Ph.D. DISSERTATION

# A STUDY OF LOW-RANK MATRIX COMPLETION ALGORITHM BASED ON RIEMANNIAN OPTIMIZATION AND GRAPH NEURAL NETWORK

리만 최적화와 그래프 신경망에 기반한 저 랭크
행렬완성 알고리듬에 관한 연구

BY

Nguyen Trung Luong
FEBRUARY 2020

DEPARTMENT OF ELECTRICAL AND
COMPUTER ENGINEERING
COLLEGE OF ENGINEERING
SEOUL NATIONAL UNIVERSITY

공학박사 학위논문

# A Study of Low-rank Matrix Completion Algorithm Based on Riemannian Optimization and Graph Neural Network

## 리만 최적화와 그래프 신경망에 기반한 저 랭크 행렬완성 알고리듬에 관한 연구

2020년 2월

서울대학교 대학원

전기·정보공학부

Nguyen Trung Luong

# A Study of Low-rank Matrix Completion Algorithm Based on Riemannian Optimization and Graph Neural Network

지도교수 심 병 효

이 논문을 공학박사 학위논문으로 제출함

2019년 11월

서울대학교 대학원

전기·정보공학부

Nguyen Trung Luong

Nguyen Trung Luong의 박사 학위논문을 인준함

2019년 12월

위 원 장 이 광 복 (인)

부위원장 심 병 효 (인)

위 원 박 대 영 (인)

위 원 양 인 순 (인)

위 원 최 준 원 (인)

# Abstract

In recent years, low-rank matrix completion (LRMC) has received much attention as a paradigm to recover the unknown entries of a matrix from partial observations. It has a wide range of applications in many areas, including recommendation system, phase retrieval, IoT localization, image denoising, milimeter wave (mmWave) communication, to name just a few. In this dissertation, we present a comprehensive overview of low-rank matrix completion. In order to have better view, insight, and understanding of potentials and limitations of LRMC, we present early scattered results in a structured and accessible way. To be specific, we classify the state-of-the-art LRMC techniques into two main categories and then explain each category in detail. We further discuss issues to be considered, including intrinsic properties required for the matrix recovery, when one would like to use LRMC techniques. However, conventional LRMC techniques have been most successful on a general setting of the low-rank matrix, say, Gaussian random matrix. In many practical situations, the desired low rank matrix might have an underlying non-Euclidean structure, such as graph or manifold structure.

In our work, we show that such additional data structures can be exploited to improve the recovery performance of LRMC in real-life applications. In particular, we propose a Euclidean distance matrix completion algorithm for internet of things (IoT) network localization. In our approach, we express the Euclidean distance matrix as a function of the low rank positive semidefinite (PSD) matrix. Since the set of these PSD matrices forms a Riemannian manifold in which the notation of differentiability can be defined, we can recycle, after a proper modification, an algorithm in the Euclidean space. In order to solve the low-rank matrix completion, we propose a modified conjugate gradient algorithm, referred to as localization in Riemannian manifold using conjugate gradient (LRM-CG). We also show that the proposed LRM-CG algorithm

can be easily extended to the scenario in which the observed pairwise distances are contaminated by the outliers. In fact, by modeling outliers as a sparse matrix and then adding a regularization term of the outlier matrix into the low-rank matrix completion problem, we can effectively control the outliers. From the convergence analysis, we show that LRM-CG converges linearly to the original Euclidean distance matrix under the extended Wolfe's conditions. From the numerical experiments, we demonstrate that LRM-CG as well as its extended version is effective in recovering the Euclidean distance matrix.

In order to solve the LRMC problem in which the desired low-rank matrix can be expressed using a graph model, we also propose a graph neural network (GNN) scheme. Our approach, referred to as graph neural network-based low-rank matrix completion (GNN-LRMC), is to use a modified convolution operation to extract the features across the graph domain. The feature data enable the training process of the proposed GNN to reconstruct the unknown entries and also optimize the graph model of the desired low-rank matrix. We demonstrate the reconstruction performance of the proposed GNN-LRMC using synthetic and real-life datasets.

**keywords**: low-rank matrix completion, Frobenius norm minimization, localization, Riemannian optimization, graph neural network.

**student number**: 2015-30751

# Contents

# List of Tables

# List of Figures

# Notation

| | |
|---|---|
| $\|\mathbf{A}\|$ | the spectral norm (i.e., the largest singular value) of $\mathbf{A}$ |
| $\|\mathbf{A}\|_*$ | the nuclear norm (i.e., the sum of singular values) of $\mathbf{A}$ |
| $\|\mathbf{A}\|_F$ | the Frobenius norm of $\mathbf{A}$ |
| $\mathbf{A}^T \in \mathbb{R}^{n_2 \times n_1}$ | the transpose of $\mathbf{A}$ |
| $\mathbf{A}^H \in \mathbb{C}^{n_2 \times n_1}$ | the conjugate transpose of $\mathbf{A}$ |
| $\text{diag}(\mathbf{A}) \in \mathbb{R}^n$ | vector formed by the diagonal entries of $\mathbf{A}$ |
| $\text{rank}(\mathbf{A})$ | the rank of $\mathbf{A}$ |
| $\text{Sym}(\mathbf{A})$ | the symmetric component of $\mathbf{A}$, i.e., $1/2(\mathbf{A} + \mathbf{A}^T)$ |
| $\text{Skew}(\mathbf{A})$ | the skew-symmetric component of $\mathbf{A}$, i.e., $1/2(\mathbf{A} - \mathbf{A}^T)$ |
| $\text{vec}(\mathbf{A})$ | the vectorization of $\mathbf{A}$ |
| $\langle \mathbf{A}, \mathbf{B} \rangle = \text{tr}(\mathbf{A}^T\mathbf{B})$ | the inner product of $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{n_1 \times n_2}$ |
| $\mathbf{A} \odot \mathbf{B}$ | the Hadamard product of $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{n_1 \times n_2}$ |
| $\mathbf{Q}_\perp \in \mathbb{R}^{n \times (n-k)}$ | an orthogonal complement of $\mathbf{Q} \in \mathbb{R}^{n \times k}$ |
| $\mathbf{I}_d$ | the $d$-dimensional identity matrix |
| $\mathbf{0}$ | matrix with all-zero entries |
| $\mathbf{1}$ | matrix with all-one entries |
| $\mathbf{a}_i \in \mathbb{R}^{n_1}$ | the $i$-th column of $\mathbf{A} \in \mathbb{R}^{n_1 \times n_2}$ |
| $\mathbf{e}_i$ | the $i$-th vector of the standard basis in $\mathbb{R}^n$ |
| $\text{diag}(\mathbf{a}) \in \mathbb{R}^{n \times n}$ | diagonal matrix formed by $\mathbf{a}$ |
| $\text{eye}(\mathbf{a})$ | diagonal matrix with the main diagonal being $\mathbf{a}$ |
| $\sigma_i(\mathbf{A})$ | the $i$-th largest singular value of $\mathbf{A}$ |
| $\nabla_{\mathbf{Y}} f(\mathbf{Y})$ | the Euclidean gradient of $f(\mathbf{Y})$, i.e., $[\nabla_{\mathbf{Y}} f(\mathbf{Y})]_{ij} = \frac{\partial f(\mathbf{Y})}{\partial y_{ij}}$ |
| $g \circ f$ | the composite mapping of two mappings $f$ and $g$ |

# Chapter 1

# Introduction

## 1.1 Motivation

In the era of big data, the low rank matrix has become a useful and popular means to express two-dimensional information. One well-known example is the rating matrix in the recommendation systems that represents users' tastes on products [1]. Since users expressing similar ratings on multiple products tend to have the same interest for a new product, columns associated with users sharing the same interest are highly likely to be the same, resulting in the low rank structure of the rating matrix. Another example is the Euclidean distance matrix formed by the pairwise distances of a large number of sensor nodes. Since the rank of an Euclidean distance matrix in the $k$-dimensional Euclidean space is at most $k + 2$ (if $k = 2$, then the rank is 4), this matrix can be readily considered as a low-rank matrix [2, 3].

One major benefit of the low rank matrix representation is that the essential information, expressed in terms of degree of freedom, in the matrix is much smaller than the total number of entries. Therefore, even though the number of observed entries is small, we still have a chance to recover the whole matrix. There are variety of scenarios where the number of observed entries of a matrix is tiny. In the recommendation systems, for example, users are recommended to submit the feedback in a form of rat-

ing number, e.g., 1 to 5 for the purchased product. However, users often do not want to leave a feedback and thus the rating matrix will have many missing entries. In the example of IoT networks, sensor nodes have a limitation on the radio communication range or the power outage so that only small portion of entries in the Euclidean distance matrix are available.

When there is no restriction on the rank of a matrix, the problem to recover unknown entries of the matrix from partial observed entries is ill-posed. This is because any value can be assigned to unknown entries, which in turn means that there are infinite number of matrices that agree with the observed entries. As a simple example, consider the following $2 \times 2$ matrix with one unknown entry marked ?

$$\mathbf{M} = \begin{bmatrix} 1 & 5 \\ 2 & ? \end{bmatrix}. \tag{1.1}$$

If $\mathbf{M}$ is a full rank, i.e., the rank of $\mathbf{M}$ is two, then any value can be assigned to ?. Whereas, if $\mathbf{M}$ is a low-rank matrix (the rank is one in this trivial example), two columns differ by only a constant and hence unknown element ? can be easily determined using a linear relationship between two columns (? = 10). This example is obviously simple, but the fundamental principle to recover a large dimensional matrix is essentially not much different from this and the *low-rank* constraint plays a central role in recovering unknown entries of the matrix.

In recent years, low rank matrix completion (LRMC) has become a powerful and attractive tool to complete the low rank matrices using a subset of entries. This paradigm has been received much attention ever since the works of Fazel [4], Candes and Recht [5], and Candes and Tao [6]. Over the years, various LRMC techniques have been proposed [7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21]. These include convex optimization, singular value thresholding (SVT), alternating minimization, heuristic greedy technique, alternating steepest descent, and optimization over Riemannian manifolds, to name just a few.

Basically, the LRMC problem can be modeled as a rank minimization problem to

find the lowest rank matrix given the observed entries. However, since the rank function is nonlinear, non-convex, and non-smooth, it is computationally infeasible to solve the rank minimization problem directly. In fact, it is known that the rank minimization problem is a NP-hard problem. Over the years, various approaches relaxing the rank constraint have been proposed. Roughly speaking, depending on the way of using the rank information, LRMC techniques can be classified into two main categories: the LRMC using the rank information and those without the rank information.

When one tries to understand LRMC, there are fundamental issues and principles that one needs to be aware of. There are two key properties characterizing the LRMC problem: the *sparsity* of the observed entries and the *incoherence* of the matrix. Sparsity indicates that an accurate reconstruction of the undersampled matrix is possible even when the number of observed entries is very small, while incoherence indicates that nonzero entries of the matrix should be spread out widely for the efficient recovery of a low-rank matrix.

Further, when the desired low-rank matrix has an underlying structure, such as graph or manifold, we want to make the most of the given structure to maximize profits in terms of performance and computational complexity. In particular, to cope with the Euclidean distance matrix completion in which the Euclidean distance matrix has an underlying Riemannian structure, we cast the LRMC problem into the unconstrained optimization problem on a Riemannian manifold. Advantage of Riemannian manifold is that the notion of differentiability is well-defined and hence many useful ingredients for solving optimization problems can be used in the design of the LRMC algorithm.

We also propose a graph neural network (GNN)-based scheme to reconstruct the low-rank matrix with underlying graph structure. In our approach, we use a modified convolutional filter to extract the meaningful features across the graph domain. These features can then be used in a neural network-based output model to justify the graph model mismatch and eventually update the desired low-rank matrix.

## 1.2 Outline of the dissertation

In Chapter 2, we study the LRMC problem and its practical significance. We present its wide range of applications and the state-of-the-art LRMC techniques. We also discuss the intrinsic properties required for the matrix recovery and present the recovery performance guarantee of the LRMC techniques. In Chapter 3, we propose the LRM-CG algorithm to reconstruct the Euclidean distance matrix of IoT sensor nodes by exploiting the low-rank structure and Riemannian manifold structure of PSD matrices. From the convergence analysis, we present the extended Wolfe's conditions under which the recovery performance of LRM-CG is guaranteed. In Chapter 4, we propose an extended LRM-CG to solve the outlier problem and also examine the recovery performance of both LRM-CG and its extended version. In Chapter 5, we propose a GNN-based LRMC algorithm to reconstruct the rating matrix using its underlying graph structure. Chapter 6 is the conclusion and the future research.

Parts of the material in Chapter 2 appear in [22]. Parts of Chapter 3 and 4 appear in [23]. Parts of Chapter 5 appear in [24].

# Chapter 2

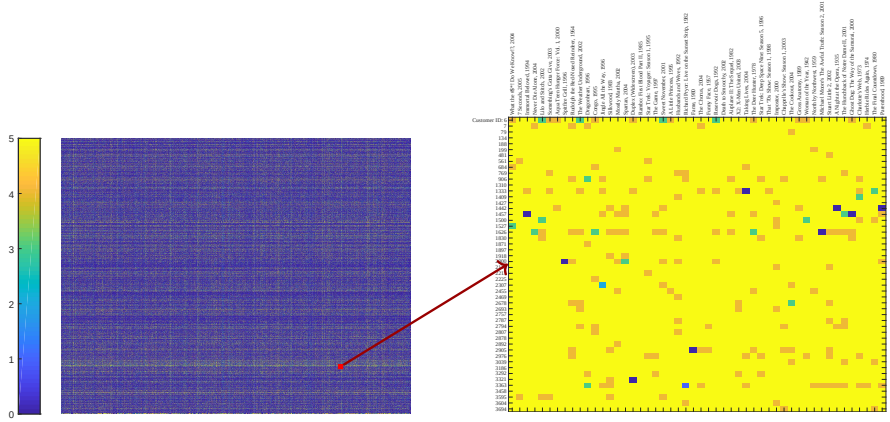
# Low-Rank Matrix Completion

## 2.1 LRMC Applications

In recent years, LRMC has received much attentions as a mean to recover the matrix accurately from small number of observed entries as long as the rank of a matrix is sufficiently small. Notable LRMC applications are as follows.

### 2.1.1 Recommendation system

In 2006, the online DVD rental company Netflix announced a contest to improve the quality of the company's movie recommendation system. The company released a training set consisting of ratings of more than 17,000 movies by more than 2.5 million users. The number of known entries is only about $1\%$, each entry an integer from $1$ to $5$ [1]. The training data can be represented in a large dimensional matrix in which the row and columns are indexed by user IDs and movie names, respectively. The primary goal of the recommendation system is to estimate the users' interests on products using the sparsely sampled rating matrix. Since many users sharing the same interests in key factors (e.g., the type, the price, and the appearance of the movie) often have the same ratings on the movies. Hence, the ratings of those users might form a low-rank column space, resulting in the low-rank model of the rating matrix (see Fig. 2.1).

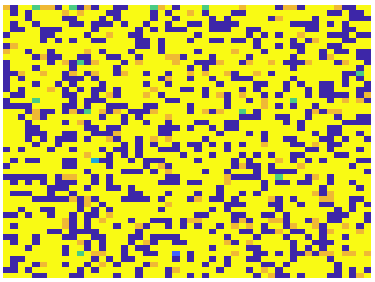



(a)

Netflix rating matrix with each entry an integer from 1 to 5 and zero for unknown

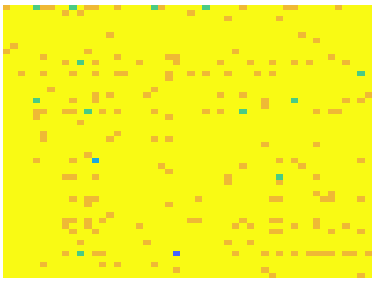
(b)

Submatrix $\mathbf{M}$ of size $50 \times 50$



(c)

Observed matrix $\mathbf{M}_o$ (70% of known entries of $\mathbf{M}$)



(d)

Reconstructed matrix $\widehat{\mathbf{M}}$ via LRMC using $\mathbf{M}_o$

Figure 2.1: Recommendation system application of LRMC. Entries of $\widehat{\mathbf{M}}$ are then simply rounded to integers, achieving 97.2% accuracy.

### 2.1.2 Phase retrieval

Phase retrieval is known as the problem to recover a signal (not necessarily sparse) from the magnitude of its observation. It is an important problem in X-ray crystallography and quantum mechanics since only the magnitude of the Fourier transform is measured in these applications [7]. Consider the unknown time-domain signal $\mathbf{m} = [m_0 \ \cdots \ m_{n-1}]$ and suppose its measured magnitude of the Fourier transform is given by

$$|z_\omega| = \frac{1}{\sqrt{n}} \left| \sum_{t=0}^{n-1} m_t e^{-j2\pi\omega t/n} \right|, \ \omega \in \Omega, \tag{2.1}$$

where $\Omega$ is the set of sampled frequencies. If we denote

$$\mathbf{f}_\omega = \frac{1}{\sqrt{n}} [1 \ e^{-j2\pi\omega/n} \ \cdots \ e^{-j2\pi\omega(n-1)/n}]^H \tag{2.2}$$

and $\mathbf{M} = \mathbf{m}\mathbf{m}^H$, then (2.1) can be expressed as

$$|z_\omega|^2 = |\langle \mathbf{f}_\omega, \mathbf{m} \rangle|^2 = \text{tr}(\mathbf{f}_\omega^H \mathbf{m}\mathbf{m}^H \mathbf{f}_\omega) = \text{tr}(\mathbf{m}\mathbf{m}^H \mathbf{f}_\omega \mathbf{f}_\omega^H) = \langle \mathbf{M}, \mathbf{F}_\omega \rangle, \tag{2.3}$$

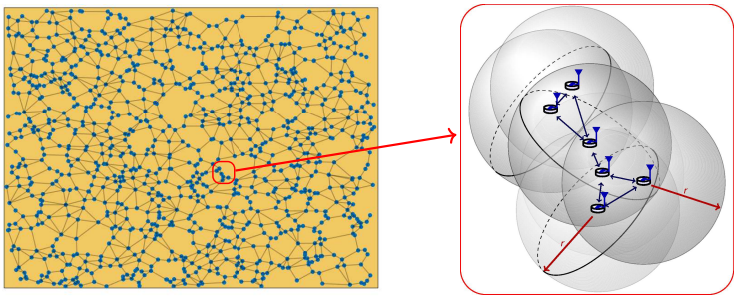
where $\mathbf{F}_w = \mathbf{f}_w \mathbf{f}_w^H$ is the rank-1 matrix of the waveform $\mathbf{f}_\omega$. Using (2.3), we can reformulate the phase retrieval problem as the problem to reconstruct the rank-1 matrix $\mathbf{M}$ in the positive semi-definite (PSD) cone [7]:

$$\begin{aligned} \min_{\mathbf{X}} \quad & \text{rank}(\mathbf{X}) \\ \text{subject to} \quad & \langle \mathbf{M}, \mathbf{F}_\omega \rangle = |z_\omega|^2, \ \omega \in \Omega \\ & \mathbf{X} \succeq 0. \end{aligned} \tag{2.4}$$

The desired signal $\mathbf{m}$ can then be computed via the eigenvalue decomposition of $\mathbf{M}$.

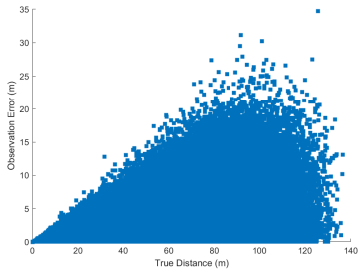
### 2.1.3 Localization in IoT networks

In big data era, internet of things (IoT) has a wide range of applications including healthcare, automatic metering, environmental monitoring (temperature, pressure, moisture), surveillance, to name just a few [25, 26, 2]. In most of IoT applications, the



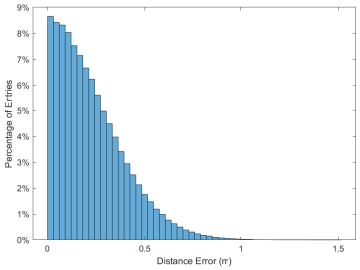
(a)

Partially observed distances of sensor nodes due to limitation of radio communication range $r$




(b)

RSSI-based observation error of 1000 sensor nodes in an 100m× 100m area




(c)

Reconstruction error

Figure 2.2: Localization via LRMC. The Euclidean distance matrix can be recovered with 92% of distance error below 0.5m using 30% of observed distances.

location information of sensor nodes is often needed to make a proper action, such as fire alarm, energy transfer, emergency request, on the data center. In network localization (a.k.a. cooperative localization), each sensor node measures the pairwise distances with its adjacent nodes and then sends it to the data center. Then the data center constructs a map of sensor nodes using the collected distance information [27]. Due to various reasons, such as the power outage of a sensor node or the limitation of radio communication range (see Fig. 2.2), only small number of distance information is available at the data center. Also, in the vehicular networks, it is not easy to measure the distance of all adjacent vehicles when a vehicle is located at the dead zone. An example of the observed Euclidean distance matrix is

$$\mathbf{M}_o = \begin{bmatrix} 0 & d_{12}^2 & d_{13}^2 & ? & ? \\ d_{21}^2 & 0 & ? & ? & ? \\ d_{31}^2 & ? & 0 & d_{34}^2 & d_{35}^2 \\ ? & ? & d_{43}^2 & 0 & d_{45}^2 \\ ? & ? & d_{53}^2 & d_{54}^2 & 0 \end{bmatrix},$$

where $d_{ij}$ is the pairwise distance between two sensor nodes $i$ and $j$. Since the rank of Euclidean distance matrix $\mathbf{M}$ is at most $k$+2 in the $k$-dimensional Euclidean space ($k = 2$ or $k = 3$) [3, 23], the problem to reconstruct $\mathbf{M}$ can be well-modeled as the LRMC problem.

### 2.1.4 Image compression and restoration

Image compression and restoration is a well-known problem to recover a "true" image from an observed image that has been corrupted by some noise process (e.g., dirt or scribble). One simple solution is to replace the contaminated pixels with the interpolated version of adjacent pixels. A better way is to exploit intrinsic domination of a few singular values in an image. In fact, one can readily approximate an image to the low-rank matrix without perceptible loss of quality. By using clean (uncontaminated) pixels as observed entries, an original image can be recovered via the low-rank matrix

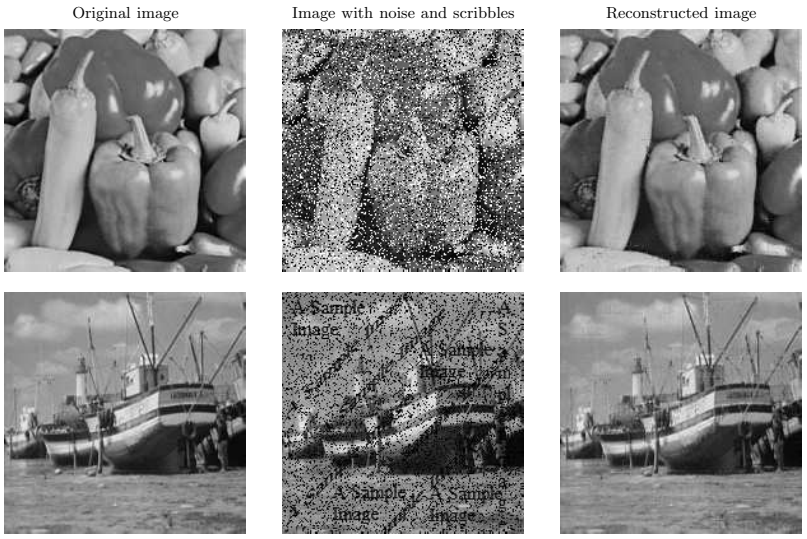



Figure 2.3: Image reconstruction via LRMC. Recovered images achieve peak SNR $\geq$ 32dB and structural similarity index (SSIM) at least 0.95.

completion.

### 2.1.5 Massive multiple-input multiple-output (MIMO)

By exploiting hundreds of antennas at the basestation (BS), massive MIMO can offer a large gain in capacity. In order to optimize the performance gain of the massive MIMO systems, the channel state information at the transmitter (CSIT) is required [28]. One way to acquire the CSIT is to let each user directly feed back its own pilot observation to BS for the joint CSIT estimation of all users [29]. In this setup, the MIMO channel matrix $\mathbf{H}$ can be reconstructed in two steps: 1) finding the pilot matrix $\mathbf{Y}$ using the least squares (LS) estimation or linear minimum mean square error (LMMSE) estimation and 2) reconstructing $\mathbf{H}$ using the model $\mathbf{Y} = \mathbf{H}\mathbf{\Phi}$ where each column of $\mathbf{\Phi}$ is the pilot signal from one antenna at BS [30, 31]. Since the number of resolvable paths $P$ is limited in most cases, one can readily assume that rank$(\mathbf{H}) \leq P$ [29]. In the massive MIMO systems, $P$ is often much smaller than the dimension of $\mathbf{H}$ due to the limited number of clusters around BS. Thus, the problem to recover $\mathbf{H}$ at BS can be solved via the rank minimization problem subject to the linear constraint $\mathbf{Y} = \mathbf{H}\mathbf{\Phi}$ [31].

### 2.1.6 Millimeter wave (mmWave) communication

In mmWave-based wireless system, training beamforming finding the beamformer-combiner pair of the highest channel gain is required to compensate the path loss of the mmWave frequency bands [32]. One way to estimate the mmWave channel is based on its sparse scattering nature. Specifically, let $\mathbf{B}$ and $\mathbf{C}$ be full-rank matrices constructed by all vectors in the pre-determined beamforming/combining codebooks $\mathcal{B}$ and $\mathcal{C}$, respectively. The observation matrix $\mathbf{Y}_o$ of the combined signal at the receiver can be expressed as $\mathbf{Y}_o = \mathbf{B}^H\mathbf{H}\mathbf{C} + \mathbf{N}$ where $\mathbf{H}$ is the channel matrix and $\mathbf{N}$ is the matrix of noise [33]. Since mmWave channels spread in the form of clusters of paths over the angular domains (e.g., angle of arrival (AoA) and angle of departure (AoD)) in many practical cases, it can be shown that the rank of $\mathbf{Y} = \mathbf{B}^H\mathbf{H}\mathbf{C}$ is less than

the sparsity level of the channel [34, 33]. The problem to recover $\mathbf{H}$ is now equivalent to the problem to reconstruct the low-rank matrix $\mathbf{Y}$ from its noisy version $\mathbf{Y}_o$ since $\mathbf{H} = (\mathbf{B}^H)^{-1}\mathbf{Y}\mathbf{C}^{-1}$.

Other than these, there are a bewildering variety of applications of LRMC in wireless communication, such as millimeter wave (mmWave) channel estimation [32, 33], topological interference management (TIM) [35, 36, 37, 38] and mobile edge caching in fog radio access networks (Fog-RAN) [39, 40].

## 2.2 Intrinsic Properties of LRMC

There are two key properties characterizing the LRMC problem: 1) *sparsity* of the observed entries and 2) *incoherence* of the matrix. Sparsity indicates that an accurate recovery of the undersampled matrix is possible even when the number of observed entries is very small. Incoherence indicates that nonzero entries of the matrix should be spread out widely for the efficient recovery of a low-rank matrix. In this section, we go over these issues in detail.

### 2.2.1 Sparsity of Observed Entries

Sparsity expresses an idea that when a matrix has a low rank property, then it can be recovered using only a small number of observed entries. Natural question arising from this is how many elements do we need to observe for the accurate recovery of the matrix. To answer this question, we need a notion of a degree of freedom (DOF). The DOF of a matrix is defined as the number of freely chosen variables in the matrix. For example, one can easily see that the DOF of the rank-one matrix in (1.1) is 3 since one entry can be determined after observing three. As an another example, we consider the

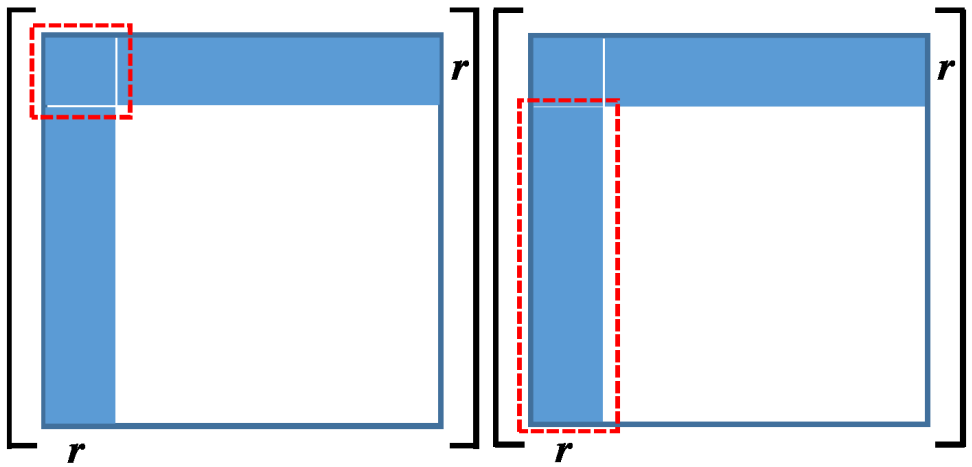



Figure 2.4: LRMC with colored entries being observed. The dotted boxes are used to compute: (a) linear coefficients and (b) unknown entries.

following rank-one matrix

$$\mathbf{M} = \begin{bmatrix} 1 & 3 & 5 & 7 \\ 2 & 6 & 10 & 14 \\ 3 & 9 & 15 & 21 \\ 4 & 12 & 20 & 28 \end{bmatrix}. \tag{2.5}$$

Suppose we observe all entries of one column and one row. Then it is true that the rest can be determined by a simple linear relationship between these. Specifically, if we observe the first row and the first column, then the first and the second columns differ by the factor of three so that as long as we know one entry in the second column, the rest would be recovered. Thus, the DOF of $\mathbf{M}$ is $4 + 4 - 1 = 7$. Following lemma generalizes our observations.

**Lemma 1** *The DOF of a square $n \times n$ matrix with rank $r$ is $2nr - r^2$. Also, the DOF of $n_1 \times n_2$-matrix is $(n_1 + n_2)r - r^2$.*

**Proof:** Since the rank of a matrix is $r$, we can freely choose values for all entries of the $r$ columns, resulting in $nr$ degrees of freedom for the first $r$ column. Once $r$ independent columns, say $\mathbf{m}_1, \cdots \mathbf{m}_r$, are constructed, then each of the rest $n - r$ columns can be expressed as a linear combinations of the first $r$ columns (e.g., $\mathbf{m}_{r+1} = \alpha_1\mathbf{m}_1 + \cdots + \alpha_r\mathbf{m}_r$) so that $r$ linear coefficients $(\alpha_1, \cdots \alpha_r)$ can be freely chosen in these columns. Adding $nr$ and $(n - r)r$, we have the desired result. Generalization to $n_1 \times n_2$ matrix is straightforward. □

This lemma says that if $n$ is large and $r$ is small enough (e.g., $r = O(1)$), essential information in a matrix is just in the order of $n$, DOF$= O(n)$, which is clearly much smaller than the total number of entries of the matrix. Appealingly, the DOF is the minimum requirement of observed entries to reconstruct a matrix. If the number of observed entries is less than the DOF (i.e., $m < 2nr - r^2$), the matrix is unrecoverable. In Fig. 2.4, we illustrate how to recover a low-rank matrix when the number of observed entries equals the DOF. In this figure, we assume that blue colored entries are

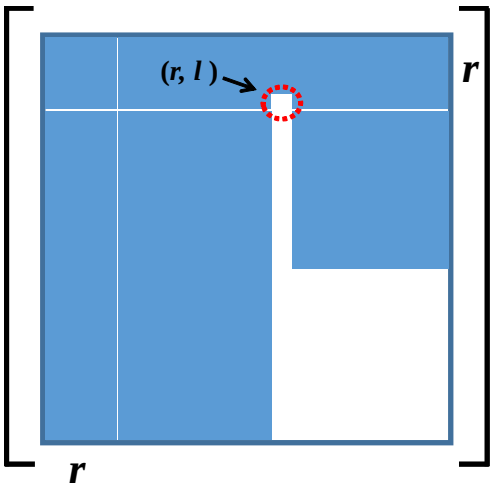



Figure 2.5: An illustration of the worst case of LRMC.

observed.[1] In essence, unknown entries of the matrix are found in two-step process. First, we identify the linear relationship between the first $r$ columns and the rest. For example, the $(r+1)$-th column can be expressed as a linear combination of the first $r$ columns. That is,

$$\mathbf{m}_{r+1} = \alpha_1 \mathbf{m}_1 + \cdots + \alpha_r \mathbf{m}_r. \tag{2.6}$$

Since the first $r$ entries of $\mathbf{m}_1, \cdots \mathbf{m}_{r+1}$ are observed (see Fig. 2.4(a)), we have $r$ unknowns $(\alpha_1, \cdots, \alpha_r)$ and $r$ equations so that we can identify the linear coefficients $\alpha_1, \cdots \alpha_r$ with the computational cost $\mathcal{O}(r^3)$ of an $r \times r$ matrix inversion. Once these coefficients are identified, we can reconstruct the unknown entries $m_{r+1,r+1} \cdots m_{r+1,n}$ of $\mathbf{m}_{r+1}$ using the linear combination in (2.6) (see Fig. 2.4(b)). By repeating this step for the rest of columns, we can reconstruct all unknown entries with $\mathcal{O}(rn^2)$ computational complexity[2].

Now, an astute reader might notice that this strategy will not work if one entry of the column (or row) is unobserved. As illustrated in Fig. 2.5, if only one entry in the $r$-th row, say $(r, l)$-th entry, is unobserved, then one cannot recover the $l$-th column simply because the matrix in Fig. 2.5 cannot be converted to the matrix form in Fig. 2.4(b). Obviously, the measurement size being equal to the DOF is a necessary condition for the accurate recovery of the rank-$r$ matrix. This seems like a depressing news. However, DOF is in any case important since it is a fundamental limit (lower bound) of the number of observed entries to ensure the exact recovery of the matrix. Recent results show that the DOF is not much different from the number of measurements ensuring the recovery of the matrix [5, 41].[3]

[1] Since we observe the first $r$ rows and columns, we have $2nr - r^2$ observations in total.

[2] For each unknown entry, it needs $r$ multiplication and $r-1$ addition operations. Since the number of unknown entries is $(n-r)^2$, the computational cost is $(2r-1)(n-r)^2$. Recall that $\mathcal{O}(r^3)$ is the cost of computing $(\alpha_1, \cdots, \alpha_r)$ in (2.6). Therefore, the total cost is $\mathcal{O}(r^3 + (2r-1)(n-r)^2) = \mathcal{O}(rn^2)$.

[3] In [41], it has been shown that the required number of entries to recover the matrix using the nuclear-norm minimization is in the order of $n^{1.2}$ when the rank is $O(1)$.

### 2.2.2 Coherence

If nontrivial entries of a low-rank matrix are concentrated in a certain region, we generally need a large number of observations to recover the matrix. In contrast, if the entries are spread out widely, then the matrix can be recovered with a relatively small number of entries. For example, consider the following two rank-one matrices in $\mathbb{R}^{n\times n}$

$$\mathbf{M}_1 = \begin{bmatrix} 1 & 1 & 0 & \cdots & 0 \\ 1 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix}, \mathbf{M}_2 = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & 1 & 1 & \cdots & 1 \\ 1 & 1 & 1 & \cdots & 1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & 1 & \cdots & 1 \end{bmatrix}.$$

The matrix $\mathbf{M}_1$ has only four nonzero entries at the top-left corner. Suppose $n$ is large, say $n = 1000$, and all entries but the four elements in the top-left corner are observed (99.99% of entries are known). In this case, even though the rank of a matrix is just one, there is no way to recover this matrix since the information bearing entries is missing. This tells us that although the rank of a matrix is very small, one might not recover it if nonzero entries of the matrix are concentrated in a certain area.

In contrast to the matrix $\mathbf{M}_1$, one can accurately recover the matrix $\mathbf{M}_2$ with the minimum requirement of known entries, only $2n - 1$ (= DOF) known entries. In other words, one row and one column are enough to recover $\mathbf{M}_2$). One can deduce from this example that the spread of observed entries is important for the identification of unknown entries.

In order to quantify this, we need to measure the concentration of a matrix, checking the concentration in both row and column directions. This can be done by checking the concentration in the left and right singular vectors. Recall that the SVD of a matrix is

$$\mathbf{M} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T = \sum_{i=1}^{r} \sigma_i \mathbf{u}_i \mathbf{v}_i^T, \tag{2.7}$$

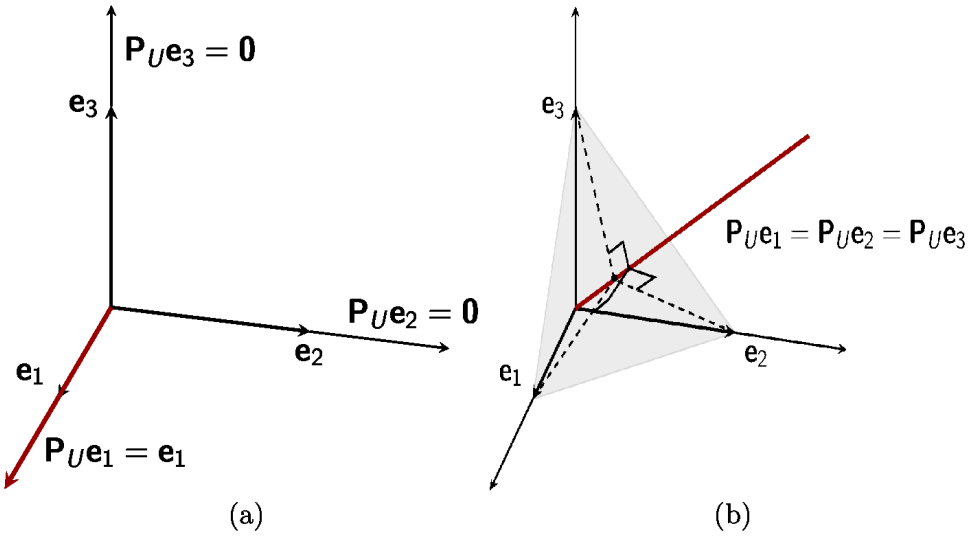



Figure 2.6: Coherence of matrices in (2.10) and (2.11): (a) maximum and (b) minimum.

where $\mathbf{U} = [\mathbf{u}_1 \ \cdots \ \mathbf{u}_r]$ and $\mathbf{V} = [\mathbf{v}_1 \ \cdots \ \mathbf{v}_r]$ are the matrices constructed by the left and right singular vectors, respectively, and $\mathbf{\Sigma}$ is the diagonal matrix whose diagonal entries are $\sigma_i$. From (2.7), we see that the concentration on the vertical direction (concentration in the row) is determined by $\mathbf{u}_i$ and that on the horizontal direction (concentration in the column) is determined by $\mathbf{v}_i$. For example, if one of the standard basis vector $\mathbf{e}_i$, say $\mathbf{e}_1 = [1 \ 0 \cdots \ 0]^T$, lies on the space spanned by $\mathbf{u}_1, \cdots \mathbf{u}_r$ while others $(\mathbf{e}_2, \mathbf{e}_3, \cdots)$ are orthogonal to this space, then it is clear that nonzero entries of the matrix are only on the first row. In this case, clearly one cannot infer the entries of the first row from the sampling of the other row.

The coherence, a measure of concentration in a matrix, is formally defined as [41]

$$\mu(\mathbf{U}) = \frac{n}{r} \max_{1 \le i \le n} \|P_{\mathbf{U}} \mathbf{e}_i\|^2, \tag{2.8}$$

where $\mathbf{e}_i$ is standard basis and $P_{\mathbf{U}}$ is the projection onto the range space of $\mathbf{U}$. Since the columns of $\mathbf{U} = [\mathbf{u}_1 \ \cdots \ \mathbf{u}_r]$ are orthonormal, we have

$$P_{\mathbf{U}} = \mathbf{U}\mathbf{U}^\dagger = \mathbf{U}(\mathbf{U}^T\mathbf{U})^{-1}\mathbf{U}^T = \mathbf{U}\mathbf{U}^T.$$

Note that both $\mu(\mathbf{U})$ and $\mu(\mathbf{V})$ are needed to check the concentration on the vertical and horizontal directions.

**Lemma 2** *(Maximum and minimum value of $\mu(\mathbf{U})$) $\mu(\mathbf{U})$ satisfies*

$$1 \le \mu(\mathbf{U}) \le \frac{n}{r}. \tag{2.9}$$

**Proof:** The upper bound is quite clear since the $\ell_2$-norm of the projection is not greater than the original vector ($\|P_{\mathbf{U}}\mathbf{e}_i\|_2^2 \leq \|\mathbf{e}_i\|_2^2$). The lower bound is because

$$\begin{aligned}
\max_i \|P_{\mathbf{U}}\mathbf{e}_i\|_2^2 &\geq \frac{1}{n}\sum_{i=1}^{n} \|P_{\mathbf{U}}\mathbf{e}_i\|_2^2 \\
&= \frac{1}{n}\sum_{i=1}^{n} \mathbf{e}_i^T P_{\mathbf{U}}\mathbf{e}_i \\
&= \frac{1}{n}\sum_{i=1}^{n} \mathbf{e}_i^T \mathbf{U}\mathbf{U}^T\mathbf{e}_i \\
&= \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{r} |u_{ij}|^2 \\
&= \frac{r}{n},
\end{aligned}$$

where the first equality is due to the idempotency of $P_{\mathbf{U}}$ (i.e., $P_{\mathbf{U}}^T P_{\mathbf{U}} = P_{\mathbf{U}}$) and the last equality is because $\sum_{i=1}^{n} |u_{ij}|^2 = 1$. □

Coherence is maximized when the nonzero entries of a matrix are concentrated in a row (or column). For example, consider the matrix whose nonzero entries are concentrated on the first row

$$\mathbf{M} = \begin{bmatrix} 3 & 2 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{2.10}$$

Note that the SVD of $\mathbf{M}$ is

$$\mathbf{M} = \sigma_1 \mathbf{u}_1 \mathbf{v}_1^T = 3.8417 \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} [0.8018\ 0.5345\ 0.2673].$$

Then, $\mathbf{U} = [1\ 0\ 0]^T$, and thus $\|P_{\mathbf{U}}\mathbf{e}_1\|_2 = 1$ and $\|P_{\mathbf{U}}\mathbf{e}_2\|_2 = \|P_{\mathbf{U}}\mathbf{e}_3\|_2 = 0$. As shown in Fig. 2.6(a), the standard basis $\mathbf{e}_1$ lies on the space spanned by $\mathbf{U}$ while others are orthogonal to this space so that the maximum coherence is achieved ($\max_i \|P_{\mathbf{U}}\mathbf{e}_i\|_2^2 = 1$ and $\mu(\mathbf{U}) = 3$).

In contrast, coherence is minimized when the nonzero entries of a matrix are spread out widely. For instance, we consider the matrix

$$\mathbf{M} = \begin{bmatrix} 2 & 1 & 0 \\ 2 & 1 & 0 \\ 2 & 1 & 0 \end{bmatrix}. \tag{2.11}$$

In this case, the SVD of $\mathbf{M}$ is

$$\mathbf{M} = 3.8730 \begin{bmatrix} -0.5774 \\ -0.5774 \\ -0.5774 \end{bmatrix} [-0.8944 \ -0.4472 \ 0].$$

Then, we have

$$P_{\mathbf{U}} = \mathbf{U}\mathbf{U}^T = \frac{1}{3} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix},$$

and thus $\|P_{\mathbf{U}}\mathbf{e}_1\|_2^2 = \|P_{\mathbf{U}}\mathbf{e}_2\|_2 = \|P_{\mathbf{U}}\mathbf{e}_3\|_2 = \frac{1}{3}$. As illustrated in Fig. 2.6(b), $\|P_{\mathbf{U}}\mathbf{e}_i\|_2$ is the same for all standard basis vector $\mathbf{e}_i$, achieving lower bound in (2.9) and the minimum coherence ($\max_i \|P_{\mathbf{U}}\mathbf{e}_i\|_2^2 = \frac{1}{3}$ and $\mu(\mathbf{U}) = 1$). It can be shown that the number of measurements to recover the low-rank matrix is proportional to the coherence of the matrix [5, 6, 41].

## 2.3 Rank Minimization Problem

The most straightforward and intuitive way to model the LRMC problem is known as the rank minimization problem:

$$\begin{aligned} \min_{\mathbf{X}} \quad & \text{rank}(\mathbf{X}) \\ \text{subject to} \quad & x_{ij} = m_{ij}, \ (i,j) \in \Omega, \end{aligned} \tag{2.12}$$

where $m_{ij}$ is the entries of the desired low-rank matrix $\mathbf{M}$ and $\Omega$ is the index set of observed entries. For example, from the example in (1.1)), we have $\Omega = \{(1,1),(1,2),(2,1)\}$.

Alternatively, we can express the problem (2.12) based on the sampling operator $P_\Omega$ defined as

$$[P_\Omega(\mathbf{A})]_{ij} = \begin{cases} a_{ij} & \text{if } (i,j) \in \Omega \\ 0 & \text{otherwise,} \end{cases}$$

where $a_{ij}$ is the $(i,j)$-th entry of $\mathbf{A}$. Using $P_\Omega$, the problem (2.12) can be equivalently formulated as

$$\begin{aligned} \min_{\mathbf{X}} \quad & \text{rank}(\mathbf{X}) \\ \text{subject to} \quad & P_\Omega(\mathbf{X}) = P_\Omega(\mathbf{M}). \end{aligned} \tag{2.13}$$

A possible simple way to solve the rank minimization problem (2.13) is based on the combinatorial search. In this approach, we start with the assumption that $\text{rank}(\mathbf{M}) = 1$. Then, any two columns of $\mathbf{M}$ are linearly dependent and thus we have the system of equations $\mathbf{m}_i = \alpha_{i,j}\mathbf{m}_j$ for some $\alpha_{i,j} \in \mathbb{R}$. The solution of this equation system is also the solution of (2.13) since $\text{rank}(\mathbf{M}) = 1$ is the smallest rank matrix satisfying the constraints[4]. If the system has no solution for the rank-one assumption, we move to the next assumption of $\text{rank}(\mathbf{M}) = 2$. In this case, we solve the new system of equations $\mathbf{m}_i = \alpha_{i,j}\mathbf{m}_j + \alpha_{i,k}\mathbf{m}_k$. This procedure is repeated until the solution is found. Obviously, the combinatorial search strategy would not be feasible for most practical scenarios since it has an exponential complexity in the problem size [42]. For example, when $\mathbf{M}$ is an $n \times n$ matrix, it is not difficult to check that the number of the system expressions to be solved is $\mathcal{O}(n2^n)$.

As a cost-effective alternative, various low-rank matrix completion (LRMC) algorithms have been proposed over the years. Depending on the availability of the rank information, the LRMC algorithms can be classified into two main categories: 1) those without the rank information and 2) those exploiting the rank information. In the next sections, we provide an in depth discussion of two categories (see the outline of LRMC algorithms in Fig. 2.7).

[4]Since the observed entries of $\mathbf{M}$ are supposed to be nonzero, $\text{rank}(\mathbf{M}) > 0$.

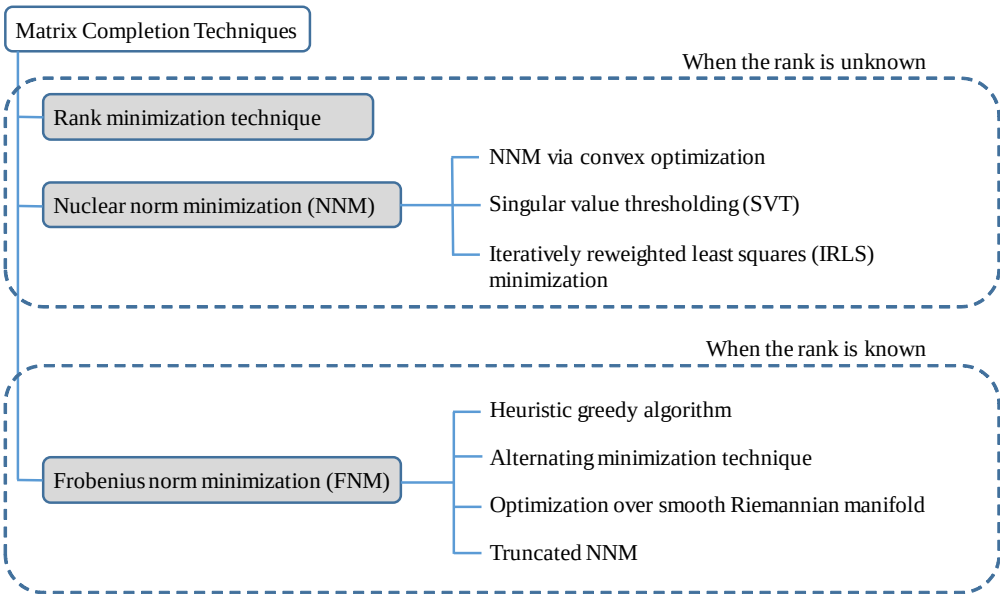



Figure 2.7: Outline of LRMC algorithms. Depending on the availability of the rank, we can naturally classify LRMC techniques to two main categories: the techniques with and without the rank information

## 2.4 LRMC Algorithms Without the Rank Information

In this section, we explain the LRMC algorithms that do not require the rank information of the original low-rank matrix.

### 2.4.1 Nuclear Norm Minimization (NNM)

NNM is known as the most popular convex relaxation of the rank minimization problem. Since the rank minimization problem (2.13) is NP-hard [4], it is computationally intractable when the dimension of a matrix is large. To overcome this, one common trick is to replace the non-convex objective function with its convex surrogate, converting the combinatorial search problem into a convex optimization problem. There are two clear advantages in solving the convex optimization problem: 1) a local optimum solution is globally optimal and 2) there are many efficient polynomial-time convex optimization solvers (e.g., interior point method [43] and semi-definite programming (SDP) solver).

In this approach, the nuclear norm $\|\mathbf{X}\|_*$, the sum of the singular values of $\mathbf{X}$, has been widely used as a convex surrogate of rank$(\mathbf{X})$ [5]:

$$\begin{aligned} \min_{\mathbf{X}} \quad & \|\mathbf{X}\|_* \\ \text{subject to} \quad & P_\Omega(\mathbf{X}) = P_\Omega(\mathbf{M}) \end{aligned} \tag{2.14}$$

In fact, it has been shown that the nuclear norm is the convex envelope (the "best" convex approximation) of the rank function on the set $\{\mathbf{X} \in \mathbb{R}^{n_1 \times n_2} : \|\mathbf{X}\| \leq 1\}$ [4].[5] Note that the relaxation from the rank function to the nuclear norm is conceptually analogous to the relaxation from $\ell_0$-norm to $\ell_1$-norm in compressed sensing (CS) [44, 45, 46].

[5]For any function $f : \mathcal{C} \to \mathbb{R}$, where $\mathcal{C}$ is a convex set, the convex envelope of $f$ is the largest convex function $g$ such that $f(x) \geq g(x)$ for all $x \in \mathcal{C}$. Note that the convex envelope of rank$(\mathbf{X})$ on the set $\{\mathbf{X} \in \mathbb{R}^{n_1 \times n_2} : \|\mathbf{X}\| \leq 1\}$ is the nuclear norm $\|\mathbf{X}\|_*$ [4].

In order to solve (2.14), we first recast it as a semidefinite program (SDP). We recall that the standard form of an SDP is

$$\begin{aligned} \min_{\mathbf{Y}} \quad & \langle \mathbf{C}, \mathbf{Y} \rangle \\ \text{subject to} \quad & \langle \mathbf{A}_k, \mathbf{Y} \rangle \leq b_k, \;\; k = 1, \cdots, l \\ & \mathbf{Y} \succeq 0 \end{aligned} \tag{2.15}$$

where $\mathbf{C}$ is a given matrix, and $\{\mathbf{A}_k\}_{k=1}^l$ and $\{b_k\}_{k=1}^l$ are given sequences of matrices and constants, respectively. There are two main steps to reformulate the NNM problem in (2.14) as the standard SDP form in (2.15). First, since $\min_{\mathbf{X}} \|\mathbf{X}\|_* = \min_{\mathbf{X}}(\min_{t:\|\mathbf{X}\|_* \leq t} t) = \min_{(\mathbf{X},t):\|\mathbf{X}\|_* \leq t} t$, we convert the NNM problem in (2.14) into the epigraph form as

$$\begin{aligned} \min_{\mathbf{X},t} \quad & t \\ \text{subject to} \quad & \|\mathbf{X}\|_* \leq t, \\ & P_\Omega(\mathbf{X}) = P_\Omega(\mathbf{M}). \end{aligned} \tag{2.16}$$

Next, we transform the constraints in (2.16) to match with the standard form in (2.15). Note that $\|\mathbf{X}\|_* \leq t$ if and only if there are symmetric matrices $\mathbf{W}_1 \in \mathbb{R}^{n_1 \times n_1}$ and $\mathbf{W}_2 \in \mathbb{R}^{n_2 \times n_2}$ such that [4, Lemma 2]

$$\text{tr}(\mathbf{W}_1) + \text{tr}(\mathbf{W}_2) \leq 2t \;\text{ and }\; \begin{bmatrix} \mathbf{W}_1 & \mathbf{X} \\ \mathbf{X}^T & \mathbf{W}_2 \end{bmatrix} \succeq 0. \tag{2.17}$$

Then, by denoting $\mathbf{Y} = \begin{bmatrix} \mathbf{W}_1 & \mathbf{X} \\ \mathbf{X}^T & \mathbf{W}_2 \end{bmatrix} \in \mathbb{R}^{(n_1+n_2)\times(n_1+n_2)}$ and $\widetilde{\mathbf{M}} = \begin{bmatrix} \mathbf{0} & \mathbf{M} \\ \mathbf{M}^T & \mathbf{0} \end{bmatrix}$ where $\mathbf{0}_{s\times t}$ is the $(s \times t)$-dimensional zero matrix, the problem in (2.16) can be reformulated as

$$\begin{aligned} \min_{\mathbf{Y},t} \quad & 2t \\ \text{subject to} \quad & \text{tr}(\mathbf{Y}) \leq 2t, \\ & \mathbf{Y} \succeq 0, \\ & P_{\widetilde{\Omega}}(\mathbf{Y}) = P_{\widetilde{\Omega}}(\widetilde{\mathbf{M}}), \end{aligned} \tag{2.18}$$

where $P_{\widetilde{\Omega}}(\mathbf{Y}) = \begin{bmatrix} \mathbf{0} & P_{\Omega}(\mathbf{X}) \\ (P_{\Omega}(\mathbf{X}))^T & \mathbf{0} \end{bmatrix}$ is the extended sampling operator. We now consider the equality constraint ($P_{\widetilde{\Omega}}(\mathbf{Y}) = P_{\widetilde{\Omega}}(\widetilde{\mathbf{M}})$) in (2.18). One can easily see that this condition is equivalent to

$$\langle \mathbf{Y}, \mathbf{e}_i \mathbf{e}_{j+n_1}^T \rangle = \langle \widetilde{\mathbf{M}}, \mathbf{e}_i \mathbf{e}_{j+n_1}^T \rangle, \quad (i,j) \in \Omega, \tag{2.19}$$

where $\{\mathbf{e}_1, \cdots, \mathbf{e}_{n_1+n_2}\}$ be the standard ordered basis of $\mathbb{R}^{n_1+n_2}$. Let $\mathbf{A}_k = \mathbf{e}_i \mathbf{e}_{j+n_1}^T$ and $\langle \widetilde{\mathbf{M}}, \mathbf{e}_i \mathbf{e}_{j+n_1}^T \rangle = b_k$ for each of $(i,j) \in \Omega$. Then,

$$\langle \mathbf{Y}, \mathbf{A}_k \rangle = b_k, \quad k = 1, \cdots, |\Omega|, \tag{2.20}$$

and thus (2.18) can be reformulated as

$$\begin{aligned} \min_{\mathbf{Y},t} \quad & 2t \\ \text{subject to} \quad & \text{tr}(\mathbf{Y}) \leq 2t \\ & \mathbf{Y} \succeq 0 \\ & \langle \mathbf{Y}, \mathbf{A}_k \rangle = b_k, \ \ k = 1, \cdots, |\Omega|. \end{aligned} \tag{2.21}$$

For example, we consider the case where the desired matrix $\mathbf{M}$ is given by $\mathbf{M} = \begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix}$ and the index set of observed entries is $\Omega = \{(2,1),(2,2)\}$. In this case,

$$\mathbf{A}_1 = \mathbf{e}_2 \mathbf{e}_3^T, \ \mathbf{A}_2 = \mathbf{e}_2 \mathbf{e}_4^T, \ b_1 = 2, \text{ and } b_2 = 4. \tag{2.22}$$

In the final step, one can express (2.21) in a concise form as

$$\begin{aligned} \min_{\mathbf{Y}} \quad & \text{tr}(\mathbf{Y}) \\ \text{subject to} \quad & \langle \mathbf{Y}, \mathbf{A}_k \rangle = b_k, \ \ k = 1, \cdots, |\Omega| \\ & \mathbf{Y} \succeq 0. \end{aligned} \tag{2.23}$$

The problem (2.23) can be solved by the off-the-shelf SDP solvers such as SDPT3 [47] and SeDuMi [48] using interior-point methods [49, 50, 51, 52, 53, 54]. It has been

shown that the computational complexity of SDP techniques is $\mathcal{O}(n^3)$ where $n = \max(n_1, n_2)$ [53]. Also, it has been shown that under suitable conditions, the output $\widehat{\mathbf{M}}$ of SDP satisfies $\|\widehat{\mathbf{M}} - \mathbf{M}\|_F \leq \epsilon$ in at most $\mathcal{O}(n^\omega \log(\frac{1}{\epsilon}))$ iterations where $\omega$ is a positive constant [54]. Alternatively, one can reconstruct $\mathbf{M}$ by solving the equivalent nonconvex quadratic optimization form of the NNM problem [55]. Note that this approach has computational benefit since the number of primal variables of NNM is reduced from $n_1 n_2$ to $r(n_1 + n_2)$ ($r \leq \min(n_1, n_2)$). Interested readers may refer to [55] for more details.

### 2.4.2 Singular Value Thresholding (SVT)

While the solution of the NNM problem in (2.14) can be obtained by solving (2.23), this procedure is computationally burdensome when the size of the matrix is large. As an effort to mitigate the computational burden, the singular value thresholding (SVT) algorithm has been proposed [10]. In essence, the key idea of this approach is to add the regularization term into the objective function of the NNM problem:

$$\begin{aligned} \min_{\mathbf{X}} \quad & \tau\|\mathbf{X}\|_* + \frac{1}{2}\|\mathbf{X}\|_F^2 \\ \text{subject to} \quad & P_\Omega(\mathbf{X}) = P_\Omega(\mathbf{M}), \end{aligned} \tag{2.24}$$

where $\tau$ is the regularization parameter. In [10, Theorem 3.1], it has been shown that the solution to the problem (2.24) converges to the solution of the NNM problem as $\tau \to \infty$.[6]

Let $\mathcal{L}(\mathbf{X}, \mathbf{Y})$ be the Lagrangian function associated with (2.24), i.e.,

$$\mathcal{L}(\mathbf{X}, \mathbf{Y}) = \tau\|\mathbf{X}\|_* + \frac{1}{2}\|\mathbf{X}\|_F^2 + \langle \mathbf{Y}, P_\Omega(\mathbf{M}) - P_\Omega(\mathbf{X})\rangle \tag{2.25}$$

where $\mathbf{Y}$ is the dual variable. Also, let $\widehat{\mathbf{X}}$ and $\widehat{\mathbf{Y}}$ be the primal and dual optimal solu-

[6] In practice, a large value of $\tau$ has been suggested (e.g., $\tau = 5n$ for an $n \times n$ low rank matrix) for the fast convergence of SVT. For example, when $\tau = 5000$, it requires 177 iterations to reconstruct a $1000 \times 1000$ matrix of rank 10 [10].

Table 2.1: The SVT Algorithm

| | |
|---|---|
| **Input** | observed entries $P_\Omega(\mathbf{M})$, |
| | a sequence of positive step sizes $\{\delta_k\}_{k\geq 1}$, |
| | a regularization parameter $\tau > 0$, |
| | and a stopping criterion $T$ |
| **Initialize** | iteration counter $k = 0$ |
| | and $\mathbf{Y}_0 = \mathbf{0}_{n_1 \times n_2}$, |
| **While** | $T =$ false **do** |
| | $k = k + 1$ |
| | $[\mathbf{U}_{k-1}, \mathbf{\Sigma}_{k-1}, \mathbf{V}_{k-1}] = \text{svd}(\mathbf{Y}_{k-1})$ |
| | $\mathbf{X}_k = \mathbf{U}_{k-1}\,\text{diag}(\{(\sigma_i(\mathbf{\Sigma}_{k-1}) - \tau)_+\}_i)\mathbf{V}_{k-1}^T$ using (2.30) |
| | $\mathbf{Y}_k = \mathbf{Y}_{k-1} + \delta_k(P_\Omega(\mathbf{M}) - P_\Omega(\mathbf{X}_k))$ |
| **End** | |
| **Output** | $\mathbf{X}^k$ |

tions. Suppose that the strong duality holds true [43]. Then, we have

$$\max_{\mathbf{Y}} \min_{\mathbf{X}} \mathcal{L}(\mathbf{X}, \mathbf{Y}) = \mathcal{L}(\widehat{\mathbf{X}}, \widehat{\mathbf{Y}}) = \min_{\mathbf{X}} \max_{\mathbf{Y}} \mathcal{L}(\mathbf{X}, \mathbf{Y}). \tag{2.26}$$

The SVT algorithm finds $\widehat{\mathbf{X}}$ and $\widehat{\mathbf{Y}}$ in an iterative fashion with the updated expressions given as

$$\mathbf{X}_k = \arg\min_{\mathbf{X}} \mathcal{L}(\mathbf{X}, \mathbf{Y}_{k-1}), \tag{2.27a}$$

$$\mathbf{Y}_k = \mathbf{Y}_{k-1} + \delta_k \frac{\partial \mathcal{L}(\mathbf{X}_k, \mathbf{Y})}{\partial \mathbf{Y}}, \tag{2.27b}$$

where $\{\delta_k\}_{k\geq 1}$ is a sequence of positive step sizes. Note that (2.27a) can be expressed as

$$\begin{aligned}
\mathbf{X}_k &= \arg\min_{\mathbf{X}} \tau\|\mathbf{X}\|_* + \frac{1}{2}\|\mathbf{X}\|_F^2 - \langle \mathbf{Y}_{k-1}, P_\Omega(\mathbf{X})\rangle \\
&\overset{(a)}{=} \arg\min_{\mathbf{X}} \tau\|\mathbf{X}\|_* + \frac{1}{2}\|\mathbf{X}\|_F^2 - \langle P_\Omega(\mathbf{Y}_{k-1}), \mathbf{X}\rangle \\
&\overset{(b)}{=} \arg\min_{\mathbf{X}} \tau\|\mathbf{X}\|_* + \frac{1}{2}\|\mathbf{X}\|_F^2 - \langle \mathbf{Y}_{k-1}, \mathbf{X}\rangle \\
&= \arg\min_{\mathbf{X}} \tau\|\mathbf{X}\|_* + \frac{1}{2}\|\mathbf{X} - \mathbf{Y}_{k-1}\|_F^2,
\end{aligned} \tag{2.28}$$

where (a) is because $\langle P_\Omega(\mathbf{A}), \mathbf{B}\rangle = \langle \mathbf{A}, P_\Omega(\mathbf{B})\rangle$ and (b) is because $\mathbf{Y}_{k-1}$ vanishes outside of $\Omega$ (i.e., $P_\Omega(\mathbf{Y}_{k-1}) = \mathbf{Y}_{k-1}$) by (2.27b). Due to the inclusion of the nuclear norm, finding out the solution $\mathbf{X}_k$ of (2.28) seems to be difficult. However, thanks to the intriguing result of Cai et al., we can easily obtain the solution.

**Theorem 3 ([10, Theorem 2.1])** *Let* $\mathbf{Z}$ *be a matrix whose singular value decomposition (SVD) is* $\mathbf{Z} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$. *Define* $t_+ = \max\{t, 0\}$ *for* $t \in \mathbb{R}$. *Then,*

$$\mathcal{D}_\tau(\mathbf{Z}) = \arg\min_{\mathbf{X}} \tau\|\mathbf{X}\|_* + \frac{1}{2}\|\mathbf{X} - \mathbf{Z}\|_F^2, \tag{2.29}$$

*where* $\mathcal{D}_\tau$ *is the singular value thresholding operator defined as*

$$\mathcal{D}_\tau(\mathbf{Z}) = \mathbf{U}\,\mathrm{diag}(\{(\sigma_i(\mathbf{\Sigma}) - \tau)_+\}_i)\mathbf{V}^T. \tag{2.30}$$

By Theorem 3, the right-hand side of (2.28) is $\mathcal{D}_\tau(\mathbf{Y}_{k-1})$. Thus, the update equations for $\mathbf{X}_k$ and $\mathbf{Y}_k$ are given by

$$\mathbf{X}_k = \mathcal{D}_\tau(\mathbf{Y}_{k-1}), \tag{2.31a}$$

$$\mathbf{Y}_k = \mathbf{Y}_{k-1} + \delta_k(P_\Omega(\mathbf{M}) - P_\Omega(\mathbf{X}_k)). \tag{2.31b}$$

From (2.31a) and (2.31b), one can notice that the SVT algorithm is computationally efficient since we only need the truncated SVD and elementary matrix operations in each iteration. In fact, let $r_k$ be the number of singular values of $\mathbf{Y}_{k-1}$ being greater than the threshold $\tau$. Also, we suppose $\{r_k\}$ converges to the rank of the original matrix, i.e., $\lim_{k\to\infty} r_k = r$. Then the computational complexity of SVT is $\mathcal{O}(rn_1n_2)$. Note also that the iteration number to achieve the $\epsilon$-approximation[7] is $\mathcal{O}(\frac{1}{\sqrt{\epsilon}})$ [10]. In Table 2.1, we summarize the SVT algorithm. For the details of the stopping criterion of SVT, see [10, Section 5].

Over the years, various SVT-based techniques have been proposed [9, 56, 57]. In [56], an iterative matrix completion algorithm using the SVT-based operator called proximal operator has been proposed. Similar algorithms inspired by the iterative hard thresholding (IHT) algorithm in CS have also been proposed [9, 57].

### 2.4.3 Iteratively Reweighted Least Squares (IRLS) Minimization

Yet another simple and computationally efficient way to solve the NNM problem is the IRLS minimization technique [58, 59]. Note that the NNM problem can be recast using the least squares minimization as

$$\begin{aligned} \min_{\mathbf{X},\mathbf{W}} \quad & \|\mathbf{W}^{\frac{1}{2}}\mathbf{X}\|_F^2 \\ \text{subject to} \quad & P_\Omega(\mathbf{X}) = P_\Omega(\mathbf{M}), \end{aligned} \tag{2.32}$$

[7] By $\epsilon$-approximation, we mean $\|\widehat{\mathbf{M}} - \mathbf{M}^*\|_F \leq \epsilon$ where $\widehat{\mathbf{M}}$ is the reconstructed matrix and $\mathbf{M}^*$ is the optimal solution of SVT.

where $\mathbf{W} = (\mathbf{X}\mathbf{X}^T)^{-\frac{1}{2}}$. It can be shown that (2.32) is equivalent to the NNM problem (2.14) since we have [58]

$$\|\mathbf{X}\|_* = tr((\mathbf{X}\mathbf{X}^T)^{\frac{1}{2}}) = \|\mathbf{W}^{\frac{1}{2}}\mathbf{X}\|_F^2. \tag{2.33}$$

The key idea of the IRLS technique is to find $\mathbf{X}$ and $\mathbf{W}$ in an iterative fashion. The update expressions are given as

$$\mathbf{X}_k = \arg \min_{P_\Omega(\mathbf{X}) = P_\Omega(\mathbf{M})} \|\mathbf{W}_{k-1}^{\frac{1}{2}}\mathbf{X}\|_F^2, \tag{2.34a}$$

$$\mathbf{W}_k = (\mathbf{X}_k\mathbf{X}_k^T)^{-\frac{1}{2}}. \tag{2.34b}$$

For the weighted least squares subproblem (2.34a), we can easily find out its solution by updating each and every column of $\mathbf{X}_k$ [58]. Then, to update $\mathbf{W}_k$, we need a matrix inversion (2.34b) of $\mathbf{X}_k\mathbf{X}_k^T$. However, it is possible that some of the singular values of $\mathbf{X}_k$ approach to zero, resulting in an ill-conditioning of the matrix. To avoid this, an approach to use the perturbation of singular values has been proposed [58, 59]. Similar to SVT, the computational complexity per iteration of the IRLS-based technique is $\mathcal{O}(rn_1n_2)$. Also, IRLS requires $\mathcal{O}(\log(\frac{1}{\epsilon}))$ iterations to achieve the $\epsilon$-approximation solution. We summarize the IRLS minimization technique in Table 2.2.

## 2.5 LRMC Algorithms Using Rank Information

In many LRMC applications, the rank of a desired matrix is known in advance. Some examples include localization in IoT networks, recommendation system, image restoration, to name just a few. As aforementioned, the rank of a Euclidean distance matrix in a localization problem is at most $k + 2$ ($k$ is the dimension of the Euclidean space). In such situation, the LRMC problem can be more suitably formulated as a Frobenius norm minimization (FNM) problem:

$$\begin{aligned} \min_{\mathbf{X}} \quad & \frac{1}{2}\|P_\Omega(\mathbf{M}) - P_\Omega(\mathbf{X})\|_F^2 \\ \text{subject to} \quad & \text{rank}(\mathbf{X}) \leq r. \end{aligned} \tag{2.35}$$

Table 2.2: The IRLS Algorithm

| | |
|---|---|
| **Input** | a constant $q \geq r$, |
| | a scaling parameter $\gamma > 0$, |
| | and a stopping criterion $T$ |
| **Initialize** | iteration counter $k = 0$, |
| | a regularizing sequence $\epsilon_0 = 1$, |
| | and $\mathbf{W}_0 = \mathbf{I}$ |
| **While** | $T =$ false **do** |
| | $k = k + 1$ |
| | $\mathbf{X}_k = \arg\min_{P_\Omega(\mathbf{X})=P_\Omega(\mathbf{M})} \|\mathbf{W}_{k-1}^{\frac{1}{2}}\mathbf{X}\|_F^2$ |
| | $\epsilon_k = \min(\epsilon_{k-1}, \gamma\sigma_{q+1}(\mathbf{X}_k))$ |
| | Compute a SVD perturbation version $\widetilde{\mathbf{X}}_k$ of $\mathbf{X}_k$ [58] |
| | $\mathbf{W}_k = (\widetilde{\mathbf{X}}_k\widetilde{\mathbf{X}}_k^T)^{-\frac{1}{2}}$ |
| **End** | |
| **Output** | $\mathbf{X}^k$ |

Due to the inequality of the rank constraint, an approach to use the approximate rank information (e.g., upper bound of the rank) has been proposed [11]. The FNM problem has two main advantages: 1) the problem is well-posed in the noisy scenario and 2) the cost function is differentiable so that various gradient-based optimization techniques (e.g., gradient descent, conjugate gradient, Newton methods, and manifold optimization) can be used to solve the problem.

Over the years, various techniques to solve the FNM problem in (2.35) have been proposed [11, 17, 13, 14, 12, 19, 21, 20, 15, 18]. Well-known FNM-based LRMC techniques include greedy techniques [11], alternating projection techniques [13], and optimization over Riemannian manifold [20]. In this section, we explain these techniques in detail.

### 2.5.1 Greedy Techniques

Greedy algorithms have been popularly used for LRMC due to the low computational simplicity. In essence, they solve the LRMC problem by making a heuristic decision at each iteration with a hope to find the right solution in the end. To be specific, let $\mathbf{M} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$ be the singular value decomposition of $\mathbf{M} \in \mathbb{R}^{n\times n}$ where $\mathbf{U}, \mathbf{V} \in \mathbb{R}^{n\times r}$ and $r = \text{rank}(\mathbf{M})$. That is, $\mathbf{M}$ can be expressed as a linear combination of $r$ rank-one matrices as

$$\mathbf{M} = \sum_{i=1}^{r} \sigma_i(\mathbf{M})\mathbf{u}_i\mathbf{v}_i^T, \tag{2.36}$$

where $\sigma_i(\mathbf{M})$, $\mathbf{u}_i$, and $\mathbf{v}_i$ are the singular value, the left singular vector, and the right singular vector, respectively. Then the main task of greedy techniques is to investigate the *atom set* $\mathcal{A}_\mathbf{M} = \{\boldsymbol{\varphi}_i = \mathbf{u}_i\mathbf{v}_i^T\}_{i=1}^r$ of rank-one matrices representing $\mathbf{M}$. Once the atom set $\mathcal{A}_\mathbf{M}$ is found, the singular values $\sigma_i(\mathbf{M}) = \sigma_i$ can be computed easily by solving the following least squares problem:

$$(\sigma_1, \cdots, \sigma_r) = \arg\min_{\alpha_i} \|P_\Omega(\mathbf{M}) - P_\Omega(\sum_{i=1}^{r} \alpha_i\boldsymbol{\varphi}_i)\|_F. \tag{2.37}$$

Let $\mathbf{A} = [\text{vec}(P_\Omega(\boldsymbol{\varphi}_1)) \ \cdots \ \text{vec}(P_\Omega(\boldsymbol{\varphi}_r))]$, $\boldsymbol{\alpha} = [\alpha_1 \ \cdots \ \alpha_r]^T$ and $\mathbf{b} = \text{vec}(P_\Omega(\mathbf{M}))$. Then, we can easily find out the solution of (2.37) as

$$(\sigma_1, \cdots, \sigma_r) = \arg\min_{\boldsymbol{\alpha}} \|\mathbf{b} - \mathbf{A}\boldsymbol{\alpha}\|_2 = \mathbf{A}^\dagger \mathbf{b}.$$

One of the well-known greedy techniques in LRMC is atomic decomposition for minimum rank approximation (ADMiRA) [11], which can be viewed as an extension of the compressive sampling matching pursuit (CoSaMP) algorithm in CS [60, 44, 45, 46]. ADMiRA is based on the idea of adding as well as pruning to identify the atom set $\mathcal{A}_\mathbf{M}$. In the adding stage, ADMiRA identifies $2r$ rank-one matrices representing a residual best and then adds the matrices to the pre-chosen atom set. To be specific, if $\mathbf{X}_{i-1}$ is the output matrix generated in the $(i-1)$-th iteration and $\mathcal{A}_{i-1}$ is its atom set, then ADMiRA computes the residual $\mathbf{R}_i = P_\Omega(\mathbf{M}) - P_\Omega(\mathbf{X}_{i-1})$ and then adds $2r$ leading principal components of $\mathbf{R}_i$ to $\mathcal{A}_{i-1}$. That is, the enlarged atom set $\Psi_i$ is given by

$$\Psi_i = \mathcal{A}_{i-1} \cup \{\mathbf{u}_{\mathbf{R}_i,j}\mathbf{v}_{\mathbf{R}_i,j}^T : 1 \le j \le 2r\}, \tag{2.38}$$

where $\mathbf{u}_{\mathbf{R}_i,j}$ and $\mathbf{v}_{\mathbf{R}_i,j}$ are the $j$-th principal left and right singular vectors of $\mathbf{R}_i$, respectively. As a result, $\Psi_i$ includes $3r$ rank-one matrices in total. In the pruning stage, ADMiRA refines $\Psi_i$ into a set of $r$ atoms. To be specific, if $\widetilde{\mathbf{X}}_i$ is the best rank-$3r$ approximation of $\mathbf{M}$, that is,[8]

$$\widetilde{\mathbf{X}}_i = \arg\min_{\mathbf{X} \in \text{span}(\Psi_i)} \|P_\Omega(\mathbf{M}) - P_\Omega(\mathbf{X})\|_F, \tag{2.39}$$

then the refined atom set $\mathcal{A}_i$ is expressed as

$$\mathcal{A}_i = \{\mathbf{u}_{\widetilde{\mathbf{X}}_i,j}\mathbf{v}_{\widetilde{\mathbf{X}}_i,j}^T : 1 \le j \le r\}, \tag{2.40}$$

where $\mathbf{u}_{\widetilde{\mathbf{X}}_i,j}$ and $\mathbf{v}_{\widetilde{\mathbf{X}}_i,j}$ are the $j$-th principal left and right singular vectors of $\widetilde{\mathbf{X}}_i$, respectively. We note that the computational complexity of ADMiRA is mainly due

[8] Note that the solution to (2.39) can be computed in a similar way as in (2.37).

Table 2.3: The ADMiRA Algorithm

**Input** observed entries $P_\Omega(\mathbf{M}) \in \mathbb{R}^{n\times n}$,
rank of a desired low-rank matrix $r$,
and a stopping criterion $T$

**Initialize** iteration counter $k = 0$,
$\mathbf{X}_0 = \mathbf{0}_{n\times n}$,
and $\mathcal{A}_0 = \emptyset$

**While** $T = \text{false}$ **do**

$\mathbf{R}_k = P_\Omega(\mathbf{M}) - P_\Omega(\mathbf{X}_k)$

$[\mathbf{U}_{\mathbf{R}_k}, \mathbf{\Sigma}_{\mathbf{R}_k}, \mathbf{V}_{\mathbf{R}_k}] = \text{svds}(\mathbf{R}_k, 2r)$

(Augment) $\Psi_{k+1} = \mathcal{A}_k \cup \{\mathbf{u}_{\mathbf{R}_k,j}\mathbf{v}^T_{\mathbf{R}_k,j} : 1 \le j \le 2r\}$

$\widetilde{\mathbf{X}}_{k+1} = \underset{\mathbf{X}\in\text{span}(\Psi_{k+1})}{\arg\min} \|P_\Omega(\mathbf{M}) - P_\Omega(\mathbf{X})\|_F$ using (2.37)

$[\mathbf{U}_{\widetilde{\mathbf{X}}_{k+1}}, \mathbf{\Sigma}_{\widetilde{\mathbf{X}}_{k+1}}, \mathbf{V}_{\widetilde{\mathbf{X}}_{k+1}}] = \text{svds}(\widetilde{\mathbf{X}}_{k+1}, r)$

(Prune) $\mathcal{A}_{k+1} = \{\mathbf{u}_{\widetilde{\mathbf{X}}_{k+1},j}\mathbf{v}^T_{\widetilde{\mathbf{X}}_{k+1},j} : 1 \le j \le r\}$

(Estimate) $\mathbf{X}_{k+1} = \underset{\mathbf{X}\in\text{span}(\mathcal{A}_{k+1})}{\arg\min} \|P_\Omega(\mathbf{M}) - P_\Omega(\mathbf{X})\|_F$
using (2.37)

$k = k + 1$

**End**

**Output** $\mathcal{A}_k$, $\mathbf{X}_k$

to two operations: the least squares operation in (2.37) and the SVD-based operation to find out the leading atoms of the required matrix (e.g., $\mathbf{R}_k$ and $\widetilde{\mathbf{X}}_{\mathbf{k+1}}$). In fact, since (2.37) involves the pseudo-inverse of $\mathbf{A}$ (size of $|\Omega| \times \mathcal{O}(r)$), its computational cost is $\mathcal{O}(r|\Omega|)$. Also, the computational cost of performing a truncated SVD of $\mathcal{O}(r)$ atoms is $\mathcal{O}(rn_1n_2)$. Since $|\Omega| < n_1n_2$, the computational complexity of ADMiRA per iteration is $\mathcal{O}(rn_1n_2)$. For the convergence rate, it has been shown that the iteration number of ADMiRA to achieve the $\epsilon$-approximation is $\mathcal{O}(\log(\frac{1}{\epsilon}))$ [11]. In Table 2.3, we summarize the ADMiRA algorithm.

Another well-known greedy method is the rank-one matrix pursuit algorithm [17], an extension of the orthogonal matching pursuit algorithm in CS [61]. This approach is also known as a simplified version of ADMiRA. In this approach, instead of choosing multiple atoms of a matrix, an atom corresponding to the largest singular value of the residual matrix $\mathbf{R}_k$ is chosen. Interested readers may refer to [17] for more details.

### 2.5.2 Alternating Minimization Techniques

As an effort to further reduce the computational burden of SVD (expressed as $\mathcal{O}(rn^2)$), which has been used in many of LRMC algorithms [10, 11], alternating minimization techniques have been proposed [13, 14, 12]. In these techniques, the desired low-rank matrix $\mathbf{M} \in \mathbb{R}^{n_1 \times n_2}$ of rank $r$ is factorized into tall and fat matrices, i.e., $\mathbf{M} = \mathbf{XY}$ where $\mathbf{X} \in \mathbb{R}^{n_1 \times r}$ and $\mathbf{Y} \in \mathbb{R}^{r \times n_2}$ ($r \ll n_1, n_2$). Using this factorization model, one can find out $\mathbf{X}$ and $\mathbf{Y}$ by minimizing the residual defined as the difference between the original matrix and the estimate of it on the sampling space. That is, they recover $\mathbf{X}$ and $\mathbf{Y}$ by solving

$$\min_{\mathbf{X},\mathbf{Y}} \quad \frac{1}{2}\|P_\Omega(\mathbf{M}) - P_\Omega(\mathbf{XY})\|_F^2. \tag{2.41}$$

As a simple alternating minimization algorithm, power factorization can be used to update $\mathbf{X}$ and $\mathbf{Y}$ in an alternating manner as [13]

$$\mathbf{X}_{i+1} = \arg\min_{\mathbf{X}} \|P_\Omega(\mathbf{M}) - P_\Omega(\mathbf{X}\mathbf{Y}_i)\|_F^2, \tag{2.42a}$$

$$\mathbf{Y}_{i+1} = \arg\min_{\mathbf{Y}} \|P_\Omega(\mathbf{M}) - P_\Omega(\mathbf{X}_{i+1}\mathbf{Y})\|_F^2. \tag{2.42b}$$

Alternating steepest descent (ASD) is another alternating method to find out the solution of (2.41) [14]. In ASD, $\mathbf{X}$ and $\mathbf{Y}$ are updated by applying the steepest gradient descent method to the objective function $f(\mathbf{X}, \mathbf{Y}) = \frac{1}{2}\|P_\Omega(\mathbf{M}) - P_\Omega(\mathbf{X}\mathbf{Y})\|_F^2$ in (2.41). To be specific, ASD first computes the gradient of $f(\mathbf{X}, \mathbf{Y})$ with respect to $\mathbf{X}$ and then updates $\mathbf{X}$ along the steepest gradient descent direction:

$$\mathbf{X}_{i+1} = \mathbf{X}_i - t_{x_i} \bigtriangledown f_{\mathbf{Y}_i}(\mathbf{X}_i), \tag{2.43}$$

where the gradient descent direction $\bigtriangledown f_{\mathbf{Y}_i}(\mathbf{X}_i)$ and stepsize $t_{x_i}$ are given by

$$\bigtriangledown f_{\mathbf{Y}_i}(\mathbf{X}_i) = -(P_\Omega(\mathbf{M}) - P_\Omega(\mathbf{X}_i\mathbf{Y}_i))\mathbf{Y}_i^T, \tag{2.44a}$$

$$t_{x_i} = \frac{\| \bigtriangledown f_{\mathbf{Y}_i}(\mathbf{X}_i)\|_F^2}{\|P_\Omega(\bigtriangledown f_{\mathbf{Y}_i}(\mathbf{X}_i)\mathbf{Y}_i)\|_F^2}. \tag{2.44b}$$

After updating $\mathbf{X}$, ASD updates $\mathbf{Y}$ in a similar way:

$$\mathbf{Y}_{i+1} = \mathbf{Y}_i - t_{y_i} \bigtriangledown f_{\mathbf{X}_{i+1}}(\mathbf{Y}_i), \tag{2.45}$$

where

$$\bigtriangledown f_{\mathbf{X}_{i+1}}(\mathbf{Y}_i) = -\mathbf{X}_{i+1}^T(P_\Omega(\mathbf{M}) - P_\Omega(\mathbf{X}_{i+1}\mathbf{Y}_i)), \tag{2.46a}$$

$$t_{y_i} = \frac{\| \bigtriangledown f_{\mathbf{X}_{i+1}}(\mathbf{Y}_i)\|_F^2}{\|P_\Omega(\mathbf{X}_{i+1} \bigtriangledown f_{\mathbf{X}_{i+1}}(\mathbf{Y}_i))\|_F^2}. \tag{2.46b}$$

The low-rank matrix fitting (LMaFit) algorithm finds out the solution in a different way by solving [12]

$$\arg\min_{\mathbf{X},\mathbf{Y},\mathbf{Z}} \{\|\mathbf{X}\mathbf{Y} - \mathbf{Z}\|_F^2 : P_\Omega(\mathbf{Z}) = P_\Omega(\mathbf{M})\}. \tag{2.47}$$

With the arbitrary input of $\mathbf{X}_0 \in \mathbb{R}^{n_1 \times r}$ and $\mathbf{Y}_0 \in \mathbb{R}^{r \times n_2}$ and $\mathbf{Z}_0 = P_\Omega(\mathbf{M})$, the variables $\mathbf{X}$, $\mathbf{Y}$, and $\mathbf{Z}$ are updated in the $i$-th iteration as

$$\mathbf{X}_{i+1} = \arg\min_{\mathbf{X}} \|\mathbf{X}\mathbf{Y}_i - \mathbf{Z}_i\|_F^2 = \mathbf{Z}_i\mathbf{Y}^\dagger, \tag{2.48a}$$

$$\mathbf{Y}_{i+1} = \arg\min_{\mathbf{Y}} \|\mathbf{X}_i\mathbf{Y} - \mathbf{Z}_i\|_F^2 = \mathbf{X}_{i+1}^\dagger\mathbf{Z}_i, \tag{2.48b}$$

$$\mathbf{Z}_{i+1} = \mathbf{X}_{i+1}\mathbf{Y}_{i+1} + P_\Omega(\mathbf{M} - \mathbf{X}_{i+1}\mathbf{Y}_{i+1}), \tag{2.48c}$$

where $\mathbf{X}^\dagger$ is Moore-Penrose pseudoinverse of matrix $\mathbf{X}$.

The computational cost of the alternating minimization algorithms is not much expensive due to the following reasons: 1) it does not require the SVD computation and 2) the size of matrices to be inverted is smaller than the size of matrices for the greedy algorithms. While the inversion of huge size matrices (size of $|\Omega| \times \mathcal{O}(1)$) is required in a greedy algorithms (see (2.37)), alternating minimization only requires the pseudo inversion of $\mathbf{X}$ and $\mathbf{Y}$ (size of $n_1 \times r$ and $r \times n_2$, respectively). In fact, the computational complexity of this approach is $\mathcal{O}(r|\Omega| + r^2 n_1 + r^2 n_2)$, which is much smaller than that of SVT and ADMiRA when $r \ll \min(n_1, n_2)$. Also, the iteration number of ASD and LMaFit to achieve the $\epsilon$-approximation is $\mathcal{O}(\log(\frac{1}{\epsilon}))$ [14, 12]. It has been shown that alternating minimization techniques are simple to implement and also require small sized memory [16]. Major drawback of these approaches is that it might converge to the local optimum.

### 2.5.3 Optimization over Smooth Riemannian Manifold

In many practical situations, when the rank of the desired matrix is known in a priori (i.e., $\text{rank}(\mathbf{M}) = r$), one can strengthen the constraint of (2.35) by defining the feasible set, denoted by $\mathcal{F}$, as

$$\mathcal{F} = \{\mathbf{X} \in \mathbb{R}^{n_1 \times n_2} \ : \ \text{rank}(\mathbf{X}) = r\}.$$

Since $\mathcal{F}$ is not a vector space[9], conventional optimization techniques in the Euclidean space cannot be used to solve the problem defined over $\mathcal{F}$. While this is bad news, a remedy for this is that $\mathcal{F}$ is a smooth Riemannian manifold [62, 19]. Loosely speaking, smooth manifold is a generalization of $\mathbb{R}^{n_1 \times n_2}$ on which a notion of differentiability exists. Interested readers may refer to [63, 64] for more rigorous definition of manifold. A smooth manifold equipped with an inner product, often called a Riemannian metric, forms a smooth Riemannian manifold. Since the smooth Riemannian manifold is a differentiable structure equipped with an inner product, one can use all necessary ingredients to solve the optimization problem with quadratic cost function, such as Riemannian gradient, Hessian matrix, exponential map, and parallel translation [63]. Therefore, optimization techniques in $\mathbb{R}^{n_1 \times n_2}$ (e.g., steepest descent, Newton method, conjugate gradient method) can be used to solve (2.35) in the smooth Riemannian manifold $\mathcal{F}$.

Recently, many efforts have been made to solve the matrix completion over smooth Riemannian manifolds. These works are classified by their specific choice of Riemannian manifold structure. One well-known approach is to solve (2.35) over the Grassmann manifold of orthogonal matrices[10] [21]. In this approach, a feasible set can be expressed as $\mathcal{F} = \{\mathbf{Q}\mathbf{R}^T : \mathbf{Q}^T\mathbf{Q} = \mathbf{I}, \mathbf{Q} \in \mathbb{R}^{n_1 \times r}, \mathbf{R} \in \mathbb{R}^{n_2 \times r}\}$ and thus solving (2.35) is to find an $n_1 \times r$ orthonormal matrix $\mathbf{Q}$ satisfying

$$f(\mathbf{Q}) = \min_{\mathbf{R} \in \mathbb{R}^{n_2 \times r}} \|P_\Omega(\mathbf{M}) - P_\Omega(\mathbf{Q}\mathbf{R}^T)\|_F^2 = 0. \tag{2.49}$$

In [21], an approach to solve (2.49) over the Grassmann manifold has been proposed.

Recently, it has been shown that the original matrix can be reconstructed by the unconstrained optimization over the smooth Riemannian manifold $\mathcal{F}$ [20]. Often, $\mathcal{F}$ is

[9]This is because if $\text{rank}(\mathbf{X}) = r$ and $\text{rank}(\mathbf{Y}) = r$, then $\text{rank}(\mathbf{X} + \mathbf{Y}) = r$ is not necessarily true (and thus $\mathbf{X} + \mathbf{Y}$ does not need to belong $\mathcal{F}$).

[10]The Grassmann manifold is defined as the set of the linear subspaces in a vector space [63].

expressed using the singular value decomposition as

$$\mathcal{F} = \{\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T : \mathbf{U} \in \mathbb{R}^{n_1 \times r}, \mathbf{V} \in \mathbb{R}^{n_2 \times r}, \mathbf{\Sigma} \succeq 0, \\ \mathbf{U}^T\mathbf{U} = \mathbf{V}^T\mathbf{V} = \mathbf{I}, \mathbf{\Sigma} = \text{diag}([\sigma_1 \ \cdots \ \sigma_r])\}. \tag{2.50}$$

The FNM problem (2.35) can then be reformulated as an unconstrained optimization over $\mathcal{F}$:

$$\min_{\mathbf{X} \in \mathcal{F}} \quad \frac{1}{2}\|P_\Omega(\mathbf{M}) - P_\Omega(\mathbf{X})\|_F^2. \tag{2.51}$$

One can easily obtain the closed-form expression of the ingredients such as tangent spaces, Riemannian metric, Riemannian gradient, and Hessian matrix in the unconstrained optimization [62, 63, 64]. In fact, major benefits of the Riemannian optimization-based LRMC techniques are the simplicity in implementation and the fast convergence. Similar to ASD, the computational complexity per iteration of these techniques is $\mathcal{O}(r|\Omega| + r^2n_1 + r^2n_2)$, and they require $\mathcal{O}(\log(\frac{1}{\epsilon}))$ iterations to achieve the $\epsilon$-approximation solution [20].

### 2.5.4 Truncated NNM

Truncated NNM is a variation of the NNM-based technique requiring the rank information $r$.[11] While the NNM technique takes into account all the singular values of a desired matrix, truncated NNM considers only the $n - r$ smallest singular values [18]. To be specific, truncated NNM finds a solution to

$$\begin{aligned} \min_{\mathbf{X}} \quad & \|\mathbf{X}\|_r \\ \text{subject to} \quad & P_\Omega(\mathbf{X}) = P_\Omega(\mathbf{M}), \end{aligned} \tag{2.52}$$

[11] Although truncated NNM is a variant of NNM, we put it into the second category since it exploits the rank information of a low-rank matrix.

Table 2.4: Truncated NNM

| | |
|---|---|
| **Input** | observed entries $P_\Omega(\mathbf{M}) \in \mathbb{R}^{n \times n}$, |
| | rank of a desired low-rank matrix $r$, |
| | and stopping threshold $\epsilon > 0$ |
| **Initialize** | iteration counter $k = 0$, |
| | and $\mathbf{X}_0 = P_\Omega(\mathbf{M})$ |
| **While** | $\|\mathbf{X}_k - \mathbf{X}_{k-1}\|_F > \epsilon$ **do** |
| | $[\mathbf{U}_k, \mathbf{\Sigma}_k, \mathbf{V}_k] = \text{svd}(\mathbf{X}_k)$ $(\mathbf{U}_k, \mathbf{V}_k \in \mathbb{R}^{n \times r})$ |
| | $\mathbf{X}^{k+1} = \underset{\mathbf{X}: P_\Omega(\mathbf{X}) = P_\Omega(\mathbf{M})}{\arg\min} \|\mathbf{X}\|_* - \text{tr}(\mathbf{U}_k^T \mathbf{X} \mathbf{V}_k)$ |
| | $k = k + 1$ |
| **End** | |
| **Output** | $\mathbf{X}^k$ |

where $\|\mathbf{X}\|_r = \sum_{i=r+1}^{n} \sigma_i(\mathbf{X})$. We recall that $\sigma_i(\mathbf{X})$ is the $i$-th largest singular value of $\mathbf{X}$. Using [18]

$$\sum_{i=1}^{r} \sigma_i = \max_{\mathbf{U}^T\mathbf{U}=\mathbf{V}^T\mathbf{V}=\mathbf{I}_r} \text{tr}(\mathbf{U}^T\mathbf{X}\mathbf{V}), \tag{2.53}$$

we have

$$\|\mathbf{X}\|_r = \|\mathbf{X}\|_* - \max_{\mathbf{U}^T\mathbf{U}=\mathbf{V}^T\mathbf{V}=\mathbf{I}_r} \text{tr}(\mathbf{U}^T\mathbf{X}\mathbf{V}), \tag{2.54}$$

and thus the problem (2.52) can be reformulated to

$$\begin{aligned} \min_{\mathbf{X}} \quad & \|\mathbf{X}\|_* - \max_{\mathbf{U}^T\mathbf{U}=\mathbf{V}^T\mathbf{V}=\mathbf{I}_r} \text{tr}(\mathbf{U}^T\mathbf{X}\mathbf{V}) \\ \text{subject to} \quad & P_\Omega(\mathbf{X}) = P_\Omega(\mathbf{M}), \end{aligned} \tag{2.55}$$

which can be solved in an iterative way. To be specific, starting from $\mathbf{X}_0 = P_\Omega(\mathbf{M})$, truncated NNM updates $\mathbf{X}_i$ by solving [18]

$$\begin{aligned} \min_{\mathbf{X}} \quad & \|\mathbf{X}\|_* - \text{tr}(\mathbf{U}_{i-1}^T\mathbf{X}\mathbf{V}_{i-1}) \\ \text{subject to} \quad & P_\Omega(\mathbf{X}) = P_\Omega(\mathbf{M}), \end{aligned} \tag{2.56}$$

where $\mathbf{U}_{i-1}, \mathbf{V}_{i-1} \in \mathbb{R}^{n\times r}$ are the matrices of left and right-singular vectors of $\mathbf{X}_{i-1}$, respectively. Note that an approach in (2.56) has two main advantages: 1) the rank information of the desired matrix can be incorporated and 2) various gradient-based techniques including alternating direction method of multipliers (ADMM) [65, 66], ADMM with an adaptive penalty (ADMMAP) [67], and accelerated proximal gradient line search method (APGL)[68] can be employed. Note also that the dominant operation is the truncated SVD operation and its complexity is $\mathcal{O}(rn_1n_2)$, which is much smaller than that of the NNM technique (see Table 2.5) as long as $r \ll \min(n_1, n_2)$. Similar to SVT, the iteration complexity of the truncated NNM to achieve the $\epsilon$-approximation is $\mathcal{O}(\frac{1}{\sqrt{\epsilon}})$ [18]. Alternatively, the difference of two convex functions (DC) based algorithm can be used to solve (2.55) [69]. In Table 2.4, we summarize the truncated NNM algorithm.

## 2.6 Performance Guarantee

When using LRMC techniques, one might concern the performance guarantee to reconstruct the desired low-rank matrix $\mathbf{M}$. In NNM-based techniques, exact recovery of $\mathbf{M}$ can be guaranteed based on the uniqueness condition of the NNM problem [5, 6, 41]. To be specific, let $\mathbf{M} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$ be the SVD of $\mathbf{M}$ where $\mathbf{U} \in \mathbb{R}^{n_1 \times r}$, $\mathbf{\Sigma} \in \mathbb{R}^{r \times r}$, and $\mathbf{V} \in \mathbb{R}^{n_2 \times r}$. Also, let $\mathbb{R}^{n_1 \times n_2} = T \bigoplus T^{\perp}$ be the orthogonal decomposition in which $T^{\perp}$ is defined as the subspace of matrices whose row and column spaces are orthogonal to the row and column spaces of $\mathbf{M}$, respectively. Here, $T$ is the orthogonal complement of $T^{\perp}$. Since the NNM problem is a convex optimization with $\mathbf{M}$ being a feasible, the important observation is that $\mathbf{M}$ is the unique solution under two conditions [5, Lemma 3.1]:

1) there exists a matrix $\mathbf{Y} = \mathbf{U}\mathbf{V}^T + \mathbf{W}$ such that $P_{\Omega}(\mathbf{Y}) = \mathbf{Y}$, $\mathbf{W} \in T^{\perp}$, and $\|\mathbf{W}\| < 1$,

2) the restriction of the sampling operation $P_{\Omega}$ to $T$ is an injective (one-to-one) mapping.

The establishment of $\mathbf{Y}$ obeying 1) and 2) are in turn conditioned on the observation model of $\mathbf{M}$ and its intrinsic coherence property.

Under a uniform sampling model of $\mathbf{M}$, suppose the coherence property of $\mathbf{M}$ satisfies

$$\max(\mu(\mathbf{U}), \mu(\mathbf{V})) \leq \mu_0, \tag{2.57a}$$

$$\max_{ij} |e_{ij}| \leq \mu_1 \sqrt{\frac{r}{n_1 n_2}}, \tag{2.57b}$$

where $\mu_0$ and $\mu_1$ are some constants, $e_{ij}$ is the entry of $\mathbf{E} = \mathbf{U}\mathbf{V}^T$, and $\mu(\mathbf{U})$ and $\mu(\mathbf{V})$ are the coherences of the column and row spaces of $\mathbf{M}$, respectively.

**Theorem 4 ([5, Theorem 1.3])** *There exists constants $\alpha$ and $\beta$ such that if the number*

*of observed entries* $m = |\Omega|$ *satisfies*

$$m \geq \alpha \max(\mu_1^2, \mu_0^{\frac{1}{2}}\mu_1, \mu_0 n^{\frac{1}{4}})\gamma n r \log n \tag{2.58}$$

*where* $\gamma > 2$ *is some constant and* $n_1 = n_2 = n$*, then* $\mathbf{M}$ *is the unique solution of the NNM problem with probability at least* $1 - \beta n^{-\gamma}$*. Further, if* $r \leq \mu_0^{-1} n^{1/5}$*,* (2.58) *can be improved to* $m \geq C\mu_0 \gamma n^{1.2} r \log n$ *with the same success probability.*

Intuitively, the desired low-rank matrix can be reconstructed (with no error) with overwhelming probability even when $m$ is much less than $n_1 n_2$. Alternatively, the exact recovery of NNM has been shown under the restricted isometry property (RIP) based condition of a linear sampling operator [55].

In FNM-based techniques, the performance guarantee of greedy algorithms has been given under the RIP-based condition of the sampling operation [11]. To be specific, suppose that $\text{rank}(\mathbf{M}) = r$ and $\delta_{4r}$ is the RIP constant [55] satisfying $\delta_{4r} \leq 0.065$. Let $\{\mathbf{X}_k\}$ be the generated sequence of ADMiRA. Then the global convergence is guaranteed by

$$\|\mathbf{X}_k - \mathbf{M}\|_F \leq 2^{-k}\|\mathbf{X}_0\|_F. \tag{2.59}$$

where $\mathbf{X}_0$ is the initial value.

Recently, the recovery guarantee of the LRMC techniques using gradient-based algorithms has been proposed [70, 71, 72]. Consider the FNM problem (2.35) with the fixed rank constraint as

$$\begin{aligned} \min_{\mathbf{X}} \quad & \frac{1}{2}\|P_\Omega(\mathbf{M}) - P_\Omega(\mathbf{X})\|_F^2 \\ \text{subject to} \quad & \text{rank}(\mathbf{X}) = r. \end{aligned} \tag{2.60}$$

Suppose there exists a positive constant $\lambda$ such that

$$\mu(\mathbf{U}) \leq \sqrt{\frac{\lambda r}{n_2}}, \text{ and } \mu(\mathbf{V}) \leq \sqrt{\frac{\lambda r}{n_1}}. \tag{2.61}$$

Then under some suitable conditions, it has been shown that if the sampling ratio $p = |\Omega|/(n_1 n_2)$ satisfies

$$p \geq \Omega(\frac{\lambda^4 r^6 \kappa^6 \log n_o}{\min(n_1, n_2)}) \tag{2.62}$$

where $n_o = \max(n_1, n_2)$ and $\kappa$ is the condition number of $\mathbf{M}$, then gradient-based algorithms to solve (2.60) globally converges to $\mathbf{M}$ with the probability at least $1 - 1/\text{poly}(n_o)$ [71]. Here, poly$(n_o)$ is some polynomial function of $n_o$. Intuitively, as the number of the observed entries is large enough, all local optima of (2.60) becomes the global optimum at $\mathbf{M}$.

## 2.7 Empirical Performance Evaluation

In this section, we test the performance of the LRMC algorithms listed in Table 2.5 and 2.6. In our experiments, we generate the original matrix as the product of two random matrices $\mathbf{A} \in \mathbb{R}^{n_1 \times r}$ and $\mathbf{B} \in \mathbb{R}^{n_2 \times r}$, i.e., $\mathbf{M} = \mathbf{A}\mathbf{B}^T$. Entries of these two matrices are identically and independently distributed (i.i.d.) random variables according to the normal distribution $\mathcal{N}(0, 1)$. Sampled elements are also chosen at random with the sampling ratio $p$ defined as

$$p = \frac{|\Omega|}{n_1 n_2},$$

where $|\Omega|$ is the cardinality (number of elements) of $\Omega$. In the noisy scenario, we use the additive noise model where the observed matrix $\mathbf{M}_o$ is expressed as $\mathbf{M}_o = \mathbf{M} + \mathbf{N}$ where the noise matrix $\mathbf{N}$ is formed by the i.i.d. random entries sampled from the Gaussian distribution $\mathcal{N}(0, \sigma^2)$. For given SNR, $\sigma^2 = \frac{1}{n_1 n_2} \|\mathbf{M}\|_F^2 10^{-\frac{\text{SNR}}{10}}$. Note that the parameters of the LRMC algorithm are chosen from the reference paper. For each point of the algorithm, we run $1,000$ independent trials and then plot the average value.

As the performance metrics, we use the mean square error (MSE) and the exact

Table 2.5: Summary of the NNM-based LRMC algorithms.

| Technique | Algorithm | Features | Cost Complexity | Iter. Complexity* |
|---|---|---|---|---|
| Convex Optimization | SDPT3 (CVX) [47] | A solver for conic programming problems | $\mathcal{O}(n^3)$ | $\mathcal{O}(n^\omega \log(\frac{1}{\epsilon}))$ |
| NNM via Singular Value Thresholding | SVT [10] | An extension of the iterative soft thresholding technique in compressed sensing for LRMC, based on a Lagrange multiplier method | $\mathcal{O}(rn^2)$ | $\mathcal{O}(\frac{1}{\sqrt{\epsilon}})$ |
| | NIHT [9] | An extension of the iterative hard thresholding technique [8] in compressed sensing for LRMC | $\mathcal{O}(rn^2)$ | $\mathcal{O}(\log(\frac{1}{\epsilon}))$ |
| IRLS Minimization | IRLS-M Algorithm [58] | An algorithm to solve the NNM problem by computing the solution of a weighted least squares subproblem in each iteration | $\mathcal{O}(rn^2)$ | $\mathcal{O}(\log(\frac{1}{\epsilon}))$ |

* The number of iterations to achieve the reconstructed matrix $\widehat{\mathbf{M}}$ satisfying $\|\widehat{\mathbf{M}} - \mathbf{M}^*\|_F \leq \epsilon$ where $\mathbf{M}^*$ is the optimal solution.

* The rank of a desired low-rank matrix is $r$ and $n = \max(n_1, n_2)$.

Table 2.6: Summary of the FNM-based LRMC algorithms.

| Technique | Algorithm | Features | Cost Complexity | Iter. Complexity* |
|---|---|---|---|---|
| Greedy Technique | ADMiRA [11] | An extension of the greedy algorithm CoSaMP [60, 44] in compressed sensing for LRMC, uses greedy projection to identify a set of rank-one matrices that best represents the original matrix | $\mathcal{O}(rn^2)$ | $\mathcal{O}(\log(\frac{1}{\epsilon}))$ |
| Alternating Minimization | LMaFit [12] | A nonlinear successive over-relaxation LRMC algorithm based on nonlinear Gauss-Seidel method | $\mathcal{O}(r\|\Omega\|+r^2n)$ | $\mathcal{O}(\log(\frac{1}{\epsilon}))$ |
| | ASD [14] | A steepest decent algorithm for the FNM-based LRMC problem (2.35) | $\mathcal{O}(r\|\Omega\|+r^2n)$ | $\mathcal{O}(\log(\frac{1}{\epsilon}))$ |
| Manifold Optimization | SET [21] | A gradient-based algorithm to solve the FNM problem on a Grassmann manifold | $\mathcal{O}(r\|\Omega\|+r^2n)$ | $\mathcal{O}(\log(\frac{1}{\epsilon}))$ |
| | LRGeomCG [20] | A conjugate gradient algorithm over a Riemannian manifold of the fixed-rank matrices | $\mathcal{O}(r\|\Omega\|+r^2n)$ | $\mathcal{O}(\log(\frac{1}{\epsilon}))$ |
| Truncated NNM | TNNR-APGL [18] | This algorithm solves the truncated NNM problem via accelerated proximal gradient line search method [68] | $\mathcal{O}(rn^2)$ | $\mathcal{O}(\frac{1}{\sqrt{\epsilon}})$ |
| | TNNR-ADMM [18] | This algorithm solves the truncated NNM problem via an alternating direction method of multipliers [65] | $\mathcal{O}(rn^2)$ | $\mathcal{O}(\frac{1}{\sqrt{\epsilon}})$ |
| CNN-based Technique | CNN-based LRMC Algorithm [73] | An gradient-based algorithm to express a low-rank matrix as a graph structure and then apply CNN to the constructed graph to recover the desired matrix | $\mathcal{O}(r\|\Omega\|+r^2n)$ | $\mathcal{O}(\log(\frac{1}{\epsilon}))$ |

* The number of iterations to achieve the reconstructed matrix $\widehat{\mathbf{M}}$ satisfying $\|\widehat{\mathbf{M}} - \mathbf{M}^*\|_F \leq \epsilon$ where $\mathbf{M}^*$ is the optimal solution.

* The rank of a desired low-rank matrix is $r$ and $n = \max(n_1, n_2)$.

recovery ratio, which are defined, respectively, as

$$\text{MSE} = \frac{1}{n_1 n_2}\|\widehat{\mathbf{M}} - \mathbf{M}\|_F^2,$$
$$R = \frac{\text{number of successful trials}}{\text{total trials}},$$

where $\widehat{\mathbf{M}}$ is the reconstructed low-rank matrix. We say the trial is successful if the MSE performance is less than the threshold $\epsilon$. In our experiments, we set $\epsilon = 10^{-6}$. Here, $R$ can be used to represent the probability of successful recovery.

We first examine the success recovery of the LRMC algorithms with respect to the sampling ratio and the rank of $\mathbf{M}$. In our experiments, we set $n_1 = n_2 = 100$ and compute the phase transition [74] of the LRMC algorithms. Note that the phase transition is a contour plot of the success probability $P$ (we set $P = 0.5$) where the sampling ratio ($x$-axis) and the rank ($y$-axis) form a regular grid of the $x$-$y$ plane. The contour plot separates the plane into two areas: the area above the curve is one satisfying $P < 0.5$ and the area below the curve is a region achieving $P > 0.5$ [74] (see Fig. 2.8). The higher the curve, therefore, the better the algorithm would be. In general, the LRMC algorithms perform poor when the matrix has a small number of observed entries and the rank is large. Overall, NNM-based algorithms perform better than FNM-based algorithms. In particular, the NNM technique using SDPT3 solver outperforms the rest because the convex optimization technique always finds a global optimum while other techniques often converge to local optimum.

In order to estimate the computational efficiency of LRMC algorithms, we measure the running time of each algorithm as a function of rank (see Fig. 2.9). The running time is measured in second, using a 64-bit PC with an Intel i5-4670 CPU running at 3.4 GHz. We observe that the convex algorithms have a relatively high running time complexity.

We next test the efficiency of the LRMC algorithms for different problem size (see Table 2.7). For iterative LRMC algorithms, we set the maximum number of iteration to 300. We see that LRMC algorithms such as SVT, IRLS-M, ASD, ADMiRA, and

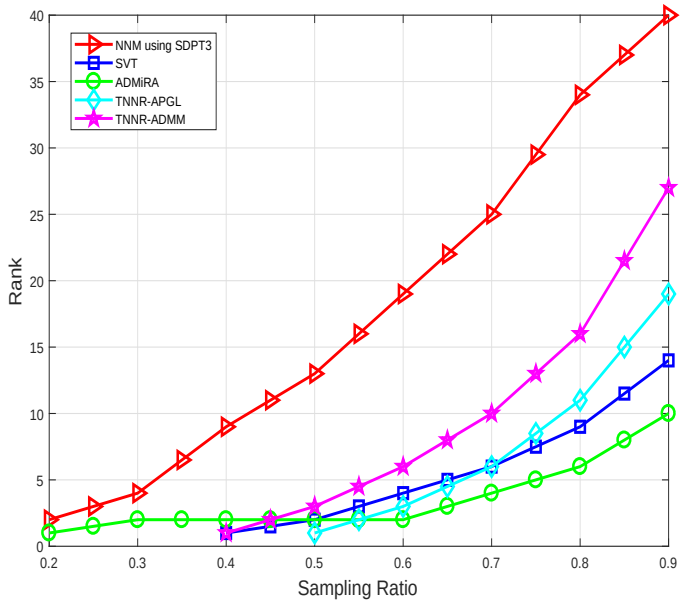



Figure 2.8: Phase transition of LRMC algorithms.

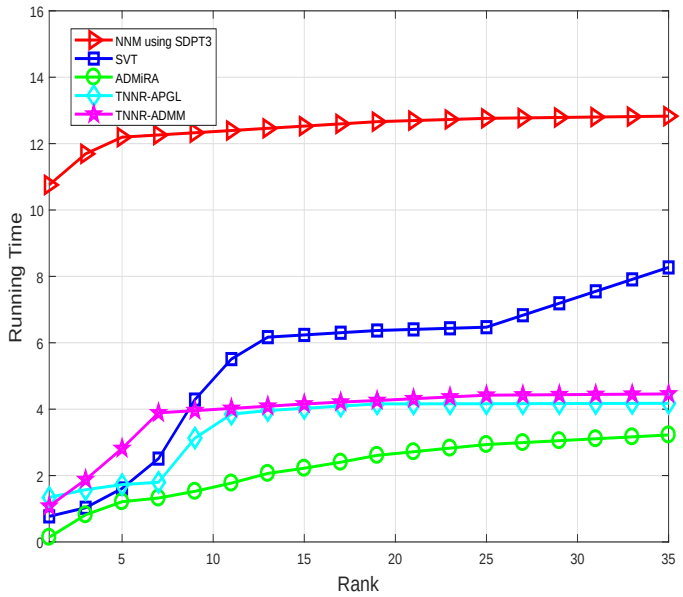



Figure 2.9: Running times of LRMC algorithms in noiseless scenario (40% of entries are observed).

LRGeomCG run fast. For example, it takes less than a minute for these algorithms to reconstruct $1000 \times 1000$ matrix, while the running time of SDPT3 solver is more than 5 minutes. Further reduction of the running time can be achieved using the alternating projection-based algorithms such as LMaFit. For example, it takes about one second to reconstruct an $(1000 \times 1000)$-dimensional matrix with rank $5$ using LMaFit. Therefore, when the exact recovery of the original matrix is unnecessary, the FNM-based technique would be a good choice.

From the simulation results in the noisy scenario, we also observe that FNM-based algorithms perform well (see Fig. 2.10 and Fig. 2.11). In this experiment, we compute the MSE of LRMC algorithms against the rank of the original low-rank matrix for different setting of SNR (i.e., SNR = 20 and 50 dB). We observe that in the low and mid SNR regime (e.g., SNR = 20 dB), TNNR-ADMM performs comparable to the NNM-based algorithms since the FNM-based cost function is robust to the noise. In the high SNR regime (e.g., SNR = 50 dB), the convex algorithm (NNM with SDPT3) exhibits the best performance in term of the MSE. The performance of TNNR-ADMM is notably better than that of the rest of LRMC algorithms. For example, given $\text{rank}(\mathbf{M}) = 20$, the MSE of TNNR-ADMM is around 0.04, while the MSE of the rest is higher than $1$.

Finally, we apply LRMC techniques to restore images corrupted by impulse noise. In this experiment, we use $256 \times 256$ standard grayscale images (e.g., boat, cameraman, lena, and pepper images) and the salt-and-pepper noise model with different noise density $\rho = 0.3, 0.5,$ and $0.7$. For the FNM-based LRMC techniques, the rank is given by the number of the singular values $\sigma_i$ being greater than a relative threshold $\epsilon > 0$, i.e., $\sigma_i > \epsilon \max_i \sigma_i$. From the simulation results, we observe that peak SNR (pSNR), defined as the ratio of the maximum pixel value of the image to noise variance, of all LRMC techniques is at least $52$dB when $\rho = 0.3$ (see Table 2.8). In particular, NNM using SDPT3, SVT, and IRLS-M outperform the rest, achieving pSNR$\geq 57$ dB even with high noise level $\rho = 0.7$.

Table 2.7: MSE results for different problem sizes where rank$(\mathbf{M}) = 5$, and $p = 2 \times$ DOF

| | $n_1 = n_2 = 50$ | | | $n_1 = n_2 = 500$ | | | $n_1 = n_2 = 1000$ | | |
|---|---|---|---|---|---|---|---|---|---|
| | MSE | Time (s) | Iter. | MSE | Time (s) | Iter. | MSE | Time (s) | Iter. |
| NNM using SDPT3 | 0.0072 | 0.6 | 13 | 0.0017 | 74 | 16 | 0.0010 | 354 | 16 |
| SVT | 0.0154 | 0.4 | 300 | 0.4564 | 10 | 300 | 0.2110 | 32 | 300 |
| NIHT | 0.0008 | 0.2 | 253 | 0.0039 | 21 | 300 | 0.0019 | 93 | 300 |
| IRLS-M | 0.0009 | 0.2 | 60 | 0.0033 | 2 | 60 | 0.0025 | 8 | 60 |
| ADMiRA | 0.0075 | 0.3 | 300 | 0.0029 | 49 | 300 | 0.0016 | 52 | 300 |
| ASD | 0.0003 | $10^{-2}$ | 227 | 0.0006 | 2 | 300 | 0.0005 | 8 | 300 |
| LMaFit | 0.0002 | $10^{-2}$ | 241 | 0.0002 | 0.5 | 300 | 0.0500 | 1 | 300 |
| SET | 0.0678 | 11 | 9 | 0.0260 | 136 | 8 | 0.0108 | 270 | 8 |
| LRGeomCG | 0.0287 | 0.1 | 108 | 0.0333 | 12 | 300 | 0.0165 | 40 | 300 |
| TNNR-ADMM | 0.0221 | 0.3 | 300 | 0.0042 | 22 | 300 | 0.0021 | 94 | 300 |
| TNNR-APGL | 0.0055 | 0.3 | 300 | 0.0011 | 21 | 300 | 0.0009 | 95 | 300 |

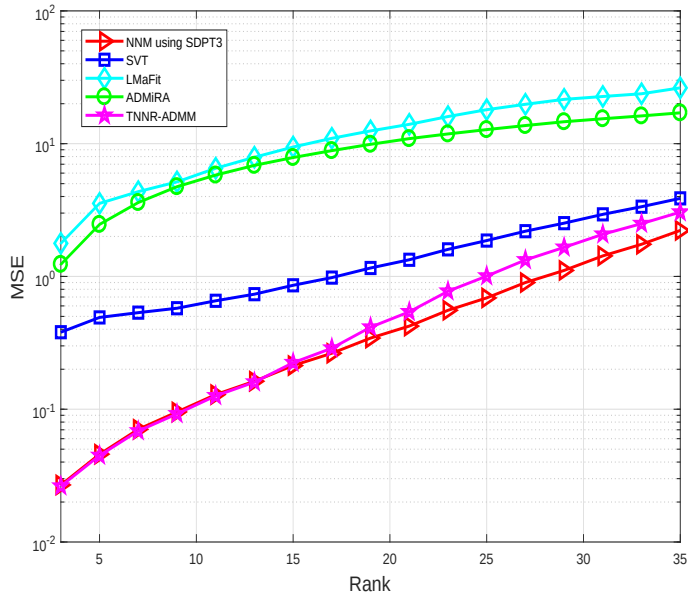



Figure 2.10: MSE performance of LRMC algorithms in noisy scenario with SNR = 20 dB (70% of entries are observed).

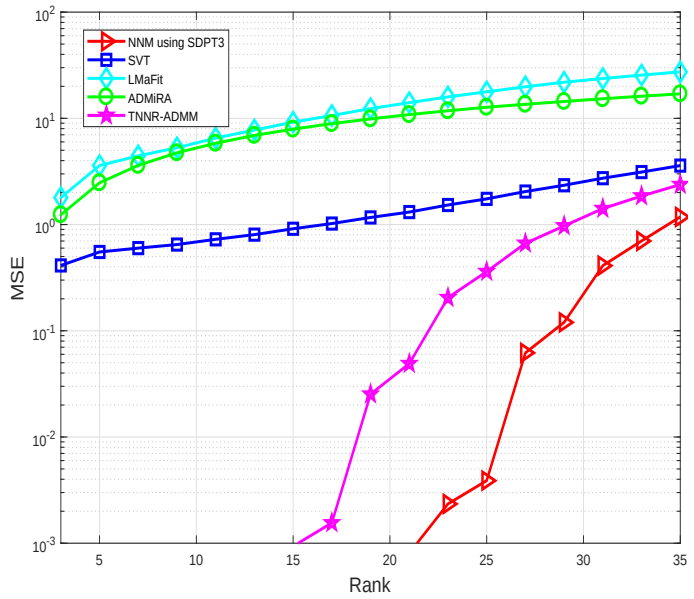



Figure 2.11: MSE performance of LRMC algorithms in noisy scenario with SNR = 50 dB (70% of entries are observed).

Table 2.8: Image recovery via LRMC for different noise levels $\rho$.

| | $\rho = 0.3$ | | | $\rho = 0.5$ | | | $\rho = 0.7$ | | |
|---|---|---|---|---|---|---|---|---|---|
| | pSNR (dB) | Time (s) | Iter. | pSNR (dB) | Time (s) | Iter. | pSNR (dB) | Time (s) | Iter. |
| NNM using SDPT3 | 66 | 1801 | 14 | 59 | 883 | 14 | 58 | 292 | 15 |
| SVT | 61 | 18 | 300 | 59 | 13 | 300 | 57 | 32 | 300 |
| NIHT | 58 | 16 | 300 | 54 | 6 | 154 | 53 | 2 | 35 |
| IRLS-M | 68 | 87 | 60 | 63 | 43 | 60 | 59 | 15 | 60 |
| ADMiRA | 57 | 1391 | 300 | 54 | 423 | 245 | 53 | 210 | 66 |
| ASD | 57 | 3 | 300 | 55 | 4 | 300 | 53 | 2 | 101 |
| LMaFit | 58 | 2 | 300 | 55 | 1 | 123 | 53 | 0.3 | 34 |
| SET | 61 | 716 | 6 | 55 | 321 | 4 | 53 | 5 | 2 |
| LRGeomCG | 52 | 47 | 300 | 48 | 18 | 75 | 44 | 5 | 21 |
| TNNR-ADMM | 57 | 15 | 300 | 54 | 18 | 300 | 53 | 18 | 300 |
| TNNR-APGL | 56 | 14 | 300 | 56 | 19 | 300 | 53 | 17 | 300 |

## 2.8 Choosing the Right Matrix Completion Algorithms

So far, we discussed various LRMC algorithms. A natural question one can ask is what algorithm should one choose? While this question is difficult to answer, one can consider NNM-based techniques including SVT and convex optimization with global convergence and exact recovery guarantee in the scenario when the rank of the original matrix is unknown. When the rank is known and the exact recovery of the original low-rank matrix is unnecessary, FNM-based techniques such as ADMiRA, LMaFit, and LRGeomCG can also be used since they have a fast convergence.

The other point that one should consider in the choice of LRMC algorithms is computational complexity. In the era of big data, the dimension of a matrix to be completed is large (in the order of hundred or thousand) so that the computational complexity is a big concern. Several options including SVT, NIHT, TNNR-APGL, TNNR-ADMM, ASD, and SET can be applied for large-scaled problems since their computational complexity is in order of the square of the dimension. The running time of LRGeomCG is proportional to the number of the observed entries. LMaFit might be the fastest matrix completion algorithm since it only requires to solve a system of linear equations. In general, there is a trade-off between the running time and the recovery performance. For example, gradient-based algorithms might converge to local optimum but with low computational complexity, whereas convex solvers always guarantees to find the global optimal solutions but might have a computational burden with the large problem size.

# Chapter 3

# IoT Localization Via LRMC

Localization refers to the problem to recovery the sensor locations (a.k.a. sensor map) in IoT. To solve this problem, a popular approach is to let each sensor node measure the pairwise distances with its neighboring nodes using various measurement techniques such as received signal strength indication (RSSI) [75], time of arrival (ToA) [76], time difference of arrival (TDoA) [77], and angle of arrival (AoA) [2]. The pairwise distances are then collected at a data center (basestation) and can be represented in the Euclidean distance matrix $\mathbf{D}$. Using the whole matrix $\mathbf{D}$, one can easily reconstructed the sensor location matrix $\mathbf{X}$ using multidimensional scaling method, each row the coordinate vector of each sensor node [78, 79].

However, in practice, the data center might not have enough distance information to identify the locations of sensor nodes. This is due to various reason such as the power outage of a sensor node or the limitation of radio communication range, only small number of distance information is available at the data center. Also, in the vehicular networks it might not be possible to measure the distance of all adjacent vehicles when a vehicle is located at the dead zone. Similar behavior can also be observed in underwater acoustic communication environments. In our approach, we propose a Euclidean distance matrix completion technique for the IoT network localization. We first express the Euclidean distance matrix $\mathbf{D}$ as a function of the low rank positive

semidefinite (PSD) matrix. Since the set of these matrices forms a Riemannian manifold in which the notation of differentiability can be defined, we can recycle, after a proper modification, an algorithm in the Euclidean space. Then to solve the problem, we propose a modified conjugate gradient algorithm, referred to as localization in Riemannian manifold using conjugate gradient (LRM-CG).

In this chapter, we first present our main FNM-based problem model in localization and useful notations used in Riemannian optimization (e.g., tangent space, Riemannian gradient, and retraction operation). Then we show the proposed LRM-CG algorithm as well as its computational complexity. Finally, we discuss on the performance guarantee of LRM-CG under extended Wolfe's conditions.

## 3.1 Problem Model

In the problem model, we suppose there is $n$ sensor nodes scattered in 2 or 3-dimensional Euclidean spaces ($k = 2$ or 3). The coordinate vectors of sensor nodes are represented as the rows of the location matrix $\mathbf{X} \in \mathbb{R}^{n\times k}$. Also, we recall that the squared pairwise distances $d_{ij}^2$ are the entries of the Euclidean distance matrix $\mathbf{D}$. From the definition of pairwise distance $d_{ij}^2 = \|\mathbf{x}_i - \mathbf{x}_j\|_2^2 = \mathbf{x}_i^T\mathbf{x}_i + \mathbf{x}_j^T\mathbf{x}_j - 2\mathbf{x}_i^T\mathbf{x}_j$, we have

$$\mathbf{D} = g(\mathbf{X}\mathbf{X}^T), \tag{3.1}$$

where $g(\mathbf{X}\mathbf{X}^T) = 2\text{Sym}(\text{diag}(\mathbf{X}\mathbf{X}^T)\mathbf{1}^T - \mathbf{X}\mathbf{X}^T)$. For example, as shown in Fig. 3.1, we have

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \mathbf{x}_3 & \mathbf{x}_4 & \mathbf{x}_5 \end{bmatrix}^T = \begin{bmatrix} 7 & 2 & 11 & 12 & 15 \\ 9 & 7 & 7 & 4 & 6 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix}^T,$$

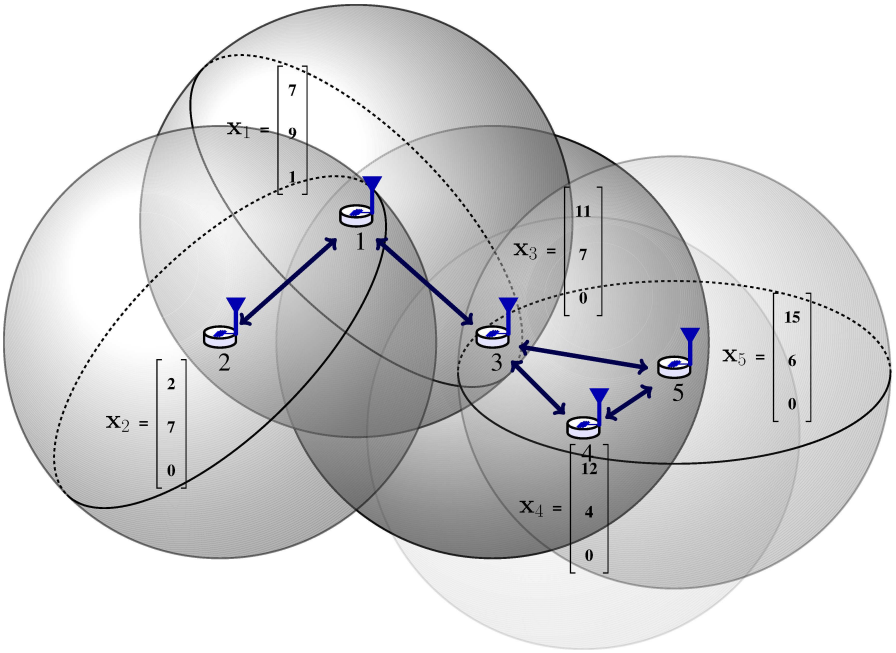



Figure 3.1: Sensor nodes deployed to measure not only environment information but also their pairwise distances. The observed distances are represented by two-sided arrows. The shadow spheres represent the radio communication range of the sensor nodes.

and

$$\mathbf{D} = g(\mathbf{X}\mathbf{X}^T) = \begin{bmatrix} 0 & 30 & 21 & 51 & 74 \\ 30 & 0 & 81 & 109 & 170 \\ 21 & 81 & 0 & 10 & 17 \\ 51 & 109 & 10 & 0 & 13 \\ 74 & 170 & 17 & 13 & 0 \end{bmatrix}.$$

The next lemma follows immediately from this.

**Lemma 5** *If $n$ sensor nodes are distributed in $k$-dimensional Euclidean space and $n \geq k$, then rank$(\mathbf{D}) \leq k + 2$.*

**Proof:** From (3.1), we have rank$(\mathbf{D})$ = rank$(g(\mathbf{X}\mathbf{X}^T))$, which gives

$$\begin{aligned} \text{rank}(\mathbf{D}) &= \text{rank}(2\text{Sym}(\text{diag}(\mathbf{X}\mathbf{X}^T)\mathbf{1}^T - \mathbf{X}\mathbf{X}^T)) \\ &\overset{(a)}{\leq} \text{rank}(2\text{Sym}(\text{diag}(\mathbf{X}\mathbf{X}^T)\mathbf{1}^T)) + \text{rank}(\mathbf{X}\mathbf{X}^T) \\ &= \text{rank}(\text{diag}(\mathbf{X}\mathbf{X}^T)\mathbf{1}^T + \mathbf{1}\text{diag}(\mathbf{X}\mathbf{X}^T)^T) + \text{rank}(\mathbf{X}\mathbf{X}^T) \\ &\overset{(b)}{\leq} 2 + \text{rank}(\mathbf{X}\mathbf{X}^T) \\ &\overset{(c)}{\leq} 2 + k, \end{aligned}$$

where (a) is because rank$(\mathbf{A} + \mathbf{B}) \leq$ rank$(\mathbf{A})$ + rank$(\mathbf{B})$ and Sym$(\mathbf{X}\mathbf{X}^T) = \mathbf{X}\mathbf{X}^T$, (b) is because rank$(\text{diag}(\mathbf{X}\mathbf{X}^T)\mathbf{1}^T) =$ rank$(\mathbf{1}\text{diag}(\mathbf{X}\mathbf{X}^T)^T) \leq 1$, and (c) is because rank$(\mathbf{X}\mathbf{X}^T) =$ rank$(\mathbf{X}) \leq k$ and rank$(\mathbf{a}\mathbf{b}^T) \leq 1$ for any two vectors $\mathbf{a}$ and $\mathbf{b}$. □

Using Lemma 5, we reformulate the matrix completion problem as

$$\begin{aligned} \min_{\widetilde{\mathbf{D}} \in \mathbb{R}^{n \times n}} \quad & \tfrac{1}{2}\|\mathcal{P}_E(\widetilde{\mathbf{D}}) - \mathcal{P}_E(\mathbf{D}_{obs})\|_F^2, \\ \text{s.t.} \quad & \text{rank}(\widetilde{\mathbf{D}}) \leq k + 2. \end{aligned} \tag{3.2}$$

Also, to account for the measurement accuracy, we can incorporate a weight matrix

$\mathbf{W}$ into (3.2) as[1]

$$\begin{array}{cc} \min\limits_{\widetilde{\mathbf{D}} \in \mathbb{R}^{n\times n}} & \frac{1}{2}\|\mathbf{W} \circ (\mathcal{P}_E(\widetilde{\mathbf{D}}) - \mathcal{P}_E(\mathbf{D}_{obs}))\|_F^2, \\ \text{s.t.} & \text{rank}(\widetilde{\mathbf{D}}) \leq k+2, \end{array} \tag{3.3}$$

where entries $w_{ij}$ of $\mathbf{W}$ satisfy $w_{ij}>0$ for $(i,j) \in E$, and zero otherwise.

Since $\text{rank}(\mathbf{D}) = \text{rank}(g(\widetilde{\mathbf{X}}\widetilde{\mathbf{X}}^T)) \leq k+2$ for any $\widetilde{\mathbf{X}}$, we can further simplify the problem as

$$\min_{\widetilde{\mathbf{X}} \in \mathbb{R}^{n\times k}} \quad \frac{1}{2}\|\mathbf{W} \circ (\mathcal{P}_E(g(\widetilde{\mathbf{X}}\widetilde{\mathbf{X}}^T)) - \mathcal{P}_E(\mathbf{D}_{obs}))\|_F^2. \tag{3.4}$$

Recalling that $\mathbf{Y} = \widetilde{\mathbf{X}}\widetilde{\mathbf{X}}^T$, we have

$$\min_{\mathbf{Y} \in \mathcal{Y}} \quad \frac{1}{2}\|\mathbf{W} \circ (\mathcal{P}_E(g(\mathbf{Y})) - \mathcal{P}_E(\mathbf{D}_{obs}))\|_F^2, \tag{3.5}$$

where $\mathcal{Y} = \{\widetilde{\mathbf{X}}\widetilde{\mathbf{X}}^T : \widetilde{\mathbf{X}} \in \mathbb{R}^{n\times k}\}$. We note that the purpose of the variable change is to simplify the problem. To be specific, let $\mathbf{X}^*$ be the solution to (3.4), then so are $\mathbf{X}^*\mathbf{F}$ for all orthonormal matrix $\mathbf{F} \in \mathbb{R}^{k\times k}$ since $\mathbf{X}^*\mathbf{F}\mathbf{F}^T\mathbf{X}^{*T} = \mathbf{X}^*\mathbf{X}^{*T}$. To avoid this confusion, we focus on the problem (3.5) instead of (3.4). We note also that many of our works in the next sections can be easily extended to (3.4) using $d\mathbf{Y} = d\widetilde{\mathbf{X}}\widetilde{\mathbf{X}}^T + \widetilde{\mathbf{X}}d\widetilde{\mathbf{X}}^T$ where $d\mathbf{Y}$ and $d\widetilde{\mathbf{X}}$ are the total differentials of $\mathbf{Y}$ and $\widetilde{\mathbf{X}}$, respectively.

When the sensor nodes are randomly distributed in $k$-dimensional Euclidean space, we can show that rank of the location matrix $\mathbf{X}$ is $k$ almost surely. In fact, consider the case that sensor nodes are randomly distributed in 2D Euclidean space, then $\text{rank}(\mathbf{X}) = 1$ if and only if all of nodes are co-linear. This event happens if there exists a constant $\rho$ such that $x_{i1} = \rho x_{i2}$ for any $i$-th row. The probability of this

[1]Note that when the observed entries are accurate, $w_{ij}$ should be the same constant for all $(i,j) \in E$, say, $w_{ij} = 1$. In many practical scenarios where range-based techniques are employed, the measurement accuracy might be proportional to the magnitude of the observed distances [80], which can be reflected in the choice of $w_{ij}$. For example, the smaller the observed distance $d_{ij}$ is, the larger the corresponding weight $w_{ij}$ would be.

event $\prod_{i=1}^{n} P(x_{i1} = \rho x_{i2}) = [P(x_{11} = \rho x_{12})]^n$ is negligible when the number of sensor nodes are sufficiently large. Thus, we can strengthen the constraint set from $\mathcal{Y}$ to $\widetilde{\mathcal{Y}} = \{\widetilde{\mathbf{X}}\widetilde{\mathbf{X}}^T : \widetilde{\mathbf{X}} \in \mathbb{R}^{n\times k}, \text{ rank}(\widetilde{\mathbf{X}}) = k\}$, and thus

$$\min_{\mathbf{Y}\in\widetilde{\mathcal{Y}}} \quad \tfrac{1}{2}\|\mathbf{W} \circ (\mathcal{P}_E(g(\mathbf{Y})) - \mathcal{P}_E(\mathbf{D}_{obs}))\|_F^2, \tag{3.6}$$

In the sequel, we denote $f(\mathbf{Y}) = \frac{1}{2}\|\mathbf{W} \circ (\mathcal{P}_E(g(\mathbf{Y})) - \mathcal{P}_E(\mathbf{D}_{obs}))\|_F^2$ for notational simplicity.

## 3.2 Optimization over Riemannian Manifold

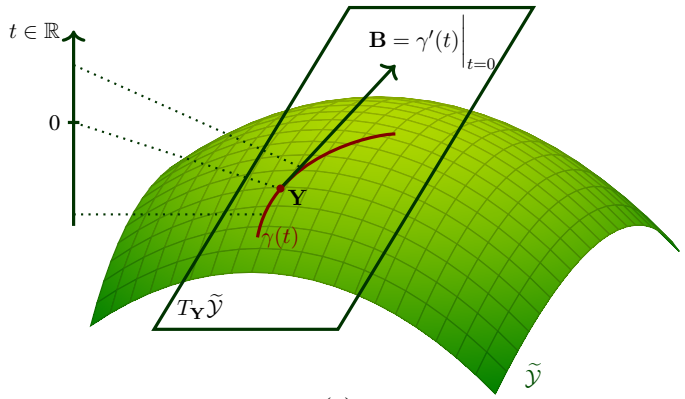
For a given $\mathbf{Y} \in \widetilde{\mathcal{Y}}$, one can take its eigendecomposition to obtain $\mathbf{Y} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T$ and thus comes up with an alternative representation of $\widetilde{\mathcal{Y}}$ as

$$\widetilde{\mathcal{Y}} = \{\mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T : \mathbf{Q} \in \mathcal{S}, \mathbf{\Lambda} \in \mathcal{L}\}, \tag{3.7}$$

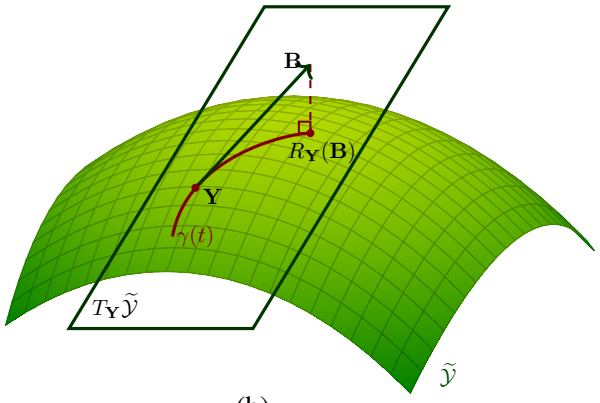
where $\mathcal{S} = \{\mathbf{Q} \in \mathbb{R}^{n\times k} : \mathbf{Q}^T\mathbf{Q} = \mathbf{I}_k\}$[2] and $\mathcal{L} = \{\text{eye}([\lambda_1 \quad \cdots \quad \lambda_k]^T) : \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_k > 0\}$. It has been shown that $\widetilde{\mathcal{Y}}$ is a smooth Riemannian manifold [81, Ch.5] [19]. Our approach to solve the problem in a smooth Riemannian manifold is beneficial in two major respects: First, one can easily compute the gradient of the cost function in (3.6) using the matrix calculus. Second, we can recycle an algorithm in the Euclidean space to solve the problem (3.6).

Since our work relies to a large extent on properties and operators of differential geometry, we briefly introduce tools and ingredients to describe the proposed algorithm. Since $\widetilde{\mathcal{Y}}$ is an embedded manifold in the Euclidean space $\mathbb{R}^{n\times n}$, its tangent spaces are determined by the derivative of its curves, where the curve $\gamma$ of $\widetilde{\mathcal{Y}}$ is a mapping from $\mathbb{R}$ to $\widetilde{\mathcal{Y}}$. Note that the tangent space on the smooth Riemannian manifold is a generalization of the notion of tangent hyperplane to surfaces in Euclidean space. Put it formally, for a given point $\mathbf{Y} \in \widetilde{\mathcal{Y}}$, the tangent space of $\widetilde{\mathcal{Y}}$ at $\mathbf{Y}$, denoted $T_{\mathbf{Y}}\widetilde{\mathcal{Y}}$, is de-

[2] $\mathcal{S}$ is an orthogonal Stiefel manifold embedded in $\mathbb{R}^{n\times k}$ [64, Theorem 5.12].




(a)




(b)

Figure 3.2: Illustration of (a) the tangent space $T_{\mathbf{Y}}\widetilde{\mathcal{Y}}$ and (b) the retraction operator $R_{\mathbf{Y}}$ at a point $\mathbf{Y}$ in the embedded manifold $\widetilde{\mathcal{Y}}$.

fined as $T_{\mathbf{Y}}\widetilde{\mathcal{Y}} = \{\gamma'(0) : \gamma \text{ is a curve in } \widetilde{\mathcal{Y}}, \gamma(0) = \mathbf{Y}\}$ (see Fig. 3.2). In the following lemma, we characterize the tangent space of $\widetilde{\mathcal{Y}}$.

**Lemma 6** *For the manifold $\widetilde{\mathcal{Y}}$ defined by (3.7), the tangent space $T_{\mathbf{Y}}\widetilde{\mathcal{Y}}$ at $\mathbf{Y}$ is*

$$T_{\mathbf{Y}}\widetilde{\mathcal{Y}} = \left\{ \begin{bmatrix} \mathbf{Q} & \mathbf{Q}_\perp \end{bmatrix} \begin{bmatrix} \mathbf{B} & \mathbf{C}^T \\ \mathbf{C} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{Q}^T \\ \mathbf{Q}_\perp^T \end{bmatrix} : \mathbf{B}^T = \mathbf{B} \in \mathbb{R}^{k\times k}, \mathbf{C} \in \mathbb{R}^{(n-k)\times k} \right\}. \tag{3.8}$$

**Proof:** See Appendix A. □

A metric on the tangent space $T_{\mathbf{Y}}\widetilde{\mathcal{Y}}$ is defined as the matrix inner product $< \mathbf{B}_1, \mathbf{B}_2 >= \text{tr}(\mathbf{B}_1^T\mathbf{B}_2)$ between two tangent vectors $\mathbf{B}_1, \mathbf{B}_2 \in T_{\mathbf{Y}}\widetilde{\mathcal{Y}}$. We next define the orthogonal projection of a matrix $\mathbf{A}$ onto the tangent space $T_{\mathbf{Y}}\widetilde{\mathcal{Y}}$, which will be used to find the closed-form expression of Riemannian gradient in Subsection 3.3.

**Definition 1** *The orthogonal projection onto $T_{\mathbf{Y}}\widetilde{\mathcal{Y}}$ is a mapping $P_{T_{\mathbf{Y}}\widetilde{\mathcal{Y}}} : \mathbb{R}^{n\times n} \to T_{\mathbf{Y}}\widetilde{\mathcal{Y}}$ such that for a given matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$, $< \mathbf{A} - P_{T_{\mathbf{Y}}\widetilde{\mathcal{Y}}}(\mathbf{A}), \mathbf{B} >= 0$ for all $\mathbf{B} \in T_{\mathbf{Y}}\widetilde{\mathcal{Y}}$.*

The following theorem provides a closed form expression of the orthogonal projection operator.

**Theorem 7** *For a given matrix $\mathbf{A}$, orthogonal projection $P_{T_{\mathbf{Y}}\widetilde{\mathcal{Y}}}(\mathbf{A})$ of $\mathbf{A}$ onto the tangent space $T_{\mathbf{Y}}\widetilde{\mathcal{Y}}$ is*

$$P_{T_{\mathbf{Y}}\widetilde{\mathcal{Y}}}(\mathbf{A}) = \mathbf{P}_{\mathbf{Q}}\mathit{Sym}(\mathbf{A}) + \mathit{Sym}(\mathbf{A})\mathbf{P}_{\mathbf{Q}} - \mathbf{P}_{\mathbf{Q}}\mathit{Sym}(\mathbf{A})\mathbf{P}_{\mathbf{Q}}, \tag{3.9}$$

*where $\mathbf{P}_{\mathbf{Q}} = \mathbf{Q}\mathbf{Q}^T$.*

**Proof:** See Appendix B. □

In order to express the concept of moving in the direction of a tangent space while staying on the manifold, an operation called *retraction* is used. As illustrated in Fig.

3.2(b), the retraction operation is a mapping from $T_{\mathbf{Y}}\widetilde{\mathcal{Y}}$ to $\widetilde{\mathcal{Y}}$ that preserves the gradient at $\mathbf{Y}$ [82, Definition 4.1.1].

**Definition 2** *The retraction $R_{\mathbf{Y}}(\mathbf{B})$ of a vector $\mathbf{B} \in T_{\mathbf{Y}}\widetilde{\mathcal{Y}}$ onto $\widetilde{\mathcal{Y}}$ is defined as*

$$R_{\mathbf{Y}}(\mathbf{B}) = arg\min_{\mathbf{Z}\in\widetilde{\mathcal{Y}}} \|\mathbf{Y}+\mathbf{B}-\mathbf{Z}\|_F. \tag{3.10}$$

In obtaining the closed form expression of $R_{\mathbf{Y}}(\mathbf{B})$, an operator $\mathcal{W}_k$ keeping $k$ largest positive eigenvalues of a matrix, referred to as *eigenvalue selection operator*, is needed. Since the projection $R_{\mathbf{Y}}(\mathbf{B})$ is an element of $\widetilde{\mathcal{Y}}$, $R_{\mathbf{Y}}(\mathbf{B})$ should be a symmetric PSD matrix with rank $k$. Thus, for given square matrix $\mathbf{A}$, we are interested only in the symmetric part $Sym(\mathbf{A})$. If we denote the eigenvalue decomposition (EVD) of this as $Sym(\mathbf{A}) = \mathbf{P}\mathbf{\Sigma}\mathbf{P}^T$ and the $k$ topmost eigenvalues of this as $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_k$ , then $\mathcal{W}_k(\mathbf{A})$ is defined as

$$\mathcal{W}_k(\mathbf{A}) = \mathbf{P}\mathbf{\Sigma}_k\mathbf{P}^T, \tag{3.11}$$

where $\mathbf{\Sigma}_k = \text{eye}\left(\begin{bmatrix} \sigma_1 & \dots & \sigma_k & 0 & \dots & 0 \end{bmatrix}^T\right)$. We can obtain an elegant expression of $R_{\mathbf{Y}}(\mathbf{B})$ using the eigenvalue selection operator $\mathcal{W}_k$.

**Theorem 8** *The retraction $R_{\mathbf{Y}}(\mathbf{B})$ of a vector $\mathbf{B} \in T_{\mathbf{Y}}\widetilde{\mathcal{Y}}$ can be expressed as*

$$R_{\mathbf{Y}}(\mathbf{B}) = \mathcal{W}_k(\mathbf{Y}+\mathbf{B}). \tag{3.12}$$

**Proof:** Our goal is to find a simple expression of the retraction operator $R_{\mathbf{Y}}(\mathbf{B})$. First, since $\mathbf{Z} = \text{Sym}(\mathbf{Z})$ for $\mathbf{Z} \in \widetilde{\mathcal{Y}}$, we have

$$\begin{aligned}
\|\mathbf{Y}+\mathbf{B}-\mathbf{Z}\|_F^2 &= \|\mathbf{Y}+\mathbf{B}-\text{Sym}(\mathbf{Z})\|_F^2 \\
&\overset{(a)}{=} \|\text{Skew}(\mathbf{Y}+\mathbf{B}) + \text{Sym}(\mathbf{Y}+\mathbf{B}) - \text{Sym}(\mathbf{Z})\|_F^2 \\
&\overset{(b)}{=} \|\text{Skew}(\mathbf{Y}+\mathbf{B})\|_F^2 + \|\text{Sym}(\mathbf{Y}+\mathbf{B}) - \mathbf{Z}\|_F^2,
\end{aligned}$$

where (a) is because $\text{Sym}(\mathbf{A})+\text{Skew}(\mathbf{A}) = \mathbf{A}$ and (b) is because $< \text{Skew}(\mathbf{C}), \text{Sym}(\mathbf{D}) >= 0$ for any $\mathbf{C}$ and $\mathbf{D}$. Since $\|\text{Skew}(\mathbf{Y}+\mathbf{B})\|_F^2$ is unrelated to $\mathbf{Z}$, it is clear that

$$R_{\mathbf{Y}}(\mathbf{B}) = \arg\min_{\mathbf{Z}\in\widetilde{\mathcal{Y}}} \|\text{Sym}(\mathbf{Y}+\mathbf{B}) - \mathbf{Z}\|_F.$$

Using the eigenvalue decomposition $\text{Sym}(\mathbf{Y}+\mathbf{B}) = \mathbf{K}\mathbf{\Sigma}\mathbf{K}^T$, we have

$$\begin{aligned} R_{\mathbf{Y}}(\mathbf{B}) &= \arg\min_{\mathbf{Z}\in\widetilde{\mathcal{Y}}} \|\mathbf{K}\mathbf{\Sigma}\mathbf{K}^T - \mathbf{Z}\|_F \\ &= \arg\min_{\mathbf{Z}\in\widetilde{\mathcal{Y}}} \|\mathbf{K}\left(\mathbf{\Sigma} - \mathbf{K}^T\mathbf{Z}\mathbf{K}\right)\mathbf{K}^T\|_F \\ &\overset{(a)}{=} \arg\min_{\mathbf{Z}\in\widetilde{\mathcal{Y}}} \|\mathbf{\Sigma} - \mathbf{K}^T\mathbf{Z}\mathbf{K}\|_F, \end{aligned}$$

where (a) is because $\|\mathbf{K}\mathbf{U}\mathbf{K}^T\|_F^2 = tr(\mathbf{K}\mathbf{U}^T\mathbf{K}^T\mathbf{K}\mathbf{U}\mathbf{K}^T) = \|\mathbf{U}\|_F^2$ for any matrix $\mathbf{U}$. Now let $R_{\mathbf{Y}}(\mathbf{B}) = \mathbf{Z}^*$, $\mathbf{\Sigma}^* = \mathbf{K}\mathbf{Z}^*\mathbf{K}^T$, and $\mathbf{Q} = \mathbf{K}^T\mathbf{Z}\mathbf{K}$, then

$$\mathbf{\Sigma}^* = \mathbf{K}\mathbf{Z}^*\mathbf{K}^T = \arg\min_{\mathbf{Q}} \|\mathbf{\Sigma} - \mathbf{Q}\|_F. \tag{3.13}$$

Since $\mathbf{\Sigma}$ is a diagonal matrix, $\mathbf{\Sigma}^*$ should also be a diagonal matrix. Also, $\mathbf{\Sigma}^* \succeq 0$ and $\text{rank}(\mathbf{\Sigma}^*) = k$.[3] Thus, $\mathbf{\Sigma}^*$ is a diagonal matrix with only $k$ positive entries and the rest being zero. That is,

$$\mathbf{\Sigma}^* = \begin{bmatrix} \sigma_1 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & \sigma_k & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \end{bmatrix}, \tag{3.14}$$

where $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_k > 0$. Recalling that $\text{Sym}(\mathbf{Y}+\mathbf{B}) = \mathbf{K}\mathbf{\Sigma}\mathbf{K}^T$, we finally have

$$R_{\mathbf{Y}}(\mathbf{B}) = \mathbf{K}\mathbf{\Sigma}^*\mathbf{K}^T = \mathcal{W}_k(\mathbf{Y}+\mathbf{B}),$$

where the last equality is from (3.11). □

Finally, to develop the conjugate gradient algorithm over the Riemannian manifold $\widetilde{\mathcal{Y}}$, we need the Euclidean gradient of the cost function $f(\mathbf{Y})$.

[3] Since $\mathbf{Z}^* \in \widetilde{\mathcal{Y}}$, $\mathbf{Z}^* \succeq 0$ and also $\mathbf{\Sigma}^* = \mathbf{K}^T\mathbf{Z}^*\mathbf{K} \succeq 0$ and $\text{rank}(\mathbf{\Sigma}^*) = \text{rank}(\mathbf{K}^T\mathbf{Z}^*\mathbf{K}) = \text{rank}(\mathbf{Z}^*) = k$.

**Theorem 9** *Euclidean gradient* $\nabla_{\mathbf{Y}} f(\mathbf{Y})$ *of* $f(\mathbf{Y})$ *with respect to* $\mathbf{Y}$ *is*

$$\nabla_{\mathbf{Y}} f(\mathbf{Y}) = 2eye(Sym(\mathbf{R})\mathbf{1}) - 2Sym(\mathbf{R}), \tag{3.15}$$

*where* $\mathbf{R} = \mathbf{W} \circ \mathbf{W} \circ (\mathcal{P}_E(g(\mathbf{Y})) - \mathcal{P}_E(\mathbf{D}_{obs}))$.

**Proof:** See Appendix D. □

## 3.3 Localization in Riemannian Manifold Using Conjugate Gradient (LRM-CG)

In our approach, we solve the problem in (3.6) using the conjugate gradient (CG) method. Note that CG method is widely used to solve sparse symmetric positive definite linear systems [83]. In CG algorithms, the optimal value can be obtained in a finite number of searching steps. This is because the conjugate direction is designed such that it is conjugate to previous directions and also the gradient of the cost function. Also, the CG algorithm can be readily applied to a sparse symmetric positive definite system and thus can be used to solve our main problem due to the quadratic form of the cost function $f(\mathbf{Y})$. In fact, noting that $\mathcal{P}_E$ and $g$ are linear mappings, one can easily show that

$$\begin{aligned} f(\mathbf{Y}) &= \frac{1}{2}\|\mathbf{W} \circ (\mathcal{P}_E(g(\mathbf{Y})) - \mathcal{P}_E(\mathbf{D}_{obs}))\|_F^2 \\ &= \frac{1}{2}\|\mathbf{W} \circ \mathcal{P}_E(g(\sum_{i,j} y_{ij}\mathbf{E}^{(i,j)}) - \mathbf{D}_{obs})\|_F^2 \\ &\overset{(a)}{=} \frac{1}{2}\|\text{vec}(\mathbf{K}) \circ (\sum_{i,j} y_{ij}\text{vec}\left(\mathcal{P}_E(g(\mathbf{E}^{(i,j)}))\right) - \text{vec}(\mathcal{P}_E(\mathbf{D}_{obs})))\|_2^2 \\ &\overset{(b)}{=} \frac{1}{2}\|\mathbf{A}\text{vec}(\mathbf{Y}) - \mathbf{b}\|_2^2, \end{aligned} \tag{3.16}$$

where (a) is because $\|\mathbf{M}\|_F = \|\text{vec}(\mathbf{M})\|_2$ and (b) follows from

$$\text{vec}(\mathbf{Y}) = \begin{bmatrix} y_{11} & y_{21} & \cdots & y_{nn} \end{bmatrix}^T,\ \mathbf{b} = \text{vec}(\mathbf{W} \circ \mathcal{P}_E(\mathbf{D}_{obs})),$$

and $\mathbf{A}$ formed by column vectors vec $\big(\mathbf{W} \circ \mathcal{P}_E(g(\mathbf{E}^{(i,j)}))\big)$. From (3.16), it is obvious that $f(\mathbf{Y})$ has the quadratic form of a sparse symmetric positive definite system.

Recall that the update equation of the conventional CG algorithm in the Euclidean space is

$$\mathbf{Y}_{i+1} = \mathbf{Y}_i + \alpha_i \mathbf{P}_i, \tag{3.17}$$

where $\alpha_i$ is the stepsize and $\mathbf{P}_i$ is the conjugate direction. The stepsize $\alpha_i$ is chosen by the line minimization technique (e.g., Armijo's rule [84, 85]) and the search direction $\mathbf{P}_i$ of the CG algorithm is chosen as a linear combination of the gradient and the previous search direction to generate a direction conjugate to the previous ones. In doing so, one can avoid unnecessary searching of directions that have been searched over and thus achieve the speedup of the algorithm [86, 83].

In our work, since we consider the optimization problem over the Riemannian manifold $\widetilde{\mathcal{Y}}$, the conjugate direction $\mathbf{P}_i$ should lie on the tangent space. To make sure that the update point $\mathbf{Y}_{i+1}$ lies on the manifold, we need a retraction operation. The update equation after applying the retraction operation is

$$\begin{aligned} \mathbf{Y}_{i+1} &= R_{\mathbf{Y}_i}(\alpha_i \mathbf{P}_i) \\ &= \mathcal{W}_k(\mathbf{Y}_i + \alpha_i \mathbf{P}_i). \end{aligned} \tag{3.18}$$

As observed in Theorem 8, the eigenvalue selection operator $\mathcal{W}_k$ guarantees that the updated point $\mathbf{Y}_{i+1}$ lies on the manifold.

We next consider the conjugate direction $\mathbf{P}_i$ of LRM-CG. In the conventional non-linear CG algorithm, conjugate direction $\mathbf{P}_i$ is updated as

$$\mathbf{P}_i = -\nabla_{\mathbf{Y}} f(\mathbf{Y}_i) + \beta_i \mathbf{P}_{i-1}, \tag{3.19}$$

where $\beta_i$ is the conjugate update parameter[4]. Since we optimize over the Riemannian manifold $\widetilde{\mathcal{Y}}$, conjugate direction in (3.19) needs to be modified. First, we need to use the Riemannian gradient of $f(\mathbf{Y})$ instead of the Euclidean gradient $\nabla_{\mathbf{Y}} f(\mathbf{Y})$ since

[4]There are a number of ways to choose $\beta_i$. See, e.g., [83, 87, 88, 89].

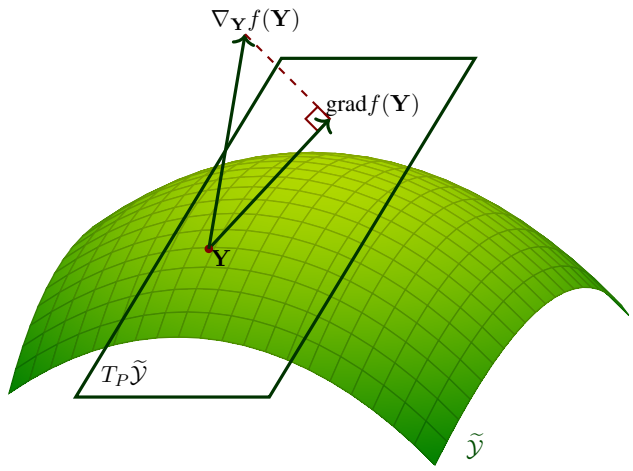



Figure 3.3: Riemannian gradient $\operatorname{grad} f(\mathbf{Y})$ is defined as the projection of the Euclidean gradient $\nabla_{\mathbf{Y}} f(\mathbf{Y})$ onto the tangent space $T_{\mathbf{Y}}\widetilde{\mathcal{Y}}$ while the Euclidean gradient is a direction for which the cost function is reduced most in $\mathbb{R}^{n\times n}$, Riemannian gradient is the direction for which the cost function is reduced most in the tangent space $T_{\mathbf{Y}}\widetilde{\mathcal{Y}}$.

we find the search direction on the tangent space of $\widetilde{\mathcal{Y}}$. Riemannian gradient, denoted grad$f(\mathbf{Y})$, is distinct from $\nabla_{\mathbf{Y}} f(\mathbf{Y})$ in the sense that it is defined on the tangent space $T_{\mathbf{Y}}\widetilde{\mathcal{Y}}$ (see Fig. 3.3).

**Definition 3** *Let $f$ be the function differentiable everywhere in the Riemannian manifold $\widetilde{\mathcal{Y}}$. The Riemannian gradient grad$f(\mathbf{Y})$ of $f$ at $\mathbf{Y}$ is defined as the unique element in $T_{\mathbf{Y}}\widetilde{\mathcal{Y}}$ satisfying*

$$< \mathbf{B}, grad f(\mathbf{Y}) >=< \mathbf{B}, \nabla_{\mathbf{Y}} f(\mathbf{Y}) >, \tag{3.20}$$

*where $\mathbf{B}$ is any element in $T_{\mathbf{Y}}\widetilde{\mathcal{Y}}$.*

As shown in Fig. 3.3, grad$f(\mathbf{Y}) \in T_{\mathbf{Y}}\widetilde{\mathcal{Y}}$ is a component of the Euclidean gradient $\nabla_{\mathbf{Y}} f(\mathbf{Y})$ in $T_{\mathbf{Y}}\widetilde{\mathcal{Y}}$. In other words, grad$f(\mathbf{Y})$ is the projection of $\nabla_{\mathbf{Y}} f(\mathbf{Y})$ onto $T_{\mathbf{Y}}\widetilde{\mathcal{Y}}$. Indeed, from Definition 1, $< \mathbf{B}, \nabla_{\mathbf{Y}} f(\mathbf{Y}) - P_{T_{\mathbf{Y}}\widetilde{\mathcal{Y}}}(\nabla_{\mathbf{Y}} f(\mathbf{Y})) >= 0$ for any matrix $\mathbf{B} \in T_{\mathbf{Y}}\widetilde{\mathcal{Y}}$. Hence,

$$< \mathbf{B}, P_{T_{\mathbf{Y}}\widetilde{\mathcal{Y}}}(\nabla_{\mathbf{Y}} f(\mathbf{Y})) >=< \mathbf{B}, \nabla_{\mathbf{Y}} f(\mathbf{Y}) > . \tag{3.21}$$

From (3.20) and (3.21), it is clear that

$$\text{grad} f(\mathbf{Y}) = P_{T_{\mathbf{Y}}\widetilde{\mathcal{Y}}}(\nabla_{\mathbf{Y}} f(\mathbf{Y})). \tag{3.22}$$

Second, since the Riemannian gradient grad$f(\mathbf{Y}_i)$ and previous conjugate direction $\mathbf{P}_{i-1}$ lie on two different vector spaces $T_{\mathbf{Y}_i}\widetilde{\mathcal{Y}}$ and $T_{\mathbf{Y}_{i-1}}\widetilde{\mathcal{Y}}$, we need to project $\mathbf{P}_{i-1}$ onto the tangent space $T_{\mathbf{Y}_i}\widetilde{\mathcal{Y}}$[5] before performing a linear combination between of two. In view of this, the conjugate direction update equation of LRM-CG is

$$\mathbf{P}_i = -\text{grad} f(\mathbf{Y}_i) + \beta_i P_{T_{\mathbf{Y}_i}\widetilde{\mathcal{Y}}}(\mathbf{P}_{i-1}). \tag{3.23}$$

[5]In transforming a vector from one tangent space to another, an operator called vector transport is used (see Definition 8.1.1 in [63]). For an embedded manifold of $\mathbb{R}^{n\times n}$, vector transport is the orthogonal projection operator [63]. Hence, the vector transport of $\mathbf{P}_{i-1}$ is the orthogonal projection of $\mathbf{P}_{i-1}$ onto $T_{\mathbf{Y}_i}\widetilde{\mathcal{Y}}$

**Algorithm 1:** LRM-CG algorithm

**Input:** $\mathbf{D}_{obs}$: the observed matrix,

$\mathbf{W}$: the weight matrix,

$\mathcal{P}_E$: the sampling operator,

$\epsilon$: tolerance,

$\mu \in (0 \quad 1)$: given constant,

$T$: number of iterations.

**Initialize:** $i = 1$,

$\mathbf{Y}_1 \in \widetilde{\mathcal{Y}}$: initial point,

$\mathbf{P}_1$: initial conjugate direction.

**While** $i \leq T$ **do**

$\mathbf{R}_i = \mathbf{W} \circ \mathbf{W} \circ (\mathcal{P}_E(g(\mathbf{Y}_i)) - \mathcal{P}_E(\mathbf{D}_{obs}))$

$\nabla_{\mathbf{Y}} f(\mathbf{Y}_i) = 2\text{eye}(\text{Sym}(\mathbf{R}_i)\mathbf{1}) - 2\mathbf{R}_i$

$\text{grad} f(\mathbf{Y}_i) = P_{T_{\mathbf{Y}_i}\widetilde{\mathcal{Y}}}(\nabla_{\mathbf{Y}} f(\mathbf{Y}_i))$

$\mathbf{H}_i = \text{grad} f(\mathbf{Y}_i) - P_{T_{\mathbf{Y}_i}\widetilde{\mathcal{Y}}}(\text{grad} f(\mathbf{Y}_{i-1}))$

$h =< \mathbf{P}_i, \mathbf{H}_i >$

$\beta_i = \frac{1}{h^2} < h\mathbf{H}_i - 2\mathbf{P}_i \|\mathbf{H}_i\|_F^2, \text{grad} f(\mathbf{Y}_i) >$

$\mathbf{P}_i = -\text{grad} f(\mathbf{Y}_i) + \beta_i P_{T_{\mathbf{Y}_i}\widetilde{\mathcal{Y}}}(\mathbf{P}_{i-1})$

Find a stepsize $\alpha_i > 0$ such that

$f(\mathbf{Y}_i) - f(R_{\mathbf{Y}_i}(\alpha_i \mathbf{P}_i)) \geq -\mu\alpha_i < \text{grad} f(\mathbf{Y}_i), \mathbf{P}_i >$

$\mathbf{Y}_{i+1} = R_{\mathbf{Y}_i}(\alpha_i \mathbf{P}_i)$

$\mathbf{D}_{i+1} = g(\mathbf{Y}_{i+1})$

**If** $\|\mathbf{W} \circ (\mathcal{P}_E(\mathbf{D}_{i+1}) - \mathcal{P}_E(\mathbf{D}_{obs}))\|_F < \epsilon$ **then**

Exit from while loop

**End If**

Obtain $\mathbf{Q}$ and $\mathbf{\Lambda}$ using the eigendecomposition

$\mathbf{Y}_{i+1} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T$

$\widehat{\mathbf{X}} = \mathbf{Q}\mathbf{\Lambda}^{1/2}$

$i = i + 1$

**End While**

**Output:** $\widehat{\mathbf{X}}$

Table 3.1: Computational complexity of LRM-CG for each iteration.

| Algorithm operation | Flops order |
|---|---|
| Euclidean gradient | $O(k\|E\| + n)$ |
| Orthogonal projection | $O(kn^2 + k^2n + k^3)$ |
| Retraction | $O(kn^2 + k^2n + k^3)$ |

Finally, in choosing the stepsize $\alpha_i$ in (3.18), we use the Armijo's rule, a widely used line search strategy. Note that the Armijo's rule is an effective way to find a stepsize $\alpha_i$ minimizing the cost function $f$, that is, $\alpha_i \approx \min_{\alpha>0} f(\mathcal{W}_k(\mathbf{Y}_i + \alpha_i \mathbf{P}_i)$ [84, 85].

The proposed LRM-CG algorithm is summarized in Algorithm 1.

## 3.4 Computational Complexity

In this subsection, we analyze the computational complexity of LRM-CG in terms of the number of floating point operations (flops). As discussed in Section 3.2, major steps in LRM-CG is to compute Euclidean gradient, Riemannian gradient, and the retraction operation.

In order to compute the Euclidean gradient $\nabla_{\mathbf{Y}} f(\mathbf{Y}_i)$ in (3.15), we need to consider the computation of $\mathbf{Y}_i$ from the $(i-1)$-th iteration. Since $\mathbf{Y}_i = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T$ ($\mathbf{Q}$ is a $n \times k$ matrix and $\mathbf{\Lambda}$ is a $k \times k$ diagonal matrix), it requires $2k$ multiplications and $(k-1)$ additions to compute $y_{ij} = \sum_{t=1}^{k} \lambda_t q_{it} q_{jt}$ so that the associated computational complexity is $(3k-1)$ flops. Further, from (3.1), we need to compute $[g(\mathbf{Y})]_{ij} = y_{ii} + y_{jj} - y_{ij}$, which requires $(9k-1)$ flops. The residual matrix $\mathbf{R}_i = \mathcal{P}_E(g(\mathbf{Y}_i)) - \mathbf{D}_{obs}$ requires $9k|E|$ flops ($|E|$ is the number of the observed entries of $\mathbf{D}_{obs}$). In addition, since it takes $(9k+2)|E|$ flops to compute $\text{Sym}(\mathbf{R}_i) = \frac{1}{2}(\mathbf{R}_i + \mathbf{R}_i^T)$, it requires at most $(9k+4)|E| + n - 1$ flops to compute $2\text{eye}(\text{Sym}(\mathbf{R}_i)\mathbf{1})$. Since the cardinality of $\text{Sym}(\mathbf{R}_i)$ is $|E|$, computational complexity of $\nabla_{\mathbf{Y}} f(\mathbf{Y}_i) = 2\text{eye}(\text{Sym}(\mathbf{R}_i)\mathbf{1}) - 2\text{Sym}(\mathbf{R}_i)$ in (3.15) is at most $(9k+5)|E| + n - 1$.

Second, recalling that the Riemannian gradient grad$f(\mathbf{Y}_i)$ is an orthogonal projection of $\nabla_{\mathbf{Y}} f(\mathbf{Y}_i)$ onto the tangent space $T_{\mathbf{Y}_i}\widetilde{\mathcal{Y}}$, we need to estimate the computational complexity of the orthogonal projection operator $P_{T_{\mathbf{Y}_i}\widetilde{\mathcal{Y}}}$. In computing $P_{T_{\mathbf{Y}_i}\widetilde{\mathcal{Y}}}(\mathbf{A})$ for an $n \times n$ matrix $\mathbf{A}$, we need Sym$(\mathbf{A})$, $\mathbf{B} = \text{Sym}(\mathbf{A})\mathbf{Q}$, and $\mathbf{C} = \mathbf{Q}^T\text{Sym}(\mathbf{A})\mathbf{Q}$, which require $(2k-1)n^2$, $2n^2 + (2n-1)kn$, and $(2n-1)kn + (2n-1)k^2$ flops, respectively. Then, from (3.9), we have

$$P_{T_{\mathbf{Y}_i}\widetilde{\mathcal{Y}}}(\mathbf{A}) \;=\; \mathbf{Q}\mathbf{B}^T + \mathbf{B}\mathbf{Q}^T - \mathbf{Q}\mathbf{C}\mathbf{Q}^T,$$

which requires $\mathcal{O}(kn^2 + k^2n + k^3)$ flops.

Finally, in applying Armijo's rule to find the stepsize $\alpha_i$, we need to compute the retraction operation $R_{\mathbf{Y}_i}(\alpha_i\mathbf{P}_i)$. From (3.12), the retraction operation is obtained via the eigenvalue selection operator $\mathcal{W}_k$ and this requires the EVD of $(\mathbf{Y}_i + \mathbf{P}_i)$. In general, computational complexity of the EVD for a $n \times n$ matrix is expressed as $\mathcal{O}(n^3)$. However, using $\mathbf{Y}_i = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T$ and $\mathbf{P}_i \in T_{\mathbf{Y}_i}\widetilde{\mathcal{Y}}$, we adopt the computational strategy in [20] to simplify the EVD operation. First, since $\mathbf{P}_i \in T_{\mathbf{Y}_i}\widetilde{\mathcal{Y}}$, we have

$$\mathbf{P}_i = \begin{bmatrix}\mathbf{Q} & \mathbf{Q}_\perp\end{bmatrix}\begin{bmatrix}\mathbf{B}_i & \mathbf{C}_i^T \\ \mathbf{C}_i & \mathbf{0}\end{bmatrix}\begin{bmatrix}\mathbf{Q}^T \\ \mathbf{Q}_\perp^T\end{bmatrix}. \tag{3.24}$$

Thus,

$$\begin{aligned}
\mathbf{Y}_i + \mathbf{P}_i &= \begin{bmatrix}\mathbf{Q} & \mathbf{Q}_\perp\end{bmatrix}\begin{bmatrix}\mathbf{B}_i + \mathbf{\Lambda} & \mathbf{C}_i^T \\ \mathbf{C}_i & \mathbf{0}\end{bmatrix}\begin{bmatrix}\mathbf{Q}^T \\ \mathbf{Q}_\perp^T\end{bmatrix} \\
&= \begin{bmatrix}\mathbf{Q} & \mathbf{Q}_c\end{bmatrix}\begin{bmatrix}\mathbf{B}_i + \mathbf{\Lambda} & \mathbf{R}_c^T \\ \mathbf{R}_c & \mathbf{0}\end{bmatrix}\begin{bmatrix}\mathbf{Q}^T \\ \mathbf{Q}_c^T\end{bmatrix} \\
&= \begin{bmatrix}\mathbf{Q} & \mathbf{Q}_c\end{bmatrix}\mathbf{K}\mathbf{\Lambda}'\mathbf{K}^T\begin{bmatrix}\mathbf{Q}^T \\ \mathbf{Q}_c^T\end{bmatrix}.
\end{aligned}$$

where $\mathbf{Q}_c\mathbf{R}_c$ is the QR-decomposition of $\mathbf{Q}_\perp\mathbf{C}_i$, which requires $(2n-2k-1)kn + \mathcal{O}(nk^2)$ flops. Now the EVD of $(\mathbf{Y}_i + \mathbf{P}_i)$ is simplified to the EVD of the $2k \times 2k$ matrix $\begin{bmatrix}\mathbf{B}_i + \mathbf{\Lambda} & \mathbf{R}_c^T \\ \mathbf{R}_c & \mathbf{0}\end{bmatrix}$, which requires only $\mathcal{O}(k^3)$ flops. Also, computing $\begin{bmatrix}\mathbf{Q} & \mathbf{Q}_c\end{bmatrix}\mathbf{K}$

needs $(4k-1)kn$ flops. Thus, computational complexity of the retraction operation is $\mathcal{O}(kn^2 + nk^2 + k^3)$, which is $\mathcal{O}(kn^2)$ for $k \ll n$. The computational complexity of Euclidean gradient, Riemannian gradient, and the retraction in each iteration is summarized in Table 3.1.

In summary, computational complexity of the proposed algorithm per iteration is $\mathcal{O}(k|E| + kn^2 + k^2n + k^3) = \mathcal{O}(kn^2)$. Since $k = 2$ or 3 in our problem, complexity per iteration can be expressed as $\mathcal{O}(n^2)$ flops.

## 3.5 Recovery Condition Analysis

In this section, we analyze a recovery condition under which the LRM-CG algorithm recovers the Euclidean distance matrices accurately. Overall, our analysis is divided into two parts. In the first part, we analyze a condition ensuring the successful recovery of the sampled (observed) entries, i.e., $\|\mathcal{P}_E(\widehat{\mathbf{D}}) - \mathcal{P}_E(\mathbf{D})\|_F = 0$. In the second part, we investigate a condition guaranteeing the exact recovery of the Euclidean distance matrices, i.e., $\|\widehat{\mathbf{D}} - \mathbf{D}\|_F = 0$. By exact recovery, we mean that the output $\mathbf{D}_i$ of LRM-CG converges to the original Euclidean distance matrix $\mathbf{D}$.

**Definition 4** *For a sequence of matrices $\{\mathbf{D}_i\}_{i=1}^{\infty}$, if $\lim_{i\to\infty} \|\mathbf{D}_i - \mathbf{D}\|_F = 0$, we say $\{\mathbf{D}_i\}_{i=1}^{\infty}$ converges to $\mathbf{D}$. Further, we say $\{\mathbf{D}_i\}_{i=1}^{\infty}$ converges linearly to $\mathbf{D}$ with convergent rate $\lambda$ if there exists $\lambda$ ($1>\lambda \geq 0$) satisfying*

$$\lim_{i\to\infty} \quad \frac{\|\mathbf{D}_{i+1} - \mathbf{D}\|_F}{\|\mathbf{D}_i - \mathbf{D}\|_F} = \lambda.$$

### 3.5.1 Convergence of LRM-CG at Sampled Entries

In this subsection, we show that $\{\mathcal{P}_\Omega(\mathbf{D}_i)\}_1^{\infty}$, sequence of matrices generated by LRM-CG at sampled points, converges to $\mathcal{P}_\Omega(\mathbf{D})$. For example, if $\mathbf{D} = \begin{bmatrix} 0 & 29 & 20 \\ 29 & 0 & 81 \\ 20 & 81 & 0 \end{bmatrix}$

and $E = \{(1,2),(1,3)\}$, then $\mathcal{P}_E(\mathbf{D}) = \begin{bmatrix} 0 & 29 & 20 \\ 29 & 0 & 0 \\ 20 & 0 & 0 \end{bmatrix}$. Thus, we will show that

$$\lim_{i\to\infty} \mathcal{P}_E(\mathbf{D}_i) = \begin{bmatrix} 0 & d_{12}^2(\infty) & d_{13}^2(\infty) \\ d_{21}^2(\infty) & 0 & 0 \\ d_{31}^2(\infty) & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 29 & 20 \\ 29 & 0 & 0 \\ 20 & 0 & 0 \end{bmatrix} = \mathcal{P}_E(\mathbf{D}).$$

The minimal set of assumptions used for the analytical tractability are as follows:

**A1** : $f(\mathbf{Y}_i) - f(R_{\mathbf{Y}_i}(\alpha_i \mathbf{P}_i)) \geq -\tau\alpha_i < \text{grad} f(\mathbf{Y}_i), \mathbf{P}_i >$ for $\tau$ satisfying $0<\tau<1/2$,

**A2** : $| < \text{grad} f(R_{\mathbf{Y}_i}(\alpha_i \mathbf{P}_i)), \mathbf{P}_i > | \leq -\mu < \text{grad} f(\mathbf{Y}_i), \mathbf{P}_i >$ for $\mu$ satisfying $\tau<\mu<1/2$,

**A3** : $c\|\text{grad} f(\mathbf{Y}_i)\|_F \geq \|\nabla_{\mathbf{Y}} f(\mathbf{Y}_i)\|_F$ for $c$ satisfying $c>1$.

In essence, **A1** and **A2** can be considered as extensions of the strong Wolfe's conditions[6] [90]. The assumption **A1** says that the cost function $f(\mathbf{Y}_i)$ decreases monotonically as long as $\mathbf{P}_i$ is chosen in an opposite direction of $\text{grad} f(\mathbf{Y}_i)$ on the tangent space $T_{\mathbf{Y}_i}\widetilde{\mathcal{Y}}$ (i.e., $< \text{grad} f(\mathbf{Y}_i), \mathbf{P}_i > \leq 0$) (see Lemma 13). Note that **A1** is reasonable assumption since there always exists a stepsize satisfying this assumption.

**Lemma 10** *There exists* $\alpha_i>0$ *satisfying* **A1**.

**Proof:** See Appendix E □

[6]Consider an unconstrained minimization in $\mathbb{R}^n$ with a differentiable cost function $f(\mathbf{x})$ (i.e., $\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x})$). The update equation is given by $\mathbf{x}_{i+1} = \mathbf{x}_i + \alpha_i \mathbf{p}_i$ for a stepsize $\alpha_i$ and a descent direction $\mathbf{p}_i$. The well-known strong Wolfe's conditions is given by

$$\begin{aligned} f(\mathbf{x}_i) - f(\mathbf{x}_{i+1}) &\geq -\tau\alpha_i < \nabla_{\mathbf{x}} f(\mathbf{x}_i), \mathbf{p}_i >, \\ | < \nabla_{\mathbf{x}} f(\mathbf{x}_{i+1}), \mathbf{p}_i > | &\leq -\mu < \nabla_{\mathbf{x}} f(\mathbf{x}_i), \mathbf{p}_i >, \end{aligned}$$

for some constants $0<\tau<\mu<1$.

Note that if the stepsize $\alpha_i$ is chosen to be very small, then $\mathbf{Y}_{i+1} = R_{\mathbf{Y}_i}(\alpha_i \mathbf{P}_i) \approx R_{\mathbf{Y}_i}(\mathbf{0}) = \mathbf{Y}_i$, and thus

$$f(\mathbf{Y}_i) - f(R_{\mathbf{Y}_i}(\alpha_i \mathbf{P}_i)) \quad \approx \quad 0.$$

and $-\tau\alpha_i < \text{grad} f(\mathbf{Y}_i), \mathbf{P}_i > \approx 0$. In this case, **A1** holds true approximately. However, there would be almost no update of $\mathbf{Y}_i$ so that the algorithm will converge extremely slowly. To circumvent this pathological scenario, we use **A2**, which is in essence an extension of the strong Wolfe's condition for the Riemannian manifold. Under this assumption, $\alpha_i$ cannot be chosen to be very small since otherwise we have $R_{\mathbf{Y}_i}(\alpha_i \mathbf{P}_i) \approx \mathbf{Y}_i$, and thus

$$| < \text{grad} f(R_{\mathbf{Y}_i}(\alpha_i \mathbf{P}_i)), \mathbf{P}_i > | \approx | < \text{grad} f(\mathbf{Y}_i), \mathbf{P}_i > | \geq -\mu < \text{grad} f(\mathbf{Y}_i), \mathbf{P}_i >,$$

which contradicts the assumption **A2**.

The assumption **A3** is needed to guarantee the global convergence of LRM-CG. We will discuss more on this in Remark 2.

Our first main result, stating successful recovery condition at sampled entries, is formally described in the following theorem.

**Theorem 11 (strong convergence of LRM-CG)** *Let* $\{\mathbf{D}_i = g(\mathbf{Y}_i)\}_{i=1}^{\infty}$ *be the sequence of the matrices generated by LRM-CG and* $\mathbf{D}$ *be the original Euclidean distance matrix. Under* $\mathbf{A1}$*,* $\mathbf{A2}$*, and* $\mathbf{A3}$*,* $\{\mathcal{P}_E(\mathbf{D}_i)\}_{i=0}^{\infty}$ *converges linearly to* $\mathcal{P}_E(\mathbf{D})$*.*

**Remark 1 (strongly convergent condition in $\mathbb{R}^n$)** *Note that* $\lim_{i\to\infty} \|\mathcal{P}_E(\mathbf{D}_i) - \mathcal{P}_E(\mathbf{D})\|_F = 0$ *is equivalent to*

$$\lim_{i\to\infty} \|\nabla_{\mathbf{Y}} f(\mathbf{Y}_i)\|_F = 0. \tag{3.25}$$

*This condition is often referred to as the strongly convergent condition of the nonlinear CG algorithms in the vector space. The equivalence can be established by the following sandwich lemma.*

***Lemma 12***

$$2\|\mathcal{P}_E(\mathbf{D}_i) - \mathcal{P}_E(\mathbf{D})\|_F \leq \|\nabla_{\mathbf{Y}} f(\mathbf{Y}_i)\|_F \leq (2\sqrt{n}+2)\|\mathcal{P}_E(\mathbf{D}_i) - \mathcal{P}_E(\mathbf{D})\|_F.$$

**Proof:** *See Appendix F* □

**Remark 2** *Recently, an attempt has been made to extend the convergent analysis of the conventional CG algorithms (over the Euclidean space $\mathbb{R}^n$) to the Riemannian manifolds. In [91, Theorem 4.3], it has been shown that under certain assumption,*

$$\lim_{i\to\infty} \inf \|grad f(\mathbf{Y}_i)\|_F = 0. \tag{3.26}$$

*One can observe that the Euclidean gradient $\nabla_{\mathbf{Y}} f(\mathbf{Y}_i)$ is replaced by the Riemannian gradient $grad f(\mathbf{Y}_i)$. Unfortunately, the convergence of the Riemannian gradient in (3.26) does not imply the convergence of Euclidean gradient in (3.25) because*

$$\begin{aligned} \|\nabla_{\mathbf{Y}} f(\mathbf{Y}_i)\|_F^2 &= \|\mathcal{P}_{T_{\mathbf{Y}}\widetilde{\mathcal{Y}}}(\nabla_{\mathbf{Y}} f(\mathbf{Y}_i))\|_F^2 + \|\mathcal{P}^{\perp}_{T_{\mathbf{Y}}\widetilde{\mathcal{Y}}}(\nabla_{\mathbf{Y}} f(\mathbf{Y}_i))\|_F^2 \\ &= \|grad f(\mathbf{Y}_i)\|_F^2 + \|\mathcal{P}^{\perp}_{T_{\mathbf{Y}}\widetilde{\mathcal{Y}}}(\nabla_{\mathbf{Y}} f(\mathbf{Y}_i))\|_F^2, \end{aligned} \tag{3.27}$$

*where $grad f(\mathbf{Y}_i) = \mathcal{P}_{T_{\mathbf{Y}}\widetilde{\mathcal{Y}}}(\nabla_{\mathbf{Y}} f(\mathbf{Y}_i))$ (see (3.22)). One can observe from this that the condition in (3.26) is not sufficient to guarantee (3.25), that is, one cannot guarantee $\lim_{i\to\infty} \|\mathcal{P}_E(\mathbf{D}_i) - \mathcal{P}_E(\mathbf{D})\|_F = 0$ just from (3.26). However, by the introduction of **A3**, equivalence between (3.25) and (3.26) can be established. We will show that the assumption **A3** holds true with overwhelming probability in Section 3.5.3.*

We are now ready to prove Theorem 11.

**Proof of Theorem 11**

First, we show that under **A1** and **A2**, $\|\mathcal{P}_E(\mathbf{D}_i) - \mathcal{P}_E(\mathbf{D})\|_F$ is non-increasing. That is, if $\chi$ is defined by

$$\chi = \begin{cases} \sup\limits_{\mathbf{Y}\in\{\mathbf{Y}_i\}_{i=1}^{\infty}} \frac{\|P^{\perp}_{T_{\mathbf{Y}}\widetilde{\mathcal{Y}}}(\nabla_{\mathbf{Y}} f(\mathbf{Y}))\|_F}{\|\nabla_{\mathbf{Y}} f(\mathbf{Y})\|_F} & \text{if } \|\nabla_{\mathbf{Y}} f(\mathbf{Y})\|_F \neq 0 \\ 1 & \text{otherwise} \end{cases}, \tag{3.28}$$

then there exists $\gamma{>}0$ such that $\gamma(1-\chi^2) \leq 1$ and

$$\|\mathcal{P}_E(\mathbf{D}_{i+1}) - \mathcal{P}_E(\mathbf{D})\|_F^2 \leq \left(1 - \gamma(1-\chi^2)\right) \|\mathcal{P}_E(\mathbf{D}_i) - \mathcal{P}_E(\mathbf{D})\|_F^2. \tag{3.29}$$

We need the following lemma to prove this.

**Lemma 13** *If $\beta_i$ is chosen based on Fletcher-Reeves' rule*[7]*, that is,*

$$\beta_i = \frac{< gradf(\mathbf{Y}_i), gradf(\mathbf{Y}_i) >}{< gradf(\mathbf{Y}_{i-1}), gradf(\mathbf{Y}_{i-1}) >}, \tag{3.31}$$

*then*

$$\frac{< gradf(\mathbf{Y}_{i+1}), \mathbf{P}_{i+1} >}{\|gradf(\mathbf{Y}_{i+1})\|_F^2} \leq -\frac{1-2\mu}{1-\mu} - \frac{\mu^{i+1}}{1-\mu}.$$

**Proof:** See Appendix G. □

**Lemma 14** $\|gradf(\mathbf{Y}_i)\|_F^2 \geq 8(1-\chi^2) f(\mathbf{Y}_i)$.

[7] In our simulation, we employ Hager-Zhang's rule in the choice of $\beta_i$ to improve the empirical performance of the CG method [89]:

$$\beta_i = \frac{1}{h^2} < h\mathbf{H}_i - 2\mathbf{P}_i \|\mathbf{H}_i\|_F^2, \text{grad} f(\mathbf{Y}_i) > \tag{3.30}$$

where $\mathbf{H}_i = \text{grad} f(\mathbf{Y}_i) - P_{T_{\mathbf{Y}_i}\widetilde{\mathcal{Y}}}(\text{grad} f(\mathbf{Y}_{i-1}))$ and $h =< \mathbf{P}_i, \mathbf{H}_i >$. In our analysis, however, we use Fletcher-Reeves' rule for mathematical tractability.

**Proof:** See Appendix H. □

We are now ready to prove (3.29). First, from **A1**, we have

$$\begin{aligned} f(\mathbf{Y}_{i+1}) &\leq f(\mathbf{Y}_i) + \tau\alpha_i < \text{grad} f(\mathbf{Y}_i), \mathbf{P}_i > \\ &\overset{(a)}{\leq} f(\mathbf{Y}_i) - \tau\alpha_i \left(\frac{1-2\mu}{1-\mu} + \frac{\mu^i}{1-\mu}\right) \|\text{grad} f(\mathbf{Y}_i)\|_F^2 \\ &\leq f(\mathbf{Y}_i) - \tau\alpha_i \left(\frac{1-2\mu}{1-\mu}\right) \|\text{grad} f(\mathbf{Y}_i)\|_F^2, \\ &\overset{(b)}{\leq} f(\mathbf{Y}_i) - 8\tau\alpha_i \left(\frac{1-2\mu}{1-\mu}\right) (1-\chi^2) f(\mathbf{Y}_i), \end{aligned}$$

where (a) and (b) follow from Lemma 13 and Lemma 14, respectively. Let

$$\gamma_i = 8\tau\alpha_i \left(\frac{1-2\mu}{1-\mu}\right),$$

then $\gamma_i > 0$ (since $\alpha_i > 0$) and hence

$$f(\mathbf{Y}_{i+1}) \leq (1 - \gamma_i(1-\chi^2)) f(\mathbf{Y}_i).$$

Recalling that $f(\mathbf{Y}_i) = \frac{1}{2}\|\mathcal{P}_E(\mathbf{D}_i) - \mathcal{P}_E(\mathbf{D})\|_F^2$, we have

$$\|\mathcal{P}_E(\mathbf{D}_{i+1}) - \mathcal{P}_E(\mathbf{D})\|_F^2 \leq \left(1 - \gamma_i(1-\chi^2)\right) \|\mathcal{P}_E(\mathbf{D}_i) - \mathcal{P}_E(\mathbf{D})\|_F^2.$$

By choosing $\gamma = \min_i \gamma_i$, we get the desired result.

Now, what remains is to show that $\lim_{i\to\infty} \|\mathcal{P}_E(\mathbf{D}_i) - \mathcal{P}_E(\mathbf{D})\|_F = 0$ under (3.29). From (3.27) and (3.28), we have $0 \leq \chi \leq 1$, and thus we need to consider two cases:

1) $\chi < 1$ case: In this case, one can easily show that $1 > (1 - \gamma(1-\chi^2))^{1/2}$. Using this together with (3.29), we have

$$\lim_{i\to\infty} \frac{\|\mathcal{P}_E(\mathbf{D}_{i+1}) - \mathcal{P}_E(\mathbf{D})\|_F}{\|\mathcal{P}_E(\mathbf{D}_i) - \mathcal{P}_E(\mathbf{D})\|_F} = (1 - h(1-\chi^2))^{1/2} < 1$$

and hence

$$\lim_{i\to\infty} \|\mathcal{P}_E(\mathbf{D}_i) - \mathcal{P}_E(\mathbf{D})\|_F = 0.$$

Thus, the sequence $\{\mathcal{P}_E(\mathbf{D}_i)\}_{i=1}^{\infty}$ converges linearly to $\mathcal{P}_E(\mathbf{D})$.

2) $\chi = 1$ case: In this case, we show that there exists $j$ satisfying $\|\nabla_{\mathbf{Y}} f(\mathbf{Y}_j)\|_F = 0$. As discussed in Remark 1 and Lemma 12, this is a sufficient condition to guarantee the strong convergence of LRM-CG. In this case, no further update can be made after $j$-th iteration (and thus linear convergence is naturally guaranteed). To show this, we use the contradiction argument. Suppose that $\|\nabla_{\mathbf{Y}} f(\mathbf{Y}_i)\|_F \neq 0$ for all $i$. Then, from (3.28) we should have

$$\sup_{\mathbf{Y}\in\{\mathbf{Y}_i\}_{i=1}^{\infty}} \frac{\|P^{\perp}_{T_{\mathbf{Y}}\widetilde{\mathcal{Y}}}(\nabla_{\mathbf{Y}} f(\mathbf{Y}))\|_F}{\|\nabla_{\mathbf{Y}} f(\mathbf{Y})\|_F} = \chi = 1.$$

Further, from (3.27), we have

$$\begin{aligned} \|P^{\perp}_{T_{\mathbf{Y}_i}\widetilde{\mathcal{Y}}}(\nabla_{\mathbf{Y}} f(\mathbf{Y}_i))\|_F^2 &= \|\nabla_{\mathbf{Y}} f(\mathbf{Y}_i)\|_F^2 - \|\text{grad} f(\mathbf{Y}_i)\|_F^2 \\ &\leq \|\nabla_{\mathbf{Y}} f(\mathbf{Y}_i)\|_F^2 - \frac{1}{c^2}\|\nabla_{\mathbf{Y}} f(\mathbf{Y}_i)\|_F^2, \end{aligned}$$

where the inequality is from **A3** ($c$>1). Thus,

$$\begin{aligned} 1 &= \sup_{\mathbf{Y}\in\{\mathbf{Y}_i\}_{i=1}^{\infty}} \frac{\|P^{\perp}_{T_{\mathbf{Y}}\widetilde{\mathcal{Y}}}(\nabla_{\mathbf{Y}} f(\mathbf{Y}))\|_F^2}{\|\nabla_{\mathbf{Y}} f(\mathbf{Y})\|_F^2} \\ &\leq \sup_{\mathbf{Y}\in\{\mathbf{Y}_i\}_{i=1}^{\infty}} \frac{\|\nabla_{\mathbf{Y}} f(\mathbf{Y})\|_F^2 - \frac{1}{c^2}\|\nabla_{\mathbf{Y}} f(\mathbf{Y})\|_F^2}{\|\nabla_{\mathbf{Y}} f(\mathbf{Y})\|_F^2} \\ &= 1 - \frac{1}{c^2}, \end{aligned}$$

which is contradiction. Thus, $\|\nabla_{\mathbf{Y}} f(\mathbf{Y}_i)\|_F = 0$ for some $j$.

### 3.5.2 Exact Recovery of Euclidean Distance Matrices

So far, we have shown that the output of LRM-CG converges to the original Euclidean distance matrix $\mathbf{D}$ at sampled entries (i.e., $\mathcal{P}_E(\mathbf{D}_\infty) = \mathcal{P}_E(\mathbf{D})$). In this subsection, we show that all entries of $\mathbf{D}_i$ converge to that of the original Euclidean distance matrix $\mathbf{D}$ with overwhelming probability.

Before we proceed, we briefly discuss the probability model of the sampling operator $\mathcal{P}_E$. Let $\delta_{ij}$ be a Bernoulli random variable that takes value 1 if $d_{ij} \leq r$ (recall that

$r$ is the radio communication range) and 0 otherwise. Since the distance is symmetric (i.e., $d_{ij} = d_{ji}$), we have $\delta_{ij} = \delta_{ji}$. Also, since the diagonal entries of $\mathbf{D}$ are all zeros, we define $\delta_{ii} = 0$ for all $i$. Then, for a matrix $\mathbf{A}$, $\mathcal{P}_E(\mathbf{A})$ can be expressed as

$$\begin{aligned} \mathcal{P}_E(\mathbf{A}) &= \sum_{i \neq j} \delta_{ij} < \mathbf{A}, \mathbf{e}_i \mathbf{e}_j^T > \mathbf{e}_i \mathbf{e}_j^T, \\ &= \sum_{i \neq j} \delta_{ij} a_{ij}, \end{aligned} \tag{3.32}$$

where $\mathbf{e}_i$ is the standard basis of $\mathbb{R}^n$. For example, if $\mathbf{A} = \begin{bmatrix} 0 & 10 & 17 \\ 10 & 0 & 3 \\ 17 & 3 & 0 \end{bmatrix}$ and $E = \{(1,2),(2,3)\}$, then

$$\begin{aligned} \mathcal{P}_E(\mathbf{A}) &= \begin{bmatrix} 0 & 10 & 0 \\ 10 & 0 & 3 \\ 0 & 3 & 0 \end{bmatrix} \\ &= 10 \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} + 3 \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} + 10 \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} + 3 \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}. \end{aligned}$$

We now characterize the random variables $\delta_{ij}$ using $P(d_{ij} \leq r)$. Since $d_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|_2$, it follows $P(d_{ij} \leq r) = P(\|\mathbf{x}_i - \mathbf{x}_j\|_2 \leq r)$. In this work, we assume that elements of $\mathbf{x}_i$ (locations of sensor nodes) are i.i.d. random and uniformly distributed over unit interval. By denoting $p = P(d_{ij} \leq r)$, the probability mass function (PMF) of $\delta_{ij}$ can be expressed as

$$f(\delta_{ij}; p) = p^{\delta_{ij}} (1-p)^{1-\delta_{ij}}. \tag{3.33}$$

The following lemma provides an explicit expression of $p$ in terms of the radio communication range $r$.

**Lemma 15** *If an element of the location vectors $\mathbf{x}_i$ is i.i.d. and uniform on unit interval, then*

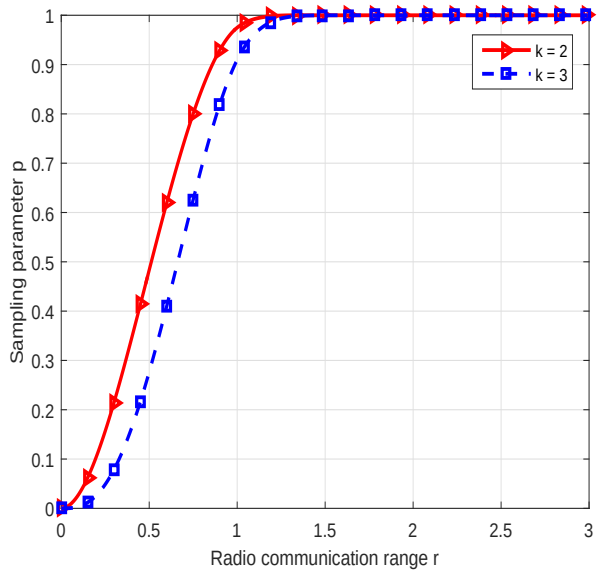



Figure 3.4: The sampling parameter $p$ gets close to 1 as $r$ increases. Here, elements of $\mathbf{x}_i$ are i.i.d. random variables according to the uniform distribution over unit interval.

*a) If $k = 2$ (2-dimensional Euclidean space),*

$$p = \begin{cases} \pi r^2 - \frac{8}{3} r^3 + \frac{1}{2} r^4 & \text{if } 0 \le r \le 1 \\ p_1(r) & \text{if } 1 \le r \le \sqrt{2} \\ 1 & \text{else} \end{cases},$$

*b) If $k = 3$ (3-dimensional Euclidean space),*

$$p = \begin{cases} \frac{4}{3}\pi r^3 - \frac{3\pi}{2} r^4 + \frac{8}{5} r^5 - \frac{1}{6} r^6 & \text{if } 0 \le r \le 1 \\ p_2(r) & \text{if } 1 \le r \le \sqrt{2} \\ p_3(r) & \text{if } \sqrt{2} \le r \le \sqrt{3} \\ 1 & \text{else} \end{cases},$$

*where*

$$\begin{aligned} p_1(r) &= -\frac{2}{3} - 2r^2 - \frac{1}{2} r^4 + \frac{1}{3}(8r^2 + 1)\sqrt{r^2 - 1} \\ &\quad + 2r^2 \sin^{-1}\left(\frac{2}{r^2} - 1\right) + \frac{2}{1 + \tan\left(\frac{\sin^{-1}\left(\frac{2}{r^2} - 1\right)}{2}\right)}, \end{aligned} \tag{3.34}$$

$$\begin{aligned} p_2(r) &= -\frac{15\pi + 37}{30} + \frac{6\pi + 1}{2} r^2 - \frac{8\pi}{3} r^3 + \frac{3\pi + 3}{2} r^4 + \frac{1}{3} r^6 \\ &\quad + 2r^4 \sin^{-1}\sqrt{1 - \frac{1}{r^2}} + \left(\frac{2}{15} - \frac{44}{15} r^2 - \frac{16}{5} r^4\right)\sqrt{r^2 - 1} \\ &\quad - 2r^4 \sin^{-1}\left(\frac{1}{r}\right) - r^4 \sin^{-1}\left(\frac{2}{r^2} - 1\right) \\ &\quad - \frac{16}{3\left(1 + \tan\left(\frac{1}{2}\sin^{-1}\left(\frac{2}{r^2} - 1\right)\right)\right)^3} \\ &\quad + \frac{4}{\left(1 + \tan\left(\frac{1}{2}\sin^{-1}\left(\frac{2}{r^2} - 1\right)\right)\right)^2}, \end{aligned} \tag{3.35}$$

$$
\begin{aligned}
p_3(r) \;=\; & -\frac{\pi}{2}+\frac{97}{30}-\frac{7}{2}r^2-\frac{3}{2}r^4-\frac{1}{6}r^6+\left(\frac{26}{15}+\frac{44}{15}r^2+\frac{8}{5}r^4\right)\sqrt{r^2-2} \\
& +\frac{16}{3}r^3\tan^{-1}\sqrt{1-\frac{2}{r^2}}+(2-8r^2)\tan^{-1}\sqrt{r^2-2} \\
& -(2r^4-4r^2)\sin^{-1}\sqrt{\frac{r^2-2}{r^2-1}}+(r^4+2r^2)\sin^{-1}\left(\frac{3-r^2}{r^2-1}\right) \\
& -\frac{16}{3}r^3\tan^{-1}\sqrt{\frac{1}{r^4-2r^2}}+8r^2\tan^{-1}\sqrt{\frac{1}{r^2-2}} \\
& +(2r^4-4r^2)\sin^{-1}\sqrt{\frac{1}{r^2-1}}+\frac{16}{3}\frac{1}{\left(1+\tan\left(\frac{1}{2}\sin^{-1}\left(\frac{3-r^2}{r^2-1}\right)\right)\right)^3} \\
& -4\frac{1}{\left(1+\tan\left(\frac{1}{2}\sin^{-1}\left(\frac{3-r^2}{r^2-1}\right)\right)\right)^2}. \qquad (3.36)
\end{aligned}
$$

**Proof:** See Appendix I. □

It is worth noting that $p_1(r)$, $p_2(r)$, and $p_3(r)$ increase monotonically with $r$ (see Fig. 3.4).

We now state our main result.

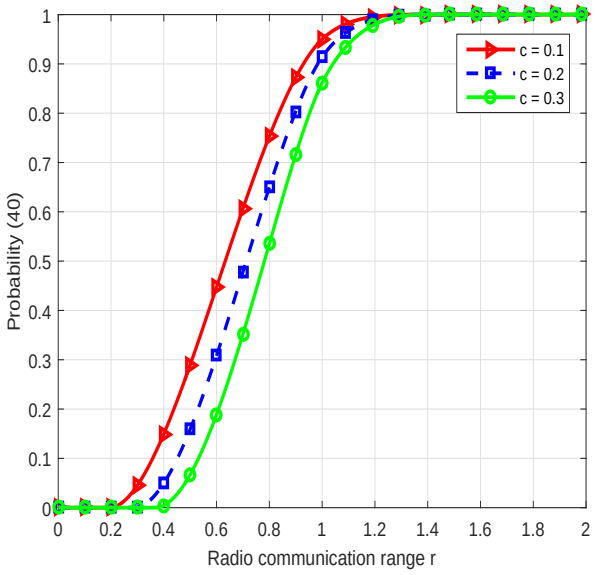
**Theorem 16** *Under the assumption* **A1**, **A2**, *and* **A3**, *the output sequence* $\{\mathbf{D}_i = g(\mathbf{Y}_i)\}_1^\infty$ *of LRM-CG converges globally to the Euclidean distance matrix* $\mathbf{D}$ *(*$\lim_{i\to\infty}\|\mathbf{D}_i-\mathbf{D}\|_F=0$*) with the probability at least*

$$
1-\exp\left(-\left((1-c)\log\left(\frac{1-c}{1-p}\right)+c\log\left(\frac{c}{p}\right)\right)\right) \qquad (3.37)
$$

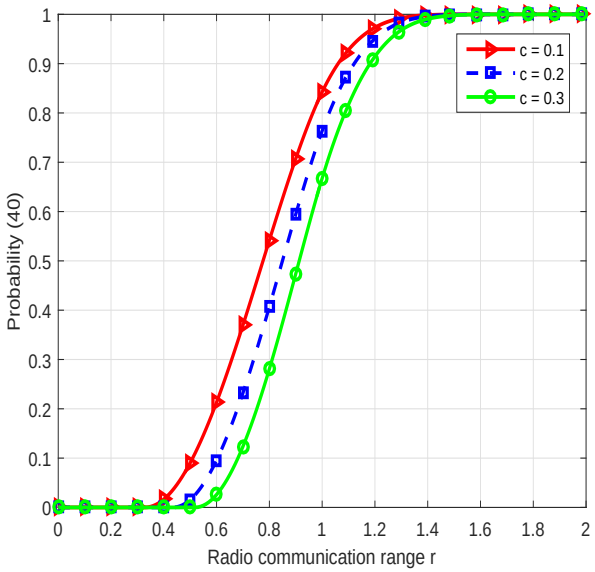
*for some constant* $c$ *satisfying* $0<c<1$ *and* $c<p$.

**Remark 3** *From Lemma 15, we see that* $p$ *gets close to 1 as the radio communication range* $r$ *increases. Thus, as shown in Fig. 3.5, the chance of recovering* $\mathbf{D}$ *increases with* $r$.

**Remark 4** *Theoretical guarantee on the recovery of a matrix has been provided by Candes and Recht in [5], and later improved in [6, 41]. In short, if entries of a*




(a)




(b)

Figure 3.5: The Euclidean distance matrix $\mathbf{D}$ can be recovered with overwhelming probability in (a) 2D and (b) 3D Euclidean space when $r$ is large.

*matrix are chosen at random, then the $n \times n$ matrix with rank $k$ can be recovered with overwhelming probability as long as the number of measurements $m$ follows $m = \mathcal{O}(kn^{1.2}\log(n))$. The analysis in these works is based on the assumption that observed entries are sampled i.i.d. ( and follow Bernoulli or uniform distribution). Whereas, our analysis does not require independence assumption among the sampled entries of $\mathbf{D}$ since the elements of $\mathbf{D}$ are related. In other words, random variables $\delta_{ij}$ do not need to be independent. For example, consider the scenario illustrated in Fig. 3.6. Since the sensor node 4 is located inside the triangle formed by three sensor nodes (nodes 1, 2, and 3), one can see that $d_{14} \leq \max(d_{12}, d_{13})$. Thus, if $d_{12}$ and $d_{13}$ are already known (i.e., $d_{12} \leq r$, $d_{13} \leq r$), then so is $d_{14}$. In other words, $P(\delta_{14} = 1 | \delta_{12} = \delta_{13} = 1) = 1$, while $P(\delta_{14} = 1)$ is not necessarily one. In our work, we do not put any assumption on the independence of the entries of $\mathbf{D}$ yet show that $\mathbf{D}$ can be recovered exactly with overwhelming probability when $r$ is large.*

Following lemma is useful to prove Theorem 16.

**Lemma 17** *For a given matrix $\mathbf{A}$, if the diagonal entries are zeros (i.e., $a_{ii} = 0$ for all $i$) and $\|\mathbf{A}\|_F < \infty$, then there exists a constant $t$ $(0<t<1)$ satisfying*

$$t\|\mathbf{A}\|_F^2 \leq \|\mathcal{P}_E(\mathbf{A})\|_F^2, \tag{3.38}$$

*with the probability at least $1 - \exp\left(-\left((1-mt)\log\left(\frac{1-mt}{1-p}\right) + mt\log\left(\frac{mt}{p}\right)\right)\right)$ for some constant $m \geq 1$, provided that $0<mt<p<1$.*

**Proof:** See Appendix J. □

*Proof of Theorem 16:* Let $\mathbf{A} = \mathbf{D}_i - \mathbf{D}$. Then from Lemma 17, we have

$$\|\mathbf{D}_i - \mathbf{D}\|_F \quad \leq \quad \frac{1}{\sqrt{t}}\|\mathcal{P}_E(\mathbf{D}_i) - \mathcal{P}_E(\mathbf{D})\|_F \tag{3.39}$$

with the probability at least $1 - \exp\left(-\left((1-mt)\log\left(\frac{1-mt}{1-p}\right) + mt\log\left(\frac{mt}{p}\right)\right)\right)$ for some constant $m$ satisfying $m \geq 1$ and $0<m<\frac{p}{t}$. Combining this with $\lim_{i\to\infty} \|\mathcal{P}_E(\mathbf{D}_i) -$

$\mathcal{P}_E(\mathbf{D})\|_F = 0$ (Theorem 11), we can conclude that

$$\lim_{i\to\infty} \|\mathbf{D}_i - \mathbf{D}\|_F = 0,$$

with the probability at least $1 - \exp\left(-\left((1-c)\log\left(\frac{1-c}{1-p}\right) + c\log\left(\frac{c}{p}\right)\right)\right)$, where $c = mt$.

### 3.5.3 Discussion on A3

In this section, we show that the assumption **A3** ($c\|\text{grad}f(\mathbf{Y}_i)\|_F^2 + \epsilon > \|\nabla_{\mathbf{Y}} f(\mathbf{Y}_i)\|_F^2$ for some $c>1$ and $\epsilon>0$) holds true with overwhelming probability when $r$ is large. Note that when $\epsilon$ is sufficiently small, one can simply put $c\|\text{grad}f(\mathbf{Y}_i)\|_F^2 \geq \|\nabla_{\mathbf{Y}} f(\mathbf{Y}_i)\|_F^2$, which is the strict form of **A3**. Intuitively, if the generated sequence of Riemannian gradient goes to zero ($\lim_{i\to\infty} \|\text{grad}f(\mathbf{Y}_i)\|_F = 0$), so does the corresponding sequence of the Euclidean gradient ($\lim_{i\to\infty} \|\nabla_{\mathbf{Y}} f(\mathbf{Y}_i)\|_F = 0$). In order to show this, we first need to define the coherence, a measure of concentration in a matrix [5].

**Definition 5 (Coherence [5])** *Let $Q$ be a subspace of $\mathbb{R}^n$ of dimension $k$ and $\mathbf{P}_Q$ be the orthogonal projection onto $Q$. Then the coherence of $Q$ is defined by*

$$\mu(Q) = \frac{n}{k} \max_{1\leq i\leq n} \|\mathbf{P}_Q \mathbf{e}_i\|_2^2.$$

Consider a matrix $\mathbf{A}$ of rank $k$ whose singular value decomposition is given by

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T = \sum_{i=1}^{k} \sigma \mathbf{u}_i \mathbf{v}_i^T, \tag{3.40}$$

where $\mathbf{U} = \begin{bmatrix} \mathbf{u}_1 & \cdots & \mathbf{u}_k \end{bmatrix}$ and $\mathbf{V} = \begin{bmatrix} \mathbf{v}_1 & \cdots & \mathbf{v}_k \end{bmatrix}$ are the matrices constructed by the left and right singular vectors, respectively, and $\mathbf{\Sigma}$ is the diagonal matrix whose diagonal entries are $\sigma_i$. From (3.40), we see that the concentration on the vertical direction (concentration in the row) is determined by $\mathbf{u}_i$ and that on the horizontal direction (concentration in the column) is determined by $\mathbf{v}_i$. For example,

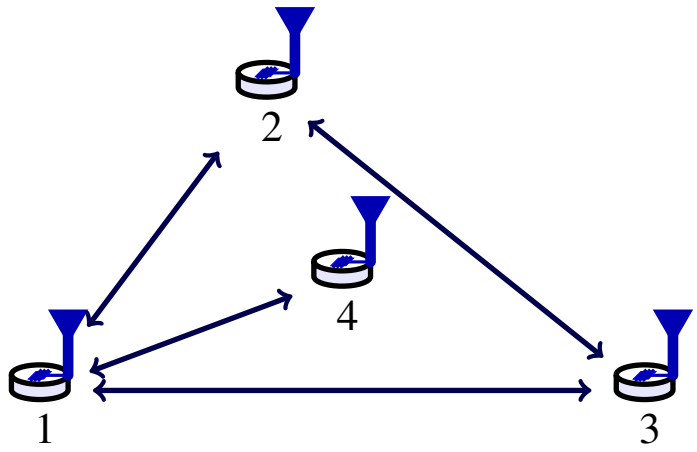



Figure 3.6: Suppose that the sensor node 4 is inside the triangle formed by three sensor nodes 1, 2, and 3. Then for a given $r$, it can be shown that $d_{14} \leq \max(d_{12}, d_{13})$, and thus $P(d_{14} \leq r | d_{12} \leq r, d_{13} \leq r) = 1$ which is not necessarily equivalent to $P(d_{14} \leq r)$.

if one of the standard basis vector $\mathbf{e}_i$, say $\mathbf{e}_1 = \begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix}^T$, lies on the space spanned by $\mathbf{u}_1, \cdots, \mathbf{u}_k$ while others $(\mathbf{e}_2, \mathbf{e}_3, \cdots)$ are orthogonal to this space, then it is clear that nonzero entries of the matrix are only on the first row. Since we need to check the concentration on both vertical and horizontal directions, we need to investigate both $\mu(\mathbf{U})$ and $\mu(\mathbf{V})$. In this regard, the coherence of a matrix $\mathbf{A}$ is defined by [5]

$$\mu(\mathbf{A}) = \max(\mu(U), \mu(V)). \tag{3.41}$$

In particular, if $\mathbf{A}$ is a positive semidefinite matrix, then it is clear that $\mathbf{U} = \mathbf{V}$ and thus $\mu(\mathbf{A}) = \mu(\mathbf{U})$.

**Theorem 18** *Suppose $\mu(\mathbf{Y}_i) \leq \mu_0$ for a given matrix $\mathbf{Y}_i \in \widetilde{\mathcal{Y}}$. Then, for any $c$>1 and $\epsilon$>0,*

$$c^2 \|grad f(\mathbf{Y}_i)\|_F^2 + \epsilon > \|\nabla_{\mathbf{Y}} f(\mathbf{Y}_i)\|_F^2, \tag{3.42}$$

*with probability at least*

$$1 - \exp\left(-\left(m\epsilon \log\left(\frac{m\epsilon}{1-p}\right) + (1-m\epsilon)\log\left(\frac{1-m\epsilon}{p}\right)\right)\right) \tag{3.43}$$

*for some constant $m$ satisfying $m$>0 and $0<m<\frac{1-p}{\epsilon}$, provided that $n \geq 2c\mu_0 k$.*

**Remark 5** *From Lemma 15, we see that $p$ gets close to 1 as $r$ increases. Thus, as shown in Fig. 3.7, when $r$ is large, (3.42) holds true with overwhelming probability.*

Following lemmas are needed to prove the theorem.

**Lemma 19**

$$\begin{aligned} & \|\nabla_{\mathbf{Y}} f(\mathbf{Y})\|_F^2 - c^2 \|grad f(\mathbf{Y})\|_F^2 \\ \leq & \sum_{i \neq j} \sum_{u \neq v} \delta_{ij} | < \mathbf{B}, \mathbf{e}_i \mathbf{e}_j^T > < \mathbf{B}, \mathbf{e}_u \mathbf{e}_v^T > \\ & \qquad < (\mathcal{I} - c^2 P_{T_{\mathbf{Y}} \widetilde{\mathcal{Y}}}) l(\mathbf{e}_i \mathbf{e}_j^T), l(\mathbf{e}_u \mathbf{e}_v^T) > |, \end{aligned} \tag{3.44}$$

*where $\mathbf{B} = g(\mathbf{Y}) - \mathbf{D}$ and $l(\mathbf{A}) = 2eye(Sym(\mathbf{A})\mathbf{1}) - 2Sym(\mathbf{A})$.*

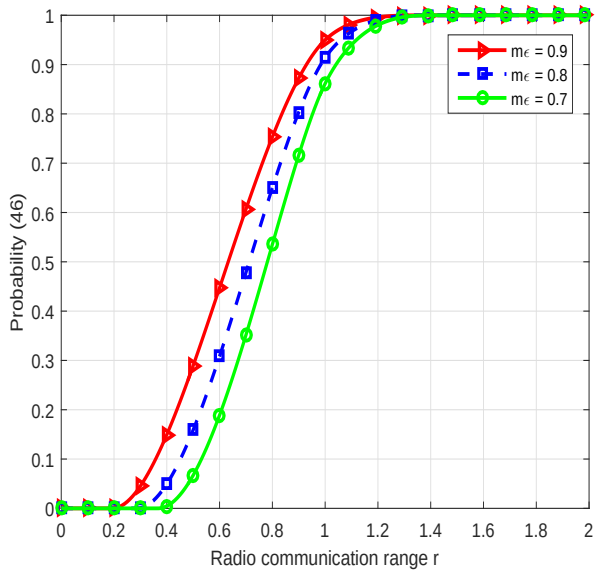



Figure 3.7: The condition (3.42) holds true with overwhelming probability when the radio communication range $r$ is large.

**Proof:** See Appendix K. □

**Lemma 20** *If* $n^2 \geq 4c\mu(\mathbf{Y})^2k^2$ *and* $i \neq j$*, then*

$$| < (\mathcal{I} - cP_{T_{\mathbf{Y}}\widetilde{\mathcal{Y}}})l(\mathbf{e}_i\mathbf{e}_j^T), l(\mathbf{e}_i\mathbf{e}_j^T) > | \geq 4\left(1 - \frac{4c\mu(\mathbf{Y})^2k^2}{n^2}\right), \tag{3.45}$$

*where* $c>0$ *and* $l(\mathbf{A}) = 2eye(Sym(\mathbf{A})\mathbf{1}) - 2Sym(\mathbf{A})$.

**Proof:** See Appendix L. □

**Lemma 21** *Let* $\delta_1, \delta_2, \cdots, \delta_N$ *be identically (not necessarily independently) distributed Bernoulli random variables with* $P(\delta_i = 1) = p$ *and* $P(\delta_i = 0) = 1 - p$*. Also, let* $a_1, a_2, \cdots, a_N$ *be positive values. Let* $q$ *be the largest integer obeying* $2^q \leq N$*. Then, for any* $\epsilon>0$*,*

$$P(\sum_{i=1}^{N} \delta_i a_i \geq \epsilon) \leq \exp\left(-\left(m\epsilon \log\left(\frac{m\epsilon}{1-p}\right) + (1 - m\epsilon)\log\left(\frac{1-m\epsilon}{p}\right)\right)\right), \tag{3.46}$$

*where* $m\epsilon = \frac{\sum_{i=1}^{N} a_i - \epsilon}{2^q a_{\min}}$ *with* $a_{\min} = \min_i a_i$*, provided that* $0<m\epsilon<1-p$.

**Proof:** See Appendix M. □

Now, we are ready to prove Theorem 18.

**Proof:** [Proof of Theorem 18] Let

$$\begin{aligned} I &= \|\nabla_{\mathbf{Y}} f(\mathbf{Y}_i)\|_F^2 - c^2\|\text{grad} f(\mathbf{Y}_i)\|_F^2, \\ s_{ij} &= < g(\mathbf{Y}_i) - \mathbf{D}, \mathbf{e}_i\mathbf{e}_j^T >, \end{aligned}$$

and

$$g_{ij} = \sum_{u \neq v} |s_{ij}s_{uv} < (\mathcal{I} - c^2 P_{T_{\mathbf{Y}_i}\widetilde{\mathcal{Y}}})l(\mathbf{e}_i\mathbf{e}_j^T), l(\mathbf{e}_u\mathbf{e}_v^T) > |.$$

In this proof, we will show that $P(I \leq \epsilon)$ is lower bounded by the quantity in (3.43). First, since $I \leq \sum_{i \neq j} \delta_{ij} g_{ij}$ from Lemma 19 and hence $P(I \leq \epsilon \mid \sum_{i \neq j} \delta_{ij} g_{ij} \leq \epsilon) = 1$, we have

$$\begin{aligned} P(\sum_{i \neq j} \delta_{ij} g_{ij} \leq \epsilon) &= P(\sum_{i \neq j} \delta_{ij} g_{ij} \leq \epsilon) P(I \leq \epsilon \mid \sum_{i \neq j} \delta_{ij} g_{ij} \leq \epsilon) \\ &= P(I \leq \epsilon, \sum_{i \neq j} \delta_{ij} g_{ij} \leq \epsilon) \\ &= P(\sum_{i \neq j} \delta_{ij} g_{ij} \leq \epsilon \mid I \leq \epsilon) P(I \leq \epsilon) \\ &\leq P(I \leq \epsilon). \end{aligned} \tag{3.47}$$

What remains is to find out a lower bound of $P(\sum_{i \neq j} \delta_{ij} g_{ij} \leq \epsilon)$. Equivalently, we find out an upper bound of $P(\sum_{i \neq j} \delta_{ij} g_{ij} \geq \epsilon)$. First, in order to use Lemma 21, we need to find a lower bound of $g_{ij}$ for $(i,j) \in \Omega$ ($\Omega = \{(i,j) : s_{ij} \neq 0\}$). Let $s = \min_{(i,j) \in \Omega} |s_{ij}|$, then

$$\begin{aligned} g_{ij} &= \sum_{u \neq v} |s_{ij} s_{uv} < (\mathcal{I} - c^2 P_{T_{\mathbf{Y}} \widetilde{\mathcal{Y}}}) l(\mathbf{e}_i \mathbf{e}_j^T), l(\mathbf{e}_u \mathbf{e}_v^T) > | \\ &\geq |s_{ij}^2 < (\mathcal{I} - c^2 P_{T_{\mathbf{Y}} \widetilde{\mathcal{Y}}}) l(\mathbf{e}_i \mathbf{e}_j^T), l(\mathbf{e}_i \mathbf{e}_j^T) > | \\ &\overset{(a)}{\geq} 4s^2 \left(1 - 4c^2 \mu(\mathbf{Y}_i)^2 k^2 / n^2\right) \\ &\geq 4s^2 \left(1 - 4c^2 \mu_0^2 k^2 / n^2\right), \end{aligned}$$

where (a) follows from Lemma 20. Now using Lemma 21, we have

$$\begin{aligned} & P(\sum_{i \neq j} \delta_{ij} g_{ij} \geq \epsilon) \\ \leq \; & \exp\left(-\left(m\epsilon \log\left(\frac{m\epsilon}{1-p}\right) + (1 - m\epsilon) \log\left(\frac{1 - m\epsilon}{p}\right)\right)\right), \end{aligned} \tag{3.48}$$

where $m = (\sum_{(i,j) \in \Omega} g_{ij} - \epsilon)/(2^q \epsilon c_1)$ ($c_1 = 4s^2 \left(1 - 4c^2 \mu_0^2 k^2 / n^2\right)$), provided that $0 < m < \frac{1-p}{\epsilon}$ and $n \geq 2c\mu_0 k$. Here, $q$ is the largest integer obeying $2^q \leq |\Omega|$ ($|\Omega|$ being the cardinality of $\Omega$). From (3.47) and (3.48), and noting that $P(\sum_{i \neq j} \delta_{ij} g_{ij} \leq \epsilon) = 1 - P(\sum_{i \neq j} \delta_{ij} g_{ij} \geq \epsilon)$, we get the desired result. □

# Chapter 4

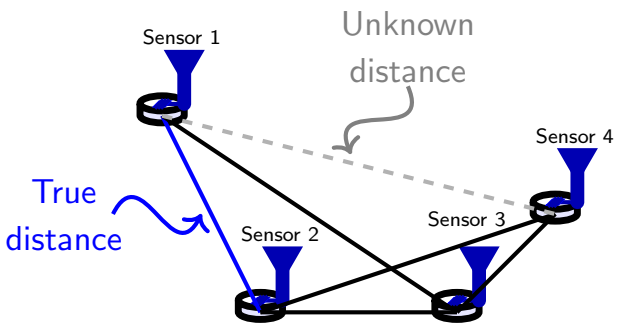
# Extended LRM-CG for The Outlier Problem

In many practical scenarios, observed pairwise distances can be contaminated by the outliers. It might be due to various reasons, including the power outage, obstacles, adversary attacks, hardware (Tx/Rx) malfunction, to name just a few. In general, the presence of outliers might reduce localization accuracy, resulting in incorrect locations of sensor nodes. As a motivation example, we consider 4 sensor nodes with the true observed distances given as (see Fig. 4.1)

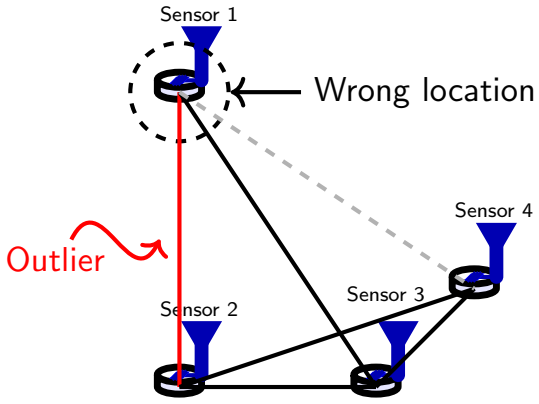
$$\mathbf{D}_o = \begin{bmatrix} 0 & 5 & 13 & ? \\ 5 & 0 & 2 & 10 \\ 5 & 2 & 0 & 2 \\ ? & 10 & 2 & 0 \end{bmatrix},$$

where ? marks the unknown distance $d_{14}$. Without the outlier, one can easily find out $d_{14} = \sqrt{17}$. Now we suppose that the distance $d_{12}$ between the node 1 and node 2 is contaminated by outlier. That is, we use an arbitrary number to substitute the true value of $d_{12}$, say, $d_{12} = 9$. As a result, the reconstructed distance $\widehat{d}_{14} = \sqrt{13}$, which has a large reconstruction error. This obviously leads to the wrong location of the sensor node 1 (see Fig. 4.1).

In this chapter, we extend the proposed LRM-CG algorithm to solve the outlier problem in IoT localization. We first present the outlier problem model, and then show




(a)




(b)

Figure 4.1: Outliers might reduce the localization accuracy: (a) accurately reconstructed locations when there is no outlier and (b) inaccurate locations in the presence of outliers

the extended LRM-CG in detail. Finally, we show the simulation results of both LRM-CG and its extended version.

## 4.1 Problem Model

In general, an entry $d_{ij}^o$ of the observed matrix $\mathbf{D}_o$ is called an outlier if $d_{ij}^o \neq d_{ij}$ [92]. Often we use the relaxed definition using the tolerance level $\rho$ of observation error. That is, $d_{ij}^o$ is defined as an outlier if $|d_{ij}^o - d_{ij}| > \rho$. Since the outlier often degrades the localization performance severely, we should control it in the recovery process. To be specific, we model the observed distance as $d_{ij}^o = d_{ij} + l_{ij}$ ($l_{ij}$ is the outlier). Thus, $\mathcal{P}_E(\mathbf{D}_o) = \mathcal{P}_E(\mathbf{D}+\mathbf{L})$ where $\mathbf{L}$ is the outlier matrix. Since $\mathbf{L}$ is considered as a sparse matrix, we can modify the problem in (3.6) as

$$\min_{\substack{\mathbf{Y}\in\widetilde{\mathcal{Y}} \\ \mathbf{L}\in\mathbb{R}^{n\times n}}} \quad \frac{1}{2}\|\mathbf{W}\odot(\mathcal{P}_E(g(\mathbf{Y})) + \mathcal{P}_E(\mathbf{L}) - \mathcal{P}_E(\mathbf{D}_o))\|_F^2 + \tau\|\mathbf{L}\|_o, \tag{4.1}$$

where $\|\mathbf{L}\|_o$ is the number of nonzero entries of $\mathbf{L}$ and $\tau$ is the regularization factor controlling the tradeoff between the sparsity of $\mathbf{L}$ and the consistency of the observed distances. Since $\|\mathbf{L}\|_o$ is nonlinear and non-convex, we instead use the convex surrogate $\|\mathbf{L}\|_1 = \sum_{i=1}^{n}\sum_{j=1}^{n}|l_{ij}|$, and thus

$$\min_{\substack{\mathbf{Y}\in\widetilde{\mathcal{Y}} \\ \mathbf{L}\in\mathbb{R}^{n\times n}}} \quad \frac{1}{2}\|\mathbf{W}\odot(\mathcal{P}_E(g(\mathbf{Y})) + \mathcal{P}_E(\mathbf{L}) - \mathcal{P}_E(\mathbf{D}_o))\|_F^2 + \tau\|\mathbf{L}\|_1. \tag{4.2}$$

Thus, the modified cost function is $\widetilde{f} = \frac{1}{2}\|\mathbf{W}\odot(\mathcal{P}_E(g(\mathbf{Y}))+\mathcal{P}_E(\mathbf{L})-\mathcal{P}_E(\mathbf{D}_o))\|_F^2 + \tau\|\mathbf{L}\|_1$.

## 4.2 Extended LRM-CG

In order to solve the outlier problem, we use a slight modification version of the proposed LRM-CG and update the solutions $\mathbf{Y}$ and $\mathbf{L}$ of (4.2) in an alternating manner.

To be specific, the problem in (4.2) can be solved iteratively using alternative minimization as

$$\begin{aligned}\mathbf{Y}_{i+1} &= \arg\min_{\mathbf{Y}\in\widetilde{\mathcal{Y}}} \frac{1}{2}\|\mathbf{W}\odot(\mathcal{P}_E(g(\mathbf{Y}))+\mathcal{P}_E(\mathbf{L}_i) \\ &\quad -\mathcal{P}_E(\mathbf{D}_o))\|_F^2+\tau\|\mathbf{L}_i\|_1 \qquad (4.3)\\ \mathbf{L}_{i+1} &= \arg\min_{\mathbf{L}\in\mathbb{R}^{n\times n}} \frac{1}{2}\|\mathbf{W}\odot(\mathcal{P}_E(g(\mathbf{Y}_{i+1}))+\mathcal{P}_E(\mathbf{L}) \\ &\quad -\mathcal{P}_E(\mathbf{D}_o))\|_F^2+\tau\|\mathbf{L}\|_1. \qquad (4.4)\end{aligned}$$

The subproblem in (4.3) can be solved using the proposed LRM-CG with simple modifications of the cost function and the residual matrix $\mathbf{R}_i$ in Algorithm 1. The modified residual is

$$\mathbf{R}_i = \mathbf{W}\odot\mathbf{W}\odot(\mathcal{P}_E(g(\mathbf{Y}_i))+\mathcal{P}_E(\mathbf{L}_i)-\mathcal{P}_E(\mathbf{D}_o)). \qquad (4.5)$$

Note that $\mathcal{P}_E(\mathbf{L}_i)$ is added to the original residual $\mathbf{R}_i$.

The subproblem in (4.4) can be solved using the soft-thresholding operator, which gradually truncates the magnitude of the entries of a matrix [93]. For a given matrix $\mathbf{A}$, the soft-thresholding operator output $\mathcal{T}(\mathbf{A})$ is defined as

$$\mathcal{T}(a_{ij}) = \begin{cases} \frac{w_{ij}a_{ij}-\tau}{w_{ij}^2} & \text{if } w_{ij}a_{ij}\geq\tau \\ \frac{w_{ij}a_{ij}+\tau}{w_{ij}^2} & \text{if } w_{ij}a_{ij}\leq-\tau \\ 0 & \text{else} \end{cases}.$$

Using the soft-thresholding operator, the solution of (4.4) is given by [93]

$$\mathbf{L}_{i+1} = \mathcal{T}(\mathbf{W}\odot(\mathcal{P}_E(\mathbf{D}_o)-\mathcal{P}_E(g(\mathbf{Y}_{i+1})))). \qquad (4.6)$$

In the sequel, we call this modified version of LRM-CG as the extended LRM-CG (ELRM-CG) (see Algorithm 2).

In addition, by extending the convergence analysis of LRM-CG, we can readily obtain the convergence guarantee of ELRM-CG. First, for the subproblem (4.3), we can trivially extend the convergence analysis of the problem (3.6) in Section 3.5 and then have $h(\mathbf{Y}_{i+1},\mathbf{L}_i)\leq h(\mathbf{Y}_i,\mathbf{L}_i)$ where $h$ is the cost function of (4.2). Second, for

**Algorithm 2:** ELRM-CG algorithm

**Input:** $\mathbf{D}_{obs}$: the observed matrix,

$\mathbf{W}$: the weight matrix,

$\mathcal{P}_E$: the sampling operator,

$\epsilon$: tolerance,

$\mu \in (0 \quad 1)$: given constant,

$T$: number of iterations.

**Initialize:** $i = 1$,

$\mathbf{Y}_1 \in \widetilde{\mathcal{Y}}$, $\mathbf{E}_1 \in \mathbb{R}^{n \times n}$: initial points,

$\mathbf{P}_1$: initial conjugate direction.

**While** $i \leq T$ **do**

$\mathbf{R}_i = \mathbf{W} \odot \mathbf{W} \odot (\mathcal{P}_E(g(\mathbf{Y}_i)) + \mathcal{P}_E(\mathbf{L}_i) - \mathcal{P}_E(\mathbf{D}_{obs}))$

$\nabla_{\mathbf{Y}} \widetilde{f}(\mathbf{Y}_i) = 2\text{eye}(\text{Sym}(\mathbf{R}_i)\mathbf{1}) - 2\mathbf{R}_i$

$\text{grad}\widetilde{f}(\mathbf{Y}_i) = P_{T_{\mathbf{Y}_i}\widetilde{\mathcal{Y}}}(\nabla_{\mathbf{Y}} \widetilde{f}(\mathbf{Y}_i))$

$\mathbf{H}_i = \text{grad}\widetilde{f}(\mathbf{Y}_i) - P_{T_{\mathbf{Y}_i}\widetilde{\mathcal{Y}}}(\text{grad}\widetilde{f}(\mathbf{Y}_{i-1}))$

$h =< \mathbf{P}_i, \mathbf{H}_i >$

$\beta_i = \frac{1}{h^2} < h\mathbf{H}_i - 2\mathbf{P}_i \|\mathbf{H}_i\|_F^2, \text{grad}\widetilde{f}(\mathbf{Y}_i) >$

$\mathbf{P}_i = -\text{grad}\widetilde{f}(\mathbf{Y}_i) + \beta_i P_{T_{\mathbf{Y}_i}\widetilde{\mathcal{Y}}}(\mathbf{P}_{i-1})$

Find a stepsize $\alpha_i > 0$ such that

$\widetilde{f}(\mathbf{Y}_i) - \widetilde{f}(R_{\mathbf{Y}_i}(\alpha_i \mathbf{P}_i)) \geq -\mu\alpha_i < \text{grad}\widetilde{f}(\mathbf{Y}_i), \mathbf{P}_i >$

$\mathbf{Y}_{i+1} = R_{\mathbf{Y}_i}(\alpha_i \mathbf{P}_i)$

$\mathbf{L}_{i+1} = \mathcal{T}(\mathbf{W} \odot (\mathcal{P}_E(\mathbf{D}_o) - \mathcal{P}_E(g(\mathbf{Y}_{i+1}))))$

$\mathbf{D}_{i+1} = g(\mathbf{Y}_{i+1})$

**If** $\|\mathbf{W} \circ (\mathcal{P}_E(\mathbf{D}_{i+1} + \mathbf{E}_{i+1}) - \mathcal{P}_E(\mathbf{D}_{obs}))\|_F < \epsilon$ **then**

Exit from while loop

**End If**

Obtain $\mathbf{Q}$ and $\mathbf{\Lambda}$ using the eigendecomposition

$\mathbf{Y}_{i+1} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T$

$\widehat{\mathbf{X}} = \mathbf{Q}\mathbf{\Lambda}^{1/2}$

$i = i + 1$

**End While**

**Output:** $\widehat{\mathbf{X}}$

the subproblem (4.4), we can compute $\mathbf{L}_{i+1}$ in one step using the soft-thresholding operator in (4.6) and thus we always have $h(\mathbf{Y}_{i+1}, \mathbf{L}_{i+1}) \leq h(\mathbf{Y}_{i+1}, \mathbf{L}_i)$. Combining these, we have $h(\mathbf{Y}_{i+1}, \mathbf{L}_{i+1}) \leq h(\mathbf{Y}_{i+1}, \mathbf{L}_i) \leq h(\mathbf{Y}_i, \mathbf{L}_i)$ for all $i$, which ensures the convergence of ELRM-CG.

## 4.3 Numerical Evaluation

In this section, we test the performance of the proposed LRM-CG and its extended version ELRM-CG. In our simulations, we compare LRM-CG with following matrix completion algorithms:

- APG [94]: an algorithm to solve the robust PCA problem via an accelerated proximal gradient method.
- LRGeomCG [20]: this algorithm can be considered as the CG algorithm defined over the Riemannian manifold of low rank matrices (but not necessarily positive definite).
- SVT [10]: an algorithm to solve the NNM problem using a singular value thresholding technique.
- TNNR-ADMM [18]: an algorithm to solve the truncated NNM problem via an alternating direction method of multipliers.

Also, we compare LRM-CG with the following localization algorithms:

- MDS [78]: this is a multiscaling dimensional algorithm based on the shortest path algorithm and truncated eigendecomposition.
- SDP [95, 96]: an algorithm to solve the localization problem using a convex relaxation of nonconvex quadratic constraints of the node locations.

### 4.3.1 Simulation Setting

In our experiments, we generate an $n \times k$ location matrix $\mathbf{X}$ whose entries are sampled independently and identically from a uniform distribution in the interval with 50 meters. Using $\mathbf{X}$, we then compute the Euclidean distance matrix $\mathbf{D} = g(\mathbf{X}\mathbf{X}^T)$. As aforementioned, an entry $d_{ij}^o$ of $\mathbf{D}_o$ is known (observed) if it is smaller than the radio communication range (i.e., $d_{ij}^o \leq r$). In the scenario with observation error, an observation error matrix $\mathbf{N} \in \mathbb{R}^{n \times n}$ is added to $\mathbf{D}$. In general, the accuracy of the observed distances is inversely proportional to the true distances [97, 80]. In our simulations, we employ the RSSI-based model in which the cumulative effect of many attenuation factors of the wireless communication environment results in a log-normal distribution of the received power [97]. Specifically, let $\delta$ be a normal random variable with zero mean and variance $\sigma_{dB}^2$. Then, each entry $n_{ij}$ of $\mathbf{N}$ is $n_{ij} = (\kappa 10^{\frac{\delta}{10n_p}} - 1)d_{ij}$ where $\delta$ is the constant dB error in the received power measurement, $n_p$ is the path loss parameter, and $\kappa = 10^{-\frac{\sigma_{dB}^2 \ln 10}{200 n_p^2}}$ is a constant to enforce the unbiasedness of the observed distances (i.e., $E[n_{ij}] = 0$). In measuring the performance for each algorithm, we perform at least 1000 independent trials.

For initialization of the parameters in the proposed LRM-CG, we simply generate the initial entries of $\mathbf{X}$ and $\mathbf{L}$ at random according to the standard normal distribution. In the simulation with observation errors, we choose the weight matrix to suppress the large magnitude errors. For the $(i, j)$-th entry $w_{ij}$ of $\mathbf{W}$ (see (3.3)), we consider two settings. To account for the RSS-based measurement model, we set $w_{ij}$ inversely proportional to the error term $|d_{ij}^o - d_{ij}|$ as

$$w_{ij} = w_{ij}^* = \begin{cases} \exp(-|d_{ij}^o - \widetilde{d}_{ij}|^{\frac{1}{4}}) & \text{if } (i,j) \in E \\ 0 & \text{else} \end{cases}, \tag{4.7}$$

where $\widetilde{d}_{ij} = d_{ij}^o c^{3/4}/(1 + \sqrt{c^{1/8} - 1})^4$ is an estimate of $d_{ij}$[1]. When we do not use the

[1]Using the moment method, we obtain the approximate distance $\widetilde{d}_{ij}$ by solving $(d_{ij}^o)^{1/4} \approx E[(d_{ij}^o)^{1/4}] + \sqrt{Var((d_{ij}^o)^{1/4})}$.

RSS-based measurement model, we set $w_{ij} = 1$ for $(i, j) \in E$ and zero otherwise.

### 4.3.2 Convergence Efficiency

As performance measures, we use the mean square error (MSE) and the root mean square errors (RMSE), which are defined respectively as

$$\begin{aligned} MSE &= \frac{1}{\sqrt{n^2 - n}} \|\widehat{\mathbf{D}} - \mathbf{D}\|_F, \\ RMSE &= \sqrt{\frac{1}{n^2 - n} \sum_i \sum_{j \neq i} (\widehat{d}_{ij} - d_{ij})^2}. \end{aligned}$$

Note that the number of non-trivial entries of $\mathbf{D}$ is $n^2 - n$ since the diagonal elements are zero (i.e., $d_{ii} = 0$). Also, in order to compare the localization performance of the proposed algorithm, we use the mean square localization error (MSLE):

$$\mathcal{E} = \frac{1}{\text{Total unknown nodes}} \sum_{\text{All unknown nodes i}} \|\widehat{\mathbf{x}}_i - \mathbf{x}_i\|_2.$$

In Fig. 4.2, we plot the log-scale MSE as a function of the number of iterations for the 2-dimensional sensor networks. Note that the results are obtained for the scenario where 200 sensor nodes are randomly distributed in $50 \times 50\text{m}^2$ square area. We observe that the log-scale MSE decreases linearly with the number of iterations, meaning that the MSE decreases exponentially with the number of iterations. For example, if $r = 35\text{m}$, it takes about 60, 80, and 100 iterations to achieve $10^{-1}$, $10^{-3}$, and $10^{-5}$, respectively. Also, as expected, required number of iterations to achieve the given performance level decreases with the radio communication range $r$.

### 4.3.3 Performance Evaluation

In this subsection, we investigate the recovery performance of LRM-CG for scenarios with and without observation error. In Fig. 4.3, we plot the performance of the scenario without the observation error as a function of the sampling ratio, which is defined as the ratio of the number of observed pairwise distances to total number of pairwise

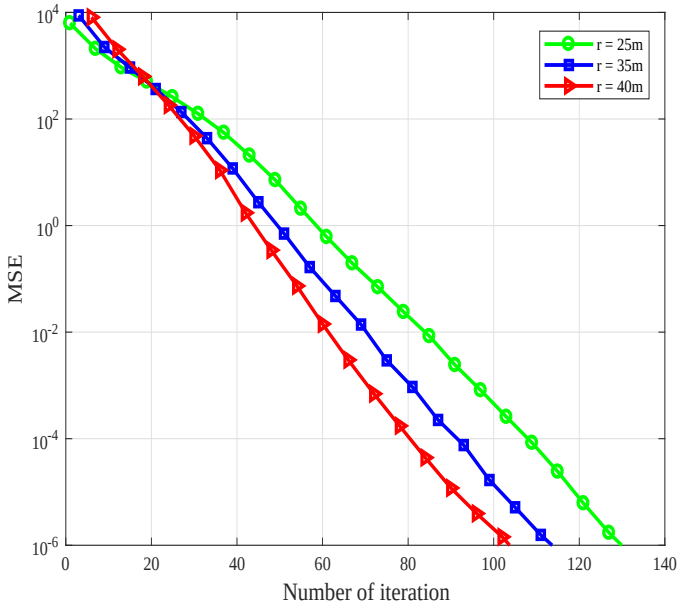
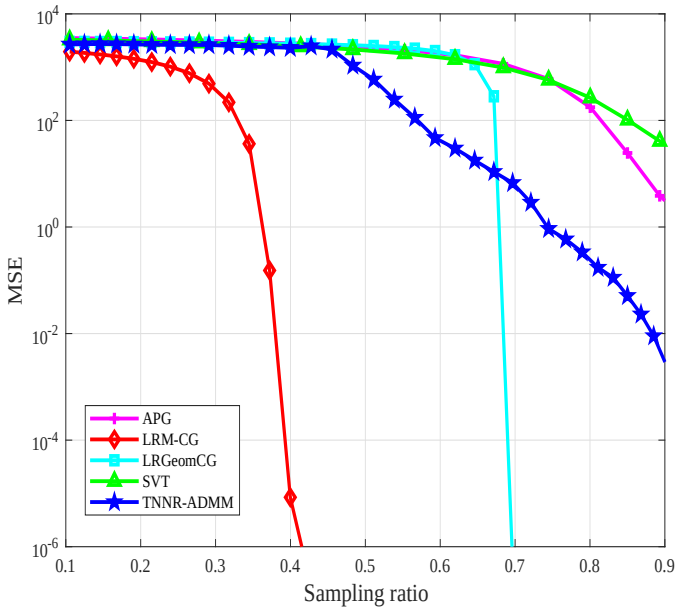
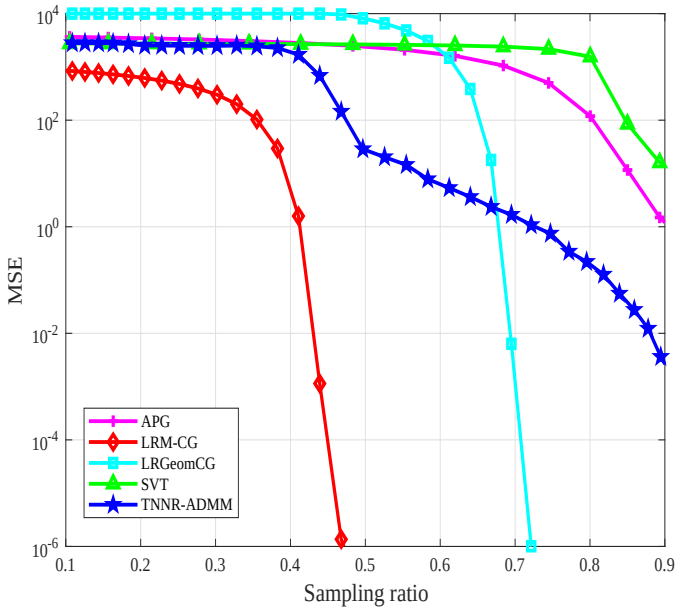



Figure 4.2: The MSE performance of LRM-CG for $k = 2$ (2-dimensional location vectors).

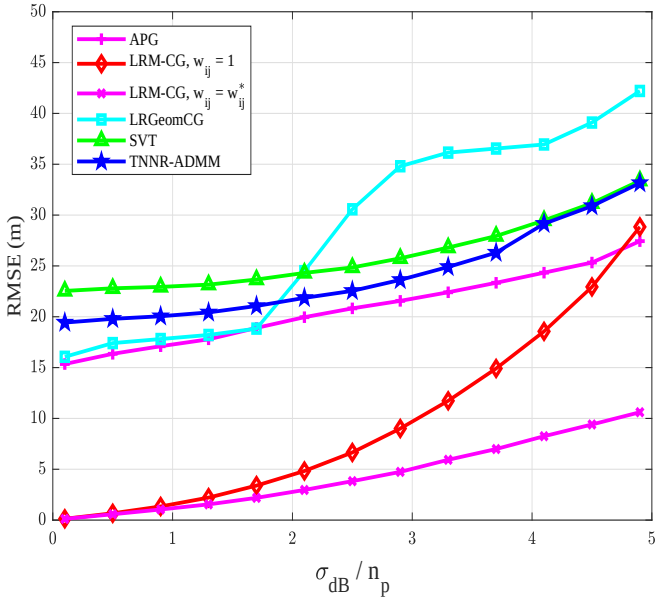



(a)

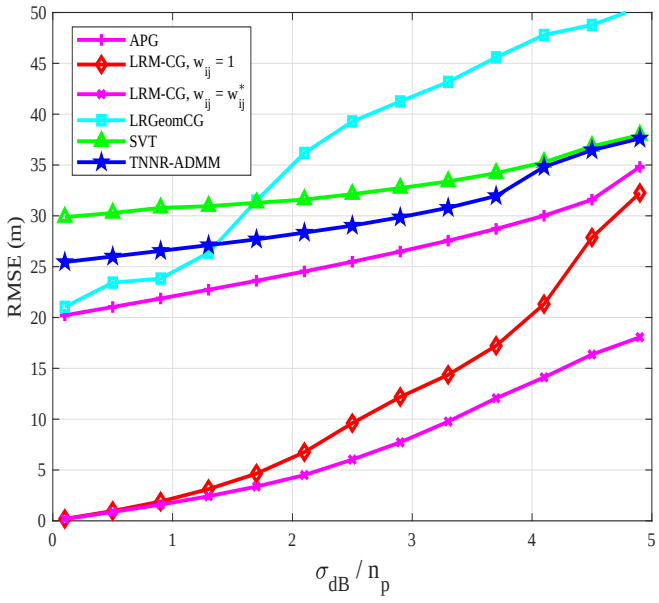



(b)

Figure 4.3: The MSE performance of the matrix completion algorithms for scenario without observation error for (a) 2-dimensional and (b) 3-dimensional location vectors.




(a)




(b)

Figure 4.4: The RMSE performance of the algorithms in presence of observation errors for (a) 2-dimensional and (b) 3-dimensional location vectors.

distances. Here, the sampling ratio is controlled by the radio communication range $r$[2]. We observe that LRM-CG outperforms conventional techniques by a large margin, achieving MSE $\leq 10^{-5}$ using 40% of measurements.

In Fig. 4.4, we plot the performance of LRM-CG as a function of $\sigma_{dB}/n_p$. In this experiment, sensor nodes are randomly distributed in $50 \times 50\text{m}^2$ square area ($k = 2$) and $50 \times 50 \times 50\text{m}^3$ cubic space ($k = 3$). We set the radio communication range $r = 30$m, resulting in 125 and 84 average connections per node for $k = 2$ and $k = 3$, respectively. While the performance of conventional matrix completion algorithms is poor (i.e., RMSE $\geq 5m$) in mid and high $\sigma_{dB}/n_p$ regime, the performance of LRM-CG is still good in small $\sigma_{dB}/n_p$ regime, achieving RMSE being less than 2.5m when $\sigma_{dB}/n_p \leq 1.5$.

We next investigate the localization performance of LRM-CG. We compare the performance of LRM-CG with the APG, LRGeomCG, SVT, TNNR-AMMD, MDS, and SDP-based algorithm [95]. In this experiment, 50 sensor nodes are randomly distributed in $50 \times 50 \times 50\text{m}^3$ ($k = 3$) and 4 anchor nodes are used to reconstruct the global node locations. The stopping threshold $\epsilon$ of LRM-CG is set to $10^{-8}$. Since the reconstructed matrix of the conventional matrix completion algorithm including APG, LRGeomCG, SVT, and TNNR-AMMD, is not necessarily an Euclidean distance matrix, we use the MDS technique [78] as a post-processing to project the output matrix on the Euclidean distance matrix cone. In Fig. 4.5, we observe that conventional localization algorithms perform poor (MSLE $\geq 5m$) for mid and high $\sigma_{dB}/n_p$ regime, but the proposed LRM-CG algorithm performs well in low $\sigma_{dB}/n_p$ regime, achieving MSLE being less than 3m for $\sigma_{dB}/n_p \leq 1$.

We next examine the running time complexity of the algorithms under test as a function of the number of sensor nodes. In our simulations, we set the maximum iteration number to 200 and the stopping threshold $\epsilon$ of the matrix completion algorithms

[2]In 2 and 3-dimensional Euclidean spaces, it can be shown that the sampling probability (sampling ratio) can be expressed as a non-decreasing function of $r$ (see Appendix B in Supplementary Material).

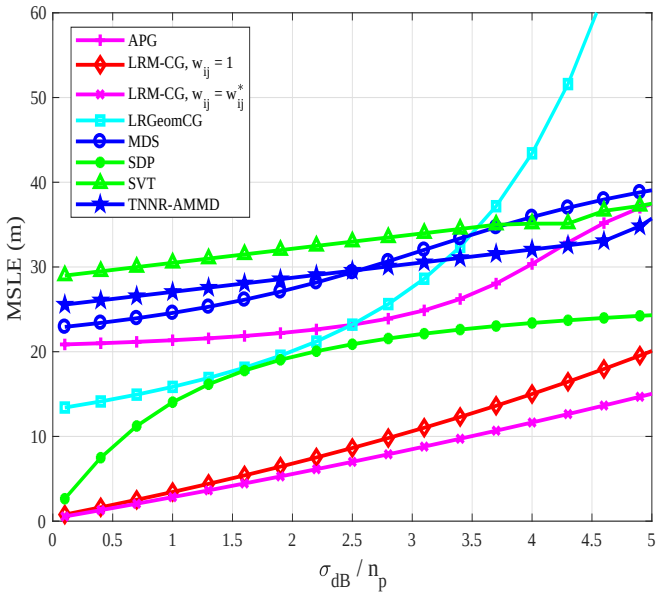
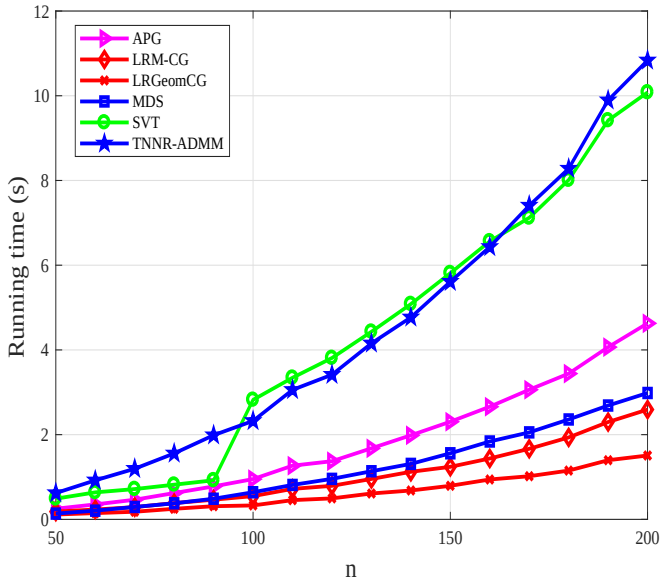



Figure 4.5: The RMSLE performance of the algorithms for 3-dimensional location vectors.

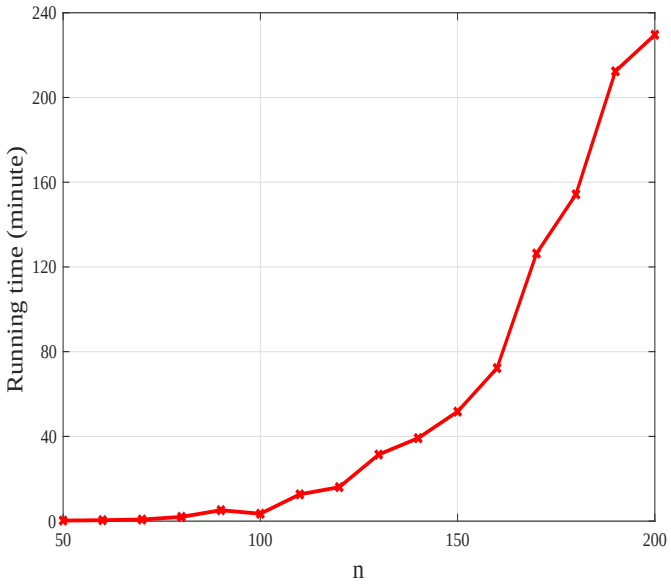
Table 4.1: Computational complexity of the matrix completion algorithms in recovery of $n \times n$ rank-$k$ matrix.

| Algorithms | Major computation | Total computational complexity per iteration |
|---|---|---|
| APG | Soft-thresholding SVD | $\mathcal{O}(\bar{k}n^2)^a$ |
| LRM-CG | Truncated EVD | $\mathcal{O}(k^2n + k\|E\|)$ |
| LRGeomCG | Truncated EVD | $\mathcal{O}(k^2n + k\|E\|)$ |
| MDS | Truncated EVD | $\mathcal{O}(kn^2)$ |
| SDP | Convex operator | $\mathcal{O}(n^3)$ |
| SVT | Soft-thresholding SVD | $\mathcal{O}(\bar{k}n^2)$ |
| TNNR-ADMM | Soft-thresholding SVD | $\mathcal{O}(\bar{k}n^2)$ |

[a]Note that $\bar{k}$ is the number of singular values being larger than the threshold used in the soft-thresholding based SVD technique [94, 10, 18].




(a)




(b)

Figure 4.6: Running time as a function of the number of sensor nodes: (a) the conventional matrix completion algorithms and the proposed LRM-CG and (b) SDP-based algorithm. Since the running time of SDP-based algorithm is much higher than that of the other algorithms, we separate the results into two plots.

to $10^{-6}$. From Fig. 4.6, we observe that the running time of the SDP-based technique is fairly large since it should solve the primal and dual problems using SDPT3 solver [47, 98]. The running time of APG, LRGeomCG, MDS, and the proposed LRM-CG is more or less similar when $n \leq 200$. In Table 4.1, we summarize the computational complexity of the algorithms under test in terms of flops. We observe that the computational complexity of LRM-CG is linearly proportional to the problem size $n$ and the number of the observed distances $|E|$, and thus competitive with the conventional approaches.

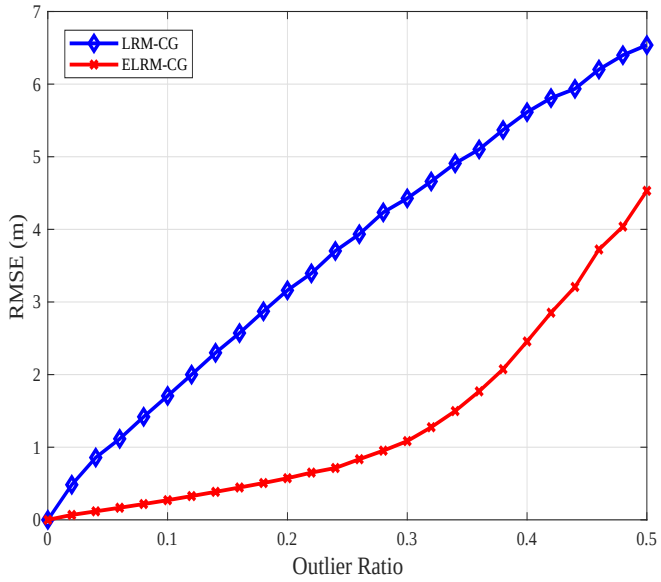
### 4.3.4 Outlier Problem

We next investigate the performance of the proposed LRM-CG algorithm and its extended version in the presence of outliers. When the outlier ratio $\theta$ is given, we randomly choose a set of the observed distances and replace this set by a set of random numbers. In this experiment, sensor nodes are randomly distributed in $50 \times 50\text{m}^2$ square area. In our simulation, we consider the scenario in which the magnitude of outliers is comparable to the distance level. We could observe that the extended LRM-CG outperforms the original LRM-CG, achieving MSLE being less than 0.5m up to the 20% outlier ratio (see Fig. 4.7).

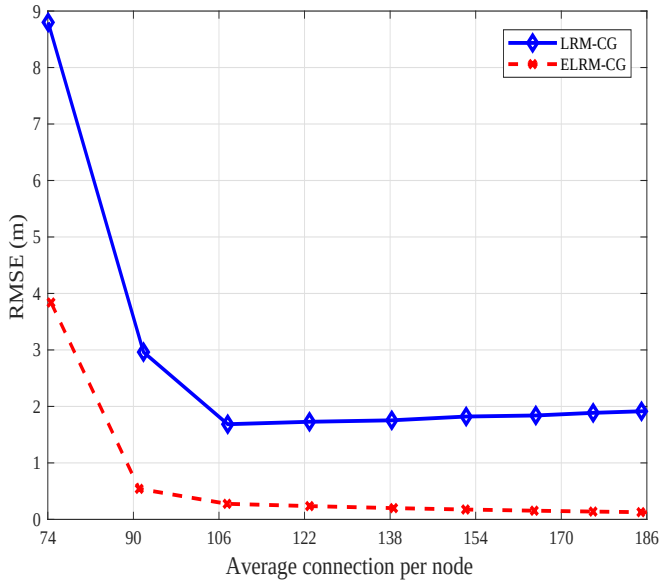
In order to show the robustness of the proposed LRM-CG algorithm, we also plot the histogram of the localization error $\|\widehat{\mathbf{x}}_i - \mathbf{x}_i\|_2$ for two outlier ratios ($\theta = 0.1$ and $0.3$). In Fig. 4.8, we observe that the localization error of the extended LRM-CG is much smaller than that of the original version. For example, when $\theta = 0.1$, the extended LRM-CG reconstructs most of the sensor locations with the error being less than 0.5m irrespective of the outliers.

### 4.3.5 Real Data

In this subsection, we examine the performance of the proposed LRM-CG algorithm using real measurements. In this simulation, we use the RSS-based measurement model

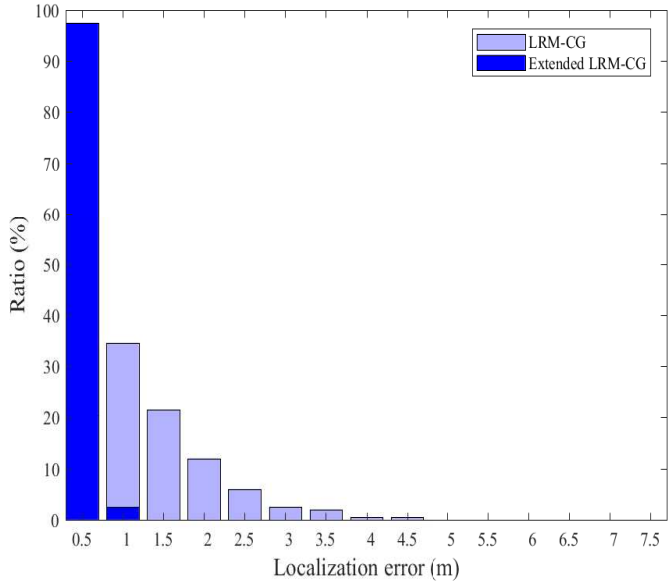



(a)

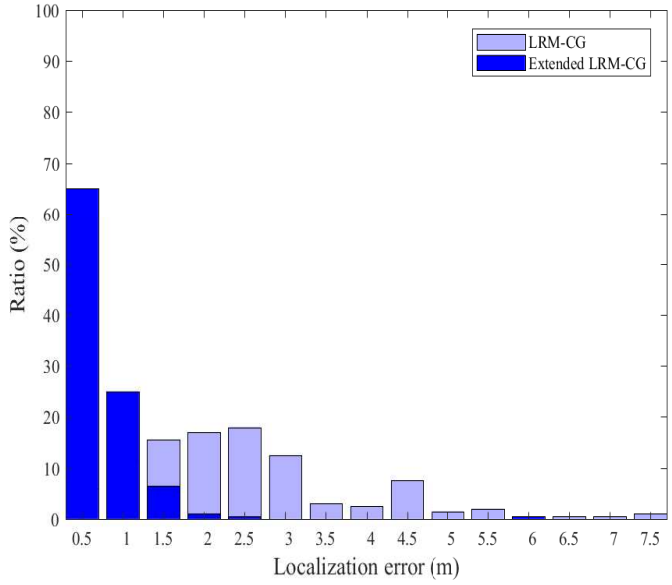



(b)

Figure 4.7: The MSLE performance of LRM-CG in terms of (a) outlier ratio $\theta$ and (b) average connection per node (for $\theta = 0.1$).




(a)




(b)

Figure 4.8: Histograms of the localization error $\|\widehat{\mathbf{x}}_i - \mathbf{x}_i\|_2$ when the outlier ratio $\theta$ satisfies (a) $\theta = 0.1$ and (b) $\theta = 0.3$.

Table 4.2: Localization errors with real measurements.

| $r$ (m) | Average connection per node | MSLE (m) | | | |
|---|---|---|---|---|---|
| | | LRM-CG | Extended LRM-CG | SDP with absolute cost function [95] | SDP with least square cost function [96] |
| 5.5 | 14 | 5.4893 | 4.9860 | 4.5038 | 3.7241 |
| 7.5 | 22 | 5.2796 | 4.9170 | 3.1287 | 3.3394 |
| 9.5 | 30 | 2.9917 | 2.8620 | 2.9274 | 3.0526 |
| 11.5 | 37 | 2.2636 | 2.2023 | 2.6272 | 2.5151 |

in [97]. This network consists of 44 sensor nodes randomly distributed in the $14\times14\text{m}^2$ square area and the transmit signal is generated via a wideband direct-sequence spread-spectrum (DSSS) operating at a center frequency of 2.4 GHz. For a given radio communication range $r$, we assume that $d_{ij}^o$ is known if $d_{ij} \leq r$ and unknown otherwise. We observe from Table 4.2 that the performance of the proposed LRM-CG is comparable to the SDP techniques in [95, 96][3] when $r = 9.5m$.

[3]The SDP-based techniques have various cost functions. In [95], the cost function is expressed as a sum of absolute errors in terms of the observed distances while that in [96] is a least squares function.

# Chapter 5

# LRMC Via Graph Neural Network

In previous chapters, we present the proposed LRM-CG algorithm to reconstruct the Euclidean distance matrix, which has the underlying Riemannian structure. We have shown that such additional structure can be used to significantly improve the recovery performance of LRMC in localization. In the same spirit, we propose a LRMC scheme, referred to as graph neural network-based LRMC (GNN-LRMC), to reconstruct the rating matrix in recommendation systems using its underlying graph structure. Recall that the rows and columns of the rating matrix are often indexed by users and products, respectively. In the user graph, users are represented as vertices and the (undirected) edge connecting two user nodes shows the correlation between the users' favorite products (see Fig. 5.1). Similarly, we use the product graph to represent the correlation among the products in term of their similar properties (e.g., color, appearance, and utility).

Such additional graph structure of the data can be easily incorporated into the matrix completion setting. To be specific, we can express the low-rank matrix $\mathbf{M}$ as $\mathbf{M} = \mathbf{U}\mathbf{V}^T \in \mathbb{R}^{n_1 \times n_2}$ where $\mathbf{U} \in \mathbb{R}^{n_1 \times k}$, $\mathbf{V} \in \mathbb{R}^{n_2 \times k}$, and $k = \text{rank}(\mathbf{M})$. Here, $\mathbf{U}$ and $\mathbf{V}$ can be related to the user and product graphs, respectively. For example, the rows of $\mathbf{U}$ are indexed by the nodes in $\mathcal{G}_r$, each row is assigned as multidimensional data for the corresponding node. The correlation between the row vectors of $\mathbf{U}$ would

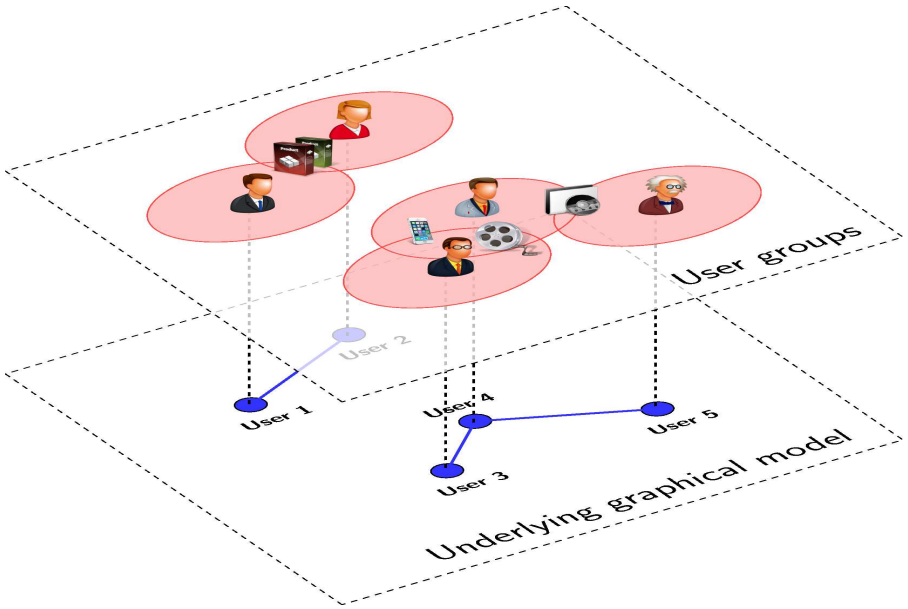



Figure 5.1: User graph with nodes indexed by user IDs and edges to show the correlation between the users' favorite products.

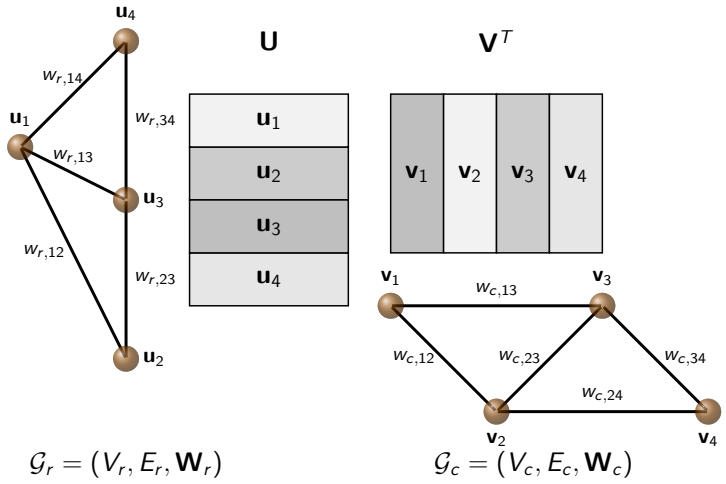
be matched to the edge weights in the graph[1]. Similarly, $\mathbf{V}$ can also be characterized by the column graph $\mathcal{G}_r$ (see Fig. 2.7). Thus, the LRMC problem can be reformulated as the problem to reconstruct the row and column graphs of $\mathbf{M}$. While it seems that the graph reconstruction problem is as difficult as the original problem, the low-rank property of $\mathbf{M}$ allows each node value to be updated based on the local connectivity of the corresponding node (see Fig. 2.7). In other words, convolutional neural networks (CNN) can be readily used to update the node values by performing the convolution across the graph domain [99, 100].

In recent years, graph-based CNN, referred to as graph neural network (GNN), has been proposed [73, 101, 102, 103]. In GNN, convolutional layers are first used to extract meaningful features in the graph and then a proper output model is used to map these features to the reconstructed low-rank matrix. This approach benefits from its low computational cost since just a few number of the shared weights are needed in the convolution. However, in the conventional GNN-based LRMC techniques, the key assumption is that the graph connections are pre-defined precisely and given as a priori in the training process, which might be impractical in real scenarios [22, 73].

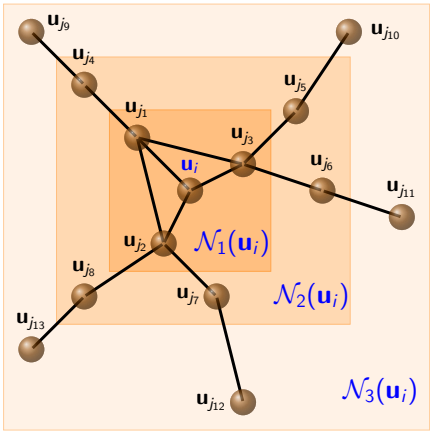
Our main goal is to propose the GNN-based LRMC scheme, referred to as GNN-LRMC, to reconstruct the row and column graphs of $\mathbf{M}$ (and eventually recover $\mathbf{M}$ itself). Our approach is motivated by recent results in GNN, which deploys multiple GNN layers to extract the feature and then update the node values using a fast and localized filter defined in the graph Fourier domain [73, 101]. In GNN-LRMC, an adaptive model is used to update the graph connections using the extracted features.

In this chapter, we first present the graph model of the low-rank matrix and then show the proposed GNN-LRMC in detail.

[1]When the weight is zero, there is no connection between two nodes in the graph.




(a)




(b)

Figure 5.2: (a) Graph model of $\mathbf{M} = \mathbf{U}\mathbf{V}^T$ and (b) the value of each node updated based on the local connectivity of this node. Each row $\mathbf{u}_i$ or $\mathbf{v}_j$ is the vector-valued representation at each node. $\mathcal{N}_t(\mathbf{u}_i)$ is the t-hop neighbors of $\mathbf{u}_i$, the nodes with the shortest path to $\mathbf{u}_i$ not being greater than $t$. In GNN, a polynomial filter of degree 3 affects on a local area of $\mathbf{u}_i$, i.e., $\mathbf{N}_3(\mathbf{u}_i)$.

## 5.1 Graph Model

We consider the LRMC problem in which the desired low-rank matrix $\mathbf{M} \in \mathbb{R}^{n_1 \times n_2}$ possesses an additional graph structure [22]. To be specific, let $\mathbf{M} = \mathbf{U}\mathbf{V}^T$, and also let $\mathcal{G}_r = (V_r, E_r, \mathbf{W}_r)$ and $\mathcal{G}_c = (V_c, E_c, \mathbf{W}_c)$ be the row and column graphs of $\mathbf{M}$, respectively. Here, $V_r$ and $V_c$ are vertex sets, $E_r$ and $E_c$ are edge sets, and $\mathbf{W}_r$ and $\mathbf{W}_c$ are the weight matrices. We suppose that there exists the mappings $f_r : V_r \to \mathbb{R}^k$ and $f_c : V_c \to \mathbb{R}^k$ such that $f_r(v_{ri}) = \mathbf{u}_i$ $f_c(v_{cj}) = \mathbf{v}_j$ where $v_{ri} \in V_r$, $v_{cj} \in V_c$, and $\mathbf{u}_i$ and $\mathbf{v}_j$ are the $i$-th row and the $j$-th row of $\mathbf{U}$ and $\mathbf{V}$, respectively. In practice, $f_r$ and $f_c$ are nonlinear functions and might not need to have closed-form expressions. Our main problem is to learn $f_r$ and $f_c$ in a supervised manner using GNN and eventually reconstruct $\mathbf{M}$. As aforementioned, $\mathbf{W}_r$ and $\mathbf{W}_c$ are also adjusted during the learning process of $f_r$ and $f_c$ to avoid the graph model mismatch.

## 5.2 Proposed GNN-LRMC

In GNN-LRMC, we first initialize $\mathcal{G}_r$ and $\mathcal{G}_c$ using trainable matrices $\mathbf{U}_o$ and $\mathbf{V}_o$, respectively. An adaptive model is used to update the graph connections of $\mathcal{G}_r$ (and also $\mathcal{G}_c$) using a modified full-connection neural network. In this model, the weight matrices of $\mathcal{G}_r$ and $\mathcal{G}_r$ are updated by applying a nonlinear activation function to normalized Euclidean distance matrices of $\mathbf{U}_o$ and $\mathbf{V}_o$, respectively. A proper choice of the activation function (e.g., rectified linear unit (ReLU)) allows us to maintain the sparsity of the graphs and thus reduce the computational cost of the proposed scheme. To update the node values of the graphs, our approach is motivated by recent results in GNN, which deploys multiple GNN layers using a polynomial filter defined in the graph Fourier domain [73, 101]. In the graph domain, such filter is expressed as a polynomial function of the graph Laplacian matrix, which is used to update each node value by performing the convolution between the shared weights and the values of the neighboring nodes of this node. In our approach, we express the filter in term of a

normalized Laplacian matrix and show that this filter can be used to maintain the DC component and thus stabilize the learning process. The filter paramters as well as $\mathbf{U}_o$ and $\mathbf{V}_o$ can be updated using a back propagation with a training cost function based on the Frobenius norm minimization in LRMC.

The main procedures of GNN-LRMC are (see Fig. 5.3):

1) Initialize $\mathbf{U}_o$ and $\mathbf{V}_o$ at random and assign each row of $\mathbf{U}_o$ and $\mathbf{V}_o$ to each vertex of the row graph $\mathcal{G}_r$ and the column graph $\mathcal{G}_c$, respectively.

2) Build the graph connection by computing the weight matrices $\mathbf{W}_r$ and $\mathbf{W}_c$ using full-connection neural network-based adaptive models (see Subsection 5.2.1).

3) Extract the feature matrices $\mathbf{U}$ and $\mathbf{V}$ by performing a graph-based convolution operation on $\mathcal{G}_r$ and $\mathcal{G}_c$, respectively (see Subsection 5.2.2).

4) Update $\mathbf{M}$ by feeding the feature matrices $\mathbf{U}$ and $\mathbf{V}$ to an output model (see Subsection 5.2.3).

5) Compute the loss function in (5.15) and (5.16) using the updated features and the weight matrices and perform the back propagation to update $\mathbf{U}_o$ and $\mathbf{V}_o$ and the filter parameters.

6) Repeat the above procedures until the value of the loss function is smaller than a pre-chosen threshold.

In the next subsections, we discuss on three fundamental components of our proposed scheme in detail. They include 1) full-connection neural network-based adaptive models to update the weight matrices, 2) multilayer GNN to extract the features across the graph domains, and 3) an output model to reconstruct the low-rank matrix $\mathbf{M}$ using the updated features.

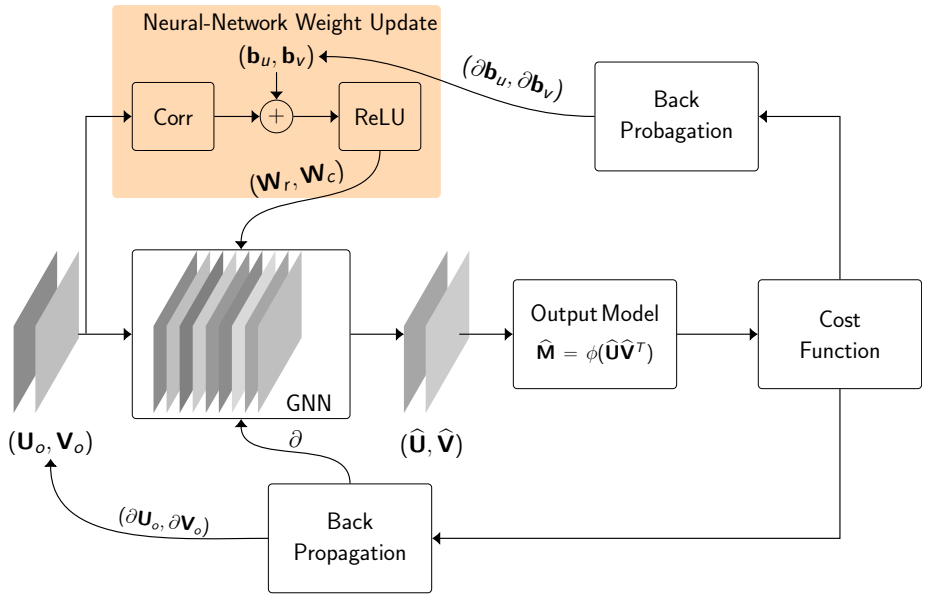



Figure 5.3: Block diagram of the proposed GNN-LRMC.

### 5.2.1 Adaptive Model

In the conventional techniques, the weight matrices $\mathbf{W}_r$ and $\mathbf{W}_c$ are often computed in a pre-processing procedure and then provided to GNN models as a priori. As a result, the mismatch of the pre-processing model might degrade the learning performance of the GNN. To overcome this issue, we propose an adaptive model to adjust the weight matrices based on full-connection neural network.

To be specific, $\mathbf{W}_r$ and $\mathbf{W}_c$ can be learned using a simplified full-connection neural layer. In the training process, the update expressions of $\mathbf{W}_r$ and $\mathbf{W}_c$ are

$$\mathbf{W}_r = \sigma(\mathbf{C}_{uu} + \mathbf{b}_u\mathbf{1}^T) \tag{5.1}$$

$$\mathbf{W}_c = \sigma(\mathbf{C}_{vv} + \mathbf{b}_v\mathbf{1}^T), \tag{5.2}$$

$\sigma$ is the rectified linear unit (ReLU) activation function and $\mathbf{C}_{uu}$ and $\mathbf{C}_{vv}$ are the correlation matrices of two trainable variables $\mathbf{U} \in \mathbb{R}^{n_1\times k_o}$ and $\mathbf{V} \in \mathbb{R}^{n_2\times k_o}$, respectively[2]. Here, we compute the correlation matrix $\mathbf{C}_{uu}$ (respective $\mathbf{C}_{vv}$) using the Euclidean distance matrix $\mathbf{D}_u$ of the trainable variable $\mathbf{U}_o \in \mathbb{R}^{n_1\times k_o}$ (respective $\mathbf{D}_v$ of $\mathbf{V}_o \in \mathbb{R}^{n_2\times k_o}$) as [23]

$$\mathbf{C}_{uu} = -\frac{1}{\beta_u}(\mathbf{D}_u - \alpha_u\mathbf{1}\mathbf{1}^T), \tag{5.3}$$

where $\alpha_u = \frac{1}{n_1^2}\sum_{ij}[\mathbf{D}_u]_{ij}$ and $\beta_u = \max_{ij}[\mathbf{D}_u]_{ij}$ are used to make $\mathbf{D}_u$ centralized and normalized.

### 5.2.2 Multilayer GNN

One important issue in the GNN-based LRMC approach is to define a graph-based convolution operation to extract the meaningful features across the graph domain. Since

[2]Note that the dimension $k_o$ of $\mathbf{U}_o$ is not necessarily the same as the dimension $k$ of $\mathbf{U}$. This is because in practice the graph models of $\mathbf{M}$ might be characterized using more degree of freedom (DOF) than the DOF of $\mathbf{M}$ itself.

the input data $G_r$ and $G_c$ do not lie on regular lattices like images, classical convolutional neural network (CNN) cannot be directly applied to $G_r$ and $G_c$. One practical way is to define the convolution operation in the Fourier domain of the graph.

To be specific, the Fourier transform of a graph can be computed using the (normalized) graph Laplacian. For simplicity, we show the useful expressions related to $\mathcal{G}_r$. The similar expressions of $\mathcal{G}_c$ can be easily derived. Let $\mathbf{R}_r$ be the graph Laplacian of $G_r$ (i.e., $\mathbf{R}_r = \mathbf{I} - \mathbf{D}_r^{-1/2}\mathbf{W}_r\mathbf{D}_r^{-1/2}$ where $\mathbf{D}_r = \text{diag}(\mathbf{W}_r\mathbf{1}_{n_2\times 1})$) [102]. Then, the graph Fourier transform $\mathcal{F}_r(\mathbf{U})$ is defined as

$$\mathcal{F}_r(\mathbf{U}) = \mathbf{Q}_r^T\mathbf{U}, \tag{5.4}$$

where $\mathbf{R}_r = \mathbf{Q}_r\mathbf{\Lambda}_r\mathbf{Q}_r^T$ is the eigen-decomposition of the graph Laplacian $\mathbf{R}_r$ [102]. Also, the inverse graph Fourier transform $\mathcal{F}_r^{-1}(\mathbf{U}')$ of $\mathbf{U}'$ is defined as[3]

$$\mathcal{F}_r^{-1}(\mathbf{U}') = \mathbf{Q}_r\mathbf{U}'. \tag{5.5}$$

Let $g(\boldsymbol{\theta}) \in \mathbb{R}^{n_1}$ be the filter characerized by the parameter vector $\boldsymbol{\theta} \in \mathbb{R}^q$, then the output $\mathbf{Z}$ of the graph-based convolution is defined as [102, 101]

$$\mathbf{Z} = g(\boldsymbol{\theta}) * \mathbf{U} = \mathcal{F}_r^{-1}(\mathcal{F}_r(g(\boldsymbol{\theta})) \odot \mathcal{F}_r(\mathbf{U})), \tag{5.6}$$

where $\odot$ is the Hadamard product (element-wise product). From (5.4) and (5.5), (5.6) can be expressed as

$$\begin{aligned}\mathbf{Z} &= \mathbf{Q}_r(\mathcal{F}_r(g(\boldsymbol{\theta})) \odot \mathbf{Q}_r^T\mathbf{U}) \\ &= \mathbf{Q}_r\text{diag}(\mathcal{F}_r(g(\boldsymbol{\theta})))\mathbf{Q}_r^T\mathbf{U} \\ &= \mathbf{Q}_r\mathbf{G}\mathbf{Q}_r^T\mathbf{U},\end{aligned} \tag{5.7}$$

where $\mathbf{G} = \text{diag}(\mathcal{F}_r(g(\boldsymbol{\theta})))$ is the diagonal matrix of filter parameters defined in the graph Fourier domain. Conventional settings of $\mathbf{G}$ include:

- *Non-parametric filter*: $\mathbf{G} = \text{diag}(\boldsymbol{\theta})$.

[3] One can easily check that $\mathcal{F}_r^{-1}(\mathcal{F}_r(\mathbf{U})) = \mathbf{U}$ and $\mathcal{F}_r(\mathcal{F}_r^{-1}(\mathbf{U}')) = \mathbf{U}'$.

- *Polynomial filter of Laplacian eigenvalues*: $\mathbf{G} = \sum_{i=0}^{q-1} \theta_i \mathbf{\Lambda}_r^i$ From (5.7), we have

$$\mathbf{Z} = \sum_{i=0}^{q-1} \theta_i \mathbf{Q}_r \mathbf{\Lambda}_r^i \mathbf{Q}_r^T \mathbf{U} = \sum_{i=0}^{q-1} \theta_i \mathbf{R}_r^i \mathbf{U} \tag{5.8}$$

- *Polynomial filter of Chebyshev basis*: $\mathbf{G} = \sum_{i=0}^{q-1} \theta_i T_i(\widetilde{\mathbf{\Lambda}}_r)$ where $T_i$ is the Chebyshev polynomial of the $i$-th order[4] and $\widetilde{\mathbf{\Lambda}}_r = \frac{2}{\max_i \lambda_i} \mathbf{\Lambda}_r - \mathbf{I}$. Then (5.7) can be expressed as

$$\mathbf{Z} = \sum_{i=0}^{q-1} \theta_i \mathbf{Q}_r T_i(\widetilde{\mathbf{\Lambda}}_r) \mathbf{Q}_r^T \mathbf{U}. \tag{5.9}$$

Among the possible choices of $\mathbf{G}$, non-parametric filter is the most simple and straightforward way. However, since this filter performs on all the graph vertices, it has no locality property as in the conventional convolutional neural network (CNN). In contrast, it has been shown that polynomial filters are exact $q$-localized filters[5] [102, 101]. Also, the computational cost of the Chebyshev polynomial filter can be further reduced using a recurrent update procedure [101].

Note that (5.8) and (5.9) use the symmetric normalized Laplacian $\mathbf{R}_r$ to learn the dissimilarity between the rows of $\mathbf{U}$. It can be shown that the eigenvalue spectrum of $\mathbf{R}_r$ carries a notion of frequency on the graph domain. However, it might not handle the DC component well. To be specific, let $\mathbf{y}_t$ be the $t$-th row of $\mathbf{Y} = \mathbf{R}_r \mathbf{U}$, which is the extracted feature in the neighborhood of $\mathbf{u}_t$. Then we have

$$\mathbf{y}_t = (1 - \sum_{j:(t,j)\in E_r} \lambda_{tj}) \mathbf{u}_t + \sum_{j:(t,j)\in E_r} \lambda_{tj} (\mathbf{u}_t - \mathbf{u}_j), \tag{5.10}$$

[4] Chebyshev polynomial $T_i(\mathbf{A})$ of order $i$ ($i \geq 2$) is computed using the recurrent expression $T_i(\mathbf{A}) = 2\mathbf{A}T_{i-1}(\mathbf{A}) - T_{i-2}(\mathbf{A})$ with $T_0(\mathbf{A}) = \mathbf{I}$ and $T_1(\mathbf{A}) = \mathbf{A}$ where $\mathbf{A}$ is a diagonal matrix.

[5] The filter affect on the neigboring nodes of a node the $(m, n)$-th entry $[\mathbf{R}_r^q]_{mn}$ vanishes (i.e., $[\mathbf{R}_r^q]_{mn} = 0$) when the minimum number of edges connecting two vertices $m$-th and $n$-th is larger than $q$.

where $\lambda_{tj}$ is the $(t,j)$-th normalized weight of $\mathbf{\Lambda} = \mathbf{D}_r^{-1/2}\mathbf{W}_r\mathbf{D}_r^{-1/2}$. In (5.10), the DC component $(1 - \sum_j \lambda_{tj})\mathbf{u}_t$ might vanish when $\sum_j \lambda_{tj} = 1$. Also, in most of practical situations, it might require that this DC component is dominant in (5.10) to stabilize the learning process. To overcome these issues, we proposed multilayer GNN scheme using an extended version of the polynomial filters based on a generalized Laplacian. That is,

$$\widetilde{\mathbf{R}}_r = (1+\tau)\mathbf{I} - \tau\mathbf{D}_r^{-1}\mathbf{W}_r = \mathbf{I} + \tau(\mathbf{I} - \mathbf{D}_r^{-1}\mathbf{W}_r), \tag{5.11}$$

where $\tau$ is some tuning parameter. Using (5.11), we reformulate (5.10) as

$$\widetilde{\mathbf{y}}_t = \mathbf{u}_t + \sum_{j:(t,j)\in E_r} \tau\widetilde{\lambda}_{tj}(\mathbf{u}_t - \mathbf{u}_j), \tag{5.12}$$

where $\widetilde{\mathbf{y}}_t$ is the $t$-th row of $\widetilde{\mathbf{Y}} = \widetilde{\mathbf{R}}_r\mathbf{U}$ and $\widetilde{\lambda}_{ij}$ is the normalized weights of $\mathbf{D}_r^{-1}\mathbf{W}_r$. We note that the DC component is also included into (5.8) and (5.9) using the multiple coefficient $\theta_0$. However, our approach is distinct from these since we maintain the DC component based on the generalized Laplacian matrix (see (5.11)).

### 5.2.3 Output Model

In GNN-based LRMC, output model is a mapping between the extracted feature and the reconstructed low-rank matrix. To be specific, let $(\widehat{\mathbf{U}}, \widehat{\mathbf{V}})$ be the output of the GNN. Then, the reconstructed matrix is

$$\widehat{\mathbf{M}} = \phi(\widehat{\mathbf{U}}\widehat{\mathbf{V}}^T), \tag{5.13}$$

where $\phi$ is the function representation of the output model. The output model heavily depends on the data type of $\mathbf{M}$. For example, in recommendation systems, the entry of $\mathbf{M}$ is an integer ranging from 1 to 5. An output model based on a recurrent neural network (RNN) has been proposed. In our work, to avoid the additional model complexity, we are interested in the simple way to reconstruct the desired low-rank matrix as $\widehat{\mathbf{M}} = \sigma(s\widehat{\mathbf{U}}\widehat{\mathbf{V}}^T + b\mathbf{1}\mathbf{1}^T)$ where $s$ is a scale parameter and $b$ is a offset constant.

### 5.2.4 Training Cost Function

In our approach, the training cost function is based on the Frobenius norm minimization in LRMC. Let $\Omega$ be the set of indices of known entries, and also let $P_\Omega$ be the sampling operator defined as

$$[P_\Omega(\mathbf{A})]_{ij} = \begin{cases} a_{ij} & \text{if } (i,j) \in \Omega \\ 0 & \text{otherwise} \end{cases}. \tag{5.14}$$

Then the training cost function of $\widehat{\mathbf{U}}$ and $\widehat{\mathbf{V}}$ is

$$\begin{aligned} l(\widehat{\mathbf{U}}, \widehat{\mathbf{V}}) = &\sum_{(i,j)} w_{r,ij} \|\widehat{\mathbf{u}}_i - \widehat{\mathbf{u}}_j\|_2 + \sum_{(i,j)} w_{c,ij} \|\widehat{\mathbf{v}}_i - \widehat{\mathbf{v}}_j\|_2 \\ &+ \rho \|P_\Omega(\sum_{i=1}^{r} \widehat{\mathbf{u}}_i \widehat{\mathbf{v}}_i^T) - P_\Omega(\mathbf{M})\|_F, \end{aligned} \tag{5.15}$$

where $\rho$ is a regularization parameter and $w_{r,ij}$ and $w_{c,ij}$ are the entries of $\mathbf{W}_r$ and $\mathbf{W}_c$, respectively. In other words, we find $\widehat{\mathbf{U}}$ and $\widehat{\mathbf{V}}$ such that the Euclidean distance between the connected vertices is minimized.

In our adaptive model, we use a pre-train cost function $\kappa(\mathbf{U}_o, \mathbf{V}_o)$ to initialize the weight matrices $\mathbf{W}_r$ and $\mathbf{W}_c$. That is, we define

$$\kappa(\mathbf{U}_o, \mathbf{V}_o) = \sum_{ij} |w_{r,ij} - w_{ro,ij}| + |w_{c,ij} - w_{co,ij}|, \tag{5.16}$$

where $w_{ro,ij}$ and $w_{co,ij}$ are the entries of the given weight matrices $\mathbf{W}_{ro}$ and $\mathbf{W}_{co}$, respectively. We note that the $\ell_1$-norm in (5.16) is useful to enhance the sparsity of $\mathbf{W}_r$ and $\mathbf{W}_c$ as long as $\mathbf{W}_{ro}$ and $\mathbf{W}_{co}$ are given sparse matrices.

## 5.3 Numerical Evaluation

In this section, we investigate numerical performance of the proposed GNN-LRMC and compare it with the state-of-the-art matrix completion techniques, including ASD, NIHT, SET, SVT, sRMGCNN, and TNNR-APGL [22, 73]. As performance measures,

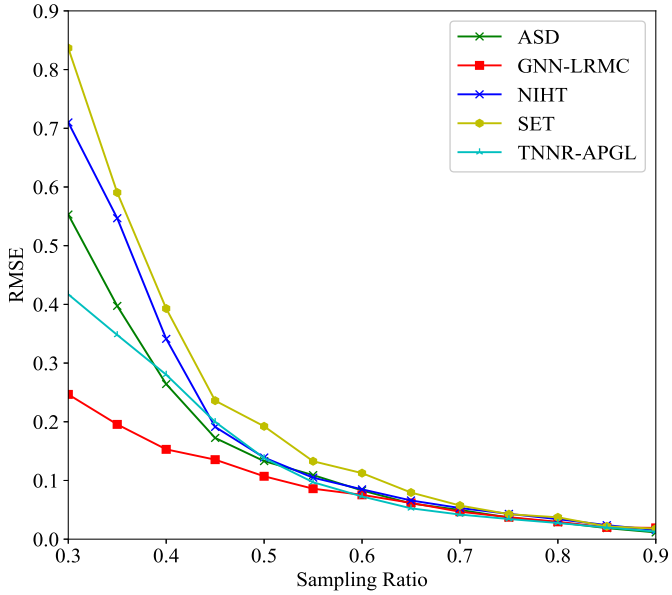



Figure 5.4: RMSE performance of the LRMC algorithms.

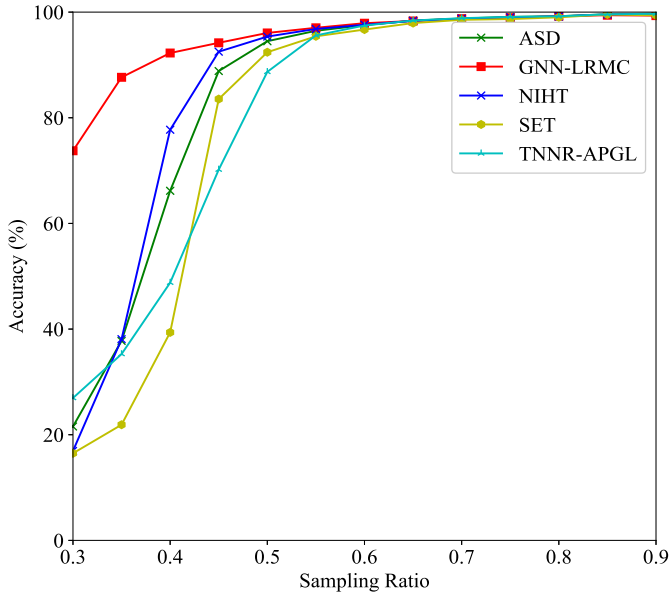



Figure 5.5: Accuracy performance of the LRMC algorithms.

Table 5.1: RMSE performance of the matrix completion algorithms using Netflix dataset.

| | n = 5000 | n = 1000 | n = 500 |
|---|---|---|---|
| ASD | 0.2968 | 0.6892 | 0.8344 |
| GNN-LRMC | 0.2233 | 0.2176 | 0.2375 |
| NIHT | 0.2921 | 0.6309 | 0.7563 |
| sRMGCNN | 0.2347 | 0.2318 | 0.2497 |
| TNNR-APGL | 0.2969 | 0.8314 | 0.8921 |

we use the relative mean square errors (RMSE) and the reconstruction accuracy $L$, which are defined as

$$RMSE = \frac{\|\widehat{\mathbf{M}} - \mathbf{M}\|_F}{\|\mathbf{M}\|_F} \text{ and } L = \frac{1}{|T|} \sum_{(i,j)\in T} I_{\{\widehat{m}_{ij}=m_{ij}\}},$$

where $\mathbf{M}$ is the desired low-rank matrix, $\widehat{\mathbf{M}}$ is the reconstructed matrix, and $T$ is the index set of test entries with the cardinality $|T|$. Here, we define $I_A$ as the indicator function satisfying $I_{\{A\}} = 1$ if $A$ holds true, and otherwise $0$. For the implementation of the proposed GNN-LRMC, we simply use two GNN layers with the filter size $q = 5$ (see (5.8) and (5.9)).

In our simulation, we generate the entries of $\mathbf{M} \in \mathbb{R}^{n_1 \times n_2}$ at random, each entry an integer ranging from 1 to 5. We set $n_1 = n_2 = 300$ and perform at least 1000 independent trials. The sampling ratio is defined as the fraction of the training entries to the total entries. From the simulation results, we observe that the proposed GNN-LRMC outperforms the conventional techniques, resulting in 50% improvement of the RMSE performance in the small regime of the sampling ratio (see Fig. 5.4 and 5.5).

We also test the recovery performance of the proposed scheme using the Netflix database [1]. In our experiment, we reconstruct $n \times n$ rating matrix $\mathbf{M}$ using 30% of

the known entries and run simulation for $n = 5000$, $1000$, and $500$. From the simulation results, we observe that the proposed GNN-LRMC outperforms the conventional techniques (see Table 5.1).

# Chapter 6

# Conculsion

In this thesis, taking into account of the availability of the rank information, we naturally classified state-of-the-art LRMC techniques into two main categories. In fact, when the rank of a desired matrix is unknown, we formulated the LRMC problem as the NNM problem and discussed several NNM-based LRMC techniques such as SDP-based NNM, SVT, and truncated NNM. When the rank of an original matrix is known a priori, the LRMC problem can be modeled as the FNM problem. We discussed various FNM-based LRMC techniques (e.g., greedy algorithms, alternating projection methods, and optimization over Riemannian manifold) and also presented fundamental issues and principles that one needs to be aware of when solving the LRMC problem.

In particular, we have proposed the LRMC algorithms to exploit the underlying structure of the desired low-rank matrix so that we can improve the recovery performance of LRMC in real-life applications, including IoT localization and recommendation systems. In IoT localization, we have proposed the LRM-CG algorithm to recover the Euclidean distance matrix (and therefore the location map) from partially observed distance information. In solving the Frobenius norm minimization problem with a rank constraint, we expressed the Euclidean distance matrix as a function of the fixed rank positive semidefinite matrix. By capitalizing on the Riemannian manifold structure for this set of matrices, we could solve the low-rank matrix completion prob-

lem using the modified nonlinear conjugate gradient algorithm. In the scenario when the observed distances contaminated by outliers, we proposed an extension of LRM-CG which is efficient in the outlier detection and the reconstruction of missing entry using an alternating algorithm. We have shown from the recovery condition analysis that the proposed LRM-CG algorithm converges to the original Euclidean distance matrix in the sampling space under the extended Wolfe's conditions. We have also shown from numerical experiments that the LRM-CG algorithm is effective in recovering the original Euclidean distance matrix while exhibiting reasonable computational cost scalable to the matrix dimension. In recommendation systems, we have proposed the GNN-LRMC scheme, which can nicely combine the multilayer GNN and the adaptive model of the graph weight matrices. Empirical study shows our proposed GNN-LRMC can significantly improve the accuracy of the low-rank matrix reconstruction and outperform conventional techniques.

While in our work, we apply LRMC to IoT localization and recommendation systems, our proposed algorithms can be easily extended to other LRMC applications in which the low-rank matrix has some underlying non-Euclidean structure (e.g., graph or manifold structure). Also, given the importance of the location-aware applications and services in the IoT era, we believe that the proposed LRM-CG algorithm will be a useful tool for various localization problems. While our work focused primarily on the centralized localization scenario, extension to the distributed and cooperative network scenarios would also be interesting direction worth pursuing.

# Chapter A

# Proof of Lemma 6

**Proof:** Let $\dot{\mathbf{Y}}$ be a tangent vector at $\mathbf{Y} \in \widetilde{\mathcal{Y}}$, i.e., $\dot{\mathbf{Y}} \in T_{\mathbf{Y}}\widetilde{\mathcal{Y}}$. Also, let

$$\begin{aligned} \mathcal{S} &= \left\{ \begin{bmatrix} \mathbf{Q} & \mathbf{Q}_\perp \end{bmatrix} \begin{bmatrix} \mathbf{B} & \mathbf{C}^T \\ \mathbf{C} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{Q}^T \\ \mathbf{Q}_\perp^T \end{bmatrix} : \right. \\ &\qquad \left. \mathbf{B}^T = \mathbf{B} \in \mathbb{R}^{k\times k}, \mathbf{C} \in \mathbb{R}^{(n-k)\times k} \right\}. \end{aligned} \tag{A.1}$$

Then, what we need to show is that $\dot{\mathbf{Y}}$ is an element in $\mathcal{S}$. By the definition of $T_{\mathbf{Y}}\widetilde{\mathcal{Y}}$, there exists a curve $\gamma(t)$ in $\widetilde{\mathcal{Y}}$ such that $\mathbf{Y} = \gamma(0)$ and $\dot{\mathbf{Y}} = \frac{d}{dt}\gamma(t)\Big|_{t=0}$. For convenience, we denote $\gamma(t) = \mathbf{Z}(t)$. Using the eigenvalue decomposition $\mathbf{Z}(t) = \mathbf{Q}(t)\mathbf{\Lambda}(t)\mathbf{Q}(t)^T$, we have

$$\begin{aligned} \dot{\mathbf{Y}} &= \frac{d}{dt}\mathbf{Z}(t)\Big|_{t=0} \\ &= \dot{\mathbf{Q}}\mathbf{\Lambda}\mathbf{Q}^T + \mathbf{Q}\dot{\mathbf{\Lambda}}\mathbf{Q}^T + \mathbf{Q}\mathbf{\Lambda}\dot{\mathbf{Q}}^T \end{aligned} \tag{A.2}$$

Since $\mathbf{Q}^T\mathbf{Q} = \mathbf{I}$, $\mathbf{Q}$ is an element of the Stiefel manifold $\mathcal{Q} = \{\mathbf{A} : \mathbf{A}^T\mathbf{A} = \mathbf{I}, \mathbf{A} \in \mathbb{R}^{n\times k}\}$. The tangent vector of $\mathcal{Q}$ at the point $\mathbf{Q}$ is given by [63, Example 3.5.2]

$$\dot{\mathbf{Q}} = \mathbf{Q}\mathbf{\Omega} + \mathbf{Q}_\perp\mathbf{K}, \tag{A.3}$$

where $\mathbf{\Omega}$ is the $k \times k$ skew-symmetric matrix (i.e., $\mathbf{\Omega} = -\mathbf{\Omega}^T$) and $\mathbf{K}$ is the $(n-k) \times (n-k)$ matrix. From (A.2) and (A.3), we have

$$\begin{aligned}\dot{\mathbf{Y}} &= (\mathbf{Q\Omega} + \mathbf{Q}_\perp\mathbf{K})\mathbf{\Lambda}\mathbf{Q}^T + \mathbf{Q}\dot{\mathbf{\Lambda}}\mathbf{Q}^T \\ &\qquad + \mathbf{Q}\mathbf{\Lambda}(\mathbf{Q\Omega} + \mathbf{Q}_\perp\mathbf{K}) \\ &= \begin{bmatrix}\mathbf{Q} & \mathbf{Q}_\perp\end{bmatrix} \begin{bmatrix}\mathbf{\Omega\Lambda} + \dot{\mathbf{\Lambda}} + \mathbf{\Lambda\Omega}^T & \mathbf{\Lambda}\mathbf{K}^T \\ \mathbf{K\Lambda} & \mathbf{0}\end{bmatrix} \begin{bmatrix}\mathbf{Q}^T \\ \mathbf{Q}_\perp^T\end{bmatrix}.\end{aligned}$$

If we denote $\mathbf{B} = \mathbf{\Omega\Lambda} + \dot{\mathbf{\Lambda}} + \mathbf{\Lambda\Omega}^T$ and $\mathbf{C} = \mathbf{K\Lambda}$, then one can easily see that $\dot{\mathbf{Y}} \in T_{\mathbf{Y}}\widetilde{\mathcal{Y}} \subseteq \mathcal{S}$.

To complete the proof, we need to show that $\mathcal{S} = T_{\mathbf{Y}}\widetilde{\mathcal{Y}}$. This implies that two vector spaces $\mathcal{S}$ and $T_{\mathbf{Y}}\widetilde{\mathcal{Y}}$ have the same dimension. Indeed, from (A.1), we can easily check that the dimension[1] of $\mathcal{S}$ is $\frac{1}{2}k(2n-k+1)$, which is the dimension of $\widetilde{\mathcal{Y}}$ [81, Proposition 1.1]. □

[1]The dimension of $\mathcal{S}$ is obtained by counting the number of independent entries of an element in $\mathcal{S}$. Since $\mathbf{B}$ is a $k \times k$ symmetric matrix, the number of independent entries of $\mathbf{B}$ is $\frac{k(k+1)}{2}$. In addition, since $\mathbf{C}$ is an arbitrary $(n-k) \times k$ matrix, the number of independent entries of $\mathbf{C}$ is $(n-k)k$. Thus, the dimension of $\mathcal{S}$ is $\frac{k(k+1)}{2} + (n-k)k = \frac{1}{2}k(2n-k+1)$.

# Chapter B

# Proof of Theorem 7

**Proof:** First, we partition a matrix $\mathbf{A}$ into two parts: $\mathbf{A} = \mathbf{A}_1 + \mathbf{A}_2$ where $\mathbf{A}_1 \in T_{\mathbf{Y}}\widetilde{\mathcal{Y}}$ and $\mathbf{A}_2 \in (T_{\mathbf{Y}}\widetilde{\mathcal{Y}})^{\perp}$. Then, it is clear that $P_{T_{\mathbf{Y}}\widetilde{\mathcal{Y}}}(\mathbf{A}) = \mathbf{A}_1$ and thus the goal is to find out the closed form expression of $\mathbf{A}_1$. From Lemma 6, there exist a symmetric matrix $\mathbf{B} \in \mathbb{R}^{k\times k}$ and a matrix $\mathbf{C} \in \mathbb{R}^{(n-k)\times k}$ such that

$$\mathbf{A}_1 = \widetilde{\mathbf{Q}} \begin{bmatrix} \mathbf{B} & \mathbf{C}^T \\ \mathbf{C} & \mathbf{0} \end{bmatrix} \widetilde{\mathbf{Q}}^T, \tag{B.1}$$

where $\widetilde{\mathbf{Q}} = \begin{bmatrix} \mathbf{Q} & \mathbf{Q}_{\perp} \end{bmatrix}$. Since $< \mathbf{A}_1, \mathbf{A}_2 >= 0$, we have

$$\begin{aligned} 0 &= < \mathbf{A}_1, \mathbf{A}_2 > \\ &= < \mathbf{A}_1, \mathbf{A} - \mathbf{A}_1 > \\ &= < \widetilde{\mathbf{Q}} \begin{bmatrix} \mathbf{B} & \mathbf{C}^T \\ \mathbf{C} & \mathbf{0} \end{bmatrix} \widetilde{\mathbf{Q}}^T, \mathbf{A} - \widetilde{\mathbf{Q}} \begin{bmatrix} \mathbf{B} & \mathbf{C}^T \\ \mathbf{C} & \mathbf{0} \end{bmatrix} \widetilde{\mathbf{Q}}^T > \\ &= < \begin{bmatrix} \mathbf{B} & \mathbf{C}^T \\ \mathbf{C} & \mathbf{0} \end{bmatrix}, \widetilde{\mathbf{Q}}^T \mathbf{A} \widetilde{\mathbf{Q}} - \begin{bmatrix} \mathbf{B} & \mathbf{C}^T \\ \mathbf{C} & \mathbf{0} \end{bmatrix} > . \end{aligned} \tag{B.2}$$

Let

$$\begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix} = \widetilde{\mathbf{Q}}^T \mathbf{A} \widetilde{\mathbf{Q}} = \begin{bmatrix} \mathbf{Q}^T \mathbf{A} \mathbf{Q} & \mathbf{Q}^T \mathbf{A} \mathbf{Q}_{\perp} \\ \mathbf{Q}_{\perp}^T \mathbf{A} \mathbf{Q} & \mathbf{Q}_{\perp}^T \mathbf{A} \mathbf{Q}_{\perp} \end{bmatrix}, \tag{B.3}$$

then we have

$$\begin{aligned} 0 &= < \begin{bmatrix} \mathbf{B} & \mathbf{C}^T \\ \mathbf{C} & \mathbf{0} \end{bmatrix}, \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix} - \begin{bmatrix} \mathbf{B} & \mathbf{C}^T \\ \mathbf{C} & \mathbf{0} \end{bmatrix} > \\ &= < \mathbf{B}, \mathbf{A}_{11} - \mathbf{B} > + < \mathbf{C}, \frac{1}{2}(\mathbf{A}_{21} + \mathbf{A}_{12}^T) - \mathbf{C} >, \end{aligned}$$

where the equality is because

$$< \begin{bmatrix} \boldsymbol{\alpha}_1 & \boldsymbol{\alpha}_2 \\ \boldsymbol{\alpha}_3 & \boldsymbol{\alpha}_4 \end{bmatrix}, \begin{bmatrix} \boldsymbol{\beta}_1 & \boldsymbol{\beta}_2 \\ \boldsymbol{\beta}_3 & \boldsymbol{\beta}_4 \end{bmatrix} >= \sum_{i=1}^{4} < \boldsymbol{\alpha}_i, \boldsymbol{\beta}_i > .$$

Since $\mathbf{B}$ and $\mathbf{C}$ are chosen arbitrarily, we should have

$$< \mathbf{B}, \mathbf{A}_{11} - \mathbf{B} >= 0, \tag{B.4}$$

$$< \mathbf{C}, \frac{1}{2}(\mathbf{A}_{21} + \mathbf{A}_{12}^T) - \mathbf{C} >= 0. \tag{B.5}$$

First, it is clear from (B.5) that

$$\mathbf{C} = \frac{1}{2}(\mathbf{A}_{21} + \mathbf{A}_{12}^T). \tag{B.6}$$

Next, noting that $\mathbf{A}_{11} = \text{Sym}(\mathbf{A}_{11}) + \text{Skew}(\mathbf{A}_{11})$, (B.4) becomes

$$\begin{aligned} 0 &= < \mathbf{B}, \mathbf{A}_{11} - \mathbf{B} > \\ &= < \mathbf{B}, \text{Sym}(\mathbf{A}_{11}) - \mathbf{B} > + < \mathbf{B}, \text{Skew}(\mathbf{A}_{11}) > \\ &\overset{(a)}{=} < \mathbf{B}, \text{Sym}(\mathbf{A}_{11}) - \mathbf{B} >, \end{aligned}$$

where (a) is because $\mathbf{B}$ is the symmetric matrix (i.e., $\mathbf{B} = \text{Sym}(\mathbf{B})$) and $< \text{Sym}(\mathbf{C}), \text{Skew}(\mathbf{D}) >= 0$ for any matrices $\mathbf{C}$ and $\mathbf{D}$. Since $\mathbf{B}$ is any symmetric matrix, we should have

$$\mathbf{B} = \text{Sym}(\mathbf{A}_{11}) = \frac{1}{2}(\mathbf{A}_{11} + \mathbf{A}_{11}^T). \tag{B.7}$$

Substituting (B.6) and (B.7) into (B.1), we have

$$\mathbf{A}_1 = \widetilde{\mathbf{Q}} \begin{bmatrix} \frac{1}{2}(\mathbf{A}_{11} + \mathbf{A}_{11}^T) & \frac{1}{2}(\mathbf{A}_{21}^T + \mathbf{A}_{12}) \\ \frac{1}{2}(\mathbf{A}_{21} + \mathbf{A}_{12}^T) & \mathbf{0} \end{bmatrix} \widetilde{\mathbf{Q}}^T, \tag{B.8}$$

where $\mathbf{A}_{11}$, $\mathbf{A}_{12}$, and $\mathbf{A}_{21}$ are the components of $\mathbf{A}$ (see (B.3)).

Now, what remains is to find a closed form expression for $\mathbf{A}_1$ in terms of $\mathbf{A}$. First, we can rewrite (B.8) as

$$\begin{aligned}
\mathbf{A}_1 &= \widetilde{\mathbf{Q}} \begin{bmatrix} \frac{1}{2}(\mathbf{A}_{11} + \mathbf{A}_{11}^T) & \frac{1}{2}(\mathbf{A}_{21}^T + \mathbf{A}_{12}) \\ \frac{1}{2}(\mathbf{A}_{21} + \mathbf{A}_{12}^T) & \frac{1}{2}(\mathbf{A}_{22} + \mathbf{A}_{22}^T) \end{bmatrix} \widetilde{\mathbf{Q}}^T \\
&\quad -\widetilde{\mathbf{Q}} \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \frac{1}{2}(\mathbf{A}_{22} + \mathbf{A}_{22}^T) \end{bmatrix} \widetilde{\mathbf{Q}}^T \\
&= \frac{1}{2}\widetilde{\mathbf{Q}}( \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix} + \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix}^T )\widetilde{\mathbf{Q}}^T \\
&\quad -\frac{1}{2}\mathbf{Q}_\perp(\mathbf{A}_{22} + \mathbf{A}_{22}^T)\mathbf{Q}_\perp^T. \qquad \text{(B.9)}
\end{aligned}$$

Substituting (B.3) into (B.9), we have

$$\begin{aligned}
\mathbf{A}_1 &= \frac{1}{2}\widetilde{\mathbf{Q}}\widetilde{\mathbf{Q}}^T(\mathbf{A} + \mathbf{A}^T)\widetilde{\mathbf{Q}}\widetilde{\mathbf{Q}}^T \\
&\quad -\frac{1}{2}\mathbf{Q}_\perp(\mathbf{Q}_\perp^T\mathbf{A}\mathbf{Q}_\perp + \mathbf{Q}_\perp^T\mathbf{A}^T\mathbf{Q}_\perp)\mathbf{Q}_\perp^T \\
&= \frac{1}{2}(\mathbf{A} + \mathbf{A}^T) - \frac{1}{2}\mathbf{Q}_\perp\mathbf{Q}_\perp^T(\mathbf{A} + \mathbf{A}^T)\mathbf{Q}_\perp\mathbf{Q}_\perp^T \\
&= \text{Sym}(\mathbf{A}) - \mathbf{Q}_\perp\mathbf{Q}_\perp^T\text{Sym}(\mathbf{A})\mathbf{Q}_\perp\mathbf{Q}_\perp^T.
\end{aligned}$$

Since $\mathbf{Q}\mathbf{Q}^T + \mathbf{Q}_\perp\mathbf{Q}_\perp^T = \mathbf{I}$ and $\mathbf{P}_\mathbf{Q} = \mathbf{Q}\mathbf{Q}^T$, we have

$$\begin{aligned}
\mathbf{A}_1 &= \text{Sym}(\mathbf{A}) - (\mathbf{I} - \mathbf{Q}\mathbf{Q}^T)\text{Sym}(\mathbf{A})(\mathbf{I} - \mathbf{Q}\mathbf{Q}^T) \\
&= \mathbf{P}_\mathbf{Q}\text{Sym}(\mathbf{A}) + \text{Sym}(\mathbf{A})\mathbf{P}_\mathbf{Q} - \mathbf{P}_\mathbf{Q}\text{Sym}(\mathbf{A})\mathbf{P}_\mathbf{Q},
\end{aligned}$$

which is the desired result. □

# Chapter C

# Proof of Lemma 8

**Proof:** Our goal is to find a simple expression of the retraction operator $R_{\mathbf{Y}}(\mathbf{B})$. First, since $\mathbf{Z} = \text{Sym}(\mathbf{Z})$ for $\mathbf{Z} \in \widetilde{\mathcal{Y}}$, we have

$$\begin{aligned}
\|\mathbf{Y}+\mathbf{B}-\mathbf{Z}\|_F^2 &= \|\mathbf{Y}+\mathbf{B}-\text{Sym}(\mathbf{Z})\|_F^2 \\
&= \|\text{Skew}(\mathbf{Y}+\mathbf{B})+\text{Sym}(\mathbf{Y}+\mathbf{B})-\text{Sym}(\mathbf{Z})\|_F^2 \qquad \text{(C.1)} \\
&= \|\text{Skew}(\mathbf{Y}+\mathbf{B})\|_F^2+\|\text{Sym}(\mathbf{Y}+\mathbf{B})-\text{Sym}(\mathbf{Z})\|_F^2 \qquad \text{(C.2)} \\
&= \|\text{Skew}(\mathbf{Y}+\mathbf{B})\|_F^2+\|\text{Sym}(\mathbf{Y}+\mathbf{B})-\mathbf{Z}\|_F^2, \qquad \text{(C.3)}
\end{aligned}$$

where (C.1) is because $\text{Sym}(\mathbf{A})+\text{Skew}(\mathbf{A}) = \mathbf{A}$ and (C.2) is because $< \text{Skew}(\mathbf{C}), \text{Sym}(\mathbf{D}) >= 0$ for any $\mathbf{C}$ and $\mathbf{D}$. Since the first term in (C.3) is unrelated to $\mathbf{Z}$, it is clear that

$$R_{\mathbf{Y}}(\mathbf{B}) = \arg\min_{\mathbf{Z}\in\widetilde{\mathcal{Y}}} \|\text{Sym}(\mathbf{Y}+\mathbf{B})-\mathbf{Z}\|_F.$$

Using the eigenvalue decomposition $\text{Sym}(\mathbf{Y}+\mathbf{B}) = \mathbf{K}\boldsymbol{\Sigma}\mathbf{K}^T$, we have

$$\begin{aligned}
R_{\mathbf{Y}}(\mathbf{B}) &= \arg\min_{\mathbf{Z}\in\widetilde{\mathcal{Y}}} \|\mathbf{K}\boldsymbol{\Sigma}\mathbf{K}^T-\mathbf{Z}\|_F \\
&= \arg\min_{\mathbf{Z}\in\widetilde{\mathcal{Y}}} \|\mathbf{K}\left(\boldsymbol{\Sigma}-\mathbf{K}^T\mathbf{Z}\mathbf{K}\right)\mathbf{K}^T\|_F \\
&\overset{(a)}{=} \arg\min_{\mathbf{Z}\in\widetilde{\mathcal{Y}}} \|\boldsymbol{\Sigma}-\mathbf{K}^T\mathbf{Z}\mathbf{K}\|_F,
\end{aligned}$$

where (a) is because $\|\mathbf{K}\mathbf{U}\mathbf{K}^T\|_F^2 = tr(\mathbf{K}\mathbf{U}^T\mathbf{K}^T\mathbf{K}\mathbf{U}\mathbf{K}^T) = \|\mathbf{U}\|_F^2$ for any matrix $\mathbf{U}$. Now let $R_{\mathbf{Y}}(\mathbf{B}) = \mathbf{Z}^*$, $\mathbf{\Sigma}^* = \mathbf{K}\mathbf{Z}^*\mathbf{K}^T$, and $\mathbf{Q} = \mathbf{K}^T\mathbf{Z}\mathbf{K}$, then

$$\mathbf{\Sigma}^* \quad = \quad \mathbf{K}\mathbf{Z}^*\mathbf{K}^T = \arg\min_{\mathbf{Q}} \|\mathbf{\Sigma} - \mathbf{Q}\|_F. \tag{C.4}$$

Since $\mathbf{\Sigma}$ is a diagonal matrix, $\mathbf{\Sigma}^*$ should also be a diagonal matrix. Also, $\mathbf{\Sigma}^* \succeq 0$ and rank$(\mathbf{\Sigma}^*) = k$.[1] Thus, $\mathbf{\Sigma}^*$ is a diagonal matrix with only $k$ positive entries and the rest being zero. That is,

$$\mathbf{\Sigma}^* = \begin{bmatrix} \sigma_1 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & \sigma_k & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \end{bmatrix}, \tag{C.5}$$

where $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_k > 0$. Recalling that Sym$(\mathbf{Y} + \mathbf{B}) = \mathbf{K}\mathbf{\Sigma}\mathbf{K}^T$, we finally have

$$R_{\mathbf{Y}}(\mathbf{B}) = \mathbf{K}\mathbf{\Sigma}^*\mathbf{K}^T = \mathcal{W}_k(\mathbf{Y} + \mathbf{B}),$$

where the last equality is from (3.11). □

[1] Since $\mathbf{Z}^* \in \widetilde{\mathcal{Y}}$, $\mathbf{Z}^* \succeq 0$ and also $\mathbf{\Sigma}^* = \mathbf{K}^T\mathbf{Z}^*\mathbf{K} \succeq 0$ and rank$(\mathbf{\Sigma}^*)$ = rank$(\mathbf{K}^T\mathbf{Z}^*\mathbf{K})$ = rank$(\mathbf{Z}^*) = k$.

# Chapter D

# Proof of Theorem 9

**Proof:** In general, Euclidean gradient $\nabla_{\mathbf{Y}} f(\mathbf{Y})$ can be obtained by taking partial derivatives with respect to each coordinate of the Euclidean space. Since $\nabla_{\mathbf{Y}} f(\mathbf{Y})$ is interpreted as a matrix whose inner product with an arbitrary matrix $\mathbf{H}$ becomes the Frechet differential $\mathrm{D}f(\mathbf{Y})[\mathbf{H}]$ of $f$ at $\mathbf{Y}$ [104], that is,

$$\mathrm{D}f(\mathbf{Y})[\mathbf{H}] = \sum_{ij} h_{ij} \frac{\partial}{\partial y_{ij}} f(\mathbf{Y}),$$

it is convenient to compute $\nabla_{\mathbf{Y}} f(\mathbf{Y})$ as a unique element of $\mathbb{R}^{n \times n}$ that satisfies

$$< \nabla_{\mathbf{Y}} f(\mathbf{Y}), \mathbf{H} >= \mathrm{D}f(\mathbf{Y})[\mathbf{H}], \tag{D.1}$$

for all $\mathbf{H}$. We first compute $\mathrm{D}f(\mathbf{Y})[\mathbf{H}]$ and then use (D.1) to obtain the expression of $\nabla_{\mathbf{Y}} f(\mathbf{Y})$. Note that the cost function $f(\mathbf{Y}) = \frac{1}{2}\|\mathcal{P}_E(g(\mathbf{Y}) - \mathcal{P}_E(\mathbf{D}_{obs})\|_F^2$ can be expressed as $f(\mathbf{Y}) = h(k(\mathbf{Y})) = (h \circ k)(\mathbf{Y})$ where

$$\begin{aligned} h(\mathbf{R}) &= \frac{1}{2}\|\mathbf{R}\|_F^2, & \text{(D.2)} \\ k(\mathbf{Y}) &= \mathbf{W} \circ (\mathcal{P}_E(g(\mathbf{Y})) - \mathcal{P}_E(\mathbf{D}_{obs})) \\ &= \mathbf{W} \circ (\mathcal{P}_E \circ g)(\mathbf{Y}) - \mathbf{W} \circ \mathcal{P}_E(\mathbf{D}_{obs}). & \text{(D.3)} \end{aligned}$$

Thus,

$$\mathrm{D}f(\mathbf{Y})[\mathbf{H}] = \mathrm{D}(h \circ k)(\mathbf{Y})[\mathbf{H}] = \mathrm{D}h(k(\mathbf{Y}))[\mathrm{D}k(\mathbf{Y})[\mathbf{H}]]. \tag{D.4}$$

For any two square matrices $\mathbf{R}$ and $\mathbf{A}$, we have

$$\begin{aligned} \mathrm{D}h(\mathbf{R})[\mathbf{A}] &= \sum_{i,j} a_{ij} \frac{\partial}{\partial r_{ij}} h(\mathbf{R}) \\ &= \sum_{i,j} a_{ij} \frac{\partial}{\partial r_{ij}} (\frac{1}{2} \sum_{p,q} r_{pq}^2) \\ &= \sum_{i,j} a_{ij} r_{ij} \\ &= < \mathbf{R}, \mathbf{A} > . \end{aligned} \tag{D.5}$$

By choosing $\mathbf{R} = k(\mathbf{Y})$ and $\mathbf{A} = \mathrm{D}k(\mathbf{Y})[\mathbf{H}]$ in (D.5), we can rewrite (D.4) as

$$\begin{aligned} \mathrm{D}f(\mathbf{Y})[\mathbf{H}] &= < k(\mathbf{Y}), \mathrm{D}k(\mathbf{Y})[\mathbf{H}] > \\ &\overset{(a)}{=} < k(\mathbf{Y}), \mathrm{D}\left(\mathbf{W} \circ (\mathcal{P}_E(g(\mathbf{Y}) - \mathbf{D}_{obs})\right)[\mathbf{H}] > \\ &\overset{(b)}{=} < k(\mathbf{Y}), \mathrm{D}\left(\mathbf{W} \circ (\mathcal{P}_E \circ g)(\mathbf{Y})\right)[\mathbf{H}] > \\ &= < \mathbf{W} \circ k(\mathbf{Y}), \mathrm{D}(\mathcal{P}_E \circ g)(\mathbf{Y})[\mathbf{H}] > \\ &\overset{(c)}{=} < \mathbf{W} \circ k(\mathbf{Y}), \mathrm{D}\mathcal{P}_E(g(\mathbf{Y}))[\mathrm{D}g(\mathbf{Y})[\mathbf{H}]] >, \end{aligned}$$

where (a) follows (D.2), (b) is because $\mathcal{P}_E(\mathbf{D}_{obs})$ is not a function of $\mathbf{Y}$ and thus the Frechet differential of this is zero, and (c) is due to the chain rule.

Before we proceed, we remark that if $S$ is a linear operator (i.e., $S(\alpha_1 \mathbf{A}_1 + \alpha_2 \mathbf{A}_2) = \alpha_1 S(\mathbf{A}_1) + \alpha_2 S(\mathbf{A}_2)$), then

$$\mathrm{D}S(\mathbf{A})[\mathbf{B}] = S(\mathbf{B}) \tag{D.6}$$

for all matrices $\mathbf{A}$ and $\mathbf{B}$ (see Example 4.4.2 [105]).

Since $\mathcal{P}_E$ is a linear operator, $\mathrm{D}\mathcal{P}_E(g(\mathbf{Y}))[\mathrm{D}g(\mathbf{Y})[\mathbf{H}]] = \mathcal{P}_E([\mathrm{D}g(\mathbf{Y})[\mathbf{H}])$ and

hence

$$\begin{aligned}
\mathrm{D}f(\mathbf{Y})[\mathbf{H}] &= < \mathbf{W} \circ k(\mathbf{Y}), \mathcal{P}_E(\mathrm{D}g(\mathbf{Y})[\mathbf{H}]) > \\
&\overset{(a)}{=} < \mathcal{P}_E(\mathbf{W} \circ k(\mathbf{Y})), \mathrm{D}g(\mathbf{Y})[\mathbf{H}] > \\
&\overset{(b)}{=} < \mathbf{W} \circ k(\mathbf{Y}), \mathrm{D}g(\mathbf{Y})[\mathbf{H}] > \\
&\overset{(c)}{=} < \mathbf{W} \circ k(\mathbf{Y}), g(\mathbf{H}) > \\
&\overset{(d)}{=} 2 < \mathbf{W} \circ k(\mathbf{Y}), \mathrm{Sym}(\mathbf{1}\mathrm{diag}(\mathbf{H})^T) > \\
&\quad -2 < \mathbf{W} \circ k(\mathbf{Y}), \mathrm{Sym}(\mathbf{H}) >, \qquad \text{(D.7)}
\end{aligned}$$

where (a) is because $\mathcal{P}_E$ is a self-adjoint operator[1], (b) is because $\mathcal{P}_E(k(\mathbf{Y})) = \mathcal{P}_E(\mathbf{W} \circ (\mathcal{P}_E \circ g)(\mathbf{Y}) - \mathbf{W} \circ \mathcal{P}_E(\mathbf{D}_{obs})) = \mathbf{W} \circ (\mathcal{P}_E \circ g)(\mathbf{Y}) - \mathbf{W} \circ \mathcal{P}_E(\mathbf{D}_{obs}) = k(\mathbf{Y})$, (c) is because $g$ is also a linear function and thus $\mathrm{D}g(\mathbf{Y})[\mathbf{H}] = g(\mathbf{H})$, and (d) is due to (3.1).

Now, the first term in (D.7) is

$$\begin{aligned}
2 < \mathbf{W} \circ k(\mathbf{Y}), \mathrm{Sym}(\mathbf{1}\mathrm{diag}(\mathbf{H})^T) > &\overset{(a)}{=} 2 < \mathrm{Sym}(\mathbf{W} \circ k(\mathbf{Y})), \mathbf{1}\mathrm{diag}(\mathbf{H})^T > \\
&\overset{(b)}{=} 2 < \mathrm{Sym}(\mathbf{W} \circ k(\mathbf{Y})), \mathrm{diag}(\mathbf{H})\mathbf{1}^T > \\
&\overset{(c)}{=} 2 < \mathrm{Sym}(\mathbf{W} \circ k(\mathbf{Y}))\mathbf{1}, \mathrm{diag}(\mathbf{H}) > \\
&\overset{(d)}{=} 2 < \mathrm{eye}(\mathrm{Sym}(\mathbf{W} \circ k(\mathbf{Y}))\mathbf{1}), \mathbf{H} >, \qquad \text{(D.8)}
\end{aligned}$$

where (a) is because Sym() is a self-adjoint operator, (b) is because $< \mathbf{U}, \mathbf{V} >=< \mathbf{U}^T, \mathbf{V}^T >$, (c) is because $< \mathbf{A}, \mathbf{b}\mathbf{1}^T >= tr(\mathbf{A}^T\mathbf{b}\mathbf{1}^T) = tr((\mathbf{A}\mathbf{1})^T\mathbf{b}) =< \mathbf{A}\mathbf{1}, \mathbf{b} >$, and (d) is because eye() is the adjoint operator of diag(). Next, the second term in (D.7) is

$$-2 < \mathbf{W} \circ k(\mathbf{Y}), \mathrm{Sym}(\mathbf{H}) > \;=\; -2 < \mathrm{Sym}(\mathbf{W} \circ k(\mathbf{Y})), \mathbf{H} > . \qquad \text{(D.9)}$$

[1]Let $\mathcal{A}$ and $\mathcal{B}$ be two linear operators in $\mathbb{R}^{n \times n}$. If $< \mathcal{A}(\mathbf{A}), \mathbf{B} >=< \mathbf{A}, \mathcal{B}(\mathbf{B}) >$, we say $\mathcal{A}$ and $\mathcal{B}$ are adjoint to each other in $\mathbb{R}^{n \times n}$. Further, if $\mathcal{A} \equiv \mathcal{B}$, then we say it is a self-adjoint operator.

From (D.7), (D.8), and (D.9), we have

$$\begin{aligned} \mathrm{D}f(\mathbf{Y})[\mathbf{H}] &= 2 < \text{eye}(\text{Sym}(\mathbf{W} \circ k(\mathbf{Y}))\mathbf{1}), \mathbf{H} > -2 < \text{Sym}(\mathbf{W} \circ k(\mathbf{Y})), \mathbf{H} > \\ &= < 2\text{eye}(\text{Sym}(\mathbf{W} \circ k(\mathbf{Y}))\mathbf{1}) - 2\text{Sym}(\mathbf{W} \circ k(\mathbf{Y})), \mathbf{H} > \end{aligned} \tag{D.10}$$

From (D.1) and (D.10), we have

$$\nabla_{\mathbf{Y}} f(\mathbf{Y}) = 2\text{eye}(\text{Sym}(\mathbf{W} \circ k(\mathbf{Y}))\mathbf{1}) - 2\text{Sym}(\mathbf{W} \circ k(\mathbf{Y})),$$

which is the desired result.

□

# Chapter E

# Proof of Lemma 10

**Proof:** If $\mathbf{Y}_i$ is the optimal point (i.e, $\mathbf{Y}_i = \arg\min_{\mathbf{Y}} f(\mathbf{Y})$), then grad$f(\mathbf{Y}_i) = \mathbf{0}$ and $\mathbf{Y}_{i+1} = \mathbf{Y}_i$. For all $\alpha_i \geq 0$, we have

$$f(R_{\mathbf{Y}_i}(\alpha_i \mathbf{P}_i)) = f(\mathbf{Y}_{i+1}) = f(\mathbf{Y}_i) + \tau\alpha_i < \text{grad} f(\mathbf{Y}_i), \mathbf{P}_i >,$$

satisfying **A1**. Next, we consider the case where $\mathbf{Y}_i \neq \arg\min_{\mathbf{Y}} f(\mathbf{Y})$. First, we let

$$\begin{aligned} g(\alpha) &= f(R_{\mathbf{Y}_i}(\alpha \mathbf{P}_i)), \\ h(\alpha) &= f(\mathbf{Y}_i) + \tau\alpha < \text{grad} f(\mathbf{Y}_i), \mathbf{P}_i > . \end{aligned} \tag{E.1}$$

Note that $< \text{grad} f(\mathbf{Y}_i), \mathbf{P}_i > \leq 0$ (see Lemma 13) and $g(\alpha) \geq 0$. Since $g(0) = f(R_{\mathbf{Y}_i}(\mathbf{0})) = f(\mathbf{Y}_i) = h(0)$, $g(\alpha)$ and $h(\alpha)$ intersect at $\alpha = 0$. Also, when $\tau$ varies from 0 to 1, the slope of $h(\alpha)$ varies from 0 to $| < \text{grad} f(\mathbf{Y}_i), \mathbf{P}_i > |$. Since $\left.\frac{dg(\alpha)}{d\alpha}\right|_{\alpha=0} = < \text{grad} f(\mathbf{Y}_i), \mathbf{P}_i >$, $h(\alpha)$ is the tangential curve of $g(\alpha)$ at $\alpha = 0$ when $\tau = 1$. Thus, there exits $0<\tau<1/2$ such that $h(\alpha)$ intersects $g(\alpha)$ at some point $\alpha>0$, which means that there exist $\alpha_i>0$ satisfying

$$f(R_{\mathbf{Y}_i}(\alpha_i \mathbf{P}_i)) = g(\alpha_i) \leq h(\alpha_i) = f(\mathbf{Y}_i) + \tau\alpha_i < \text{grad} f(\mathbf{Y}_i), \mathbf{P}_i >,$$

which completes the proof.

□

# Chapter F

# Proof of Lemma 12

**Proof:** First, a lower bound of $\|\nabla_{\mathbf{Y}} f(\mathbf{Y}_i)\|_F$ is given by

$$\begin{aligned}
\|\nabla_{\mathbf{Y}} f(\mathbf{Y}_i)\|_F^2 &\overset{(a)}{=} \|2\text{eye}(\mathbf{R}_i\mathbf{1}) - 2\mathbf{R}_i\|_F^2 \\
&\overset{(b)}{=} \|2\text{eye}(\mathbf{R}_i\mathbf{1})\|_F^2 + \|2\mathbf{R}_i\|_F^2 \\
&\geq \|2\mathbf{R}_i\|_F^2 \\
&= 4\|\mathbf{W}\circ\mathbf{W}\circ(\mathcal{P}_E(\mathbf{D}_i) - \mathcal{P}_E(\mathbf{D}))\|_F^2, \qquad \text{(F.1)}
\end{aligned}$$

where (a) is from (3.15) and (b) is from the fact that diagonal entries of $\mathbf{R}_j$ are all zeros and eye$(\mathbf{R}_j\mathbf{1})$ is a diagonal matrix. That is, positions of nonzero elements in eye$(\mathbf{R}_i\mathbf{1})$ and $\mathbf{R}_i$ are disjoint. An upper bound is obtained as follows.

$$\begin{aligned}
\|\nabla_{\mathbf{Y}} f(\mathbf{Y}_i)\|_F &\leq \|2\text{eye}(\mathbf{R}_i\mathbf{1})\|_F + \|2\mathbf{R}_i\|_F \\
&\overset{(a)}{\leq} \|2\mathbf{R}_i\mathbf{1}\|_2 + \|2\mathbf{R}_i\|_F \\
&\overset{(b)}{\leq} 2\|\mathbf{R}_i\|_F\|\mathbf{1}\|_2 + 2\|\mathbf{R}_i\|_F \\
&\leq (2\sqrt{n}+2)\|\mathbf{R}_i\|_F \\
&\leq (2\sqrt{n}+2)\|\mathcal{P}_E(\mathbf{D}_i) - \mathcal{P}_E(\mathbf{D})\|_F, \qquad \text{(F.2)}
\end{aligned}$$

where (a) is because $\|\text{eye}(\mathbf{b})\|_F = \|\mathbf{b}\|_2$ for any vector $\mathbf{b}$, and (b) is because $\|\mathbf{A}\mathbf{b}\|_2 \leq \|\mathbf{A}\|_F\|\mathbf{b}\|_2$ for any matrix $\mathbf{A}$ and any vector $\mathbf{b}$. By combining (F.1) and (F.2), we obtain the desired result. □

# Chapter G

# Proof of Lemma 13

**Proof:** Recall from (3.23) that we have

$$\mathbf{P}_{i+1} = -\text{grad}f(\mathbf{Y}_{i+1}) + \beta_{i+1} P_{T_{\mathbf{Y}_{i+1}}\widetilde{\mathcal{Y}}}(\mathbf{P}_i).$$

Thus,

$$\begin{aligned}
& < -\text{grad}f(\mathbf{Y}_{i+1}), \mathbf{P}_{i+1} > \\
= \;& \| - \text{grad}f(\mathbf{Y}_{i+1})\|_F^2 + \beta_{i+1} < -\text{grad}f(\mathbf{Y}_{i+1}), P_{T_{\mathbf{Y}_{i+1}}\widetilde{\mathcal{Y}}}(\mathbf{P}_i) > \\
= \;& \| - \text{grad}f(\mathbf{Y}_{i+1})\|_F^2 + \beta_{i+1} < -P_{T_{\mathbf{Y}_{i+1}}\widetilde{\mathcal{Y}}}(\text{grad}f(\mathbf{Y}_{i+1})), \mathbf{P}_i > \\
\overset{(a)}{=} \;& \| - \text{grad}f(\mathbf{Y}_{i+1})\|_F^2 + \beta_{i+1} < -\text{grad}f(\mathbf{Y}_{i+1}), \mathbf{P}_i >,
\end{aligned}$$

where (a) is because $\text{grad}f(\mathbf{Y}_{i+1}) \in T_{\mathbf{Y}_{i+1}}\widetilde{\mathcal{Y}}$. Then we have

$$\begin{aligned}
\left| < \text{grad}f(\mathbf{Y}_{i+1}), \mathbf{P}_{i+1} > + \|\text{grad}f(\mathbf{Y}_{i+1})\|_F^2 \right| &= \beta_{i+1} \left| < \text{grad}f(\mathbf{Y}_{i+1}), \mathbf{P}_i > \right| \\
&\overset{(a)}{\leq} \beta_{i+1}\mu < -\text{grad}f(\mathbf{Y}_i), \mathbf{P}_i >,
\end{aligned}$$

where (a) is from the assumption **A2**.

If we denote $\zeta_i = -\frac{<\text{grad}f(\mathbf{Y}_i), \mathbf{P}_i>}{\|\text{grad}f(\mathbf{Y}_i)\|_F^2}$, then

$$\left| -\zeta_{i+1}\|\text{grad}f(\mathbf{Y}_{i+1})\|_F^2 + \|\text{grad}f(\mathbf{Y}_{i+1})\|_F^2 \right| \leq \beta_{i+1}\mu\zeta_i \|\text{grad}f(\mathbf{Y}_i)\|_F^2,$$

and also

$$| - \zeta_{i+1} + 1| \leq \mu\beta_{i+1} \frac{\|\text{grad} f(\mathbf{Y}_i)\|_F^2}{\|\text{grad} f(\mathbf{Y}_{i+1})\|_F^2} \zeta_i. \tag{G.1}$$

From Fletcher-Reeves rule in (3.31), we have $\beta_{i+1} \frac{\|\text{grad} f(\mathbf{Y}_i)\|_F^2}{\|\text{grad} f(\mathbf{Y}_{i+1})\|_F^2} = 1$ and thus

$$| - \zeta_{i+1} + 1| \leq \mu\zeta_i.$$

In other words,

$$\begin{aligned} \zeta_{i+1} &\geq 1 - \mu\zeta_i, \qquad (G.2) \\ &\text{and} \\ \zeta_{i+1} &\leq 1 + \mu\zeta_i \\ \zeta_i &\leq 1 + \mu\zeta_{i-1} \\ &\vdots \\ \zeta_2 &\leq 1 + \mu\zeta_1, \end{aligned}$$

where we set $\zeta_1 = 1$. Thus,

$$\begin{aligned} \zeta_i &\leq \sum_{j=0}^{i-1} \mu^j \\ &= \frac{1 - \mu^i}{1 - \mu}. \qquad (G.3) \end{aligned}$$

From (G.2) and (G.3), we finally have

$$\begin{aligned} \zeta_{i+1} &\geq 1 - \mu \frac{1 - \mu^i}{1 - \mu} \\ &= \frac{1 - 2\mu + \mu^{i+1}}{1 - \mu}, \end{aligned}$$

which is the desired result. □

# Chapter H

# Proof of Lemma 14

**Proof:** From (3.22), we have

$$\text{grad} f(\mathbf{Y}_i) = P_{T_{\mathbf{Y}_i}\widetilde{\mathcal{Y}}}(\nabla_{\mathbf{Y}} f(\mathbf{Y}_i)),$$

where $\nabla_{\mathbf{Y}} f(\mathbf{Y}_i)$ is the Euclidean gradient. Let $P^{\perp}_{T_{\mathbf{Y}_i}\widetilde{\mathcal{Y}}}$ be the orthogonal operator on the complement space of $T_{\mathbf{Y}_i}\widetilde{\mathcal{Y}}$, then we obtain

$$\begin{aligned} \|\nabla_{\mathbf{Y}} f(\mathbf{Y}_i)\|_F^2 &= \|P_{T_{\mathbf{Y}_i}\widetilde{\mathcal{Y}}}(\nabla_{\mathbf{Y}} f(\mathbf{Y}_i)) + P^{\perp}_{T_{\mathbf{Y}_i}\widetilde{\mathcal{Y}}}(\nabla_{\mathbf{Y}} f(\mathbf{Y}_i))\|_F^2 \\ &= \|P_{T_{\mathbf{Y}_i}\widetilde{\mathcal{Y}}}(\nabla_{\mathbf{Y}} f(\mathbf{Y}_i))\|_F^2 + \|P^{\perp}_{T_{\mathbf{Y}_i}\widetilde{\mathcal{Y}}}(\nabla_{\mathbf{Y}} f(\mathbf{Y}_i))\|_F^2, \quad \text{(H.1)} \end{aligned}$$

and hence

$$\begin{aligned} \|\text{grad} f(\mathbf{Y}_i)\|_F^2 &= \|P_{T_{\mathbf{Y}_i}\widetilde{\mathcal{Y}}}(\nabla_{\mathbf{Y}} f(\mathbf{Y}_i))\|_F^2 \\ &= \|\nabla_{\mathbf{Y}} f(\mathbf{Y}_i)\|_F^2 - \|P^{\perp}_{T_{\mathbf{Y}_i}\widetilde{\mathcal{Y}}}(\nabla_{\mathbf{Y}} f(\mathbf{Y}_i))\|_F^2. \quad \text{(H.2)} \end{aligned}$$

Now, we define

$$\chi = \begin{cases} \sup\limits_{\mathbf{Y}\in\{\mathbf{Y}_i\}_{i=1}^{\infty}} \frac{\|P^{\perp}_{T_{\mathbf{Y}}\widetilde{\mathcal{Y}}}(\nabla_{\mathbf{Y}} f(\mathbf{Y}))\|_F}{\|\nabla_{\mathbf{Y}} f(\mathbf{Y})\|_F} & \text{if } \|\nabla_{\mathbf{Y}} f(\mathbf{Y})\|_F \neq 0 \\ 1 & \text{otherwise} \end{cases}. \quad \text{(H.3)}$$

Note that $1 \geq \chi \geq 0$ because $\|\nabla_{\mathbf{Y}} f(\mathbf{Y})\|_F \geq \|P^{\perp}_{T_{\mathbf{Y}}\widetilde{\mathcal{Y}}}(\nabla_{\mathbf{Y}} f(\mathbf{Y}))\|_F$ (see (H.1)). From (H.2) and (H.3), we have

$$\begin{aligned} \|\text{grad} f(\mathbf{Y}_i)\|_F^2 &= \left(1 - \frac{\|P^{\perp}_{T_{\mathbf{Y}_i}\widetilde{\mathcal{Y}}}(\nabla_{\mathbf{Y}} f(\mathbf{Y}_i))\|_F^2}{\|\nabla_{\mathbf{Y}} f(\mathbf{Y}_i)\|_F^2}\right) \|\nabla_{\mathbf{Y}} f(\mathbf{Y}_i)\|_F^2 \qquad \text{(H.4)} \\ &\geq (1-\chi^2)\|\nabla_{\mathbf{Y}} f(\mathbf{Y}_i)\|_F^2. \qquad \text{(H.5)} \end{aligned}$$

Now, what remains is to show that $\|\nabla_{\mathbf{Y}} f(\mathbf{Y}_i)\|_F^2 \geq 8 f(\mathbf{Y}_i)$. Indeed, from Lemma 9, we have

$$\|\nabla_{\mathbf{Y}} f(\mathbf{Y}_i)\|_F^2 = \|\text{eye}((\mathbf{R} + \mathbf{R}^T)\mathbf{1}) - 2\mathbf{R}\|_F^2,$$

where $\mathbf{R} = \mathcal{P}_E(g(\mathbf{Y}_i)) - \mathcal{P}_E(\mathbf{D})$. Noting that $\mathbf{R}$ is symmetric with zero diagonal entries $r_{ii} = 0$, we have

$$\begin{aligned} \frac{1}{4}\|\nabla_{\mathbf{Y}} f(\mathbf{Y}_i)\|_F^2 &= \|\text{eye}(\mathbf{R}\mathbf{1})\|_F^2 + \|\mathbf{R}\|_F^2 - 2 < \text{eye}(\mathbf{R}\mathbf{1}), \mathbf{R} > \\ &= \|\mathbf{R}\mathbf{1}\|_2^2 + \|\mathbf{R}\|_F^2 - 2\sum_i \left(\sum_j r_{ij}\right) r_{ii} \\ &= \|\mathbf{R}\mathbf{1}\|_2^2 + \|\mathbf{R}\|_F^2 \\ &\geq \|\mathbf{R}\|_F^2 \\ &= 2 f(\mathbf{Y}_i). \qquad \text{(H.6)} \end{aligned}$$

By substituting (H.6) into (H.5), we obtain the desired result. □

# Chapter I

# Proof of Lemma 15

**Proof:** By denoting $\mathbf{x}_i = \begin{bmatrix} x_{i1} & x_{i2} & \cdots & x_{ik} \end{bmatrix}^T$, we have

$$p = P(d_{ij} \leq r) = P(\|\mathbf{x}_i - \mathbf{x}_j\|_2^2 \leq r^2) = P(\sum_{t=1}^{k}(x_{it} - x_{jt})^2 \leq r^2).$$

Before finding the general form of $p$, we compute the distribution of $Y = (X_1 - X_2)^2$ where $X_1$ and $X_2$ are i.i.d. uniformly distributed random variables at unit interval. Let $Z = X_1 - X_2$, then the cdf of $Z$ is given by

$$\begin{aligned} F_Z(z) &= P(Z \leq z) \\ &= P(X_1 - X_2 \leq z) \\ &= \int_0^1 P(X_1 \leq z + x_2 \big| X_2 = x_2) f_{X_2}(x_2) dx_2 \\ &= \int_0^1 P(X_1 \leq z + x_2) f_{X_2}(x_2) dx_2 \\ &= \int_0^1 F_{X_1}(z + x_2) f_{X_2}(x_2) dx_2. \end{aligned}$$

Thus,

$$\begin{aligned} f_Z(z) &= \frac{d}{dz} F_Z(z) \\ &= \begin{cases} 1 - |z| & \text{if } |z| \leq 1 \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

Now, we can easily obtain the cdf of $Y = Z^2$

$$F_Y(y) = \begin{cases} 1 & \text{if } 1 \le y \\ 2\sqrt{y} - y & \text{if } 0 \le y \le 1 \\ 0 & \text{if } y \le 0 \end{cases}.$$

and also the pdf of $Y$

$$f_Y(y) = \begin{cases} \frac{1}{\sqrt{y}} - 1 & \text{if } 0 \le y \le 1 \\ 0 & \text{otherwise} \end{cases}. \tag{I.1}$$

Using this, we can compute $p$ as

$$\begin{aligned} p &= Pr\left(\sum_{t=1}^{k} y_t \le r^2\right) \\ &= \int\cdots\int_{\alpha_1+\alpha_2+\ldots+\alpha_k \le r^2} f_{Y_1,\ldots,Y_k}(\alpha_1,\ldots,\alpha_k)d\alpha_1\ldots d\alpha_k \\ &= \int\cdots\int_{\alpha_1+\alpha_2+\ldots+\alpha_k \le r^2} f_{Y_1}(\alpha_1)\ldots f_{Y_k}(\alpha_k)d\alpha_1\ldots d\alpha_k, \end{aligned} \tag{I.2}$$

where $y_t = (x_{it} - x_{jt})^2$. When the sensor nodes are located in two dimensional Euclidean space ($k = 2$), we have

$$p = \iint_{\alpha_1+\alpha_2 \le r^2} f_{Y_1}(\alpha_1) f_{Y_2}(\alpha_2) d\alpha_1 d\alpha_2.$$

Let $t = \alpha_1 + \alpha_2$, then we have

$$\begin{aligned} p &= \iint_{\alpha_1+\alpha_2 \le r^2} f_{Y_1}(\alpha_1) f_{Y_2}(\alpha_2) d\alpha_1 d\alpha_2 \\ &= \int_0^{r^2} \left[\int_0^1 f_{Y_1}(\alpha_1) f_{Y_2}(t-\alpha_1) d\alpha_1\right] dt \\ &= \int_0^{r^2} f_{Y_1}(t) * f_{Y_2}(t) dt. \end{aligned} \tag{I.3}$$

After some manipulations, we have

$$f_{Y_1}(t) * f_{Y_2}(t) = \begin{cases} \pi - 4\sqrt{t} + t & \text{if } 0 \le t \le 1 \\ 2\sin^{-1}(\frac{2}{t} - 1) + 4\sqrt{t-1} - t - 2 & \text{if } 1 \le t \le 2 \\ 0 & \text{otherwise} \end{cases}. \tag{I.4}$$

Let $h_1(u) = \int_0^u (\pi - 4\sqrt{t} + t)dt$ $(0 \le u \le 1)$ and $h_2(u) = \int_1^u (2\sin^{-1}(\frac{2}{t} - 1) + 4\sqrt{t-1} - t - 2)dt$ $(1 \le u \le 2)$, then we have

$$p = \begin{cases} h_1(r^2) & \text{if } 0 \le r \le 1 \\ h_1(1) + h_2(r^2) & \text{if } 1 \le r \le \sqrt{2} \\ 1 & \text{otherwise} \end{cases},$$

where $h_1(u)$ and $h_2(u)$ are given by

$$\begin{aligned} h_1(u) &= \pi t - \frac{8}{3}t^{3/2} + \frac{1}{2}t^2\Big|_0^u = \pi u - \frac{8}{3}u^{3/2} + \frac{1}{2}u^2, \\ h_2(u) &= \frac{8}{3}(t-1)^{3/2} - \frac{1}{2}t^2 - 2t\Big|_1^u + 2\int_1^u \sin^{-1}\left(\frac{2}{t} - 1\right)dt \\ &= \frac{8}{3}(u-1)^{3/2} - \frac{1}{2}(u-1)(u+5) + 2\int_1^u \sin^{-1}\left(\frac{2}{t} - 1\right)dt, \\ &= \frac{8}{3}(u-1)^{3/2} - \frac{1}{2}(u-1)(u+5) + 2u\sin^{-1}\left(\frac{2}{u} - 1\right) \\ &\quad + \frac{2}{1 + \tan\left(\frac{\sin^{-1}\left(\frac{2}{u}-1\right)}{2}\right)} + 3\sqrt{u-1} - \pi - 1. \end{aligned}$$

Denoting $p_1(r) = h_1(1) + h_2(r^2)$, we get the desired result for $k = 2$.

Similarly, when the sensor nodes are located in three dimensional Euclidean space $(k = 3)$, we have

$$\begin{aligned} p &= \iiint_{\alpha_1+\alpha_2+\alpha_3 \le r^2} f_{Y_1}(\alpha_1) f_{Y_2}(\alpha_2) f_{Y_3}(\alpha_3) d\alpha_1 d\alpha_2 d\alpha_3 \\ &= \int_0^{r^2} \left[ \int_0^1 \int_0^1 f_{Y_1}(\alpha_1) f_{Y_2}(\alpha_2) f_{Y_3}(t - \alpha_1 - \alpha_2) d\alpha_1 d\alpha_2 \right] dt \\ &= \int_0^{r^2} \left[ \int_0^1 \left[ \int_0^1 f_{Y_1}(\alpha_1) f_{Y_2}(u - \alpha_1) d\alpha_1 \right] f_{Y_3}(t-u) du \right] dt \\ &= \int_0^{r^2} [f_{Y_1}(t) * f_{Y_2}(t)] * f_{Y_3}(t) dt. \end{aligned}$$

After some manipulations, we have

$$[f_{Y_1}(t) * f_{Y_2}(t)] * f_{Y_3}(t) = \begin{cases} \widetilde{h}_3(t) & \text{if } 0 \le t \le 1 \\ \widetilde{h}_4(t) & \text{if } 1 \le t \le 2 \\ \widetilde{h}_5(t) & \text{if } 2 \le t \le 3 \\ 0 & \text{otherwise} \end{cases}, \tag{I.5}$$

where

$$\begin{aligned}
\widetilde{h}_3(t) &= 4t\sqrt{t} + 2\pi\sqrt{t} - 3\pi t - \frac{1}{2}t^2, \\
\widetilde{h}_4(t) &= 3\pi - 4\pi\sqrt{t} + (3\pi+4)t + \frac{1}{2}t^2 - 8t\sqrt{t-1} - 3\sqrt{t-1} + \frac{1}{2}(t-1)^2 \\
&\quad -4t\sin^{-1}\sqrt{\frac{1}{t}} + 4t\sin^{-1}\sqrt{1-\frac{1}{t}} - 2t\sin^{-1}\left(\frac{2}{t}-1\right) \\
&\quad -\frac{2}{1+\tan\left(\frac{\sin^{-1}\left(\frac{2}{t}-1\right)}{2}\right)}, \\
\widetilde{h}_5(t) &= 3\sqrt{t-2} + 4t\sqrt{t-2} - 4t - \frac{1}{2}(t-1)^2 - 3 + 8\sqrt{t}\tan^{-1}\sqrt{\frac{t-2}{t}} \\
&\quad -8\tan^{-1}\sqrt{t-2} - 4(t-1)\sin^{-1}\sqrt{\frac{t-2}{t-1}} + 2(t+1)\sin^{-1}\left(\frac{3-t}{t-1}\right) \\
&\quad -8\sqrt{t}\tan^{-1}\sqrt{\frac{1}{t(t-2)}} + 8\tan^{-1}\sqrt{\frac{1}{t-2}} + 4(t-1)\sin^{-1}\sqrt{\frac{1}{t-1}} \\
&\quad +\frac{2}{1+\tan\left(\frac{\sin^{-1}\left(\frac{3-t}{t-1}\right)}{2}\right)}.
\end{aligned}$$

Now, we let $h_3(u) = \int_0^u \widetilde{h}_3(t)dt$ $(0 \le u \le 1)$, $h_4(u) = \int_1^u \widetilde{h}_4(t)dt$ $(1 \le u \le 2)$, and $h_5(u) = \int_2^u \widetilde{h}_5(t)dt$ $(2 \le u \le 3)$, then we have

$$p = \begin{cases} h_3(r^2) & \text{if } 0 \le r \le 1 \\ h_3(1) + h_4(r^2) & \text{if } 1 \le r \le \sqrt{2} \\ h_3(1) + h_4(2) + h_5(r^2) & \text{if } \sqrt{2} \le r \le \sqrt{3} \\ 1 & \text{otherwise} \end{cases}, \tag{I.6}$$

where $h_3(u)$, $h_4(u)$, and $h_5(u)$ are given by

$$
\begin{aligned}
h_3(u) &= \frac{4\pi}{3}u\sqrt{u}+\frac{8}{5}u^2\sqrt{u}-\frac{3\pi}{2}u^2-\frac{1}{6}u^3,\\
h_4(u) &= 3\pi u-\frac{8\pi}{3}u\sqrt{u}+\frac{3\pi+4}{2}u^2+\frac{1}{6}u^3+\frac{1}{6}(u-1)^3-\frac{\pi}{3}-\frac{5}{2}-6\sqrt{u-1}\\
&\quad -\frac{28}{3}(u-1)\sqrt{u-1}-\frac{16}{5}(u-1)^2\sqrt{u-1}-2u^2\sin^{-1}\sqrt{\frac{1}{u}}\\
&\quad 2u^2\sin^{-1}\sqrt{1-\frac{1}{u}}-u^2\sin^{-1}\left(\frac{2}{u}-1\right)-\frac{16}{3\left(1+\tan\left(\frac{1}{2}\sin^{-1}\left(\frac{2}{u}-1\right)\right)\right)^3}\\
&\quad +\frac{4}{\left(1+\tan\left(\frac{1}{2}\sin^{-1}\left(\frac{2}{u}-1\right)\right)\right)^2},\\
h_5(u) &= 2(u-2)\sqrt{u-2}-2u^2-\frac{1}{6}(u-1)^3-3u+\frac{29}{2}+\frac{8}{5}(u-2)^2\sqrt{u-2}\\
&\quad +\frac{22}{3}(u-2)\sqrt{u-2}+14\sqrt{u-2}+\left(\frac{16\sqrt{2}}{3}-12\right)\pi+2\tan^{-1}\sqrt{u-2}\\
&\quad -8u\tan^{-1}\sqrt{u-2}+\frac{16}{3}u\sqrt{u}\tan^{-1}\sqrt{\frac{u-2}{u}}-2u(u-2)\sin^{-1}\sqrt{\frac{u-2}{u-1}}\\
&\quad +u(u+2)\sin^{-1}\left(\frac{3-u}{u-1}\right)-\frac{16}{3}u\sqrt{u}\tan^{-1}\sqrt{\frac{1}{u(u-2)}}+8u\tan^{-1}\sqrt{\frac{1}{u-2}}\\
&\quad +2u(u-2)\sin^{-1}\sqrt{\frac{1}{u-1}}+\frac{16}{3}\frac{1}{\left(1+\tan\left(\frac{1}{2}\sin^{-1}\left(\frac{3-u}{u-1}\right)\right)\right)^3}\\
&\quad -4\frac{1}{\left(1+\tan\left(\frac{1}{2}\sin^{-1}\left(\frac{3-u}{u-1}\right)\right)\right)^2}.
\end{aligned}
$$

By denoting $p_2(r)=h_3(1)+h_4(r^2)$ and $p_3(r)=h_3(1)+h_4(2)+h_5(r^2)$, we get the desired result for $k=3$.

□

# Chapter J

# Proof of Lemma 17

**Proof:** Since $\|\mathbf{A}\|_F^2 = \|\mathcal{P}_E(\mathbf{A})\|_F^2 + \|\mathcal{P}_E^{\perp}(\mathbf{A})\|_F^2$, we can rewrite $t\|\mathbf{A}\|_F^2 \leq \|\mathcal{P}_E(\mathbf{A})\|_F^2$ as

$$t\|\mathbf{A}\|_F^2 \leq \|\mathbf{A}\|_F^2 - \|\mathcal{P}_E^{\perp}(\mathbf{A})\|_F^2,$$

and also

$$\|\mathcal{P}_E^{\perp}(\mathbf{A})\|_F^2 \leq (1-t)\|\mathbf{A}\|_F^2. \tag{J.1}$$

To show that (J.1) holds true with overwhelming probability, we first have

$$\begin{aligned}
& P(\|\mathcal{P}_E^{\perp}(\mathbf{A})\|_F^2 \geq (1-t)\|\mathbf{A}\|_F^2) \\
= \;& P(\exp(\epsilon\|\mathcal{P}_E^{\perp}(\mathbf{A})\|_F^2) \geq \exp(\epsilon(1-t)\|\mathbf{A}\|_F^2)) \\
\overset{(a)}{\leq} \;& \exp(-\epsilon(1-t)\|\mathbf{A}\|_F^2)\mathrm{E}\left[\exp(\epsilon\|\mathcal{P}_E^{\perp}(\mathbf{A})\|_F^2)\right] \\
\overset{(b)}{=} \;& \exp(-\epsilon(1-t)\|\mathbf{A}\|_F^2)\mathrm{E}\left[\prod_{i\neq j}\exp(\epsilon(1-\delta_{ij})a_{ij}^2)\right],
\end{aligned} \tag{J.2}$$

for any $\epsilon>0$, where (a) follows from the Markov inequality and (b) is from $\|\mathcal{P}_E^{\perp}(\mathbf{A})\|_F^2 = \sum_{i\neq j}(1-\delta_{ij})a_{ij}^2$ (see (3.32)). Let $\Omega = \{(i,j) : a_{ij} \neq 0\}$ (i.e., $\Omega$ is the index set of nonzero entries of $\mathbf{A}$), and $N = 2^{\lfloor \log_2 |\Omega| \rfloor}$ ($|\Omega|$ is the cardinality of $\Omega$). Also, let $\widetilde{\Omega}$ be

a subset of $\Omega$ such that $|\widetilde{\Omega}| = N$, then

$$
\begin{aligned}
& \mathrm{E}\left[\prod_{i\neq j}\exp(\epsilon(1-\delta_{ij})a_{ij}^2)\right] \\
\leq\ & \prod_{(i,j)\in(\Omega\setminus\widetilde{\Omega})}\exp(\epsilon a_{ij}^2)\mathrm{E}\left[\prod_{(i,j)\in\widetilde{\Omega}}\exp(\epsilon(1-\delta_{ij})a_{ij}^2)\right] \\
\overset{(a)}{=}\ & \prod_{(i,j)\in(\Omega\setminus\widetilde{\Omega})}\exp(\epsilon a_{ij}^2)\left(\prod_{(i,j)\in\widetilde{\Omega}}\mathrm{E}\left[\exp(N\epsilon(1-\delta_{ij})a_{ij}^2)\right]\right)^{1/N} \\
=\ & \prod_{(i,j)\in(\Omega\setminus\widetilde{\Omega})}\exp(\epsilon a_{ij}^2)\prod_{(i,j)\in\widetilde{\Omega}}\left((1-p)\exp(N\epsilon a_{ij}^2)+p\right)^{1/N} \\
=\ & \prod_{(i,j)\in(\Omega\setminus\widetilde{\Omega})}\exp(\epsilon a_{ij}^2)\prod_{(i,j)\in\widetilde{\Omega}}\exp(\epsilon a_{ij}^2)\left(1-p+p\exp(-N\epsilon a_{ij}^2)\right)^{1/N} \\
=\ & \prod_{(i,j)\in\Omega}\exp(\epsilon a_{ij}^2)\prod_{(i,j)\in\widetilde{\Omega}}\left(1-p+p\exp(-N\epsilon a_{ij}^2)\right)^{1/N} \\
=\ & \exp(\epsilon\|\mathbf{A}\|_F^2)\prod_{(i,j)\in\widetilde{\Omega}}\left(1-p+p\exp(-N\epsilon a_{ij}^2)\right)^{1/N} \\
\overset{(b)}{\leq}\ & \exp(\epsilon\|\mathbf{A}\|_F^2)\prod_{(i,j)\in\widetilde{\Omega}}\left(1-p+p\exp(-N\epsilon a_{\min}^2)\right)^{1/N} \\
\overset{(c)}{=}\ & \exp(\epsilon\|\mathbf{A}\|_F^2)\left(1-p+p\exp(-N\epsilon a_{\min}^2)\right), \qquad \text{(J.3)}
\end{aligned}
$$

where (a) is because $\mathrm{E}\left[\prod_{i=1}^{M}A_i\right] \leq \left(\prod_{i=1}^{M}\mathrm{E}\left[A_i^M\right]\right)^{1/M}$ for positive random variable $A_i$ and $M = 2^q$ ($q \geq 1$), (b) is because $a_{\min} = \min_{(i,j)\in\Omega}|a_{ij}|$, and (c) is because $|\widetilde{\Omega}| = N$.

In summary, we have

$$P(\|\mathcal{P}_E^{\perp}(\mathbf{A})\|_F^2 \geq (1-t)\|\mathbf{A}\|_F^2) \quad \leq \quad g(\epsilon),$$

where $g(\epsilon) = \exp(mtN\epsilon a_{\min}^2)\left(1-p+p\exp(-N\epsilon a_{\min}^2)\right)$ ($m = \|\mathbf{A}\|_F^2/(a_{\min}^2 N)$). If $0<mt<1$, we obtain the minimum value of $g(\epsilon)$ at $\epsilon^* = 1/(Na_{\min}^2)\log((1-$

$mt)p/((1-p)mt))$. Thus,

$$\begin{aligned} & P(\|\mathcal{P}_E^{\perp}(\mathbf{A})\|_F^2 \geq (1-t)\|\mathbf{A}\|_F^2) \\ \leq \; & g(\epsilon^*) \\ = \; & \left(\frac{1-p}{1-mt}\right)^{1-mt}\left(\frac{p}{mt}\right)^{mt} \\ = \; & \exp\left(-\left((1-mt)\log\left(\frac{1-mt}{1-p}\right)+mt\log\left(\frac{mt}{p}\right)\right)\right), \end{aligned}$$

which is the desired result. □

# Chapter K

# Proof of Lemma 19

**Proof:** We first denote $\mathbf{B} = g(\mathbf{Y}) - \mathbf{D}$, and then express $\mathbf{B} = \sum_{ij} < \mathbf{B}, \mathbf{e}_i\mathbf{e}_j^T > \mathbf{e}_i\mathbf{e}_j^T$, which gives

$$\mathcal{P}_E(\mathbf{B}) = \sum_{i \neq j} \delta_{ij} < \mathbf{B}, \mathbf{e}_i\mathbf{e}_j^T > \mathbf{e}_i\mathbf{e}_j^T.$$

Recalling that $l(\mathbf{A}) = 2\text{eye}(\text{Sym}(\mathbf{A})\mathbf{1}) - 2\text{Sym}(\mathbf{A})$, we have

$$\begin{aligned}
\nabla_{\mathbf{Y}} f(\mathbf{Y}) &= l(\mathcal{P}_E(\mathbf{B})) \\
&= l(\sum_{i \neq j} \delta_{ij} < \mathbf{B}, \mathbf{e}_i\mathbf{e}_j^T > \mathbf{e}_i\mathbf{e}_j^T) \\
&\overset{(a)}{=} \sum_{i \neq j} \delta_{ij} < \mathbf{B}, \mathbf{e}_i\mathbf{e}_j^T > l(\mathbf{e}_i\mathbf{e}_j^T) \\
&= \sum_{i \neq j} \delta_{ij} s_{ij} l(\mathbf{e}_i\mathbf{e}_j^T),
\end{aligned}$$

where $s_{ij} =< \mathbf{B}, \mathbf{e}_i\mathbf{e}_j^T >$ and (a) is because $l(\alpha\mathbf{C} + \beta\mathbf{D}) = \alpha l(\mathbf{C}) + \beta l(\mathbf{D})$.

Now, if we let $I = \|\nabla_{\mathbf{Y}} f(\mathbf{Y})\|_F^2 - c^2 \|\text{grad} f(\mathbf{Y})\|_F^2$, then we have

$$
\begin{aligned}
I &= \sum_{i\neq j}\sum_{u\neq v} \delta_{ij}\delta_{uv} s_{ij}s_{uv} < l(\mathbf{e}_i\mathbf{e}_j^T), l(\mathbf{e}_u\mathbf{e}_v^T) > \\
&\quad -c^2 \sum_{i\neq j}\sum_{u\neq v} \delta_{ij}\delta_{uv} s_{ij}s_{uv} < P_{T_{\mathbf{Y}}\widetilde{\mathcal{Y}}}(l(\mathbf{e}_i\mathbf{e}_j^T)), P_{T_{\mathbf{Y}}\widetilde{\mathcal{Y}}}(l(\mathbf{e}_u\mathbf{e}_v^T)) > \\
&\overset{(a)}{=} \sum_{i\neq j}\sum_{u\neq v} \delta_{ij}\delta_{uv} s_{ij}s_{uv} < l(\mathbf{e}_i\mathbf{e}_j^T), l(\mathbf{e}_u\mathbf{e}_v^T) > \\
&\quad -c^2 \sum_{i\neq j}\sum_{u\neq v} \delta_{ij}\delta_{uv} s_{ij}s_{uv} < P_{T_{\mathbf{Y}}\widetilde{\mathcal{Y}}}(l(\mathbf{e}_i\mathbf{e}_j^T)), l(\mathbf{e}_u\mathbf{e}_v^T) > \\
&= \sum_{i\neq j}\sum_{u\neq v} \delta_{ij}\delta_{uv} s_{ij}s_{uv} < (\mathcal{I} - c^2 P_{T_{\mathbf{Y}}\widetilde{\mathcal{Y}}}) l(\mathbf{e}_i\mathbf{e}_j^T), l(\mathbf{e}_u\mathbf{e}_v^T) > \\
&\leq \sum_{i\neq j}\sum_{u\neq v} \delta_{ij}\delta_{uv} |s_{ij}s_{uv} < (\mathcal{I} - c^2 P_{T_{\mathbf{Y}}\widetilde{\mathcal{Y}}}) l(\mathbf{e}_i\mathbf{e}_j^T), l(\mathbf{e}_u\mathbf{e}_v^T) > | \\
&\overset{(b)}{\leq} \frac{1}{2}\sum_{i\neq j}\sum_{u\neq v} (\delta_{ij}^2 + \delta_{uv}^2) |s_{ij}s_{uv} < (\mathcal{I} - c^2 P_{T_{\mathbf{Y}}\widetilde{\mathcal{Y}}}) l(\mathbf{e}_i\mathbf{e}_j^T), l(\mathbf{e}_u\mathbf{e}_v^T) > | \\
&= \frac{1}{2}\sum_{i\neq j}\sum_{u\neq v} (\delta_{ij} + \delta_{uv}) |s_{ij}s_{uv} < (\mathcal{I} - c^2 P_{T_{\mathbf{Y}}\widetilde{\mathcal{Y}}}) l(\mathbf{e}_i\mathbf{e}_j^T), l(\mathbf{e}_u\mathbf{e}_v^T) > | \\
&= \frac{1}{2}\sum_{i\neq j}\delta_{ij}\sum_{u\neq v} |s_{ij}s_{uv} < (\mathcal{I} - c^2 P_{T_{\mathbf{Y}}\widetilde{\mathcal{Y}}}) l(\mathbf{e}_i\mathbf{e}_j^T), l(\mathbf{e}_u\mathbf{e}_v^T) > | \\
&\quad + \frac{1}{2}\sum_{i\neq j}\delta_{ij}\sum_{u\neq v} |s_{ij}s_{uv} < (\mathcal{I} - c^2 P_{T_{\mathbf{Y}}\widetilde{\mathcal{Y}}}) l(\mathbf{e}_u\mathbf{e}_v^T), l(\mathbf{e}_i\mathbf{e}_j^T) > | \\
&\overset{(c)}{=} \frac{1}{2}\sum_{i\neq j}\delta_{ij}\sum_{u\neq v} |s_{ij}s_{uv} < (\mathcal{I} - c^2 P_{T_{\mathbf{Y}}\widetilde{\mathcal{Y}}}) l(\mathbf{e}_i\mathbf{e}_j^T), l(\mathbf{e}_u\mathbf{e}_v^T) > | \\
&\quad + \frac{1}{2}\sum_{i\neq j}\delta_{ij}\sum_{u\neq v} |s_{ij}s_{uv} < l(\mathbf{e}_u\mathbf{e}_v^T), (\mathcal{I} - c^2 P_{T_{\mathbf{Y}}\widetilde{\mathcal{Y}}}) l(\mathbf{e}_i\mathbf{e}_j^T) > | \\
&= \sum_{i\neq j}\delta_{ij}\sum_{u\neq v} |s_{ij}s_{uv} < (\mathcal{I} - c^2 P_{T_{\mathbf{Y}}\widetilde{\mathcal{Y}}}) l(\mathbf{e}_i\mathbf{e}_j^T), l(\mathbf{e}_u\mathbf{e}_v^T) > |,
\end{aligned}
$$

where (a) is because

$$
\begin{aligned}
< P_{T_{\mathbf{Y}}\widetilde{\mathcal{Y}}}(\mathbf{E}), \mathbf{F} > &= < P_{T_{\mathbf{Y}}\widetilde{\mathcal{Y}}}(\mathbf{E}), P_{T_{\mathbf{Y}}\widetilde{\mathcal{Y}}}(\mathbf{F}) > + < P_{T_{\mathbf{Y}}\widetilde{\mathcal{Y}}}(\mathbf{E}), P^{\perp}_{T_{\mathbf{Y}}\widetilde{\mathcal{Y}}}(\mathbf{F}) > \\
&= < P_{T_{\mathbf{Y}}\widetilde{\mathcal{Y}}}(\mathbf{E}), P_{T_{\mathbf{Y}}\widetilde{\mathcal{Y}}}(\mathbf{F}) >,
\end{aligned}
$$

(b) is because $x^2 + y^2 \geq 2xy$ $(x, y \geq 0)$, and (c) is because $(\mathcal{I} - c^2 P_{T_{\mathbf{Y}}\widetilde{\mathcal{Y}}})$ is a self-adjoint operator. $\square$

# Chapter L

# Proof of Lemma 20

**Proof:** Recalling that $l(\mathbf{A}) = 2\text{eye}(\text{Sym}(\mathbf{A})\mathbf{1}) - 2\text{Sym}(\mathbf{A})$, we have

$$\begin{aligned} l(\mathbf{e}_i\mathbf{e}_j^T) &= 2\text{eye}(\text{Sym}(\mathbf{e}_i\mathbf{e}_j^T)\mathbf{1}) - 2\text{Sym}(\mathbf{e}_i\mathbf{e}_j^T) \\ &= \text{eye}(\mathbf{e}_i\mathbf{e}_j^T\mathbf{1} + \mathbf{e}_j\mathbf{e}_i^T\mathbf{1}) - \mathbf{e}_i\mathbf{e}_j^T - \mathbf{e}_j\mathbf{e}_i^T \\ &\overset{(a)}{=} \text{eye}(\mathbf{e}_i + \mathbf{e}_j) - \mathbf{e}_i\mathbf{e}_j^T - \mathbf{e}_j\mathbf{e}_i^T \\ &= \mathbf{e}_i\mathbf{e}_i^T + \mathbf{e}_j\mathbf{e}_j^T - \mathbf{e}_i\mathbf{e}_j^T - \mathbf{e}_j\mathbf{e}_i^T \\ &= (\mathbf{e}_i - \mathbf{e}_j)(\mathbf{e}_i - \mathbf{e}_j)^T, \end{aligned}$$

where (a) is because $\text{eye}(\mathbf{e}_i\mathbf{e}_j^T\mathbf{1}) = \text{eye}(\mathbf{e}_i)$. Let $\boldsymbol{\delta} = \mathbf{e}_i - \mathbf{e}_j$, then $l(\mathbf{e}_i\mathbf{e}_j^T) = \boldsymbol{\delta}\boldsymbol{\delta}^T$.

Also, if $n^2 \geq 4c\mu(\mathbf{Y})^2k^2$, then

$$
\begin{aligned}
& | < (\mathcal{I} - cP_{T_{\mathbf{Y}}\widetilde{\mathcal{Y}}})l(\mathbf{e}_i\mathbf{e}_j^T), l(\mathbf{e}_i\mathbf{e}_j^T) > | \\
= \; & | < (\mathcal{I} - cP_{T_{\mathbf{Y}}\widetilde{\mathcal{Y}}})\boldsymbol{\delta}\boldsymbol{\delta}^T, \boldsymbol{\delta}\boldsymbol{\delta}^T > | \\
= \; & | < \boldsymbol{\delta}\boldsymbol{\delta}^T - cP_{T_{\mathbf{Y}}\widetilde{\mathcal{Y}}}(\boldsymbol{\delta}\boldsymbol{\delta}^T), \boldsymbol{\delta}\boldsymbol{\delta}^T > | \\
= \; & | < \boldsymbol{\delta}\boldsymbol{\delta}^T, \boldsymbol{\delta}\boldsymbol{\delta}^T > - c < P_{T_{\mathbf{Y}}\widetilde{\mathcal{Y}}}(\boldsymbol{\delta}\boldsymbol{\delta}^T), \boldsymbol{\delta}\boldsymbol{\delta}^T > | \\
\overset{(a)}{=} \; & | < \boldsymbol{\delta}\boldsymbol{\delta}^T, \boldsymbol{\delta}\boldsymbol{\delta}^T > - c < \mathbf{P}_Q\boldsymbol{\delta}\boldsymbol{\delta}^T + \boldsymbol{\delta}\boldsymbol{\delta}^T\mathbf{P}_Q - \mathbf{P}_Q\boldsymbol{\delta}\boldsymbol{\delta}^T\mathbf{P}_Q, \boldsymbol{\delta}\boldsymbol{\delta}^T > | \\
= \; & | < \boldsymbol{\delta}\boldsymbol{\delta}^T, \boldsymbol{\delta}\boldsymbol{\delta}^T > - c(< \mathbf{P}_Q\boldsymbol{\delta}\boldsymbol{\delta}^T, \boldsymbol{\delta}\boldsymbol{\delta}^T > + < \boldsymbol{\delta}\boldsymbol{\delta}^T\mathbf{P}_Q, \boldsymbol{\delta}\boldsymbol{\delta}^T > \\
& - < \mathbf{P}_Q\boldsymbol{\delta}\boldsymbol{\delta}^T\mathbf{P}_Q, \boldsymbol{\delta}\boldsymbol{\delta}^T >)| \\
\overset{(b)}{=} \; & |\boldsymbol{\delta}^T\boldsymbol{\delta}\boldsymbol{\delta}^T\boldsymbol{\delta} - c(\boldsymbol{\delta}^T\mathbf{P}_Q\boldsymbol{\delta}\boldsymbol{\delta}^T\boldsymbol{\delta} + \boldsymbol{\delta}^T\boldsymbol{\delta}\boldsymbol{\delta}^T\mathbf{P}_Q\boldsymbol{\delta} - \boldsymbol{\delta}^T\mathbf{P}_Q\boldsymbol{\delta}\boldsymbol{\delta}^T\mathbf{P}_Q\boldsymbol{\delta})| \\
\overset{(c)}{=} \; & |\|\boldsymbol{\delta}\|_2^4 - c(2\|\mathbf{P}_Q\boldsymbol{\delta}\|_2^2\|\boldsymbol{\delta}\|_2^2 - \|\mathbf{P}_Q\boldsymbol{\delta}\|_2^4)| \\
\overset{(d)}{=} \; & |4 - 4c\|\mathbf{P}_Q\boldsymbol{\delta}\|_2^2 + c\|\mathbf{P}_Q\boldsymbol{\delta}\|_2^4|. \\
\overset{(e)}{\geq} \; & 4 - 4c\frac{4\mu(\mathbf{Y})^2r^2}{n^2} + c\frac{16\mu(\mathbf{Y})^4r^4}{n^4} \\
\geq \; & 4\left(1 - \frac{4c\mu(\mathbf{Y})^2r^2}{n^2}\right),
\end{aligned}
$$

where (a) follows from Proposition 7, (b) is because $< \mathbf{X}, \mathbf{z}\mathbf{z}^T >= tr(\mathbf{X}\mathbf{z}\mathbf{z}^T) = \mathbf{z}^T\mathbf{X}\mathbf{z}$, (c) is because $\mathbf{P}_Q^T\mathbf{P}_Q = \mathbf{P}_Q$ and thus $\boldsymbol{\delta}^T\mathbf{P}_Q\boldsymbol{\delta} = \boldsymbol{\delta}^T\mathbf{P}_Q^T\mathbf{P}_Q\boldsymbol{\delta} = \|\mathbf{P}_Q\boldsymbol{\delta}\|_2^2$, (d) is because $\|\boldsymbol{\delta}\|_2^2 = 2$ for $i \neq j$, and (e) is because $\|\mathbf{P}_Q\boldsymbol{\delta}\|_2 = \|\mathbf{P}_Q\mathbf{e}_i - \mathbf{P}_Q\mathbf{e}_j\|_2 \leq \|\mathbf{P}_Q\mathbf{e}_i\|_2 + \|\mathbf{P}_Q\mathbf{e}_j\|_2 \leq 2\mu(\mathbf{Y})r/n$ (see Definition 5). □

# Chapter M

# Proof of Lemma 21

**Proof:** For any $t>0$, we have

$$
\begin{aligned}
P(\sum_{i=1}^{N}\delta_i a_i \geq \epsilon) &= P(\exp(t\sum_{i=1}^{N}\delta_i a_i) \geq \exp(t\epsilon)) \\
&\overset{(a)}{\leq} \exp(-t\epsilon)E\left[\exp(t\sum_{i=1}^{N}\delta_i a_i)\right] \\
&\overset{(b)}{\leq} \exp(-t\epsilon + t\sum_{i=2^n+1}^{N} a_i)E\left[\exp(t\sum_{i=1}^{2^n}\delta_i a_i)\right] \\
&= \exp(-t\epsilon + t\sum_{i=2^n+1}^{N} a_i)E\left[\prod_{i=1}^{2^n}\exp(t\delta_i a_i)\right] \\
&\overset{(c)}{\leq} \exp(-t\epsilon + t\sum_{i=2^n+1}^{N} a_i)\left(\prod_{i=1}^{2^n}E\left[(\exp(t\delta_i a_i))^{2^n}\right]\right)^{\frac{1}{2^n}} \\
&= \exp(-t\epsilon + t\sum_{i=2^n+1}^{N} a_i)\left(\prod_{i=1}^{2^n}E\left[\exp(t\delta_i a_i 2^n)\right]\right)^{\frac{1}{2^n}} \\
&\overset{(d)}{=} \exp(-t\epsilon + t\sum_{i=2^n+1}^{N} a_i)\left(\prod_{i=1}^{2^n}(1-p+p\exp(t2^n a_i))\right)^{\frac{1}{2^n}}
\end{aligned}
$$

$$
\begin{aligned}
&= \exp(-t\epsilon + t\sum_{i=2^n+1}^{N} a_i) \\
&\quad \left(\prod_{i=1}^{2^n} \exp(t2^n a_i)((1-p)\exp(-t2^n a_i)+p)\right)^{\frac{1}{2^n}} \\
&= \exp(-t\epsilon + t\sum_{i=2^n+1}^{N} a_i) \\
&\quad \left(\prod_{i=1}^{2^n} \exp(t2^n a_i)\right)^{\frac{1}{2^n}} \left(\prod_{i=1}^{2^n}((1-p)\exp(-t2^n a_i)+p)\right)^{\frac{1}{2^n}} \\
&= \exp(-t\epsilon + t\sum_{i=2^n+1}^{N} a_i) \\
&\quad \prod_{i=1}^{2^n} \exp(ta_i) \left(\prod_{i=1}^{2^n}((1-p)\exp(-t2^n a_i)+p)\right)^{\frac{1}{2^n}} \\
&= \exp(-t\epsilon + t\sum_{i=2^n+1}^{N} a_i) \\
&\quad \exp(t\sum_{i=1}^{2^n} a_i) \left(\prod_{i=1}^{2^n}((1-p)\exp(-t2^n a_i)+p)\right)^{\frac{1}{2^n}} \\
&= \exp(-t\epsilon + t\sum_{i=1}^{N} a_i) \left(\prod_{i=1}^{2^n}((1-p)\exp(-t2^n a_i)+p)\right)^{\frac{1}{2^n}} \\
&\overset{(e)}{\leq} \exp(-t\epsilon + t\sum_{i=1}^{N} a_i) \left(\prod_{i=1}^{2^n}((1-p)\exp(-t2^n a_{\min})+p)\right)^{\frac{1}{2^n}} \\
&= \exp(-t\epsilon + t\sum_{i=1}^{N} a_i)((1-p)\exp(-t2^n a_{\min})+p),
\end{aligned}
$$

where (a) follows from the Markov's inequality, (b) is because $\delta_i \leq 1$ for all $i$, (c) is because $E[\prod_{i=1}^{M} X_i] \leq (\prod_{i=1}^{M} E[X_i^M])^{1/M}$ for positive random variables $X_i$ and $M = 2^q$ ($q \geq 1$), (d) is because $\delta_i$ is Bernoulli random variable with $P(\delta_i = 1) = p$, and (e) is because $a_{\min} = \min_i a_i$.

Let $g(t) = \exp(-t\epsilon + t\sum_{i=1}^{N} a_i)((1-p)\exp(-t2^n a_{\min})+p)$, then the minimum of $g(t)$ is obtained at $t^* = 1/(2^n a_{\min})\ln((1-m\epsilon)(1-p)/(m\epsilon p))$ ($m = (\sum_{i=1}^{N} a_i -$

$\epsilon)/(2^n \epsilon a_{\min}))$. Thus, we have

$$\begin{aligned} P(\sum_{i=1}^{N} \delta_i a_i \geq \epsilon) &\leq g(t^*) \\ &= \left(\frac{p}{1-m\epsilon}\right)^{1-m\epsilon} \left(\frac{1-p}{m\epsilon}\right)^{m\epsilon} \\ &= \exp\left(-\left(m\epsilon \log\left(\frac{m\epsilon}{1-p}\right) + (1-m\epsilon)\log\left(\frac{1-m\epsilon}{p}\right)\right)\right) \end{aligned}$$

when $0<m\epsilon<1-p$, which establishes the lemma. □

# Bibliography


[1] Netflix prize. [Online]. Available: http://www.netflixprize.com

[2] A. Pal, "Localization algorithms in wireless sensor networks: Current approaches and future challenges," *Netw. Protocols Algorithms*, vol. 2, no. 1, pp. 45–74, 2010.

[3] L. Nguyen, S. Kim, and B. Shim, "Localization in internet of things network: Matrix completion approach," *in Proc. Inform. Theory Appl. Workshop, San Diego, CA, USA*, pp. 1–4, 2016.

[4] M. Fazel, "Matrix rank minimization with applications," *Ph.D. dissertation, Elec. Eng. Dept., Standford Univ., Stanford, CA*, 2002.

[5] E. J. Candes and B. Recht, "Exact matrix completion via convex optimization," *Found. Comput. Math.*, vol. 9, no. 6, pp. 717–772, Dec. 2009.

[6] E. J. Candes and T. Tao, "The power of convex relaxation: Near-optimal matrix completion," *IEEE Trans. Inform. Theory*, vol. 56, no. 5, pp. 2053–2080, May 2010.

[7] E. J. Candes, Y. C. Eldar, and T. Strohmer, "Phase retrieval via matrix completion," *SIAM Rev.*, vol. 52, no. 2, pp. 225–251, May 2015.

[8] T. Blumensath and M. E. Davies, "Iterative hard thresholding for compressed sensing," *Appl. Comput. Harmon. Anal.*, vol. 27, no. 3, pp. 265–274, Nov. 2009.

[9] J. Tanner and K. Wei, "Normalized iterative hard thresholding for matrix completion," *SIAM J. Sci. Comput.*, vol. 35, no. 5, pp. S104–S125, Oct. 2013.

[10] J. F. Cai, E. J. Candes, and Z. Shen, "A singular value thresholding algorithm for matrix completion," *SIAM J. Optim.*, vol. 20, no. 4, pp. 1956–1982, Mar. 2010.

[11] K. Lee and Y. Bresler, "Admira: Atomic decomposition for minimum rank approximation," *IEEE Trans. Inform. Theory*, vol. 56, no. 9, pp. 4402–4416, Sep. 2010.

[12] Z. Wen, W. Yin, and Y. Zhang, "Solving a low-rank factorization model for matrix completion by a nonlinear successive over-relaxation algorithm," *Math. Prog. Comput.*, vol. 4, no. 4, pp. 333–361, Dec. 2012.

[13] J. P. Haldar and D. Hernando, "Rank-constrained solutions to linear matrix equations using power factorization," *IEEE Signal Process. Lett.*, vol. 16, no. 7, pp. 584–587, Jul. 2009.

[14] J. Tanner and K. Wei, "Low rank matrix completion by alternating steepest descent methods," *Appl. Comput. Harmon. Anal.*, vol. 40, no. 2, pp. 417–429, Mar. 2016.

[15] T. Ngo and Y. Saad, "Scaled gradients on grassmann manifolds for matrix completion," *in Proc. Adv. Neural Inform. Process. Syst. Conf., Lake Tahoe, Nevada, USA*, pp. 1412–1420, 2012.

[16] R. Escalante and M. Raydan, "Alternating projection methods," *Philadelphia, PA, USA: SIAM*, 2011.

[17] Z. Wang, M.-J. Lai, Z. Lu, W. Fan, H. Davulcu, and J. Ye, "Rank-one matrix pursuit for matrix completion," *in Proc. Int. Conf. Mach. Learn., Beijing, China*, pp. 91–99, 2014.

[18] Y. Hu, D. Zhan, J. Ye, X. Li, and X. He, "Fast and accurate matrix completion via truncated nuclear norm regularization," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 9, pp. 2117–2130, Sep. 2013.

[19] B. Mishra, G. Meyer, S. Bonnabel, and R. Sepulchre, "Fixed-rank matrix factorizations and riemannian low-rank optimization," *Comput. Stat.*, vol. 3, no. 4, pp. 591–621, 2014.

[20] B. Vandereycken, "Low-rank matrix completion by riemannian optimization," *SIAM J. Optim.*, vol. 23, no. 2, pp. 1214–1236, Jun. 2013.

[21] W. Dai and O. Milenkovic, "Set: An algorithm for consistent matrix completion," *in Proc. Int. Conf. Acoust., Speech, Signal Process., Dallas, Texas, USA*, pp. 3646–3649, 2010.

[22] L. T. Nguyen, J. Kim, and B. Shim, "Low-rank matrix completion: A contemporary survey," *IEEE Access*, vol. 7, no. 1, pp. 94 215–94 237, Jul. 2019.

[23] L. T. Nguyen, J. Kim, S. Kim, and B. Shim, "Localization of IoT networks via low-rank matrix completion," *IEEE Trans. Commun.*, vol. 67, no. 8, pp. 5833–5847, Aug. 2019.

[24] L. T. Nguyen and B. Shim, "Low-rank matrix completion via graph neural network," *Submitted to IEEE Signal Process. Lett.*, 2019.

[25] M. Delamom, S. Felici-Castell, J. J. Perez-Solano, and A. Foster, "Designing an open source maintenance-free environmental monitoring application for wireless sensor networks," *J. Syst. Softw.*, vol. 103, pp. 238–247, May 2015.

[26] G. Hackmann, W. Guo, G. Yan, Z. Sun, C. Lu, and S. Dyke, "Cyber-physical codesign of distributed structural health monitoring with wireless sensor networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 1, pp. 63–72, Jan. 2014.

[27] W. S. Torgerson, "Multidimensional scaling: I. theory and method," *Psychometrika*, vol. 17, no. 4, pp. 401–419, Dec. 1952.

[28] H. Ji, Y. Kim, J. Lee, E. Onggosanusi, Y. Nam, J. Zhang, B. Lee, and B. Shim, "Overview of full-dimension mimo in lte-advanced pro," *IEEE Commun. Mag.*, vol. 55, no. 2, pp. 176–184, Feb. 2017.

[29] W. Shen, L. Dai, B. Shim, S. Mumtaz, and Z. Wang, "Joint csit acquisition based on low-rank matrix completion for fdd massive mimo systems," *IEEE Commun. Lett.*, vol. 19, no. 12, pp. 2178–2181, Dec. 2015.

[30] T. L. Marzetta and B. M. Hochwald, "Fast transfer of channel state information in wireless systems," *IEEE Trans. Signal Process.*, vol. 54, no. 4, pp. 1268–1278, Apr. 2006.

[31] F. Rusek, D. Persson, B. K. Lau, E. G. Larsson, T. L. Marzetta, O. Edfors, and F. Tufvesson, "Scaling up mimo: Opportunities and challenges with very large arrays," *IEEE Signal Process. Mag.*, vol. 30, no. 1, pp. 40–60, Jan. 2013.

[32] T. S. Rappaport and et al., "Millimeter wave mobile communications for 5g cellular: It will work!" *IEEE Access*, vol. 1, no. 1, pp. 335–349, May 2013.

[33] X. Li, J. Fang, H. Li, H. Li, and P. Wang, "Millimeter wave channel estimation via exploiting joint sparse and low-rank structures," *IEEE Trans. Wireless Commun.*, vol. 17, no. 2, pp. 1123–1133, Feb. 2018.

[34] P. Wang, M. Pajovic, P. V. Orlik, T. Koike-Akino, K. J. Kim, and J. Fang, "Sparse channel estimation in millimeter wave communications: Exploiting joint aod-aoa angular spread," *in Proc. IEEE Int. Conf. Commun. (ICC), Paris, France*, p. 1–6, May 2017.

[35] Y. Shi, J. Zhang, and K. B. Letaief, "Low-rank matrix completion for topological interference management by riemannian pursuit," *IEEE Trans. Wireless Commun.*, vol. 15, no. 7, pp. 4703–4717, Jul. 2016.

[36] Y. Shi, B. Mishra, and W. Chen, "Topological interference management with user admission control via riemannian optimization," *IEEE Trans. Wireless Commun.*, vol. 16, no. 11, pp. 7362–7375, Nov. 2017.

[37] Y. Shi, J. Zhang, W. Chen, and K. B. Letaief, "Generalized sparse and low-rank optimization for ultra-dense networks," *IEEE Commun. Mag.*, vol. 56, no. 6, pp. 42–48, Jun. 2018.

[38] G. Sridharan and W. Yu, "Linear beamforming design for interference alignment via rank minimization," *IEEE Trans. Signal Process.*, vol. 63, no. 22, pp. 5910–5923, Nov. 2015.

[39] M. Peng, S. Yan, K. Zhang, and C. Wang, "Fog-computing-based radio access networks: issues and challenges," *IEEE Network*, vol. 30, pp. 46–53, Jul. 2016.

[40] K. Yang, Y. Shi, and Z. Ding, "Low-rank matrix completion for mobile edge caching in fog-ran via riemannian optimization," *in Proc. IEEE Global Communications Conf. (GLOBECOM), Washington, DC*, Dec. 2016.

[41] B. Recht, "A simpler approach to matrix completion," *J. Mach. Learning Research*, vol. 12, pp. 3413–3430, Dec. 2011.

[42] C. R. Berger, "Double exponential," *IEEE Trans. Signal Process.*, vol. 56, no. 5, pp. 1708–1721, 2010.

[43] S. Boyd and Van, "Convex optimization," *Cambridge, England: Cambridge Univ.*, 2004.

[44] J. W. Choi, B. Shim, Y. Ding, B. Rao, and D. I. Kim, "Compressed sensing for wireless communications: Useful tips and tricks," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1527–1550, Feb. 2017.

[45] S. Kwon, J. Wang, and B. Shim, "Multipath matching pursuit," *IEEE Trans. Inform. Theory*, vol. 60, no. 5, pp. 2986–3001, Mar. 2014.

[46] J. Wang, S. Kwon, and B. Shim, "Generalized orthogonal matching pursuit," *IEEE Trans. Signal Process.*, vol. 60, no. 12, pp. 6202–6216, Sep. 2012.

[47] K. C. Toh, M. J. Todd, and R. H. Tutuncu, "Sdpt3 — a matlab software package for semidefinite programming," *Optim. Methods Softw.*, vol. 11, pp. 545–581, 1999.

[48] J. F. Sturm, "Using sedumi 1.02, a matlab toolbox for optimization over symmetric cones," *Optim. Methods Softw.*, vol. 11, pp. 625–653, 1999.

[49] L. Vandenberghe and S. Boyd, "Semidefinite programming," *SIAM Rev.*, vol. 38, no. 1, pp. 49–95, 1996.

[50] Y. Zhang, "On extending some primal-dual interior-point algorithms from linear programming to semidefinite programming," *SIAM J. Optim.*, vol. 8, no. 2, pp. 365–386, 1998.

[51] Y. E. Nesterov and M. Todd, "Primal-dual interior-point methods for self-scaled cones," *SIAM J. Optim.*, vol. 8, no. 2, pp. 324–364, 1998.

[52] F. A. Potra and R. Sheng, "A superlinearly convergent primal-dual infeasible-interior-point algorithm for semidefinite programming," *SIAM J. Optim.*, vol. 8, no. 4, pp. 1007–1028, 1998.

[53] L. Vandenberghe, V. R. Balakrishnan, R. Wallin, A. Hansson, and T. Roh, "Interior-point algorithms for semidefinite programming problems derived

from the kyp lemma," *in Positive polynomials in control. Berlin, Heidelberg: Springer*, pp. 195–238, 2005.

[54] F. A. Potra and S. J. Wright, "Interior-point methods," *J. Comput. Appl. Math.*, vol. 124, no. 1-2, pp. 281–302, 2000.

[55] B. Recht, M. Fazel, and P. A. Parillo, "Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization," *SIAM Rev.*, vol. 52, no. 3, pp. 471–501, 2010.

[56] P. Combettes and J. C. Pesquet, "Proximal splitting methods in signal processing," *New York, NY, USA: Springer*, 2011.

[57] P. Jain, R. Meka, and I. Dhillon, "Guaranteed rank minimization via singular value projection," *in Proc. Neural Inform. Process. Syst. Conf., Vancouver, Canada*, pp. 937–945, 2010.

[58] M. Fornasier, H. Rauhut, and R. Ward, "Low-rank matrix recovery via iteratively reweighted least squares minimization," *SIAM J. Optim.*, vol. 21, no. 4, pp. 1614–1640, Dec. 2011.

[59] K. Mohan and M. Fazel, "Iterative reweighted algorithms for matrix rank minimization," *J. Mach. Learning Research*, vol. 13, pp. 3441–3473, Nov. 2012.

[60] D. Needell and J. A. Tropp, "Cosamp: Iterative signal recovery from incomplete and inaccurate samples," *Appl. Comput. Harmon. Anal.*, vol. 26, no. 3, pp. 301–321, May 2009.

[61] J. A. Tropp and A. C. Gilbert, "Signal recovery from random measurements via orthogonal matching pursuit," *IEEE Trans. Inform. Theory*, vol. 53, no. 12, pp. 4655–4666, Dec. 2007.

[62] U. Helmke and J. B. Moore, "Optimization and dynamical systems," *New York, NY, USA: Springer*, 1994.

[63] P. A. Absil, R. Mahony, and R. Sepulchre, *Optimization Algorithms on Matrix Manifolds*. Princeton Univ. Press, 2008.

[64] J. Lee, *Introduction to Smooth Manifolds*, 2nd ed. New York: NY:Springer, 2013, vol. 218.

[65] M. Tao and X. Yuan, "Recovering low-rank and sparse components of matrices from incomplete and noisy observations," *SIAM J. Optim.*, vol. 21, no. 1, pp. 57–81, Jan. 2011.

[66] Z. Lin, R. Liu, and Z. Su, "Linearized alternating direction method with adaptive penalty for low-rank representation," *in Proc. Adv. Neural Inform. Process. Syst., Montreal, Canada*, pp. 612–620, 2011.

[67] B. S. He, H. Yang, and S. L. Wang, "Alternating direction method with self-adaptive penalty parameters for monotone variational inequalities," *J. Optim. Theory Appl.*, vol. 106, no. 2, pp. 337–356, Aug. 2000.

[68] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM J. Imaging Sci.*, vol. 2, no. 1, pp. 183–202, Mar. 2009.

[69] J. Y. Gotoh, A. Takeda, and K. Tono, "Dc formulations and algorithms for sparse optimization problems," *Math. Programming*, pp. 1–36, May 2018.

[70] R. Ge, J. D. Lee, and T. Ma, "Matrix completion has no spurious local minimum," *in Advances Neural Inform. Process. Syst.*, pp. 2973–2981, 2016.

[71] R. Ge, C. Jin, and Y. Zheng, "No spurious local minima in nonconvex low rank problems: A unified geometric analysis," *in Proc. 34th Int. Conf. on Machine Learning, JMLR. org.*, vol. 70, pp. 1233–1242, Aug. 2017.

[72] S. S. Du, C. Jin, J. D. Lee, M. I. Jordan, A. Singh, and B. Poczos, "Gradient descent can take exponential time to escape saddle points," *in Advances Neural Inform. Process. Syst.*, pp. 1067–1077, 2017.

[73] F. Monti, M. Bronstein, and X. Bresson, "Geometric matrix completion with recurrent multi-graph neural networks," *in Proc. Adv. Neural Inform. Process. Syst., Long Beach, CA, USA*, pp. 3700–3710, 2017.

[74] D. L. Donoho, A. Maleki, and A. Montanari, "Message passing algorithms for compressed sensing," *Proc. Nat. Acad. Sci.*, vol. 106, no. 45, pp. 18 914–18 919, Nov. 2009.

[75] R. Parker and S. Valaee, "Vehicular node localization using received-signal-strength indicator," *IEEE Trans. Veh. Technol.*, vol. 56, pp. 3371–3380, Nov. 2007.

[76] D. Dardari, C.-C. Chong, and M. Z. Win, "Threshold-based time-of-arrival estimators in uwb dense multipath channels," *IEEE Trans. Commun.*, vol. 56, pp. 1366–1378, Aug. 2008.

[77] Y. Zhang and J. Zha, "Indoor localization using time difference of arrival and time-hopping impulse radio," *IEEE Int. Symp. Commun. Inform. Technol.*, pp. 964–967, Oct. 2005.

[78] Y. Shang, W. Ruml, Y. Zhang, and M. Fromherz, "Localization from mere connectivity," in *Proc. ACM Symp. Mobile Ad Hoc Netw. Comput.*, Annapolis, Maryland, USA, Jun. 2003, pp. 201–212.

[79] P. Rawat, K. D. Singh, H. Chaouchi, and J. M. Bonnin, "Wireless sensor networks: a survey on recent developments and potential synergies," *J. Supercomput.*, vol. 68, no. 1, pp. 1–48, Apr. 2014.

[80] Z. Jianwu and Z. Lu, "Research on distance measurement based on rssi of zigbee," *ISECS Int. Colloq. Computing, Commun., Control, and Manage.*, pp. 210–212, 2009.

[81] U. Helmke and J. B. Moore, *Optimization and Dynamical Systems*. London: Springer-Verlag, 1994.

[82] P. A. Absil and J. Malick, "Projection-like retractions on matrix manifolds," *SIAM J. Optimiz.*, vol. 22, pp. 135–158, 2012.

[83] Y. H. Dai, "Nonlinear conjugate gradient methods," *Wiley Encyclopedia of Operations Research and Manage. Sci.*, 2011.

[84] L. Armijo, "Minimization of functions having lipschitz continuous first partial derivatives," *Pacific J. Math.*, no. 1, 1966.

[85] C. T. Kelley, *Iterative methods for optimization*. Philadelphia: PA: Frontiers in Applied Mathematics, 1999.

[86] M. R. Hestenes and E. Stiefel, "Methods of conjugate gradients for solving linear systems," *NBS*, 1952.

[87] R. Fletcher and C. M. Reeves, "Function minimization by conjugate gradients," *J. Comput.*, no. 2, pp. 149–154, 1964.

[88] Y. H. Dai and Y. Yuan, "A nonlinear conjugate gradient method with a strong global convergence property," *SIAM J. Optimiz.*, no. 1, pp. 177–182, 1999.

[89] W. W. Hager and H. Zhang, "A new conjugate gradient method with guaranteed descent and an efficient line search," *SIAM J. Optimiz.*, no. 1, pp. 170–192, 2005.

[90] P. Wolfe, "Convergence conditions for ascent methods," *SIAM Rev.*, no. 2, pp. 226–235, 1969.

[91] H. Sato and T. Iwai, "A new, globally convergent riemannian conjugate gradient method," *Optim. J. Math. Program. Oper. Res.*, no. 4, pp. 1011–1031, 2015.

[92] Z. Yang, C. Wu, T. Chen, Y. Zhao, W. Gong, and Y. Liu, "Detecting outlier measurements based on graph rigidity for wireless sensor network localization," *IEEE Trans. Veh. Technol.*, vol. 62, no. 1, pp. 374–383, Jan. 2013.

[93] Z. Lin, M. Chen, and Y. Ma., "The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices," 2010. [Online]. Available: http://arxiv.org/abs/1009.5055

[94] Z. Lin, A. Ganesh, J. Wright, L. Wu, M. Chen, and Y. Ma, "Fast convex optimization algorithms for exact recovery of a corrupted low-rank matrix," *in Proc. Int. Workshop Comput. Adv. Multi-Sensor Adapt. Process.*, pp. 1–18, 2009.

[95] P. Biswas and Y. Ye, "Semidefinite programming for ad hoc wireless sensor network localization," *in Proc. Int. Symp. Inform. Process. Sensor Netw., California*, pp. 46–54, 2004.

[96] X. Guo, L. Chu, and X. Sun, "Accurate localization of multiple sources using semidefinite programming based on in complete range matrix," *IEEE Sensors J.*, vol. 16, no. 13, pp. 5319–5324, Jul. 2016.

[97] J. A. Costa, N. Patwari, and I. A. O. Hero, "Distributed weighted-multidimensional scaling for node localization in sensor networks," *ACM Trans. Sensor Netw.*, vol. 2, no. 1, pp. 39–64, 2006.

[98] R. H. Tutuncu, K. C. Toh, and M. J. Todd, "Solving semidefinite quadratic linear programs using sdpt3," *Math. Programming Ser. B*, vol. 95, pp. 189–217, 2003.

[99] M. Niepert, M. Ahmed, and K. Kutzkov, "Learning convolutional neural networks for graphs," *in Proc. Int. Conf. Mach. Learn.*, pp. 2014–2023, 2016.

[100] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and deep locally connected networks on graphs," *arXiv:1312.6203*, 2013.

[101] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," *in Proc. Adv. Neural Inform. Process. Syst., Barcelona, Spain*, pp. 3844–3852, 2016.

[102] D. K. Hammond, P. Vandergheynst, and R. Gribonval, "Wavelets on graphs via spectral graph theory," *Appl. Comput. Harmon. Anal.*, vol. 30, no. 2, pp. 129–150, Mar. 2011.

[103] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Trans. Neural Netw.*, vol. 20, no. 1, pp. 61–80, Jan. 2009.

[104] W. Cheney, *Analysis for applied mathematics*. New York: Springer, 2013.

[105] V. Hutson, J. Pym, and M. Cloud, *Applications of functional analysis and operator theory*. Elsevier, 2005.

# 초 록

최근, 일부의 관측치로부터 행렬의 모든 원소들을 복원하는 방법으로 저 랭크 행렬 완성 (LRMC)이 많은 주목을 받고 있다. LRMC는 추천 시스템, 위상 복원, 사물 인터넷 지역화, 영상 잡음 제거, 밀리미터 웨이브 통신 등을 포함한 다양한 응용분야에서 사용되고 있다. 본 논문에서는 LRMC에 대해 연구하여 LRMC의 가능성과 한계에 대한 더 나은 이해를 할 수 있도록 기존 결과들을 구조적이고 접근 가능한 방식으로 분류한다. 구체적으로, 최신 LRMC 기법들을 두 가지 범주로 분류한 다음 각각의 범주를 분석한다. 특히, 행렬의 고유한 성질과 같은 LRMC 기법을 사용할 때 고려해야 할 사항들을 분석한다. 기존의 LRMC 기법은 가우시안 랜덤 행렬과 같은 일반적인 상황에서 성공적이었으나 많은 실제 상황에서는 복원하고자 하는 저 랭크 행렬이 그래프 구조 또는 다양체 구조와 같은 비 유클리드 구조를 가질 수 있다.

본 논문에서는 실제 응용에서 LRMC의 성능을 향상시키기 위해 이런 추가적인 구조가 활용될 수 있음을 보인다. 특히, 사물 인터넷 네트워크 지역화를 위한 유클리드 거리 행렬 완성 알고리듬을 제안한다. 유클리드 거리 행렬을 낮은 랭크를 갖는 양의 준정부호 행렬의 함수로 표현한다. 이러한 양의 준정부호 행렬들의 집합은 미분이 잘 정의되어 있는 리만 다양체를 형성하므로 유클리드 공간에서의 알고리듬을 적당히 변형하여 LRMC에 사용할 수 있다. LRMC를 위해 우리는 켤레 기울기를 활용한 리만 다양체에서의 지역화 (LRM-CG)라 불리는 변경된 켤레 기울기 기반 알고리듬을 제안한다. 제안하는 LRM-CG 알고리듬은 관측된 쌍 거리가 특이값에 의해 오염되는 시나리오로 쉽게 확장 될 수 있음을 보인다. 실제로 특이값을 희소 행렬로 모델링 한 다음 특이값 행렬을 규제 항으로 LRMC에 추가함으로써 특이값

을 효과적으로 제어 할 수 있다. 분석을 통해 LRM-CG 알고리듬이 확장된 Wolfe 조건 아래 원래 유클리드 거리 행렬에 선형적으로 수렴하는 것을 보인다. 모의 실험을 통해 LRM-CG와 확장 버전이 유클리드 거리 행렬을 복구하는 데 효과적임을 보인다.

또한, 그래프 모델을 사용하여 표현될 수 있는 저 랭크 행렬 복원을 위한 그래프 신경망 (GNN) 기반 기법을 제안한다. 그래프 신경망 기반의 LRMC (GNN-LRMC) 라 불리는 기법은 복원하고자 하는 행렬의 그래프 영역 특징들을 추출하기 위해 변형된 합성곱 연산을 사용한다. 이렇게 추출된 특징들을 GNN의 학습 과정에 활용하여 행렬의 원소들을 복원할 수 있다. 합성 및 실제 데이터를 사용한 모의 실험을 통하여 제안하는 GNN- LRMC의 우수한 복구 성능을 보였다.

**주요어**: 저 랭크 행렬 완성, 프로베니우스 놈 최소화, 지역화, 리만 최적화, 그래프 신경망

**학번**: 2015-30751

# ACKNOWLEGEMENT

First and foremost, I would like to thank my adviser Professor Byonghyo Shim. His wealth of idea, clarity of thought, enthusiasm, and energy have made working with him an exceptional experience for me. I am very thankful and feel very fortunate to become his student. I will never forget how he teach me to do research and how to always “keep it simple and stupid” when dealing with difficulties. This gives me a strong believe that doing research must be a joyful and happy work.

I am thankful to Professors for being on my defense and reading committees, for many helpful discussions. I thank for providing much valuable feedback. I would also like to thank Professor Kwang Bok Lee for acting as the chair of my defense committee, and Professor Insoon Yang, Professor Jun-Won Choi, and Professor Daeyoung Park for being members of the committee and for Profs’ thought-provoking questions and comments on my work.

It has been a pleasure to be a part of Information System Laboratory (ISL), where I have made many great friends. I would like to especially acknowledge ISL members for their collaborations on my work and co-authorship of several papers, and also for their friendship and constant support. My heartfelt acknowledgement go to my teachers in all stages of my education, the individuals who have inspired me to learn and grow. I am grateful to all friends who have supported me through the years and have made my time at Seoul National University so enjoyable.

My deepest gratitude and love belong to my parents, my sister, and my brothers, for their unconditional love and support all though my life. To them I owe all that I am and all that I have ever accomplished, and it is to them that I dedicate this thesis.