \boldsymbol{V}^{(k)} + \alpha \Delta \boldsymbol{V}^+$.
11: Compute the relative decrease of the loss function: rel $= |\frac{\ell(\boldsymbol{U}^{(k+1)}, \boldsymbol{V}^{(k+1)})}{\ell(\boldsymbol{U}^{(k)}, \boldsymbol{V}^{(k)})} - 1|$.
12: $k = k + 1$.
13: **end while**
14: Return $\hat{\boldsymbol{M}} = \boldsymbol{U}^{(k)} \boldsymbol{V}^{(k)^T}$.

4.3 Numerical Experiments

We evaluate the performances of the MMGN under this section by doing several synthetic experiments. Based on our claims in Section 4.1, we design different scenarios where the MMGN method can beat existing methods. Two previous methods are included in comparison: the **1Bit** method [Dav14] and the **Maxnorm** method [Cai13].

We construct the underlying matrix $\boldsymbol{M} \in \mathbb{R}^{m\times n}$ with target low rank r by forming $\boldsymbol{M} = \boldsymbol{M}_1 \boldsymbol{M}_2^T$ where $\boldsymbol{M}_1 \in \mathbb{R}^{m\times r}$ and $\boldsymbol{M}_2 \in \mathbb{R}^{n\times r}$. The strategy of generating $\boldsymbol{M}_1$ and $\boldsymbol{M}_2$ will determine the "spiki-

ness" of $\boldsymbol{M}$. We include two approaches here:

- Uniform underlying: entries of $\boldsymbol{M}_1$ and $\boldsymbol{M}_2$ are i.i.d. from a uniform distribution on $[-\frac{1}{2}, \frac{1}{2}]$. After doing $\boldsymbol{M} = \boldsymbol{M}_1 \boldsymbol{M}_2^T$, it is scaled such that $\|\boldsymbol{M}\|_\infty = 1$.
- Normal underlying: entries of $\boldsymbol{M}_1$ and $\boldsymbol{M}_2$ are i.i.d. from a standard normal distribution, i.e., $N(0,1)$. We then do $\boldsymbol{M} = \boldsymbol{M}_1 \boldsymbol{M}_2^T$ without scaling.

With the uniform underlying, the absolute values of entries of $\boldsymbol{M}$ are bounded above by 1. So the matrix is not spiky. This is precisely the setting that Davenport et al. [Dav14] and Cai & Zhou [Cai13] used in their simulation studies. We will show the MMGN is better than the 1Bit and the Maxnorm even under the setting where they are designed. With the normal underlying, we do not scale $\boldsymbol{M}$. So the maximum entry does not have a theoretical bound. In this case, the matrix is kind of spiky. We will show the MMGN still recovers reasonable estimates. How spiky is spiky enough for the problem will not recover good results still lack academic discussions. So it is valuable to investigate the current methods' stability when the underlying matrix is a little spiky. Also, as far as we know, there is no standard way in published papers to design a spiky underlying matrix. We try a simple example here but claim that the study of this topic should be our future work.

After determining matrix size m and n we randomly select Ω from a uniform distribution with a user-defined fraction of observations p, i.e., $p = \frac{|\Omega|}{mn}$. To evaluate the performance, we use the relative error. For an estimated $\hat{\boldsymbol{M}}$, its relative error is:

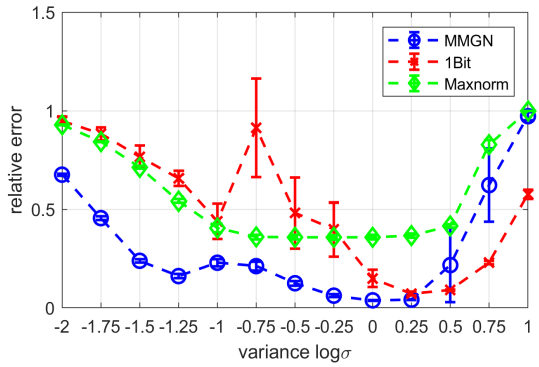
$$\frac{\|\hat{\boldsymbol{M}} - \boldsymbol{M}\|_F^2}{\|\boldsymbol{M}\|_F^2}.$$

We report the relative error and corresponding run time(sec) in logarithm scale of three methods. Each data point in the following figures is an average over 10 replicates.

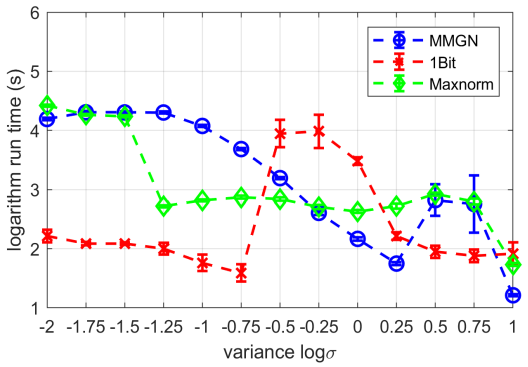
4.3.1 Spiky Case

Under the spiky underlying matrix, we first test the performances against different noise levels under the probit model. Set $m = n = 500$, rank $r = 3$, fraction of observations $p = 0.85$. Select $\log\sigma \in \{-2, -1.75, -1.5, -1.25, -1, -0.75, -0.5, -0.25, 0, 0.25, 0.5, 0.75, 1\}$. Figure 4.1a shows the relative error of three methods as a function of noise level $\log\sigma$. Note that the Maxnorm method, labeled in green, shows poor recovery accuracy with the relative error always near 0.5, which implies that the Maxnorm method is quite sensitive to the spikiness of the underlying matrix. It fails to recover a reasonable estimate whenever the true matrix is spiky. The 1Bit method in red achieves some good results when the noise level is in a specific range. The MMGN, labeled in blue, is the best one when $\log\sigma \leq 0.25$. It is quite stable remains the best one for a wide range of noise levels. For all methods, we see the relative error first has a decreasing trend up to $\log\sigma = 0.25$ and then increases. This shows that a specific range of noise levels is necessary to recover the true matrix in the 1-bit matrix completion problem. Figure 4.1b reports the logarithm of run time. We see the 1Bit is the fastest one at the beginning; and then the MMGN outperforms except for $\log\sigma = 0.5$. We claim that the

MMGN should still be our first choice since it gives almost twice better recovery accuracy among a wide range of noise levels.



(a) Relative error.

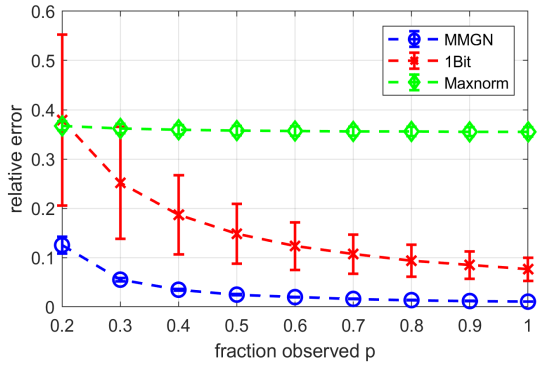


(b) Logarithm of run time (sec).

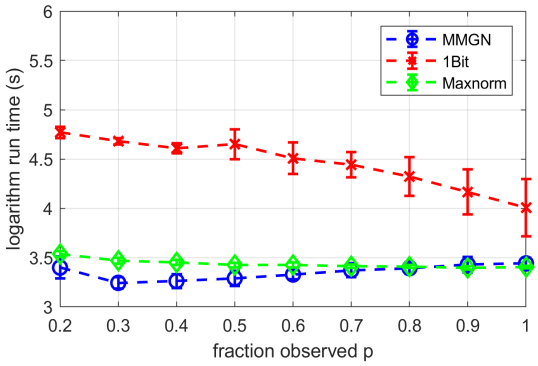
Figure 4.1 The relative error and corresponding logarithm of run time (sec) of three methods as a function of noise level $\log\sigma$ under *probit* model with $m = n = 500$, rank $r = 3$ and spiky underlying matrix. Panel (a): the MMGN gives the smallest relative error when $\log\sigma \leq 0.25$; the 1Bit method works better when $\log\sigma \geq 0.5$; the Maxnorm method is quite sensitive to the spikiness of the underlying matrix and gives poor recovery accuracy. The relative error first has a decreasing trend for all methods and then increases when the noise level is huge, which implies that a specific range of noise levels is essential for the 1-bit matrix completion. Panel (b): the 1Bit is the fastest one for smaller variances; then, the MMGN starts outperforming the other two methods. The Maxnorm method is always slow.

Figure 4.1a provides a good choice of noise level for the following experiments. We select $\log\sigma = 0$ to test the performances under the probit model with different fraction of observations $p \in \{0.2, 0.3, \ldots, 1\}$. Set $m = n = 1200$, rank $r = 3$. Figure 4.2a reports the relative errors. First, look at the Maxnorm method. It shows a slightly decreasing trend as the increasing of fractions p. However, it generally gives a very high relative error no matter what p is. The 1Bit and MMGN methods work much better. Both show a reasonable decreasing trend, implying that we can get a better estimate when we observe more entries. The MMGN method is consistently the best one: it gives the smallest error among all p values. Figure 4.2b reports the logarithm of run time. The MMGN is the most efficient one. The Maxnorm method is also fast, but it does not give a good enough estimate. The 1Bit is pretty slow, especially when the observed fraction is low.

We then test the same setting under the logistic model. Set $m = n = 1200$, rank $r = 3$. Plot the relative error as a function of different fraction of observations $p \in \{0.2, 0.3, \ldots, 1\}$. Note that none of the previous studies does the simulation testing under the logistic model. Figure 4.3a reports the relative errors. The MMGN consistently gives the smallest relative error. The 1Bit method also works well. Both MMGN and 1Bit give a decreasing trend as increasing of the observed fraction p, which is not obvious in the Maxnorm curve. In terms of the run time in Figure 4.3b, the MMGN is still the fastest one. We see a huge gap between the MMGN and the other two methods.

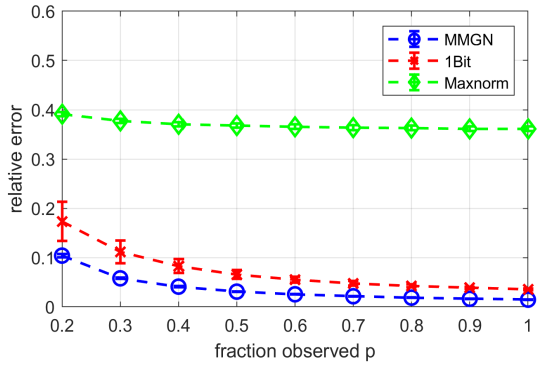


(a) Relative error.

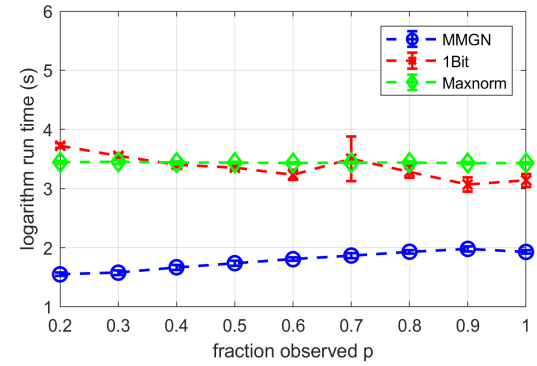


(b) Logarithm of run time (sec).

Figure 4.2 The relative error and corresponding logarithm of run time (sec) of three methods as a function of observed fraction p under *probit* model with $m = n = 1200$, rank $r = 3$ and spiky underlying matrix. Panel (a): the MMGN consistently gives the smallest relative error; the 1Bit method also gives good results; the Maxnorm method works poorly, the curve does not have an obvious decreasing as increasing of the observed fraction p. Panel (b): the MMGN is the most efficient one; the Maxnorm method shows competitive results; the 1Bit method is slow, especially when p is small.



(a) Relative error.



(b) Logarithm run time (sec).

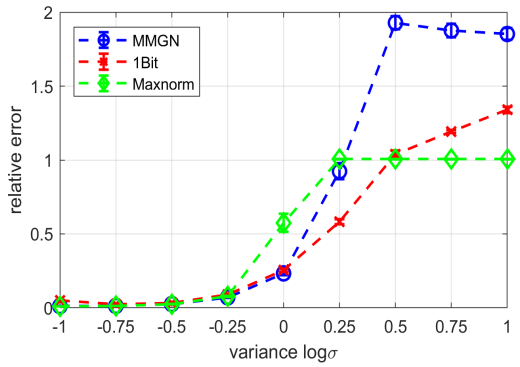
Figure 4.3 The relative error and corresponding logarithm of run time (sec) of three methods as a function of observed fraction p under *logistic* model with $m = n = 1200$, rank $r = 3$ and spiky underlying matrix. Panel (a): the MMGN gives the smallest relative error among all p; the 1Bit method also works well; the Maxnorm shows poor performance. Panel (b): the MMGN is fastest; the other two methods are much slower than the MMGN.

Based on these three experiments, we see the MMGN outperforms the other two methods under the spiky underlying matrix. Both 1Bit and Maxnorm are designed for non-spiky underlying matrices, but we see their stability against spikiness are quite different. The 1Bit method can still produces some reasonable estimates under spiky underlying, but the Maxnorm method fails. Our MMGN method achieves the best recovery accuracy among a wide range of noise levels. Moreover, under a specific choice of σ, the MMGN is also the best one among different choices of the observed fraction.

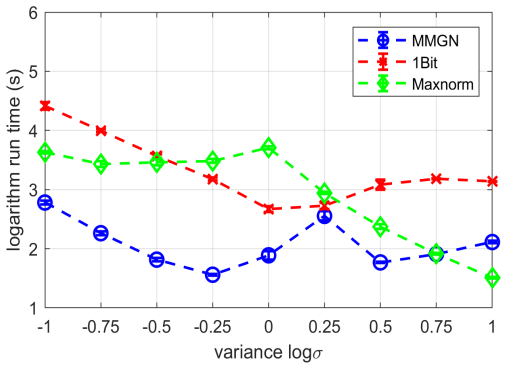
4.3.2 Non-Spiky Case

We repeat the same procedure under the non-spiky case where we generate the synthetic underlying matrices from a uniform distribution. This exactly matches the setting used in [Cai13; Dav14]. We will show that the MMGN still outperforms the other two methods even under this specific setting that the previous studies used. Again, we first test the performances under different noise level $\log\sigma \in \{-1, -0.75, -0.5, -0.25, 0, 0.25, 0.5, 0.75, 1\}$. Set $m = n = 1200$, rank $r = 3$, fraction of observations $p = 0.85$. Figure 4.4a reports the relative errors. None of the methods works well when $\log\sigma \geq 0.25$: all of them return a huge error. The non-spiky underlying matrix is more sensitive against significant variances than the spiky underlying matrix, where we can still achieve minor errors under $\log\sigma = 0.25$ as shown in Figure 4.1a. When $\log\sigma = 0$ or -0.25, the MMGN method gives the smallest error. When $\log\sigma \leq -0.5$, all three methods give similar results. Even though it is hard to detect from the plot, the Maxnorm method achieves the smallest error. We see the improvement of the Maxnorm method from the spiky underlying matrix to the non-spiky underlying matrix, which also implies its sensitivity to spikiness. Consider logarithm of run time shown in Figure 4.4b, the MMGN is the fastest one except for $\log\sigma = 1$. We claim that the MMGN should be our first choice since it takes the shortest run time to achieve similar or even better results than the 1Bit or Maxnorm when $\log\sigma \leq 0$. With a non-spiky underlying matrix, we do not try the case where $\log\sigma < 1$. There are some computing issues such that the MMGN gives a `Nan` value during the iterations. We can see the advantages of using the MMGN from current results. However, it could be better to compare the performances with smaller noise levels. We leave this as future work.

We also select $\log\sigma = 0$ under the probit model to test the performances with different fraction of observations $p \in \{0.2, 0.3, \ldots, 1\}$. Set $m = n = 3000$, and rank $r = 3$. Figure 4.5a shows the relative errors. We plot another horizontal line represents the error equals to 0.5. We treat the estimate whose relative error is near or bigger 0.5 as a poor estimate since it indicates that near or more than 50% of the actual underlying is not recovered correctly. The MMGN method consistently beats the other two methods except for $p = 0.2$, where even the best estimate does not return a good enough recovery accuracy. Also, consider the logarithm of run time shown in Figure 4.5b, we claim that the MMGN should be a better choice. The Maxnorm method outperforms the 1Bit when $p \geq 0.4$, but neither is better than the MMGN. Even under the non-spiky underlying matrix where Davenport et al. [Dav14] and Cai & Zhou [Cai13] developed the 1Bit and the Maxnorm method, the MMGN can still achieve better results in terms of both relative errors and run time.

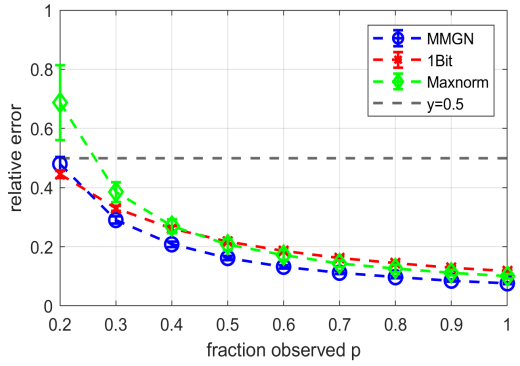


(a) Relative error.

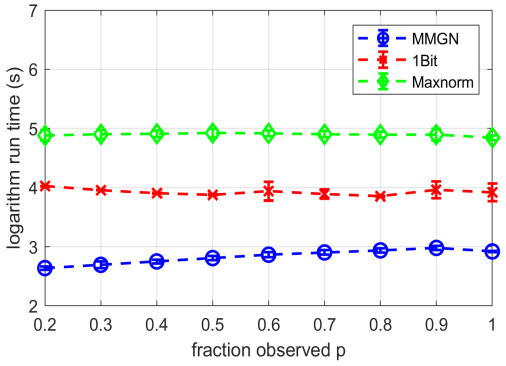


(b) Logarithm of run time (sec).

Figure 4.4 The relative error and corresponding logarithm of run time (sec) of three methods as a function of noise level $\log\sigma$ under *probit* model with $m = n = 1200$, rank $r = 3$ and non-spiky underlying matrix. Panel (a): none of three methods work well when $\log\sigma \geq 0.25$; the MMGN method gives best result when $\log\sigma = 0, -0.25$; the Maxnorm method gives the best result when $\log\sigma \leq -0.25$, but the other two methods also give similar errors. Panel (b): the MMGN is the most efficient one except for $\log\sigma = 1$; both 1Bit and Maxnorm are slow when $\log\sigma$ is small. The MMGN takes the shortest run time to achieve similar or even better results than the 1Bit or Maxnorm with $\log\sigma \leq 0$.



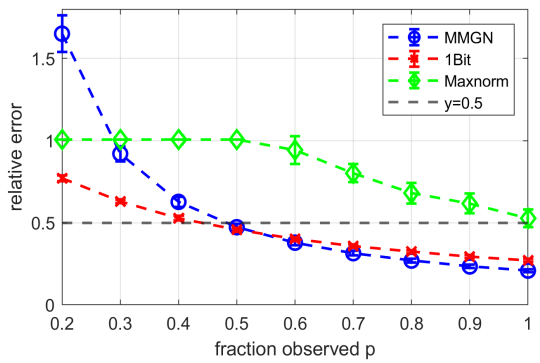
(a) Relative error.



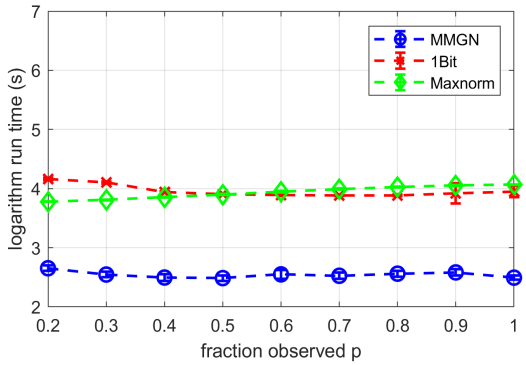
(b) Logarithm of run time (sec).

Figure 4.5 The relative error and corresponding logarithm of run time (sec) of three methods as a function of observed fraction p under *probit* model with $m = n = 3000$, rank $r = 3$ and non-spiky underlying matrix. Panel (a): the MMGN gives the smallest relative error except for $p = 0.2$. When $p = 0.2$, none of the three methods shows good enough recovery accuracy: all of them give an error near 0.5. Panel (b): the MMGN is the most efficient one; the 1Bit method is slightly slower; the Maxnorm method takes a much longer time to recover competitive results.

Next, we test the same setting under the logistic model. Set $m = n = 3000$, rank $r = 3$. Figure 4.6a reports the relative errors. None of the methods performs as well as that with spiky underlying matrices. We also plot a horizontal line representing the bad error 0.5. For $p \leq 0.4$, all methods return errors bigger than this threshold line. Even though the 1Bit method performs good, we claim that the results under this part do not make too much sense since even the best estimate still produces a huge error. For $p \geq 0.6$, we see the MMGN becomes the best one. The 1Bit method beats the MMGN when $p = 0.5$. However, consider the run time reported in Figure 4.6b, the MMGN is most desirable.



(a) Relative error.



(b) Logarithm of run time (sec).

Figure 4.6 The relative error and corresponding logarithm of run time (sec) of three methods as a function of observed fraction p under *logistic* model with $m = n = 3000$, rank $r = 3$ and non-spiky underlying matrix. Panel (a): the 1Bit performs best with $p \leq 0.5$ where all methods return an error larger than 0.5. We claim all methods are hard to recover reasonable enough estimate when the observed fraction is too small. The MMGN returns the smallest error when $p \geq 0.6$. Panel (b): the MMGN improves the efficiency dramatically.

4.4 Discussion and Future Work

This chapter proposed a new iterative strategy method, MMGN, for doing the 1-bit matrix completion problem. Compared to the existing methods, the MMGN shows better recovery accuracy, faster computing efficiency, and more stability against the spikiness of the underlying matrix.

However, several directions require future work. First, the thoroughly proof of the monotonicity in Proposition 7 still needs to be implemented. We believe this is true from the empirical studies but cannot give complete proof. Second, the discussion of the smaller noise level under the probit model with the non-spiky underlying matrix. Currently, we do not get the output of the MMGN because of the numerical errors. Another minor issue is the investigation of a standard way to generate a spiky underlying matrix.

CHAPTER

5

DISCUSSION

This dissertation introduces the Majorization-Minimization (MM) algorithm and illustrates how it improves the current optimization methods by showing three examples. We propose ARFMM, sharp-majorized ADMM, and MMGN for solving histogram valued data clustering, constrained binomial regression, and 1-bit matrix completion, respectively. Simulation studies and real data applications show that the newly proposed methods outperform the existing methods. Also, the ARFMM and MMGN are pretty stable against the perturbations and spikiness, which is not achieved by the previous methods.

However, we still have some directions that require future work. The essential one should be the proof of convergence of the three methods. Especially in the sharp-majorized ADMM, we derive the algorithm based on an existing method whose convergence is well-established. However, we still see some in-convergent during the simulation studies. Also, Propositions 4 and 7 also need further discussions. We are confident in the correctness of these two propositions from empirical testings, but complete proofs are still necessary. Throughout the dissertation, we also see the limitations of using the MM algorithm. It produces an approximate version of the true objective function in the ARFMM, or it requires strict conditions in the sharp-majorized ADMM and MMGN. It is valuable to investigate the generalization under other types of problems.

BIBLIOGRAPHY

[Arm66] Armijo, L. "Minimization of functions having Lipschitz continuous first partial derivatives". *Pacific Journal of mathematics* **16**.1 (1966), pp. 1–3.

[Bau21] Bauch, J. et al. "Rank 2r iterative least squares: efficient recovery of ill-conditioned low rank matrices from few entries". *SIAM Journal on Mathematics of Data Science* **3**.1 (2021), pp. 439–465.

[Bau11] Bauschke, H. H., Combettes, P. L., et al. *Convex analysis and monotone operator theory in Hilbert spaces*. Vol. 408. Springer, 2011.

[Ber14] Bertsekas, D. P. *Constrained optimization and Lagrange multiplier methods*. Academic press, 2014.

[Bha15] Bhaskar, S. A. & Javanmard, A. "1-bit matrix completion under exact low-rank constraint". *2015 49th Annual Conference on Information Sciences and Systems (CISS)*. IEEE. 2015, pp. 1–6.

[Bil17] Billard, L. & Kim, J. "Hierarchical clustering for histogram data". *Wiley Interdisciplinary Reviews: Computational Statistics* **9**.5 (2017), e1405.

[Bir00] Birgin, E. G. et al. "Nonmonotone spectral projected gradient methods on convex sets". *SIAM Journal on Optimization* **10**.4 (2000), pp. 1196–1211.

[Bon06] Bonnans, J.-F. et al. *Numerical optimization: theoretical and practical aspects*. Springer Science & Business Media, 2006.

[Bos92] Boser, B. E. et al. "A training algorithm for optimal margin classifiers". *Proceedings of the fifth annual workshop on Computational learning theory*. 1992, pp. 144–152.

[Boy11] Boyd, S. et al. "Distributed optimization and statistical learning via the alternating direction method of multipliers". *Foundations and Trends® in Machine learning* **3**.1 (2011), pp. 1–122.

[Cai13] Cai, T. & Zhou, W.-X. "A max-norm constrained minimization approach to 1-bit matrix completion". *The Journal of Machine Learning Research* **14**.1 (2013), pp. 3619–3647.

[Can10] Candes, E. J. & Plan, Y. "Matrix completion with noise". *Proceedings of the IEEE* **98**.6 (2010), pp. 925–936.

[Can09] Candès, E. J. & Recht, B. "Exact matrix completion via convex optimization". *Foundations of Computational mathematics* **9**.6 (2009), pp. 717–772.

[Chi15] Chi, E. C. & Lange, K. "Splitting methods for convex clustering". *Journal of Computational and Graphical Statistics* **24**.4 (2015), pp. 994–1013.

[Chi19] Chi, E. C. & Steinerberger, S. "Recovering Trees with Convex Clustering". *SIAM Journal on Mathematics of Data Science* **1**.3 (2019), pp. 383–407.

[Coi13] Coifman, R. R. & Leeb, W. “Earth mover’s distance and equivalent metrics for spaces with hierarchical partition trees”. *Yale CS Techinical Report* (2013).

[Com11] Combettes, P. L. & Pesquet, J.-C. “Proximal splitting methods in signal processing”. *Fixed-point algorithms for inverse problems in science and engineering*. Springer, 2011, pp. 185–212.

[Com05] Combettes, P. L. & Wajs, V. R. “Signal recovery by proximal forward-backward splitting”. *Multiscale Modeling & Simulation* **4**.4 (2005), pp. 1168–1200.

[Cot18] Cottet, V. & Alquier, P. “1-Bit matrix completion: PAC-Bayesian analysis of a variational approximation”. *Machine Learning* **107**.3 (2018), pp. 579–603.

[Cou94] Courant, R. et al. “Variational methods for the solution of problems of equilibrium and vibrations”. *Lecture notes in pure and applied mathematics* (1994), pp. 1–1.

[Cui15] Cui, Y. et al. “On the convergence properties of a majorized ADMM for linearly constrained convex optimization problems with coupled objective functions”. *arXiv preprint arXiv:1502.00098* (2015).

[Cur44] Curry, H. B. “The method of steepest descent for non-linear minimization problems”. *Quarterly of Applied Mathematics* **2**.3 (1944), pp. 258–261.

[Dav14] Davenport, M. A. et al. “1-bit matrix completion”. *Information and Inference: A Journal of the IMA* **3**.3 (2014), pp. 189–223.

[Dru97] Drucker, H. et al. “Support vector regression machines”. *Advances in neural information processing systems* **9** (1997), pp. 155–161.

[Dun78] Dunn, J. C. & Harshbarger, S. “Conditional gradient algorithms with open loop step size rules”. *Journal of Mathematical Analysis and Applications* **62**.2 (1978), pp. 432–444.

[Eck94] Eckstein, J. “Some saddle-function splitting methods for convex programming”. *Optimization Methods and Software* **4**.1 (1994), pp. 75–83.

[Faz13] Fazel, M. et al. “Hankel matrix rank minimization with applications to system identification and realization”. *SIAM Journal on Matrix Analysis and Applications* **34**.3 (2013), pp. 946–977.

[Fis36] Fisher, R. A. “The use of multiple measurements in taxonomic problems”. *Annals of eugenics* **7**.2 (1936), pp. 179–188.

[Fra56] Frank, M., Wolfe, P., et al. “An algorithm for quadratic programming”. *Naval research logistics quarterly* **3**.1-2 (1956), pp. 95–110.

[Gab76] Gabay, D. & Mercier, B. “A dual algorithm for the solution of nonlinear variational problems via finite element approximation”. *Computers & mathematics with applications* **2**.1 (1976), pp. 17–40.

[Gey91] Geyer, C. J. “Constrained maximum likelihood exemplified by isotonic convex logistic regression”. *Journal of the American Statistical Association* **86**.415 (1991), pp. 717–724.

[Glo75] Glowinski, R. & Marroco, A. “Sur l’approximation, par éléments finis d’ordre un, et la résolution, par pénalisation-dualité d’une classe de problèmes de Dirichlet non linéaires”. *ESAIM: Mathematical Modelling and Numerical Analysis-Modélisation Mathématique et Analyse Numérique* **9**.R2 (1975), pp. 41–76.

[Gol92] Goldberg, D. et al. “Using collaborative filtering to weave an information tapestry”. *Communications of the ACM* **35**.12 (1992), pp. 61–70.

[Gro03] Groenen, P. et al. *Weighted majorization algorithms for weighted least squares decomposition models*. Tech. rep. 2003.

[Gro11] Gross, D. “Recovering low-rank matrices from few coefficients in any basis”. *IEEE Transactions on Information Theory* **57**.3 (2011), pp. 1548–1566.

[Guy93] Guyon, I. et al. “Automatic capacity tuning of very large VC-dimension classifiers”. *Advances in neural information processing systems*. 1993, pp. 147–155.

[Had08] Hadamard, J. *Mémoire sur le problème d’analyse relatif à l’équilibre des plaques élastiques encastrées*. Vol. 33. Imprimerie nationale, 1908.

[Hag19] Hager, W. W. & Zhang, H. “Inexact alternating direction methods of multipliers for separable convex optimization”. *Computational Optimization and Applications* **73**.1 (2019), pp. 201–235.

[Hay18] Hayashi, M. et al. “Binary matrix completion using unobserved entries”. *arXiv preprint arXiv:1803.04663* (2018).

[Hoc11] Hocking, T. D. et al. “Clusterpath: an algorithm for clustering using convex fusion penalties”. *28th International Conference on Machine Learning*. Ed. by Getoor, L. & Scheffer, T. ACM, 2011, p. 1.

[Hoe70] Hoerl, A. E. & Kennard, R. W. “Ridge regression: Biased estimation for nonorthogonal problems”. *Technometrics* **12**.1 (1970), pp. 55–67.

[Hoo77] Hook, E. B. & Chambers, G. M. “Estimated rates of Down syndrome in live births by one year maternal age intervals for mothers aged 20-49 in a New York State study-implications of the risk figures for genetic counseling and cost-benefit analysis of prenatal diagnosis programs”. *Birth defects original article series* **13**.3A (1977), pp. 123–141.

[Hoo78a] Hook, E. B. & Fabia, J. J. “Frequency of Down syndrome in livebirths by single-year maternal age interval: results of a Massachusetts study”. *Teratology* **17**.3 (1978), pp. 223–228.

[Hoo78b] Hook, E. B. & Lindsjö, A. “Down syndrome in live births by single year maternal age interval in a Swedish study: comparison with results from a New York State study.” *American journal of human genetics* **30**.1 (1978), p. 19.

[Hsi15] Hsieh, C.-J. et al. “PU learning for matrix completion”. *International Conference on Machine Learning*. PMLR. 2015, pp. 2445–2453.

[Hu16] Hu, Y. et al. “ADMM algorithmic regularization paths for sparse statistical machine learning”. *Splitting Methods in Communication, Imaging, Science, and Engineering*. Springer, 2016, pp. 433–459.

[Hun04] Hunter, D. R. & Lange, K. “A tutorial on MM algorithms”. *The American Statistician* **58**.1 (2004), pp. 30–37.

[Inc] Inc., W. R. “Mathematica, Version 12.3.1” (). Champaign, IL, 2021.

[Irp06] Irpino, A. & Verde, R. “A new Wasserstein based distance for the hierarchical clustering of histogram symbolic data”. *Data Science and Classification*. Springer, 2006, pp. 185–192.

[Ius99] Iusem, A. N. “Augmented Lagrangian methods and proximal point methods for convex optimization”. *Investigación Operativa* **8**.11-49 (1999), p. 7.

[Kar39] Karush, W. “Minima of functions of several variables with inequalities as side constraints”. *M. Sc. Dissertation. Dept. of Mathematics, Univ. of Chicago* (1939).

[Kim13] Kim, J. & Billard, L. “Dissimilarity measures for histogram-valued observations”. *Communications in Statistics-Theory and Methods* **42**.2 (2013), pp. 283–303.

[Kje00] Kjeldsen, T. H. “A contextualized historical analysis of the Kuhn–Tucker theorem in nonlinear programming: the impact of World War II”. *Historia mathematica* **27**.4 (2000), pp. 331–361.

[Kuh51] Kuhn, H & Tucker, A. “Nonlinear programming In Proceedings of 2nd Berkeley symposium (pp. 481–492)”. *Berkeley: University of California Press.[Google Scholar]* (1951).

[Lan67] Lance, G. N. & Williams, W. T. “A general theory of classificatory sorting strategies: 1. Hierarchical systems”. *The Computer Journal* **9**.4 (1967), pp. 373–380.

[Lar06] Larsen, F. H. et al. “An exploratory chemometric study of 1H NMR spectra of table wines”. *Journal of Chemometrics: A Journal of the Chemometrics Society* **20**.5 (2006), pp. 198–208.

[Lee10] Lee, J. et al. “Practical large-scale optimization for max-norm regularization”. Neural Information Processing Systems. 2010.

[Lee09] Leeuw, J. de & Lange, K. “Sharp quadratic majorization in one dimension”. *Computational statistics & data analysis* **53**.7 (2009), pp. 2471–2484.

[Lem89] Lemaire, B. “The proximal algorithm”. *International series of numerical mathematics* **87** (1989), pp. 73–87.

[Lem12] Lemaréchal, C. “Cauchy and the gradient method”. *Doc Math Extra* **251**.254 (2012), p. 10.

[Lev66] Levitin, E. S. & Polyak, B. T. “Constrained minimization methods”. *USSR Computational mathematics and mathematical physics* **6**.5 (1966), pp. 1–50.

[Li16] Li, M. et al. “A majorized ADMM with indefinite proximal terms for linearly constrained convex composite optimization”. *SIAM Journal on Optimization* **26**.2 (2016), pp. 922–950.

[Lin11] Lindsten, F. et al. *Just Relax and Come Clustering! A Convexification of k-Means Clustering*. Tech. rep. Linköpings Universitet, 2011.

[Lin07] Linial, N. et al. “Complexity measures of sign matrices”. *Combinatorica* **27**.4 (2007), pp. 439–463.

[Mac67] MacQueen, J. et al. “Some methods for classification and analysis of multivariate observations”. *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*. Vol. 1. 14. Oakland, CA, USA. 1967, pp. 281–297.

[McL04] McLachlan, G. J. *Discriminant analysis and statistical pattern recognition*. Vol. 544. John Wiley & Sons, 2004.

[Mor82] Moré, J. J. & Sorensen, D. C. *Newton's method*. Tech. rep. Argonne National Lab., IL (USA), 1982.

[Mos10] Mosci, S. et al. “Solving structured sparsity regularization with proximal methods”. *Joint European conference on machine learning and knowledge discovery in databases*. Springer. 2010, pp. 418–433.

[Nat15] Natarajan, N. et al. “PU matrix completion with graph information”. *2015 IEEE 6th international workshop on computational advances in multi-sensor adaptive processing (CAMSAP)*. IEEE. 2015, pp. 37–40.

[Ni16] Ni, R. & Gu, Q. “Optimal statistical and computational rates for one bit matrix completion”. *Artificial Intelligence and Statistics*. PMLR. 2016, pp. 426–434.

[Par14] Parikh, N. & Boyd, S. “Proximal algorithms”. *Foundations and Trends in optimization* **1**.3 (2014), pp. 127–239.

[Par19] Park, C. et al. “Convex clustering analysis for histogram-valued data”. *Biometrics* **75**.2 (2019), pp. 603–612.

[Pel05] Pelckmans, K. et al. “Convex clustering shrinkage”. *PASCAL Workshop on Statistics and Optimization of Clustering Workshop*. 2005.

[Qui86] Quinlan, J. R. “Induction of decision trees”. *Machine learning* **1**.1 (1986), pp. 81–106.

[Qui14] Quinlan, J. R. *C4. 5: programs for machine learning*. Elsevier, 2014.

[Rav16] Ravanbakhsh, S. et al. “Boolean matrix factorization and noisy completion via message passing”. *International Conference on Machine Learning*. PMLR. 2016, pp. 945–954.

[Roc76] Rockafellar, R. T. “Monotone operators and the proximal point algorithm”. *SIAM journal on control and optimization* **14**.5 (1976), pp. 877–898.

[Roc78] Rockafellar, R. “Monotone operators and augmented Lagrangian methods in nonlinear programming”. *Nonlinear Programming 3*. Elsevier, 1978, pp. 1–25.

[Rub00] Rubner, Y. et al. “The Earth Mover’s Distance as a metric for image retrieval”. *International Journal of Computer Vision* **40**.2 (2000), pp. 99–121.

[Saa98] Saad, D. “Online algorithms and stochastic approximations”. *Online Learning* **5** (1998), pp. 6–3.

[Scu18] Scutari, G. & Sun, Y. “Parallel and distributed successive convex approximation methods for big-data optimization”. *Multi-agent Optimization*. Springer, 2018, pp. 141–308.

[Sin10] Sindhwani, V. et al. “One-class matrix completion with low-density factorizations”. *2010 IEEE International Conference on Data Mining*. IEEE. 2010, pp. 1055–1060.

[Tib96] Tibshirani, R. “Regression shrinkage and selection via the lasso”. *Journal of the Royal Statistical Society: Series B (Methodological)* **58**.1 (1996), pp. 267–288.

[Tri78] Trimble, B. K. et al. “Maternal age and Down syndrome: age-specific incidence rates by single-year intervals”. *American journal of medical genetics* **2**.1 (1978), pp. 1–5.

[VR05] Van Ruitenburg, J. “Algorithms for parameter estimation in the Rasch model”. *Measurement and Research Department Reports* **4** (2005), p. 116.

[Wey19] Weylandt, M. et al. “Dynamic Visualization and Fast Computation for Convex Clustering via Algorithmic Regularization”. *Journal of Computational and Graphical Statistics* (2019).

[Wri99] Wright, S., Nocedal, J., et al. “Numerical optimization”. *Springer Science* **35**.67-68 (1999), p. 7.

[Zha20] Zhang, N. et al. “A linearly convergent majorized ADMM with indefinite proximal terms for convex composite programming and its applications”. *Mathematics of Computation* **89**.324 (2020), pp. 1867–1894.

[Zhu17] Zhu, Y. “An augmented ADMM algorithm with application to the generalized lasso problem”. *Journal of Computational and Graphical Statistics* **26**.1 (2017), pp. 195–204.

APPENDICES

APPENDIX

A

SUPPLEMENTARY MATERIALS FOR "ALGORITHMIC REGULARIZATION FUSION MAJORIZATION-MINIMIZATION METHOD FOR CLUSTERING HISTOGRAM DATA"

A.1 QP Formulation of the Clustering Problem with EMD

The Earth Mover's Distance (EMD) between two histograms is defined as the minimum amount of "work" required to reshape one histogram into the other [Rub00]. It can be computed via a Linear Program (LP). For simplicity, suppose two histograms $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^p$ are normalized, i.e. $\sum_{r=1}^{p} a_r = \sum_{s=1}^{p} b_s = 1$, and $a_r \geq 0$, $b_s \geq 0$ for all r, s. We define a ground distance d_{rs} between the rth bin and the sth bin of a histogram, a common example is $d_{rs} = |r-s|$ for $r, s = 1, \ldots, p$. We want to find a flow $\boldsymbol{F} = [\mathscr{F}_{rs}]$, with $\mathscr{F}_{rs}$ the flow from a_r to b_s, that minimizes the overall cost:

$$\text{WORK}(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{F}) \quad = \quad \sum_{r=1}^{p} \sum_{s=1}^{p} d_{rs} \mathscr{F}_{rs},$$

subject to the following constraints:

$$\mathscr{F}_{rs} \geq 0, \quad 1 \leq r \leq p,\, 1 \leq s \leq p, \tag{A.1}$$

$$\sum_{s=1}^{p} \mathscr{F}_{rs} \leq a_r, \quad 1 \leq r \leq p, \tag{A.2}$$

$$\sum_{r=1}^{p} \mathscr{F}_{rs} \leq b_s, \quad 1 \leq s \leq p, \tag{A.3}$$

$$\sum_{r=1}^{p}\sum_{s=1}^{p} \mathscr{F}_{rs} = 1. \tag{A.4}$$

Constraint (A.1) allows moving “earth" only from $\boldsymbol{a}$ to $\boldsymbol{b}$ but not vice versa. Constraint (A.2) limits the flows that can be sent by $\boldsymbol{a}$ to its mass a_r. Constraint (A.3) ensures that the histogram $\boldsymbol{b}$ receives no more “earth" than its mass b_s. Constraint (A.4) caps the maximum amount of “earth" that is possible to be moved. Then the EMD $d_{\text{EMD}}(\boldsymbol{a},\boldsymbol{b})$ is the optimal cost $\sum_{r=1}^{p}\sum_{s=1}^{p} d_{rs}\hat{\mathscr{F}}_{rs}$ attained under the optimal flow $\hat{\mathscr{F}}_{rs}$. Note that two histograms do not have to sum to 1; we refer to Rubner et al. [Rub00] for a general definition.

Recall that the clustering problem with the true EMD in the penalty is:

$$\hat{\boldsymbol{V}}(\lambda) \;=\; \underset{\boldsymbol{V}\in\mathbb{R}^{n\times p}}{\arg\min}\, \frac{1}{2}\|\boldsymbol{X}-\boldsymbol{V}\|_{\text{F}}^2 + \lambda\left(\sum_{i<j} w_{ij}\, \text{d}_{\text{EMD}}\left(\boldsymbol{V}_{i.}, \boldsymbol{V}_{j.}\right)\right). \tag{A.5}$$

In light of the LP formulation of the EMD, Problem (A.5) can be written as a standard QP:

$$(\hat{\boldsymbol{V}}, \hat{\mathscr{F}}_{rs}^{(i,j)}) \;=\; \underset{\boldsymbol{V},\mathscr{F}_{rs}^{(i,j)}}{\arg\min}\, \frac{1}{2}\|\boldsymbol{X}-\boldsymbol{V}\|_{F}^2 + \lambda\left(\sum_{i<j} w_{ij} \sum_{r=1}^{p}\sum_{s=1}^{p} d_{rs}\mathscr{F}_{rs}^{(i.j)}\right), \tag{A.6}$$

subject to:

$$\begin{aligned}
\mathscr{F}_{rs}^{(i,j)} &\geq 0, \quad 1 \leq r,s \leq p, 1 \leq i < j \leq n,\\
\sum_{s=1}^{p} \mathscr{F}_{rs}^{(i,j)} &\leq (\boldsymbol{V}_{i.})_r, \quad 1 \leq r \leq p, 1 \leq i < j \leq n,\\
\sum_{r=1}^{p} \mathscr{F}_{rs}^{(i,j)} &\leq (\boldsymbol{V}_{j.})_s, \quad 1 \leq s \leq p, 1 \leq i < j \leq n,\\
\sum_{r=1}^{p}\sum_{j=1}^{p} \mathscr{F}_{rs}^{(i,j)} &= 1, \quad 1 \leq i < j \leq p,\\
\sum_{r=1}^{p} (\boldsymbol{V}_{i.})_r &= 1, \quad 1 \leq r \leq p,
\end{aligned}$$

where $\mathscr{F}_{rs}^{(i,j)}$ is the flow from the rth bin of $\boldsymbol{V}_{i.}$ to the sth bin of $\boldsymbol{V}_{j.}$ and d_{rs} is the known ground

distance between rth bin and sth bin. Here we specify a set of flows $\{\mathscr{F}_{rs}^{(i,j)}\}$ for the pair of observations (i, j), so the pairwise distances drastically increase the size of the problem. Problem (A.6) is a standard QP with np variables from $\boldsymbol{V}$ and $n(n-1)p^2/2$ variables from $\mathscr{F}_{rs}^{(i,j)}$, so the total size of the variables in the QP is $np + n(n-1)p^2/2$, and the number of linear constraints is even larger. The size of the QP grows rapidly in n and p making it hard to solve in a reasonable amount of time and memory with an off-the-shelf QP solver and motivating the use of the approximate EMD.

A.2 Visualization from `fusion.information`

ARFMM outputs `fusion.information`, which we use to recover the hierarchical clustering tree. Assume that there are n observations $(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n)$ with M tuning parameters $(\lambda_1, \ldots, \lambda_M)$ used for the algorithm. Assume that λ_1 gives no fusion and λ_M makes the final fusion. With R's grammars, we then define `fusion.information` as a list with length M and with each element `fusion.information`[[k]] is another level of list recording the clustering assignment after iterating λ_k. Take the first entry as an example, `fusion.information`[[1]] is a list with length equal to the sample size n, representing no fusions have occurred and each observation represents a singleton cluster. In another word, there no fusions have occurred after the first iteration. So we have n clusters and each cluster just contains the corresponding observation:

- `fusion.information`[[1]]:
  - [[1]]: 1
  - [[2]]: 2
  - ...
  - [[n]]: n.

Suppose that λ_2 only makes observation $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ fuse together, then `fusion.information`[[2]] has $(n-1)$ clusters:

- `fusion.information`[[2]]:
  - [[1]]: 1,2
  - [[2]]: 3
  - ...
  - [[$n-1$]]: n.

Suppose that we have c_k clusters after iterating λ_k. Then `fusion.information`[[k]] is a list with length c_k, each `fusion.information`[[k]][[j]] contains the observations that belong to the jth cluster for $j = 1, \ldots, c_k$. Final λ_M fuses all observations together, so we only have 1 cluster that contains $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$:

- `fusion.information`[[M]]:

– [[1]]: 1,2,...,n.

For λ_k, we will know whether there is a fusion by checking if the length of fusion.information[[k]] shrinks compared to fusion.information[[k−1]]. If there is indeed some fusions, we know which observations have been fused together by checking the elements in each entry of fusion.information[[k]].

We determine the heights of branches in clustering tree as follow. We fix a total height d as the top level where final fusion occurs. Suppose that some fusion occurs under the kth iteration with λ_k, we then proportionally assign height $(k/M)d$ for it. In this way, different heights are proportional to the iteration out of M in which the corresponding fusion occurs. In practice, many of the fusions are at very beginning, leading to a very compressed tree. For visualization purposes, we magnify the heights (except the final one) if necessary but ensure relative proportions among them are left unchanged. This finally gives a tree that preserves the structure of the clustering pattern.

We apply the same strategy for creating trees with cvxclustrWK. The method returns a clustering assignment under a specific λ. We construct the tree from the collection of clustering assignments from a sequence of λs.

A.3 Drawback of cvxclustrWK

We next discuss a potential drawback of cvxclustrWK to argue why it may not sufficiently account for location information. Park et al. [Par19] proposed this method based on the convex clustering problem and Wasserstein-Kantorovich (WK) metric. For a histogram-valued observation $\boldsymbol{x}_i = (\boldsymbol{x}_{i1}, \ldots, \boldsymbol{x}_{ip})$, $i = 1, \ldots, n$, suppose that each variable $\boldsymbol{x}_{ij}$ consists of K_j non-overlapped intervals I_{jk} with corresponding relative frequencies π_{ijk} for $k = 1, \ldots, K_j$:

$$\boldsymbol{x}_{ij} \quad = \quad \left\{(I_{j1}, \pi_{ij1}), \ldots, (I_{jK_j}, \pi_{ijK_j})\right\}, \tag{A.7}$$

with $\pi_{ijk} \geq 0$ and $\sum_k \pi_{ijk} = 1$.

This is a common type of histogram-valued data from a survey. For example, suppose that we aim to clustering regions based on the exercises habits of the local people. We take a survey and get feedback from participants. Then n is the number of regions; p is the number of variables we investigate, such as age, height, weight, etc. So, a specific $\boldsymbol{x}_{ij}$ may represent the statistics of variable j from people in region i, which is usually summarized as another histogram with non-overlapped intervals with associated frequencies. More general, consider the time-series of daily measurements of average humidity in a year as an example. We treats 12 months as different variables and generate a histogram inside each month. Then $p = 12$ is the number of months in year; we have $\boldsymbol{x}_{ij}$ is the histogram of humidity in month j of observation i.

Park et al. [Par19] proposed two versions of cvxclustrWK. They start from a same structure but use different dissimilarity to measure the discrepancy of two histogram observations.

A.3.1 Standard cvxclustrWK

Usually, the histograms $\boldsymbol{x}_{ij}$ do not have unified format. For example, the value of K_j changes under different variables, and the intervals I_{jk} are not uniformly dense under different observations. These lead to troubles in constructing the WK metric. For a specific variable j and two observations i and l, the paper propose a method to create m common weights $\{\pi_{j1},\ldots,\pi_{jm}\}$ and find two uniformly dense intervals using these m common weights $\{J_{ij1},\ldots,J_{ijm}\}$ and $\{J_{lj1},\ldots,J_{ljm}\}$. So the presentation becomes more regular under variable j among all observations. We refer to Irpino & Verde [Irp06], Kim & Billard [Kim13], and Park et al. [Par19] more for details of this construction.

We then find the center c_{ijk} and radii r_{ijk} of each interval J_{ijk} for $i=1,\ldots,n$, $j=1,\ldots,p$ and $k=1,\ldots,m$. The jth variable of observations $\boldsymbol{x}_i$ and $\boldsymbol{x}_l$ can be expresses as: $\boldsymbol{x}_{ij}=\{(c_{ij1},r_{ij1}),\ldots,(c_{ijm},r_{ijm})\}$ and $\boldsymbol{x}_{lj}=\{(c_{lj1},r_{lj1}),\ldots,(c_{ljm},r_{ljm})\}$ with m common weights $\{\pi_{j1},\ldots,\pi_{jm}\}$. Then the square of the WK distance between $\boldsymbol{x}_i$ and $\boldsymbol{x}_l$ can be written as

$$\begin{aligned} d_W^2(\boldsymbol{x}_i,\boldsymbol{x}_l) &= \sum_{j=1}^{p}\sum_{k=1}^{m}\pi_{jk}\Big\{(c_{ijk}-c_{ljk})^2+\frac{1}{3}(r_{ijk}-r_{ljk})^2\Big\} \\ &= \sum_{j=1}^{p}\sum_{k=1}^{m}\pi_{jk}\left\{(\sqrt{\pi_{jk}}c_{ijk}-\sqrt{\pi_{jk}}c_{ljk})^2+(\sqrt{\frac{\pi_{jk}}{3}}r_{ijk}-\sqrt{\frac{\pi_{jk}}{3}}r_{ljk})^2\right\}. \end{aligned} \tag{A.8}$$

To incorporate the WK into a standard convex clustering setting, we first denote:

$$\begin{aligned} \boldsymbol{x}^*_{ij} &= (\sqrt{\pi_{j1}}c_{ij1},\ldots,\sqrt{\pi_{jm}}c_{ijm},\sqrt{\frac{\pi_{j1}}{3}}r_{ij1},\ldots,\sqrt{\frac{\pi_{jm}}{3}}r_{ijm})^T, \\ \boldsymbol{\theta}^*_{ij} &= (\sqrt{\pi_{j1}}\theta_{ij1},\ldots,\sqrt{\pi_{jm}}\theta_{ijm},\sqrt{\frac{\pi_{j1}}{3}}\theta_{ij1},\ldots,\sqrt{\frac{\pi_{jm}}{3}}\theta_{ijm})^T. \end{aligned}$$

Write $\boldsymbol{x}^*_i=\left(\boldsymbol{x}^{*T}_{i1},\ldots,\boldsymbol{x}^{*T}_{ip}\right)^T$ and $\boldsymbol{\theta}^*_i=\left(\boldsymbol{\theta}^{*T}_{i1},\ldots,\boldsymbol{\theta}^{*T}_{ip}\right)^T$. We can devise the clustering problem for histogram-valued data using the convex clustering problem:

$$\min_{\boldsymbol{\theta}^*}\left(\frac{1}{2}\sum_{i=1}^{n}\|\boldsymbol{x}^*_i-\boldsymbol{\theta}^*_i\|_2^2+\lambda\sum_{i<j}w_{ij}\|\boldsymbol{\theta}^*_i-\boldsymbol{\theta}^*_j\|_1\right). \tag{A.9}$$

A.3.2 Formulation with Sample Quantiles

Given observation $\boldsymbol{x}_i$ and $\boldsymbol{x}_l$ for the jth variable $\boldsymbol{x}_{ij}$ and $\boldsymbol{x}_{lj}$, we compute d sample quantiles $(q_{ij\alpha_1},\ldots,q_{ij\alpha_d})$ and $(q_{lj\alpha_1},\ldots,q_{lj\alpha_d})$ by assuming a specific distribution, for example, (10th, 20th,...,90th) quantiles from two histograms. This can be done efficiently by using the empirical quantile function [Par19]. Then the quantile distance is defined by:

$$d_Q^2(\boldsymbol{x}_i,\boldsymbol{x}_l) = \sum_{j=1}^{p}\sum_{k=1}^{d}\pi_{jk}(q_{ij\alpha_1}-q_{lj\alpha_k})^2.$$

Similarly, we define:

$$\begin{aligned} \boldsymbol{x}^*_{ij} &= (\sqrt{\pi_{j1}}q_{ij\alpha_1},\ldots,\sqrt{\pi_{jd}}q_{ij\alpha_d})^T, \\ \boldsymbol{\theta}^*_{ij} &= (\sqrt{\pi_{j1}}\theta_{ij\alpha_1},\ldots,\sqrt{\pi_{jd}}\theta_{ij\alpha_d})^T, \end{aligned}$$

with $\{\pi_{j1},\ldots,\pi_{jd}\}$ is extracted following same procedure in Section A.3.1. Write $\boldsymbol{x}^*_i = \left(\boldsymbol{x}^{*^T}_{i1},\ldots,\boldsymbol{x}^{*^T}_{ip}\right)^T$ and $\boldsymbol{\theta}^*_i = \left(\boldsymbol{\theta}^{*^T}_{i1},\ldots,\boldsymbol{\theta}^{*^T}_{ip}\right)^T$, we devise the convex clustering problem with quantile distance:

$$\min_{\boldsymbol{\theta}^*}\left(\frac{1}{2}\sum_{i=1}^{n}\|\boldsymbol{x}^*_i-\boldsymbol{\theta}^*_i\|_2^2+\lambda\sum_{i<j}w_{ij}\|\boldsymbol{\theta}^*_i-\boldsymbol{\theta}^*_j\|_1\right). \tag{A.10}$$

A.3.3 Drawbacks

No matter which formulation used, we argue that some ordering structure (location information) is missed. Let us go back to the time-series of daily measurements of average humidity in a year. Counting the frequency and generating another histogram $\boldsymbol{x}_{ij}$ under a jth variable (month) only considers the number of appearances of a specific value but does not record on which day this value appeared. For example, suppose that in January ($j=1$), the values are $\{1,2,\ldots,31\}$ from the first day to the last day in observation i. But observation l has $\{31,30,\ldots,1\}$ from the first day to the last day. Then $\boldsymbol{x}_{i1}$ will be same as $\boldsymbol{x}_{l1}$ in both WK and quantile distance. But we know two observations are quite different, at least under jth variable. In another word, the cvxclustrWK is not a good choice if the data also have some ordering structure under each variable, which is a common case in the time-series or the image data.

On the one hand, it reduces the dimensions by generating a histogram under a single piece, which can greatly reduce computational costs. But on the other hand, it does not capture enough characterizations of a histogram data, with the ultimate trade-off being possibly mis-specifying clusters within convex clustering framework. Furthermore, generating a histogram under each piece ignores joint properties since it always collects information on the margins.

APPENDIX

B

SUPPLEMENTARY MATERIALS FOR "A SHARP-MAJORIZED ADMM ALGORITHM TO SOLVE SHAPE RESTRICTED BINOMIAL REGRESSION PROBLEM"

B.1 Proof of Proposition 4

We prove the optimality of $\delta(\theta,\tilde{\theta})$ claimed in Proposition 4. Recall that given (N,Y), the binomial loss function is:

$$l(\theta) \quad = \quad -Y\theta + N\log(1+e^{\theta}).$$

And the δ function at some is:

$$\delta(\theta,\tilde{\theta}) \quad = \quad \frac{l(\theta)-l(\tilde{\theta})-l'(\tilde{\theta})(\theta-\tilde{\theta})}{\frac{1}{2}(\theta-\tilde{\theta})^2}.$$

Suppose that $\tilde{\theta}\neq 0$, we would like to show that $\max_{\theta\neq\tilde{\theta}}\ \delta(\theta,\tilde{\theta}) = \delta(-\tilde{\theta},\tilde{\theta})$, i.e., the maximum is achieved at $s=-\tilde{\theta}$.

Proof. Take derivative with respect to θ:

$$\delta'(\theta,\tilde{\theta}) = \frac{\frac{1}{2}(\theta-\tilde{\theta})^2\left[l'(\theta)+l'(\tilde{\theta})\right]-(\theta-\tilde{\theta})\left[l(\theta)-l(\tilde{\theta})\right]}{\frac{1}{4}(\theta-\tilde{\theta})^4}.$$

Suppose that s is the optimum. Then $\delta'(s,\tilde{\theta})$ should vanish, which is equivalent to:

$$\frac{l(s)-l(\tilde{\theta})}{s-\tilde{\theta}} = \frac{1}{2}\left[l'(s)+l'(\tilde{\theta})\right]. \tag{B.1}$$

Recall that $l(\theta)=-Y\theta+N\log(1+e^{\theta})$ and $l'(\theta)=-Y+N\frac{e^{\theta}}{1+e^{\theta}}$. Set $s=-\tilde{\theta}$, the left hand side of Equation (B.1) becomes:

$$\begin{aligned}
\frac{l(s)-l(\tilde{\theta})}{s-\tilde{\theta}} &= \frac{-Y(s-\tilde{\theta})+N\left[\log(1+e^{s})-\log(1+e^{\tilde{\theta}})\right]}{s-\tilde{\theta}} \\
&= -Y+\frac{N}{s-\tilde{\theta}}\left[\log(1+e^{-\tilde{\theta}})-\log(1+e^{\tilde{\theta}})\right] \\
&= -Y+\frac{N}{-2\tilde{\theta}}\left[\log(e^{-\tilde{\theta}}(1+e^{\tilde{\theta}}))-\log(1+e^{\tilde{\theta}})\right] \\
&= -Y+\frac{N}{-2\tilde{\theta}}(-\tilde{\theta}) \\
&= \frac{N}{2}-Y,
\end{aligned} \tag{B.2}$$

and the right hand side of (B.1) gives:

$$\begin{aligned}
\frac{1}{2}\left[l'(s)+l'(\tilde{\theta})\right] &= \frac{1}{2}\left[-2Y+N\left(\frac{e^{s}}{1+e^{s}}+\frac{e^{\tilde{\theta}}}{1+e^{\tilde{\theta}}}\right)\right] \\
&= -Y+\frac{N}{2}\left(\frac{e^{\tilde{-\theta}}}{1+e^{-\tilde{\theta}}}+\frac{e^{\tilde{\theta}}}{1+e^{\tilde{\theta}}}\right) \\
&= -Y+\frac{N}{2}\left(\frac{1}{1+e^{\tilde{\theta}}}+\frac{e^{\tilde{\theta}}}{1+e^{\tilde{\theta}}}\right) \\
&= -Y+\frac{N}{2}.
\end{aligned} \tag{B.3}$$

The equivalence of (B.2) and (B.3) implies that $s=-\tilde{\theta}$ is a stationary point. We then check the second-order derivative:

$$\delta''(\theta,\tilde{\theta}) = \frac{(\theta-\tilde{\theta})^2 l''(\theta)-\left[l'(\theta)-l'(\tilde{\theta})(\theta-\tilde{\theta})\right]}{\frac{1}{2}(\theta-\tilde{\theta})^4}. \tag{B.4}$$

At a true maximum point $s=-\tilde{\theta}$ we must have:

$$\delta''(s,\tilde{\theta})\leq 0 \iff (s-\tilde{\theta})^2 l''(s)-\left[l'(\theta)-l'(\tilde{\theta})(s-\tilde{\theta})\right]\leq 0$$

$$\Leftrightarrow \quad \frac{e^{\tilde{\theta}}}{1+e^{\tilde{\theta}}} \leq -\frac{1}{2\tilde{\theta}}(1-e^{\tilde{\theta}}).$$

It can be shown that:

$$h(x) = \frac{e^x}{1+e^x} + \frac{1}{2x}(1-e^x) \tag{B.5}$$

is negative for all $x \neq 0$. So the second-order condition is also satisfied and $s = -\tilde{\theta}$ is the maximum point. □

Here we only show that $s = -\tilde{\theta}$ is a stationary point and satisfies the second-order condition. We claim that this local maximum is actually a global maximum from empirical studies. It is better to illustrate from the monotonicity perspective: $\delta(\theta, \tilde{\theta})$ first increases up to $-\tilde{\theta}$ and then decrease. We leave this as a conjecture. Please refer to [Lee09] for more discussions.

B.2 Proof of Proposition 5

Recall that we $a(\tilde{\theta})$ as:

$$a(\tilde{\theta}) \quad = \quad \begin{cases} \delta(-\tilde{\theta}, \tilde{\theta}) = \frac{N}{2\tilde{\theta}^2}\left[\log\frac{1+e^{-\tilde{\theta}}}{1+e^{\tilde{\theta}}} + 2\tilde{\theta}\frac{e^{\tilde{\theta}}}{1+e^{\tilde{\theta}}}\right] & \text{if } \tilde{\theta} \neq 0, \\ \frac{1}{4}N & \text{otherwise.} \end{cases}$$

We will show $a(\tilde{\theta}) \leq \frac{1}{4}N$, which implies it produces a "smaller" quadratic majorization than the regular L-majorization with $L = \frac{1}{4}N$.

Proof. We only need to show $a(\tilde{\theta}) \leq L$ for $\tilde{\theta} \neq 0$. Suppose that $s = -\tilde{\theta}$ is the optimum. Then $a(\tilde{\theta}) = \delta(s, \tilde{\theta})$:

$$a(\tilde{\theta}) \quad = \quad \frac{l(s) - l(\tilde{\theta}) - l'(\tilde{\theta})(s-\tilde{\theta})}{\frac{1}{2}(s-\tilde{\theta})^2}. \tag{B.6}$$

As the optimum, s must satisfy the first order condition in (B.1):

$$\frac{l(s) - l(\tilde{\theta})}{s-\tilde{\theta}} \quad = \quad \frac{1}{2}\left[l'(s) + l'(\tilde{\theta})\right].$$

Apply this condition to Equation (B.6) produces:

$$a(\tilde{\theta}) \quad = \quad \frac{l'(s) - l'(\tilde{\theta})}{s-\tilde{\theta}}.$$

Note that l is twice differentiable, we can apply the Mean Value Theorem. First assume that $\tilde{\theta} < 0$,

then $s > \tilde{\theta}$. There exists $\omega \in (\tilde{\theta}, s)$ such that:

$$a(\tilde{\theta}) \;=\; \frac{l''(\omega)(s-\tilde{\theta})}{s-\tilde{\theta}} \;=\; l''(\omega) \;\leq\; \frac{1}{4}N. \tag{B.7}$$

Similarly, assume that $\tilde{\theta} > 0$, then $s < \tilde{\theta}$. There exists $\omega^* \in (s, \tilde{\theta})$ such that:

$$a(\tilde{\theta}) \;=\; \frac{l''(\omega^*)(\tilde{\theta}-s)}{\tilde{\theta}-s} \;=\; l''(\omega^*) \;\leq\; \frac{1}{4}N. \tag{B.8}$$

So $a(\tilde{\theta}) \leq L = \frac{1}{4}N$ for all $\tilde{\theta}$. □

APPENDIX

C

SUPPLEMENTARY MATERIALS FOR "A MAJORIZATION-MINIMIZATION GAUSS-NEWTON METHOD FOR 1-BIT MATRIX COMPLETION"

C.1 Proof of Proposition 7

We work on the following function:

$$\phi_\sigma(x) = \frac{f''(x)}{\Phi(x/\sigma)} - \left[\frac{f'(x)}{\Phi(x/\sigma)}\right]^2, \tag{C.1}$$

where $f(x) = \Phi(x/\sigma)$ and Φ is the cumulative distribution function of a standard normal under the probit model. We want to show function ϕ_σ is: i) non-decreasing; ii) $\lim_{x\to+\infty} \phi_\sigma(x) = 0$ and iii) $\lim_{x\to-\infty} \phi_\sigma(x) = -\frac{1}{\sigma^2}$. The limit under $x \to \infty$ is straightforward: it is easy to detect that the numerators in Equation (C.1) go to 0, and the denominators go to 1 when $x \to \infty$, which produces the 0 as the final result. For the limit under $x \to -\infty$, we use an online analysis software called `Wolfram|Alpha` [Inc]. It shows that:

$$\lim_{x\to-\infty} \phi_\sigma(x) = \lim_{x\to-\infty} \frac{-\frac{2}{\pi x^2} + \frac{6\sigma^2}{\pi x^4} + \mathcal{O}(\frac{1}{x^6})}{\left(-\frac{\sigma\sqrt{2}}{\sqrt{\pi}x} + \frac{\sigma^3\sqrt{2}}{\sqrt{\pi}x^3} - \frac{3\sigma^5\sqrt{2}}{\sqrt{\pi}x^5} + \mathcal{O}(\frac{1}{x^6})\right)^2}$$

$$= -\frac{1}{\sigma^2}.$$

The proof requires some complex analysis techniques and uses the Laurent series to approximate the integrals. We then investigate the monotonicity. Take the derivative we get:

$$\begin{aligned}\phi_\sigma^{'}(x) &= \frac{f'''(x)\Phi(x/\sigma)-f''(x)f'(x)}{\Phi(x/\sigma)^2}-2\frac{f'(x)}{\Phi(x/\sigma)}\frac{f''(x)\Phi(x/\sigma)-f'(x)^2}{\Phi(x/\sigma)^2} \\ &= \frac{1}{\Phi(x/\sigma)^3}\Big(f'''(x)\Phi\big(\frac{x}{\sigma}\big)^2-3f''(x)f'(x)\Phi\big(\frac{x}{\sigma}\big)+2f'(x)^3\Big). \end{aligned} \tag{C.2}$$

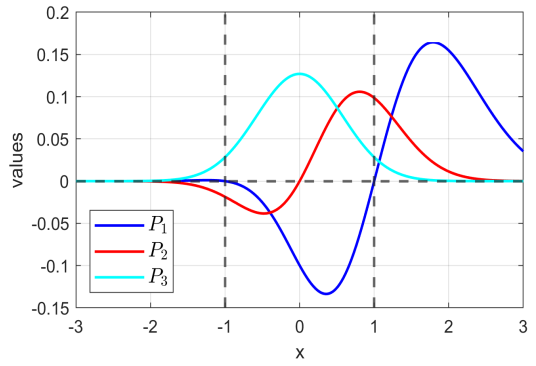
We denote:

$$\begin{aligned} P_1 &= f'''(x)\Phi\big(\frac{x}{\sigma}\big)^2 \\ &= \Phi\big(\frac{x}{\sigma}\big)^2\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{x^2}{2\sigma^2}}\left(\frac{x^2}{\sigma^4}-\frac{1}{\sigma^2}\right), \\ P_2 &= -3f''(x)f'(x)\Phi\big(\frac{x}{\sigma}\big) \\ &= \Phi\big(\frac{x}{\sigma}\big)\frac{1}{2\pi\sigma^2}e^{-\frac{x^2}{\sigma^2}}\left(\frac{3x}{\sigma^2}\right), \\ P_3 &= 2f'(x)^3 \\ &= 2\left(\frac{1}{\sqrt{2\pi}\sigma}\right)^3 e^{-\frac{3x^2}{2\sigma^2}}. \end{aligned}$$

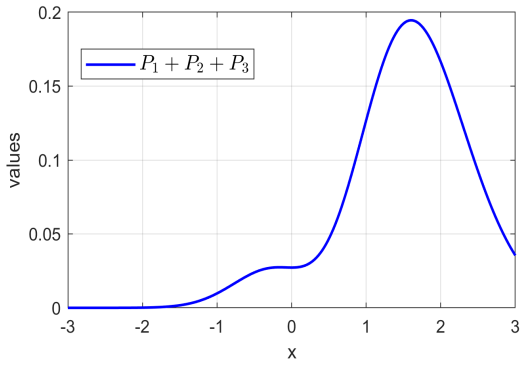
To show the function is non-decreasing, we need $\phi'_\sigma(x)\geq 0$, equivalent to $P_1+P_2+P_3\geq 0$. We illustrate an example under $\sigma=1$ by plotting P_1, P_2, P_3 in one panel. Figure C.1a shows the behaviors of these three parts as a function of x on the interval $[-3,3]$. It is easy to see when $\sigma\geq 1$, all three parts are positive. The case becomes complex when $\sigma<1$. When $\sigma\in(0,1)$, both P_2 and P_3 are positive, which surpasses the negative value introduced by P_1; when $\sigma\in(-1,0)$, the positiveness given by P_3 overwhelm the negative values P_2 and P_3; when $\sigma\leq 1$, P_1 and P_3 give positive values that exceed the negative P_2. Figure C.1b shows the summation $P_1+P_2+P_3$, we see they always sum up to a positive value. Numerical experiments with other choices of σ give the same results. Even though we can not thoroughly prove $\phi_\sigma(x)\geq 0$, the empirical testings tell us this derivative is always positive, which indicates that the function $\phi_\sigma(x)$ is a non-decreasing function. We leave this as a conjecture which requires future work.

C.2 Equivalence of the Jacobian Matrices

In this section, we show the equivalence between the Jacobian matrices $\boldsymbol{J}(\boldsymbol{p})$ in GN algorithm and the matrix $\boldsymbol{J}$ defined in (4.33) under our MMGN update. Without loss of generality, assume that we observe all the entries, i.e., $\Omega=[m]\times[n]$ or $\tilde{\Omega}=[mn]$. If some entries are missing, then we just need to delete the rows of matrix $\boldsymbol{J}$ and $\boldsymbol{J}(\boldsymbol{p})$ whose corresponding indices are not in $\tilde{\Omega}$. The equivalence



(a) Curves of P_1, P_2 and P_3.



(b) Curve of $P_1 + P_2 + P_3$.

Figure C.1 Plots of three parts of $\phi_\sigma(x)$ in (C.2), P_1, P_2, P_3, and their summation as a function of x on interval $[-3,3]$ under the probit model with $\sigma = 1$. Panel (a): curves of P_1, P_2, P_3 respectively. The negative values are overwhelmed by the positive values. Panel (b): curve of $P_1 + P_2 + P_3$. The summation of them is always non-negative, which implies the $\phi_\sigma(x)$ is non-decreasing.

does not change. Note that we assume $\tilde{\boldsymbol{M}} = \boldsymbol{U}\boldsymbol{V}^T$ with $\boldsymbol{U} \in \mathbb{R}^{m\times r}$ and $\boldsymbol{V} \in \mathbb{R}^{n\times r}$ are known. We write:

$$\boldsymbol{U} = \begin{pmatrix} u_{11} & u_{12} & \dots & u_{1r} \\ u_{21} & u_{22} & \dots & u_{2r} \\ \vdots & \vdots & \vdots & \vdots \\ u_{m1} & u_{m2} & \dots & u_{mr} \end{pmatrix}, \quad \boldsymbol{V} = \begin{pmatrix} v_{11} & v_{12} & \dots & v_{1r} \\ v_{21} & v_{22} & \dots & v_{2r} \\ \vdots & \vdots & \vdots & \vdots \\ v_{n1} & v_{m2} & \dots & v_{nr} \end{pmatrix}.$$

Then the variable $\boldsymbol{p}$ can be expressed as:

$$\boldsymbol{p} = \begin{pmatrix} \text{vec}(\boldsymbol{U}) \\ \text{vec}(\boldsymbol{V}^T) \end{pmatrix}$$

$$= \begin{pmatrix} u_{11} \\ \vdots \\ u_{m1} \\ u_{12} \\ \vdots \\ u_{m2} \\ \vdots \\ u_{1r} \\ \vdots \\ u_{mr} \\ v_{11} \\ \vdots \\ v_{1r} \\ \vdots \\ v_{n1} \\ \vdots \\ v_{nr} \end{pmatrix}. \tag{C.3}$$

The matrix $\boldsymbol{J}$ under the MMGN algorithm in (4.33) is defined as:

$$\boldsymbol{J} = [\boldsymbol{V} \otimes \boldsymbol{I}_m \mid \boldsymbol{I}_n \otimes \boldsymbol{U}] \in \mathbb{R}^{mn \times (m+n)r}. \tag{C.4}$$

The matrix $\boldsymbol{J}(\boldsymbol{p})$ with entries $\boldsymbol{J}_{ij}$ under the standard GN algorithm defined as:

$$\boldsymbol{J}_{ij} = \frac{\partial r_i(\boldsymbol{p})}{\partial p_j}. \tag{C.5}$$

with $i = 1, \ldots, mn$ and $j = 1, \ldots, (m+n)r$. And the residual r_i is defined as:

$$r_i(\boldsymbol{p}) = \text{vec}(\boldsymbol{U}\boldsymbol{V}^T)_i - \text{vec}(\boldsymbol{C})_i,$$

where $\boldsymbol{C} \in \mathbb{R}^{m \times n}$ is a constant matrix that does not depend on $\boldsymbol{U}$ and $\boldsymbol{V}$. We will illustrate the equivalence of matrices in (C.4) and in (C.5) by analyzing the first two entries of residuals: r_1 and r_2. Note that $i = 1, 2$, we have:

$$\begin{aligned} r_1(\boldsymbol{p}) &= \text{vec}(\boldsymbol{U}\boldsymbol{V}^T)_1 - \text{vec}(\boldsymbol{Z})_1 \\ &= (\boldsymbol{U}\boldsymbol{V}^T)_{11} - \boldsymbol{Z}_{11} \\ &= u_{11}v_{11} + u_{12}v_{12} + \cdots + u_{1r}v_{1r} - \boldsymbol{Z}_{11}, \\ r_2(\boldsymbol{p}) &= \text{vec}(\boldsymbol{U}\boldsymbol{V}^T)_2 - \text{vec}(\boldsymbol{Z})_2 \end{aligned}$$

$$
\begin{aligned}
&= (\boldsymbol{U}\boldsymbol{V}^T)_{21} - \boldsymbol{Z}_{21} \\
&= u_{21}v_{11} + u_{22}v_{12} + \cdots + u_{2r}v_{1r} - \boldsymbol{Z}_{21}.
\end{aligned}
$$

Take partial derivative with respect to $\boldsymbol{p}$ produces the first two rows of $J(\boldsymbol{p})$:

$$
\begin{pmatrix}
\overbrace{v_{11}}^{m} & 0 & \dots & 0 & \overbrace{v_{12}}^{m} & 0 & \dots & 0 & \dots & \overbrace{v_{1r}}^{m} & 0 & \dots & 0 & \overbrace{u_{11}}^{r} & u_{12} & \dots & u_{1r} & \overbrace{0}^{(n-1)r} & 0 & \dots & 0 \\
0 & v_{11} & \dots & 0 & 0 & v_{12} & \dots & 0 & \dots & 0 & v_{1r} & \dots & 0 & u_{21} & u_{22} & \dots & u_{2r} & 0 & 0 & \dots & 0
\end{pmatrix},
$$

which is the first two rows of $\boldsymbol{J} = [\boldsymbol{V} \otimes \boldsymbol{I}_m \mid \boldsymbol{I}_n \otimes \boldsymbol{U}]$. The rest rows can be derived from the same procedure.

C.3 Computation in Backtracking Line Search

This part proves the Equation (4.45):

$$
\langle \nabla \ell(\boldsymbol{\theta}), \boldsymbol{\theta}^+ \rangle = -\left\langle \boldsymbol{Y} \circ \frac{f'(\tilde{\boldsymbol{M}})}{f(\boldsymbol{Y} \circ \tilde{\boldsymbol{M}})}, \Delta\boldsymbol{U}^+\boldsymbol{V}^T + \boldsymbol{U}(\Delta\boldsymbol{V}^+)^T \right\rangle_\Omega, \tag{C.6}
$$

where $\tilde{\boldsymbol{M}} = \boldsymbol{U}\boldsymbol{V}^T$ is the current estimate; $\Delta\boldsymbol{U}^+$ and $\Delta\boldsymbol{V}^+$ are the descending directions; $\boldsymbol{\theta}^+ = \begin{pmatrix} \Delta\boldsymbol{u}^+ \\ \Delta\tilde{\boldsymbol{v}}^+ \end{pmatrix}$ and $\boldsymbol{\theta} = \begin{pmatrix} \Delta\boldsymbol{u} \\ \Delta\tilde{\boldsymbol{v}} \end{pmatrix}$ are vectorized versions of $\Delta\boldsymbol{U}^+$, $\Delta\boldsymbol{V}^+$ and $\boldsymbol{U}$, $\boldsymbol{V}$ respectively; the objective function $\ell(\boldsymbol{U}, \boldsymbol{V})$ is:

$$
\ell(\boldsymbol{U}, \boldsymbol{V}) = -\left[\sum_{(i,j)\in\Omega} Z_{ij} \log f\big((\boldsymbol{U}\boldsymbol{V}^T)_{ij}\big) + (1 - Z_{ij}) \log\Big(1 - f\big((\boldsymbol{U}\boldsymbol{V}^T)_{ij}\big)\Big) \right], \tag{C.7}
$$

with $\boldsymbol{Z} = (1 + \boldsymbol{Y})/2$. It is easy to check that:

$$
\langle \nabla \ell(\boldsymbol{\theta}), \boldsymbol{\theta}^+ \rangle = \sum_{i'=1}^{m} \sum_{t=1}^{r} \frac{\partial}{\partial U_{i't}} \ell(\boldsymbol{U}, \boldsymbol{V}) \Delta U^+_{i't} + \sum_{j'=1}^{n} \sum_{t=1}^{r} \frac{\partial}{\partial V_{j't}} \ell(\boldsymbol{U}, \boldsymbol{V}) \Delta V^+_{j't}. \tag{C.8}
$$

We start from analyzing the derivative of a single entry $U_{i't}$ for a specific $i' = 1, \dots, m$ and $t = 1, \dots, r$. The rule of matrix multiplication indicates that $U_{i't}$ appears only in the i'th row of matrix $\boldsymbol{U}\boldsymbol{V}^T$ among the observed entries. Specifically, define $\Omega_{i'} = \{j = 1 \dots, n \mid (i', j) \in \Omega\}$:

$$
\begin{aligned}
\frac{\partial}{\partial U_{i't}} \sum_{(i,j)\in\Omega} \log f\big((\boldsymbol{U}\boldsymbol{V}^T)_{ij}\big) &= \sum_{j\in\Omega_{i'}} \left[\frac{\partial}{\partial U_{i't}} \log f\left((\boldsymbol{U}\boldsymbol{V}^T)_{i'j}\right) \right] \\
&= \sum_{j\in\Omega_{i'}} \left[\frac{\partial}{\partial U_{i't}} \log f(U_{i'1}V_{j1} + U_{i'2}V_{j2} + \cdots + U_{i'r}V_{jr}) \right]
\end{aligned}
$$

$$= \sum_{j\in\Omega_{i'}}\left[\frac{f'(\tilde{M}_{i'j})}{f(\tilde{M}_{i'j})}V_{jt}\right].$$

Thus:

$$\begin{aligned}
\sum_{i'=1}^{m}\sum_{t=1}^{r}\left[\frac{\partial}{\partial U_{i't}}\sum_{(i,j)\in\Omega}\log f\Big((\boldsymbol{U}\boldsymbol{V}^T)_{ij}\Big)\right]\Delta U^+_{i't} &= \sum_{i'=1}^{m}\sum_{t=1}^{r}\sum_{j\in\Omega_{i'}}\left[\frac{f'(\tilde{M}_{i'j})}{f(\tilde{M}_{i'j})}V_{jt}\right]\Delta U^+_{i't}\\
&= \sum_{i'=1}^{m}\sum_{j\in\Omega_{i'}}\frac{f'(\tilde{M}_{i'j})}{f(\tilde{M}_{i'j})}\sum_{t=1}^{r}\Delta U^+_{i't}V_{jt}\\
&= \sum_{i'=1}^{m}\sum_{j\in\Omega_{i'}}\frac{f'(\tilde{M}_{i'j})}{f(\tilde{M}_{i'j})}(\Delta\boldsymbol{U}^+\boldsymbol{V}^T)_{i'j}\\
&= \Big\langle\frac{f'(\tilde{\boldsymbol{M}})}{f(\tilde{\boldsymbol{M}})},\Delta\boldsymbol{U}^+\boldsymbol{V}^T\Big\rangle_\Omega. \qquad \text{(C.9)}
\end{aligned}$$

Similarly, we have:

$$\begin{aligned}
\sum_{i'=1}^{m}\sum_{t=1}^{r}\left[\frac{\partial}{\partial U_{i't}}\sum_{(i,j)\in\Omega}\log\Big[1-f\Big((\boldsymbol{U}\boldsymbol{V}^T)_{ij}\Big)\Big]\right]\Delta U^+_{i't} &= \sum_{i'=1}^{m}\sum_{t=1}^{r}\sum_{j\in\Omega_{i'}}\frac{\partial}{\partial U_{i't}}\log\Big[1-f\Big((\boldsymbol{U}\boldsymbol{V}^T)_{ij}\Big)\Big]\Delta U^+_{i't}\\
&= \sum_{i'=1}^{m}\sum_{t=1}^{r}\sum_{j\in\Omega_{i'}}\left[\frac{-f'(\tilde{M}_{i'j})}{f(-\tilde{M}_{i'j})}V_{jt}\right]\Delta U^+_{i't}\\
&= \sum_{i'=1}^{m}\sum_{j\in\Omega_{i'}}\frac{-f'(\tilde{M}_{i'j})}{f(-\tilde{M}_{i'j})}\sum_{t=1}^{r}\Delta U^+_{i't}V_{jt}\\
&= \sum_{i'=1}^{m}\sum_{j\in\Omega_{i'}}\frac{-f'(\tilde{M}_{i'j})}{f(-\tilde{M}_{i'j})}(\Delta\boldsymbol{U}^+\boldsymbol{V}^T)_{i'j}\\
&= \Big\langle\frac{-f'(\tilde{\boldsymbol{M}})}{f(-\tilde{\boldsymbol{M}})},\Delta\boldsymbol{U}^+\boldsymbol{V}^T\Big\rangle_\Omega. \qquad \text{(C.10)}
\end{aligned}$$

Combine the results in (C.9) and (C.10), we have:

$$\sum_{i'=1}^{m}\sum_{t=1}^{r}\frac{\partial}{\partial U_{i't}}\ell(\boldsymbol{U},\boldsymbol{V})\Delta U^+_{i't} = -\Big\langle\boldsymbol{Y}\circ\frac{f'(\tilde{\boldsymbol{M}})}{f(\boldsymbol{Y}\circ\tilde{\boldsymbol{M}})},\Delta\boldsymbol{U}^+\boldsymbol{V}^T\Big\rangle_\Omega. \qquad \text{(C.11)}$$

Note that matrix $\boldsymbol{U}$ and $\boldsymbol{V}$ performs symmetrically in $\ell(\boldsymbol{U},\boldsymbol{V})$ in (C.7). So we can repeat the similar procedure to compute the partial derivative with respect to $\boldsymbol{V}$:

$$\sum_{j'=1}^{n}\sum_{t=1}^{r}\frac{\partial}{\partial V_{j't}}\ell(\boldsymbol{U},\boldsymbol{V})\Delta V^+_{j't} = -\Big\langle\boldsymbol{Y}\circ\frac{f'(\tilde{\boldsymbol{M}})}{f(\boldsymbol{Y}\circ\tilde{\boldsymbol{M}})},\boldsymbol{U}(\Delta\boldsymbol{V}^+)^T\Big\rangle_\Omega. \qquad \text{(C.12)}$$

So Equation (C.8) finally becomes:

$$\begin{aligned}\langle \nabla \ell(\boldsymbol{\theta}), \boldsymbol{\theta}^{+} \rangle &= \sum_{i'=1}^{m} \sum_{t=1}^{r} \frac{\partial}{\partial U_{i't}} \ell(\boldsymbol{U}, \boldsymbol{V}) \Delta U_{i't}^{+} + \sum_{j'=1}^{n} \sum_{t=1}^{r} \frac{\partial}{\partial V_{j't}} \ell(\boldsymbol{U}, \boldsymbol{V}) \Delta V_{j't}^{+} \\ &= -\left\langle \boldsymbol{Y} \circ \frac{f'(\tilde{\boldsymbol{M}})}{f(\boldsymbol{Y} \circ \tilde{\boldsymbol{M}})}, \Delta \boldsymbol{U}^{+} \boldsymbol{V}^{T} + \boldsymbol{U} (\Delta \boldsymbol{V}^{+})^{T} \right\rangle_{\Omega}.\end{aligned}$$