Robust Low-Rank Approximation of Matrices in ℓ_p-Space

Vom Fachbereich 18
Elektrotechnik und Informationstechnik
der Technischen Universität Darmstadt
zur Erlangung der Würde eines
Doktor-Ingenieurs (Dr.-Ing.)
genehmigte Dissertation

von
Wenjun Zeng, M.Sc.
geboren am 08.12.1982 in Jiangxi, China

Referent:	Prof. Dr.-Ing. Abdelhak M. Zoubir
Korreferent:	Prof. Dr. Hing Cheung So
Tag der Einreichung:	11.04.2018
Tag der mündlichen Prüfung:	05.07.2018

D 17
Darmstadt, 2018

Zeng, Wenjun: Robust Low-Rank Approximation of Matrices in lp-Space
Darmstadt, Technische Universität Darmstadt
Jahr der Veröffentlichung der Dissertation auf TUprints: 2018
Tag der mündlichen Prüfung: 05.07.2018

Veröffentlicht unter CC BY-SA 4.0 International
https://creativecommons.org/licenses/

Acknowledgments

In 2012, while I was a research associate at City University of Hong Kong, I read a paper "Robust estimation in signal processing: A tutorial-style treatment of fundamental concepts" published in *IEEE Signal Processing Magazine*, whose first author is Prof. Abdelhak M. Zoubir. It is this paper that brought me to the field of robust signal processing. I would never have imagined that several years later I will have the great fortune of having Prof. Zoubir as my doctoral advisor. I would like to express my heartfelt thanks to Prof. Zoubir, who is a true gentleman. His considerate guidance, careful nurturing, constant encouragement, and generous support led me to complete my doctoral dissertation. Without his advice and support, my academic career would not have been the same. I am greatly indebted to him.

I am grateful to my co-advisor, Prof. Hing Cheung So of City University of Hong Kong. These years I worked as a senior research associate with him. He guides me to do many exciting research topics including robust estimation, sparse/low-rank recovery, greedy algorithms and phase retrieval. He is like my senior brother to take care of me in life. I cherish the good times with him in Hong Kong.

I would like to thank Prof. Jürgen Adamy, Prof. Udo Schwalke, and Prof. Florian Steinke for serving on my Ph.D. oral examination committee. I also thank Prof. Marius Pesavento for his encouragements.

Thanks to Prof. Jieping Ye of University of Michigan, Ann Arbor, for his valuable discussion, guidance and encouragement. Appreciations go to Prof. Anthony Man-Cho So of the Chinese University of Hong Kong. I attended two courses on optimization lectured by him, which greatly consolidated my mathematical foundations. Thanks also to Prof. Emmanuel Candes and Dr. Ju Sun of Stanford University for discussing the nonconvex optimization.

Many thanks to all members of our great Signal Processing Group of Technische Universität Darmstadt: Dr. Michael Fauß, Di Jin, Mark Ryan Leonard, Dr. Michael Muma, Afief Dias Pambudi, Dominik Reinhard, Tim Schäck, Ann-Kathrin Seifert, Adrian Šošić, Sergey Sukhanov, and Freweyni Kidane Teklehaymanot. I appreciate our secretary, Renate Koschella, and system administrator, Hauke Fath for their supports. Special thanks to Adrian Šošić and Tim Schäck, for their careful assistance when I prepared the doctoral thesis. Gratitude also to the group members at City University of Hong Kong: Wai Ming Chan, Liping Guo, Qi Liu, and Dr. Jingwei Xu. I deeply appreciate my good friend, Prof. Jiayi Chen of Shenzhen University, who generously helped me in my difficult times.

Going back to my time at Tsinghua University, I wish to thank my master advisors Prof. Xian-Da Zhang and Prof. Hongxing Zou, who introduced the broad topic of signal processing to me. I also would like to thank Prof. Changshui Zhang and Dr. Xi-Lin Li for teaching me many topics in blind source separation, computer vision, and machine learning. They were instrumental in fostering my interest in research. When I was a faculty member at Xiamen University from 2009 to 2012, Prof. En Cheng, the dean of the department of communications engineering, provided me much assistance. Thanks to Prof. Cheng.

Being raised in a traditional Chinese family, it is difficult to express love and affection verbally to my dear parents, Bingsheng Zeng and Fanxiu Yang. They have worked very hard to provide me with my education. Without their unconditional love, I could not grow to become the man that I am today. I hope to make them proud. In addition, I would like to sincerely thank my younger brother, Zihui Zeng, for his support and for taking care of my parents when I pursue my professional goal.

I owe my deepest gratitude to my wife Xue Jiang for her love, encouragement, and unwavering support. Also, I am indebted to Xue's parents, Jianping Jiang and Lu Zhang. They have been treating me like their own son. This thesis is dedicated to them all.

Darmstadt, 08.07.2018

Kurzfassung

Low-Rank Approximationen spielen eine bedeutende Rolle in vielen Bereichen der Wissenschaft und Technik, wie zum Beispiel in der Signal- und Bildverarbeitung, im maschinellen Lernen und Data Mining, sowie in der Bildgebung, der Bioinformatik, der Musterklassifizierung und im Bereich Computer Vision. Grund hierfür ist, dass viele Daten, die realen Anwendungsgebieten entstammen, von Natur aus niederrangig sind.

Ziel dieser Dissertation ist die Entwicklung neuer Algorithmen zur robusten Low-Rank Approximation einzelner und mehrerer Matrizen in Gegenwart von Ausreißern—einem Problem bei dem konventionelle Techniken der Dimensionalitätsreduktion wie beispielsweise die Hauptkomponentenanalyse (engl. *principial component analysis*, PCA) häufig versagen. Die in dieser Arbeit vorgestellte Methodik basiert auf einer Residuen-Minimierung unter Verwendung der l_p-Norm, einschließlich des nicht-konvexen und nicht-stetigen Falles von $p < 1$. Im Zentrum der Arbeit stehen sowohl die theoretische Analyse des Problems als auch dessen praktische Anwendung. Die gewonnen experimentellen Erkenntnisse zeigen die Überlegenheit der vorgestellten Methodik gegenüber aktuellen Vergleichsverfahren.

Zunächst werden zwei iterative Algorithmen zur Low-Rank Approximation einer einzelnen Matrix konzipiert. Die sogenannte *iteratively reweighted singular value decomposition* (IR-SVD) Methode basiert auf einer Matrix-Singulärwertzerlegung, bei der die zugrundeliegende Matrix in jeder Iteration des Verfahrens neu gewichtet wird. In der zweiten Methode wird das nicht-konvexe l_p-Matrixfaktorisierungsproblem durch eine Reihe einfacherer l_p-Minimierungsprobleme ersetzt, wobei die auftretenden Vektoren als eigenständige Variablen betrachtet werden. Zu beiden Verfahren werden Anwendungsbeispiele aus verschiedenen Bereichen diskutiert, darunter die Separation von Bildkomponenten, die Vordergrunddetektion in der Videoüberwachung, Beispiele aus dem Bereich Array-Signalverarbeitung, sowie die Richtungsschätzung zur Quellenlokalisierung in impulsivem Rauschen.

Anschließend wird die Low-Rank Approximation mit fehlenden Werten (engl. *robust matrix completion*) behandelt, für welche ebenfalls zwei Verfahren vorgestellt werden. Das erste Verfahren bietet einen iterativen Lösungsansatz, welcher auf der Berechnung linearer l_p-Regressionsproblemen basiert. Das zweite Verfahren beruht auf der sogenannten *alternating direction method of multipliers* (ADMM) im l_p-Raum. Bei jeder Iteration von ADMM wird eine Matrixfaktorisierung im Sinne der kleinsten Fehlerquadrate (engl. *least squares*, LS) durchgeführt, wofür ein Näherungsoperator basierend auf der p-ten Potenz der l_p-Norm berechnet wird. Die LS-Faktorisierung wird effizient durch lineare Regression gelöst, wobei der Nährerungsoperator über die Nullstellen einer skalaren nichtlinearen Funktion berechnet wird. Beide Algorithmen sind in der Problemgröße skalierbar. Die Verfahren werden anhand von Beispielen zur kollaborativen Filterung, der automatischen Bildvervollständigung, sowie anhand einer Anwendung im Bereich Empfehlungssysteme demonstriert.

Für die robuste Low-Rank Approximation mehrerer Matrizen (RLRAMM) mit Ausreißern werden l_p*-greedy pursuit* (l_p-GP) Algorithmen konzipiert. Das l_p-GP Verfahren mit $0 < p < 2$ löst die RLRAMM Problematik, indem das Kernproblem in eine Reihe von Rang-Eins Approximationsproblemen zerlegt wird. Bei jeder Iteration des Verfahrens wird die beste Rang-Eins Approximation durch Minimierung der l_p-Norm des Residuums gefunden, woraufhin die Rang-Eins Basismatrizen vom Residuum subtrahiert werden. Anschließend wird ein Minimierungsansatz zur l_p-Rang-Eins Berechnung vorgestellt. Da der Sonderfall $p = 1$ nur die Berechnung gewichteter Mediane erfordert, ist die Komplexität des Verfahrens beinahe linear in der Anzahl und Dimension der Matrizen, wodurch l_1-GP nahezu skalierbar für große Probleme wird. Die Konvergenz von l_p-GP wird formal nachgewiesen, wobei gezeigt wird, dass die Summe der l_p-Normen der Residuen exponentiell abklingt. Hierbei wird ein Zusammenhang zwischen der Konvergenzrate im ungünstigsten Fall und der l_p-Korrelation zwischen den Residuen und der aktuellen Lösung hergestellt. Des Weiteren wird gezeigt, dass l_p-GP ein höheres Kompressionsverhältnis gegenüber bisherigen Methoden aufweist. Für den Spezialfall $p = 2$ wird die *orthogonal greedy pursuit* (OGP) Methode weiterentwickelt, um deren Konvergenz zu beschleunigen. Gleichzeitig wird der Berechnungsaufwand der erforderlichen Neugewichtung durch ein rekursives Updateverfahren reduziert. Abschließend werden festere und genauere Grenzen der Konvergenzraten für den Fall $p = 2$ abgeleitet und Anwendungen zur Datenkompression, zur robusten Bildrekonstruktion und zur Bildverarbeitung diskutiert.

Abstract

Low-rank approximation plays an important role in many areas of science and engineering such as signal/image processing, machine learning, data mining, imaging, bioinformatics, pattern classification and computer vision because many real-world data exhibit low-rank property. This dissertation devises advanced algorithms for robust low-rank approximation of a single matrix as well as multiple matrices in the presence of outliers, where the conventional dimensionality reduction techniques such as the celebrated principal component analysis (PCA) are not applicable. The proposed methodology is based on minimizing the entry-wise ℓ_p-norm of the residual including the challenging nonconvex and nonsmooth case of $p < 1$. Theoretical analyses are also presented. Extensive practical applications are discussed. Experimental results demonstrate that the superiority of the proposed methods over the state-of-the-art techniques.

Two iterative algorithms are designed for low-rank approximation of a single matrix. The first is the iteratively reweighted singular value decomposition (IR-SVD), where the SVD of a reweighted matrix is performed at each iteration. The second converts the nonconvex ℓ_p-matrix factorization into a series of easily solvable ℓ_p-norm minimization with vectors being variables. Applications to image demixing, foreground detection in video surveillance, array signal processing, and direction-of-arrival estimation for source localization in impulsive noise are investigated.

The low-rank approximation with missing values, i.e., robust matrix completion, is also addressed. Two algorithms are developed for it. The first iteratively solves a set of linear ℓ_p-regression problems while the second applies the alternating direction method of multipliers (ADMM) in the ℓ_p-space. At each iteration of the ADMM, it requires performing a least squares (LS) matrix factorization and calculating the proximity operator of the pth power of the ℓ_p-norm. The LS factorization is efficiently solved using linear LS regression while the proximity operator is obtained by root finding of a scalar nonlinear equation. The two proposed algorithms are scalable to the problem size. Applications to recommender systems, collaborative filtering, and image inpainting are provided.

The ℓ_p-greedy pursuit (ℓ_p-GP) algorithms are devised for joint robust low-rank approximation of multiple matrices (RLRAMM) with outliers. The ℓ_p-GP with $0 < p < 2$ solves the RLRAMM by decomposing it into a series of rank-one approximations. At each iteration, it finds the best rank-one approximation by minimizing the ℓ_p-norm of the residual and then, the rank-one basis matrices are subtracted from the residual. A successive minimization approach is designed for the ℓ_p-rank-one fitting. Only weighted medians are required to compute for solving the most attractive case with $p = 1$, yielding that the complexity is near-linear with the number and dimension of the matrices. Thus, the ℓ_1-GP is near-scalable to large-scale problems. The convergence of the ℓ_p-GP is theoretically proved. In particular, the sum of the ℓ_p-norms of the residuals decays exponentially. We reveal that the worst-case bound of the convergence rate is related to the ℓ_p-correlation of the residual and the current solution. The ℓ_p-GP has a higher compression ratio than the existing methods. For the special case of $p = 2$, the orthogonal greedy pursuit (OGP) is further developed to accelerate the convergence, where the cost of weight re-computation is reduced by a recursive update manner. Tighter and more accurate bounds of the convergence rates are theoretically derived for $p = 2$. Applications to data compression, robust image reconstruction and computer vision are provided.

Contents

1 Introduction 1

1.1 Background . . . 1

1.2 Motivation . . . 3

1.3 State-of-the-Art . . . 5

1.3.1 Overview of Robust Single Matrix Approximation . . . 5

1.3.2 Overview of Matrix Completion . . . 5

1.3.3 Overview of Multiple Matrix Approximation . . . 7

1.4 Goal and Contributions . . . 8

1.5 Organization . . . 10

1.6 Publications . . . 11

2 Robust Low-Rank Approximation of A Single Matrix 13

2.1 Least ℓ_p-Norm Criterion for Low-Rank Factorization . . . 13

2.2 Iteratively Reweighted SVD Algorithm . . . 14

2.3 Alternating Minimization Algorithm . . . 16

2.3.1 IRLS for ℓ_p-Fitting . . . 17

2.3.2 Complex-Valued Newton's Methods for ℓ_p-Fitting . . . 19

2.4 ADMM for ℓ_1-Low-Rank Factorization . . . 24

2.4.1 Principles of ADMM . . . 24

2.4.2 Summary of ADMM . . . 26

2.5 Differences with Other Robust PCA Methods . . . 28

2.6 Application to Array Processing . . . 29

2.6.1 Overview on DOA Estimation . . . 29

2.6.2 Signal Model and ℓ_p-MUSIC . . . 30

2.7 Experimental Results . . . 32

2.7.1 Impulsive Noise Model . . . 32

2.7.2 Convergence Behavior and Running Time . . . 33

2.7.3 Results of DOA Estimation . . . 36

2.7.4 Results of Image Demixing . . . 38

2.7.5 Results of Foreground Detection for Video Surveillance . . . 40

2.8 Summary . . . 42

3 Robust Low-Rank Matrix Completion in ℓ_p-Space 45

3.1 Problem Formulation and Preliminaries . . . 45

3.2 Iterative ℓ_p-Regression Algorithm . . . 47

3.3 ADMM for Robust Matrix Completion . . . 51

3.3.1 Framework of ADMM . . . 51

3.3.2 Proximity Operator of pth Power of ℓ_p-Norm 53
3.3.3 Summary of ADMM 55
3.3.4 Algorithmic Parameter Selection 57
3.4 Experimental Results 59
3.4.1 Results of Synthetic Random Data 59
3.4.2 Image Inpainting in Salt-and-Pepper Noise 69
3.4.3 Results of Recommender Systems 71
3.5 Summary 74

4 Greedy Pursuit for Approximation of Multiple Matrices in ℓ_2-Space 77
4.1 Introduction 78
4.2 Problem Formulation and Preliminaries 79
4.2.1 Problem Formulation 79
4.2.2 Prior Arts on Low-Rank Representation 81
4.3 Greedy Pursuit Algorithms 82
4.3.1 Greedy Pursuit 82
4.3.2 Solution to Rank-One Fitting 83
4.3.3 Economic Greedy Pursuit 86
4.3.4 Orthogonal Greedy Pursuit 87
4.3.5 Feature Extraction for Pattern Classification 89
4.4 Convergence Analysis 91
4.4.1 Key Lemma 91
4.4.2 Convergence Analysis for GP and EGP 92
4.4.3 Convergence Analysis for OGP 95
4.5 Experimental Results 99
4.5.1 Convergence Behaviors 100
4.5.2 Results of Image Reconstruction 101
4.5.3 Results of Face Recognition 105
4.6 Summary 107

5 ℓ_p-Greedy Pursuit for Robust Approximation of Multiple Matrices 109
5.1 Introduction 109
5.2 Problem Formulation and Preliminaries 110
5.2.1 Problem Formulation 110
5.2.2 RPCA for Multiple Matrix Approximation 112
5.3 Greedy Pursuit Algorithms in ℓ_p-Space 113
5.3.1 Greedy Pursuit in ℓ_p-Space 113
5.3.2 Solution to ℓ_p-Rank-One Fitting 113
5.3.3 Selection of p 117
5.4 Convergence Analysis 119

5.4.1 ℓ_p-Correlation 119
5.4.2 Descent Lemma 120
5.4.3 Convergence of ℓ_p-GP 122
5.5 Experimental Results 123
5.5.1 Convergence Behaviors 124
5.5.2 Results of Robust Image Reconstruction 125
5.6 Summary 131

6 Conclusions and Future Work **133**
6.1 Conclusions 133
6.2 Future Work 135

Appendix **137**
A.1 Polynomial-Time Algorithm for Univariate ℓ_p-Regression 137

List of Acronyms **139**

List of Symbols **142**

References **143**

Curriculum Vitae **155**

Chapter 1

Introduction

1.1 Background

The amount and dimension of data that are being generated, collected, stored, and processed have been increasing explosively in recent years. We have already entered the era of "big data" [1, 2]. High-dimensional data such as high resolution images and videos, gene expression data from a DNA microarray, and social network data, are ubiquitous. For example, as shown in Figure 1.1, more than 100-hour videos are being uploaded to YouTube every minute.

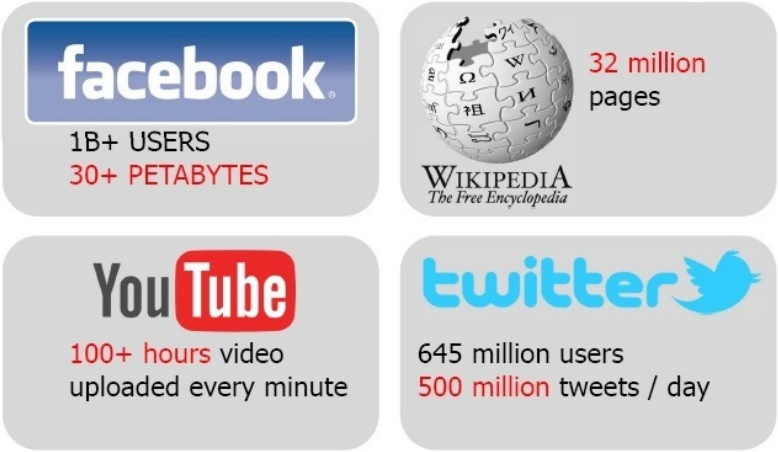
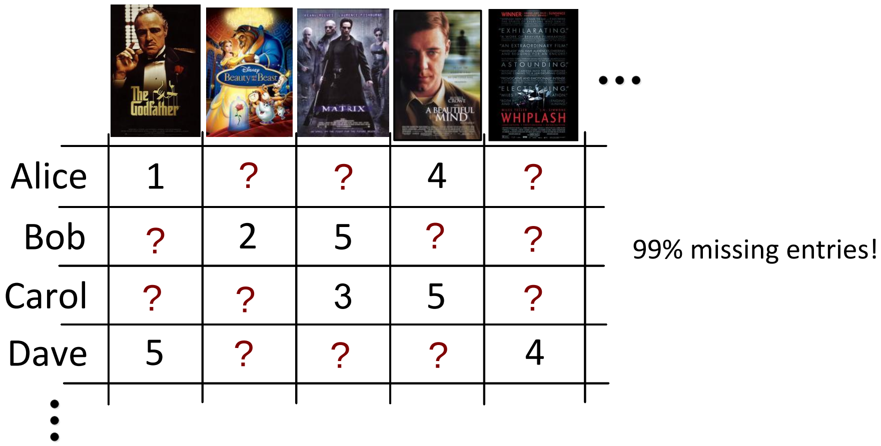


Figure 1.1: Explosively increasing amount and dimension of data nowadays.

However, directly dealing with them are unrealistic due to the *curse of dimensionality*. Fortunately, many high-dimensional data exhibit specific low-dimensional structure, in which most useful information is hidden. As a result, a core problem in data science is

"How to effectively and efficiently learn the low-dimensional representation of high-dimensional data?"

It is evident that in many cases the data naturally have or can be organized in the form of matrices which often exhibit low-rank property [3]. Therefore, it is of great interest



Alice	1	?	?	4	?
Bob	?	2	5	?	?
Carol	?	?	3	5	?
Dave	5	?	?	?	4

Figure 1.2: Netfix Prize as an example of recommender systems. The ratings take values from $1, 2, \cdots, 5$, where 5 stands for the highest rating while 1 is the lowest. The question mark denotes unknown entries.

in taking advantage of low-rankness in the matrices in order to achieve *dimensionality reduction* and extract the desired information. The low-rank property is also referred to *rank-sparsity* [4, 5] because the vector composing of the singular values of a low-rank matrix is sparse [5]. A representative example exploiting the low-rankness for information extraction and inference is the collaborative filtering in Netflix Prize [6–8], shown as in Figure 1.2. In recommender systems [9], only a fraction of movie ratings can be observed from a large data matrix in which rows are users and columns are movies because each user typically rates a few movies rather than all movies. The database of the Netflix Prize has over 100 million movie ratings made by 480,189 users in 17,770 films, which corresponds to recovery of a huge matrix with around 99% missing entries. Collaborative filtering is the task of making automatic predictions about the ratings of a user by collecting preference information from many users [10]. Generally, recovering a data matrix from a subset of its entries is impossible. However, if the unknown matrix is of low-rank or approximately low-rank, then accurate prediction is possible. This type of low-rank matrix approximation with missing entries is referred to as *matrix completion* [9, 11, 12], which will be discussed in Chapter 3.

Low-rank approximation refers to approximating one matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ by another of lower rank, say, $\widehat{\boldsymbol{A}}$. Based on the minimum square loss criterion, low-rank approximation is mathematically formulated as

$$\begin{aligned} &\min_{\widehat{\boldsymbol{A}}} \|\widehat{\boldsymbol{A}} - \boldsymbol{A}\|_{\mathrm{F}}^2 \\ &\text{s.t. } \operatorname{rank}(\widehat{\boldsymbol{A}}) = r \end{aligned} \tag{1.1}$$

where the target rank $r \leq \min(m, n)$. Using the decomposition $\widehat{\boldsymbol{A}} = \boldsymbol{U}\boldsymbol{V}^{\mathrm{T}}$ where

$\boldsymbol{U} \in \mathbb{R}^{m \times r}$ and $\boldsymbol{V} \in \mathbb{R}^{n \times r}$, the low-rank constraint in (1.1) is automatically fulfilled and (1.1) is equivalent to the following *low-rank factorization*

$$\min_{\boldsymbol{U},\boldsymbol{V}} \|\boldsymbol{U}\boldsymbol{V}^{\mathrm{T}} - \boldsymbol{A}\|_{\mathrm{F}}^2. \tag{1.2}$$

The columns of $\boldsymbol{U}$ and $\boldsymbol{V}$ span the r-dimensional subspaces of the column and row spaces of $\boldsymbol{A}$, respectively. Therefore, the low-rank approximation also achieves the task of *subspace learning* [13,14]. Low-rank approximation has a wide range of applications, including signal/image processing [15], machine learning [16,17], computer vision [18,19], data mining [20], pattern classification [21], medical imaging [22], bioinformatics [23,24], and social networks [25]. Also, it is closely related to dimensionality reduction and subspace learning since the subspace with a lower dimensionality can be calculated via low-rank factorization. By Eckart-Young Theorem [26], the global minimizer of (1.1) is given by the truncated singular value decomposition (SVD) of $\boldsymbol{A}$, which is expressed as

$$\widehat{\boldsymbol{A}} = \sum_{i=1}^{r} \sigma_i(\boldsymbol{A}) \boldsymbol{y}_i \boldsymbol{z}_i^{\mathrm{T}} \tag{1.3}$$

where $\sigma_i(\boldsymbol{A})$ is the ith singular value of $\boldsymbol{A}$ while $\boldsymbol{y}_i \in \mathbb{R}^m$ and $\boldsymbol{z}_i \in \mathbb{R}^n$ are the corresponding left and right singular vectors, respectively. The largest r singular values and the corresponding singular vectors $\{\sigma_i(\boldsymbol{A}), \boldsymbol{y}_i, \boldsymbol{z}_i\}_{i=1}^r$ are called “principal components”. Thus, low-rank approximation under Frobenius norm minimization amounts to the celebrated *principal component analysis (PCA)* [27,28]. The PCA aims to find a subspace with a given dimension (rank) that best preserves the energy in the reduced space. As a fundamental tool for dimensionality reduction and data compression, the PCA has very wide applicability. For example, the eigenface method for face recognition is based on PCA [21].

1.2 Motivation

Although the PCA is extremely useful in various scenes, it has the following three drawbacks.

1. The first drawback is that it is not robust to outliers[1] or impulsive noise. It is known that the PCA fails in properly capturing the low-rank structure when the observations contain outliers, as illustrated in Figure 1.3. However, the occurrence of outliers has been reported in many fields [29–31]. For example, the

---

[1] In this thesis, outliers refer to outlying entries whose values are abnormally large, and they are often sparse corruptions in the observed entries.

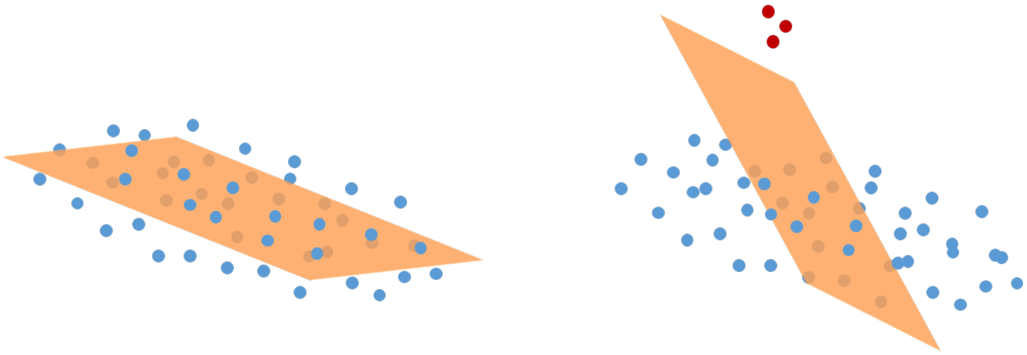


Figure 1.3: Sensitivity of the classic PCA to outliers. Blue points stand for "normal" data while red points are outliers. In the figure on the left where there is no outlier, the PCA successfully captures the dominate low-rank structure, i.e., the two-dimensional yellow plane, which most points lie on. However, in the presence of outliers, as show in the figure on the right, the plane given by the PCA significantly deviates from the true one.

salt-and-pepper noise is a common impulsive noise type in image processing [31]. Another example that the outlier plays an important role is the foreground and background separation in video surveillance, where the foreground can be well modeled by sparse outliers [32–34].

2. The second drawback is that the PCA based on the SVD is not applicable to the low-rank matrix completion where there are missing entries. In other words, it is not robust to missing values.

3. The third drawback is that the PCA can only handle a single matrix, where each of its columns corresponds to a data point. Thus, it needs to first convert the data into vectors to apply the PCA if the data points are not in the form of vectors. For two-dimensional (2D) array data such as images and frames of video, which are represented by matrices, the "vectorization" yields a long vector and then a matrix with very large size. This results in an expensive computational cost because the complexity of the SVD is cubic with respect to the matrix dimension [35]. Moreover, the vectorization breaks the 2D structure and the innate relation between row and column [16, 36].

We are motivated to explore advanced techniques for low-rank matrix approximation to overcome these drawbacks. The new schemes have enhanced performance over the conventional methods.

1.3 State-of-the-Art

1.3.1 Overview of Robust Single Matrix Approximation

It is worth pointing out that a variety of approaches aiming to enhance the robustness of the PCA have been developed in recent years. The celebrated robust principal component analysis (RPCA) [32,33] models the data matrix as the superposition of a low-rank component and a sparse component that represents the outliers. The RPCA minimizes the nuclear norm of the unknown low-rank matrix plus the ℓ_1-norm of the outlier component to separate the low-rank structure from the sparse outliers. At the earliest, this minimization problem is converted into a semi-definite programming (SDP) whose complexity is very high. A faster scheme based on the augmented Lagrange method (ALM) is developed to reduce the complexity [37]. The ALM needs to calculate the proximity operators of the nuclear norm and ℓ_1-norm at each iteration [37]. Thus, the full SVD is required, making the computational cost of the ALM still expensive. The nonconvex RPCA [38] replaces the nuclear norm and ℓ_1-norm with the rank constraint and ℓ_0-norm, respectively. Its performance guarantee is theoretically analyzed in [38]. The nonconvex RPCA just needs the truncated SVD rather than the full SVD. As a result, when the target rank is much smaller the matrix dimension, its computational cost is evidently cheaper than the convex version.

1.3.2 Overview of Matrix Completion

Matrix completion, i.e., low-rank matrix recovery with missing entries, is a very hot research topic in recent years due to its importance and wide applications in information retrieval and inference [9,11,12,17,31]. It can be formulated as a constrained rank minimization problem [9]. Unfortunately, this problem is NP-hard in general because the rank is discrete and nonconvex. Analogous to the strategy of employing the ℓ_1-norm instead of the ℓ_0-norm for sparse signal recovery [39, 40, 87], convex relaxation for rank minimization replaces the nonconvex rank by the convex nuclear norm, which is the sum of all singular values of the matrix [9, 42], and its theoretical guarantees have been provided in [11]. Typically, nuclear norm minimization is converted into an SDP [9, 42, 43] and hence can be solved by the interior-point methods [43, 44]. However, directly realizing the SDP leads to a high computational load. On the other hand, algorithms which are more computationally efficient than the SDP-based methods have been suggested, such as singular value thresholding (SVT) [45], fixed point continuation (FPC) [46], and proximal gradient descent [47]. Nevertheless, these faster schemes still require performing full SVD of the matrix at each iteration, implying the high complexity to deal with a large matrix. Using the Schatten p-quasi-norm with $0 < p < 1$, namely, ℓ_p-norm of the singular values instead of the nuclear norm can further improve the recovery performance [48–52]. Note that the Schatten p-quasi-norm minimization also involves the time-consuming full SVD calculation, which constitutes its dominant computational cost. As a modification to the standard nuclear norm minimization treating each singular value equally, the weighted nuclear norm minimization (WNNM) method [53] adaptively assigns weights to different singular values to enhance the rank sparsity. However, the WNNM for matrix completion [53] is designed for the noiseless case and hence, is not robust to outliers.

In sparse signal recovery, the ℓ_1-norm minimization can be solved by iterative soft thresholding (IST) [87]. The SVT [45] in fact iteratively applies thresholding and shrinkage to the singular values to achieve "rank sparsity". Different from the IST, the iterative hard thresholding (IHT) [54] for sparse recovery constrains that the number of nonzero elements does not exceed a specific value to obtain a sparse result. Borrowing the idea from IHT, a class of approaches, including the singular value projection (SVP) [55], normalized IHT [56], and alternating projection (AP) [57], directly exploits a rank constraint to ensure a low-rank solution. In particular, the SVP and normalized IHT adopt gradient projection method [58] to solve the rank constrained problem. Compared with the nuclear norm or Schatten p-norm minimization that needs to calculate the full SVD, the IHT-type method only requires performing truncated SVD to obtain the r dominant singular values and singular vectors, assuming that the rank information is available. Hence, the computational cost can be greatly reduced especially when the rank is much smaller compared to the matrix dimensions [59].

The third approach for matrix completion utilizes low-rank matrix factorization, where the target matrix is represented by the product of two much smaller matrices so that the low-rank property is automatically fulfilled [60–64]. The gradient descent method can be applied as the solver [60, 61], but it suffers from slow convergence. To speed up the convergence rate, the alternating least squares is employed to tackle the resultant bi-convex problem [61, 63, 64]. It is worth pointing out that one main advantage of the matrix factorization based solutions is that they avoid the SVD.

Conventional techniques for matrix completion often rely on the Gaussian noise assumption and their derivation is based on the ℓ_2-space. In spite of providing theoretical and computational convenience, it is generally understood that the validity of Gaussian distribution is at best approximate in reality. The occurrence of non-Gaussian outliers is also frequently encountered in matrix completion [31, 65–69]. The algorithms based on Frobenius norm minimization severely degrade in the presence of outliers.

Several existing schemes have utilized the fact that the entry-wise ℓ_p-norm with $p < 2$ is less sensitive to outliers than the Frobenius norm for robust matrix factorization [18]. The ℓ_1-Wiberg algorithm [18] that is applicable for the case of incomplete observations exploits ℓ_1-norm to enhance the robustness to outliers, but it has a very high computational complexity. In [48], the ℓ_p-norm and Schatten p-norm are jointly used for robust matrix completion. The ALM is employed to solve the resultant joint ℓ_p-norm and Schatten p-norm minimization, in which the full SVD is required. Thus, the computational cost of the ALM is also high. In [57], matrix completion is formulated as a feasibility problem, where the target matrix lies in the intersection of low-rank constraint set and fidelity constraint set. The AP algorithm is developed to find a common point of the two sets. By modeling the fidelity constraint set as an ℓ_p-ball with the center of the ball being the observed entries, the AP achieves robustness to outliers if $p < 2$ is adopted. However, the AP needs the prior knowledge on the ℓ_p-norm of the noise, which is difficult to obtain in practice. The proximal alternating robust subspace minimization (PARSuMi) algorithm is proposed in [67], which directly exploits rank constraint on the completed matrix and ℓ_0 pseudo-norm constraint to enhance the robustness to sparse outliers. However, the rank and an upper bound of the number of outliers are required in this method. Unlike most approaches based on standard basis, matrix completion with column-sparse outliers in general basis is addressed in [66]. The RPCA [32] that is originally designed for the case with full observations can also be extended to the case with missing entries. Other two state-of-the-art robust matrix completion methods include the hierarchical system performing bootstrapping [68] and variational Bayesian matrix factorization based on L_1-norm (VBMFL_1) [69].

1.3.3 Overview of Multiple Matrix Approximation

The two-dimensional PCA (2DPCA) [36], which is one of the first methods dealing with *multiple matrices* without vectorization, directly transforms the original matrices into ones with lower column number. The optimal transform of the 2DPCA is given by the principal eigenvectors of the covariance matrix of the 2D matrices whose size is much smaller than that of the traditional PCA, resulting in a significant complexity reduction. However, the 2DPCA merely reduces the column size while the row size remains unchanged since it only applies a single-sided transform. It implies that the compression capability is limited. The generalized low rank approximations of matrices (GLRAM) [16] apply a double-sided transform to reduce both row and column sizes, which considerably improves compression capability. Under the same compression ratio, the GLRAM achieves smaller reconstruction error than the 2DPCA. The 2D-SVD [70] uses similar idea to the GLRAM. The 2DPCA, GLRAM and 2D-SVD belong to orthogonal transform. That is, the columns of the resultant subspaces are orthogonal. Note that unlike SVD, the GLRAM and 2D-SVD do not achieve diagonal decomposition. In [71], the two-dimensional linear discriminant analysis (2DLDA) extends the conventional Fisher linear discriminant analysis [72] by directly handling multiple 2D data points.

A set of matrices can be viewed as a third-order tensor [73]. With the use of existing low-rank tensor decomposition techniques [73–75] such as higher-order SVD (HOSVD) [73], one can also obtain a low-rank approximation of the multiple matrices. However, our methodology in this thesis is different from the *low-rank tensor approximation* in two aspects. First, the factorized results of tensor decomposition are tensors with smaller sizes while those of our method are matrices. Second, our method is computationally more efficient and conceptually simpler than tensor decomposition.

1.4 Goal and Contributions

The goal of this dissertation is to devise new algorithms for several types of low-rank matrix approximation robust against outliers and/or missing values and develop corresponding convergence theories of the proposed algorithms. We list our contributions in four topics as follows.

1. *Robust low-rank approximation of a single matrix.* Three iterative algorithms are designed for this task. The first is the iteratively reweighted singular value decomposition (IR-SVD), where the SVD of a reweighted matrix is performed at each iteration. The second converts the nonconvex ℓ_p-matrix factorization into a series of easily solvable ℓ_p-norm minimization with vectors being variables. The third is alternating direction method of multipliers (ADMM) in ℓ_p-space. Moreover, the pseudo and full Newton's methods with quadratic convergence rate which are applicable to complex-valued variables are developed for the ℓ_p-norm minimization.
2. *Robust low-rank approximation of a single matrix with missing entries.* Two computationally attractive algorithms, namely, iterative ℓ_p-regression algorithm and ADMM, for outlier-robust matrix completion under ℓ_p-minimization. The complexity of the two algorithms is proportional to the number of observed entries and thus, scalable to the matrix dimension.
3. *Low-rank approximation of multiple matrices (LRAMM).* Four contributions in LRAMM (Chapter 4) are:

   i) Three greedy algorithms, namely, greedy pursuit (GP), economic greedy pursuit (EGP) and orthogonal greedy pursuit (OGP) are devised for LRAMM. The proposed algorithms are scalable to the problem size and computationally more efficient than the celebrated SVD since it directly deals with the 2D matrices.
   ii) Compared with other 2D based approaches such as 2DPCA and GLRAM, the greedy algorithms achieve a joint diagonal decomposition for multiple matrices and hence, has a higher compression ratio given the same target rank. In other words, the proposed methods achieve smaller reconstruction errors under the same compression ratio.
   iii) The convergence of the three greedy algorithms is theoretically proved. We show that the reconstruction errors of the three algorithms decay exponentially. The lower bound of the exponential decay factor, i.e., the worst-case convergence rate, is derived.
   iv) The finite convergence property of the OGP is proved. We quantitatively show that how much faster the OGP converges than the GP. The exact expression of the acceleration factor of the OGP over GP, which is dominated by the angle between the current iterate and the subspace spanned by the previous iterates, is derived.

4. *Robust low-rank approximation of multiple matrices (RLRAMM) in ℓ_p-space.* Three contributions in RLRAMM (Chapter 5) are:

   i) An ℓ_p-GP algorithm is devised for RLRAMM. It is near-scalable to the problem size and computationally more efficient than the PCA and RPCA since it directly deals with the 2D matrices. It provides a novel viewpoint for robust low-rank representation and hence, is competitive to the celebrated RPCA.
   ii) Compared with other 2D based approaches such as 2DPCA and GLRAM, the greedy algorithm achieves a joint diagonal decomposition for multiple matrices and hence, has a higher compression ratio given the same target rank. Moreover, the ℓ_p-GP is robust to outliers.
   iii) The convergence of ℓ_p-GP is theoretically proved. We show that the sum of the ℓ_p-norms of the residuals decays exponentially. Furthermore, the worst-case bound of the exponential decay factor or convergence rate is related to the ℓ_p-correlation of the residual and the current iterates.

In addition to the algorithm design and theoretical analysis of convergence, extensive applications to foreground detection in video surveillance, array signal processing, direction-of-arrival estimation for source localization in impulsive noise, image inpainting, recommender systems, data compression, face recognition, and multiple image reconstruction in salt-and-pepper noise, are investigated in this thesis.

1.5 Organization

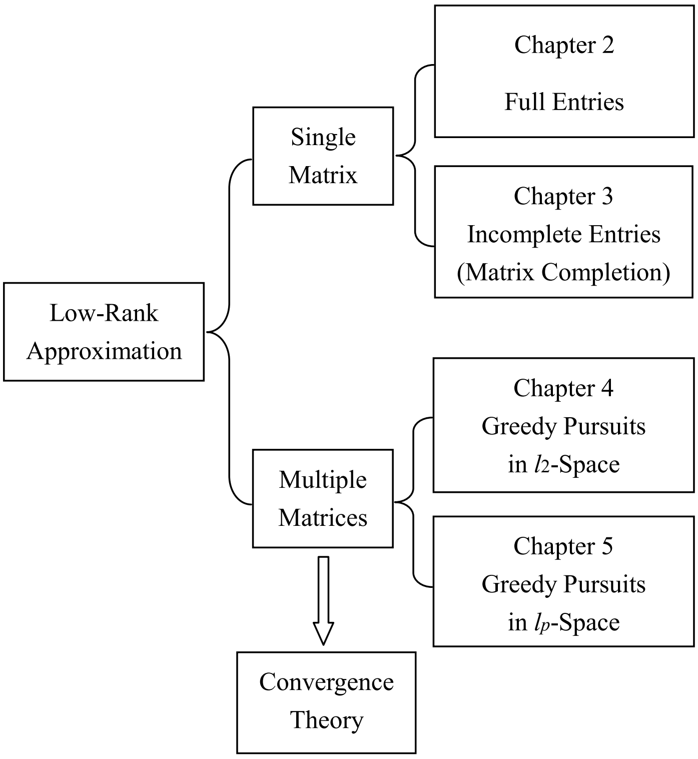

Low-Rank Approximation
Single Matrix
Multiple Matrices
Chapter 2 Full Entries
Chapter 3 Incomplete Entries (Matrix Completion)
Chapter 4 Greedy Pursuits in l_2-Space
Chapter 5 Greedy Pursuits in l_p-Space
Convergence Theory

Figure 1.4: Illustration of the structure of the dissertation.

The remainder of this dissertation is organized as follows. Chapter 2 addresses the robust low-rank approximation of a single matrix with full observations, where three algorithms, IR-SVD, alternating minimization, and ADMM in ℓ_p-space, are proposed. Extending from the case with full observations of Chapter 2 to that with missing values, Chapter 3 discusses the robust low-rank matrix completion with applications in recommender systems and image inpainting. Chapter 4 presents a greedy pursuit algorithmic framework including three variants for low-rank approximation of multiple matrix in ℓ_2-space with theoretical proof of the convergence. Chapter 5 generalizes the greedy algorithms to the ℓ_p-space, making it robust to outliers. Also, new convergence theory of the ℓ_p-GP that is different from that of Chapter 4 is developed in Chapter

5. Concluding remarks, open problems and topics for future research are provided in Chapter 6. Figure 1.4 depicts the structure of the dissertation.

Throughout the thesis, we use bold upper-case and lower-case letters to represent matrices and vectors, respectively. The acronyms and notations are listed in pages 135–137 and 138–139, respectively.

1.6 Publications

The following publications have been produced during the period of doctoral candidacy.

Internationally Refereed Journal Articles

- W.-J. Zeng and H. C. So, "Outlier-robust matrix completion via ℓ_p-minimization," *IEEE Transactions on Signal Processing,* vol. 66, no. 5, pp. 1125–1140, March 2018.
- W.-J. Zeng, H. C. So, J. Chen, and A. M. Zoubir, "Coordinate descent for unit-modulus least squares," submitted to *IEEE Transactions on Signal Processing.*

Internationally Refereed Patents

- W.-J. Zeng, H. C. So, J. Chen, and A. M. Zoubir, "Systems and methods for signal processing using coordinate descent for unit-modulus least squares (UMLS) and unit-modulus quadratic program (UMQP)," U.S. Patent (No. 15/957,657), February 2018.
- W.-J. Zeng, H. C. So, and J. Chen, "Systems and methods for robust low-rank matrix approximation," U.S. Patent (No. 15/676,600), August 2017.
- W.-J. Zeng, H. C. So, J. Chen, and A. M. Zoubir, "Systems and methods for robust low-rank approximation of multiple matrices using greedy pursuit," U.S. Patent, submitted, April, 2018.
- W.-J. Zeng, H. C. So, J. Chen, and A. M. Zoubir, "Systems and methods for low-rank approximation of multiple matrices using greedy pursuit," U.S. Patent, submitted, February, 2018.

Chapter 2

Robust Low-Rank Approximation of A Single Matrix

This chapter focuses on low-rank approximation of a single matrix, where there are no missing entries. As a fundamental tool for this task, the singular value decomposition (SVD) is not robust against outliers. Based on minimizing the entry-wise ℓ_p-norm of the residual instead of the Frobenius norm, this chapter develops three algorithms to achieve outlier-robustness. The first algorithm is the iteratively reweighted singular value decomposition (IR-SVD), where the SVD of a reweighted data matrix is performed in each iteration. The second is the alternating minimization (AM), where the objective function is minimized over one factored matrix while the other factor is fixed. The convergence of the AM is proved. Two complex-valued Newton's methods with optimal step size are devised to solve the resulting ℓ_p-fitting problems. It is revealed that the iteratively reweighted least squares (IRLS) is a special case of the pseudo-Newton's method. The third is the alternating direction method of multipliers (ADMM). The ADMM casts the difficult nonsmooth ℓ_1-subspace decomposition into an ℓ_2-one, which can be efficiently solved via the truncated SVD with a marginal computational increase of soft-thresholding. Experimental results on random data verify the superior performance of the proposed methodology. Wide applicability of the techniques of this chapter is demonstrated by the application examples to DOA estimation, image demixing and video surveillance.

2.1 Least ℓ_p-Norm Criterion for Low-Rank Factorization

In this chapter, complex-valued matrix is considered since application to array processing will be investigated, where the baseband signal is complex. Given a matrix $\boldsymbol{A} \in \mathbb{C}^{m\times n}$, its r-dimension subspace can be computed by the following *low-rank factorization*

$$\min_{\boldsymbol{U},\boldsymbol{V}} \|\boldsymbol{U}\boldsymbol{V} - \boldsymbol{A}\|_{\mathrm{F}}^2 \tag{2.1}$$

where $\boldsymbol{U} \in \mathbb{C}^{m\times r}$ and $\boldsymbol{V} \in \mathbb{C}^{r\times n}$, $\|\boldsymbol{A}\|_{\mathrm{F}} = \left(\sum_{i,j}|\boldsymbol{A}_{ij}|^2\right)^{1/2}$ is the Frobenius norm of $\boldsymbol{A}$ with $\boldsymbol{A}_{ij}$ being its (i,j)th entry and $|\cdot|$ being the absolute value of a real number or the modulus of a complex number. The columns of $\boldsymbol{U}$ and the rows of $\boldsymbol{V}$ span the

r-dimensional subspaces of the column and row spaces of $\boldsymbol{A}$, respectively. Therefore, the low-rank factorization also achieves the task of *subspace learning* [13, 14]. The Frobenius norm minimization (2.1) results in the ML estimates of $\boldsymbol{U}$ and $\boldsymbol{V}$ if the noise is independent and identically distributed (i.i.d.) Gaussian. By Eckart-Young Theorem [26], we can obtain the global minimum of (2.1) via the truncated SVD of $\boldsymbol{A}$:

$$\mathrm{TSVD}_r(\boldsymbol{A}) = \boldsymbol{T}\boldsymbol{\Sigma}\boldsymbol{Q}^{\mathrm{H}} \tag{2.2}$$

where $\boldsymbol{\Sigma} \in \mathbb{R}_{+}^{r\times r}$ is a diagonal matrix whose diagonal elements are the r dominant singular values of $\boldsymbol{A}$, the columns of $\boldsymbol{T} \in \mathbb{C}^{m\times r}$ and $\boldsymbol{Q} \in \mathbb{C}^{n\times r}$ are the corresponding left and right singular vectors, respectively. Hence, we have

$$\boldsymbol{U} = \boldsymbol{T}\boldsymbol{\Sigma}^{1/2},\ \boldsymbol{V} = \boldsymbol{\Sigma}^{1/2}\boldsymbol{Q}^{\mathrm{H}}. \tag{2.3}$$

It has been known that the Frobenius norm is not an outlier-robust cost function since it is based on the square error. Hence, the performance of conventional SVD will degrade when the noise is impulsive.

To make the subspace estimation more robust to outlier, we propose to use the entry-wise ℓ_p-norm of the residual matrix instead of the Frobenius norm in (2.1), expressed as

$$\min_{\boldsymbol{U},\boldsymbol{V}} f_p(\boldsymbol{U},\boldsymbol{V}) := \|\boldsymbol{U}\boldsymbol{V} - \boldsymbol{A}\|_p^p \tag{2.4}$$

where the entry-wise ℓ_p-norm $\|\cdot\|_p$ with $0 < p < 2$ is defined as

$$\|\boldsymbol{A}\|_p = \left(\sum_{i=1}^{m}\sum_{j=1}^{n}|\boldsymbol{A}_{ij}|^p\right)^{1/p}. \tag{2.5}$$

When $p = 2$, (2.4) reduces to the Frobenius norm minimization of (2.1). Minimization of the ℓ_p-norm error function with $p < 2$ is a more suitable criterion in the presence of impulsive noise. Again, the ℓ_p-minimization of (2.4) with respect to $\boldsymbol{U}$ and $\boldsymbol{V}$ is a nonconvex optimization problem and the SVD cannot be applied except for $p = 2$.

2.2 Iteratively Reweighted SVD Algorithm

Denoting the error matrix as $\boldsymbol{E} = \boldsymbol{A} - \boldsymbol{U}\boldsymbol{V}$ with the (i, j)th entry e_{ij}, the entry-wise ℓ_p-norm of (2.4) can be expressed as

$$\begin{aligned} f_p(\boldsymbol{U},\boldsymbol{V}) &= \|\boldsymbol{E}\|_p^p = \sum_{i=1}^{m}\sum_{j=1}^{n}|e_{ij}|^p = \sum_{m,n}|e_{ij}|^{p-2}|e_{ij}|^2 \\ &= \|\boldsymbol{D}\odot\boldsymbol{E}\|_{\mathrm{F}}^2 = \|\boldsymbol{D}\odot\boldsymbol{A} - \boldsymbol{D}\odot(\boldsymbol{U}\boldsymbol{V})\|_{\mathrm{F}}^2 \end{aligned} \tag{2.6}$$

where $\odot$ denotes the element-wise multiplication and $\boldsymbol{D}$ is the weighting matrix with its (i,j)th entry $d_{ij} = |e_{ij}|^{(p-2)/2}$. Equation (2.6) means that the entry-wise ℓ_p-norm minimization problem can be converted into a weighted Frobenius norm minimization one. We can perform SVD to the weighted matrix $\boldsymbol{D}\odot\boldsymbol{A}$ to obtain a better result. Note that the weighting matrix $\boldsymbol{D}$ depends on the unknowns $\boldsymbol{U}$ and $\boldsymbol{V}$. That is, it is a function of $\boldsymbol{U}$ and $\boldsymbol{V}$ which is written as $\boldsymbol{D}(\boldsymbol{U},\boldsymbol{V})$. Due to this reason, we cannot immediately obtain the optimal solution by performing the SVD of the weighted matrix $\boldsymbol{D}\odot\boldsymbol{A}$ only once. An iterative procedure must be employed, which is shown in Algorithm 1, where the superscript $(\cdot)^k$ is used to denote the result at the kth iteration. At each iteration, the SVD of a reweighted matrix is performed. Therefore we refer to this algorithm as iteratively reweighted SVD (IR-SVD). Note that $\epsilon\boldsymbol{I}$ is added to $|\boldsymbol{E}^k|$ to avoid the ill-conditioning. A typical value of ϵ is taken as $\epsilon = 100\epsilon_{\text{machine}}$ with $\epsilon_{\text{machine}}$ being the machine precision.

Algorithm 1 IR-SVD

```
Initialize $\boldsymbol{U}^0$ with a random matrix of full column rank and $\boldsymbol{V}^0$ of full row rank.
for $k = 0, 1, 2, \cdots$ do
    Compute the error matrix $\boldsymbol{E}^k = \boldsymbol{A} - \boldsymbol{U}^k\boldsymbol{V}^k$ and the weighting matrix $\boldsymbol{D}^k = (|\boldsymbol{E}^k| + \epsilon\boldsymbol{I})^{(p-2)/2}$.
    Perform rank-$r$ truncated SVD:
        $\mathrm{TSVD}_r\big(\boldsymbol{D}^k \odot \boldsymbol{A}\big) = \boldsymbol{T}^k\boldsymbol{\Sigma}^k(\boldsymbol{Q}^k)^{\mathrm{H}}$
    Set $\boldsymbol{U}^{k+1} = \boldsymbol{T}^k(\boldsymbol{\Sigma}^k)^{1/2}$ and $\boldsymbol{V}^{k+1} = (\boldsymbol{\Sigma}^k)^{1/2}(\boldsymbol{Q}^k)^{\mathrm{H}}$.
end for
```

It should be pointed out that the IR-SVD algorithm does not always converge when p is much smaller than 2. It rapidly decreases the objective function $f_p(\boldsymbol{U},\boldsymbol{V})$ to a low level at the first few iterations. The IR-SVD procedure will converge toward to this low level value if it converges. This is the general convergence behavior of the IR-SVD algorithm. If it does not converge, it oscillates around this low level value. Although its convergence is not guaranteed, it is a simple and effective approach to achieve a satisfactory low-rank approximation. If it oscillates around a lower level value, this means that the objective function has attained a lower value. Then, the algorithm is terminated and the low-rank approximation can be obtained from the minimum point among all the iterations. The IR-SVD is also applicable to the case with $0 < p < 1$.

2.3 Alternating Minimization Algorithm

Despite the simplicity of the IR-SVD algorithm, it may not converge. In this section, we propose the AM approach, whose convergence is guaranteed, to efficiently solve the nonconvex problem of (2.4). In the AM, the objective function is minimized over one factored matrix while the other factor is fixed. To be more specific, in the $(k+1)$th $(k = 0, 1, \cdots)$ iteration, $\boldsymbol{U}$ and $\boldsymbol{V}$ are alternatingly minimized:

$$\boldsymbol{V}^{k+1} = \arg\min_{\boldsymbol{V}} \|\boldsymbol{U}^k\boldsymbol{V} - \boldsymbol{A}\|_p^p \tag{2.7}$$

$$\boldsymbol{U}^{k+1} = \arg\min_{\boldsymbol{U}} \|\boldsymbol{U}\boldsymbol{V}^{k+1} - \boldsymbol{A}\|_p^p. \tag{2.8}$$

Note that both (2.7) and (2.8) are convex for $1 \leq p \leq 2$ and the global minima of them are guaranteed. For $p < 1$, (2.7) and (2.8) are nonconvex and only stationary points can be guaranteed. The minimization problem of (2.7) with a matrix being variable to be optimized can be decomposed into n independent ℓ_p-fitting subproblems

$$\min_{\boldsymbol{v}_j \in \mathbb{C}^r} \|\boldsymbol{U}^k\boldsymbol{v}_j - \boldsymbol{a}_j\|_p^p, \ j = 1, \cdots, n \tag{2.9}$$

where $\boldsymbol{v}_j$ and $\boldsymbol{a}_j$ are the jth columns of $\boldsymbol{V}$ and $\boldsymbol{A}$, respectively. Two algorithms, namely, the IRLS and complex-valued Newton's method will be presented later.

The convergence of the AM is illustrated in following theorem.

Theorem 1 *The AM algorithm monotonically non-increases the value of the objective function defined in* (2.4)*, thus the sequence* $\{f_p(\boldsymbol{U}^k, \boldsymbol{V}^k)\}$ *converges to a limit point.*

Proof. Denote the residual error value at the kth iteration of the alternating algorithm as $f_p(\boldsymbol{U}^k, \boldsymbol{V}^k)$. From (2.7) and (2.8), it follows that

$$f_p(\boldsymbol{U}^{k+1}, \boldsymbol{V}^{k+1}) \leq f_p(\boldsymbol{U}^k, \boldsymbol{V}^{k+1}) \leq f_p(\boldsymbol{U}^k, \boldsymbol{V}^k). \tag{2.10}$$

This means that $f_p(\boldsymbol{U}, \boldsymbol{V})$ does not increase at each iteration. In addition, $f_p(\boldsymbol{U}, \boldsymbol{V})$ is bounded from below by 0. Therefore, $\{f_p(\boldsymbol{U}^k, \boldsymbol{V}^k)\}$ converges to a limit point. □

The relative reduction of the error value can be used to examine the convergence. Specifically, the convergence is determined by checking whether the following inequality holds:

$$\frac{f_p(\boldsymbol{U}^k, \boldsymbol{V}^k) - f_p(\boldsymbol{U}^{k+1}, \boldsymbol{V}^{k+1})}{f_p(\boldsymbol{U}^k, \boldsymbol{V}^k)} < \varepsilon \tag{2.11}$$

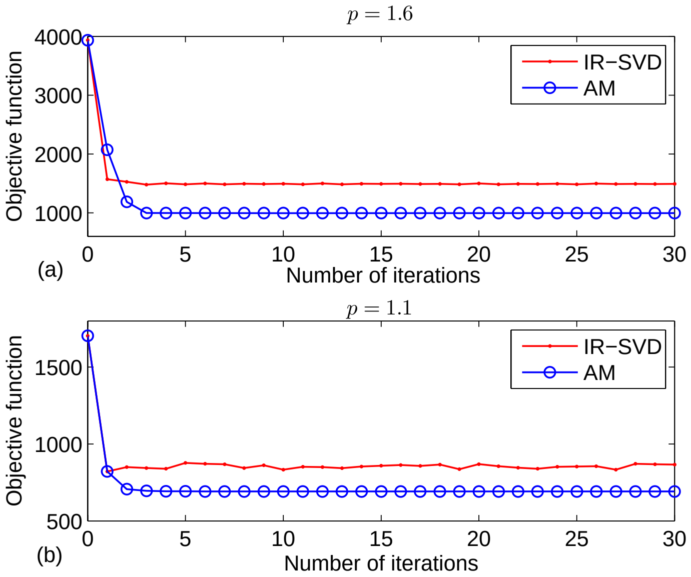


Figure 2.1: Convergence behavior of IR-SVD and AM algorithms. (a) $p = 1.6$; (b) $p = 1.1$.

for a small tolerance $\varepsilon > 0$. In the simulations, we choose $\varepsilon = 10^{-7}$.

Using random data with $m = 50$, $n = 200$, and $r = 20$, Figure 2.5 plots the convergence behaviors of the IR-SVD and AM algorithms. In Figure 2.5 (a), the IR-SVD algorithm converges for $p = 1.6$ while it slightly oscillates for $p = 1.1$ in Figure 2.5 (b). Figure 2.2 further shows the relative error of the AM, which converges in several tens of iterations with the specific tolerance of $\varepsilon = 10^{-7}$.

We see that the AM is superior to the IR-SVD. Not only its convergence is guaranteed but also it more rapidly decreases the objective function value than the latter. It is noticed that the AM does not necessarily converge to the global minimum point. The point that it converges to depends on the initial value. We can initialize $\boldsymbol{U}$ using random matrix of full column rank or the result obtained from the SVD or IR-SVD. Experimental results show that the AM provides a good enough low-rank decomposition although the global optimum is not guaranteed.

2.3.1 IRLS for ℓ_p-Fitting

For $0 < p < 2$, the ℓ_p-fitting of (2.9) can be efficiently solved by the IRLS algorithm [76, 77] where global convergence can be achieved for the convex case of $p \geq 1$ while

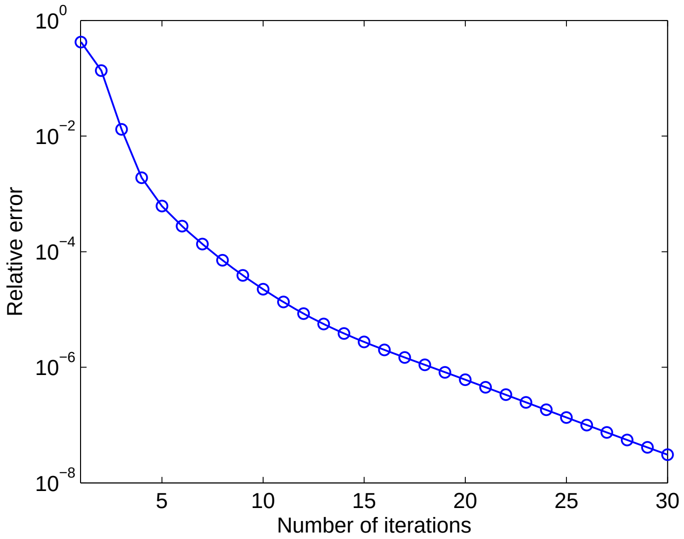


Figure 2.2: Relative error versus number of iterations of AM algorithm. It converges in several tens of iterations with a tolerance of 10^{-7}.

only a stationary point is obtained for the nonconvex case of $p < 1$. At the tth iteration, the IRLS solves the following weighted LS problem:

$$\boldsymbol{v}_j^{t+1} = \arg\min_{\boldsymbol{v}_j} \|\boldsymbol{W}^t(\boldsymbol{U}^k \boldsymbol{v}_j^t - \boldsymbol{a}_j)\|^2 \tag{2.12}$$

where $\boldsymbol{W}^t = \operatorname{diag}\{w_1^t, \cdots, w_m^t\}$ is a diagonal weighting matrix with the ith diagonal element being

$$w_i^t = \frac{1}{(|r_i^t|^2 + \epsilon)^{\frac{1-p/2}{2}}}. \tag{2.13}$$

The r_i^t is the ith element of the residual vector $\boldsymbol{r}^t = \boldsymbol{U}^k \boldsymbol{v}_j^t - \boldsymbol{a}_j$. Like the term $\epsilon \boldsymbol{I}$ in IR-SVD, here $\epsilon > 0$ is a small positive parameter to avoid division by zero and ensure numerical stability, especially for $p \leq 1$. Only one LS problem is required to solve in each IRLS iteration. Therefore, the computational complexity of ℓ_p-fitting is $\mathcal{O}(mr^2 N_{\text{IRLS}})$, where N_{IRLS} is the iteration number required for the IRLS to converge. A typical value of N_{IRLS} is several tens. Then the complexity for solving (2.7) is $\mathcal{O}(mnr^2 N_{\text{IRLS}})$. Since (2.7) and (2.8) have the same structure, (2.7) can be solved by the same way with the same complexity. The total complexity of the AM is $\mathcal{O}(mnr^2 N_{\text{IRLS}} N_{\text{AM}})$, where N_{AM} is the iteration number required for the AM to converge. Generally, a common value of N_{AM} is also several tens.

2.3.2 Complex-Valued Newton's Methods for ℓ_p-Fitting

The IRLS is the most widely used and considered as the standard method for solving the following ℓ_p-fitting problem

$$\min_{\boldsymbol{z}\in\mathbb{C}^r} f(\boldsymbol{z}) := \|\boldsymbol{C}\boldsymbol{z}-\boldsymbol{b}\|_p^p \tag{2.14}$$

where $\boldsymbol{C}\in\mathbb{C}^{m\times r}$ and $\boldsymbol{b}\in\mathbb{C}^m$. However, we observe that the IRLS diverges when p is larger than 3 in the simulations. Although we may not be interested in the setting of $p>2$ in robust estimation, this divergence phenomenon indeed reveals the drawback of the IRLS. For $p>1$, we devise two complex-valued Newton's methods, called pseudo-Newton's and full-Newton's methods with more stable and faster convergence. The two schemes adopt optimal step size at each iteration. Interestingly, it is revealed that the IRLS can be interpreted as a special case of the pseudo-Newton's method using a fixed step size of $p/2$, which is suboptimal.

Gradient and Hessian of ℓ_p-Norm. By defining the residual vector

$$\boldsymbol{r}=\boldsymbol{C}\boldsymbol{z}-\boldsymbol{b}=[r_1,\cdots,r_m]^{\mathrm{T}}, \tag{2.15}$$

the objective function $f(\boldsymbol{z})$ can be expressed as

$$f(\boldsymbol{z})=\|\boldsymbol{r}\|_p^p=\sum_{i=1}^{m}|r_i|^p. \tag{2.16}$$

It is not difficult to derive the partial derivative with respect to the complex quantity r_i as

$$\begin{aligned}\frac{\partial f}{\partial r_i^*}&=\frac{1}{2}\left(\frac{\partial f}{\partial \mathrm{Re}(r_i)}+\imath\frac{\partial f}{\partial \mathrm{Im}(r_i)}\right)\\&=\frac{p}{2}|r_i|^{p-2}r_i,\ i=1,\cdots,m\end{aligned} \tag{2.17}$$

where $\mathrm{Re}(\cdot)$ and $\mathrm{Im}(\cdot)$ are the real and imaginary parts of a complex number, respectively. The gradient of $f(\boldsymbol{r})$ is given by

$$\frac{\partial f}{\partial \boldsymbol{r}^*}=\left[\frac{\partial f}{\partial r_1^*},\cdots,\frac{\partial f}{\partial r_m^*}\right]^{\mathrm{T}}=\frac{p}{2}|\boldsymbol{r}|^{p-2}\odot\boldsymbol{r} \tag{2.18}$$

with $|\boldsymbol{r}|^{p-2}=[|r_1|^{p-2},\cdots,|r_m|^{p-2}]^{\mathrm{T}}$. Define a diagonal matrix

$$\boldsymbol{\Phi}=\mathrm{diag}\{|r_1|^{p-2},\cdots,|r_m|^{p-2}\}. \tag{2.19}$$

Equation (2.18) can be rewritten as

$$\frac{\partial f}{\partial \boldsymbol{r}^*}=\frac{p}{2}\boldsymbol{\Phi}\boldsymbol{r}. \tag{2.20}$$

Note that sometimes we use $\boldsymbol{\Phi}(\boldsymbol{z})$ to emphasize that $\boldsymbol{\Phi}$ is a function of $\boldsymbol{z}$. Clearly, $\boldsymbol{\Phi}(\boldsymbol{z})$ is positive definite. We compute the second-order partial derivative as

$$\frac{\partial^2 f}{\partial r_i^* \partial r_j} = \begin{cases} p^2 |r_i|^{p-2}/4, & \text{if } i = j \\ 0, & \text{if } i \neq j \end{cases}. \tag{2.21}$$

Hence the $m \times m$ *partial Hessian* matrix of f with respect to $\boldsymbol{r}$, denoted by $\boldsymbol{H}_{\boldsymbol{r}^*\boldsymbol{r}}$, is diagonal and has the form:

$$\boldsymbol{H}_{\boldsymbol{r}^*\boldsymbol{r}} = \frac{\partial^2 f}{\partial \boldsymbol{r}^* \partial \boldsymbol{r}^{\mathrm{T}}} = \frac{p^2}{4}\boldsymbol{\Phi} \tag{2.22}$$

The $m \times r$ Jacobian matrix of $\boldsymbol{r}^*(\boldsymbol{z})$ with respect to $\boldsymbol{z}^*$ is given by

$$\frac{\partial \boldsymbol{r}^*}{\partial \boldsymbol{z}^{\mathrm{H}}} = \boldsymbol{C}^*. \tag{2.23}$$

Then the gradient of $f(\boldsymbol{z})$ with respect to $\boldsymbol{z}$ is computed as

$$\boldsymbol{g}(\boldsymbol{z}) = \frac{\partial f}{\partial \boldsymbol{z}^*} = \left(\frac{\partial \boldsymbol{r}^*}{\partial \boldsymbol{z}^{\mathrm{H}}}\right)^{\mathrm{T}} \frac{\partial f}{\partial \boldsymbol{r}^*} = \boldsymbol{C}^{\mathrm{H}} \frac{\partial f}{\partial \boldsymbol{r}^*} = \frac{p}{2}\boldsymbol{C}^{\mathrm{H}}\boldsymbol{\Phi}(\boldsymbol{C}\boldsymbol{z} - \boldsymbol{b}). \tag{2.24}$$

The $r \times r$ leading partial Hessian matrix of f with respect to $\boldsymbol{z}$ is

$$\begin{aligned} \boldsymbol{H}_{\boldsymbol{z}^*\boldsymbol{z}} &= \frac{\partial^2 f}{\partial \boldsymbol{z}^* \partial \boldsymbol{z}^{\mathrm{T}}} = \left(\frac{\partial \boldsymbol{r}^*}{\partial \boldsymbol{z}^{\mathrm{H}}}\right)^{\mathrm{T}} \boldsymbol{H}_{\boldsymbol{r}^*\boldsymbol{r}} \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{z}^{\mathrm{T}}} \\ &= \boldsymbol{C}^{\mathrm{H}} \boldsymbol{H}_{\boldsymbol{r}^*\boldsymbol{r}} \boldsymbol{C} = \frac{p^2}{4} \boldsymbol{C}^{\mathrm{H}} \boldsymbol{\Phi}(\boldsymbol{z}) \boldsymbol{C}. \end{aligned} \tag{2.25}$$

Note that the partial Hessian $\boldsymbol{H}_{\boldsymbol{z}^*\boldsymbol{z}}$ is positive definite because $\boldsymbol{\Phi}(\boldsymbol{z})$ is positive definite. The pseudo-Newton's method only uses the $r \times r$ partial Hessian $\boldsymbol{H}_{\boldsymbol{z}^*\boldsymbol{z}}$ whereas the full-Newton's method exploits the following $2r \times 2r$ full Hessian matrix

$$\boldsymbol{H} = \begin{bmatrix} \boldsymbol{H}_{\boldsymbol{z}^*\boldsymbol{z}} & \boldsymbol{H}_{\boldsymbol{z}^*\boldsymbol{z}^*} \\ \boldsymbol{H}_{\boldsymbol{z}\boldsymbol{z}} & \boldsymbol{H}_{\boldsymbol{z}\boldsymbol{z}^*} \end{bmatrix}. \tag{2.26}$$

The full Hessian matrix is positive definite when $p > 1$. The other three partial Hessian matrices are given by

$$\boldsymbol{H}_{\boldsymbol{z}^*\boldsymbol{z}^*} = \frac{\partial^2 f}{\partial \boldsymbol{z}^* \partial \boldsymbol{z}^{\mathrm{H}}} = \boldsymbol{C}^{\mathrm{H}} \boldsymbol{H}_{\boldsymbol{r}^*\boldsymbol{r}^*} \boldsymbol{C}^* \tag{2.27}$$

where

$$\begin{aligned} \boldsymbol{H}_{\boldsymbol{r}^*\boldsymbol{r}^*} &= \frac{\partial^2 f}{\partial \boldsymbol{r}^* \partial \boldsymbol{r}^{\mathrm{H}}} \\ &= \frac{p(p-2)}{4} \mathrm{diag}\left\{ |r_1|^{p-4} r_1^2, \cdots, |r_m|^{p-4} r_m^2 \right\} \end{aligned} \tag{2.28}$$

is a diagonal matrix, and $\boldsymbol{H}_{\boldsymbol{z}\boldsymbol{z}} = \boldsymbol{H}^*_{\boldsymbol{z}^*\boldsymbol{z}^*}$, $\boldsymbol{H}_{\boldsymbol{z}\boldsymbol{z}^*} = \boldsymbol{H}^*_{\boldsymbol{z}^*\boldsymbol{z}}$. It is noticed that the two off-diagonal block matrices $\boldsymbol{H}^*_{\boldsymbol{z}^*\boldsymbol{z}^*}$ and $\boldsymbol{H}_{\boldsymbol{z}\boldsymbol{z}}$ become zero if $p = 2$. In this case, these two partial Hessian matrices contain no information. When $p \neq 2$, these two matrices do not vanish and contain useful information for optimization.

Newton's Method with Optimal Step Size. The pseudo-Newton's method generates a sequence $\{\boldsymbol{z}^k\}$ $(k = 0, 1, \cdots)$ through the following iteration

$$\boldsymbol{z}^{k+1} = \boldsymbol{z}^k + \mu_k \Delta \boldsymbol{z}^k \tag{2.29}$$

to find the minimum of $f(\boldsymbol{z})$, where $\mu_k \geq 0$ is a positive step size, and

$$\Delta \boldsymbol{z}^k = -\boldsymbol{H}_{\boldsymbol{z}^*\boldsymbol{z}}^{-1} \boldsymbol{g}(\boldsymbol{z}^k) \tag{2.30}$$

is the search direction or *pseudo-Newton step* in the kth iteration. According to (2.24) and (2.25), the pseudo-Newton step is computed as

$$\Delta \boldsymbol{z}^k = -\frac{2}{p}\left(\boldsymbol{C}^{\mathrm{H}} \boldsymbol{\Phi}(\boldsymbol{z}^k) \boldsymbol{C}\right)^{-1} \boldsymbol{C}^{\mathrm{H}} \boldsymbol{\Phi}(\boldsymbol{z}^k)(\boldsymbol{C}\boldsymbol{z}^k - \boldsymbol{b}). \tag{2.31}$$

Selection of the step size is an important issue. In conventional Newton's method, the fixed step size $\mu_k = 1$ is adopted, which is clearly not an optimal choice. If a fixed step size $\mu_k = p/2$ is used, then (2.29) can be simplified to a fixed-point iteration

$$\boldsymbol{z}^{k+1} = \left(\boldsymbol{C}^{\mathrm{H}} \boldsymbol{\Phi}(\boldsymbol{z}^k) \boldsymbol{C}\right)^{-1} \boldsymbol{C}^{\mathrm{H}} \boldsymbol{\Phi}(\boldsymbol{z}^k) \boldsymbol{b} \tag{2.32}$$

which is reduced to the widely used IRLS algorithm [76–78] with $\boldsymbol{\Phi}(\boldsymbol{z}^k)$ being the weighting matrix. The relation between the IRLS and pseudo-Newton's methods for ℓ_p-norm minimization is revealed in the following theorem.

Theorem 2 *The widely used IRLS approach to robust linear fitting based on the ℓ_p-norm minimization is a special case of the pseudo-Newton's method using a fixed step size of $p/2$.*

However, the fixed step size strategy is not optimal. We consider the variable step size. For a given Newton direction $\Delta \boldsymbol{z}^k$, the optimal step size is given by solving the line search

$$\mu_k = \arg\min_{\mu \geq 0} \left\| \boldsymbol{C}\left(\boldsymbol{z}^k + \mu \Delta \boldsymbol{z}^k\right) - \boldsymbol{b} \right\|_p^p. \tag{2.33}$$

Denoting the residual vector in the kth iteration as

$$\boldsymbol{r}^k = \boldsymbol{C}\boldsymbol{z}^k - \boldsymbol{b} \tag{2.34}$$

the optimal step size is determined by

$$\min_{\mu \geq 0} f(\mu) := \left\| \boldsymbol{r}^k + \mu \boldsymbol{C} \Delta \boldsymbol{z}^k \right\|_p^p. \tag{2.35}$$

This is a one-dimensional optimization problem and can be easily solved by the existing line search techniques, such as Golden section search or tangential method [79]. The global optimality of μ is guaranteed since $f(\mu)$ is unimodal with respect to μ if $p > 1$.

Unlike the pseudo-Newton's and IRLS algorithms that only use the leading partial Hessian matrix, the full-Newton's method employs the full $2Q \times 2Q$ Hessian matrix to compute the search direction

$$\begin{bmatrix} \Delta\boldsymbol{z}^k \\ (\Delta\boldsymbol{z}^k)^* \end{bmatrix} = -\begin{bmatrix} \boldsymbol{H}_{\boldsymbol{z}^*\boldsymbol{z}} & \boldsymbol{H}_{\boldsymbol{z}^*\boldsymbol{z}^*} \\ \boldsymbol{H}_{\boldsymbol{z}\boldsymbol{z}} & \boldsymbol{H}_{\boldsymbol{z}\boldsymbol{z}^*} \end{bmatrix}^{-1} \begin{bmatrix} \boldsymbol{g}(\boldsymbol{z}^k) \\ \boldsymbol{g}^*(\boldsymbol{z}^k) \end{bmatrix} \tag{2.36}$$

and the updating rule is

$$\begin{bmatrix} \boldsymbol{z}^{k+1} \\ (\boldsymbol{z}^{k+1})^* \end{bmatrix} = \begin{bmatrix} \boldsymbol{z}^k \\ (\boldsymbol{z}^k)^* \end{bmatrix} + \mu_k \begin{bmatrix} \Delta\boldsymbol{z}^k \\ (\Delta\boldsymbol{z}^k)^* \end{bmatrix} \tag{2.37}$$

where the optimal step size is determined according to (2.33). Equations (2.36) and (2.37) give a true Newton's method since it utilizes all second-order derivatives. The initial value of the IRLS and two Newton's methods can be taken as the least-squares (LS) solution $\boldsymbol{z}^0 = \widehat{\boldsymbol{z}}_{\mathrm{LS}} = \left(\boldsymbol{C}^{\mathrm{H}}\boldsymbol{C}\right)^{-1}\boldsymbol{C}^{\mathrm{H}}\boldsymbol{b}$.

Note that it is not necessary to directly compute the inverse of the Hessian matrices of (2.30) and (2.36) when computing the pseudo-Newton step and Newton step. For example, we can use some efficient algorithms such as the conjugate gradient (CG) method [80] for solving the linear equation $\boldsymbol{H}_{\boldsymbol{z}^*\boldsymbol{z}}\Delta\boldsymbol{z}^k = -\boldsymbol{g}(\boldsymbol{z}^k)$ to obtain the pseudo-Newton step $\Delta\boldsymbol{z}^k$.

The complexity of matrix multiplication $\boldsymbol{C}^{\mathrm{H}}\boldsymbol{\Phi}(\boldsymbol{z})\boldsymbol{C}$ is $\mathcal{O}(mr^2)$ because $\boldsymbol{\Phi}(\boldsymbol{z})$ is diagonal. This is the complexity of computing the partial and full Hessian matrices. The complexity for solving the partial and full Newton steps of (2.30) and (2.36) is $\mathcal{O}(r^3)$. Hence the complexity of the Newton's methods is $\mathcal{O}(mr^2)$ in each iteration due to $m > r$, which is the same as that of the IRLS.

The convergence rates of the IRLS and two Newton's methods with optimal step sizes for a variety of values of p are compared. We takes four values of p as example, i.e., $p = 1.2$, 1.5, 3.3, and 4. In this numerical example, we randomly generate the coefficient matrix $\boldsymbol{C} \in \mathbb{C}^{50\times 20}$ and the vector $\boldsymbol{b} \in \mathbb{C}^{50}$. We are primarily interested in the behavior, as a function of the number of iterations, of the relative error $|f(\boldsymbol{z}^k) - f(\boldsymbol{z}^\star)|/f(\boldsymbol{z}^\star)$, where $f(\boldsymbol{z}^\star)$ is the global minimum. We can calculate this global minimum exactly (in practice up to computer round-off precision) with a finite number of steps using the proposed Newton's method or any optimization software package in advance. Figure 2.3

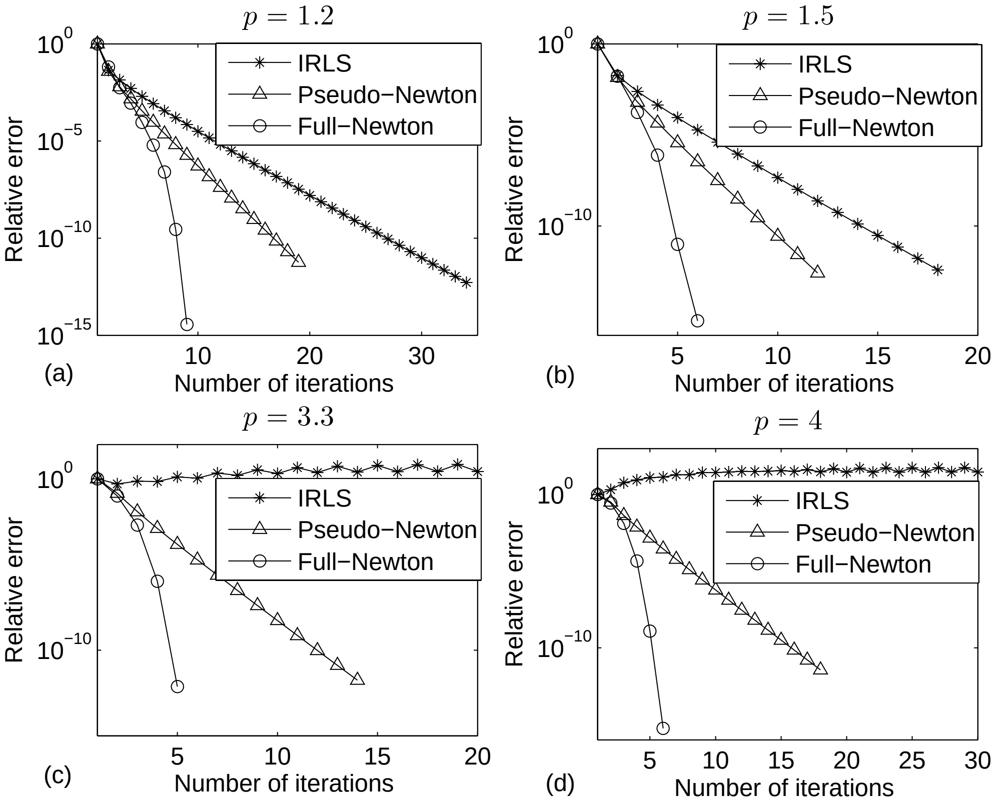


Figure 2.3: Convergence rate versus number of iterations of IRLS and two Newton's methods with optimal step sizes for $p = 1.2$, 1.5, 3.3, and 4.

shows the convergence rates of the three methods. It is observed that the IRLS does not converge for $p = 3.3$ and $p = 4$ while the two Newton's methods converge in all cases. When the IRLS converges, it has a linear convergence rate. The pseudo-Newton's method also has a linear convergence rate but it converges faster than the IRLS. The full-Newton's method has a quadratic convergence rate and converges very fast. It only needs several iterations for convergence with a high accuracy.

Although the AM provides an better approach to robust low-rank factorization under ℓ_p-minimization compared with the IR-SVD, it has two drawbacks. First, its computational complexity is still a bit high. Second, the point that the AM converges to severely depends on the initial value. In the simulation results of Section 2.7, we observe that the AM may converge to an inferior solution sometimes even with good initialization. In the next section, we will discuss the ADMM for robust low-rank factorization, which is computationally more efficient and more effective than the AM.

2.4 ADMM for ℓ_1-Low-Rank Factorization

This section describes the ADMM for solving the problem of ℓ_1-low-rank factorization

$$\min_{\boldsymbol{U},\boldsymbol{V}} f(\boldsymbol{U},\boldsymbol{V}) := \|\boldsymbol{U}\boldsymbol{V} - \boldsymbol{A}\|_1. \tag{2.38}$$

Here we only consider the ℓ_1-norm, i.e., $p = 1$, rather than all possible value in $1 \le p < 2$ because the choice of $p = 1$ is more robust to outliers and computationally simpler than $p \in (1, 2)$, as we will see later. Another reason is that the proximity operator of the ℓ_1-norm has the closed-form expression while those with other values of p are more difficult to compute for the complex-valued variables.

2.4.1 Principles of ADMM

By introducing the matrix $\boldsymbol{E} = \boldsymbol{U}\boldsymbol{V} - \boldsymbol{A}$, (2.38) is equivalent to a linearly constrained problem

$$\begin{aligned} &\min_{\boldsymbol{U},\boldsymbol{V},\boldsymbol{E}} \|\boldsymbol{E}\|_1 \\ &\text{s.t. } \boldsymbol{E} = \boldsymbol{U}\boldsymbol{V} - \boldsymbol{A}. \end{aligned} \tag{2.39}$$

The augmented Lagrangian function of (2.39) is

$$\begin{aligned} \mathcal{L}_\mu(\boldsymbol{U},\boldsymbol{V},\boldsymbol{E},\boldsymbol{\Lambda}) = \|\boldsymbol{E}\|_1 &+ \mathrm{Re}(\langle \boldsymbol{\Lambda}, \boldsymbol{U}\boldsymbol{V} - \boldsymbol{E} - \boldsymbol{A}\rangle) \\ &+ \frac{\mu}{2}\|\boldsymbol{U}\boldsymbol{V} - \boldsymbol{E} - \boldsymbol{A}\|_{\mathrm{F}}^2 \end{aligned} \tag{2.40}$$

where the matrix $\boldsymbol{\Lambda} \in \mathbb{C}^{m\times n}$ contains mn Lagrange multipliers, $\langle \boldsymbol{A}, \boldsymbol{B}\rangle = \sum_{(i,j)} \boldsymbol{A}_{ij}^* \boldsymbol{B}_{ij}$ represents the inner product of two complex-valued matrices where $\boldsymbol{A}_{ij}$ and $\boldsymbol{B}_{ij}$ are the (i, j)th entries of $\boldsymbol{A}$ and $\boldsymbol{B}$, respectively, and $\mu > 0$ is the penalty parameter. The augmented Lagrangian reduces to the unaugmented one if $\mu = 0$. The selection of μ is flexible [81]. One can simply use a fixed positive constant for μ. Of course, using possibly different penalty parameters for each iteration may improve the convergence in practice [82, 83]. The Lagrange multiplier method solves the constrained problem of (2.39) by finding a saddle point of the augmented Lagrangian

$$\min_{\boldsymbol{U},\boldsymbol{V},\boldsymbol{E}} \max_{\boldsymbol{\Lambda}} \mathcal{L}_\mu(\boldsymbol{U},\boldsymbol{V},\boldsymbol{E},\boldsymbol{\Lambda}). \tag{2.41}$$

This saddle point problem is a minimax problem, where the primal variables $(\boldsymbol{U},\boldsymbol{V},\boldsymbol{E})$ aims at decreasing $\mathcal{L}_\mu(\boldsymbol{U},\boldsymbol{V},\boldsymbol{E},\boldsymbol{\Lambda})$ while the dual variable $\boldsymbol{\Lambda}$ aims at increasing this function. ADMM uses the following iteration

$$(\boldsymbol{U}^{k+1}, \boldsymbol{V}^{k+1}) = \arg\min_{\boldsymbol{U},\boldsymbol{V}} \mathcal{L}_\mu(\boldsymbol{U},\boldsymbol{V},\boldsymbol{E}^k,\boldsymbol{\Lambda}^k) \tag{2.42}$$

$$\boldsymbol{E}^{k+1} = \arg\min_{\boldsymbol{E}} \mathcal{L}_\mu(\boldsymbol{U}^{k+1},\boldsymbol{V}^{k+1},\boldsymbol{E},\boldsymbol{\Lambda}^k) \tag{2.43}$$

$$\boldsymbol{\Lambda}^{k+1} = \boldsymbol{\Lambda}^k + \mu\big(\boldsymbol{U}^{k+1}\boldsymbol{V}^{k+1} - \boldsymbol{E}^{k+1} - \boldsymbol{A}\big) \tag{2.44}$$

to calculate the saddle point in (2.41), where $(\boldsymbol{U}^k, \boldsymbol{V}^k, \boldsymbol{E}^k, \boldsymbol{\Lambda}^k)$ denotes the result at the kth iteration. Some remarks and explanations on three subproblems of (2.42), (2.43), and (2.44) are given as follows.

Noting that the gradient of $\mathcal{L}_\mu(\boldsymbol{U}^{k+1}, \boldsymbol{V}^{k+1}, \boldsymbol{E}^{k+1}, \boldsymbol{\Lambda})$ with respect to $\boldsymbol{\Lambda}$ is

$$\frac{\partial \mathcal{L}_\mu(\boldsymbol{U}^{k+1}, \boldsymbol{V}^{k+1}, \boldsymbol{E}^{k+1}, \boldsymbol{\Lambda})}{\partial \boldsymbol{\Lambda}^*} = \boldsymbol{U}^{k+1}\boldsymbol{V}^{k+1} - \boldsymbol{E}^{k+1} - \boldsymbol{A} \tag{2.45}$$

we can see that (2.44) adopts a gradient ascent with a step size μ to update the dual variable $\boldsymbol{\Lambda}$. Since $\boldsymbol{\Lambda}$ is complex-valued, the gradient is defined as the Wirtinger derivatives, which is written as $\frac{\partial \mathcal{L}_\mu}{\partial \boldsymbol{\Lambda}^*}$. ADMM updates $(\boldsymbol{U}, \boldsymbol{V})$ and $\boldsymbol{E}$ in an alternating or sequential fashion to circumvent the difficulty in jointly minimizing with respect to the two primal blocks. Note that (2.42) minimizes $(\boldsymbol{U}, \boldsymbol{V})$ simultaneously. Thus, (2.42)–(2.44) is a two-block ADMM but not a three-block one. The two blocks refer to $(\boldsymbol{U}, \boldsymbol{V})$ and $\boldsymbol{E}$. The convergence of two-block ADMM is guaranteed but the multi-block (block number larger than 2) one is not necessarily convergent [84]. Clearly, the proposed ADMM does not have the divergence problem of multi-block ADMM since it is a two-block one.

Denoting a matrix

$$\boldsymbol{Y}^k = \boldsymbol{E}^k - \boldsymbol{\Lambda}^k/\mu + \boldsymbol{A} \tag{2.46}$$

and ignoring the constant term independent on $(\boldsymbol{U}, \boldsymbol{V})$, we can derive the subproblem of (2.42) is equivalent to the following Frobenius norm minimization problem

$$(\boldsymbol{U}^{k+1}, \boldsymbol{V}^{k+1}) = \arg\min_{\boldsymbol{U}, \boldsymbol{V}} \|\boldsymbol{U}\boldsymbol{V} - \boldsymbol{Y}^k\|_{\mathrm{F}}^2 \tag{2.47}$$

whose global minimizer can be obtained by the truncated SVD of $\boldsymbol{Y}^k$, which is denoted as

$$\mathrm{TSVD}_r(\boldsymbol{Y}^k) = \boldsymbol{G}^k \boldsymbol{\Sigma}_Y^k (\boldsymbol{P}^k)^{\mathrm{H}} \tag{2.48}$$

where $\boldsymbol{\Sigma}_Y^k \in \mathbb{R}_+^{r\times r}$ is a diagonal matrix whose diagonal elements are the r dominant singular values of $\boldsymbol{Y}^k$, the columns of $\boldsymbol{G}^k \in \mathbb{C}^{m\times r}$ and $\boldsymbol{P}^k \in \mathbb{C}^{n\times r}$ are the corresponding left and right singular vectors, respectively. Clearly, we have

$$\boldsymbol{U}^{k+1} = \boldsymbol{G}^k (\boldsymbol{\Sigma}_Y^k)^{1/2},\ \boldsymbol{V}^{k+1} = (\boldsymbol{\Sigma}_Y^k)^{1/2} (\boldsymbol{P}^k)^{\mathrm{H}}. \tag{2.49}$$

The complexity of the truncated SVD is $\mathcal{O}(mnr)$ [35]. When the rank r is smaller than the matrix dimension (m, n), the computational cost of the truncated SVD is cheaper than the full SVD which requires a complexity of $\mathcal{O}(\max(mn^2, m^2n))$. After simplification, the subproblem of (2.43) is concisely expressed as

$$\min_{\boldsymbol{E}\in\mathbb{C}^{m\times n}} \frac{1}{2}\left\|\boldsymbol{E} - \boldsymbol{Y}^k\right\|_{\mathrm{F}}^2 + \frac{1}{\mu}\|\boldsymbol{E}\|_1. \tag{2.50}$$

where

$$\boldsymbol{Y}^k = \boldsymbol{U}^{k+1}\boldsymbol{V}^{k+1} + \boldsymbol{\Lambda}^k/\mu - \boldsymbol{A}. \tag{2.51}$$

The solution of (2.50) defines the proximity operator [85] of the ℓ_1-norm of a complex-valued matrix. Observing that (2.50) is separable, it can be decomposed into mn independent scalar minimization problems

$$\min_{e_{ij}\in\mathbb{C}} \frac{1}{2}\left(e_{ij} - y_{ij}^k\right)^2 + \frac{1}{\mu}|e_{ij}| \tag{2.52}$$

where $1 \leq i \leq m$, $1 \leq j \leq n$, e_{ij} and y_{ij}^k are the (i,j)th entries of $\boldsymbol{E}$ and $\boldsymbol{Y}^k$, respectively. The solution of (2.52) is the soft-thresholding operator for *complex variables*, which is a generalization of that for real variables [86] and has the following closed-form expression [87, 88]

$$e_{ij}^{k+1} = \frac{\max\left(|y_{ij}^k| - 1/\mu, 0\right)}{\max\left(|y_{ij}^k| - 1/\mu, 0\right) + 1/\mu} y_{ij}^k. \tag{2.53}$$

Obviously, it only needs a marginal complexity of $\mathcal{O}(mn)$ to update $\boldsymbol{E}$. Note that the variables are complex-valued in array processing. We should adopt the complex soft-thresholding operator of (2.53), rather than the celebrated one for real variables [86]. Now it is clear that why we only consider the choice of $p = 1$. This is because the proximity operator of the ℓ_1-norm has a simple closed-form solution while the ℓ_p-norm with $1 < p < 2$ does not. Although the proximity operator of the pth power of the ℓ_p-norm can be solved since it is a convex problem for $1 < p < 2$, it has no closed-form solution and requires an iterative procedure, which is time-consuming. It is well-known that the soft-thresholding shrinks the value larger than the threshold towards to zero [40]. Therefore, it automatically achieves outlier reduction. This is why the ℓ_1-subspace decomposition is more robust against outliers.

2.4.2 Summary of ADMM

The steps of ADMM for robust low-rank approximation are summarized in Algorithm 2.

From the steps of Algorithm 2, we see that the ADMM converts the minimization of a nonsmooth ℓ_1-norm into a Frobenius norm minimization at each iteration, which can be efficiently solve by truncated SVD. The additional cost for computing the soft-thresholding operator is quite marginal because it has a simple closed-form solution. The residual

$$\boldsymbol{R}^k = \boldsymbol{U}^k\boldsymbol{V}^k - \boldsymbol{E}^k - \boldsymbol{A} \tag{2.54}$$

Algorithm 2 ADMM for Robust Low-Rank Approximation

```
Input: $\boldsymbol{A}$, rank $r$, and $\mu > 0$
  Initialize: $\boldsymbol{E}^0 = \mathbf{0}$ and $\boldsymbol{\Lambda}^0 = \mathbf{0}$
  for $k = 0, 1, 2 \cdots$ do
      1) Calculate $\boldsymbol{Y}^k = \boldsymbol{E}^k - \boldsymbol{\Lambda}^k/\mu + \boldsymbol{A}$
      2) Compute rank-$r$ truncated SVD of $\boldsymbol{Y}^k$:
           $\mathrm{TSVD}_r(\boldsymbol{Y}^k) = \boldsymbol{G}^k \boldsymbol{\Sigma}_{\mathrm{Y}}^k (\boldsymbol{P}^k)^{\mathrm{H}}$
      3) Update $\boldsymbol{U}^{k+1} = \boldsymbol{G}^k(\boldsymbol{\Sigma}_y^k)^{1/2}$ and $\boldsymbol{V}^{k+1} = (\boldsymbol{\Sigma}^k)^{1/2}(\boldsymbol{P}^k)^{\mathrm{H}}$.
      4) Compute $\boldsymbol{Y}^k = \boldsymbol{U}^{k+1}\boldsymbol{V}^{k+1} + \boldsymbol{\Lambda}^k/\mu - \boldsymbol{A}$
      5) Update the elements of $\boldsymbol{E}$ using soft-thresholding:
           $e_{ij}^{k+1} = \frac{\max\left(|y_{ij}^k| - 1/\mu, 0\right)}{\max\left(|y_{ij}^k| - 1/\mu, 0\right) + 1/\mu} y_{ij}^k$
      6) $\boldsymbol{\Lambda}^{k+1} = \boldsymbol{\Lambda}^k + \mu\left(\boldsymbol{U}^{k+1}\boldsymbol{V}^{k+1} - \boldsymbol{E}^{k+1} - \boldsymbol{A}\right)$
      Stop if termination condition satisfied.
  end for
Output: $(\boldsymbol{U}^{k+1}, \boldsymbol{V}^{k+1})$
```

reflects how well the current iterate satisfies the linear constraint and can be used to check for convergence. Specifically, the iteration is terminated when the normalized Frobenius norm of the residual

$$\frac{\|\boldsymbol{R}^k\|_{\mathrm{F}}}{\|\boldsymbol{A}\|_{\mathrm{F}}} < \delta \tag{2.55}$$

satisfies, where $\delta > 0$ is a small tolerance parameter. A reasonable value can be taken as $\delta = 10^{-3}$. The dominant complexity of the ADMM per iteration is calculating the truncated SVD. Hence, the total complexity of the ADMM is $\mathcal{O}(mnrN_{\mathrm{ADMM}})$ where N_{ADMM} is the iteration number of the ADMM. As can be seen from the numerical results in Section 2.7, several tens is enough for N_{ADMM} for attaining a small estimate error in impulsive noise. The computational cost of the ADMM is much lower than that of the AM that requires $\mathcal{O}(mnr^2 N_{\mathrm{IRLS}} N_{\mathrm{AM}})$ operations.

As an important technique in optimization, the ADMM has been widely used in signal processing, machine learning and statistics. However, we still have novel contributions to the ADMM in this chapter, which is summarized as follows.

i) The ADMM is first to apply to efficiently solve the challenging nonconvex and nonsmooth problem induced by the low-rank approximation using ℓ_1-minimization, which is much more difficult than the ℓ_2-subspace decomposition that can be solved by truncated SVD.

ii) The proposed ADMM converts the nonsmooth ℓ_1-subspace factorization into a series of ℓ_2-subspace factorization. At each iteration, it just needs calculation of truncated SVD and soft-thresholding, which automatically achieves outlier reduction. The ADMM has a lower computational complexity than the AM.

iii) The ADMM provides an improved and more effective approach to robust subspace decomposition. It is revealed that the AM sometimes yields inferior solutions. The ADMM converges to a better solution having smaller objective function value and more accurate subspace estimation, indicating that it is numerically superior to the AM.

2.5 Differences with Other Robust PCA Methods

The RPCA proposed by Candès *et al.* [32] models the observed data matrix as a superposition of a low-rank matrix $\boldsymbol{L}$ and a sparse outlier matrix $\boldsymbol{O}$, i.e., $\boldsymbol{A} = \boldsymbol{L} + \boldsymbol{O}$. It solves the following convex programming

$$\begin{aligned} &\min_{\boldsymbol{L},\boldsymbol{O}} \|\boldsymbol{L}\|_* + \alpha\|\boldsymbol{O}\|_1 \\ &\text{s.t. } \boldsymbol{L} + \boldsymbol{O} = \boldsymbol{A}. \end{aligned} \tag{2.56}$$

to separate the low-rank matrix $\boldsymbol{L}$ and the outlier matrix $\boldsymbol{O}$, where $\alpha > 0$. The convex nuclear norm, which is defined as the sum of the singular values, is taken as the surrogate of the nonconvex rank while the entry-wise ℓ_1-norm is used instead of the nonconvex ℓ_0-norm for sparsity-promoting. If the outlier matrix has other special structures, the ℓ_1-norm can be adapted to other norms. For example, the $\ell_{1,2}$-norm $\|\boldsymbol{O}\|_{1,2}$, which is the sum of ℓ_2-norms of the columns, is taken if $\boldsymbol{O}$ is column-sparse, which is used in the outlier pursuit method [89]. Similarly, the $\ell_{2,1}$-norm $\|\boldsymbol{O}\|_{2,1}$, which is the sum of ℓ_2-norms of the rows, if $\boldsymbol{O}$ has row-sparse structure [90]. The difference between our method and the RPCA is mainly that our method directly uses low-rank factorization to ensure low-rank property while the RPCA adopts nuclear norm minimization. Although the global minimum of the convex programming in (2.56) is guaranteed, its complexity is much higher than the proposed algorithms, which limits its application to large-scale data. In addition, the parameter α is not easy to determine even when the rank is known. A careful parameter tune for selecting an appropriate α is required. In many applications such as dimensionality reduction and subspace decomposition, after obtaining the "clean" data $\boldsymbol{L}$ with the outlier being removed, the RPCA still requires performing SVD of $\boldsymbol{L}$ to obtain the principal components, but our method directly calculates the subspace.

2.6 Application to Array Processing

2.6.1 Overview on DOA Estimation

DOA estimation of multiple emitting sources is an important issue in array processing and has various applications in radar, sonar, wireless communications, and source localization [91–93]. The MUSIC [94] is one of the most well-known high resolution DOA estimation techniques and it belongs to the subspace methodology [95]. It estimates the DOAs by exploiting the orthogonality between the noise subspace and array manifold. It has been shown that the MUSIC method is an asymptotically unbiased and efficient DOA estimator based on the Gaussian noise assumption [96–98].

Many existing DOA estimators explicitly or implicitly assume that the ambient noise is Gaussian distributed. However, the noise in practice often exhibits non-Gaussian properties. The performance of the conventional DOA estimators may severely degrade in the presence of non-Gaussian noise. One important class of non-Gaussian noises that are frequently encountered in many practical wireless radio systems is impulsive noise, also known as burst noise [99–101]. The probability density function (p.d.f.) of impulsive noise has heavier tails than the Gaussian distribution. The property of impulsive noise is somewhat similar to outliers in statistics. It is because the heavy tailed distributions give higher probability of occurrence to values which exceed a few standard deviations than the Gaussian distribution. Under a nominal Gaussian noise model, these large values are unlikely to appear and can therefore be considered as outliers.

The conventional subspace based DOA estimation techniques exploit eigenvalue decomposition (EVD) of the covariance matrix of the received data. The DOA estimators based on the second-order sample covariance are not robust against outliers. A class of subspace based DOA estimation algorithms use the fractional lower-order statistics such as the robust covariation (ROC) [102], fractional lower-order moments (FLOM) [103], sign covariance matrix (SCM), and Kendall's tau covariance matrix (TCM) [104], instead of the second-order sample covariance. However, the fractional lower-order statistics based algorithms are suboptimal and require large sample sizes for a satisfactory performance [102, 105]. Swami *et al.* have proposed to apply zero-memory nonlinear (ZMNL) functions to limit the influence of outliers by clipping the amplitude of the received signal [106]. The ZMNL preprocessing achieves robust covariance estimation and provides more accurate DOA estimates than the fractional lower-order schemes [105, 106]. Furthermore, the data-adaptive ZMNL approach is simple and has a low computational complexity. Despite these advantages, there is a tradeoff in ZMNL between outlier suppression and subspace preservation. The ZMNL preprocessing generally destroys the low-rank property of the signal subspace. Its performance may degrade due to the rank increase of the signal subspace [105]. Similar idea using outlier-trimming has been developed in [107], where the Shapiro-Wilk W test for Gaussianity is used. One of its limitations is that Gaussian distributed source signal is required [107].

Another representative DOA estimation scheme resistant to impulsive noise is based on robust statistics [29, 78, 108]. This approach first uses a robust scheme such as the M-estimator [29], S-estimator [78], or MM-estimator [109], to estimate the covariance matrix, and then conventional subspace decomposition is exploited to obtain the DOA estimates. The success of this method depends on choosing the appropriate robust statistics. Different from the covariance based methodology which exploits the sample covariance, fractional-order moment, or any robust statistics computed from the received data, we directly compute the signal subspace without explicitly constructing the covariance. Naturally, the EVD is not required.

The key step of the MUSIC method is computing the signal or noise subspace. The subspace decomposition rule in MUSIC is equivalent to minimization of the Frobenius norm of the residual fitting error matrix. The resulting Frobenius norm minimization can be efficiently solved by the SVD of the received data matrix. The orthonormal bases of the signal and noise subspaces are given by the singular vectors associated with the principal and minor singular values, respectively. The subspace decomposition using Frobenius norm minimization is statistically optimal when the additive noise is Gaussian distributed. It, however, is no longer optimal and the performance of the conventional MUSIC. based on SVD will degrade in the presence of impulsive noise. We call the robust DOA estimator employing ℓ_p-low-rank matrix factorization as ℓ_p-MUSIC method [110].

2.6.2 Signal Model and ℓ_p-MUSIC

Consider a uniform linear array (ULA) of m sensors with half-wavelength inter-sensor spacing. The ULA receives r far-field and narrowband sources emitting plane waves. We assume that the number of sources is less than the number of sensors, i.e., $r < m$. This assumption is common for subspace based array processing. Letting the first sensor as the reference sensor, then the complex baseband signal received by the ith

$(i = 1, \cdots, m)$ sensor is

$$a_i(t) = \sum_{j=1}^{r} s_j(t) \mathrm{e}^{\imath\pi(i-1)\sin\theta_j} + \xi_i(t) \tag{2.57}$$

where t is the discrete-time index, $s_j(t)$ is the jth $(j = 1, \cdots, r)$ source with θ_j being its DOA, $\xi_i(t)$ is the non-Gaussian additive noise of the ith sensor. Stacking the output of all the sensors in a vector $\boldsymbol{a}_t = [a_1(t), \cdots, a_m(t)]^{\mathrm{T}} \in \mathbb{C}^m$, the matrix-vector formulation of (2.57) is expressed as

$$\boldsymbol{a}_t = \boldsymbol{F}\boldsymbol{s}_t + \boldsymbol{\xi}_t \tag{2.58}$$

where $\boldsymbol{s}_t = [s_1(t), \cdots, s_r(t)]^{\mathrm{T}} \in \mathbb{C}^r$ is the source vector, $\boldsymbol{\xi}_t = [\xi_1(t), \cdots, \xi_m(t)]^{\mathrm{T}} \in \mathbb{C}^m$ is the noise vector, and $\boldsymbol{F} \in \mathbb{C}^{m \times r}$ is the array manifold matrix having the following form

$$\boldsymbol{F} = [\boldsymbol{f}(\theta_1), \cdots, \boldsymbol{f}(\theta_r)] \tag{2.59}$$

with $\boldsymbol{f}(\theta)$ being the steering vector:

$$\boldsymbol{f}(\theta) = \left[1, \mathrm{e}^{\imath\pi\sin\theta}, \cdots, \mathrm{e}^{\imath\pi(m-1)\sin\theta}\right]^{\mathrm{T}}. \tag{2.60}$$

We aim at estimating the DOAs of the r sources based on the n snapshots, which are collected in the following matrix

$$\boldsymbol{A} \triangleq [\boldsymbol{a}_1, \cdots, \boldsymbol{a}_n] \in \mathbb{C}^{m \times n}. \tag{2.61}$$

The source number r is assumed known or has been determined by an outlier-resistant source enumeration method [108, 111, 112]. It is also assumed that the zero-mean sources are mutually independently with each other, while the noises $\{\xi_i(t)\}_{i=1}^{m}$ are spatially uncorrelated and temporally white, and statistically independent of the sources. These assumptions are mild in practical applications.

After obtaining the subspace $\boldsymbol{U}$ through the robust matrix factorization algorithm, e.g., the AM, ADMM, or IR-SVD, the projection matrix onto the signal subspace is computed as

$$\boldsymbol{\Pi}_{\boldsymbol{U}} = \boldsymbol{U}\left(\boldsymbol{U}^{\mathrm{H}}\boldsymbol{U}\right)^{-1}\boldsymbol{U}^{\mathrm{H}}. \tag{2.62}$$

Because the steering vector of any source is orthogonal to the null space, the robust spatial spectrum is given by

$$P(\theta) = \frac{1}{\boldsymbol{f}^{\mathrm{H}}(\theta)(\boldsymbol{I} - \boldsymbol{\Pi}_{\boldsymbol{U}})\boldsymbol{f}(\theta)}. \tag{2.63}$$

The DOA estimates can be obtained by searching for the peaks of (2.63). The root finding technique, namely, root-MUSIC [113], can also be used to compute the DOAs instead of the spectrum search. Other subspace DOA estimation schemes such as ESPRIT [114], can also be applied.

It is expected that the estimated signal subspace $\boldsymbol{U}$ spans the same range space of $\boldsymbol{F}$. Here we take the normalized *subspace distance* between $\boldsymbol{U}$ and $\boldsymbol{F}$ as the performance measure to evaluate the quality of subspace estimate, which is defined as [35]

$$\mathrm{SD}(\boldsymbol{U},\boldsymbol{F}) = \frac{\|\boldsymbol{\Pi}_{\boldsymbol{U}} - \boldsymbol{\Pi}_{\boldsymbol{F}}\|_{\mathrm{F}}}{\|\boldsymbol{\Pi}_{\boldsymbol{F}}\|_{\mathrm{F}}} \tag{2.64}$$

where $\boldsymbol{\Pi}_{\boldsymbol{F}} = \boldsymbol{F}(\boldsymbol{F}^{\mathrm{H}}\boldsymbol{F})^{-1}\boldsymbol{F}^{\mathrm{H}}$ is the projection matrix onto the column space of $\boldsymbol{F}$. The subspace distance will become zero if $\boldsymbol{U}$ and $\boldsymbol{F}$ spans the same column space. A smaller subspace distance indicates a better subspace estimate. It should be pointed out that the global optimum is not guaranteed for both ADMM and AM, but the ADMM can provide an good enough subspace estimate for robust array processing, as we will see in the simulation results.

2.7 Experimental Results

2.7.1 Impulsive Noise Model

We use two widely used p.d.f. models for impulsive noise, i.e., the Gaussian mixture model (GMM) and generalized Gaussian distribution (GGD) in the experiments.

GMM. The p.d.f. of the two-term circular Gaussian mixture noise $\xi(t)$ is

$$p_\xi(\xi) = \sum_{i=1}^{2} \frac{c_i}{\pi\sigma_i^2} \exp\left(-\frac{|\xi|^2}{\sigma_i^2}\right) \tag{2.65}$$

where $c_i \in [0,1]$ and σ_i^2 are the probability and variance of the ith term, respectively, with $c_1 + c_2 = 1$. If $\sigma_2^2 \gg \sigma_1^2$ and $c_2 < c_1$ are selected, large noise samples of variance σ_2^2 occurring with a smaller probability c_2 are the outliers embedded in Gaussian background noise of variance σ_1^2. Thus, GMM well models the phenomenon in the presence of both Gaussian thermal noise and impulsive noise. The total variance of $\xi(t)$ is $\sigma_\xi^2 = \sum_i c_i\sigma_i^2$. In the simulations, we set $\sigma_2^2 = 100\sigma_1^2$ and $c_2 = 0.1$, i.e., there are 10% outliers.

GGD. The p.d.f. of the circular zero-mean GGD with variance σ_ξ^2 is

$$p_\xi(\xi) = \frac{\beta\Gamma(4/\beta)}{2\pi\sigma_\xi^2\Gamma^2(2/\beta)} \exp\left(-\frac{|\xi|^\beta}{c\sigma_\xi^\beta}\right) \tag{2.66}$$

where $\beta > 0$ is the shape parameter, $\Gamma(\cdot)$ is the Gamma function, and $c = (\Gamma(2/\beta)/\Gamma(4/\beta))^{\beta/2}$ [115]. The GGD reduces to the circular Gaussian distribution for $\beta = 2$. $\beta > 2$ models sub-Gaussian noise while $\beta < 2$ models heavy-tailed one. Especially, $\beta = 1$ corresponds to the Laplacian distribution. The smaller the value of β, the more impulsive the noise is. We adopt $\beta = 0.4$ in the simulations.

2.7.2 Convergence Behavior and Running Time

We first compare the convergence behavior of the proposed ADMM and the AM for solving the ℓ_1-subspace decomposition using randomly generated data. For fair comparison, the two algorithms use the same initial value that takes the result of conventional truncated SVD. A noise-free matrix $\boldsymbol{A}_0 \in \mathbb{C}^{m\times n}$ of rank r is generated by the product of two random matrices $\boldsymbol{U}_0 \in \mathbb{C}^{m\times r}$ and $\boldsymbol{V}_0 \in \mathbb{C}^{r\times n}$ whose entries satisfy the standard circular Gaussian distribution, i.e., $\boldsymbol{A}_0 = \boldsymbol{U}_0\boldsymbol{V}_0$. Then, the GMM or GGD noise of variance σ_ξ^2 is added to $\boldsymbol{A}_0$ to obtain the noisy observation $\boldsymbol{A} = \boldsymbol{A}_0 + \boldsymbol{\Xi}$. The signal-to-noise ratio (SNR) is defined as

$$\mathrm{SNR} = \frac{\|\boldsymbol{A}_0\|_\mathrm{F}^2}{mn\sigma_\xi^2} \tag{2.67}$$

where $\|\boldsymbol{A}_0\|_\mathrm{F}^2/(mn)$ represents the average power of the noiseless observation. In this example, we adopt the GMM noise with SNR = 6 dB and set $m = 20$, $n = 50$, and $r = 4$. The results up to the computer round-off precision of AM and ADMM can be obtained using finite iterations, which are denoted as $f_\mathrm{AM}^\star$ and $f_\mathrm{ADMM}^\star$, respectively. Figure 2.4 shows the differences of objective functions of AM and ADMM, i.e., $f_\mathrm{AM}^\star - f_\mathrm{ADMM}^\star$, of 50 independent experiments. We see that all differences are positive, which indicates $f_\mathrm{ADMM}^\star < f_\mathrm{AM}^\star$. Therefore, the ADMM and AM converge to different points. The AM yields inferior solutions. The ADMM converges to a better solution with a smaller objective function value.

Figure 2.5 shows the normalized decreases of the objective function $|f(\boldsymbol{U}^k, \boldsymbol{V}^k) - f^\star|/f(\boldsymbol{U}^0, \boldsymbol{V}^0)$ of the AM and ADMM with different penalty parameters $\mu = 1, 5$, and 10, where $f^\star$ is the minimum. It should be pointed out that the global minimum $f^\star$ is very difficult to obtain and we use $f_\mathrm{AM}^\star$ instead of it when plotting Figure 2.5. Again, $f_\mathrm{AM}^\star$ can be calculated up to the computer round-off precision using finite iterations in advance. Figure 2.6 plots the subspace distance versus iteration number. As we can see, the AM is premature and sticks at an inferior point that has larger objective function value and subspace distance, although it has a rapid decreasing rate at the initial stage. Obviously, the ADMM is more effective. It significantly improves the numerical performance. The subspace estimate obtained by the ADMM is much more accurate

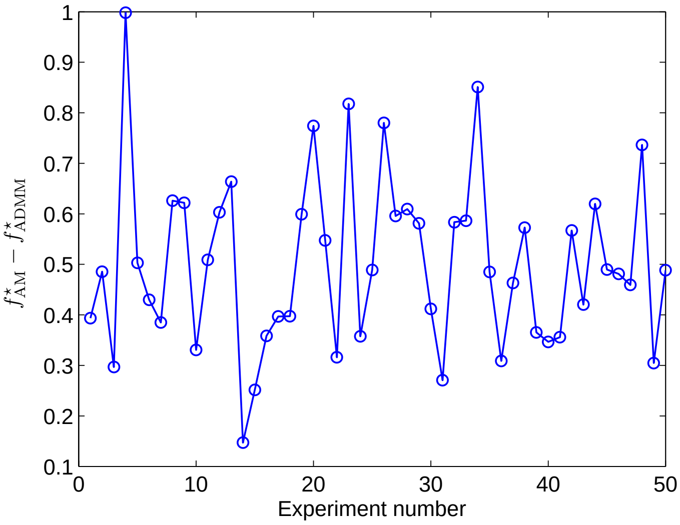


Figure 2.4: Differences of objective functions of AM and ADMM of 50 independent experiments.

than the AM. The normalized Frobenius norm of the residual, i.e., $\|\boldsymbol{E}^k\|_{\mathrm{F}}/\|\boldsymbol{A}\|_{\mathrm{F}}$, versus iteration number is illustrated in Figure 2.7. From Figures 2.5 to Fig:Res:Iter, we see that the ADMM with different values of penalty parameter μ converges to the same solution finally but the convergence rates are different. The penalty parameter μ only influences the convergence speed. In this experiment, the smaller value of $\mu = 1$ has the fastest rate at the initial stage but it slows down later. The results also imply that selection of μ is quite flexible. Figure 2.7 indicates that several tens of iterations are required for the ADMM to attain a normalized residual of 10^{-3} to 10^{-4}. This order-of-magnitude of iterations is also enough for the subspace distance converging.

To compare the computational efficiency, we test the ADMM and AM in MATLAB on a computer with a 3.2 GHz CPU and 4 GB memory. The same experimental settings are taken except that m and n vary. The CPU times (in seconds) of the two algorithms with various values of m and n, which are based on an average of 20 independent runs, are listed in Tables 2.1. The stopping parameter of the ADMM is $\delta = 10^{-3}$. This value is also used as tolerance for the AM, i.e., the AM stops the iteration when the relative change of the objective function is less than 10^{-3}. We see that ADMM is much faster than AM, especially for large-scale matrix. The AM is too time-consuming when the problem size is large. Therefore, the ADMM is much more efficient than the AM for solving large-scale robust subspace decomposition.

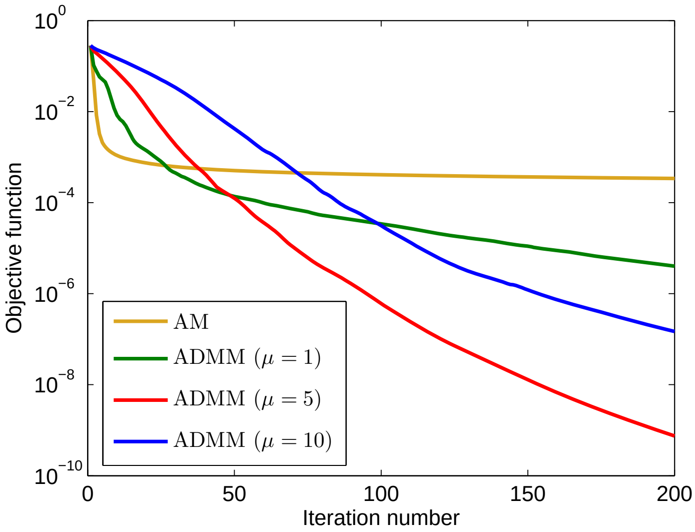


Figure 2.5: Normalized decrease of the objective function of AM and ADMM versus iteration number.

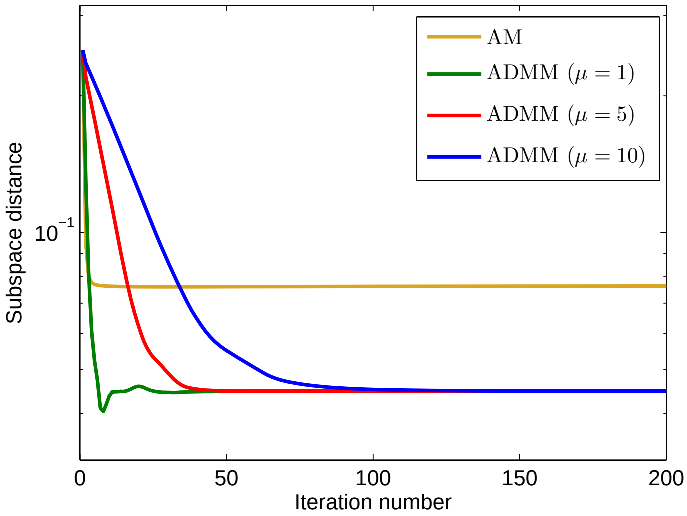


Figure 2.6: Subspace distance versus iteration number.

Table 2.1: CPU times of ADMM and AM

	$m = 20$ $n = 50$	$m = 40$ $n = 100$	$m = 80$ $n = 200$	$m = 200$ $n = 500$	$m = 1000$ $n = 2000$
ADMM	0.137	0.321	0.652	3.05	106.4
AM	11.04	27.82	115.1	1252.7	1.28×10^5

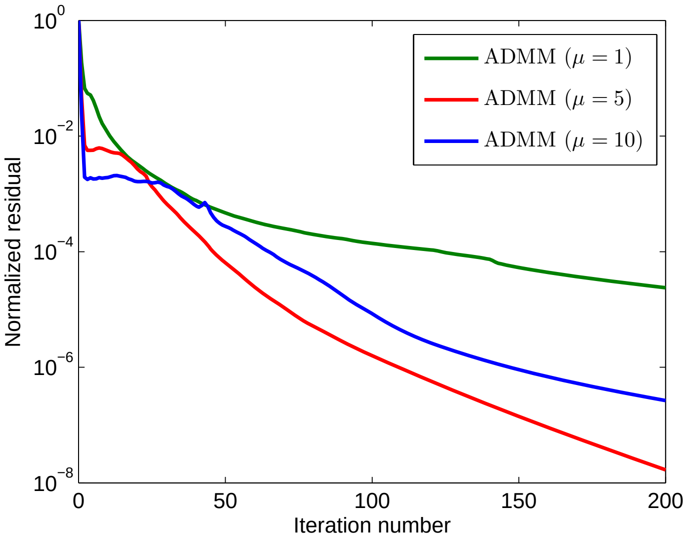


Figure 2.7: Normalized residual of the ADMM versus iteration number.

2.7.3 Results of DOA Estimation

In the second simulation, the performance of the ℓ_1-MUSIC using the ADMM and AM for subspace computation, conventional MUSIC [94], FLOM [103], ROC [102], ZMNL [106], and MM-estimator [108, 109] methods, as well as the Cramér-Rao bound (CRB), are compared for DOA estimation. The ADMM and AM directly compute a robust estimated the subspace but the FLOM, ROC, ZMNL, and MM robustly estimate the covariance and then use conventional SVD or EVD routine to compute the subspace. The ZMNL first uses a Gaussian-tailed ZMNL (GZMNL) function to clip outliers. After this data preprocessing, the conventional MUSIC is applied. Therefore this method is referred to as GZMNL for short. The method in [108] first robustly estimates the covariance matrix by MM-estimator [109], and then employs MUSIC for DOA estimation. ROC and FLOM use fractional lower-order moments instead of the second-order sample covariance matrix. For the purpose of fair comparison, the fractional order used in ROC and FLOM is set as $p = 1$, the same as the ADMM and AM. We consider a ULA with inter-sensor spacing being half a wavelength. The emitting sources are two independent quadrature phase-shift keying (QPSK) signals with equal power. The CRB of for DOA estimation $\boldsymbol{\theta} = [\theta_1, \cdots, \theta_r]^{\mathrm{T}}$ under non-Gaussian noise [105], has been derived by Kozick and Sadler, which is

$$\mathrm{CRB}(\boldsymbol{\theta}) = \frac{1}{I_c}\mathrm{diag}\left\{\sum_{t=1}^{n}\mathrm{Re}\big(\boldsymbol{S}_t^{\mathrm{H}}\boldsymbol{B}^{\mathrm{H}}(\boldsymbol{\theta})\boldsymbol{\Pi}_{\boldsymbol{F}}^{\perp}\boldsymbol{B}(\boldsymbol{\theta})\boldsymbol{S}_t\big)\right\} \tag{2.68}$$

where $\boldsymbol{S}_t = \mathrm{diag}\{s_1(t), \cdots, s_r(t)\}$ is a diagonal matrix, $\boldsymbol{B}(\boldsymbol{\theta}) = [\boldsymbol{b}(\theta_1), \cdots, \boldsymbol{b}(\theta_r)]$ with $\boldsymbol{b}(\theta) = \mathrm{d}\boldsymbol{a}(\theta)/\mathrm{d}\theta$, $\boldsymbol{\Pi}_{\boldsymbol{F}}^{\perp} = \boldsymbol{I} - \boldsymbol{\Pi}_{\boldsymbol{F}}$ is the projection onto the orthogonal complementary space of $\boldsymbol{F}$, and

$$I_c = \pi \int_0^{\infty} \frac{(p_{\xi}'(\rho))^2}{p_{\xi}(\rho)} \rho \mathrm{d}\rho \tag{2.69}$$

with $\rho = |\xi|$ being the modulus of the complex variable ξ and $p_{\xi}'(\rho)$ the derivative of $p_{\xi}(\rho)$. The p.d.f. of the noise effects the CRB only through the scalar I_c. The CRBs of GMM and GGD noises can be numerically computed using (2.68) and (2.69).

Monte Carlo trials have been carried out to evaluate the performance of the DOA estimators. The DOAs of two uncorrelated sources are $\theta_1 = -8°$ and $\theta_2 = 10°$. This means that the target rank is $r = 2$. The numbers of sensors and snapshots are $m = 6$ and $n = 100$. The root mean square errors (RMSEs) of the subspace distance and DOA estimates are taken as the performance measures, which are defined as

$$\mathrm{RMSE}(\boldsymbol{U}) = \frac{1}{\|\boldsymbol{\Pi}_{\boldsymbol{F}}\|_{\mathrm{F}}} \sqrt{\frac{1}{N_m} \sum_{i=1}^{N_m} \|\boldsymbol{\Pi}_{\boldsymbol{U}_i} - \boldsymbol{\Pi}_{\boldsymbol{F}}\|_{\mathrm{F}}^2} \tag{2.70}$$

and

$$\mathrm{RMSE}\left(\widehat{\theta}_j\right) = \sqrt{\frac{1}{N_m} \sum_{i=1}^{N_m} \left(\widehat{\theta}_{j,i} - \theta_j\right)^2} \tag{2.71}$$

respectively, where N_m is the number of Monte Carlo trials, $\boldsymbol{U}_i$ and $\widehat{\theta}_{j,i}$ are the subspace and DOA estimates of the jth source in the ith trial, respectively. To avoid grid search over the spectrum, the root-MUSIC [113] is employed to calculate the DOA parameters after the signal subspace is obtained.

RMSEs of subspace DOA estimates versus SNR are studied under GMM and GGD noise in this experiment. At each SNR, 200 Monte Carlo trials are carried out. Figures 2.8 and 2.9 plot the RMSEs of subspace distance and DOA estimate of the first source versus SNR in GMM noise, respectively. While Figures 2.10 and 2.11 plot the results of in GMM noise. In addition, the CRBs for DOA estimate are also plotted for comparison.

As can be observed from Figures 2.8 to 2.11, the conventional MUSIC is not robust in the presence of impulsive noise. The ADMM using ℓ_1-subspace estimation has the best performance. The AM also exhibits good performance but it is inferior to the ADMM. MM and GZMNL also show good robustness to outlier but GZMNL suffers a performance saturation as the SNR increases. This is because it generally destroys the low-rank structure of the signal subspace, which leads to a performance saturation or

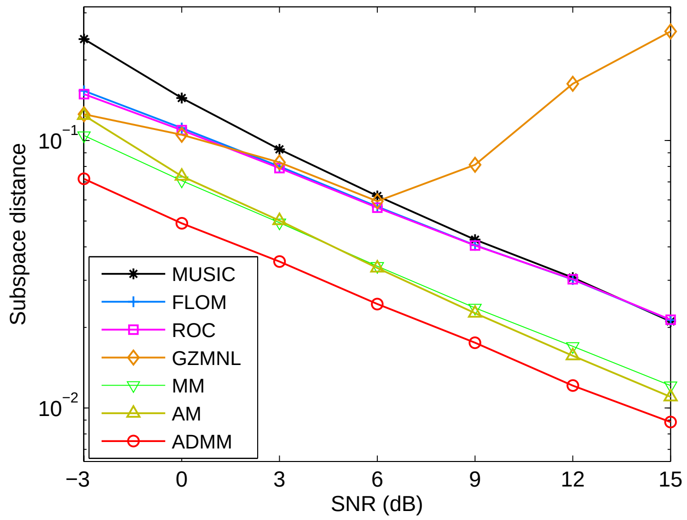


Figure 2.8: RMSE of subspace distance in GMM noise versus SNR.

even degradation [105]. Although the ROC and FLOM are better than the conventional MUSIC, they are inferior to the ADMM, AM, MM, and GZMNL. It has been analyzed that the FLOM does not have a satisfactory performance if the sample size is not large enough [102, 105].

2.7.4 Results of Image Demixing

The application to real-world image demixing [116] is investigated. Many natural and man-made images include highly regular textures, corresponding to low-rank structure. In texture impainting, the task is to demix the background texture which is sparsely occluded by untextured components. For the observed image, textured and untextured components are modeled as low-rank matrix and sparse outlier, respectively. After obtaining the low-rank component $\boldsymbol{UV}$ using the AM and ADMM, the outlier component is computed as $\boldsymbol{A} - \boldsymbol{UV}$. In the example herein, the PCA, RPCA, AM with $p = 1$, and ADMM are applied to an image of a chessboard with 377×370 pixels [116]. From Figure 2.12, we clearly see that the background of the chessboard has low-rank structure with rank $r = 2$ while the chessmen can be viewed as sparse outliers. The PCA cannot exactly separate chessboard and chessmen while the RPCA, AM, and ADMM perfectly demix them.

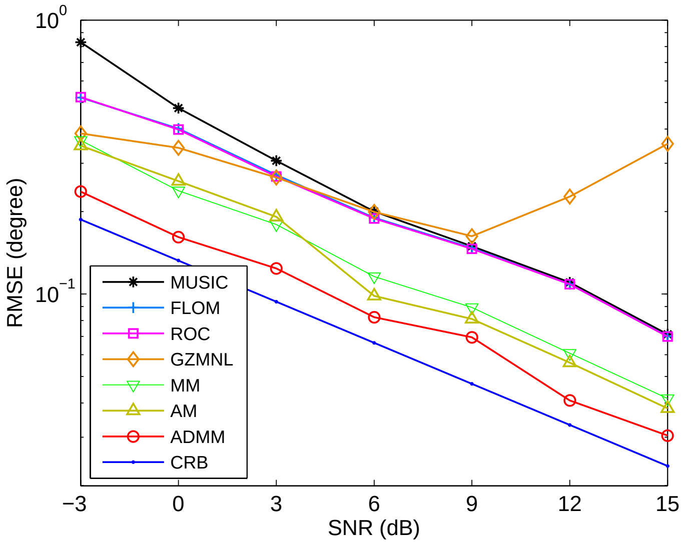


Figure 2.9: RMSE of DOA estimate of the first source in GMM noise versus SNR.

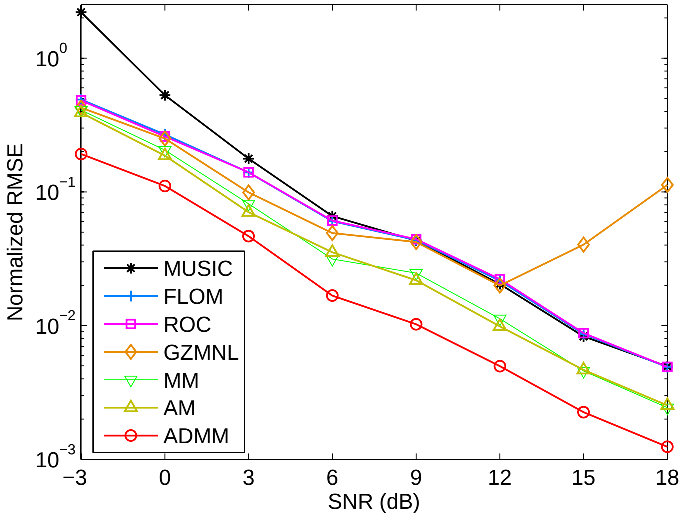


Figure 2.10: RMSE of subspace distance in GGD noise versus SNR.

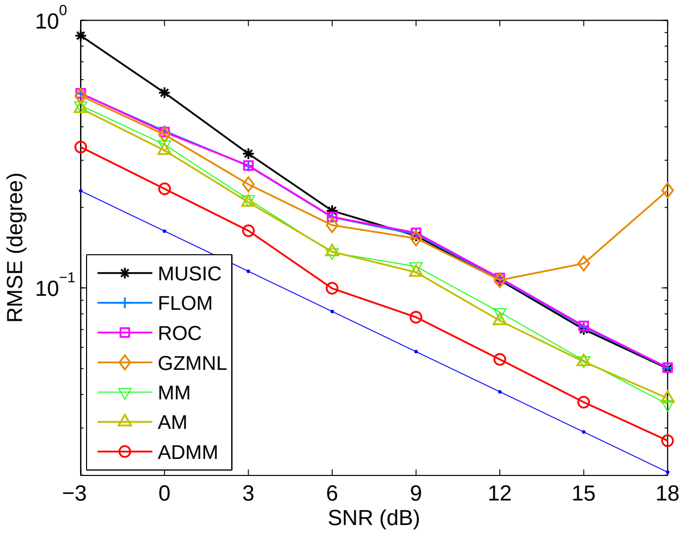


Figure 2.11: RMSE of DOA estimate of the first source in GGD noise versus SNR.

2.7.5 Results of Foreground Detection for Video Surveillance

We apply the robust low-rank approximation to foreground detection in video surveillance. In this application, the background scene is extracted from a number of video frames. Converting each frame of a video as a column of a matrix, the resultant matrix is of low-rank intrinsically due to the correlation between frames. In the presence of foreground objects especially in busy scenes, every frame may contain some anomalies. Foreground objects such as moving cars or walking pedestrians, generally occupy only a fraction of the image pixels, and thus may be treated as sparse outliers. If the background is invariant, the rank can be set as $r = 1$. Otherwise the rank may be selected slightly larger than one to accommodate small changes in the background.

The examples of video foreground detection (or equivalently background extraction) herein consider two video datasets available from CDNET, which is a video database for testing change detection algorithms [117]. Two examples in the video database, namely, the “backdoor” comprising a video sample of 2000 color frames with prevalent hard and soft shadow intermittent shades and the “streetlight” comprising a video sample of 3200 color frames containing background objects stopping for a short while and then moving away, are used in our experiment. In the examples herein, for both datasets, the first 200 frames of the video samples were selected and converted to grayscale versions. All frames of these examples have a size of 240×320, corresponding to 76800 pixels. Thus, the observed data matrix constructed from each video is $\mathbb{R}^{76800\times 200}$ where $m = 76800$ and $n = 200$. As the two videos have relatively static backgrounds,

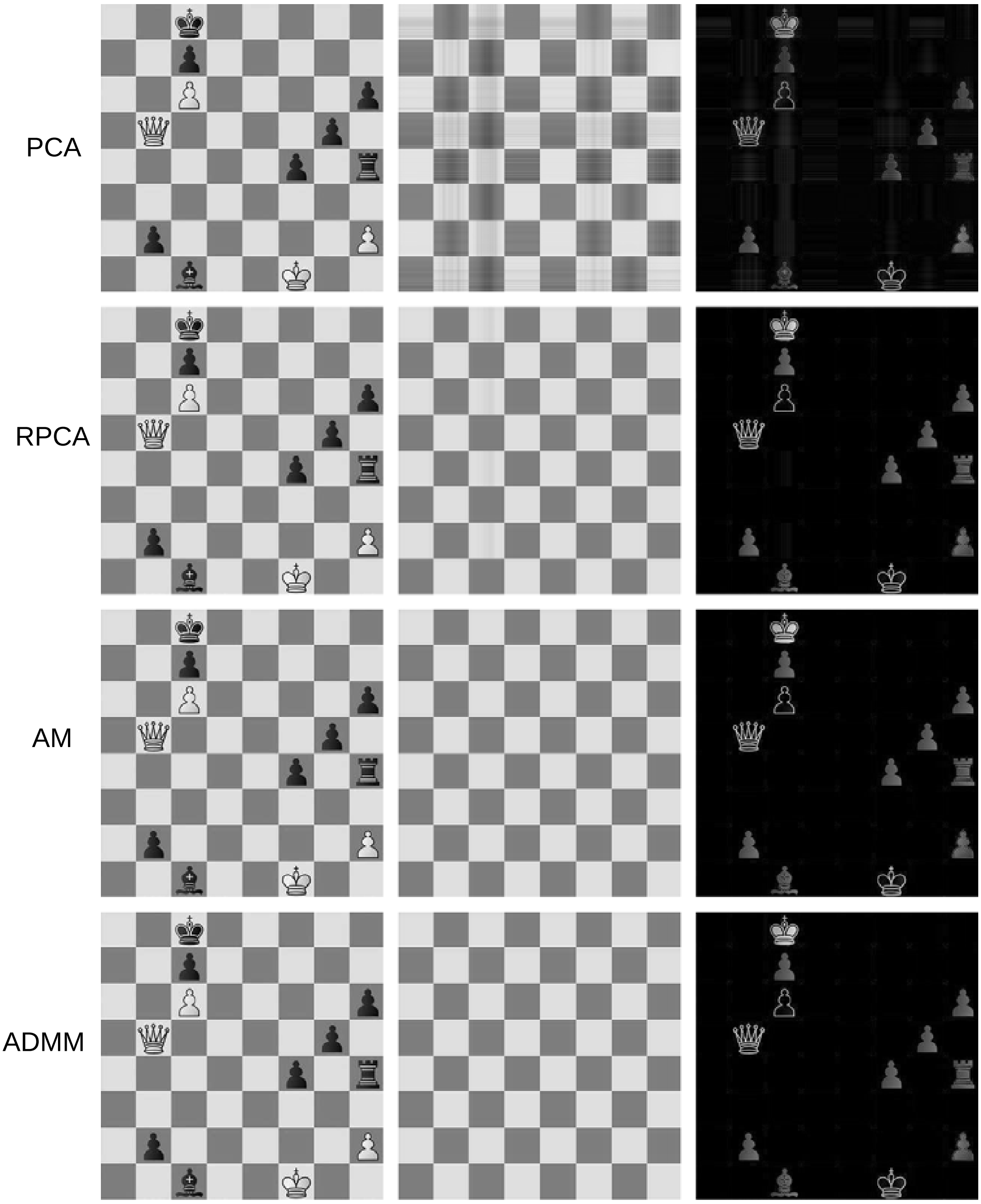


Figure 2.12: Demixing of chessboard and chessmen. The first, second, third, and fourth rows correspond to the results of the PCA, RPCA, AM, and ADMM. The first, second, and third columns show the images of mixture, background, and chessmen, respectively.

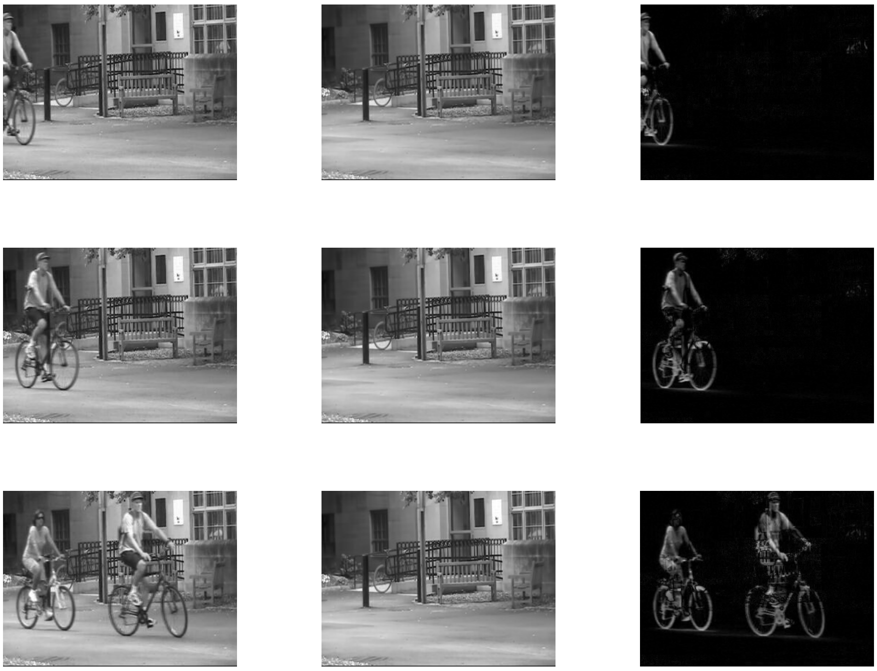


Figure 2.13: Foreground the background separation of the "backdoor" database for video surveillance. The first column is the original frames while the second and third columns are the background and foreground, respectively.

$r = 1$ is selected in both cases. Figures 2.13 and 2.14 show three representative frames and the corresponding separated results in the "backdoor" and "streetLight" datasets, respectively. It can be seen from these representative frames the ADMM successfully separates the foreground from the background.

2.8 Summary

The conventional tool for low-rank matrix approximation is not robust against outliers since it uses Frobenius norm minimization. In this chapter, we develop three algorithms, namely, the IR-SVD, AM, and ADMM for robust low-rank factorization based on minimization of the ℓ_p-norm of the residual. The ADMM is somewhat a bit better than the IR-SVD and AM with improvement of numerical effectiveness. Experimental results on random data, DOA estimation, image separation, and video surveillance verify the superior outlier-robustness of the three proposed algorithm.

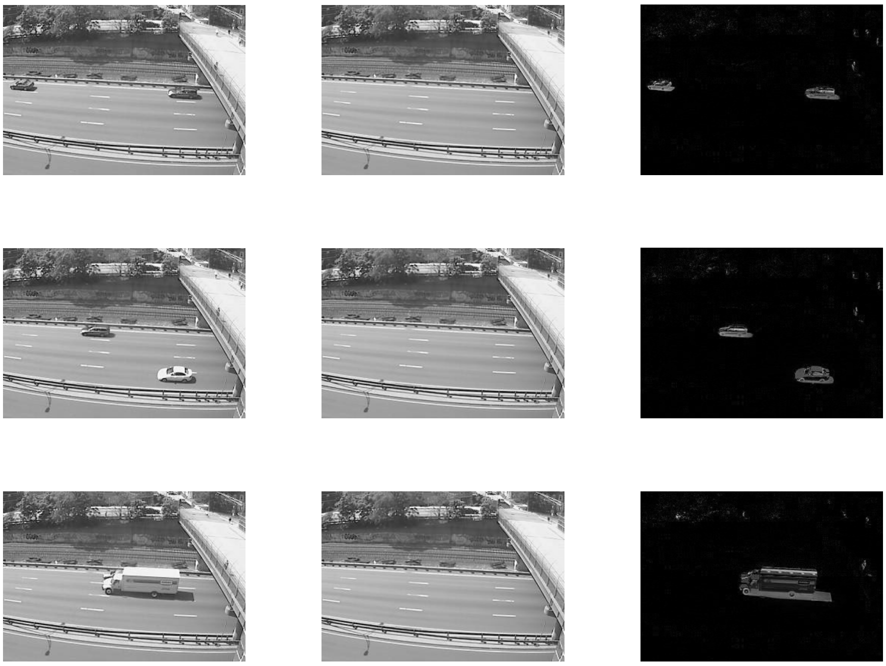


Figure 2.14: Foreground the background separation of the “streetLight” database for video surveillance. The first column is the original frames while the second and third columns are the background and foreground, respectively.

Chapter 3

Robust Low-Rank Matrix Completion in ℓ_p-Space

In Chapter 2, we discuss robust low-rank approximation for a single matrix where the entries are fully observed. In many applications, only partial entries of the matrix are available. In this chapter, we investigate low-rank matrix approximation with missing values. This problem is called as *matrix completion*. Most existing techniques for matrix completion assume Gaussian noise and thus they are not robust to outliers. In this paper, we devise two algorithms for robust matrix completion based on low-rank matrix factorization and minimizing the entry-wise ℓ_p-norm of the fitting error with $0 < p < 2$. The first method tackles the low-rank matrix factorization with missing data by iteratively solving $(m+n)$ linear ℓ_p-regression problems, where m and n are the numbers of rows and columns, respectively. The second applies the alternating direction method of multipliers (ADMM) in the ℓ_p-space. At each iteration of the ADMM, it requires performing a least squares (LS) matrix factorization and calculating the proximity operator of the pth power of the ℓ_p-norm. The LS factorization is efficiently solved using linear LS regression while the proximity operator has closed-form solution for $p = 1$ or can be obtained by root finding of a scalar nonlinear equation for other values of p. The two proposed algorithms have comparable recovery capability and computational complexity of $\mathcal{O}(K|\Omega|r^2)$, where $|\Omega|$ is the number of observed entries and K is a fixed constant of several hundreds to thousands and dimension-independent. It is demonstrated that they are superior to the state-of-the-art methods in terms of computational simplicity, statistical accuracy and outlier-robustness.

3.1 Problem Formulation and Preliminaries

Let $\boldsymbol{A}_\Omega \in \mathbb{R}^{m\times n}$ be a matrix with missing entries where Ω is a subset of the complete set of entries $[m]\times[n]$, with $[n]$ being the list $\{1,\cdots,n\}$. Throughout the chapter, the subscript $(\cdot)_\Omega$ denotes the projection on the known entries. The (i,j)th entry of $\boldsymbol{A}_\Omega$, denoted by $[\boldsymbol{A}_\Omega]_{ij}$, can be written as:

$$[\boldsymbol{A}_\Omega]_{ij} = \begin{cases} \boldsymbol{A}_{ij}, & \text{if } (i,j)\in\Omega \\ 0, & \text{otherwise.} \end{cases} \tag{3.1}$$

In addition, we use the lower-case bold letter $\boldsymbol{a}_\Omega \in \mathbb{R}^{|\Omega|}$ to represent the vector stacking all the observed entries of $\boldsymbol{A}_\Omega$ in a column-by-column manner, where $|\Omega|$ stands for the cardinality of Ω, that is, number of observed entries. As an illustration, suppose the original matrix is:

$$\boldsymbol{A} = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 6 \\ 3 & 6 & 9 \end{bmatrix}. \tag{3.2}$$

If only four elements are observed over $\Omega = \{(1,1),(2,1),(3,2),(1,3)\}$, we then have:

$$\boldsymbol{A}_\Omega = \begin{bmatrix} 1 & 0 & 3 \\ 2 & 0 & 0 \\ 0 & 6 & 0 \end{bmatrix} \tag{3.3}$$

with $\boldsymbol{a}_\Omega = [1,2,6,3]^{\mathrm{T}}$.

The task of matrix completion is to find a matrix $\boldsymbol{M} \in \mathbb{R}^{m\times n}$ given incomplete observations $\boldsymbol{A}_\Omega$ by incorporating the low-rank information. Mathematically, it is formulated as a rank minimization problem:

$$\begin{aligned} &\min_{\boldsymbol{M}} \ \operatorname{rank}(\boldsymbol{M}) \\ &\text{s.t. } \boldsymbol{M}_\Omega = \boldsymbol{A}_\Omega. \end{aligned} \tag{3.4}$$

That is, among all matrices consistent with the observed entries, we look for the one with minimum rank. However, (3.4) is NP-hard. A popular and practical solution is to replace the nonconvex rank by convex nuclear norm [9, 42–44], resulting in

$$\begin{aligned} &\min_{\boldsymbol{M}} \|\boldsymbol{M}\|_* \\ &\text{s.t. } \boldsymbol{M}_\Omega = \boldsymbol{A}_\Omega \end{aligned} \tag{3.5}$$

where the nuclear norm $\|\boldsymbol{M}\|_*$ equals the sum of singular values of $\boldsymbol{M}$. This convex relaxation is analogous to the relaxation of the intractable problem of ℓ_0-minimization to ℓ_1-minimization in sparse signal recovery [39]. In the presence of noise, (3.5) is modified as

$$\begin{aligned} &\min_{\boldsymbol{M}} \|\boldsymbol{M}\|_* \\ &\text{s.t. } \|\boldsymbol{M}_\Omega - \boldsymbol{A}_\Omega\|_{\mathrm{F}} \le \epsilon_{\mathrm{F}} \end{aligned} \tag{3.6}$$

where $\|\cdot\|_{\mathrm{F}}$ denotes the Frobenius norm of a matrix and $\epsilon_{\mathrm{F}} > 0$ is a tolerance parameter that controls the fitting error. By converting (3.5) and (3.6) into SDP [9, 42], they can be solved by interior-point methods [42, 44]. The complexity of the interior-point method for solving SDP is high. But there exists faster alternatives such as SVT [45], FPC [46] and proximal gradient descent [47]. Note that full SVD is still required for these faster methods.

In order to avoid SVD, matrix factorization has been exploited, corresponding to the following optimization:

$$\min_{\boldsymbol{U},\boldsymbol{V}} f_2(\boldsymbol{U},\boldsymbol{V}) := \|(\boldsymbol{U}\boldsymbol{V})_\Omega - \boldsymbol{A}_\Omega\|_{\mathrm{F}}^2 \tag{3.7}$$

where $\boldsymbol{U} \in \mathbb{R}^{m\times r}$ and $\boldsymbol{V} \in \mathbb{R}^{r\times n}$. After determining $\boldsymbol{U}$ and $\boldsymbol{V}$, the target matrix is obtained as $\boldsymbol{M} = \boldsymbol{U}\boldsymbol{V}$. Apparently, the low-rank property of $\boldsymbol{M}$ is automatically fulfilled. To handle (3.7), it can be relaxed as a bi-convex problem [64], which is then solved via alternating least squares. To be more specific, in the $(k+1)$th $(k = 0, 1, \cdots)$ iteration, $\boldsymbol{U}$ and $\boldsymbol{V}$ are alternately minimized according to

$$\begin{aligned} \boldsymbol{V}^{k+1} &= \arg\min_{\boldsymbol{V}} \|(\boldsymbol{U}^k\boldsymbol{V})_\Omega - \boldsymbol{A}_\Omega\|_{\mathrm{F}}^2 \\ \boldsymbol{U}^{k+1} &= \arg\min_{\boldsymbol{U}} \|(\boldsymbol{U}\boldsymbol{V}^{k+1})_\Omega - \boldsymbol{A}_\Omega\|_{\mathrm{F}}^2 \end{aligned} \tag{3.8}$$

where the algorithm is initialized with $\boldsymbol{U}^0$ and $\boldsymbol{U}^k$ represents the estimate of $\boldsymbol{U}$ at the kth iteration.

Although the formulations (3.6) and (3.7) work well in the presence of additive Gaussian disturbance [9], its performance can significantly degrade when $\boldsymbol{A}_\Omega$ contains outliers.

3.2 Iterative ℓ_p-Regression Algorithm

To achieve outlier resistance, we robustify (3.7) by replacing the Frobenius norm by the ℓ_p-norm where $0 < p < 2$, that is:

$$\min_{\boldsymbol{U},\boldsymbol{V}} f_p(\boldsymbol{U},\boldsymbol{V}) := \|(\boldsymbol{U}\boldsymbol{V})_\Omega - \boldsymbol{A}_\Omega\|_p^p, \quad 0 < p < 2 \tag{3.9}$$

where $\|\cdot\|_p$ denotes the element-wise ℓ_p-norm of a matrix, which has the form of:

$$\|\boldsymbol{E}_\Omega\|_p = \left(\sum_{(i,j)\in\Omega} |\boldsymbol{E}_{ij}|^p\right)^{1/p}. \tag{3.10}$$

where $\boldsymbol{E}_\Omega = (\boldsymbol{U}\boldsymbol{V})_\Omega - \boldsymbol{A}_\Omega$ with $\boldsymbol{E} = \boldsymbol{U}\boldsymbol{V} - \boldsymbol{A}$ being the error matrix. Note that (3.9) can be considered as a generalization of (3.7) because substituting $p = 2$ into the former reduces to the latter. For the special case with $p = 1$, (3.9) corresponds to the least absolute deviations (LAD), which was first proposed by Laplace [118] and has been widely used in statistics for robust estimation and regression [119, 120]. Different from the least squares (LS) using ℓ_2-minimization, the LAD aims at minimizing the sum of the absolute errors, i.e., the ℓ_1-norm of the residual. Furthermore, the nonconvex

ℓ_p-minimization of (3.9) is different from the robust low-rank matrix approximation discussed in Chapter 2, which addresses

$$\min_{\boldsymbol{U},\boldsymbol{V}} \|\boldsymbol{U}\boldsymbol{V}-\boldsymbol{A}\|_p^p \tag{3.11}$$

where there are no missing entries. While in matrix completion, we only have incomplete observations over Ω. It is clear that (3.9) and (3.11) are different. In this work, we devise two algorithms for solving (3.9) and the first one adopts the alternating minimization strategy:

$$\boldsymbol{V}^{k+1} = \arg\min_{\boldsymbol{V}} \|(\boldsymbol{U}^k\boldsymbol{V})_\Omega - \boldsymbol{A}_\Omega\|_p^p \tag{3.12}$$

$$\boldsymbol{U}^{k+1} = \arg\min_{\boldsymbol{U}} \|(\boldsymbol{U}\boldsymbol{V}^{k+1})_\Omega - \boldsymbol{A}_\Omega\|_p^p \tag{3.13}$$

which generalizes (3.8). We now focus on solving (3.12) for a fixed $\boldsymbol{U}$:

$$\min_{\boldsymbol{V}} f_p(\boldsymbol{V}) := \|(\boldsymbol{U}\boldsymbol{V})_\Omega - \boldsymbol{A}_\Omega\|_p^p \tag{3.14}$$

where the superscript $(\cdot)^k$ is dropped for notational simplicity. Denoting the ith row of $\boldsymbol{U}$ and the jth column of $\boldsymbol{V}$ as $\boldsymbol{u}_i^{\mathrm{T}}$ and $\boldsymbol{v}_j$, respectively, where $\boldsymbol{u}_i, \boldsymbol{v}_j \in \mathbb{R}^r$, $i = 1,\cdots,m$, $j = 1,\cdots,n$, (3.14) can be rewritten as

$$\min_{\boldsymbol{V}} f_p(\boldsymbol{V}) := \sum_{(i,j)\in\Omega} |\boldsymbol{u}_i^{\mathrm{T}}\boldsymbol{v}_j - \boldsymbol{A}_{ij}|^p. \tag{3.15}$$

Since $f_p(\boldsymbol{V})$ is decoupled with respect to $\boldsymbol{v}_j$, (3.15) is equivalent to solving the following n independent subproblems:

$$\min_{\boldsymbol{v}_j} f_p(\boldsymbol{v}_j) := \sum_{i\in\mathcal{I}_j} |\boldsymbol{u}_i^{\mathrm{T}}\boldsymbol{v}_j - \boldsymbol{A}_{ij}|^p,\ j = 1,\cdots,n \tag{3.16}$$

where $\mathcal{I}_j = \{i_1,\cdots,i_{|\mathcal{I}_j|}\} \subseteq \{1,\cdots,m\}$ denotes the set containing the row indices for the jth column in Ω. Here $|\mathcal{I}_j|$ stands for the cardinality of $\mathcal{I}_j$ and in general $|\mathcal{I}_j| > r$. As an illustration, we provide a simple example for determining $\mathcal{I}_j$ as follows. Consider $\boldsymbol{A}_\Omega \in \mathbb{R}^{4\times 3}$:

$$\boldsymbol{A}_\Omega = \begin{bmatrix} 0 & \times & 0 \\ \times & 0 & \times \\ 0 & \times & \times \\ \times & 0 & \times \end{bmatrix} \tag{3.17}$$

where the observed and missing entries are represented by × and 0, respectively. For $j = 1$, the $(2,1)$ and $(4,1)$ entries are observed. Thus we have $\mathcal{I}_1 = \{2,4\}$. It is easy to see that $\mathcal{I}_2 = \{1,3\}$ and $\mathcal{I}_3 = \{2,3,4\}$. Apparently, $\sum_{j=1}^n |\mathcal{I}_j| = |\Omega|$. Defining a matrix $\boldsymbol{U}_{\mathcal{I}_j} \in \mathbb{R}^{|\mathcal{I}_j| \times r}$ containing the $|\mathcal{I}_j|$ rows indexed by $\mathcal{I}_j$:

$$\boldsymbol{U}_{\mathcal{I}_j} = \begin{bmatrix} \boldsymbol{u}_{j_1}^{\mathrm{T}} \\ \vdots \\ \boldsymbol{u}_{j_{|\mathcal{I}_j|}}^{\mathrm{T}} \end{bmatrix} \tag{3.18}$$

and a vector $\boldsymbol{b}_{\mathcal{I}_j} = [\boldsymbol{A}_{j_1 j}, \cdots, \boldsymbol{A}_{j_{|\mathcal{I}_j|} j}]^{\mathrm{T}} \in \mathbb{R}^{|\mathcal{I}_j|}$, then (3.16) is compactly rewritten as

$$\min_{\boldsymbol{v}_j} f_p(\boldsymbol{v}_j) := \|\boldsymbol{U}_{\mathcal{I}_j} \boldsymbol{v}_j - \boldsymbol{b}_{\mathcal{I}_j}\|_p^p \tag{3.19}$$

which is a robust linear regression in ℓ_p-space. It is worth mentioning that for $p = 2$, (3.19) is an LS problem with solution being $\boldsymbol{v}_j = \boldsymbol{U}_{\mathcal{I}_j}^{\dagger} \boldsymbol{b}_{\mathcal{I}_j}$, and the corresponding computational complexity is $\mathcal{O}(|\mathcal{I}_j| r^2)$.

For $0 < p < 2$, the ℓ_p-regression of (3.19) can be efficiently solved by the iteratively reweighted least squares (IRLS) algorithm [76, 77] where global convergence can be achieved for the convex case of $p \geq 1$ while only a stationary point is obtained for the nonconvex case of $p < 1$. At the tth iteration[1], the IRLS solves the following weighted LS problem:

$$\boldsymbol{v}_j^{t+1} = \arg\min_{\boldsymbol{v}_j} \|\boldsymbol{W}^t (\boldsymbol{U}_{\mathcal{I}_j} \boldsymbol{v}_j - \boldsymbol{b}_{\mathcal{I}_j})\|^2 \tag{3.20}$$

where $\boldsymbol{W}^t = \mathrm{diag}\{w_1^t, \cdots, w_m^t\}$ is a diagonal weighting matrix with the ith diagonal element being

$$w_i^t = \frac{1}{(|\xi_i^t|^2 + \epsilon)^{\frac{1-p/2}{2}}}. \tag{3.21}$$

The ξ_i^t is the ith element of the residual vector $\boldsymbol{\xi}^t = \boldsymbol{U}_{\mathcal{I}_j} \boldsymbol{v}_j^t - \boldsymbol{b}_{\mathcal{I}_j}$ and $\epsilon > 0$ is a small positive parameter to avoid division by zero and ensure numerical stability, especially for $p \leq 1$. A typical value of ϵ is taken as $\epsilon = 100\epsilon_{\text{machine}}$ with $\epsilon_{\text{machine}}$ being the machine precision. Only one LS problem is required to solve in each IRLS iteration. Therefore, the complexity of ℓ_p-regression is $\mathcal{O}(|\mathcal{I}_j| r^2 N_{\text{IRLS}})$ where N_{IRLS} is the iteration number required for the IRLS algorithm to converge. Due to its fast convergence rate [76], N_{IRLS} will not be large, with a typical value of several tens, and is independent of the problem dimension. The total complexity for handling the n ℓ_p-regressions of (3.15) is $\mathcal{O}(|\Omega| r^2 N_{\text{IRLS}})$ due to $\sum_{j=1}^{n} |\mathcal{I}_j| = |\Omega|$.

Since (3.12) and (3.13) have the same structure, we solve (3.13) in the same manner. The ith row of $\boldsymbol{U}$ is updated by

$$\min_{\boldsymbol{u}_i^{\mathrm{T}}} \|\boldsymbol{u}_i^{\mathrm{T}} \boldsymbol{V}_{\mathcal{J}_i}^{k+1} - \boldsymbol{b}_{\mathcal{J}_i}^{\mathrm{T}}\|_p^p \tag{3.22}$$

---

[1] It should be pointed out that the iteration number t refers to IRLS iteration and should not be mixed up with the iteration number k. That is, k is the index of outer iteration while t is the index of inner iteration.

where $\mathcal{J}_i = \{j_1, \cdots, j_{|\mathcal{J}_i|}\} \subseteq \{1, \cdots, n\}$ is the set containing the column indices for the ith row in Ω. Employing (3.17) again, only the $(1,2)$th entry is observed for $i = 1$, and thus $\mathcal{J}_1 = \{2\}$. We also easily obtain $\mathcal{J}_2 = \{1,3\}$, $\mathcal{J}_3 = \{2,3\}$, and $\mathcal{J}_4 = \{1,3\}$. Here, $\boldsymbol{V}^{k+1}_{\mathcal{J}_i} \in \mathbb{R}^{r\times|\mathcal{J}_i|}$ contains the $|\mathcal{J}_i|$ columns indexed by $\mathcal{J}_i$ and $\boldsymbol{b}^{\mathrm{T}}_{\mathcal{J}_i} = [\boldsymbol{A}_{ii_1}, \cdots, \boldsymbol{A}_{ii_{|\mathcal{J}_i|}}]^{\mathrm{T}} \in \mathbb{R}^{|\mathcal{J}_i|}$. The involved complexity in (3.22) is $\mathcal{O}(|\mathcal{J}_i|r^2 N_{\mathrm{IRLS}})$ and hence the total complexity for solving the m ℓ_p-regressions of (3.22) is $\mathcal{O}(|\Omega|r^2 N_{\mathrm{IRLS}})$ because of $\sum_{i=1}^{m} |\mathcal{J}_i| = |\Omega|$.

The steps of the iterative ℓ_p-regression for matrix completion is summarized in Algorithm 3. Note that the complexity for a k-iteration is $\mathcal{O}(|\Omega|r^2 N_{\mathrm{IRLS}})$. For the special case when $p = 2$, Algorithm 3 reduces to solving the problem of (3.7). In this case, we have $N_{\mathrm{IRLS}} = 1$ and the complexity reduces to $\mathcal{O}(|\Omega|r^2)$ per k-iteration. In many practical applications, the number of observed entries is much smaller than the number of total entries, that is, $|\Omega| \ll mn$. Thus, the proposed algorithm becomes more computationally efficient as the percentage of the observations decreases. Now it is clear that the total complexity of the iterative ℓ_p-regression is $\mathcal{O}(|\Omega|r^2 N_{\mathrm{IRLS}} K_{\mathrm{reg}})$ where K_{reg} is the number of outer iterations, namely, the k-iteration. Empirically, a value of several tens for K_{reg} is sufficient for convergence. Finally, it is worth pointing out that the n problems of (3.19) and m problems of (3.22) are independent and hence can be realized in a parallel or distributed manner. As the number of processors increases, the complexity reduces.

Algorithm 3 Iterative ℓ_p-Regression for Robust Matrix Completion

```
Input: $\boldsymbol{A}_\Omega$, $\Omega$, and rank $r$
  Initialize: Randomly initialize $\boldsymbol{U}^0 \in \mathbb{R}^{m\times r}$
  Determine $\{\mathcal{I}_j\}_{j=1}^n$ and $\{\mathcal{J}_i\}_{i=1}^m$ according to $\Omega$.
  for $k = 0, 1, \cdots$ do
      // Fix $\boldsymbol{U}^k$, optimize $\boldsymbol{V}$
      for $j = 1, 2, \cdots, n$ do
          $\boldsymbol{v}_j^{k+1} \leftarrow \arg\min_{\boldsymbol{v}_j} \|\boldsymbol{U}^k_{\mathcal{I}_j}\boldsymbol{v}_j - \boldsymbol{b}_{\mathcal{I}_j}\|_p^p$
      end for
      // Fix $\boldsymbol{V}^{k+1}$, optimize $\boldsymbol{U}$
      for $i = 1, 2, \cdots, m$ do
          $(\boldsymbol{u}_i^{\mathrm{T}})^{k+1} \leftarrow \arg\min_{\boldsymbol{u}_i^{\mathrm{T}}} \|\boldsymbol{u}_i^{\mathrm{T}}\boldsymbol{V}^{k+1}_{\mathcal{J}_i} - \boldsymbol{b}^{\mathrm{T}}_{\mathcal{J}_i}\|_p^p$
      end for
      Stop if a termination condition is satisfied.
  end for
Output: $\boldsymbol{M} = \boldsymbol{U}^{k+1}\boldsymbol{V}^{k+1}$
```

We give a short remark on the convergence of the iterative ℓ_p-regression. Since Algorithm 3 monotonically non-increases a below-bounded objective function for all $p \leq 2$, the sequence $\{f_p(\boldsymbol{U}^k, \boldsymbol{V}^k)\}$ converges to a limit point. However, it does not imply that $\{(\boldsymbol{U}^k, \boldsymbol{V}^k)\}$ converges. If we further assume that either (3.12) or (3.13) has a unique minimizer, then Algorithm 3 converges to a stationary point based on the convergence result of a block coordinate descent method using the cyclic rule in [121], which contains Algorithm 3 as a special case.

3.3 ADMM for Robust Matrix Completion

In this section, we apply the ADMM to solve (3.9). Note that the ADMM developed here is different from that of Chapter 2. The former is applicable to the case with missing values while the latter is for low-rank approximation with fully observed entries.

3.3.1 Framework of ADMM

With the use of $\boldsymbol{E}_\Omega = (\boldsymbol{U}\boldsymbol{V})_\Omega - \boldsymbol{A}_\Omega$, (3.9) is equivalent to a linearly constrained optimization problem:

$$\begin{aligned} &\min_{\boldsymbol{U},\boldsymbol{V},\boldsymbol{E}_\Omega} \|\boldsymbol{E}_\Omega\|_p^p \\ &\text{s.t. } \boldsymbol{E}_\Omega = (\boldsymbol{U}\boldsymbol{V})_\Omega - \boldsymbol{A}_\Omega. \end{aligned} \tag{3.23}$$

where $\boldsymbol{E}_\Omega$ is treated as decision variables that are independent of $\boldsymbol{U}$ and $\boldsymbol{V}$. Note that $[\boldsymbol{E}_\Omega]_{ij} = 0$ if $(i,j) \notin \Omega$. The augmented Lagrangian of (3.23) is

$$\begin{aligned} \mathcal{L}_\mu(\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{E}_\Omega, \boldsymbol{\Lambda}_\Omega) =& \|\boldsymbol{E}_\Omega\|_p^p + \langle \boldsymbol{\Lambda}_\Omega, (\boldsymbol{U}\boldsymbol{V})_\Omega - \boldsymbol{E}_\Omega - \boldsymbol{A}_\Omega \rangle \\ &+ \frac{\mu}{2}\|(\boldsymbol{U}\boldsymbol{V})_\Omega - \boldsymbol{E}_\Omega - \boldsymbol{A}_\Omega\|_{\mathrm{F}}^2 \end{aligned} \tag{3.24}$$

where $\boldsymbol{\Lambda}_\Omega \in \mathbb{R}^{m\times n}$ with $[\boldsymbol{\Lambda}_\Omega]_{ij} = 0$ for $(i,j) \notin \Omega$ contains $|\Omega|$ Lagrange multipliers (dual variables), $\langle \boldsymbol{A}, \boldsymbol{B} \rangle = \sum_{(i,j)} \boldsymbol{A}_{ij}\boldsymbol{B}_{ij}$ represents the inner product of two matrices $\boldsymbol{A}$ and $\boldsymbol{B}$, and $\mu > 0$ is the penalty parameter. The augmented Lagrangian reduces to the unaugmented one if $\mu = 0$. If the objective function is closed, proper and convex, and the unaugmented Lagrangian $\mathcal{L}_0$ has a saddle point, then the iterates approach feasibility and the objective function of the iterates approaches the optimal value [81]. However, the objective function of our problem is nonconvex. The theoretical proof of the convergence of the nonconvex ADMM is very challenging and remains an open problem. We give a brief discussion on this issue at the end of Section 3.3.3. Empirically, numerical examples [81] demonstrate that the selection of μ is flexible. We can use a fixed appropriate positive constant for μ or properly adapt the penalty parameter at each iteration for convergence speedup [82, 83]. In the simulations, we simply use $\mu = 5$ and it is observed that this value always makes the ADMM converge. The Lagrange multiplier method solves (3.23) by finding a saddle point of the augmented Lagrangian

$$\max_{\boldsymbol{\Lambda}_\Omega} \min_{\boldsymbol{U},\boldsymbol{V},\boldsymbol{E}_\Omega} \mathcal{L}_\mu(\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{E}_\Omega, \boldsymbol{\Lambda}_\Omega). \tag{3.25}$$

The ADMM uses the following iterative steps:

$$(\boldsymbol{U}^{k+1}, \boldsymbol{V}^{k+1}) = \arg\min_{\boldsymbol{U},\boldsymbol{V}} \mathcal{L}_\mu(\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{E}_\Omega^k, \boldsymbol{\Lambda}_\Omega^k) \tag{3.26}$$

$$\boldsymbol{E}_\Omega^{k+1} = \arg\min_{\boldsymbol{E}_\Omega} \mathcal{L}_\mu(\boldsymbol{U}^{k+1}, \boldsymbol{V}^{k+1}, \boldsymbol{E}_\Omega, \boldsymbol{\Lambda}_\Omega^k) \tag{3.27}$$

$$\boldsymbol{\Lambda}_\Omega^{k+1} = \boldsymbol{\Lambda}_\Omega^k + \mu\big((\boldsymbol{U}^{k+1}\boldsymbol{V}^{k+1})_\Omega - \boldsymbol{E}_\Omega^{k+1} - \boldsymbol{A}_\Omega\big) \tag{3.28}$$

to calculate the saddle point in (3.25), where $(\boldsymbol{U}^k, \boldsymbol{V}^k, \boldsymbol{E}_\Omega^k, \boldsymbol{\Lambda}_\Omega^k)$ denotes the result at the kth iteration. Several remarks and explanations on the three subproblems (3.26), (3.27), and (3.28) are given as follows.

Since the gradient of $\mathcal{L}_\mu(\boldsymbol{U}^{k+1}, \boldsymbol{V}^{k+1}, \boldsymbol{E}_\Omega^{k+1}, \boldsymbol{\Lambda}_\Omega)$ with respect to $\boldsymbol{\Lambda}_\Omega$ is

$$\frac{\partial \mathcal{L}_\mu(\boldsymbol{U}^{k+1}, \boldsymbol{V}^{k+1}, \boldsymbol{E}_\Omega^{k+1}, \boldsymbol{\Lambda}_\Omega)}{\partial \boldsymbol{\Lambda}_\Omega} = (\boldsymbol{U}^{k+1}\boldsymbol{V}^{k+1})_\Omega - \boldsymbol{E}_\Omega^{k+1} - \boldsymbol{A}_\Omega \tag{3.29}$$

we can see that (3.28) adopts a gradient ascent with a step size μ to update the dual variable $\boldsymbol{\Lambda}_\Omega$. ADMM updates $(\boldsymbol{U}, \boldsymbol{V})$ and $\boldsymbol{E}_\Omega$ in an alternating or sequential fashion to circumvent the difficulty in jointly minimizing with respect to the two primal blocks. Noting that (3.26) minimizes $(\boldsymbol{U}, \boldsymbol{V})$ simultaneously, (3.26)–(3.28) correspond to a two-block ADMM where the blocks refer to $(\boldsymbol{U}, \boldsymbol{V})$ and $\boldsymbol{E}_\Omega$, and are not of three blocks. It has been observed that updating more than two blocks may result in divergence of the ADMM [84]. Nevertheless, the divergence caused by multi-block update will not happen to the proposed ADMM since it is a two-block one.

By ignoring the constant term independent of $(\boldsymbol{U}, \boldsymbol{V})$, we derive that the subproblem (3.26) is equivalent to the following Frobenius norm minimization problem:

$$\min_{\boldsymbol{U},\boldsymbol{V}} \left\| (\boldsymbol{U}\boldsymbol{V})_\Omega - \left(\boldsymbol{E}_\Omega^k - \frac{\boldsymbol{\Lambda}_\Omega^k}{\mu} + \boldsymbol{A}_\Omega \right) \right\|_{\mathrm{F}}^2 \tag{3.30}$$

which can be solved by the iterative ℓ_2-regression, namely, Algorithm 3 with $p = 2$, with a complexity bound of $\mathcal{O}(K_{\ell_2}|\Omega|r^2)$. Here, K_{ℓ_2} is the iteration number for Algorithm 3 to converge at $p = 2$.

On the other hand, the subproblem (3.27) is concisely simplified as

$$\min_{\boldsymbol{E}_\Omega} \frac{1}{2} \left\| \boldsymbol{E}_\Omega - \boldsymbol{Y}_\Omega^k \right\|_{\mathrm{F}}^2 + \frac{1}{\mu} \|\boldsymbol{E}_\Omega\|_p^p \tag{3.31}$$

where

$$\boldsymbol{Y}_\Omega^k = (\boldsymbol{U}^{k+1}\boldsymbol{V}^{k+1})_\Omega + \frac{\boldsymbol{\Lambda}_\Omega^k}{\mu} - \boldsymbol{A}_\Omega. \tag{3.32}$$

We only need to consider the entries indexed by Ω because other entries of $\boldsymbol{E}_\Omega$ and $\boldsymbol{Y}_\Omega^k$ which are not in Ω are zero. Define $\boldsymbol{e}_\Omega$, $\boldsymbol{y}_\Omega^k$, $\boldsymbol{\lambda}_\Omega^k$, and $\boldsymbol{t}_\Omega^k \in \mathbb{R}^{|\Omega|}$ as the vectors that contain the observed entries in $\boldsymbol{E}_\Omega$, $\boldsymbol{Y}_\Omega^k$, $\boldsymbol{\Lambda}_\Omega^k$, and $(\boldsymbol{U}^k\boldsymbol{V}^k)_\Omega$, respectively, in a column-by-column manner. Apparently, (3.31) is equivalent to the vector optimization problem:

$$\min_{\boldsymbol{e}_\Omega} \frac{1}{2}\left\|\boldsymbol{e}_\Omega - \boldsymbol{y}_\Omega^k\right\|^2 + \frac{1}{\mu}\|\boldsymbol{e}_\Omega\|_p^p \tag{3.33}$$

whose solution defines the proximity operator [85] of the pth power of ℓ_p-norm, which is written as

$$\boldsymbol{e}_\Omega^{k+1} = \operatorname{prox}_{1/\mu}(\boldsymbol{y}_\Omega^k). \tag{3.34}$$

After obtaining $\boldsymbol{e}_\Omega^{k+1}$, $\boldsymbol{E}_\Omega^{k+1}$ is then determined. We will address computing this proximity operator shortly. For (3.28), its equivalent form in terms of vectors is:

$$\boldsymbol{\lambda}_\Omega^{k+1} = \boldsymbol{\lambda}_\Omega^k + \mu\big(\boldsymbol{t}_\Omega^{k+1} - \boldsymbol{e}_\Omega^{k+1} - \boldsymbol{a}_\Omega\big) \tag{3.35}$$

That is, the operations are now in terms of vectors but not matrices, and its complexity is $\mathcal{O}(|\Omega|)$. Also, at each iteration, we just need to compute $(\boldsymbol{U}\boldsymbol{V})_\Omega$ instead of $\boldsymbol{U}\boldsymbol{V}$, whose complexity is $\mathcal{O}(|\Omega|r)$ because only $|\Omega|$ inner products $\{\boldsymbol{u}_i^{\mathrm{T}}\boldsymbol{v}_j\}_{(i,j)\in\Omega}$ are calculated.

3.3.2 Proximity Operator of pth Power of ℓ_p-Norm

In this section, the proximity operator is determined. First, we rewrite (3.33) as:

$$\min_{\boldsymbol{e}\in\mathbb{R}^{|\Omega|}} \frac{1}{2}\|\boldsymbol{e} - \boldsymbol{y}\|^2 + \frac{1}{\mu}\|\boldsymbol{e}\|_p^p \tag{3.36}$$

where the subscripts and superscripts are ignored for notational simplicity. Denote e_i and y_i, $i = 1, \cdots, |\Omega|$, as the ith entry of $\boldsymbol{e}$ and $\boldsymbol{y}$, respectively. As (3.36) is separable, it can be decomposed into $|\Omega|$ independent scalar problems:

$$\min_{e_i\in\mathbb{R}} \; g(e_i) := \frac{1}{2}(e_i - y_i)^2 + \frac{1}{\mu}|e_i|^p, \; i = 1, \cdots, |\Omega|. \tag{3.37}$$

The closed-form solution of (3.37) for $p = 1$ is

$$e_i^\star = \operatorname{sgn}(y_i)\max(|y_i| - 1/\mu, 0) \tag{3.38}$$

which is known as the soft-thresholding operator [86, 87] and is easily computed with a marginal complexity of $\mathcal{O}(|\Omega|)$.

When the noise is very impulsive, the value of $p < 1$ may be required. The scalar minimization problem of (3.37) with $p < 1$ has already been solved recently in [48, 52, 123], whose solution is:

$$e_i^\star = \begin{cases} 0, & \text{if } |y_i| \le \tau \\ \arg\min\limits_{e_i \in \{0, t_i\}} g(e_i), & \text{if } |y_i| > \tau \end{cases} \tag{3.39}$$

where

$$\tau = \left(\frac{p(1-p)}{\mu}\right)^{\frac{1}{2-p}} + \frac{p}{\mu}\left(\frac{p(1-p)}{\mu}\right)^{\frac{p-1}{2-p}} \tag{3.40}$$

is the threshold and $t_i = \operatorname{sgn}(y_i) r_i$ with r_i being the unique root of the nonlinear equation:

$$h(\theta) := \theta + \frac{p}{\mu}\theta^{p-1} - |y_i| = 0 \tag{3.41}$$

in the interval $\left[(p(1-p)/\mu)^{\frac{1}{2-p}}, |y_i|\right]$ where the bisection method [122] can be applied. Although computing the proximity operator for $p < 1$ still has a complexity of $\mathcal{O}(|\Omega|)$, it is more complicated than $p = 1$ because there is no closed-form solution. On the other hand, the case of $p \in (1, 2)$ is not difficult to solve since (3.37) is a scalar convex problem but it also requires an iterative procedure for numerical calculation. For the purpose of completeness, we present the solver of (3.37) for $p \in (1, 2)$ since this has not been addressed. Obviously, if $y_i \ge 0$, the minimizer $e_i^\star \ge 0$. Otherwise, $e_i^\star < 0$. That is to say, we only need to consider minimizing $g(e_i)$ in $[0, \infty)$ if $y_i \ge 0$. The minimizer is either the stationary point satisfying the nonlinear equation

$$g'(e_i) = e_i - y_i + \frac{p}{\mu} e_i^{p-1} = 0 \tag{3.42}$$

or the boundary point 0. Due to $g'(0) = -y_i \le 0$ and $g'(y_i) = p y_i^{p-1}/\mu \ge 0$, i.e., $g'(0)g'(y_i) \le 0$, there exists a root in $[0, y_i]$ for the equation $g'(e_i) = 0$. Moreover, $g''(e_i) = 1 + \frac{p(p-1)}{\mu} e_i^{p-2} > 0$ holds for all $e_i \ge 0$, implying that $g'(e_i)$ monotonically increases in $[0, +\infty)$. Thus, the positive root of $g'(e_i) = 0$ in $[0, y_i]$ is unique, which is denoted as r_i^+. This root can be quickly found using the bisection or secant method with a complexity of $\mathcal{O}(1)$ [122]. After obtaining r_i^+, the minimizer in $[0, -\infty)$ is $e_i^\star = \arg\min\{g(0), g(r_i^+)\}$.

Similarly, we only need to minimize $g(e_i)$ in $(-\infty, 0]$ if $y_i < 0$. The minimizer is either the stationary point fulfilling

$$g'(e_i) = e_i - y_i - \frac{p}{\mu}(-e_i)^{p-1} = 0 \tag{3.43}$$

or the boundary point 0. Since $g'(y_i) = -p(-y_i)^{p-1}/\mu \le 0$ and $g'(0) = -y_i \ge 0$, namely., $g'(0)g'(y_i) \le 0$, $g'(e_i) = 0$ has a root in $[y_i, 0]$. Noting that $g''(e_i) = 1 +$

$\frac{p(p-1)}{\mu}(-e_i)^{p-2} > 0$ holds for all $e_i \leq 0$, $g'(e_i)$ monotonically increases in $(-\infty, 0]$. Then, the negative root of $g'(e_i) = 0$ in $[y_i, 0]$, which is denoted as r_i^-, is unique and can be solved easily. Once r_i^- is obtained, the minimizer in $(-\infty, 0]$ is $e_i^\star = \arg\min\{g(0), g(r_i^-)\}$. The solution of (3.37) for $p \in (1, 2)$ is compactly written as

$$e_i^\star = \begin{cases} \arg\min\{g(0), g(r_i^+)\}, & \text{if } y_i \geq 0 \\ \arg\min\{g(0), g(r_i^-)\}, & \text{if } y_i < 0. \end{cases} \tag{3.44}$$

Again, calculating the proximity operator for $1 < p < 2$ has a complexity of $\mathcal{O}(|\Omega|)$ although an iterative procedure for root finding is required. Nevertheless, the choice of $p = 1$ is more robust than employing $p \in (1, 2)$ and is computationally simpler. In the case of very impulsive noise, $p < 1$ will be adopted.

3.3.3 Summary of ADMM

The steps of ADMM for robust matrix completion are summarized in Algorithm 4. The ℓ_2-norm of the residual, that is, $\|\boldsymbol{t}_\Omega^k - \boldsymbol{e}_\Omega^k - \boldsymbol{a}_\Omega\|$ is used to check for convergence. Specifically, the iteration is terminated when

$$\|\boldsymbol{t}_\Omega^k - \boldsymbol{e}_\Omega^k - \boldsymbol{a}_\Omega\| < \delta \tag{3.45}$$

where $\delta > 0$ is a small tolerance parameter.

Algorithm 4 ADMM for Robust Matrix Completion

```
Input: $\boldsymbol{A}_\Omega$, $\Omega$, and rank $r$
  Initialize: $\boldsymbol{e}^0 = \boldsymbol{0}$ and $\boldsymbol{\lambda}^0 = \boldsymbol{0}$
  for $k = 0, 1, \cdots$ do
      1) Solve LS matrix factorization
         $(\boldsymbol{U}^{k+1}, \boldsymbol{V}^{k+1}) = \arg\min_{\boldsymbol{U},\boldsymbol{V}} \left\|(\boldsymbol{U}\boldsymbol{V})_\Omega - \left(\boldsymbol{E}_\Omega^k - \boldsymbol{\Lambda}_\Omega^k/\mu + \boldsymbol{A}_\Omega\right)\right\|_F^2$
         using Algorithm 3 with $p = 2$.
      2) Compute $\boldsymbol{Y}_\Omega^k = (\boldsymbol{U}^{k+1}\boldsymbol{V}^{k+1})_\Omega + \boldsymbol{\Lambda}_\Omega^k/\mu - \boldsymbol{A}_\Omega$ and form $\boldsymbol{y}_\Omega^k$ and $\boldsymbol{t}_\Omega^{k+1} \leftarrow (\boldsymbol{U}^{k+1}\boldsymbol{V}^{k+1})_\Omega$.
      3) $\boldsymbol{e}_\Omega^{k+1} \leftarrow \text{prox}_{1/\mu}(\boldsymbol{y}_\Omega^k)$
      4) $\boldsymbol{\lambda}_\Omega^{k+1} \leftarrow \boldsymbol{\lambda}_\Omega^k + \mu\left(\boldsymbol{t}_\Omega^{k+1} - \boldsymbol{e}_\Omega^{k+1} - \boldsymbol{a}_\Omega\right)$
      Stop if a termination condition is satisfied.
  end for
Output: $\boldsymbol{M} = \boldsymbol{U}^{k+1}\boldsymbol{V}^{k+1}$
```

The dominant complexity of the ADMM is $\mathcal{O}(|\Omega|r^2 K_{\ell_2} K_{\text{ADMM}})$ where K_{ADMM} is the number of outer iterations of the ADMM, namely, the k-iteration. Empirically, a value of several tens for K_{ADMM} will result in an accurate estimation.

All values of $p > 0$, including the nonconvex and nonsmooth case with $p < 1$, can be set for the ADMM of this chapter. However, the ADMM of Chapter 2 just considers $p = 1$ because the proximity operator of the ℓ_p-norm with $p < 1$ for complex variables is difficult to calculate.

It should be pointed out that our ADMM is different from the ALM of [48] for matrix completion that solves

$$\min_{\boldsymbol{M}} \|\boldsymbol{M}_\Omega - \boldsymbol{A}_\Omega\|_p^p + \gamma \|\boldsymbol{M}\|_{S_p}^p \tag{3.46}$$

where $\gamma > 0$ is the regularization parameter and $\|\boldsymbol{M}\|_{S_p}$ is the Schatten p-norm, which equals the ℓ_p-norm of the vector containing all singular values of $\boldsymbol{M}$. As $p \to 0$, $\|\boldsymbol{M}\|_{S_p}$ approaches the rank of $\boldsymbol{M}$. Therefore, the Schatten p-norm regularization with $p \leq 1$ can be employed to find a low-rank solution. Especially, when $p = 1$, (3.46) is a convex program because $\|\boldsymbol{M}\|_{S_p}$ is the nuclear norm. In [48], the ALM is applied to solve (3.46), in which the full SVD of a $m \times n$ matrix is computed. Thus, the complexity of the ALM [48] is $\mathcal{O}(m^2 n)$ per iteration, assuming that $m \geq n$ without loss of generality.

The proposed method is also different from the RPCA that models the observed matrix as the sum of a low-rank matrix $\boldsymbol{L}$ and a sparse outlier matrix $\boldsymbol{S}$. When partial observations are available, the RPCA can be applied for matrix completion by solving the minimization problem

$$\begin{aligned} &\min_{\boldsymbol{M},\boldsymbol{S}} \|\boldsymbol{L}\|_* + \alpha \|\boldsymbol{S}\|_1 \\ &\text{s.t. } [\boldsymbol{L} + \boldsymbol{S}]_\Omega = \boldsymbol{A}_\Omega \end{aligned} \tag{3.47}$$

where $\alpha > 0$ is the regularization parameter that needs to estimate in practice. Although (3.47) is a convex optimization and the global minimum is guaranteed, it has a high computational cost even fast algorithms are employed because the full SVD is required [32, 33].

In a unified manner, the total complexity of the iterative ℓ_p-regression and ADMM can be written as $\mathcal{O}(K|\Omega|r^2)$ where

$$K = \begin{cases} N_{\text{IRLS}} K_{\text{reg}}, & \text{for } \ell_p\text{–regression} \\ K_{\ell_2} K_{\text{ADMM}}, & \text{for ADMM.} \end{cases} \tag{3.48}$$

The magnitude-of-order of K corresponds to several hundreds to thousands because N_{IRLS}, K_{reg}, K_{ℓ_2}, and K_{ADMM}, are of several tens.

The convergence of the two-block ADMM has only been proved for convex optimization [81]. Although the convergence of the ADMM for a class of nonconvex and nonsmooth optimization problems, including the ℓ_p-regularization with $p < 1$, has been established very recently in [124], the corresponding results are not applicable to our problem. The first reason is that the ℓ_p-norm appears as a regularization term to promote sparsity in [124] while our problem minimizes the ℓ_p-norm of the fitting error. This results in that the mathematical formulations of [124] and our problem are different. The second reason is that the nonconvexity of our problem is not only due to the ℓ_p-norm with $p < 1$ but also induced by the matrix product $\boldsymbol{UV}$. These two reasons make the theoretical proof of the convergence of the proposed ADMM challenging. It remains an open problem for future research. Although the convergence is not proved theoretically, we observe that the proposed ADMM always converges in the simulations. Thus, it is deemed that the proposed ADMM is empirically convergent in practice.

3.3.4 Algorithmic Parameter Selection

There are two parameters of the proposed algorithms, namely, the rank r and p. We discuss how to appropriately select them.

If the true rank is unknown, it needs to be estimated. Determining the rank is a model selection problem [125]. However, conventional model selection methods such as Akaike information criterion and minimum description length [125] are not applicable because there are missing data and outliers in our problem. Denoting the estimate for a given r as $\widehat{\boldsymbol{M}}(r)$, the optimal r aims at minimizing the estimation error

$$\min_{r \in \mathbb{Z}^+} \|\widehat{\boldsymbol{M}}(r) - \boldsymbol{A}\|_{\mathrm{F}}^2 \tag{3.49}$$

where $\mathbb{Z}^+$ is the set of positive integers. However, we cannot obtain the optimal r from (3.49) because $\boldsymbol{A}$ is not available.

In this chapter, we estimate the rank by cross-validation [23, 126]. Specifically, the observation set Ω is divided into two disjoint subsets Ω_1 and Ω_2 such that $\Omega_1 \cup \Omega_2 = \Omega$. In cross-validation, we just randomly select a portion of the observed entries, i.e., $\boldsymbol{A}_{\Omega_1}$, as the training data for matrix completion. The portion of training data $|\Omega_1|/|\Omega|$ can be set to 95%. For a given rank, matrix completion is performed based on $\boldsymbol{A}_{\Omega_1}$. We then compute the mean prediction error on the testing data $\boldsymbol{A}_{\Omega_2}$ based on multiple random divisions of Ω_1 and Ω_2. The rank is chosen as the one which corresponds to the smallest prediction error. Suppose that L random trials are carried out for calculating the prediction error. In the lth trial, the two sets are randomly generated, which are denoted as Ω_1^l and Ω_2^l. A matrix completion algorithm using partial noisy observations $\boldsymbol{A}_{\Omega_1^l}$ and rank r gives the result $\widehat{\boldsymbol{M}}(r)$. Since $\boldsymbol{A}$ is unknown, we cannot calculate the estimation error of (3.49). Instead, the prediction error of the testing data $\boldsymbol{A}_{\Omega_2}$ is evaluated. That is, the rank is estimated by minimizing the following root mean square prediction error (RMSPE)

$$\widehat{r} = \arg\min_{r\in\mathbb{Z}^+} \sum_{l=1}^{L} \frac{\left\| [\widehat{\boldsymbol{M}}(r)]_{\Omega_2^l} - \boldsymbol{A}_{\Omega_2^l} \right\|_{\mathrm{F}}^2}{\left\| \boldsymbol{A}_{\Omega_2^l} \right\|_{\mathrm{F}}^2} \tag{3.50}$$

or mean absolute prediction error (MAPE)

$$\widehat{r} = \arg\min_{r\in\mathbb{Z}^+} \sum_{l=1}^{L} \frac{\left\| [\widehat{\boldsymbol{M}}(r)]_{\Omega_2^l} - \boldsymbol{A}_{\Omega_2^l} \right\|_1}{\left\| \boldsymbol{A}_{\Omega_2^l} \right\|_1}. \tag{3.51}$$

The reason why we also adopt the MAPE is that $\boldsymbol{A}_{\Omega_2}$ can contain outliers and the ℓ_1-norm is a more outlier-robust distance measure. Simulation results on the choice of r with outliers are provided in Section 3.4.

On the other hand, the optimal choice of p is case-dependent. It relies on the statistical properties of the noise. As mentioned in Section II, $p = 2$ is optimal for Gaussian noise. In the presence of impulsive noise or outliers, $p < 2$ will bring a better performance. Consider a special case where the noise satisfies a zero-mean generalized Gaussian distribution (GGD) [110,115], whose probability density function (p.d.f.) with variance σ_v^2 is

$$p_v(v) = \frac{\beta\Gamma(4/\beta)}{2\pi\sigma_v^2\Gamma^2(2/\beta)} \exp\left(-\frac{|v|^\beta}{\kappa\sigma_v^\beta}\right) \tag{3.52}$$

where $\beta > 0$ is the shape parameter, $\Gamma(\cdot)$ is the Gamma function, and $\kappa = (\Gamma(2/\beta)/\Gamma(4/\beta))^{\beta/2}$ [110]. When $\beta = 2$, GGD reduces to the Gaussian distribution. The case of $\beta < 2$ models super-Gaussian distributions. Especially, $\beta = 1$ corresponds to the Laplacian distribution [110]. The smaller the value of β is, the more impulsive the noise is. If the shape parameter β is known, then we can select $p = \beta$ which gives the maximum likelihood (ML) estimate. Since the ML estimate asymptotically approaches the minimum variance, $p = \beta$ is statistically optimal for GGD noise. In the general case with possibly unknown noise statistics, the optimal p aims at minimizing the estimation error

$$\min_{p>0} \|\widehat{\boldsymbol{M}}(p) - \boldsymbol{A}\|_{\mathrm{F}}^2. \tag{3.53}$$

where $\widehat{\boldsymbol{M}}(p)$ denotes the solution of (3.9) for a given p. Again, it is impractical to obtain the optimal p because $\boldsymbol{A}$ is not available in practice. Roughly speaking, to select a proper p from $(0, 2)$, we need to consider the following two aspects.

1) Statistical perspective. The statistical property of the noise needs to be taken into account. The more impulsive the noise is, the smaller value of p is preferred. If the noise is not so impulsive, the choice of $1 < p \leq 2$ is suitable. If the noise is more impulsive, it has a more spike-like property, which is somewhat analogous to sparsity. As analyzed in the literature on sparse signal recovery and compressed sensing [39, 127], sparsity can be well measured using the ℓ_1-norm or even better using ℓ_p-norm with $p < 1$.

2) Computational perspective. As p decreases to zero, the nonconvexity and nonsmoothness of the ℓ_p-norm becomes stronger, which brings more difficulties in minimization. The computational challenges induced by a very small p includes increased probability of being trapped into local minima far away from the global minimum and slow convergence rate. Therefore, it is not recommended to choose p close to 0.

To summarize, choosing an appropriate p is a trade-off between the statistical and computational aspects. For ADMM, the proximity operator of the ℓ_1-norm is computationally simplest since it has closed-form expression. Thus, it is preferred to choose $p = 1$ for the ADMM. If there is no prior information for the noise, we can resort to cross-validation, which has been discussed above for rank selection, to determine p. The reader is referred to the simulation results on the choice of p with outliers in Section 3.4.

3.4 Experimental Results

All the experiments are conducted using a computer with a 3.2 GHz CPU and 4 GB memory.

3.4.1 Results of Synthetic Random Data

Under stated otherwise, a typical experimental setting in [45] is considered where $m = 150$, $n = 300$, and the rank is $r = 10$. The proposed algorithms are compared with SVT [45], SVP [55], and AP [57], WNNM [53], RPCA for matrix completion (RPCA-MC) [32], PARSuMi [67], VBMFL_1 [69]. A noise-free matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ of rank r is generated by the product of $\boldsymbol{A}_1 \in \mathbb{R}^{m\times r}$ and $\boldsymbol{A}_2 \in \mathbb{R}^{r\times n}$ whose entries satisfy the standard Gaussian distribution. We randomly select 45% entries of $\boldsymbol{A}$ as the available

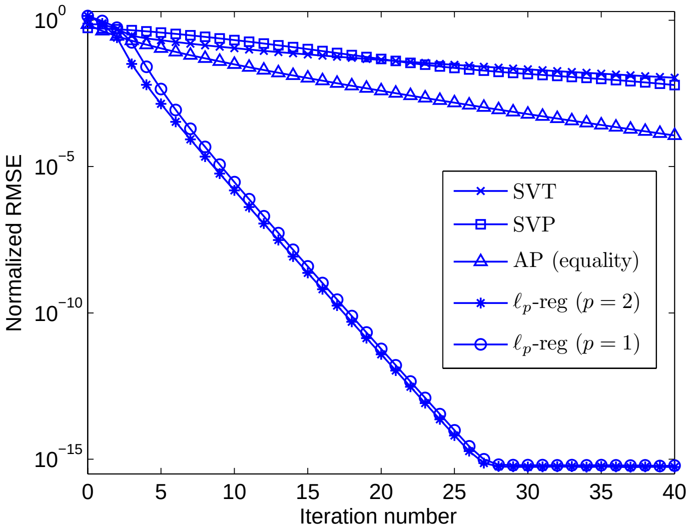


Figure 3.1: Normalized RMSE versus iteration number for noise-free case.

observations. The normalized root mean square error (RMSE) is employed as the performance measure, which is defined as:

$$\mathrm{RMSE}(\widehat{\boldsymbol{M}}) = \sqrt{\mathrm{E}\left\{\frac{\|\widehat{\boldsymbol{M}} - \boldsymbol{A}\|_{\mathrm{F}}^2}{\|\boldsymbol{A}\|_{\mathrm{F}}^2}\right\}} \tag{3.54}$$

where $\widehat{\boldsymbol{M}}$ is the result obtained by a matrix completion method, and is computed based on 100 independent trials.

Figure 3.1 plots the RMSE versus iteration number in the noise-free case where ℓ_p-reg represents the iterative ℓ_p-regression method in Algorithm 1. Note that we do not show the result of the ADMM because for any p, Algorithm 2 converges to the true solution in one iteration. It is observed that the SVT, SVP, AP with equality projection, and ℓ_p-regression schemes converge to the true matrix with a linear rate. However, our proposed method converges much faster and only about ten iterations are needed to obtain an accurate solution. The CPU times for attaining RMSE $\leq 10^{-5}$ of the SVT, SVP, AP with equality projection, ℓ_p-reg with $p = 2$ and $p = 1$ are 10.7 s, 8.0 s, 6.7 s, 0.28 s, and 4.5 s, respectively.

We then consider the noisy scenario where impulsive components are added to the available entries in $\boldsymbol{A}$. They are modeled by the two-term zero-mean Gaussian mixture model (GMM) whose p.d.f. is given by

$$p_v(v) = \sum_{i=1}^{2} \frac{c_i}{\sqrt{2\pi}\sigma_i} \exp\left(-\frac{v^2}{2\sigma_i^2}\right) \tag{3.55}$$

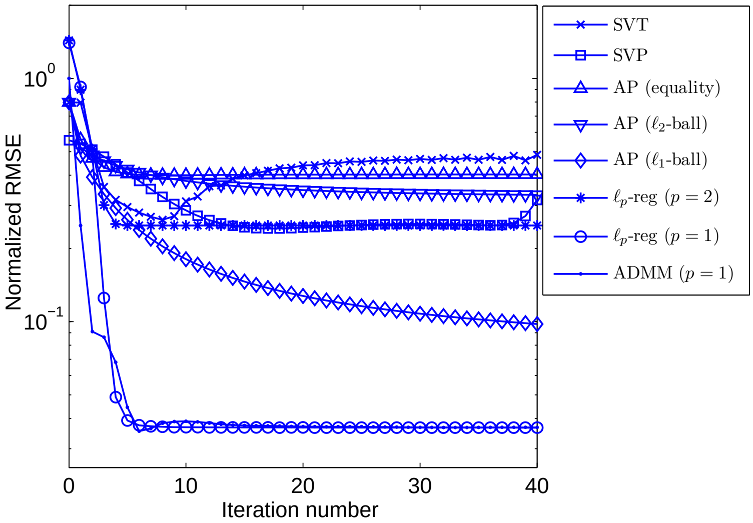


Figure 3.2: Normalized RMSE versus iteration number in GMM noise at SNR = 6 dB.

where $0 \leq c_i \leq 1$ and σ_i^2 are the probability and variance of the ith term, respectively, with $c_1 + c_2 = 1$. If $\sigma_2^2 \gg \sigma_1^2$ and $c_2 < c_1$ are selected, large noise samples of variance σ_2^2 occurring with a smaller probability c_2 can be viewed as outliers embedded in Gaussian background noise of variance σ_1^2. Thus, the GMM can well model the phenomenon with both Gaussian noise and outliers. The total noise variance is $\sigma_v^2 = \sum_i c_i \sigma_i^2$ and the signal-to-noise ratio (SNR) is defined as

$$\mathrm{SNR} = \frac{\|\boldsymbol{A}_\Omega\|_\mathrm{F}^2}{|\Omega|\sigma_v^2}. \tag{3.56}$$

Figure 3.2 plots the RMSE versus iteration number in additive GMM noise at SNR = 6 dB with $\sigma_2^2 = 100\sigma_1^2$ and $c_2 = 0.1$. We see that the SVT and SVP cannot stably converge to a reasonable solution. The iterative ℓ_p-regression and ADMM with $p = 1$ converge fast to a solution with a higher accuracy while those with $p = 2$ and the AP with projections onto equality and ℓ_2-ball cannot achieve a reliable estimation in impulsive noise. The AP with projection onto ℓ_1-ball is somewhat robust to outliers. Still, its performance is worse than the proposed schemes. Importantly, we see that about ten iterations are enough for our two algorithms to converge. That is, a value of several tens for K_reg and K_ADMM is enough for convergence. Employing the stopping criteria of relative change of the current and previous iterations is less than 10^{-4} and (3.45) with $\delta = 10^{-3}$ in the ℓ_p-regression and ADMM algorithms, respectively, the CPU times of the SVT, SVP, AP with projections onto equality, ℓ_2-ball, and ℓ_1-ball, ℓ_p-reg

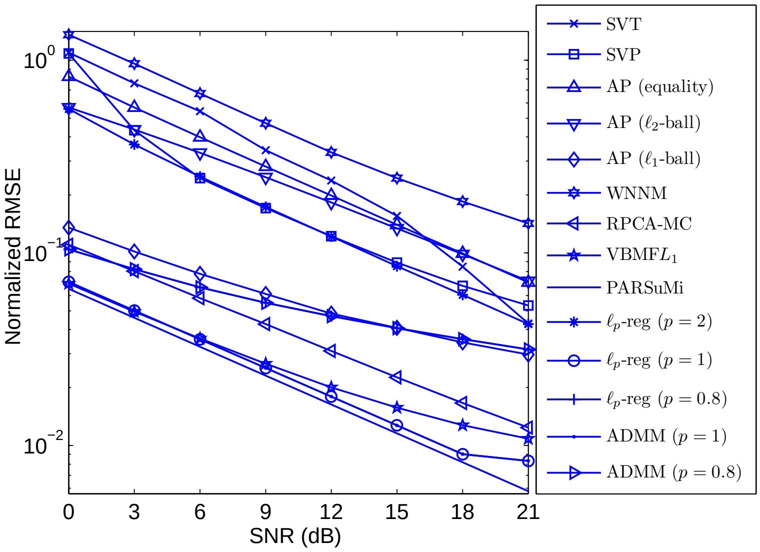


Figure 3.3: Normalized RMSE versus SNR in GMM noise.

with $p = 2$ and $p = 1$, and ADMM with $p = 1$ are 197.3 s, 10.6 s, 7.5 s, 7.9 s, 8.4 s, 0.25 s, 5.2 s, and 3.1 s, respectively.

Figure 3.3 plots the RMSE versus SNR for different methods. It is seen that the ℓ_1-regression, ADMM with $p = 1$, PARSuMi and VBMFL_1 have comparable performance. The four schemes have the minimum RMSE for all SNRs and thus they are superior to the remaining schemes in terms of robustness. Though it is slightly inferior to the four methods above, the RPCA-MC performs better than the SVT, SVP, AP, and WNNM. Figure 3.4 plots the RMSE versus percentage of observations, i.e., $|\Omega|/(mn)$ at SNR = 9 dB in GMM noise and $r = 5$. Again, the two proposed methods with $p = 1$, PARSuMi and VBMFL_1 have the best performance. Note that the SVT reports divergence for percentage of 10% and thus the result at this point is not included.

Figure 3.5 plots the average running time versus percentage of observations of the ℓ_p-reg with $p = 1, 2$ and ADMM with $p = 1$. We observe that the running time linearly increases with the percentage of observations. Note that the computational complexity of the two proposed methods is $\mathcal{O}(K|\Omega|r^2)$ where the number of observations $|\Omega|$ is the product of the observation percentage and the number of total entries of the matrix. Therefore, the complexity is linearly proportional to the percentage of observations, which aligns the results of Figure 3.5.

Impact of Rank and p: The impact of p on the performance is investigated. First a strongly impulsive GMM noise with SNR = 6 dB is used. Figure 3.6 plots the

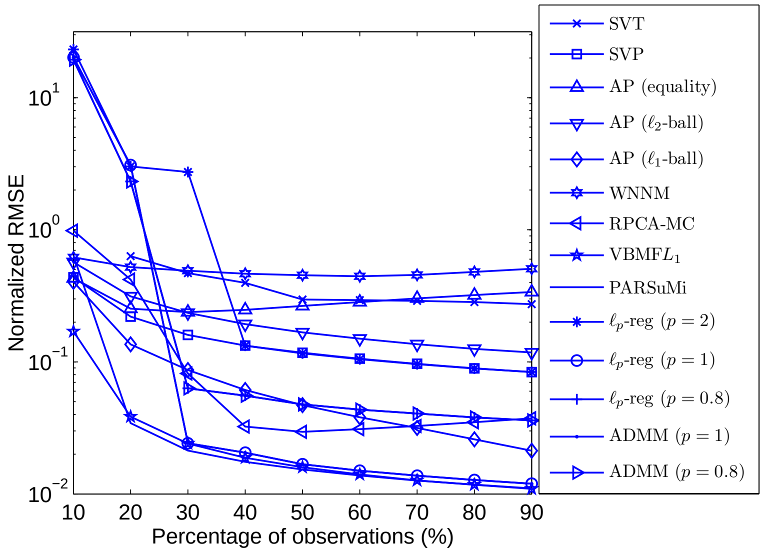


Figure 3.4: Normalized RMSE versus percentage of observations in GMM noise at SNR = 9 dB.

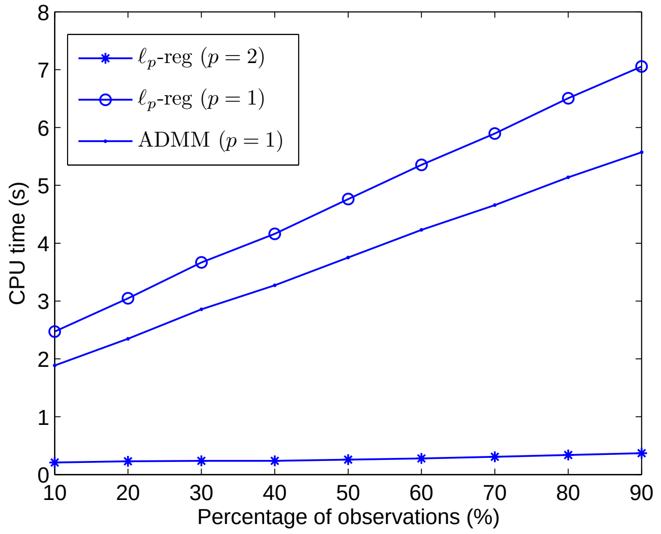


Figure 3.5: CPU time versus percentage of observations.

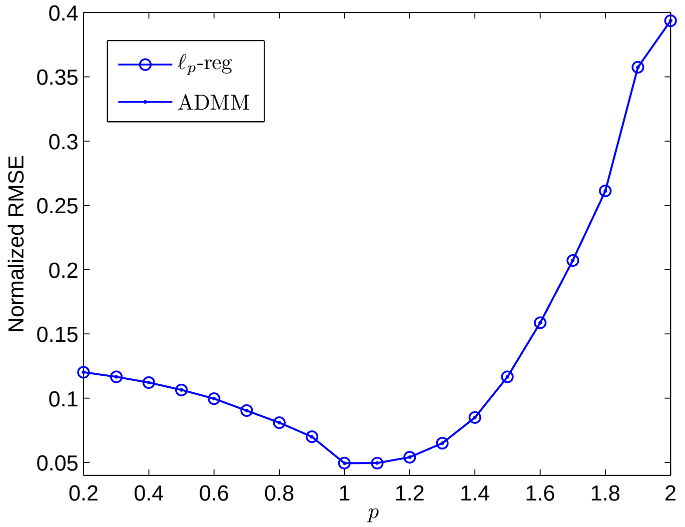


Figure 3.6: Normalized RMSE versus p in strongly impulsive GMM noise.

RMSE versus $p \in [0.2, 2]$. It is seen that using $p < 1$ is worse than $p = 1$. This may be explained from the computational perspective. As p decreases to zero, the nonconvexity and nonsmoothness of the ℓ_p-norm makes its minimization more difficult. Therefore, it is not recommended to choose p close to 0. Also, as p increases in $[1, 2]$, the robustness degrades. For computational simplicity and performance improvement, the value of $p = 1$ is the best choice for strongly impulsive noise. Then the moderately impulsive GGD noises at SNR $= 6$ dB with $\beta = 1.3$ and $\beta = 1.6$ are used. Figure 3.7 shows the RMSE versus $p \in [0.2, 2]$ in GGD noise. As we see, the optimal p is close to the shape parameter β of the GGD noise. If the noise is not so impulsive, it is preferred to employ $p \in (1, 2)$ instead of $p \leq 1$.

We study how the presumed rank affects the performance of the proposed approaches as well as SVP and AP, which also require rank information. The experimental setting is the same as above except SNR $= 9$ dB. Figure 3.8 shows the normalized RMSE versus the presumed rank varying from 4 to 16 while the true value is 10. All the methods degrade when the rank is not accurately estimated. In addition, the performance degradation when the rank is underestimated ($r < 10$) is much severer than the case when the rank is overestimated ($r > 10$). This result implies that it is not preferred to underestimate the rank. The ℓ_p-reg and ADMM with $p = 1$ exhibit the best robustness to the rank estimation error.

Results of Cross-Validation: Rank estimation using cross-validation is investigated. The experimental setting is the same as above except that only 95% of the observed

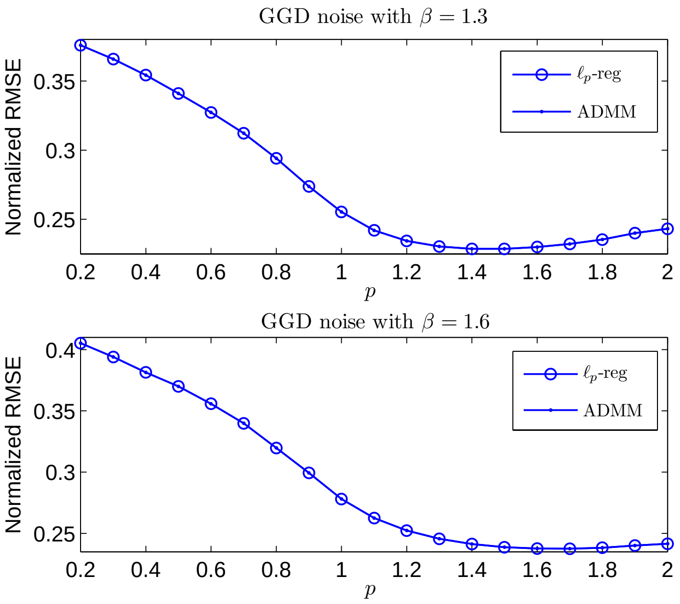


Figure 3.7: Normalized RMSE versus p in moderately impulsive GGD noise.

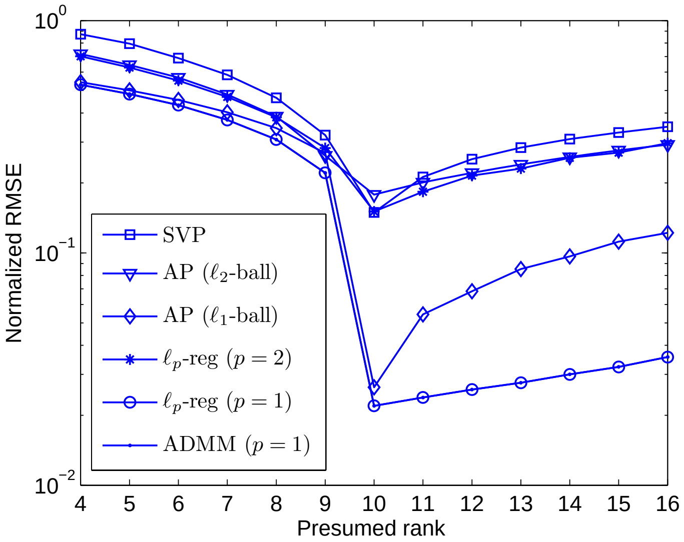


Figure 3.8: Normalized RMSE versus presumed rank.

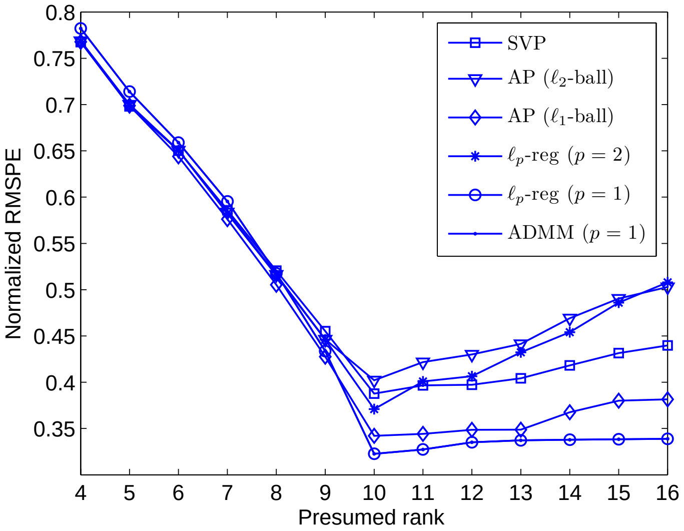


Figure 3.9: Normalized RMSPE versus presumed rank in cross-validation.

entries are randomly drawn out as training data while the remaining 5% observed entries are taken as testing data, i.e., $|\Omega_1|/|\Omega| = 0.95$. For each rank $r \in [4, 16]$, 100 random trials are conducted to compute the average RMSPE of (3.50) and MAPE of (3.51). Figures 3.9 and 3.10 plot the RMSPE and MAPE versus the presumed rank. It is clearly observed that all methods, including the proposed algorithms, RMSPE and MAPE are minimized at $r = 10$, which is exactly the true rank. The effectiveness of cross validation for rank estimation is thus verified. Nevertheless, the MAPE gives more stable result than the RMSPE, indicating that it is more suitable in the presence of outliers.

Phase Transition: Phase transition figures, i.e., the probability of recovery and normalized RMSE, versus rank and percentage of missing entries, are shown in Figures 3.11 and 3.12, respectively. For each pair of rank and missing percentage, 100 independent trials are carried out. Since the observations are noisy, we declare a trial to be successful if the normalized RMSE is less than 0.2. The SNR is fixed as 9 dB while the rank and missing percentage vary. In addition to the two proposed methods with $p = 1$ and 2, the result of SVP is included for comparison. From Figure 3.11, it is observed that the "white region" of the ℓ_p-reg and ADMM with $p = 1$ is larger than those of $p = 2$ and SVP. This means that the proposed methods with $p = 1$ perform better when the rank or missing percentage is large. The smaller RMSEs of the ℓ_p-reg/ADMM with $p = 1$ in Figure 3.12 also validate their superior performance in the presence of outliers.

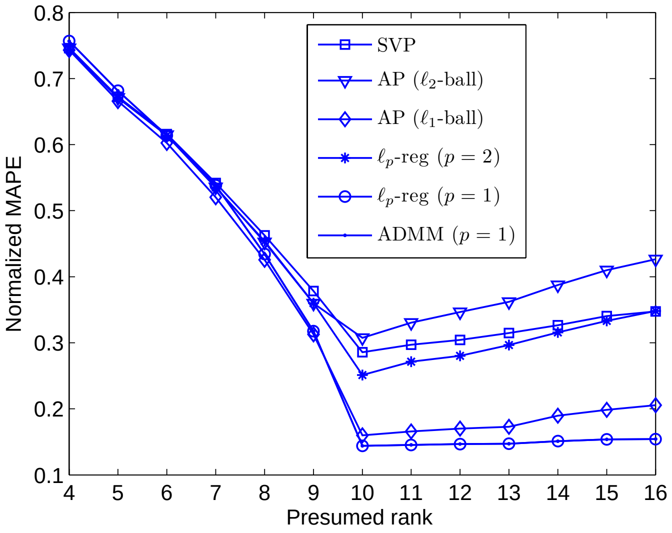


Figure 3.10: Normalized MAPE versus presumed rank in cross-validation.

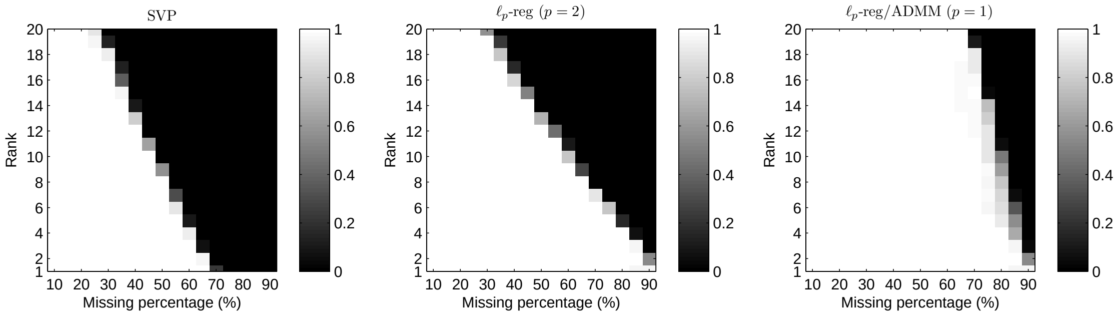


Figure 3.11: Phase transition of probability of recovery versus missing percentage and rank.

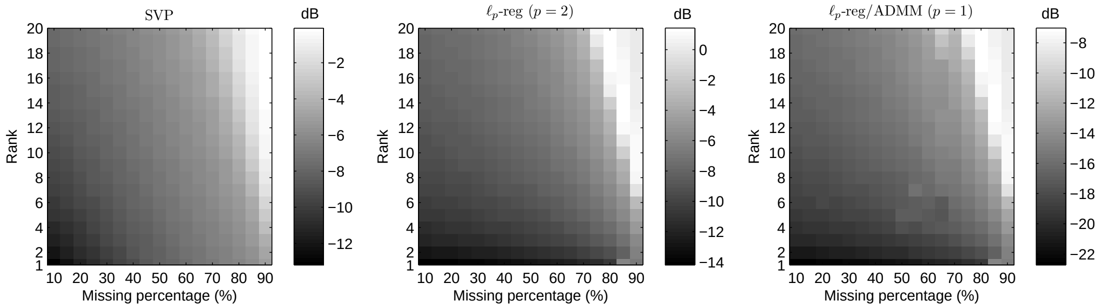


Figure 3.12: Phase transition of normalized RMSE versus missing percentage and rank.

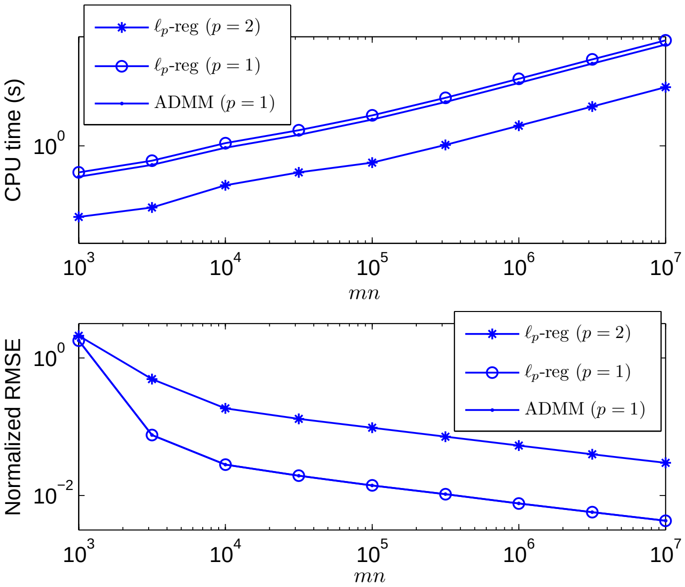


Figure 3.13: Running time and normalized RMSE versus matrix dimension mn.

Scalability: In the era of big data, it is of great interest to know whether a matrix completion algorithm is scalable to the dimension of the problem. Theoretically, the computational complexity of the two proposed methods is $\mathcal{O}(Kp_{\text{obs}}mnr^2)$ where $p_{\text{obs}} \in (0, 1]$ is the percentage of observations. Herein, simulations are conducted to check this computational complexity. First we fix $p_{\text{obs}} = 0.45$ and $r = 10$ while the matrix dimension mn varies from 10^3 to 10^7. In this simulation, we assign $m = n$, meaning that m varies from 32 to 3162. Figure 3.13 shows the CPU time and RMSE versus mn. It is seen that the CPU time is linearly proportional to mn. This result verifies the linear time-complexity and hence the scalability of the proposed algorithms. Also, it is observed that the RMSE decreases as the matrix dimension increases provided that the rank and observation percentage remain unchanged. We then fix $m = n = 200$ and $p_{\text{obs}} = 0.45$ while the rank varies from 1 to 29. Figure 3.14 shows the CPU time and RMSE versus the rank. We observe that the CPU time quadratically increases with the rank, which aligns the complexity of $\mathcal{O}(Kp_{\text{obs}}mnr^2)$. In this sense, the proposed schemes are not scalable to the rank. Fortunately, the rank is often much smaller than the size of the matrix in practical applications. The low-rank property is helpful to reduce the computational cost and improve the recovery performance.

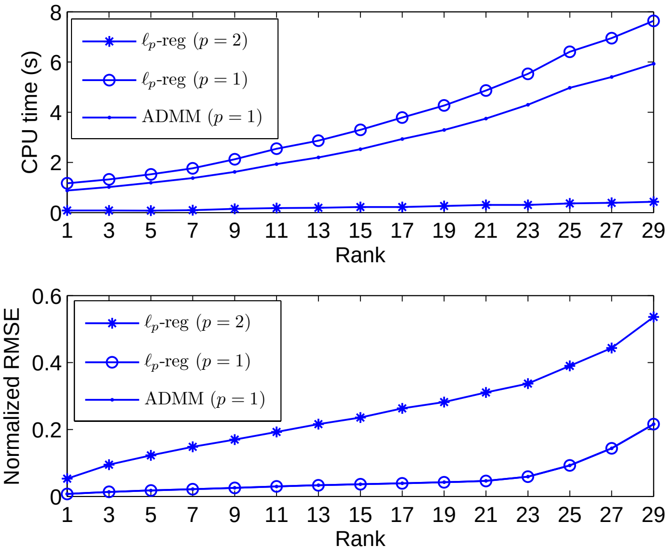


Figure 3.14: Running time and normalized RMSE versus rank.

3.4.2 Image Inpainting in Salt-and-Pepper Noise

Now matrix completion is applied to image inpainting in salt-and-pepper noise. A color image in [75, 128] is adopted where we first convert it to gray-scale so that it can be represented by a matrix. As shown in Figure 3.15, the missing data of the original image correspond to “ICCV”, “2009”, and “LRTC”. The available entries are contaminated by adding salt-and-pepper noise. We use the function “`imnoise(I, 'salt & pepper',` ρ`)` ” in MATLAB, where the normalized noise intensity is ρ corresponding to SNR $= 1/\rho$, to generate the salt-and-pepper noise. The widely-used peak signal-to-noise ratio (PSNR)

$$\text{PSNR} = 255^2/\text{MSE} \tag{3.57}$$

where 255 is the peak value of a gray-scale image and

$$\text{MSE} = \frac{1}{mn}\|\widehat{\boldsymbol{M}} - \boldsymbol{A}\|_{\text{F}}^2. \tag{3.58}$$

Obviously, the smaller MSE, the larger PSNR. That is, a larger PSNR implies a better image reconstruction. The PSNR of the noisy image with missing values without any processing can be considered as the baseline. Generally, the PSNR will be increased after processing by an image inpainting algorithm.

We first set the rank as $r = 6$. The SVT shows divergence and we cannot include its result while it is observed that the SVP, AP with equality projection, and WNNM fail

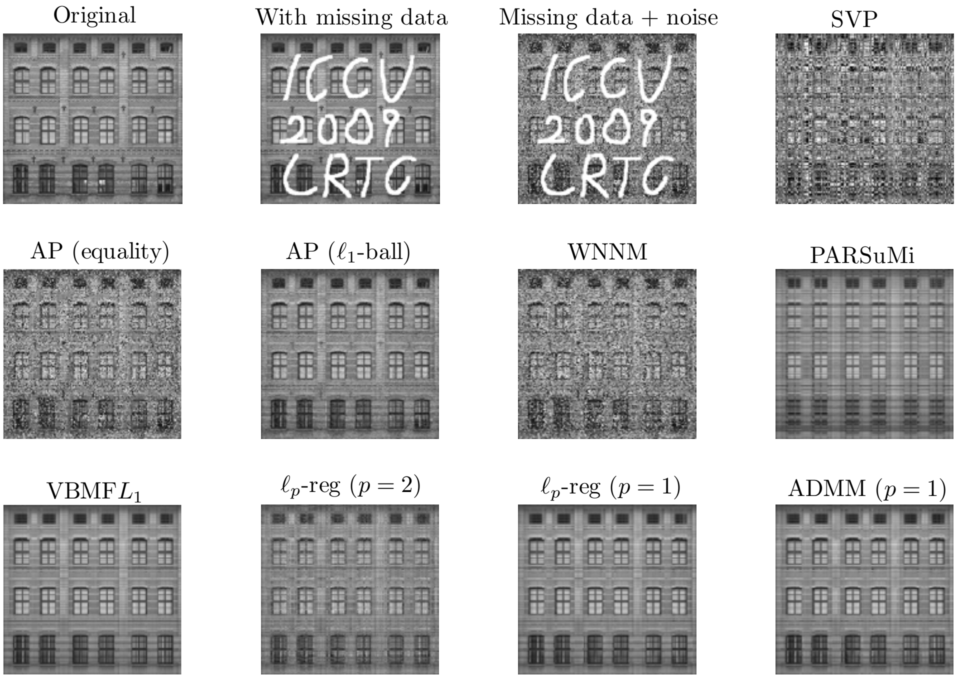


Figure 3.15: Noisy image with missing data and recovered results of SVP, AP, WNNM, PARSuMi, VBMFL_1, iterative ℓ_p-regression and ADMM.

in recovering the image. The AP with projection onto ℓ_1-ball gives a satisfactory result but is still inferior to our two methods. Most important, the ℓ_p-regression and ADMM with $p = 1$ are quite robust to the salt-and-pepper noise and they provide accurate estimates of the original image. We also see that the ℓ_1-regression greatly improves the performance compared with the ℓ_2-regression in impulsive noise environment. The PARSuMi and VBMFL_1 also exhibit robustness to salt-and-pepper noise. The CPU times for the SVP, AP with projections onto equality and ℓ_1-ball, WNNM, PARSuMi, VBMFL_1, ℓ_p-regression with $p = 2$ and $p = 1$, and ADMM with $p = 1$ are 20.3 s, 15.6 s, 17.4 s, 140.3 s, 9.0 s, 7.4 s, 0.4 s, 7.8 s and 4.9 s, respectively.

The effect of rank selection to the performance of image inpainting is investigated. Figure 3.16 shows the PSNR versus rank in salt-and-pepper noise at SNR = 7 dB. The baseline is also plotted. We see that the two proposed algorithms and the AP with $p = 1$ have the highest PSNR around $r = 6$ or 7. The PARSuMi is not sensitive to rank in this experiment example. The PSNR of VBMFL_1 quickly increases as the rank increases when $r \leq 6$ and it slowly decreases when $r \geq 14$. Therefore, the rank of VBMFL_1 can take values in $r \in [6, 14]$. Because the computational load becomes heavier as the rank increases, it is preferred to select a smaller rank when the performance is similar. Figure 3.16 shows the PSNR versus SNR at $r = 6$. From Figure 3.17, the VBMFL_1,

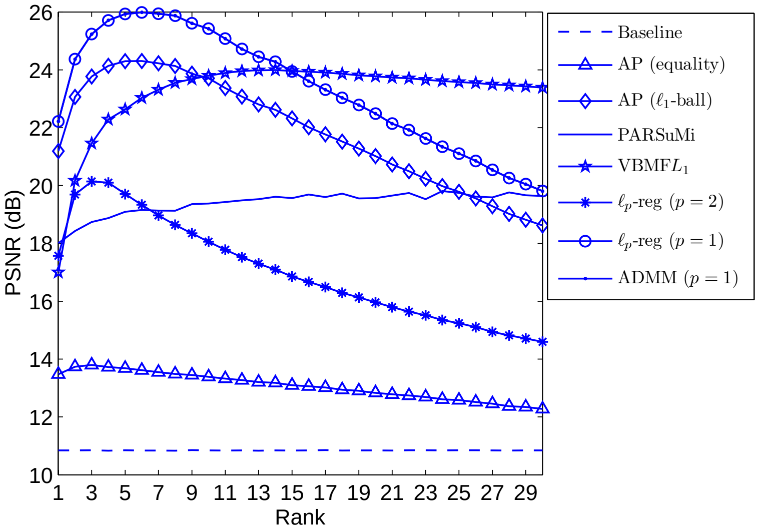


Figure 3.16: PSNR versus rank in salt-and-pepper noise at SNR = 7 dB.

AP with ℓ_1-ball projection, and the two proposed approaches with $p = 1$ have the best performance. Since the SVP and WNNM are not robust to the salt-and-pepper noise and their PSNRs are low, we do not show the corresponding results of them in Figures 3.16 and 3.17.

We then investigate inpainting of another two images whose original versions are taken from [75,128]. The color images are converted to gray-scale for a matrix representation. The first image is a building and the second is a texture. Both of them are structured and approximately have a low-rank property. The rank is set to $r = 6$. Figures 3.18 and 3.19 show the original and incomplete images corrupted by salt-and-pepper noise, and the recovered results of SVP, AP, WNNM, PARSuMi, VBMFL_1, iterative ℓ_p-regression and ADMM. Again, we see that the VBMFL_1, ℓ_p-regression with $p = 1$ and ADMM are quite robust to impulsive noise and have the best recovery performance. The PARSuMi and AP with projection onto ℓ_1-ball are inferior to the three methods although they yield satisfactory results. The SVP, AP with equality projection, WNNM are not robust to salt-and-pepper noise.

3.4.3 Results of Recommender Systems

The application of our matrix completion methods to recommender systems is considered. The MovieLens 100K Data set, which is available at [129], is used. This data set consists of 100,000 ratings from 943 users on 1,682 movies. The rating varies from

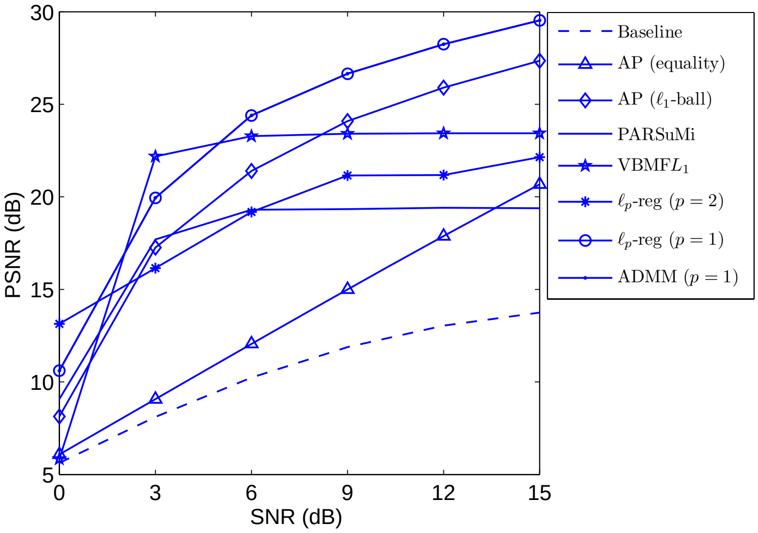


Figure 3.17: PSNR versus SNR in salt-and-pepper noise.

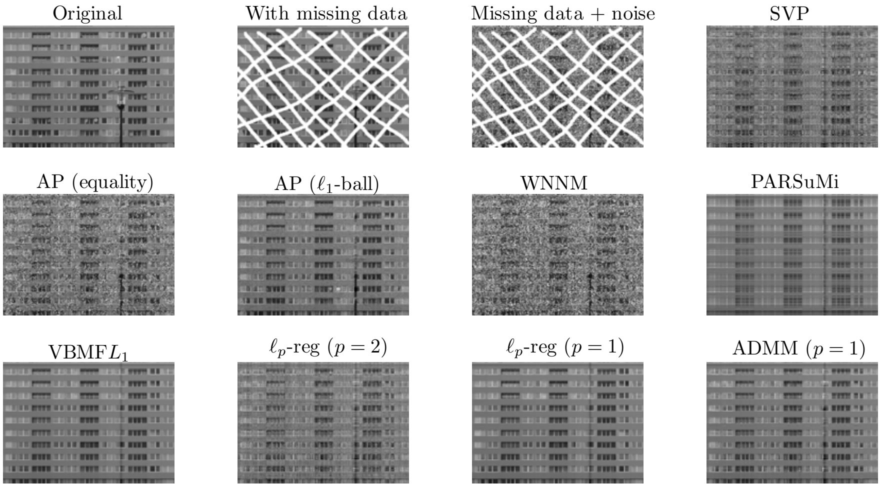


Figure 3.18: Noisy image of a building with missing data and recovered results of SVP, AP, WNNM, PARSuMi, VBMFL_1, iterative ℓ_p-regression and ADMM.

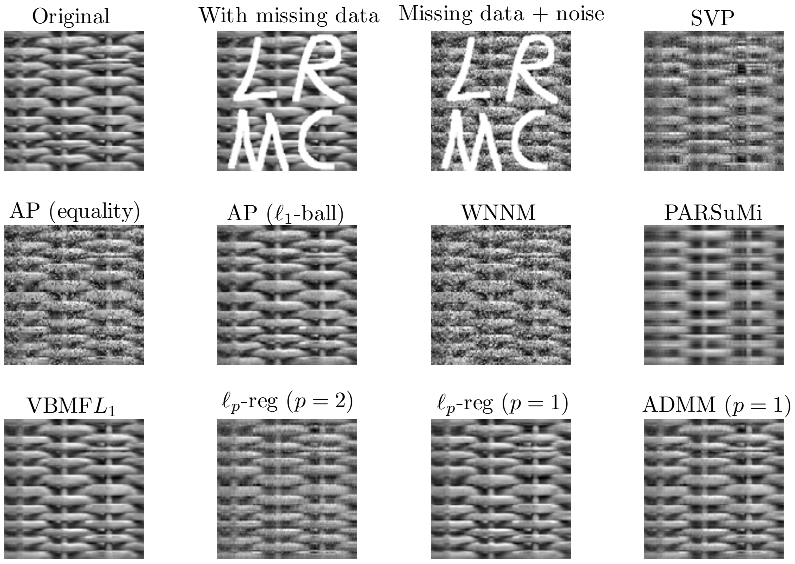

Original
With missing data
Missing data + noise
SVP
AP (equality)
AP (ℓ_1-ball)
WNNM
PARSuMi
VBMFL_1
ℓ_p-reg ($p = 2$)
ℓ_p-reg ($p = 1$)
ADMM ($p = 1$)

Figure 3.19: Noisy image of a texture with missing values and recovered results of SVP, AP, WNNM, PARSuMi, VBMFL_1, iterative ℓ_p-regression and ADMM.

1 to 5. Each user has rated at least 20 movies. The rating of the ith user to the jth movie is stored as the (i, j)th entry of the matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$. We have $m = 943$, $n = 1682$, and the number of known entries $|\Omega| = 10^5$, which is much smaller than the number of all entries $mn = 1.586 \times 10^6$. That is, the percentage of observations is only 6.3%. Low-rank matrix completion is applied to infer other unknown entries for a recommender system. Since the remaining 93.7% entries are unknown, we cannot judge whether the inferred entries are correct. Like the strategy in cross-validation, Ω is divided into Ω_1 and Ω_2 such that $\Omega_1 \cup \Omega_2 = \Omega$. In this experiment, $\boldsymbol{A}_{\Omega_1}$ and $\boldsymbol{A}_{\Omega_2}$ are used for matrix completion and prediction error computation, respectively. Define the result of matrix completion using partial observations $\boldsymbol{A}_{\Omega_1}$ as $\widehat{\boldsymbol{M}}$, we evaluate the mean absolute error (MAE)

$$\mathrm{MAE} = \frac{1}{4|\Omega_2|}\left\|[\widehat{\boldsymbol{M}}]_{\Omega_2} - \boldsymbol{A}_{\Omega_2}\right\|_1 \tag{3.59}$$

using $\boldsymbol{A}_{\Omega_2}$, where the factor 4 is the difference of the maximum and minimum scores, namely, 5 and 1. Note that the MAE has been widely used as the performance measure of recommender systems [50, 65].

We first use cross-validation to estimate an appropriate rank, where the portion of training data $|\Omega_1|/|\Omega|$ is set to 95%. For a given rank, matrix completion is performed using $\boldsymbol{A}_{\Omega_1}$. For each rank $r \in [1, 15]$, 100 random divisions of Ω_1 and Ω_2 are conducted to compute the average MAE. Figure 3.20 plots the MAE versus the estimated rank.

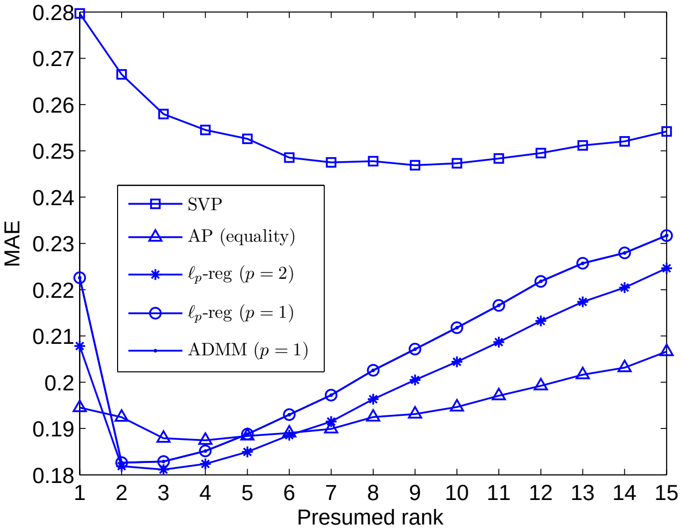


Figure 3.20: MAE versus rank in cross-validation with MovieLens data set.

For the AP and two proposed methods, $r = 3$ is the best rank estimate while for the SVP the rank estimate is $r = 7$. These rank estimates are adopted in the following tests.

Figure 3.21 plots the MAE versus $|\Omega_1|/|\Omega|$ varying from 20% to 80% of the SVP, AP with equality projection, and ℓ_p-regression and ADMM. The prediction accuracy of two proposed methods is higher than AP and SVP. The performances using $p = 1$ and $p = 2$ are quite similar. This is because the ratings are integers whose values are taken from $\{1, \cdots, 5\}$ and there are no random noises or outliers. In the absence of noise/outlier, the method with $p = 2$ is good enough for matrix completion. Still, the proposed scheme with $p = 1$ also works well and useful for the case where there is no noise or outlier.

3.5 Summary

Many existing techniques for matrix completion are not robust to outliers. To overcome this drawback, we have devised two algorithms for robust matrix completion using low-rank factorization via the ℓ_p-minimization criterion with $0 < p < 2$. The first method tackles the nonconvex factorization with missing data by iteratively solving multiple independent linear ℓ_p-regressions. On the other hand, the second solution exploits the ADMM for incomplete factorization in ℓ_p-space. Each iteration of the

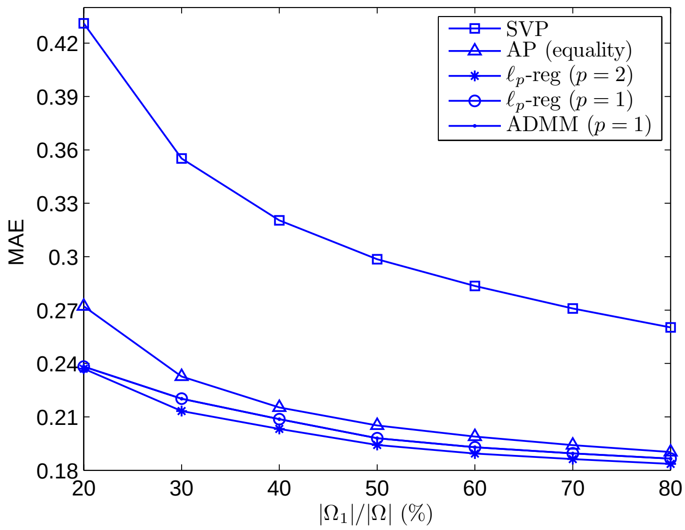


Figure 3.21: MAE versus $|\Omega_1|/|\Omega|$ with MovieLens data set.

ADMM requires solving an LS factorization problem and calculating the proximity operator of the ℓ_p-norm. The two algorithms have comparable recovery performance as well as computational efficiency and allow parallel or distributed realization. Their total complexity is $\mathcal{O}(K|\Omega|r^2)$, where K is a fixed constant of several hundreds to thousands, which is lower than the popular schemes employing the nuclear/Schatten p-norm minimization that require SVD. Furthermore, our solutions generalize the conventional matrix factorization based on Frobenius norm minimization. The superiority of the developed algorithms over the SVT, SVP, and AP in terms of implementation complexity, recovery capability and outlier-robustness is demonstrated using synthetic and real-world data.

Chapter 4

Greedy Pursuit for Approximation of Multiple Matrices in ℓ_2-Space

In many practical applications, the data have the form of two-dimensional array and thus, can be naturally described by a matrix. To handle multiple data points, we face with processing *a set of matrices.* A popular method is based on "vectorization", where each matrix is converted into a vector and hence, all data points are reorganized into a single matrix. In this manner, the low-rank approximation techniques such as the celebrated singular SVD for a single matrix can be applied. However, the vectorization strategy yields a long vector and then a matrix with very large size, resulting in a high time complexity of the SVD. Moreover, this manner breaks the 2D structure and the innate relation between row and column.

Aiming to overcome the two drawbacks, this chapter investigates how to efficiently find the common low-rank structure of *multiple matrices* beginning with in the ℓ_2-space. We devise greedy pursuit (GP) algorithms for joint low-rank approximation of multiple matrices (LRAMM), where the celebrated singular value decomposition (SVD) for a single matrix is not applicable. The GP solves the LRAMM by decomposing it into a series of rank-one approximations. At each iteration, it finds the best rank-one approximation of the residual matrices and then, the rank-one basis matrices are subtracted from the residual. An alternating optimization approach is designed for the rank-one fitting. To further reduce the complexity, an economic greedy pursuit (EGP) that avoids the iterative procedure for rank-one fitting is proposed. The orthogonal greedy pursuit (OGP) is also developed to accelerate the convergence, where the cost of weight re-computation is reduced by a recursive update manner. The per-iteration complexity of the three algorithms is linear with the number and dimension of the matrices. Thus, they are scalable to large-scale problems. The convergence of the GP, EGP and OGP is theoretically proved. In particular, the reconstruction error of each algorithm decays exponentially. A lower bound of the exponential decay factor or convergence rate is derived. Different from the generalized low rank approximations of matrices (GLRAM) belonging to non-diagonal and orthogonal decompositions, the greedy methodology achieves a nonorthogonal but joint diagonal decomposition of multiple matrices, yielding a higher compression ratio. Experimental results demonstrate that the superiority of the greedy schemes in terms of computational simplicity, fast convergence and accurate reconstruction.

4.1 Introduction

It is known that the celebrated PCA is only applicable to a single matrix, where each of its columns corresponds to a data point. Thus, it needs to first convert the data into vectors to apply the PCA if the data points are not in the form of vectors. For two-dimensional (2D) array data such as images and frames of video, which are represented by matrices, the "vectorization" yields a long vector and then a matrix with very large size. This results in a high time complexity of the SVD [35]. In addition, the vectorization breaks the 2D structure and the innate relation between row and column

To overcome the drawback of the vectorization based approaches, the two-dimensional PCA (2DPCA) [36], GLRAM [16] and 2D-SVD [70] directly deal with multiple matrices without vectorization, resulting in a significant complexity reduction. Under the same compression ratio, the GLRAM and 2D-SVD achieve smaller reconstruction error than the 2DPCA. The three methods belong to orthogonal transform, where the columns of the resultant factorized subspaces are orthogonal. Note that unlike SVD, the GLRAM and 2D-SVD do not achieve diagonal decomposition.

This chapter addresses learning the common low-rank structure of multiple matrices. The joint low-rank approximation of multiple matrices (LRAMM) can be viewed as an extension of the single matrix case. As in the 2DPCA, GLRAM and 2D-SVD, the LRAMM also does not convert matrices into vectors and thus can avoid processing the matrix with much larger size. Different from the GLRAM and 2D-SVD, the LRAMM achieves a nonorthogonal but joint diagonal factorization of multiple matrices, which leads to a more compact representation and a higher compression ratio.

A greedy pursuit (GP) algorithmic framework including three variants is designed for the learning task. The GP decomposes the problem into a serial of rank-one approximations and works in an iterative manner. At each iteration, a rank-one approximation of the residuals is conducted. Then, the rank-one matrices are subtracted from the residuals. It is worth pointing out that the greedy algorithms [130–132] have been widely applied to numerous signal processing and machine learning problems. The matching pursuit (MP) [133], orthogonal matching pursuit (OMP) [134] and their robust versions in ℓ_p-space [135] for sparse signal recovery and compressed sensing use the idea of "one at a time". They greedily pick up an atom that is most correlated with the current residual at each iteration. The OMP for recovery of a sparse one-dimensional vector has been extended to recovering a single low-rank matrix [136,137] with missing values, which is also referred to as matrix completion. The greedy algorithm of [136] for square loss function is extended to more general models whose loss function can be nonsmooth in [138]. The performance guarantee of greedy pursuit for matrix completion is theoretically analyzed in [139]. It is worth pointing out that all the existing greedy pursuit methods [136]– [139] are designed for the completion of a single low-rank matrix, which are different from the proposed GP for multiple matrices.

A set of matrices can be viewed as a third-order tensor [73]. With the use of existing low-rank tensor decomposition techniques [73–75] such as higher-order SVD (HOSVD), one can also obtain a low-rank approximation of the multiple matrices. However, our methodology for LRAMM is different from the low-rank tensor approximation.

4.2 Problem Formulation and Preliminaries

4.2.1 Problem Formulation

Given K matrices $\{\boldsymbol{A}_1, \cdots, \boldsymbol{A}_K\} \in \mathbb{R}^{m\times n}$, we aim at finding a low-rank approximation of the K matrices. To be more specific, consider the following low-rank approximation

$$\boldsymbol{A}_k \approx \boldsymbol{U}\boldsymbol{S}_k\boldsymbol{V}^{\mathrm{T}},\ k = 1, \cdots, K \tag{4.1}$$

where $\boldsymbol{U} \in \mathbb{R}^{m\times r}$, $\boldsymbol{V} \in \mathbb{R}^{n\times r}$, and the diagonal matrix $\boldsymbol{S}_k \in \mathbb{R}^{r\times r}$ is

$$\boldsymbol{S}_k = \mathrm{diag}\{s_{k,1}, \cdots, s_{k,r}\} \tag{4.2}$$

with r being the target "rank". Note that it allows that $r > \min(m, n)$ but generally $r \leq mn$ is required. In this case, r is not the rank but we still use the name of "low-rank approximation". As we will see later, (4.1) can still achieve data compression even when $r > \min(m, n)$. Clearly, $\boldsymbol{U}\boldsymbol{S}_k\boldsymbol{V}^{\mathrm{T}}$ is an approximation of $\boldsymbol{A}_k$. Note that $\boldsymbol{U}$ and $\boldsymbol{V}$ are the same for all k but $\boldsymbol{S}_k$ can be different with each other. The columns of $\boldsymbol{U}$ and $\boldsymbol{V}$ span the r-dimensional subspaces of the column and row spaces of $\{\boldsymbol{A}_k\}_{k=1}^{K}$. Therefore, the low-rank approximation also achieves subspace learning. Figure 4.1 illustrates the 2D low-rank factorization of multiple matrices given by (4.1).

If the matrices are not strictly low-rank or there exists noise, we solve the following minimization problem

$$\min_{\boldsymbol{U},\boldsymbol{V},\{\boldsymbol{S}_k\}_{k=1}^{K}} \sum_{k=1}^{K} \left\|\boldsymbol{U}\boldsymbol{S}_k\boldsymbol{V}^{\mathrm{T}} - \boldsymbol{A}_k\right\|_{\mathrm{F}}^2 \tag{4.3}$$

to obtain the low-rank approximation. One application of the above low-rank approximation is for data compression. Obviously, it requires mnK real numbers to

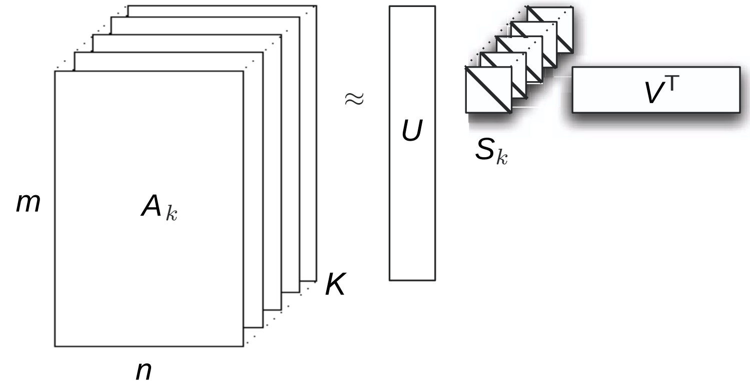


Figure 4.1: 2D low-rank factorization of multiple matrices.

store the K original matrices. Using the low-rank approximation of (4.3), the storage complexity is reduced to $(m+n+K)r$ numbers since only $\boldsymbol{U}$, $\boldsymbol{V}$, and the diagonal elements of $\{\boldsymbol{S}_k\}_{k=1}^K$ are needed to store. Thus, the compression ratio of our LRAMM is $mnK/((m+n+K)r)$. This implies that (4.1) can still achieve data compression when $r > \min(m,n)$ if $r \ll mn$. Indeed, the compression ratio and reconstruction error decrease as r increases. To guarantee a satisfying reconstruction error, we may use a $r > \min(m,n)$.

The GLRAM [16] requires $\boldsymbol{U}$ and $\boldsymbol{V}$ to be orthonormal, i.e., $\boldsymbol{U}^{\mathrm{T}}\boldsymbol{U} = \boldsymbol{V}^{\mathrm{T}}\boldsymbol{V} = \boldsymbol{I}_r$. The orthonormal constraint results in that $r \le \min(m,n)$ must be satisfied for GLRAM. Denoting the ith $(i = 1,\cdots,r)$ columns of $\boldsymbol{U}$ and $\boldsymbol{V}$ as $\boldsymbol{u}_i$ and $\boldsymbol{v}_i$, respectively, (4.3) is rewritten as

$$\begin{aligned} &\min f\left(\boldsymbol{u}_i, \boldsymbol{v}_i, \{\boldsymbol{s}^i\}_{i=1}^r\right) := \sum_{k=1}^K \left\| \sum_{i=1}^r s_{k,i} \boldsymbol{u}_i \boldsymbol{v}_i^{\mathrm{T}} - \boldsymbol{A}_k \right\|_{\mathrm{F}}^2 \\ &\text{s.t. } \|\boldsymbol{u}_i\| = \|\boldsymbol{v}_i\| = 1, \ i = 1,\cdots,r \end{aligned} \tag{4.4}$$

where the vector

$$\boldsymbol{s}^i = [s_{1,i},\cdots,s_{K,i}]^{\mathrm{T}} \in \mathbb{R}^K \tag{4.5}$$

and the unit-norm constraint is imposed to $\{\boldsymbol{u}_i, \boldsymbol{v}_i\}_{i=1}^r$ to eliminate the scaling indeterminacy. Because if $(\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{S}_k)$ is an optimal solution of (4.3), then $(\alpha_1\boldsymbol{U}, \alpha_2\boldsymbol{V}, \alpha_3\boldsymbol{S}_k)$ with the scalars $\{\alpha_1, \alpha_2, \alpha_3\} \in \mathbb{R}$ satisfying $\alpha_1\alpha_2\alpha_3 = 1$ is also an solution. This scaling indeterminacy is avoided by constraining the norms of $\{\boldsymbol{u}_i, \boldsymbol{v}_i\}_{i=1}^r$ to be unit.

If the number of matrices is $K = 1$, by Eckart-Young Theorem [26], the solution of (4.3) is the truncated SVD of $\boldsymbol{A}_1$. That is, $\{s_{1,i}\}_{i=1}^r$ are the r largest singular values of $\boldsymbol{A}_1$, and $\{\boldsymbol{u}_i\}_{i=1}^r$ and $\{\boldsymbol{v}_i\}_{i=1}^r$ are the corresponding left and right singular vectors, respectively. When the number of matrices is $K > 1$, the truncated SVD cannot be applied to solving (4.4). In this chapter, we devise greedy pursuit algorithms for efficiently solving (4.4).

4.2.2 Prior Arts on Low-Rank Representation

The most representative approach for low-rank representation is the PCA [27]. However, the PCA cannot directly handle multiple matrices whereas each matrix needs to be converted into a vector $\boldsymbol{a}_k = \text{vec}(\boldsymbol{A}_k) \in \mathbb{R}^{mn}$. Then, the K vectors $\{\boldsymbol{a}_k\}_{k=1}^K$ form the columns of following data matrix

$$\boldsymbol{X} = [\boldsymbol{a}_1, \cdots, \boldsymbol{a}_K] \in \mathbb{R}^{mn \times K}. \tag{4.6}$$

whose rank satisfies $\text{rank}(\boldsymbol{X}) \leq \min(mn, K)$. The PCA aims to find a lower rank matrix $\widehat{\boldsymbol{X}}$ to best approximate the original data matrix $\boldsymbol{X}$:

$$\min_{\text{rank}(\widehat{\boldsymbol{X}})=r} \|\widehat{\boldsymbol{X}} - \boldsymbol{X}\|_{\text{F}}^2 \tag{4.7}$$

where the target rank r is usually taken as $r \ll \min(mn, K)$ to achieve data compression or dimensionality reduction. Again, by Eckart-Young Theorem [26], the solution of (4.7) is given by the truncated SVD of $\boldsymbol{X}$, which is expressed as

$$\widehat{\boldsymbol{X}} = \sum_{l=1}^{r} \sigma_l(\boldsymbol{X}) \boldsymbol{t}_l \boldsymbol{y}_l^{\text{T}} \tag{4.8}$$

where $\sigma_l(\boldsymbol{X})$ is the lth singular value of $\boldsymbol{X}$ while $\boldsymbol{t}_l \in \mathbb{R}^{mn}$ and $\boldsymbol{y}_l \in \mathbb{R}^K$ are the corresponding left and right singular vectors. In data compression, we only need to store the so-called "principal components": the largest r singular values and the corresponding singular vectors $\{\sigma_l(\boldsymbol{X}), \boldsymbol{t}_l, \boldsymbol{y}_l\}_{l=1}^r$. Clearly, the compression ratio of the PCA is $mnK/((mn + K + 1)r)$. There are two drawbacks of the traditional PCA applied to processing multiple matrices. The first drawback is that it needs to handle a matrix of a much larger size due to transforming the original matrix into a long vector. Indeed, the truncated SVD of $\boldsymbol{X}$ has a high complexity of $\mathcal{O}(\max(m^2n^2, K^2)r)$. For high-dimensional data with $mn > K$, this complexity becomes $\mathcal{O}(m^2n^2r)$, which quadratically increases as the matrix dimension mn. The second drawback is that the vectorization breaks the 2D structure and the innate relation between row and column.

The 2DPCA computes a linear transformation $\boldsymbol{W} \in \mathbb{R}^{n \times r}$ with $r < n$, such that each matrix is projected to $\boldsymbol{C}_k = \boldsymbol{A}_k \boldsymbol{W} \in \mathbb{R}^{m \times r}$. The $\boldsymbol{W}$ maximizes the variance in the transformed space

$$\max_{\boldsymbol{W}^{\text{T}}\boldsymbol{W}=\boldsymbol{I}_r} \boldsymbol{W}^{\text{T}} \left(\sum_{k=1}^{K} (\boldsymbol{A}_k - \bar{\boldsymbol{A}})^{\text{T}} (\boldsymbol{A}_k - \bar{\boldsymbol{A}}) \right) \boldsymbol{W} \tag{4.9}$$

where $\bar{\boldsymbol{A}} = \left(\sum_{k=1}^{K} \boldsymbol{A}_k\right)/K$ is the mean of the matrices. The columns of the optimal $\boldsymbol{W}$ are the r eigenvectors of $\sum_{k=1}^{K}\left(\boldsymbol{A}_k - \bar{\boldsymbol{A}}\right)^{\mathrm{T}}\left(\boldsymbol{A}_k - \bar{\boldsymbol{A}}\right)$ corresponding to the largest r eigenvalues. The matrices can be reconstructed by $\widehat{\boldsymbol{A}}_k = \boldsymbol{C}_k \boldsymbol{W}^{\mathrm{T}}$. The 2DPCA needs to store $\{\boldsymbol{C}_k\}_{k=1}^{K}$ and $\boldsymbol{W}$, implying a compression ratio of $(mnK)/((mK+n)r)$. Its computational complexity is $\mathcal{O}(mn^2K)$. We see that the 2DPCA only applies a single-sided transform to the matrices, which results in a limited compression capability.

The GLRAM solves the constrained problem

$$\min_{\boldsymbol{U}^{\mathrm{T}}\boldsymbol{U}=\boldsymbol{V}^{\mathrm{T}}\boldsymbol{V}=\boldsymbol{I}_r, \{\boldsymbol{S}_k\}} \sum_{k=1}^{K} \left\|\boldsymbol{U}\boldsymbol{S}_k\boldsymbol{V}^{\mathrm{T}} - \boldsymbol{A}_k\right\|_{\mathrm{F}}^2 \tag{4.10}$$

where the orthonormal constraints makes it different from the proposed formulation of (4.3), where $\boldsymbol{U}$ and $\boldsymbol{V}$ are not required to be orthogonal. In addition, the matrices $\{\boldsymbol{S}_k\}_{k=1}^{K}$ given by the GLRAM are not diagonal whereas those of the LRAMM are diagonal. Hence, the compression ratio of the GLRAM is $mnK/((m+n+Kr)r)$, which is lower than the proposed approach for the same r. Exploiting the orthogonality of $\boldsymbol{U}$ and $\boldsymbol{V}$, the GLRAM is reduced to

$$\max_{\boldsymbol{U}^{\mathrm{T}}\boldsymbol{U}=\boldsymbol{V}^{\mathrm{T}}\boldsymbol{V}=\boldsymbol{I}_r} \sum_{k=1}^{K} \left\|\boldsymbol{U}^{\mathrm{T}}\boldsymbol{A}_k\boldsymbol{V}\right\|_{\mathrm{F}}^2 \tag{4.11}$$

where a two-sided transform is performed.

4.3 Greedy Pursuit Algorithms

The idea of greedy algorithms is to decompose the r-term approximation into a series of rank-one approximations. At each iteration, the greedy algorithms perform rank-one approximation of the residual matrices obtained from the previous iteration. Then, the rank-one matrices are subtracted from the residual and never revisited.

4.3.1 Greedy Pursuit

The GP for LRAMM is described in Algorithm 5. It works in an iterative fashion. We use $(\boldsymbol{u}_i, \boldsymbol{v}_i, \boldsymbol{s}^i)$ and $\{\boldsymbol{R}_k^i\}_{k=1}^{K}$ to denote the solution and the K residual matrices, respectively, at the ith iteration. In the ith iteration, the GP finds rank-one approximation of $\left\{\boldsymbol{R}_k^{i-1}\right\}_{k=1}^K$, which is formally expressed as

$$\begin{aligned} &\min f(\boldsymbol{u},\boldsymbol{v},\boldsymbol{s}) := \sum_{k=1}^{K}\left\|s_k\boldsymbol{u}\boldsymbol{v}^{\mathrm{T}} - \boldsymbol{R}_k^{i-1}\right\|_{\mathrm{F}}^2 \\ &\text{s.t. } \|\boldsymbol{u}\| = \|\boldsymbol{v}\| = 1 \end{aligned} \tag{4.12}$$

where $\boldsymbol{s} = [s_1, \cdots, s_K]^{\mathrm{T}}$ collects the K variables $\{s_k\}_{k=1}^K$ to be optimized. The optimal solution of (4.12) is taken as the solution of the ith iteration, which is denoted as $(\boldsymbol{u}_i, \boldsymbol{v}_i, \boldsymbol{s}^i)$. In some applications where the target rank r is given or can be estimated, the algorithm is terminated when $i > r$. If the target rank is unavailable, the normalized reconstruction error (NRE), which is defined as the energy of the K residual matrices

$$\mathrm{NRE} = \frac{\sum_{k=1}^K \|\boldsymbol{R}_k^i\|_{\mathrm{F}}^2}{\sum_{k=1}^K \|\boldsymbol{R}_k^0\|_{\mathrm{F}}^2} \le \delta \tag{4.13}$$

is adopted instead as the stopping criterion, where $\delta > 0$ is the tolerance. In Section 4.4, we will prove that the sequence of the reconstruction error $\left\{\sum_{k=1}^K \|\boldsymbol{R}_k^i\|_{\mathrm{F}}^2\right\}_i$ converges to zero with an exponential rate. Therefore, (4.13) is well defined for any $\delta > 0$.

Algorithm 5 GP for LRAMM

```
Input: Matrices $\{\boldsymbol{A}_k\}_{k=1}^K$ and target rank $r$.
  Initialization: Set residual $\boldsymbol{R}_k^0 = \boldsymbol{A}_k$ for $k = 1, \cdots, K$.
  for $i = 1, 2, \cdots, r$ do
      Solve rank-one approximation
          $\left(\boldsymbol{u}_i, \boldsymbol{v}_i, \boldsymbol{s}^i\right) = \arg\min_{\|\boldsymbol{u}\|=\|\boldsymbol{v}\|=1,\boldsymbol{s}} \sum_{k=1}^{K}\left\|s_k\boldsymbol{u}\boldsymbol{v}^{\mathrm{T}} - \boldsymbol{R}_k^{i-1}\right\|_{\mathrm{F}}^2.$   (4.14)
      Update residual
          $\boldsymbol{R}_k^i = \boldsymbol{R}_k^{i-1} - s_{k,i}\boldsymbol{u}_i\boldsymbol{v}_i^{\mathrm{T}},\ k = 1, \cdots, K.$   (4.15)
  end for
Output: $\boldsymbol{U} = [\boldsymbol{u}_1, \cdots, \boldsymbol{u}_r]$, $\boldsymbol{V} = [\boldsymbol{v}_1, \cdots, \boldsymbol{v}_r]$, and $\{\boldsymbol{s}^i\}_{i=1}^r$.
```

The remaining problem is how to efficiently solve the rank-one approximation of multiple matrices, which is described in the next subsection.

4.3.2 Solution to Rank-One Fitting

For the purpose of simplicity, we omit the superscript $(\cdot)^{i-1}$ of $\boldsymbol{R}_k^{i-1}$ and rewrite (4.12) as

$$\min_{\|\boldsymbol{u}\|=\|\boldsymbol{v}\|=1,\boldsymbol{s}} \sum_{k=1}^{K}\left\|s_k\boldsymbol{u}\boldsymbol{v}^{\mathrm{T}} - \boldsymbol{R}_k\right\|_{\mathrm{F}}^2 \tag{4.16}$$

which is not easy to solve since the unit-norm constraints are nonconvex and the product term of the objective function $s_k \boldsymbol{u}\boldsymbol{v}^{\mathrm{T}}$ is also nonconvex. We first eliminate $\boldsymbol{s}$ to obtain an optimization problem only with $\boldsymbol{u}$ and $\boldsymbol{v}$. The kth term of (4.16) is computed as

$$\begin{aligned} f_k(\boldsymbol{u},\boldsymbol{v},s_k) &= \left\| s_k \boldsymbol{u}\boldsymbol{v}^{\mathrm{T}} - \boldsymbol{R}_k \right\|_{\mathrm{F}}^2 \\ &= s_k^2 \|\boldsymbol{u}\boldsymbol{v}^{\mathrm{T}}\|_{\mathrm{F}}^2 - 2s_k \mathrm{tr}\big(\boldsymbol{v}\boldsymbol{u}^{\mathrm{T}}\boldsymbol{R}_k\big) + \|\boldsymbol{R}_k\|_{\mathrm{F}}^2 \\ &= s_k^2 - 2\big(\boldsymbol{u}^{\mathrm{T}}\boldsymbol{R}_k\boldsymbol{v}\big) s_k + \|\boldsymbol{R}_k\|_{\mathrm{F}}^2 \end{aligned} \tag{4.17}$$

where we have exploited $\|\boldsymbol{A}\|_{\mathrm{F}}^2 = \mathrm{tr}(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A})$ and $\mathrm{tr}(\boldsymbol{A}\boldsymbol{B}) = \mathrm{tr}(\boldsymbol{B}\boldsymbol{A})$ as well as the fact $\|\boldsymbol{u}\boldsymbol{v}^{\mathrm{T}}\|_{\mathrm{F}}^2 = \mathrm{tr}\big(\boldsymbol{v}\boldsymbol{u}^{\mathrm{T}}\boldsymbol{u}\boldsymbol{v}^{\mathrm{T}}\big) = \|\boldsymbol{u}\|^2\|\boldsymbol{v}\|^2 = 1$. Apparently, $\{s_k\}_{k=1}^K$ are decoupled with each other and can be solved independently. For fixed $\boldsymbol{u}$ and $\boldsymbol{v}$, the optimal s_k minimizing f_k is given by

$$\frac{\partial f_k}{\partial s_k} = 2\big(s_k - \boldsymbol{u}^{\mathrm{T}}\boldsymbol{R}_k\boldsymbol{v}\big) = 0$$

yielding

$$s_k^\star = \boldsymbol{u}^{\mathrm{T}}\boldsymbol{R}_k\boldsymbol{v}. \tag{4.18}$$

Plugging (4.18) back into (4.17) yields the minimum value of the objective function

$$\left\{ \min_{s_k} f_k(\boldsymbol{u},\boldsymbol{v},s_k) \right\} = \|\boldsymbol{R}_k\|_{\mathrm{F}}^2 - \big(\boldsymbol{u}^{\mathrm{T}}\boldsymbol{R}_k\boldsymbol{v}\big)^2. \tag{4.19}$$

Then, we obtain the following problem with $\boldsymbol{s}$ being eliminated

$$\min_{\|\boldsymbol{u}\|=\|\boldsymbol{v}\|=1} \left\{ \sum_{k=1}^K \|\boldsymbol{R}_k\|_{\mathrm{F}}^2 - \sum_{k=1}^K \big(\boldsymbol{u}^{\mathrm{T}}\boldsymbol{R}_k\boldsymbol{v}\big)^2 \right\} \tag{4.20}$$

where the reconstruction error $\sum_{k=1}^K \|\boldsymbol{R}_k\|_{\mathrm{F}}^2$ is a constant at the current iteration. It is clear that (4.20) amounts to

$$\max_{\|\boldsymbol{u}\|=\|\boldsymbol{v}\|=1} \sum_{k=1}^K \big(\boldsymbol{u}^{\mathrm{T}}\boldsymbol{R}_k\boldsymbol{v}\big)^2. \tag{4.21}$$

For $K = 1$, the optimal solution of (4.21) $\boldsymbol{u}^\star$ and $\boldsymbol{v}^\star$ are the left and right singular vectors associated with the largest singular value $\sigma_{\max}(\boldsymbol{R}_1)$, respectively, which can be efficiently computed by the power method with a low complexity of $\mathcal{O}(mn)$. The corresponding maximum of the objective function is $\sigma_{\max}^2(\boldsymbol{R}_1)$. Note that the largest singular value of a matrix is its spectral norm, i.e., $\sigma_{\max}(\boldsymbol{R}_1) = \|\boldsymbol{R}_1\|_2$.

For $K > 1$, we use the alternating maximization strategy to solve (4.21), where the objective function is maximized over one vector while the other vector is fixed. To be more specific, at the $(j+1)$th $(j = 0, 1, \cdots)$ iteration, $\boldsymbol{u}$ and $\boldsymbol{v}$ are alternately maximized:

$$\boldsymbol{u}^{j+1} = \arg\max_{\|\boldsymbol{u}\|=1} \sum_{k=1}^{K} \left(\boldsymbol{u}^{\mathrm{T}} \boldsymbol{R}_k \boldsymbol{v}^j\right)^2 \tag{4.22}$$

$$\boldsymbol{v}^{j+1} = \arg\max_{\|\boldsymbol{v}\|=1} \sum_{k=1}^{K} \left(\left(\boldsymbol{u}^{j+1}\right)^{\mathrm{T}} \boldsymbol{R}_k \boldsymbol{v}\right)^2. \tag{4.23}$$

It is easy to rewrite (4.22) as

$$\max_{\|\boldsymbol{u}\|=1} \boldsymbol{u}^{\mathrm{T}} \left(\sum_{k=1}^{K} \boldsymbol{R}_k \boldsymbol{v}^j \left(\boldsymbol{R}_k \boldsymbol{v}^j\right)^{\mathrm{T}} \right) \boldsymbol{u} \tag{4.24}$$

whose optimal solution is the unit-norm eigenvector corresponding to the maximum eigenvalue of the positive definite matrix $\sum_{k=1}^{K} \boldsymbol{R}_k \boldsymbol{v}^j (\boldsymbol{R}_k \boldsymbol{v}^j)^{\mathrm{T}}$. Similarly, the solution of (4.23) is the unit-norm eigenvector associated with the maximum eigenvalue of $\sum_{k=1}^{K} \boldsymbol{R}_k^{\mathrm{T}} \boldsymbol{u}^{j+1} \left(\boldsymbol{R}_k^{\mathrm{T}} \boldsymbol{u}^{j+1}\right)^{\mathrm{T}}$. Since (4.21) is nonconvex, the final convergence result relies on the initial values $\boldsymbol{u}^0$ and $\boldsymbol{v}^0$. Appropriate selection of the initial values is important. Suppose that the k_mth ($k_m \in \{1, \cdots, K\}$) matrix $\boldsymbol{R}_{k_m}$ has the maximum spectral norm. That is,

$$\|\boldsymbol{R}_{k_m}\|_2 = \max_{1 \le k \le K} \|\boldsymbol{R}_k\|_2. \tag{4.25}$$

We use the principal singular vectors of $\boldsymbol{R}_{k_m}$ as the initial value

$$\boldsymbol{u}^0 = \mathrm{LSV}_{\max}(\boldsymbol{R}_{k_m}),\ \boldsymbol{v}^0 = \mathrm{RSV}_{\max}(\boldsymbol{R}_{k_m}). \tag{4.26}$$

The algorithm for solving the rank-one approximation of multiple matrices of (4.16) is summarized in Algorithm 6.

Computational Complexity: The initialization needs to compute the largest singular values of K matrices and the principal singular vectors of one matrix, whose complexity is $\mathcal{O}(mnK)$ if the power method is employed. The matrix-vector products $\boldsymbol{R}_k \boldsymbol{v}^j$ and $\boldsymbol{R}_k^{\mathrm{T}} \boldsymbol{u}^{j+1}$ require a cost of $\mathcal{O}(mn)$. The costs of computing the outer products $\boldsymbol{R}_k \boldsymbol{v}^j (\boldsymbol{R}_k \boldsymbol{v}^j)^{\mathrm{T}}$ and $\boldsymbol{R}_k^{\mathrm{T}} \boldsymbol{u}^{j+1} \left(\boldsymbol{R}_k^{\mathrm{T}} \boldsymbol{u}^{j+1}\right)^{\mathrm{T}}$ are $\mathcal{O}(m^2)$ and $\mathcal{O}(n^2)$, respectively. Thus, forming the matrices in (4.27) and (4.28) requires $\mathcal{O}((mn + m^2)K)$ and $\mathcal{O}((mn + n^2)K)$ operations, respectively. Calculation of principal eigenvectors of (4.27) and (4.28) needs $\mathcal{O}(m^2)$ and $\mathcal{O}(n^2)$. In summary, the per-iteration complexity of rank-one approximation of Algorithm 6 is $\mathcal{O}((\max(m,n))^2 K)$ and the total complexity is $\mathcal{O}((\max(m,n))^2 K N_{\mathrm{iter}})$, where N_{iter} is the number of iterations required for convergence. From the simulation results, it is observed that Algorithm 6 converges fast. Typically, several tens of iterations are enough to converge with high accuracy. Also, N_{iter} can be viewed as a constant independent of the dimension. Then, it is clear

Algorithm 6 Rank-One Approximation of Multiple Matrices

```
Input: Matrices $\{\boldsymbol{R}_k\}_{k=1}^{K}$.
  Initialization: Determine $k_m$ by
      $k_m = \arg\max_{1\le k\le K} \|\boldsymbol{R}_k\|_2.$
  Set $\boldsymbol{u}^0 = \mathrm{LSV}_{\max}(\boldsymbol{R}_{k_m})$ and $\boldsymbol{v}^0 = \mathrm{RSV}_{\max}(\boldsymbol{R}_{k_m})$.
  for $j = 0, 1, 2, \cdots$ do
      Update $\boldsymbol{u}$ and $\boldsymbol{v}$:
          $\boldsymbol{u}^{j+1} = \mathrm{EV}_{\max}\left(\sum_{k=1}^{K} \boldsymbol{R}_k \boldsymbol{v}^j \left(\boldsymbol{R}_k \boldsymbol{v}^j\right)^{\mathrm{T}}\right)$   (4.27)
          $\boldsymbol{v}^{j+1} = \mathrm{EV}_{\max}\left(\sum_{k=1}^{K} \boldsymbol{R}_k^{\mathrm{T}} \boldsymbol{u}^{j+1} \left(\boldsymbol{R}_k^{\mathrm{T}} \boldsymbol{u}^{j+1}\right)^{\mathrm{T}}\right)$   (4.28)
      Stop until convergence satisfies.
  end for
  $s_k^\star = (\boldsymbol{u}^{j+1})^{\mathrm{T}} \boldsymbol{R}_k \boldsymbol{v}^{j+1}$, $k = 1, \cdots, K$.
Output: $\boldsymbol{u}^\star = \boldsymbol{u}^{j+1}$, $\boldsymbol{v}^\star = \boldsymbol{v}^{j+1}$, and $\boldsymbol{s}^\star = [s_1^\star, \cdots, s_K^\star]^{\mathrm{T}}$.
```

that the complexity of the GP for LRAMM of Algorithm 5 is $\mathcal{O}((\max(m,n))^2 rKN_{\mathrm{iter}})$. When m and n have the same order-of-magnitude, say, $\mathcal{O}(m) \sim \mathcal{O}(n)$, it follows $\mathcal{O}((\max(m,n))^2) \sim \mathcal{O}(mn)$. The complexity is approximately $\mathcal{O}(mnrKN_{\mathrm{iter}})$. This implies that the complexity of the GP algorithm is linear with respect to the matrix size mn and the number of matrices K. Hence, it is scalable to problem size.

4.3.3 Economic Greedy Pursuit

The dominant cost of the GP is the iterative procedure in Algorithm 6 for solving the rank-one fitting problem. To reduce the complexity, we develop an economic version of the GP, namely, the EGP, which is listed in Algorithm 7. It just takes the initial values of (4.25) and (4.26), i.e., the principal singular value/vectors of the matrix having the maximum spectral norm, as an approximate solution to the rank-one fitting. It discards the time-consuming iterative procedure. Surprisingly, using the inexact solution of (4.25) and (4.26) also makes the EGP converge. The convergence of the EGP will be proved in Section 4.4. Clearly, the complexity of the EGP is reduced to $\mathcal{O}(mnrK)$. However, exploiting the inexact solution to the rank-one fitting results in a convergence rate slower than the GP in general.

Algorithm 7 EGP for LRAMM

```
Input: Matrices $\{\boldsymbol{A}_k\}_{k=1}^K$ and target rank $r$.
  Initialization: Set residual $\boldsymbol{R}_k^0 = \boldsymbol{A}_k$ for $k = 1, \cdots, K$.
  for $i = 1, 2, \cdots, r$ do
      Determine $k_m = \arg \max_{1 \le k \le K} \|\boldsymbol{R}_k^{i-1}\|_2$ and set
          $\boldsymbol{u}_i = \mathrm{LSV}_{\max}\big(\boldsymbol{R}_{k_m}^{i-1}\big), \ \boldsymbol{v}_i = \mathrm{RSV}_{\max}\big(\boldsymbol{R}_{k_m}^{i-1}\big),$
      and $s_{k,i} = \boldsymbol{u}_i^{\mathrm{T}} \boldsymbol{R}_k^{i-1} \boldsymbol{v}_i$ for $k = 1, \cdots, K$.
      Update residual
          $\boldsymbol{R}_k^i = \boldsymbol{R}_k^{i-1} - s_{k,i} \boldsymbol{u}_i \boldsymbol{v}_i^{\mathrm{T}}, \ k = 1, \cdots, K.$
  end for
Output: $\boldsymbol{U} = [\boldsymbol{u}_1, \cdots, \boldsymbol{u}_r]$, $\boldsymbol{V} = [\boldsymbol{v}_1, \cdots, \boldsymbol{v}_r]$, and $\{\boldsymbol{s}^i\}_{i=1}^r$.
```

4.3.4 Orthogonal Greedy Pursuit

From Algorithm 5, we see that once a rank-one solution $(\boldsymbol{u}_i, \boldsymbol{v}_i, \boldsymbol{s}^i)$ is obtained, it is never revisited and hence remains unchanged all the time. The OGP is a modification to the GP. Analogous to the OMP for sparse recovery, the OGP still keeps $(\boldsymbol{u}_i, \boldsymbol{v}_i)$ unchanged but re-computes all coefficients $\{s_{k,l}\}$ by least squares, which achieves so-called "orthogonalization" to the coefficients. To be more specific, after obtaining $\{\boldsymbol{u}_l\}_{l=1}^i$ and $\{\boldsymbol{v}_l\}_{l=1}^i$ at the ith iteration, $\{s_{k,l}\}$ are re-computed by

$$\min_{\{s_{k,l}\}} \sum_{k=1}^K \left\| \sum_{l=1}^i s_{k,l} \boldsymbol{u}_l \boldsymbol{v}_l^{\mathrm{T}} - \boldsymbol{A}_k \right\|_{\mathrm{F}}^2 \tag{4.29}$$

which can be decomposed into the following K independent minimization problems:

$$\min_{s_{k,1}, \cdots, s_{k,i}} \left\| \sum_{l=1}^i s_{k,l} \boldsymbol{u}_l \boldsymbol{v}_l^{\mathrm{T}} - \boldsymbol{A}_k \right\|_{\mathrm{F}}^2 \tag{4.30}$$

for $k = 1, \cdots, K$, since $\{s_{k,l}\}$ can be separably solved with respect to k. We further define the vector

$$\boldsymbol{s}_k = [s_{k,1}, \cdots, s_{k,i}]^{\mathrm{T}} \in \mathbb{R}^i \tag{4.31}$$

which should be distinguished from $\boldsymbol{s}^i \in \mathbb{R}^K$ in (4.5). Obviously, if $r = 1$, the OGP is the same as the GP because both of them are just a rank-one approximation problem. Therefore, we only need to discuss the case of $r \geq 2$ for the OGP.

We now derive the solution of (4.29). Recalling $\boldsymbol{a}_k = \mathrm{vec}(\boldsymbol{A}_k) \in \mathbb{R}^{mn}$ and defining

$\boldsymbol{b}_l = \mathrm{vec}\big(\boldsymbol{u}_l\boldsymbol{v}_l^{\mathrm{T}}\big) \in \mathbb{R}^{mn}$, (4.30) amounts to

$$\min_{\boldsymbol{s}_k}\left\{\left\|\sum_{l=1}^{i} s_{k,l}\boldsymbol{b}_l - \boldsymbol{a}_k\right\|^2 = \|\boldsymbol{B}_i\boldsymbol{s}_k - \boldsymbol{a}_k\|^2\right\} \tag{4.32}$$

where

$$\boldsymbol{B}_i = [\boldsymbol{b}_1, \cdots, \boldsymbol{b}_i] \in \mathbb{R}^{mn\times i}. \tag{4.33}$$

It is worth pointing out that there is no need to convert the matrices in practice and we just use the vectors for derivation. Note that the column number of $\boldsymbol{B}_i$ is the iteration number i and varies during the iteration process. The solution of (4.32) is

$$\boldsymbol{s}_k = \big(\boldsymbol{B}_i^{\mathrm{T}}\boldsymbol{B}_i\big)^{-1}\boldsymbol{B}_i^{\mathrm{T}}\boldsymbol{a}_k. \tag{4.34}$$

However, we do not compute $\boldsymbol{s}_k$ by the direct use of (4.34) since it involves matrix multiplication and inversion, which is computationally expensive. Instead, a recursive calculation is exploited to reduce the complexity. It is clear that $\boldsymbol{B}_i^{\mathrm{T}}\boldsymbol{a}_k$ can be recursively calculated as

$$\boldsymbol{c}_k^i \triangleq \boldsymbol{B}_i^{\mathrm{T}}\boldsymbol{a}_k = \begin{bmatrix} \boldsymbol{B}_{i-1}^{\mathrm{T}}\boldsymbol{a}_k \\ \boldsymbol{b}_i^{\mathrm{T}}\boldsymbol{a}_k \end{bmatrix} = \begin{bmatrix} \boldsymbol{c}_k^{i-1} \\ \boldsymbol{u}_i^{\mathrm{T}}\boldsymbol{A}_k\boldsymbol{v}_i \end{bmatrix} \tag{4.35}$$

where $\boldsymbol{c}_k^{i-1} = \boldsymbol{B}_{i-1}^{\mathrm{T}}\boldsymbol{a}_k \in \mathbb{R}^{i-1}$ is the result of the $(i-1)$th iteration, and can be employed in the current iteration. At the beginning of the iteration, we set $\boldsymbol{c}_k^0 = \emptyset$. Note that $\boldsymbol{b}_i^{\mathrm{T}}\boldsymbol{a}_k$ can be computed as

$$\begin{aligned} \boldsymbol{b}_i^{\mathrm{T}}\boldsymbol{a}_k &= \langle \boldsymbol{b}_i, \boldsymbol{a}_k\rangle = \big\langle \boldsymbol{u}_i\boldsymbol{v}_i^{\mathrm{T}}, \boldsymbol{A}_k\big\rangle \\ &= \mathrm{tr}\big(\boldsymbol{v}_i\boldsymbol{u}_i^{\mathrm{T}}\boldsymbol{A}_k\big) = \boldsymbol{u}_i^{\mathrm{T}}\boldsymbol{A}_k\boldsymbol{v}_i \end{aligned} \tag{4.36}$$

where we have used the fact that $\langle \mathrm{vec}(\boldsymbol{A}), \mathrm{vec}(\boldsymbol{B})\rangle = \langle \boldsymbol{A}, \boldsymbol{B}\rangle$ satisfies for two arbitrary matrices $\boldsymbol{A}$ and $\boldsymbol{B}$. When $\boldsymbol{c}_k^{i-1}$ is available, the cost to obtain $\boldsymbol{c}_k^i$ is computing $\boldsymbol{b}_i^{\mathrm{T}}\boldsymbol{a}_k$, which requires $\mathcal{O}(mn)$ operations.

Next, we discuss how to recursively compute $\big(\boldsymbol{B}_i^{\mathrm{T}}\boldsymbol{B}_i\big)^{-1}$. Obviously, $\boldsymbol{B}_i^{\mathrm{T}}\boldsymbol{B}_i$ is the Gram matrix of the vectors $\{\boldsymbol{b}_1, \cdots, \boldsymbol{b}_i\}$ and is denoted as $\boldsymbol{G}_i = \boldsymbol{B}_i^{\mathrm{T}}\boldsymbol{B}_i \in \mathbb{R}^{i\times i}$. The $\boldsymbol{G}_i$ and $\boldsymbol{G}_{i-1}$ are related via

$$\begin{aligned} \boldsymbol{G}_i = \boldsymbol{B}_i^{\mathrm{T}}\boldsymbol{B}_i &= \begin{bmatrix} \boldsymbol{B}_{i-1}^{\mathrm{T}} \\ \boldsymbol{b}_i^{\mathrm{T}} \end{bmatrix}[\boldsymbol{B}_{i-1}, \boldsymbol{b}_i] \\ &= \begin{bmatrix} \boldsymbol{B}_{i-1}^{\mathrm{T}}\boldsymbol{B}_{i-1} & \boldsymbol{B}_{i-1}^{\mathrm{T}}\boldsymbol{b}_i \\ \boldsymbol{b}_i^{\mathrm{T}}\boldsymbol{B}_{i-1} & \boldsymbol{b}_i^{\mathrm{T}}\boldsymbol{b}_i \end{bmatrix} \\ &= \begin{bmatrix} \boldsymbol{G}_{i-1} & \boldsymbol{g}_{i-1} \\ \boldsymbol{g}_{i-1}^{\mathrm{T}} & 1 \end{bmatrix} \end{aligned} \tag{4.37}$$

where

$$\boldsymbol{g}_{i-1} = \boldsymbol{B}_{i-1}^{\mathrm{T}} \boldsymbol{b}_i \in \mathbb{R}^{i-1} \tag{4.38}$$

and

$$\begin{aligned} \boldsymbol{b}_i^{\mathrm{T}} \boldsymbol{b}_i &= \langle \boldsymbol{b}_i, \boldsymbol{b}_i \rangle = \left\langle \boldsymbol{u}_i \boldsymbol{v}_i^{\mathrm{T}}, \boldsymbol{u}_i \boldsymbol{v}_i^{\mathrm{T}} \right\rangle = \operatorname{tr}\left(\boldsymbol{v}_i \boldsymbol{u}_i^{\mathrm{T}} \boldsymbol{u}_i \boldsymbol{v}_i^{\mathrm{T}} \right) \\ &= \|\boldsymbol{u}_i\|^2 \|\boldsymbol{v}_i\|^2 = 1 \end{aligned} \tag{4.39}$$

are used. Denoting

$$\boldsymbol{h}_{i-1} = \boldsymbol{G}_{i-1}^{-1} \boldsymbol{g}_{i-1} \text{ and } \beta = \frac{1}{1 - \boldsymbol{g}_{i-1}^{\mathrm{T}} \boldsymbol{h}_{i-1}} \tag{4.40}$$

we get the inverse of $\boldsymbol{G}_i$ as

$$\boldsymbol{G}_i^{-1} = \begin{bmatrix} \boldsymbol{G}_{i-1}^{-1} + \beta \boldsymbol{h}_{i-1} \boldsymbol{h}_{i-1}^{\mathrm{T}} & -\beta \boldsymbol{h}_{i-1} \\ -\beta \boldsymbol{h}_{i-1}^{\mathrm{T}} & \beta \end{bmatrix} \tag{4.41}$$

with the use of the block matrix inversion formula [140]. Again, the result of the $(i-1)$th iteration $\boldsymbol{G}_{i-1}^{-1}$ is already available for the current iteration. The initial value is set $\boldsymbol{G}_0^{-1} = \emptyset$ while $\boldsymbol{G}_1^{-1} = 1$ at the first iteration. For $i \geq 2$, $\boldsymbol{G}_i^{-1}$ is recursively calculated by (4.41). We only need to compute $\boldsymbol{g}_{i-1}$, whose pth ($p = 1, \cdots, i-1$) entry is

$$\begin{aligned} [\boldsymbol{g}_{i-1}]_p &= \boldsymbol{b}_p^{\mathrm{T}} \boldsymbol{b}_i = \langle \boldsymbol{b}_p, \boldsymbol{b}_i \rangle = \left\langle \boldsymbol{u}_p \boldsymbol{v}_p^{\mathrm{T}}, \boldsymbol{u}_i \boldsymbol{v}_i^{\mathrm{T}} \right\rangle \\ &= \operatorname{tr}\left(\boldsymbol{v}_p \boldsymbol{u}_p^{\mathrm{T}} \boldsymbol{u}_i \boldsymbol{v}_i^{\mathrm{T}} \right) = \left(\boldsymbol{u}_p^{\mathrm{T}} \boldsymbol{u}_i \right) \left(\boldsymbol{v}_p^{\mathrm{T}} \boldsymbol{v}_i \right) \end{aligned} \tag{4.42}$$

which only requires a complexity of $\mathcal{O}(m+n)$ rather than $\mathcal{O}(mn)$. Then, we have

$$\boldsymbol{g}_{i-1} = \left[\left(\boldsymbol{u}_1^{\mathrm{T}} \boldsymbol{u}_i \right) \left(\boldsymbol{v}_1^{\mathrm{T}} \boldsymbol{v}_i \right), \cdots, \left(\boldsymbol{u}_{i-1}^{\mathrm{T}} \boldsymbol{u}_i \right) \left(\boldsymbol{v}_{i-1}^{\mathrm{T}} \boldsymbol{v}_i \right) \right]^{\mathrm{T}}. \tag{4.43}$$

Computing $\boldsymbol{g}_{i-1}$ costs $\mathcal{O}(i(m+n))$. It is either lower than or similar to the cost of computing $\boldsymbol{b}_i^{\mathrm{T}} \boldsymbol{a}_k$ in (4.36), namely, $\mathcal{O}(mn)$ due to $i < r$ and $r \sim \max(m, n)$ in general. In summary, the dominant computational load for re-computation of the coefficients in the OGP is calculating $\left\{ \boldsymbol{b}_i^{\mathrm{T}} \boldsymbol{a}_k \right\}_{k=1}^{K}$, which is $\mathcal{O}(mnK)$.

The OGP for LRAMM is summarized in Algorithm 8. The compression ratios and computational complexities of the PCA, 2DPCA, GLRAM, GP/OGP and EGP are compared in Table I.

4.3.5 Feature Extraction for Pattern Classification

In addition to the direct application to data compression, the greedy pursuit algorithms can also be applied to feature extraction for classification. Suppose we have obtained a set of basis matrices $\left\{ \boldsymbol{u}_i \boldsymbol{v}_i^{\mathrm{T}} \right\}_{i=1}^{r}$ from the training data $\{\boldsymbol{A}_k\}_{k=1}^{K}$ using the GP, EGP or OGP. In the training stage, we do not use the class label information of the training

Algorithm 8 OGP for LRAMM

```
Input: Matrices $\{\boldsymbol{A}_k\}_{k=1}^{K}$ and target rank $r$.
  Initialization: Set residual $\boldsymbol{R}_k^0 = \boldsymbol{A}_k$ for $k = 1, \cdots, K$.
  For $i = 1$, use Algorithm 1 to obtain $(\boldsymbol{u}_1, \boldsymbol{v}_1)$ and $\{\boldsymbol{R}_k^1\}_{k=1}^{K}$.
  Set $\boldsymbol{G}_1^{-1} = 1$ and $\boldsymbol{c}_k^1 = \boldsymbol{u}_i^{\mathrm{T}} \boldsymbol{A}_k \boldsymbol{v}_i$ for $k = 1, \cdots, K$.
  for $i = 2, 3, \cdots, r$ do
      Solve rank-one approximation
          $$(\boldsymbol{u}_i, \boldsymbol{v}_i) = \underset{\|\boldsymbol{u}\|=\|\boldsymbol{v}\|=1, \boldsymbol{s}}{\arg\min} \sum_{k=1}^{K} \left\| s_k \boldsymbol{u}\boldsymbol{v}^{\mathrm{T}} - \boldsymbol{R}_k^{i-1} \right\|_{\mathrm{F}}^2.$$
      Compute
          $\boldsymbol{g}_{i-1} = \left[ (\boldsymbol{u}_1^{\mathrm{T}} \boldsymbol{u}_i)(\boldsymbol{v}_1^{\mathrm{T}} \boldsymbol{v}_i), \cdots, (\boldsymbol{u}_{i-1}^{\mathrm{T}} \boldsymbol{u}_i)(\boldsymbol{v}_{i-1}^{\mathrm{T}} \boldsymbol{v}_i) \right]^{\mathrm{T}}$
          $\boldsymbol{h}_{i-1} = \boldsymbol{G}_{i-1}^{-1} \boldsymbol{g}_{i-1}$
          $\beta = 1 / \left(1 - \boldsymbol{g}_{i-1}^{\mathrm{T}} \boldsymbol{h}_{i-1}\right)$
          $\boldsymbol{G}_i^{-1} = \begin{bmatrix} \boldsymbol{G}_{i-1}^{-1} + \beta \boldsymbol{h}_{i-1} \boldsymbol{h}_{i-1}^{\mathrm{T}} & -\beta \boldsymbol{h}_{i-1} \\ -\beta \boldsymbol{h}_{i-1}^{\mathrm{T}} & \beta \end{bmatrix}$
          $\boldsymbol{c}_k^i = \begin{bmatrix} \boldsymbol{c}_k^{i-1} \\ \boldsymbol{u}_i^{\mathrm{T}} \boldsymbol{A}_k \boldsymbol{v}_i \end{bmatrix}, \ k = 1, \cdots, K$
          $[s_{k,1}, \cdots, s_{k,i}]^{\mathrm{T}} = \boldsymbol{G}_i^{-1} \boldsymbol{c}_k^i, \ k = 1, \cdots, K.$
      Calculate residual
          $$\boldsymbol{R}_k^i = \boldsymbol{A}_k - \sum_{l=1}^{i} s_{k,l} \boldsymbol{u}_l \boldsymbol{v}_l^{\mathrm{T}}, \ k = 1, \cdots, K. \tag{4.44}$$
  end for
Output: $\boldsymbol{U} = [\boldsymbol{u}_1, \cdots, \boldsymbol{u}_r]$, $\boldsymbol{V} = [\boldsymbol{v}_1, \cdots, \boldsymbol{v}_r]$, and $\{s_{k,l}\}$ with $1 \le k \le K, 1 \le l \le r$.
```

Table 4.1: Compression ratio and computational complexity.

Methods	Compression ratio	Complexity
PCA	$\frac{mnK}{(mn+K+1)r}$	$\mathcal{O}(\max(m^2n^2, K^2)r)$
2DPCA	$\frac{mnK}{(mK+n)r}$	$\mathcal{O}(mn^2K)$
GLRAM	$\frac{mnK}{(m+n+Kr)r}$	$\mathcal{O}((m+n)^2 rKN_{\mathrm{iter}})$
GP/OGP	$\frac{mnK}{(m+n+K)r}$	$\mathcal{O}((\max(m,n))^2 rKN_{\mathrm{iter}})$
EGP	$\frac{mnK}{(m+n+K)r}$	$\mathcal{O}(mnrK)$

data. Thus, the greedy pursuit belongs to unsupervised learning. The tested matrix $\boldsymbol{Z}$ is deemed to have the similar low-rank structure as the training matrices $\{\boldsymbol{A}_k\}_{k=1}^K$. We do not handle the original matrices any more but use the following reconstructions instead

$$\widehat{\boldsymbol{A}}_k = \sum_{i=1}^{r} s_{k,i}\boldsymbol{u}_i\boldsymbol{v}_i^{\mathrm{T}},\ \widehat{\boldsymbol{Z}} = \sum_{i=1}^{r} s_{z,i}\boldsymbol{u}_i\boldsymbol{v}_i^{\mathrm{T}} \tag{4.45}$$

where $\{s_{z,i}\}$ is the solution of $\min\left\|\sum_{i=1}^{r} s_i^z\boldsymbol{u}_i\boldsymbol{v}_i^{\mathrm{T}} - \boldsymbol{Z}\right\|_{\mathrm{F}}^2$. Like (4.34), this solution is expressed as

$$\boldsymbol{s}_z = \boldsymbol{G}_r^{-1}\boldsymbol{B}_r^{\mathrm{T}}\boldsymbol{z} \tag{4.46}$$

where $\boldsymbol{s}_z = [s_{z,1}, \cdots, s_{z,r}]^{\mathrm{T}}$ collects the r coefficients, $\boldsymbol{z} = \mathrm{vec}(\boldsymbol{Z})$, $\boldsymbol{B}_r$ has the same form as (4.33) with $i = r$, and $\boldsymbol{G}_r = \boldsymbol{B}_r\boldsymbol{B}_r^{\mathrm{T}}$ is the Gram matrix of the basis matrices. The distance between $\widehat{\boldsymbol{Z}}$ and $\widehat{\boldsymbol{A}}_k$ is taken as the similarity measure after rank reduction, which is computed as

$$\begin{aligned}\left\|\widehat{\boldsymbol{Z}} - \widehat{\boldsymbol{A}}_k\right\|_{\mathrm{F}}^2 &= \left\|\mathrm{vec}(\widehat{\boldsymbol{Z}} - \widehat{\boldsymbol{A}}_k)\right\|^2 = \|\boldsymbol{B}_r(\boldsymbol{s}_z - \boldsymbol{s}_k)\|^2 \\ &= (\boldsymbol{s}_z - \boldsymbol{s}_k)^{\mathrm{T}}\boldsymbol{G}_r(\boldsymbol{s}_z - \boldsymbol{s}_k) = \left\|\boldsymbol{G}_r^{\frac{1}{2}}(\boldsymbol{s}_z - \boldsymbol{s}_k)\right\|^2\end{aligned} \tag{4.47}$$

where $\boldsymbol{G}_r^{\frac{1}{2}} \in \mathbb{R}^{r\times r}$ is the square root matrix of $\boldsymbol{G}_r$. Thus, $\boldsymbol{G}_r^{\frac{1}{2}}\boldsymbol{s}_z \in \mathbb{R}^r$ can be viewed as the "feature vector" of $\boldsymbol{Z}$ extracted by the greedy algorithm. Since we have $\boldsymbol{s}_k = \boldsymbol{G}_r^{-1}\boldsymbol{B}_r^{\mathrm{T}}\boldsymbol{a}_k$ for the OGP, (4.47) is further simplified to

$$\left\|\widehat{\boldsymbol{Z}} - \widehat{\boldsymbol{A}}_k\right\|_{\mathrm{F}}^2 = (\boldsymbol{d}_z - \boldsymbol{d}_k)^{\mathrm{T}}\boldsymbol{G}_r^{-1}(\boldsymbol{d}_z - \boldsymbol{d}_k) \tag{4.48}$$

where

$$\boldsymbol{d}_z = \boldsymbol{B}_r^{\mathrm{T}}\boldsymbol{z} = \left[\boldsymbol{u}_1^{\mathrm{T}}\boldsymbol{Z}\boldsymbol{v}_1, \cdots, \boldsymbol{u}_r^{\mathrm{T}}\boldsymbol{Z}\boldsymbol{v}_r\right]^{\mathrm{T}}. \tag{4.49}$$

The $\boldsymbol{d}_k$ has similar expression as $\boldsymbol{d}_z$. Noting that the OGP already outputs $\boldsymbol{G}_r^{-1}$, only $\boldsymbol{d}_z$ and $\boldsymbol{d}_k$ need to calculate. The feature vector of $\boldsymbol{Z}$ extracted by the OGP can also be expressed as $\boldsymbol{G}_r^{-\frac{1}{2}}\boldsymbol{d}_z$. In the test stage, the nearest neighbor (NN) classifier is employed. The tested sample is assigned to the class of its closest neighbor that has the minimum distance of the reconstructed matrices $\widehat{\boldsymbol{Z}}$ and $\widehat{\boldsymbol{A}}_k$. However, we do not need to perform reconstruction in practice. Instead, the distance can be efficiently computed according to (4.47) or (4.48), where only r-dimensional vectors are involved.

4.4 Convergence Analysis

4.4.1 Key Lemma

Prior to presenting the convergence results of the GP and OGP, we prove the following key lemma which facilitates the convergence analysis.

Lemma 1 *At the ith iteration, when applying Algorithm 6 to the rank-one approximation of matrices* $\{\boldsymbol{R}_k^{i-1}\}_{k=1}^{K}$*, the objective function of the subproblem* (4.21) *at the solution* $(\boldsymbol{u}_i, \boldsymbol{v}_i)$ *is guaranteed*

$$\sum_{k=1}^{K} \left(\boldsymbol{u}_i^{\mathrm{T}} \boldsymbol{R}_k^{i-1} \boldsymbol{v}_i\right)^2 \geq \frac{1}{\min(m,n)K} \sum_{k=1}^{K} \|\boldsymbol{R}_k^{i-1}\|_{\mathrm{F}}^2. \tag{4.50}$$

Proof. Note that when we emphasize the iteration number, $(\boldsymbol{u}_i, \boldsymbol{v}_i)$ denotes the solution of rank-one approximation at the ith iteration, as well as the solution of subproblem (4.21) given by Algorithm 6. Because Algorithm 6 adopts the manner of alternating maximization, it non-decreases the objective function and indicates that the objective function at $(\boldsymbol{u}_i, \boldsymbol{v}_i)$ is no less than that at the initial value $(\boldsymbol{u}^0, \boldsymbol{v}^0)$. Hence, we have

$$\begin{aligned}
\sum_{k=1}^{K} \left(\boldsymbol{u}_i^{\mathrm{T}} \boldsymbol{R}_k^{i-1} \boldsymbol{v}_i\right)^2 &\geq \sum_{k=1}^{K} \left((\boldsymbol{u}^0)^{\mathrm{T}} \boldsymbol{R}_k^{i-1} \boldsymbol{v}^0\right)^2 && (4.51)\\
&\geq \max_{1 \leq k \leq K} \left((\boldsymbol{u}^0)^{\mathrm{T}} \boldsymbol{R}_k^{i-1} \boldsymbol{v}^0\right)^2 \\
&= \left((\boldsymbol{u}^0)^{\mathrm{T}} \boldsymbol{R}_{k_m}^{i-1} \boldsymbol{v}^0\right)^2 = \sigma_{\max}^2(\boldsymbol{R}_{k_m}^{i-1}) && (4.52)\\
&\geq \frac{1}{K} \sum_{k=1}^{K} \sigma_{\max}^2(\boldsymbol{R}_k^{i-1}) && (4.53)\\
&\geq \frac{1}{K} \sum_{k=1}^{K} \frac{1}{\mathrm{rank}(\boldsymbol{R}_k^{i-1})} \sum_{l=1}^{\mathrm{rank}(\boldsymbol{R}_k^{i-1})} \sigma_l^2(\boldsymbol{R}_k^{i-1}) \\
&= \frac{1}{K} \sum_{k=1}^{K} \frac{1}{\mathrm{rank}(\boldsymbol{R}_k^{i-1})} \|\boldsymbol{R}_k^{i-1}\|_{\mathrm{F}}^2 && (4.54)\\
&\geq \frac{1}{\min(m,n)K} \sum_{k=1}^{K} \|\boldsymbol{R}_k^{i-1}\|_{\mathrm{F}}^2
\end{aligned}$$

where (4.52) exploits that the initial value $\boldsymbol{u}^0$ and $\boldsymbol{v}^0$ are the principal singular vectors of $\boldsymbol{R}_{k_m}^{i-1}$ and (4.53) follows from that $\boldsymbol{R}_{k_m}^{i-1}$ has the maximum spectral norm, i.e., $\sigma_{\max}^2(\boldsymbol{R}_{k_m}^{i-1}) = \max_k \sigma_{\max}^2(\boldsymbol{R}_k^{i-1})$. Equation (4.54) is based on the fact the square of the Frobenius norm of a matrix equals the sum of the squared singular values and the last inequality is due to $\mathrm{rank}(\boldsymbol{R}_k^{i-1}) \leq \min(m,n)$. □

4.4.2 Convergence Analysis for GP and EGP

Now we are ready to prove the following theorem, which guarantees the convergence of the GP for LRAMM and gives a lower bound on the convergence rate.

Theorem 3 *The reconstruction error, i.e., the energy of the residual matrices of the GP for LRAMM in Algorithm 5 decays exponentially:*[1]

$$\sum_{k=1}^{K} \|\boldsymbol{R}_k^i\|_{\mathrm{F}}^2 \leq \gamma_{\mathrm{GP}}^i \sum_{k=1}^{K} \|\boldsymbol{A}_k\|_{\mathrm{F}}^2 \tag{4.55}$$

for the iteration number $i = 0, 1, 2, \cdots$, *where*

$$\gamma_{\mathrm{GP}} = 1 - \frac{1}{\min(m,n)K} \tag{4.56}$$

satisfying $0 < \gamma_{\mathrm{GP}} < 1$ *is a lower bound of the convergence rate.*

Proof. Starting with the residual update formula in (4.15), it follows that

$$\begin{aligned}
\sum_{k=1}^{K} \|\boldsymbol{R}_k^i\|_{\mathrm{F}}^2 &= \sum_{k=1}^{K} \left\|\boldsymbol{R}_k^{i-1} - s_{k,i}\boldsymbol{u}_i\boldsymbol{v}_i^{\mathrm{T}}\right\|_{\mathrm{F}}^2 \\
&= \min_{\|\boldsymbol{u}\|=\|\boldsymbol{v}\|=1,\boldsymbol{s}} \sum_{k=1}^{K} \left\|s_k\boldsymbol{u}\boldsymbol{v}^{\mathrm{T}} - \boldsymbol{R}_k^{i-1}\right\|_{\mathrm{F}}^2 && (4.57) \\
&= \min_{\|\boldsymbol{u}\|=\|\boldsymbol{v}\|=1} \left\{ \sum_{k=1}^{K} \|\boldsymbol{R}_k^{i-1}\|_{\mathrm{F}}^2 - \sum_{k=1}^{K} \left(\boldsymbol{u}^{\mathrm{T}}\boldsymbol{R}_k^{i-1}\boldsymbol{v}\right)^2 \right\} && (4.58) \\
&= \sum_{k=1}^{K} \|\boldsymbol{R}_k^{i-1}\|_{\mathrm{F}}^2 - \max_{\|\boldsymbol{u}\|=\|\boldsymbol{v}\|=1} \sum_{k=1}^{K} \left(\boldsymbol{u}^{\mathrm{T}}\boldsymbol{R}_k^{i-1}\boldsymbol{v}\right)^2 \\
&= \sum_{k=1}^{K} \|\boldsymbol{R}_k^{i-1}\|_{\mathrm{F}}^2 - \sum_{k=1}^{K} \left(\boldsymbol{u}_i^{\mathrm{T}}\boldsymbol{R}_k^{i-1}\boldsymbol{v}_i\right)^2 && (4.59) \\
&\leq \sum_{k=1}^{K} \|\boldsymbol{R}_k^{i-1}\|_{\mathrm{F}}^2 - \frac{1}{\min(m,n)K} \sum_{k=1}^{K} \|\boldsymbol{R}_k^{i-1}\|_{\mathrm{F}}^2 && (4.60) \\
&= \left(1 - \frac{1}{\min(m,n)K}\right) \sum_{k=1}^{K} \|\boldsymbol{R}_k^{i-1}\|_{\mathrm{F}}^2 \\
&= \gamma_{\mathrm{GP}} \sum_{k=1}^{K} \|\boldsymbol{R}_k^{i-1}\|_{\mathrm{F}}^2
\end{aligned}$$

where (4.57) and (4.59) follow from (4.14), i.e., $(\boldsymbol{u}_i, \boldsymbol{v}_i, \boldsymbol{s}^i)$ is the minimizer of the rank-one approximation or equivalently is the maximizer of (4.21) at the ith iteration. Meanwhile, (4.58) is a reduced result of (4.57) with $\boldsymbol{s}$ being eliminated, which follows

---

[1] In optimization literature, the exponential convergence is also referred to geometric convergence or linear convergence.

from (4.20). The key inequality of (4.60) is according to Lemma 1. Successively applying the above relation, we at once obtain

$$\sum_{k=1}^{K} \|\boldsymbol{R}_k^i\|_{\mathrm{F}}^2 \leq \gamma_{\mathrm{GP}}^i \sum_{k=1}^{K} \|\boldsymbol{R}_k^0\|_{\mathrm{F}}^2 = \gamma_{\mathrm{GP}}^i \sum_{k=1}^{K} \|\boldsymbol{A}_k\|_{\mathrm{F}}^2 \tag{4.61}$$

where we have used $\boldsymbol{R}_k^0 = \boldsymbol{A}_k$ for $k = 1, \cdots, K$. Since the decay ratio satisfies $0 < \gamma_{\mathrm{GP}} < 1$, the reconstruction error strictly decreases at each iteration and the GP algorithm converges with a worst decay rate of γ_{GP}. □

By Theorem 3, it is apparent that the reconstruction error approaches zero:

$$\lim_{i\to\infty} \sum_{k=1}^{K} \|\boldsymbol{R}_k^i\|_{\mathrm{F}}^2 = 0 \tag{4.62}$$

due to $\gamma_{\mathrm{GP}} \in (0, 1)$. This implies that the stopping criterion in (4.13) is well defined for any $\delta > 0$. Obviously, (4.62) also indicates

$$\lim_{i\to\infty} \|\boldsymbol{R}_k^i\|_{\mathrm{F}}^2 = 0, \text{ and } \lim_{i\to\infty} \boldsymbol{R}_k^i = \boldsymbol{0}, \ k = 1, \cdots, K. \tag{4.63}$$

As a direct conclusion obtained from Theorem 1, the following corollary allows an infinite series expansion for an arbitrary set of matrices $\{\boldsymbol{A}_k\}_{k=1}^K$.

Corollary 1 *For any matrix set $\{\boldsymbol{A}_k\}_{k=1}^K$, the GP algorithm leads to an infinite series expansion, which is shown as*

$$\boldsymbol{A}_k = \sum_{i=1}^{\infty} s_{k,i} \boldsymbol{u}_i \boldsymbol{v}_i^{\mathrm{T}}, \ k = 1, \cdots, K \tag{4.64}$$

where $(\boldsymbol{u}_i, \boldsymbol{v}_i, \boldsymbol{s}^i)$ is the result obtained by Algorithm 5 at the ith iteration.

Proof. Successive application of the residual update formula of (4.15) results in

$$\begin{aligned} \boldsymbol{R}_k^i &= \boldsymbol{R}_k^{i-1} - s_{k,i} \boldsymbol{u}_i \boldsymbol{v}_i^{\mathrm{T}} \\ &= \boldsymbol{R}_k^0 - \sum_{l=1}^{i} s_{k,l} \boldsymbol{u}_l \boldsymbol{v}_l^{\mathrm{T}} \\ &= \boldsymbol{A}_k - \sum_{l=1}^{i} s_{k,l} \boldsymbol{u}_l \boldsymbol{v}_l^{\mathrm{T}} \end{aligned}$$

which is rewritten as

$$\boldsymbol{A}_k = \sum_{l=1}^{i} s_{k,l} \boldsymbol{u}_l \boldsymbol{v}_l^{\mathrm{T}} + \boldsymbol{R}_k^i. \tag{4.65}$$

Exploiting $\lim_{i\to\infty} \boldsymbol{R}_k^i = \boldsymbol{0}$ in (4.63) and taking limits as $i \to \infty$ on both sides of (4.65) yields (4.64). $\square$

Remark: Theoretically, the EGP has the same convergence result in Theorem 1. To prove the convergence of the EGP, we just need to modify the "$\geq$" in (4.51) to "=" while other steps remain the same as the GP. It should be pointed out that the GP and EGP merely have the *worst-case* lower bounds of the convergence rates. Their practical convergence rates are different in general. The worst case refers to that there is no improvement using the iterative procedure in Algorithm 6 compared with merely using the initial value of (4.26). This case happens when $\boldsymbol{v}^0$ is in the intersection of the null spaces of $\left\{\boldsymbol{R}_k^{i-1}\right\}_{k\neq k_m}$, resulting in $\boldsymbol{R}_k^{i-1}\boldsymbol{v}^0 = \boldsymbol{0}$ for $k \neq k_m$. However, if $\left\{\boldsymbol{R}_k^{i-1}\right\}_k$ have similar low-rank structure, as considered in this chapter, the principal singular vector of one matrix lies in all null spaces of other matrices does not happen. That is, the iterative procedure in Algorithm 6 will improve the rank-one solution, which makes the GP converge faster than the EGP, in general. It is also observed that the convergence rate of the GP is faster than the EGP in the numerical results in Section 4.5.

4.4.3 Convergence Analysis for OGP

It is clear that the convergence of the OGP is guaranteed since its reconstruction error decreases faster than that of the GP due to the re-minimization with respect to the weight coefficients of (4.29) at each iteration. This means that the convergence rate of the OGP is faster than the GP. Theorem 4 further states how much the OGP is faster than the GP, where the acceleration factor is quantitatively given.

Theorem 4 *The reconstruction error of the OGP for LRAMM in Algorithm 8 decays exponentially*

$$\sum_{k=1}^{K} \|\boldsymbol{R}_k^i\|_{\mathrm{F}}^2 \leq \gamma_{\mathrm{OGP}}^{i-1}\gamma_{\mathrm{GP}} \sum_{k=1}^{K} \|\boldsymbol{A}_k\|_{\mathrm{F}}^2 \tag{4.66}$$

for the iteration number $i = 0, 1, 2, \cdots$, *where a lower bound of the convergence rate*

$$\gamma_{\mathrm{OGP}} = 1 - \frac{\rho}{\min(m,n)K} \tag{4.67}$$

satisfies $0 < \gamma_{\mathrm{OGP}} < 1$ *with* $\rho > 1$ *being the acceleration factor. Also, it follows that* $\gamma_{\mathrm{OGP}} < \gamma_{\mathrm{GP}}$ *due to* $\rho > 1$ *and hence the OGP converges faster than the GP.*

Proving Theorem 4 requires the following lemma.

Lemma 2 *For the OGP, the squared Frobenius norms of* $\boldsymbol{R}_k^i$ *and* $\boldsymbol{R}_k^{i-1}$ *for* $i = 1, 2, \cdots,$ *are linked by*

$$\|\boldsymbol{R}_k^i\|_{\mathrm{F}}^2 = \|\boldsymbol{R}_k^{i-1}\|_{\mathrm{F}}^2 - \rho_i\big(\boldsymbol{u}_i^{\mathrm{T}}\boldsymbol{R}_k^{i-1}\boldsymbol{v}_i\big)^2 \tag{4.68}$$

where $\rho_1 = 1$ *and for* $i \geq 2$

$$\rho_i = \frac{1}{\sin^2\theta_i} > 1 \tag{4.69}$$

with θ_i *being the angle between* $\mathrm{vec}(\boldsymbol{u}_i\boldsymbol{v}_i^{\mathrm{T}})$ *and the subspace spanned by* $\big\{\mathrm{vec}(\boldsymbol{u}_l\boldsymbol{v}_l^{\mathrm{T}})\big\}_{l=1}^{i-1}$.

Proof. At the first iteration or $i = 1$, the OGP obtains the same result as the GP. By (4.19), we at once get $\rho_1 = 1$. We then discuss the case of $i \geq 2$. Denoting $\boldsymbol{r}_k^i = \mathrm{vec}(\boldsymbol{R}_k^i)$ and $\boldsymbol{r}_k^{i-1} = \mathrm{vec}(\boldsymbol{R}_k^{i-1})$, clearly we have $\|\boldsymbol{R}_k^i\|_{\mathrm{F}}^2 = \|\boldsymbol{r}_k^i\|^2$ and $\|\boldsymbol{R}_k^{i-1}\|_{\mathrm{F}}^2 = \|\boldsymbol{r}_k^{i-1}\|^2$. We now derive the relation between $\|\boldsymbol{r}_k^i\|^2$ and $\|\boldsymbol{r}_k^{i-1}\|^2$. Recalling $\boldsymbol{r}_k^i = \boldsymbol{B}_i\boldsymbol{s}^i - \boldsymbol{a}_k$, by (4.34) it follows that

$$\boldsymbol{r}_k^i = (\boldsymbol{\Pi}_i - \boldsymbol{I})\boldsymbol{a}_k = -\boldsymbol{\Pi}_i^{\perp}\boldsymbol{a}_k \tag{4.70}$$

where

$$\boldsymbol{\Pi}_i = \boldsymbol{B}_i\big(\boldsymbol{B}_i^{\mathrm{T}}\boldsymbol{B}_i\big)^{-1}\boldsymbol{B}_i^{\mathrm{T}} \in \mathbb{R}^{mn\times mn} \tag{4.71}$$

is the orthogonal projection matrix onto range($\boldsymbol{B}_i$), i.e., the column space of $\boldsymbol{B}_i$, while

$$\boldsymbol{\Pi}_i^{\perp} = \boldsymbol{I} - \boldsymbol{\Pi}_i \tag{4.72}$$

is the projection matrix onto the complementary subspace of range($\boldsymbol{B}_i$), i.e., range($\boldsymbol{B}_i^{\perp}$). Then, we get

$$\|\boldsymbol{r}_k^i\|^2 = \|\boldsymbol{\Pi}_i^{\perp}\boldsymbol{a}_k\|^2 = \boldsymbol{a}_k^{\mathrm{T}}(\boldsymbol{\Pi}_i^{\perp})^{\mathrm{T}}\boldsymbol{\Pi}_i^{\perp}\boldsymbol{a}_k = \boldsymbol{a}_k^{\mathrm{T}}\boldsymbol{\Pi}_i^{\perp}\boldsymbol{a}_k \tag{4.73}$$

where we have exploited that the projection matrix is symmetric and idempotent, i.e., $\boldsymbol{\Pi}_i^{\perp} = (\boldsymbol{\Pi}_i^{\perp})^{\mathrm{T}} = (\boldsymbol{\Pi}_i^{\perp})^2$. Similarly, we have $\|\boldsymbol{r}_k^{i-1}\|^2 = \boldsymbol{a}_k^{\mathrm{T}}\boldsymbol{\Pi}_{i-1}^{\perp}\boldsymbol{a}_k$ where

$$\boldsymbol{\Pi}_{i-1} = \boldsymbol{B}_{i-1}\big(\boldsymbol{B}_{i-1}^{\mathrm{T}}\boldsymbol{B}_{i-1}\big)^{-1}\boldsymbol{B}_{i-1}^{\mathrm{T}}. \tag{4.74}$$

Plugging $\boldsymbol{B}_i = [\boldsymbol{B}_{i-1}, \boldsymbol{b}_i]$ into (4.71) and using the block matrix inversion formula [140], we obtain

$$\boldsymbol{\Pi}_i = \boldsymbol{\Pi}_{i-1} + \rho_i\boldsymbol{\Pi}_{i-1}^{\perp}\boldsymbol{b}_i\boldsymbol{b}_i^{\mathrm{T}}\boldsymbol{\Pi}_{i-1}^{\perp} \tag{4.75}$$

and

$$\boldsymbol{\Pi}_i^{\perp} = \boldsymbol{\Pi}_{i-1}^{\perp} - \rho_i\boldsymbol{\Pi}_{i-1}^{\perp}\boldsymbol{b}_i\boldsymbol{b}_i^{\mathrm{T}}\boldsymbol{\Pi}_{i-1}^{\perp} \tag{4.76}$$

after some tedious but straightforward derivations, where

$$\rho_i = \frac{1}{\boldsymbol{b}_i^{\mathrm{T}}\boldsymbol{\Pi}_{i-1}^{\perp}\boldsymbol{b}_i}. \tag{4.77}$$

Now we prove $\rho_i > 1$. The non-expansive property of projection onto convex sets[2] [79] elicits

$$\boldsymbol{b}_i^{\mathrm{T}} \boldsymbol{\Pi}_{i-1}^{\perp} \boldsymbol{b}_i = \|\boldsymbol{\Pi}_{i-1}^{\perp} \boldsymbol{b}_i\|^2 \leq \|\boldsymbol{b}_i\|^2 = 1 \tag{4.78}$$

where $\|\boldsymbol{b}_i\|^2 = 1$ is due to (4.39). Only when $\boldsymbol{b}_i$ is orthogonal to range$(\boldsymbol{B}_i)$ or $\langle \boldsymbol{b}_i, \boldsymbol{b}_l \rangle = 0$ for all $l = 1, \cdots, i-1$, $\|\boldsymbol{\Pi}_{i-1}^{\perp} \boldsymbol{b}_i\|^2 = \|\boldsymbol{b}_i\|^2$ happens and $\rho_i = 1$ in this case. Since

$$\langle \boldsymbol{b}_i, \boldsymbol{b}_l \rangle = \left\langle \boldsymbol{u}_i \boldsymbol{v}_i^{\mathrm{T}}, \boldsymbol{u}_l \boldsymbol{v}_l^{\mathrm{T}} \right\rangle = \langle \boldsymbol{u}_i, \boldsymbol{u}_l \rangle \langle \boldsymbol{v}_i, \boldsymbol{v}_l \rangle \tag{4.79}$$

$\langle \boldsymbol{b}_i, \boldsymbol{b}_l \rangle = 0$ implies $\langle \boldsymbol{u}_i, \boldsymbol{v}_l \rangle = 0$ or $\langle \boldsymbol{v}_i, \boldsymbol{v}_l \rangle = 0$ for all $l = 1, \cdots, i-1$. However, unlike the orthogonal requirement in the GLRAM, we never perform orthogonalization to the columns of $\boldsymbol{U}$ or $\boldsymbol{V}$. For random matrices or matrices containing random components, the probability of $\langle \boldsymbol{u}_i, \boldsymbol{v}_l \rangle = 0$ or $\langle \boldsymbol{v}_i, \boldsymbol{v}_l \rangle = 0$ is zero. Hence, we have $\|\boldsymbol{\Pi}_{i-1}^{\perp} \boldsymbol{b}_i\|^2 < \|\boldsymbol{b}_i\|^2$ and $\rho_i > 1$ in general. The cosine of the angle between $\boldsymbol{b}_i$ and the subspace range$(\boldsymbol{B}_{i-1}^{\perp})$ is

$$\cos \phi_i = \frac{\left\langle \boldsymbol{b}_i, \boldsymbol{\Pi}_{i-1}^{\perp} \boldsymbol{b}_i \right\rangle}{\|\boldsymbol{b}_i\| \|\boldsymbol{\Pi}_{i-1}^{\perp} \boldsymbol{b}_i\|} = \|\boldsymbol{\Pi}_{i-1}^{\perp} \boldsymbol{b}_i\| = \frac{1}{\sqrt{\rho_i}} \tag{4.80}$$

where we have used $\left\langle \boldsymbol{b}_i, \boldsymbol{\Pi}_{i-1}^{\perp} \boldsymbol{b}_i \right\rangle = \boldsymbol{b}_i^{\mathrm{T}} \boldsymbol{\Pi}_{i-1}^{\perp} \boldsymbol{b}_i = \|\boldsymbol{\Pi}_{i-1}^{\perp} \boldsymbol{b}_i\|^2$ and $\|\boldsymbol{b}_i\| = 1$. Hence,

$$\rho_i = \frac{1}{\cos^2 \phi_i} = \frac{1}{\sin^2 \theta_i} \tag{4.81}$$

where θ_i is the angle between $\boldsymbol{b}_i$ and the subspace range$(\boldsymbol{B}_{i-1})$ and clearly it has $\theta_i + \phi_i = \pi/2$ since range$(\boldsymbol{B}_{i-1}^{\perp})$ is the orthogonal compliment of range$(\boldsymbol{B}_{i-1})$.

By (4.76), it follows that

$$\boldsymbol{a}_k^{\mathrm{T}} \boldsymbol{\Pi}_i^{\perp} \boldsymbol{a}_k = \boldsymbol{a}_k^{\mathrm{T}} \boldsymbol{\Pi}_{i-1}^{\perp} \boldsymbol{a}_k - \rho_i \boldsymbol{a}_k^{\mathrm{T}} \boldsymbol{\Pi}_{i-1}^{\perp} \boldsymbol{b}_i \boldsymbol{b}_i^{\mathrm{T}} \boldsymbol{\Pi}_{i-1}^{\perp} \boldsymbol{a}_k \tag{4.82}$$

Substituting (4.70) and (4.73) into (4.82) yields

$$\|\boldsymbol{r}_k^i\|^2 = \|\boldsymbol{r}_k^{i-1}\|^2 - \rho_i \big(\boldsymbol{b}_i^{\mathrm{T}} \boldsymbol{r}_k^{i-1}\big)^2. \tag{4.83}$$

Since

$$\begin{aligned} \boldsymbol{b}_i^{\mathrm{T}} \boldsymbol{r}_k^{i-1} &= \left\langle \boldsymbol{b}_i, \boldsymbol{r}_k^{i-1} \right\rangle = \left\langle \boldsymbol{u}_i \boldsymbol{v}_i^{\mathrm{T}}, \boldsymbol{R}_k^{i-1} \right\rangle \\ &= \operatorname{tr}\big(\boldsymbol{v}_i \boldsymbol{u}_i^{\mathrm{T}} \boldsymbol{R}_k^{i-1}\big) = \boldsymbol{u}_i^{\mathrm{T}} \boldsymbol{R}_k^{i-1} \boldsymbol{v}_i \end{aligned} \tag{4.84}$$

(4.83) in the form of vector is equivalent to the following matrix form

$$\|\boldsymbol{R}_k^i\|_{\mathrm{F}}^2 = \|\boldsymbol{R}_k^{i-1}\|_{\mathrm{F}}^2 - \rho_i \big(\boldsymbol{u}_i^{\mathrm{T}} \boldsymbol{R}_k^{i-1} \boldsymbol{v}_i\big)^2$$

which completes the proof. □

---

[2] Denote the projection onto a convex set $\mathcal{C}$ as $\boldsymbol{\Pi}_{\mathcal{C}}(\cdot)$. The non-expansiveness states that $\|\boldsymbol{\Pi}_{\mathcal{C}}(\boldsymbol{b})\| \leq \|\boldsymbol{b}\|$ is true for any vector $\boldsymbol{b}$ [79]. Since range$(\boldsymbol{B}_{i-1})$ is a subspace and also a convex set, the projections $\boldsymbol{\Pi}_{i-1}$ and $\boldsymbol{\Pi}_{i-1}^{\perp}$ are non-expansive.

Proof of Theorem 4. Now we are ready to prove Theorem 4. Summing (4.68) in Lemma 2 from $k=1$ to K and using the key inequality (4.50) in Lemma 1 leads to

$$\begin{aligned}\sum_{k=1}^{K}\|\boldsymbol{R}_k^i\|_{\mathrm{F}}^2 &= \sum_{k=1}^{K}\|\boldsymbol{R}_k^{i-1}\|_{\mathrm{F}}^2 - \rho_i\sum_{k=1}^{K}\left(\boldsymbol{u}_i^{\mathrm{T}}\boldsymbol{R}_k^{i-1}\boldsymbol{v}_i\right)^2\\ &\le \sum_{k=1}^{K}\|\boldsymbol{R}_k^{i-1}\|_{\mathrm{F}}^2 - \frac{\rho_i}{\min(m,n)K}\sum_{k=1}^{K}\|\boldsymbol{R}_k^{i-1}\|_{\mathrm{F}}^2\\ &= \left(1-\frac{\rho_i}{\min(m,n)K}\right)\sum_{k=1}^{K}\|\boldsymbol{R}_k^{i-1}\|_{\mathrm{F}}^2\end{aligned}\tag{4.85}$$

which holds true for $i, i-1, \cdots, 2$. Successively applying (4.85) results in

$$\begin{aligned}\sum_{k=1}^{K}\|\boldsymbol{R}_k^i\|_{\mathrm{F}}^2 &\le \prod_{l=2}^{i}\left(1-\frac{\rho_l}{\min(m,n)K}\right)\sum_{k=1}^{K}\|\boldsymbol{R}_k^1\|_{\mathrm{F}}^2\\ &\le \left(1-\frac{\rho}{\min(m,n)K}\right)^{i-1}\sum_{k=1}^{K}\|\boldsymbol{R}_k^1\|_{\mathrm{F}}^2\\ &= \gamma_{\mathrm{OGP}}^{i-1}\sum_{k=1}^{K}\|\boldsymbol{R}_k^1\|_{\mathrm{F}}^2\end{aligned}\tag{4.86}$$

where $\rho=\min\{\rho_2,\cdots,\rho_i\}$ and $\rho>1$. For $i=1$, the OGP is the same as GP and thus the reconstruction error obeys

$$\sum_{k=1}^{K}\|\boldsymbol{R}_k^1\|_{\mathrm{F}}^2 \le \gamma_{\mathrm{GP}}\sum_{k=1}^{K}\|\boldsymbol{A}_k\|_{\mathrm{F}}^2.\tag{4.87}$$

Combining (4.86) and (4.87) yields (4.66). □

From the second iteration ($i\ge 2$), the OGP converges faster than the GP because of $\gamma_{\mathrm{OGP}}<\gamma_{\mathrm{GP}}$. The acceleration ratio depends on ρ. The larger ρ, the faster the OGP is. Furthermore, the OGP has finite convergence property, as stated in the following theorem.

Theorem 5 *The current residual matrices $\{\boldsymbol{R}_k^i\}_{k=1}^K$ generated by the OGP are orthogonal to all rank-one matrices $\left\{\boldsymbol{u}_l\boldsymbol{v}_l^{\mathrm{T}}\right\}_{l=1}^{i}$ that have been chosen. These selected rank-one matrices are linearly independent with each other. The reconstruction error of the OGP will be zero after at most mn iterations.*

Proof. Two matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ are orthogonal if their inner product $\langle\boldsymbol{A},\boldsymbol{B}\rangle=0$, or equivalently, $\langle\mathrm{vec}(\boldsymbol{A}),\mathrm{vec}(\boldsymbol{B})\rangle=0$. According to (4.70), it follows that

$$\boldsymbol{B}_i^{\mathrm{T}}\boldsymbol{r}_k^i = -\boldsymbol{B}_i^{\mathrm{T}}\boldsymbol{\Pi}_i^{\perp}\boldsymbol{a}_k = \boldsymbol{0},\ k=1,\cdots,K\tag{4.88}$$

where $\boldsymbol{B}_i^{\mathrm{T}}\boldsymbol{\Pi}_i^{\perp} = \boldsymbol{0}$ is used, which is obviously true because $\boldsymbol{\Pi}_i^{\perp}$ is the projection onto the orthogonal complement of range$(\boldsymbol{B}_i)$. Then, (4.88) amounts to $\langle \boldsymbol{b}_l, \boldsymbol{r}_k^i \rangle = 0$ or $\left\langle \boldsymbol{u}_l\boldsymbol{v}_l^{\mathrm{T}}, \boldsymbol{R}_k^i \right\rangle = 0$, for $l = 1, \cdots, i$. This completes the proof of $\{\boldsymbol{R}_k^i\}_{k=1}^K$ being orthogonal to $\left\{\boldsymbol{u}_l\boldsymbol{v}_l^{\mathrm{T}}\right\}_{l=1}^i$. To prove that $\left\{\boldsymbol{u}_l\boldsymbol{v}_l^{\mathrm{T}}\right\}$ are linearly independent with each other, we only need to prove the rank-one matrix obtained in the next iteration is linearly independent of all rank-one matrices in previous iterations. Now, we prove $\boldsymbol{u}_{i+1}\boldsymbol{v}_{i+1}^{\mathrm{T}}$ is linearly independent of $\left\{\boldsymbol{u}_l\boldsymbol{v}_l^{\mathrm{T}}\right\}_{l=1}^i$ by contradiction. Suppose that $\boldsymbol{u}_{i+1}\boldsymbol{v}_{i+1}^{\mathrm{T}}$ is linearly dependent of $\left\{\boldsymbol{u}_l\boldsymbol{v}_l^{\mathrm{T}}\right\}_{l=1}^i$. Then, it can be represented by the linear combination of $\left\{\boldsymbol{u}_l\boldsymbol{v}_l^{\mathrm{T}}\right\}_{l=1}^i$, i.e., $\boldsymbol{u}_{i+1}\boldsymbol{v}_{i+1}^{\mathrm{T}} = \sum_{l=1}^i \alpha_l \boldsymbol{u}_l\boldsymbol{v}_l^{\mathrm{T}}$ with $\alpha_l \in \mathbb{R}$. The inner product

$$\left\langle \boldsymbol{u}_{i+1}\boldsymbol{v}_{i+1}^{\mathrm{T}}, \boldsymbol{R}_k^i \right\rangle = \sum_{l=1}^{i} \alpha_l \left\langle \boldsymbol{u}_l\boldsymbol{v}_l^{\mathrm{T}}, \boldsymbol{R}_k^i \right\rangle = 0 \tag{4.89}$$

implies $\boldsymbol{u}_{i+1}^{\mathrm{T}}\boldsymbol{R}_k^i\boldsymbol{v}_{i+1} = 0$ for $k = 1, \cdots, K$ as well as $\sum_{k=1}^K \left(\boldsymbol{u}_{i+1}^{\mathrm{T}}\boldsymbol{R}_k^i\boldsymbol{v}_{i+1}\right)^2 = 0$. According to (4.21), the OGP selects $(\boldsymbol{u}_{i+1}, \boldsymbol{v}_{i+1})$ such that

$$(\boldsymbol{u}_{i+1}, \boldsymbol{v}_{i+1}) = \arg \max_{\|\boldsymbol{u}\|=\|\boldsymbol{v}\|=1} \sum_{k=1}^{K} \left(\boldsymbol{u}^{\mathrm{T}}\boldsymbol{R}_k^i\boldsymbol{v}\right)^2. \tag{4.90}$$

But $\sum_{k=1}^K \left(\boldsymbol{u}_{i+1}^{\mathrm{T}}\boldsymbol{R}_k^i\boldsymbol{v}_{i+1}\right)^2 = 0$ attains the minimum value 0 due to the assumption that $\boldsymbol{u}_{i+1}\boldsymbol{v}_{i+1}^{\mathrm{T}}$ is linearly dependent of $\left\{\boldsymbol{u}_l\boldsymbol{v}_l^{\mathrm{T}}\right\}_{l=1}^i$. This case only happens when all residual matrices $\{\boldsymbol{R}_k^i\}_{k=1}^K$ vanish and the reconstruction error becomes zero. In this case, the algorithm has already converged and terminated. Otherwise, it contradicts. We complete the proof that $\left\{\boldsymbol{u}_l\boldsymbol{v}_l^{\mathrm{T}}\right\}_l$ or $\left\{\boldsymbol{b}_l = \mathrm{vec}(\boldsymbol{u}_l\boldsymbol{v}_l^{\mathrm{T}})\right\}_l$ are linearly independent with each other provided that any residual matrix does not vanish. After $i = mn$ iterations, $\boldsymbol{B}_i$ contains mn linearly independent columns which span the whole space of $\mathbb{R}^{mn}$. Then, the residual $\boldsymbol{r}_k^i = -\boldsymbol{\Pi}_i^{\perp}\boldsymbol{a}_k = \boldsymbol{0}$ due to $\boldsymbol{\Pi}_i^{\perp} = \boldsymbol{0}$, which also implies $\boldsymbol{R}_k^i = \boldsymbol{0}$ and $\sum_{k=1}^K \|\boldsymbol{R}_k^i\|_{\mathrm{F}}^2 = 0$ after at most mn iterations. □

In practical applications, mn is usually very large. Generally, a target rank $r \ll mn$ is enough to capture the low-rank structure of natural images and achieves a small reconstruction error.

4.5 Experimental Results

All experiments are conducted using a computer with a 2.2 GHz CPU and 4 GB memory. In addition to synthetic random data, the following four real-world databases, including two face datasets, one dataset of handwritten digits and one object dataset, are employed.

- ORL face database [141]. It consists of 10 different images of each of 40 distinct subjects for a total of 400 images. The resolution of the gray-scale images is 112×92 and we have $m = 92$, $n = 112$, and $K = 400$.
- Georgia Tech face database [142]. It contains 750 images of 50 individuals. There are 15 images for each individual. The original images are colored and with different sizes. We convert them to gray-scale with the same size of 111×156 so that they can be represented by matrices. We have $m = 156$, $n = 111$, and $K = 750$.
- MNIST database of handwritten digits [143]. It is composed of images of digits 0 to 9 written by 500 different people. There are 70,000 images with size of 28×28 in total while here we select a smaller subset of 2000 samples, which results in $m = n = 28$ and $K = 2000$.
- COIL-20 database [144]. There are 1440 gray-scale images of 20 different objects, which corresponds to 72 images per object. The image resolution is 128×128 and it follows $m = n = 128$ and $K = 1440$.

4.5.1 Convergence Behaviors

The convergence behaviors are investigated. First random data are used. We set $m = 100$, $n = 120$, and $K = 15$. A set of noise-free matrices of rank 10 is generated by $\boldsymbol{A}_k = \boldsymbol{U}\boldsymbol{S}_k\boldsymbol{V}^{\mathrm{T}}$, $k = 1, \cdots, K$, where the entries of $\boldsymbol{U} \in \mathbb{R}^{m\times 10}$ and $\boldsymbol{V} \in \mathbb{R}^{10\times n}$ satisfy the standard Gaussian distribution while the diagonal entries of $\boldsymbol{S}_k$ are uniformly distributed in $[1, 2]$ to avoid any diagonal entry being too close to zero.

Figure 4.2 plots the NRE of (4.13) versus iteration number for noise-free case. We see that the reconstruction error rapidly decreases when the iteration number is not larger than the rank. The reconstruction error approaches zero as the iteration proceeds. The OGP converges fastest while the EGP converges slowest, although all of them have linear convergence rates. Then, the noise matrices $\boldsymbol{N}_k$ whose entries are independent and identically distributed Gaussian with variance σ_n^2 are added to $\boldsymbol{A}_k$. The signal-to-noise ratio (SNR) is defined as $\mathrm{SNR} = \left(\sum_{k=1}^{K} \|\boldsymbol{A}_k\|_{\mathrm{F}}^2\right)/(mnK\sigma_n^2)$. In the presence of noise, the oracle bound of the NRE is dominated by the noise level. Figure 4.3 shows the NRE as well as the oracle bound versus iteration number at SNRs of 10, 20, and 30 dB. The oracle bounds equal the reciprocal of the SNRs. As we see, the reconstruction error rapidly decreases when $i \leq 10$. After approaching the oracle bounds, the NREs decrease slowly. This means that the greedy methods have captured

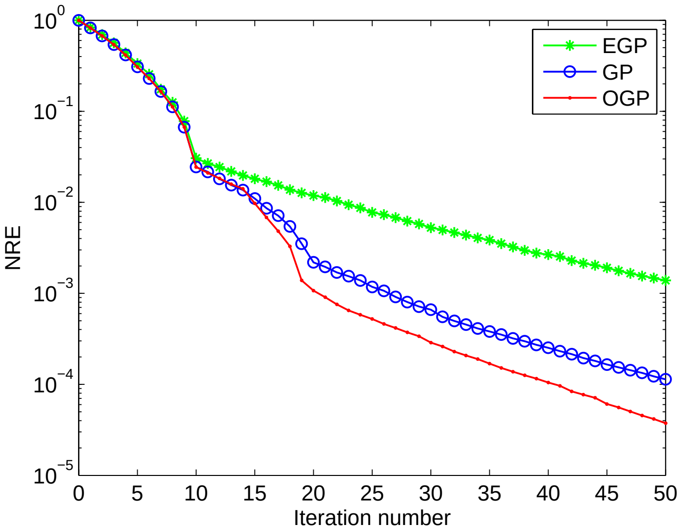


Figure 4.2: NRE versus iteration number using random data for noise-free case.

the dominating low-rank structure and the iteration procedure can be stopped since little improvement will be achieved due to the impact of the noise.

Figure 4.4 shows the NREs versus iteration number on the four real-world image databases. It is observed that the three greedy algorithms significantly decrease the reconstruction error at the beginning stage. This implies that these real-world images exhibit several "principal components" and the proposed methods successfully find these components although they are not strictly low-rank. The maximum iteration numbers r are set to 149, 231, 53, and 378 for the ORL face, Georgia Tech face, MNIST, and COIL-20 databases, respectively. To achieve a sufficiently small reconstruction error, the maximum iteration number r needs to be larger than $\min(m, n)$ but we still have $r \ll mn$. Again, the OGP has the fastest convergence rate while the EGP is the slowest one.

4.5.2 Results of Image Reconstruction

The reconstruction performances of the three greedy algorithms are compared with the PCA, 2DPCA, and GLARM. For fair comparison, the NREs of the six methods are computed under the same (or close) compression ratios. According to Table I, r_1, r_2, r_3, and r_4, which denote the target ranks of PCA, 2DPCA, GLARM, and GP/EGP/OGP,

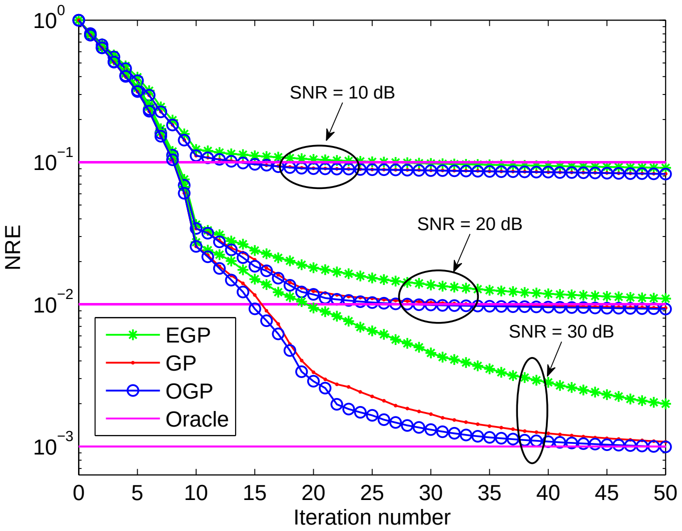


Figure 4.3: NRE versus iteration number using random data in Gaussian noise at SNRs of 10, 20, and 30 dB.

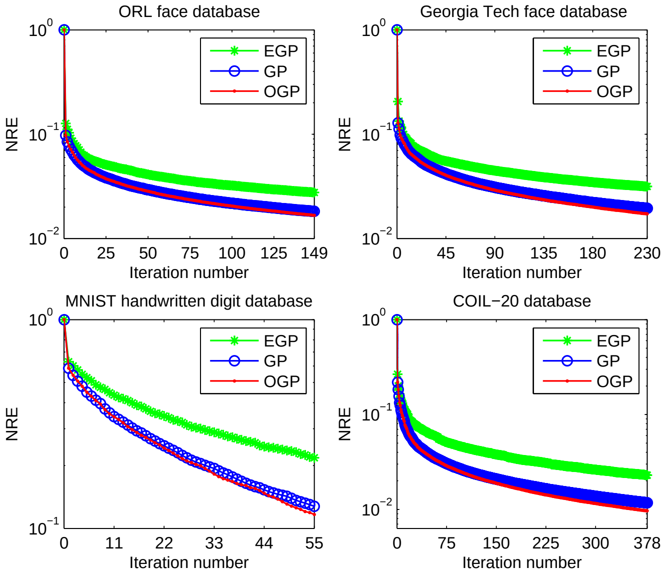


Figure 4.4: NRE versus iteration number on ORL face, Georgia Tech face, MNIST handwritten digit, and COIL-20 databases.

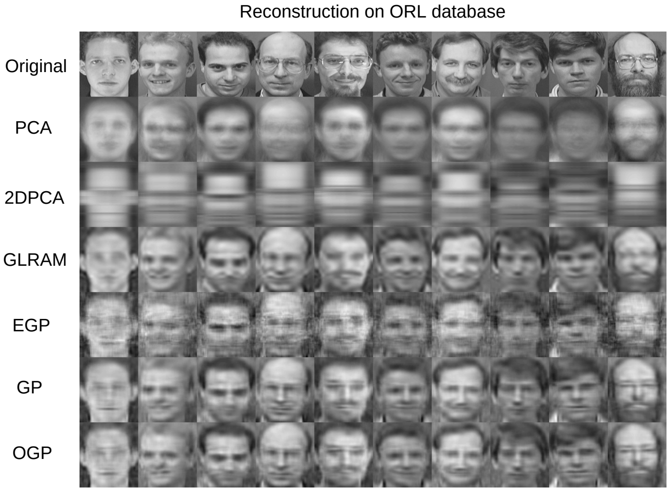


Figure 4.5: Samples of reconstructed images on ORL face database. The compression ratios of the PCA, 2DPCA, GLARM, and EGP/GP/OGP are 48.1, 45.9, 44.3, and 45.8 corresponding to the ranks of 8, 2, 15, and 149. The NREs of the above six methods are 4.56×10^{-2}, 5.02×10^{-2}, 1.63×10^{-2}, 2.75×10^{-2}, 1.82×10^{-2}, and 1.66×10^{-2}, respectively.

respectively, should satisfy

$$\begin{aligned}&(mn + K + 1)r_1 = (mK + n)r_2\\ &= (m + n + Kr_3)r_3 = (m + n + K)r_4.\end{aligned} \tag{4.91}$$

to make the compression ratios of the six methods the same. Noting that $r_1, \cdots, r_4$ are positive integers, (4.91) may not be strictly satisfied. We select the positive integers such that the compression ratios are as close as possible.

Figures 4.5, 4.6, 4.7, and 4.8 display several samples of the reconstructed images obtained by the six algorithms on the ORL face, Georgia Tech face, MNIST, and COIL-20 databases, respectively. The corresponding samples of the original images are also shown for comparison. Obviously, the three greedy algorithms have much smaller reconstruction error than the PCA and 2DPCA under similar compression ratios. In other words, the greedy methods can achieve higher compression ratio if the reconstruction errors are constrained to the same. This is because the reconstruction error monotonically increases with the compression ratio. In general, the GP, OGP and GLRAM have comparable performance. Note that for the COIL-20 database, the NRE of the OGP, 9.70×10^{-3}, is moderately smaller than that of the GLRAM, 1.22×10^{-2}, while their compression ratios are nearly the same.

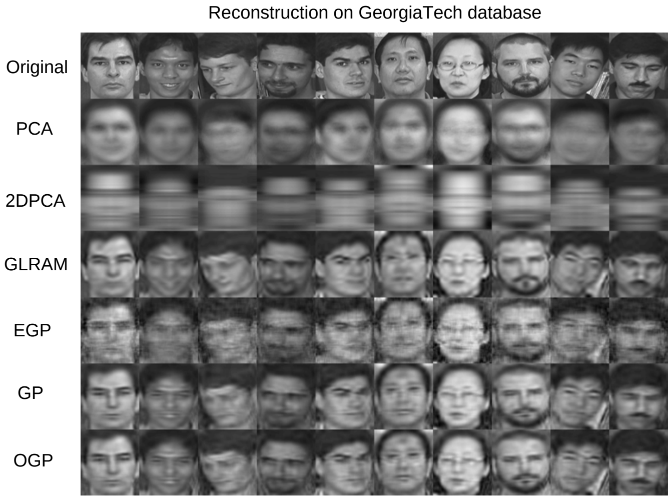


Figure 4.6: Samples of reconstructed images on Georgia Tech face database. The compression ratios of the PCA, 2DPCA, GLARM, and EGP/GP/OGP are 55.3, 55.4, 58.7, and 55.5 corresponding to the ranks of 13, 2, 17, and 230. The NREs of the above six methods are 5.51×10^{-2}, 6.69×10^{-2}, 1.82×10^{-2}, 3.15×10^{-2}, 1.94×10^{-2}, and 1.72×10^{-2}, respectively. A compression ratio of 55.5 implies that only 1.8% storage space of the original data is needed after compression.

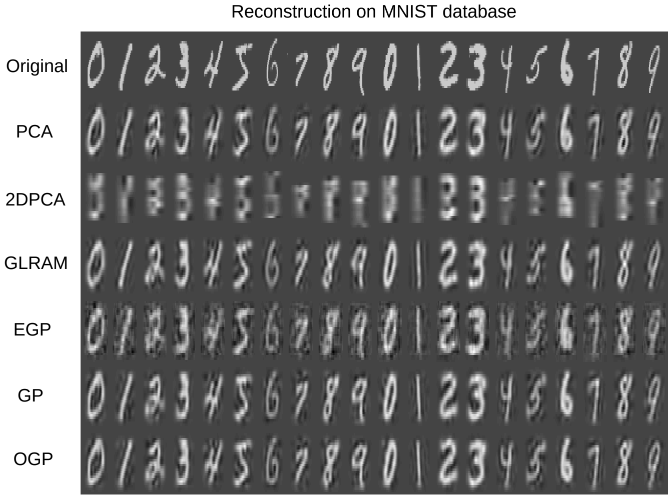


Figure 4.7: Samples of reconstructed images on MNIST database of handwritten digits. The compression ratios of the PCA, 2DPCA, GLARM, and EGP/GP/OGP are 14.1, 14.0, 15.9, and 13.9 corresponding to the ranks of 40, 2, 7 and 55. The NREs of the above six methods are 0.129, 0.363, 0.15, 0.218, 0.128, and 0.117, respectively.

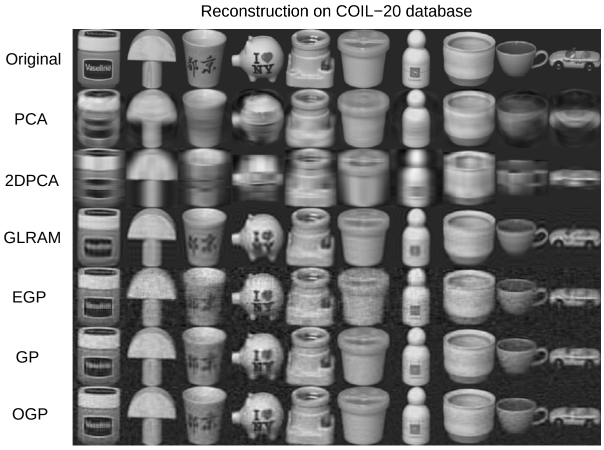


Figure 4.8: Samples of reconstructed images on COIL-20 database of objectives. The compression ratios of the PCA, 2DPCA, GLARM, and EGP/GP/OGP are 31.3, 32.0, 33.3, and 32.0 corresponding to the ranks of 22, 4, 22, and 378. The NREs of the above six methods are 3.45×10^{-2}, 4.63×10^{-2}, 1.22×10^{-2}, 2.29×10^{-2}, 1.17×10^{-2}, and 9.70×10^{-3}, respectively.

We then investigate how the reconstruction error varies with the compression ratio. Figure 4.9 plots the NREs of the six methods versus compression ratio on the four image databases. The reconstruction errors of all schemes monotonically increase with the compression ratio. The OGP has the best reconstruction performance for all databases. The GP and GLRAM have similar performance as the OGP. Despite the low computational cost of the 2DPCA, its performance is not satisfactory because it only uses a single side transform resulting limited compression capability. Note that for the MNIST database, the performance of the PCA is good and comparable to the GLRAM, GP, and OGP, where $mn = 788$ is less than $K = 2000$. For other three databases where $mn \gg K$, the performance of the PCA has a large gap compared with the GLRAM and greedy methods. Therefore, for high-dimensional data with very large mn, the advantage of 2D based methods over the vectorization based one is more evident.

4.5.3 Results of Face Recognition

Figure 4.10 plots the recognition rate versus compression ratio on the ORL face database. The 60% samples of each class are randomly selected as training samples while the remaining 40% are used for test samples. The six methods learn the low-rank structure using the training set and then the NN classifier is employed to the test

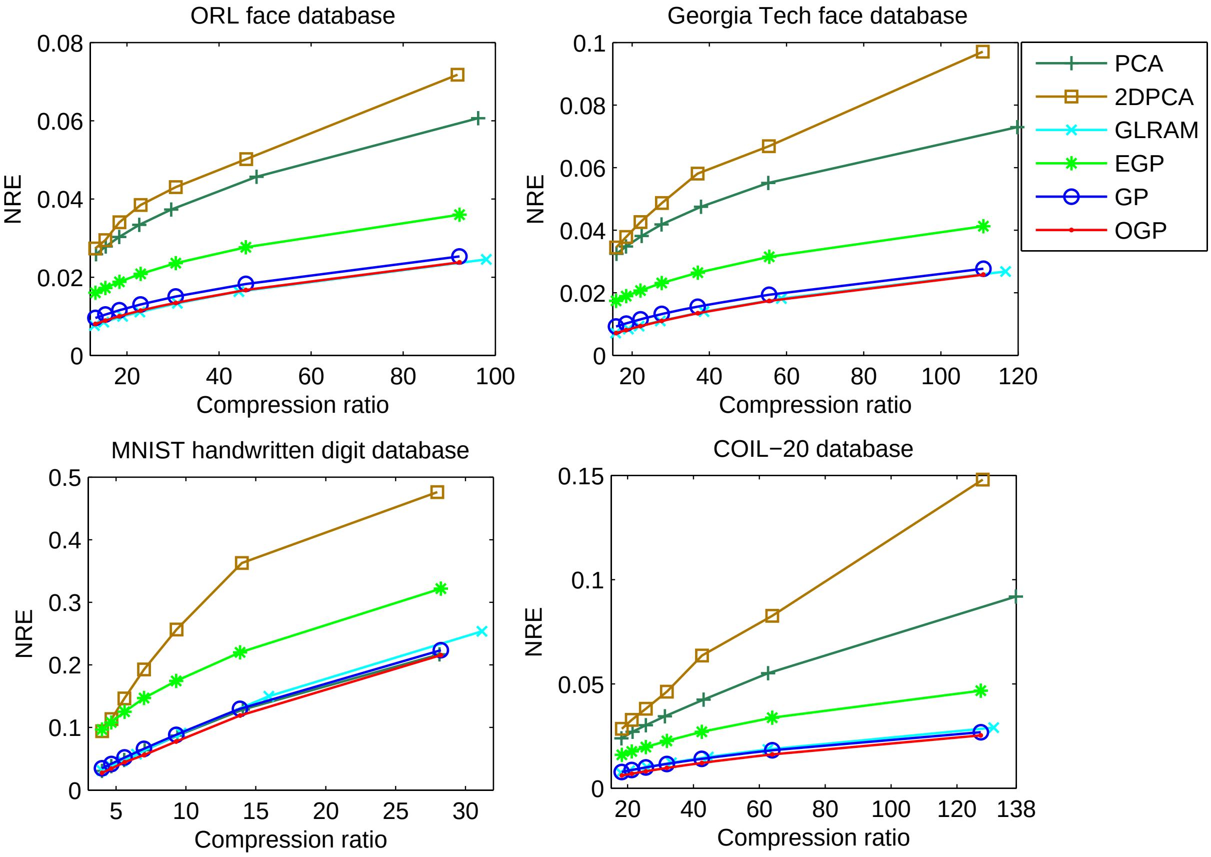


Figure 4.9: NREs of PCA, 2DPCA, GLARM, EGP, GP and OGP versus compression ratio on four real-world databases.

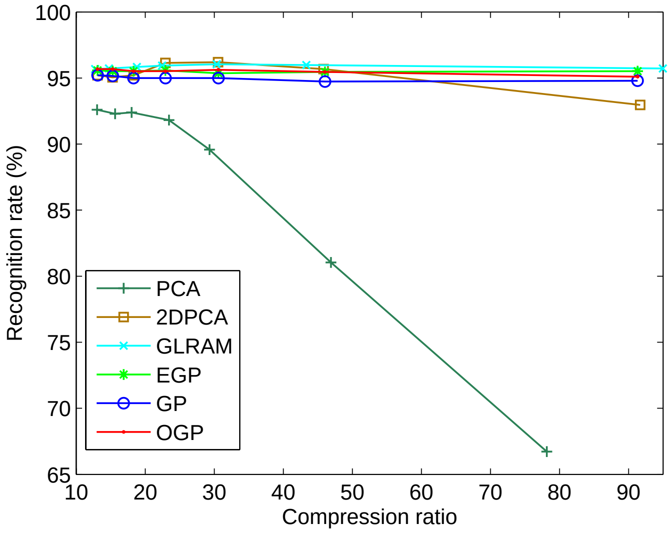


Figure 4.10: Recognition rate versus compression ratio on ORL face database.

samples. We observe that the performance of the PCA is the worst for classification. The performances of the other five methods are similar for small compression ratios. For large compression ratios, the 2DPCA becomes slightly worse. It is also seen that the recognition performance is not so sensitive to the compression ratio or reduced dimension.

4.6 Summary

Three greedy algorithms, namely, GP, EGP, and OGP are devised for LRAMM. They are scalable to the problem size and achieve a higher compression ratio. We also develop the corresponding convergence theories of the three algorithms. We prove that the reconstruction errors of the three algorithms decay exponentially. The lower bound of the worst-case convergence rate is derived. The acceleration mechanism of the OGP over GP is revealed and quantitatively determined. Furthermore, the finite convergence property of the OGP is proved.

Chapter 5

ℓ_p-Greedy Pursuit for Robust Approximation of Multiple Matrices

In Chapter 4, we have discussed three greedy pursuit algorithms for multiple matrix approximation. Compared with the vectorization based methods such as PCA, the GP, EGP and OGP for LRAMM are more "robust" to the 2D data structure and provide higher compression ratio by directly handling multiple matrices. However, they are not robust to outliers. In this Chapter, the GP algorithm is generalized from the ℓ_2-space to ℓ_p-space with $0 < p < 2$ to enhance the outlier-robustness. We propose ℓ_p-greedy pursuit (ℓ_p-GP) algorithms for robust low-rank approximation of multiple matrices (RLRAMM). The ℓ_p-GP with $p \in (0, 2)$ solves the RLRAMM by decomposing it into a series of rank-one approximations. At each iteration, it finds the best rank-one approximation by minimizing the ℓ_p-norm of the residual and then, the rank-one basis matrices are subtracted from the residual. A successive minimization approach is designed for the ℓ_p-rank-one fitting. Only computation of weighted medians is required for solving the most attractive case with $p = 1$, yielding that the complexity is near-linear with the number and dimension of the matrices. Thus, the ℓ_1-GP is near-scalable to large-scale problems. The convergence of the ℓ_p-GP is theoretically proved. In particular, the sum of the ℓ_p-norms of the residuals decays exponentially. We reveal that the worst-case bound of the exponential decay factor or convergence rate is related to the ℓ_p-correlation of the residual and the current iterates. Experimental results on image reconstruction with outliers demonstrate the super robustness of the ℓ_p-GP.

5.1 Introduction

A flaw of the square loss based low-rank approximation techniques, including the PCA, 2DPCA, GLRAM, and GP, is that it is not robust to outliers or impulsive noise. To enhance the robustness, the convex and nonconvex RPCAs [32,38] aims to separate the low-rank component from the sparse outliers. The RPCA requires to solve a minimization of the sum of nuclear norm and ℓ_1-norm, which leads to a high time complexity. To avoid the computationally expensive nuclear norm minimization, the robust low-rank approximation approach of Chapter 2 utilizes direct matrix factorization, where the target matrix is represented by the product of two or more matrices with smaller sizes so that the low-rank constraint is automatically fulfilled. Nevertheless, the algorithms of Chapter 2 are designed for a single matrix and hence, cannot be applied to directly handle multiple matrices.

This chapter addresses robustly learning the common low-rank structure of multiple matrices in the presence of outliers. Our proposed RLRAMM can be viewed as an extension of the RPCA from single matrix to multiple matrices. It can be viewed as a robust version of the techniques of Chapter 4 as well. As in the 2DPCA, GLRAM and 2D-SVD, the RLRAMM does not convert matrices into vectors and thus can avoid processing the matrix with much larger size. Different from the GLRAM and 2D-SVD, the RLRAMM achieves a nonorthogonal but joint diagonal decomposition of multiple matrices, which leads to a more compact representation and a higher compression ratio. An ℓ_p-GP algorithm based on the minimization of the sum of the entry-wise ℓ_p-norms ($0 < p < 2$) of the residuals is designed for this robust learning task. The linear convergence of ℓ_p-GP is theoretically proved and the worst-case bound of the convergence rate is derived in terms of ℓ_p-correlation.

5.2 Problem Formulation and Preliminaries

5.2.1 Problem Formulation

Given a set of matrices $\{\boldsymbol{A}_1, \cdots, \boldsymbol{A}_K\} \in \mathbb{R}^{m\times n}$, we consider finding a low-rank approximation of the K matrices

$$\boldsymbol{A}_k \approx \boldsymbol{U}\boldsymbol{S}_k\boldsymbol{V}^{\mathrm{T}},\ k = 1, \cdots, K \tag{5.1}$$

where $\boldsymbol{U} \in \mathbb{R}^{m\times r}$, $\boldsymbol{V} \in \mathbb{R}^{n\times r}$, and the diagonal matrix $\boldsymbol{S}_k \in \mathbb{R}^{r\times r}$ is

$$\boldsymbol{S}_k = \mathrm{diag}\{s_{k,1}, \cdots, s_{k,r}\}$$

with r being the target "rank". Here we allow that $r > \min(m, n)$ but generally $r \leq mn$ is required. Note that r is not the rank if $r > \min(m, n)$, but we still use the name of "low-rank approximation". As we will see later, (5.1) can still achieve data compression even when $r > \min(m, n)$. Note that $\boldsymbol{U}$ and $\boldsymbol{V}$ are the same for all k but $\boldsymbol{S}_k$ can be different with each other. The columns of $\boldsymbol{U}$ and $\boldsymbol{V}$ span the r-dimensional subspaces of the column and row spaces of $\{\boldsymbol{A}_k\}_{k=1}^K$. If the matrices are not strictly low-rank or there exists noise/outlier, we solve the following minimization problem

$$\min_{\boldsymbol{U},\boldsymbol{V},\{\boldsymbol{S}_k\}_{k=1}^K} \sum_{k=1}^{K} \left\|\boldsymbol{U}\boldsymbol{S}_k\boldsymbol{V}^{\mathrm{T}} - \boldsymbol{A}_k\right\|_p^p \tag{5.2}$$

to obtain the low-rank approximation, where $\|\cdot\|_p$ with $0 < p \leq 2$ denotes the element-wise ℓ_p-norm of a matrix, which has the form of

$$\|\boldsymbol{A}\|_p = \left(\sum_{i,j} |\boldsymbol{A}_{ij}|^p\right)^{1/p}.$$

Note that (5.2) is nonconvex even for $p \geq 1$ due to the product term $\boldsymbol{U}\boldsymbol{S}_k\boldsymbol{V}^{\mathrm{T}}$. For $p = 2$, $\|\cdot\|_p$ becomes the Frobenius norm and it is suitable for noise-free or Gaussian noise case, which has been discussed in Chapter 4. A benefit of using the Frobenius norm is the computational convenience. However, this choice is not robust to outliers. To achieve outlier-robustness, we consider the use of $0 < p < 2$, especially $p = 1$. For $p \leq 1$, (5.2) becomes more challenging since the ℓ_p-norm is nonsmooth in the case.

One application of the above low-rank approximation is for robust data compression. Obviously, it requires mnK real numbers to store the K original matrices. Using the low-rank approximation of (5.2), the storage complexity is reduced to $(m + n + K)r$ numbers since only $\boldsymbol{U}$, $\boldsymbol{V}$, and the diagonal elements of $\{\boldsymbol{S}_k\}_{k=1}^K$ are needed to store. Thus, the compression ratio of our RLRAMM is $mnK/((m + n + K)r)$. This implies that (5.1) can still achieve data compression when $r > \min(m, n)$ if $r \ll mn$. Indeed, the compression ratio and reconstruction error decrease as r increases. To guarantee a satisfying reconstruction error, we may use a $r > \min(m, n)$. Recall that the GLRAM [16] requires $\boldsymbol{U}$ and $\boldsymbol{V}$ to be orthonormal, i.e., $\boldsymbol{U}^{\mathrm{T}}\boldsymbol{U} = \boldsymbol{V}^{\mathrm{T}}\boldsymbol{V} = \boldsymbol{I}_r$, which results in that $r \leq \min(m, n)$ must be satisfied for GLRAM.

Denoting the ith $(i = 1, \cdots, r)$ columns of $\boldsymbol{U}$ and $\boldsymbol{V}$ as $\boldsymbol{u}_i$ and $\boldsymbol{v}_i$, respectively, (5.2) is rewritten as

$$\min_{\{\boldsymbol{s}_i, \boldsymbol{u}_i, \boldsymbol{v}_i\}} \sum_{k=1}^{K} \left\| \sum_{i=1}^{r} s_{k,i} \boldsymbol{u}_i \boldsymbol{v}_i^{\mathrm{T}} - \boldsymbol{A}_k \right\|_p^p \tag{5.3}$$

where the vector

$$\boldsymbol{s}_i = [s_{1,i}, \cdots, s_{K,i}]^{\mathrm{T}} \in \mathbb{R}^K. \tag{5.4}$$

If $p = 2$ and the number of matrices is $K = 1$, the solution of (5.2) is the truncated SVD of $\boldsymbol{A}_1$ by Eckart-Young Theorem [26]. That is, $\{s_{1,i}\}_{i=1}^r$ are the r largest singular values of $\boldsymbol{A}_1$, and $\{\boldsymbol{u}_i\}_{i=1}^r$ and $\{\boldsymbol{v}_i\}_{i=1}^r$ are the corresponding left and right singular vectors, respectively. When $p \neq 2$ or the number of matrices is $K > 1$, the truncated SVD cannot be applied to solving (5.3). The goal of this chapter is to devise greedy pursuit algorithms for efficiently solving (5.3).

5.2.2 RPCA for Multiple Matrix Approximation

The PCA, 2DPCA and GLRAM for low-rank representation have been discussed in Chapter 4. They are not robust to outliers since they are based on Frobenius norm minimization. Here we review the RPCA. Like the PCA, the RPCA cannot directly handle multiple matrices whereas each matrix needs to be converted into a vector $\boldsymbol{a}_k = \mathrm{vec}(\boldsymbol{A}_k) \in \mathbb{R}^{mn}$. Then, the K vectors $\{\boldsymbol{a}_k\}_{k=1}^K$ form the columns of following data matrix

$$\boldsymbol{X} = [\boldsymbol{a}_1, \cdots, \boldsymbol{a}_K] \in \mathbb{R}^{mn\times K}. \tag{5.5}$$

The RPCA achieves robustness against outliers by modeling the matrix $\boldsymbol{X}$ as the superposition of a low-rank matrix and a sparse matrix which represents the outliers [32, 33]. It minimizes the nuclear norm of the unknown low-rank matrix plus the ℓ_1-norm of the outlier component as a regularization term to robustly recover the low-rank matrix:

$$\begin{aligned} &\min_{\boldsymbol{L},\boldsymbol{O}} \|\boldsymbol{L}\|_* + \alpha\|\boldsymbol{O}\|_1 \\ &\text{s.t. } \boldsymbol{L} + \boldsymbol{O} = \boldsymbol{X} \end{aligned} \tag{5.6}$$

where $\alpha > 0$ is the regularization parameter that needs to estimate in practice. The nuclear norm, which is the convex envelop of the rank, prompts low-rank while the ℓ_1-norm encourages sparsity. Although (5.6) is a convex optimization and the global minimum is guaranteed, it has a high computational cost even fast algorithms are employed because the full SVD to an $mn \times K$ matrix is required at each iteration [32,33,37]. The complexity of the full SVD is $\mathcal{O}(\max^3(mn^2, K))$. After separating the outlier component $\boldsymbol{O}$, the truncated SVD may be required to the "clean" data matrix $\boldsymbol{L}$ to ensure the rank to be r to achieve data compression or dimensionality reduction, where r is usually taken as $r \ll \min(mn, K)$. Then, the reconstruction given by the RPCA is

$$\widehat{\boldsymbol{X}} = \sum_{l=1}^{r} \sigma_l(\boldsymbol{L})\boldsymbol{y}_l\boldsymbol{z}_l^{\mathrm{T}}$$

where $\sigma_l(\boldsymbol{L})$ is the lth singular value of $\boldsymbol{L}$ while $\boldsymbol{y}_l \in \mathbb{R}^{mn}$ and $\boldsymbol{z}_l \in \mathbb{R}^K$ are the corresponding left and right singular vectors. In data compression, it only needs to store the so-called "principal components", i.e., the largest r singular values and the corresponding singular vectors $\{\sigma_l(\boldsymbol{L}), \boldsymbol{y}_l, \boldsymbol{z}_l\}_{l=1}^r$. Clearly, the compression ratio of the RPCA is the same as the PCA, which is $mnK/((mn + K + 1)r)$. Although the RPCA achieves outlier-robustness, it belongs to the vectorization based methods and thus, needs to handle a matrix of a much larger size due to transforming the original matrix into a long vector. Also, the RPCA breaks the 2D structure and the innate relation between row and column due to vectorization.

5.3 Greedy Pursuit Algorithms in ℓ_p-Space

The idea of greedy algorithms is to decompose the r-term approximation into a series of rank-one approximations. At each iteration, the greedy algorithms perform robust rank-one approximation in ℓ_p-space of the residual matrices obtained from the previous iteration. Then, the rank-one matrices are subtracted from the residual and never revisited.

5.3.1 Greedy Pursuit in ℓ_p-Space

The ℓ_p-GP for RLRAMM is described in Algorithm 9. It works in an iterative fashion. We use $(\boldsymbol{s}_i, \boldsymbol{u}_i, \boldsymbol{v}_i)$ and $\{\boldsymbol{R}_k^i\}_{k=1}^K$ to denote the solution and the K residual matrices, respectively, at the ith iteration. In the ith iteration, the GP finds rank-one approximation of $\left\{\boldsymbol{R}_k^{i-1}\right\}_{k=1}^K$, which is formally expressed as

$$\min f(\boldsymbol{s}, \boldsymbol{u}, \boldsymbol{v}) := \sum_{k=1}^{K} \left\| s_k \boldsymbol{u}\boldsymbol{v}^{\mathrm{T}} - \boldsymbol{R}_k^{i-1} \right\|_p^p \tag{5.7}$$

where $\boldsymbol{s} = [s_1, \cdots, s_K]^{\mathrm{T}}$ collects the K variables $\{s_k\}_{k=1}^K$ to be optimized. The optimal solution of (5.7) is taken as the solution of the ith iteration, which is denoted as $(\boldsymbol{s}_i, \boldsymbol{u}_i, \boldsymbol{v}_i)$. In some applications where the target rank r is given or can be estimated, the algorithm is terminated when $i > r$. If the target rank is unavailable, the normalized objective function, i.e., the sum of the pth power of the ℓ_p-norms of the K residual matrices

$$\frac{\sum_{k=1}^K \|\boldsymbol{R}_k^i\|_p^p}{\sum_{k=1}^K \|\boldsymbol{A}_k\|_p^p} \leq \delta \tag{5.8}$$

is adopted instead as the stopping criterion, where $\delta > 0$ is the tolerance. In Section 5.4, we will prove that the sequence $\left\{\sum_{k=1}^K \|\boldsymbol{R}_k^i\|_p^p\right\}_i$ converges to zero with an exponential rate. Therefore, (5.8) is well defined for any $\delta > 0$.

The remaining problem is how to efficiently solve the rank-one approximation of multiple matrices in the sense of ℓ_p-minimization, which is described in the next subsection.

5.3.2 Solution to ℓ_p-Rank-One Fitting

For the purpose of notational simplicity, we omit the superscript $(\cdot)^{i-1}$ of $\boldsymbol{R}_k^{i-1}$ and rewrite (5.7) as

$$\min f(\boldsymbol{s}, \boldsymbol{u}, \boldsymbol{v}) := \sum_{k=1}^{K} \left\| s_k \boldsymbol{u}\boldsymbol{v}^{\mathrm{T}} - \boldsymbol{R}_k \right\|_p^p \tag{5.11}$$

Algorithm 9 ℓ_p-GP for RLRAMM

```
**Input:** Matrices $\{\boldsymbol{A}_k\}_{k=1}^K$ and target rank $r$.
  **Initialization:** Set residual $\boldsymbol{R}_k^0 = \boldsymbol{A}_k$ for $k = 1, \cdots, K$.
  **for** $i = 1, 2, \cdots, r$ **do**
      Solve $\ell_p$-rank-one fitting
          $(\boldsymbol{s}_i, \boldsymbol{u}_i, \boldsymbol{v}_i) = \arg\min_{(\boldsymbol{s},\boldsymbol{u},\boldsymbol{v})} \sum_{k=1}^{K} \left\| s_k \boldsymbol{u}\boldsymbol{v}^{\mathrm{T}} - \boldsymbol{R}_k^{i-1} \right\|_p^p.$   (5.9)
      Update residual
          $\boldsymbol{R}_k^i = \boldsymbol{R}_k^{i-1} - s_{k,i}\boldsymbol{u}_i\boldsymbol{v}_i^{\mathrm{T}}, \ k = 1, \cdots, K.$   (5.10)
  **end for**
**Output:** $\boldsymbol{U} = [\boldsymbol{u}_1, \cdots, \boldsymbol{u}_r]$, $\boldsymbol{V} = [\boldsymbol{v}_1, \cdots, \boldsymbol{v}_r]$, and $\{\boldsymbol{s}_i\}_{i=1}^r$.
```

which is not easy to solve since the product term $s_k\boldsymbol{u}\boldsymbol{v}^{\mathrm{T}}$ is nonconvex and the ℓ_p-norm is nonsmooth when $p \leq 1$. The task of (5.11) is to robustly find the common dominant (rank-one) principal component of $\{\boldsymbol{R}_k\}_{k=1}^K$. By observing that there are three vectors to be optimized in (5.11), we use successive minimization strategy to solve it. That is, the objective function is minimized over one vector while the other two vectors are fixed. To be more specific, at the $(j+1)$th $(j = 0, 1, \cdots)$ iteration, f is successively minimized over $\boldsymbol{s}$, $\boldsymbol{u}$, and $\boldsymbol{v}$:

$$\boldsymbol{s}^{j+1} = \arg\min_{\boldsymbol{s}} f(\boldsymbol{s}, \boldsymbol{u}^j, \boldsymbol{v}^j) \tag{5.12}$$

$$\boldsymbol{u}^{j+1} = \arg\min_{\boldsymbol{u}} f(\boldsymbol{s}^{j+1}, \boldsymbol{u}, \boldsymbol{v}^j) \tag{5.13}$$

$$\boldsymbol{v}^{j+1} = \arg\min_{\boldsymbol{v}} f(\boldsymbol{u}^{j+1}, \boldsymbol{s}^{j+1}, \boldsymbol{v}). \tag{5.14}$$

Observing that $\{s_k\}_{k=1}^K$ are decoupled with each other and can be solved independently, for fixed $\boldsymbol{u}^j$ and $\boldsymbol{v}^j$, the optimal s_k minimizing the kth term of (5.11) is given by

$$s_k^{j+1} = \arg\min_{s_k \in \mathbb{R}} \left\| s_k \boldsymbol{u}^j (\boldsymbol{v}^j)^{\mathrm{T}} - \boldsymbol{R}_k \right\|_p^p$$

which amounts to the following scalar minimization problem

$$s_k^{j+1} = \arg\min_{s_k \in \mathbb{R}} \left\| s_k \boldsymbol{b}^j - \boldsymbol{r}_k \right\|_p^p \tag{5.15}$$

where $\boldsymbol{r}_k = \mathrm{vec}(\boldsymbol{R}_k) \in \mathbb{R}^{mn}$ and $\boldsymbol{b}^j = \mathrm{vec}\big(\boldsymbol{u}^j(\boldsymbol{v}^j)^{\mathrm{T}}\big) \in \mathbb{R}^{mn}$. When $p = 2$, s_k^{j+1} has the following closed-form solution

$$s_k^{j+1} = \frac{\langle \boldsymbol{b}^j, \boldsymbol{r}_k \rangle}{\langle \boldsymbol{b}^j, \boldsymbol{b}^j \rangle} = \frac{(\boldsymbol{u}^j)^{\mathrm{T}} \boldsymbol{R}_k \boldsymbol{v}^j}{\|\boldsymbol{u}^j\|^2 \|\boldsymbol{v}^j\|^2}$$

which merely needs $\mathcal{O}(mn)$ operations. When $p = 1$, denoting the lth elements of $\boldsymbol{b}^j$ and $\boldsymbol{r}_k$ as $\boldsymbol{b}^j(l)$ and $\boldsymbol{r}_k(l)$, respectively, (5.15) is rewritten as

$$\min_{s_k} \sum_{l=1}^{mn} \left|\boldsymbol{b}^j(l)\right| \left| s_k - \frac{\boldsymbol{r}_k(l)}{\boldsymbol{b}^j(l)} \right| \tag{5.16}$$

where $|\boldsymbol{b}^j(l)| > 0$ is the positive weight[1]. Defining a new vector

$$\boldsymbol{t}_k(l) = \frac{\boldsymbol{r}_k(l)}{\boldsymbol{b}^j(l)},\ l = 1, \cdots, mn \tag{5.17}$$

the optimal solution of (5.16) is the weighted median of $\boldsymbol{t}_k$ associated with the weight $|\boldsymbol{b}^j|$. We use the notation

$$s_k^{j+1} = \mathrm{WMED}\big(|\boldsymbol{b}^j|, \boldsymbol{t}_k\big) \tag{5.18}$$

to represent the weighted median, which can be computed by Algorithm 10 [135]. The major operation in the weighted median calculation is to sort the weighting coefficients. Thus, the computational complexity for exactly solving (5.15) with $p = 1$ is $\mathcal{O}(mn \log(mn))$ if the quick sorting algorithm is adopted.

Algorithm 10 Computation of Weighted Median

```
Input: Weight vector $|\boldsymbol{b}^j|$ and data vector $\boldsymbol{t}_k$.
Output: Weighted median $s_k^{j+1} = \mathrm{WMED}(|\boldsymbol{b}^j|, \boldsymbol{t}_k)$.
1. Determine the threshold $b_0 = \|\boldsymbol{b}^j\|_1/2$.
2. Sort $\boldsymbol{t}_k$ in ascending order with the corresponding concomitant weight $|\boldsymbol{b}^j|$.
3. Sum the concomitant weights, beginning with $|\boldsymbol{b}^j(1)|$ and increasing the order.
4. The weighted median is $\boldsymbol{t}_k(q)$ whose weight leads to the inequality $\sum_{l=1}^{q} |\boldsymbol{b}^j(l)| \geq b_0$ to hold first.
```

When $p > 1$, (5.15) is a scalar convex optimization problem. Since the objective function is twice differentiable and strictly convex, its unique global minimum can be obtained by the gradient or Newton's method. The complexity to obtain an ϵ-accuracy[2] solution is at most $\mathcal{O}(mn \log(1/\epsilon))$ because the gradient and Newton's methods have a global linear convergence rate at least for smooth convex optimization problems [145]. An algorithm with a complexity of $\mathcal{O}(m^2 n^2)$ has been provided in Appendix A.1 to

---

[1] Note that the zero element of $\boldsymbol{b}^j$ has no effect on the minimizer of (5.15). As a result, without loss of generality, it is assumed that $\boldsymbol{b}^j$ does not contain zero elements when finding the minimizer of (5.15).

[2] The tolerance ϵ denotes the accuracy of a solution obtained by an iterative algorithm, which is a small positive number, e.g., 10^{-6}.

exactly solve the more challenging case of $0 < p < 1$, where the objective function is neither differentiable nor convex. In summary, the complexities for calculating $\left\{s_k^{j+1}\right\}_{k=1}^K$ are $\mathcal{O}(mnK)$, $\mathcal{O}(mnK\log(mn))$, $\mathcal{O}(mnK\log(1/\epsilon))$, and $\mathcal{O}(m^2n^2K)$ for $p=2$, $p=1$, $p\in(1,2)$, and $p\in(0,1)$, respectively. The choice of $p=1$ is more robust and computationally simpler than the setting of $p\in(1,2)$. In addition, the choice of $p<1$ results in a quadratic complexity and it is not preferred unless the noise is very impulsive. Thus, it is preferred to choose $p=1$ by taking into account both robustness and complexity.

Now we discuss how to solve (5.13). Denoting $\boldsymbol{c}_k^j = s_k^{j+1}\boldsymbol{v}^j \in \mathbb{R}^n$, (5.13) is rewritten as

$$\min_{\boldsymbol{u}\in\mathbb{R}^m} \sum_{k=1}^K \left\|\boldsymbol{c}_k^j\boldsymbol{u}^{\mathrm{T}} - \boldsymbol{R}_k^{\mathrm{T}}\right\|_p^p$$

which amounts to the following m independent scalar minimization problems

$$\min_{u_l\in\mathbb{R}} \sum_{k=1}^K \left\|\boldsymbol{c}_k^j u_l - \boldsymbol{R}_k^{\mathrm{T}}(:,l)\right\|_p^p,\ l=1,\cdots,m$$

where u_l is the lth entry of $\boldsymbol{u}$ and $\boldsymbol{R}_k^{\mathrm{T}}(:,l)$ stands for the lth column of $\boldsymbol{R}_k^{\mathrm{T}}$. Defining $(\boldsymbol{c}^j)^{\mathrm{T}} = \left[(\boldsymbol{c}_1^j)^{\mathrm{T}},\cdots,(\boldsymbol{c}_K^j)^{\mathrm{T}}\right]$ and collecting $\boldsymbol{R}_1^{\mathrm{T}}(:,l),\cdots,\boldsymbol{R}_K^{\mathrm{T}}(:,l)$ into a vector $\boldsymbol{g}_l\in\mathbb{R}^{nK}$, u_l is updated by

$$u_l^{j+1} = \arg\min_{u_l\in\mathbb{R}} \left\|u_l\boldsymbol{c}^j - \boldsymbol{g}_l\right\|_p^p,\ l=1,\cdots,m \tag{5.19}$$

which can be solved with the use of the same method for solving (5.15). The complexities for calculating $\left\{u_l^{j+1}\right\}_{l=1}^m$ are $\mathcal{O}(mnK)$, $\mathcal{O}(mnK\log(nK))$, $\mathcal{O}(mnK\log(1/\epsilon))$, and $\mathcal{O}(mn^2K^2)$ for $p=2$, $p=1$, $p\in(1,2)$, and $p\in(0,1)$, respectively. Similarly, the qth element of $\boldsymbol{v}$, i.e., v_q is updated by

$$v_q^{j+1} = \arg\min_{v_q\in\mathbb{R}} \left\|v_q\boldsymbol{d}^j - \boldsymbol{h}_q\right\|_p^p,\ q=1,\cdots,n \tag{5.20}$$

where $(\boldsymbol{d}^j)^{\mathrm{T}} = \left[(\boldsymbol{d}_1^j)^{\mathrm{T}},\cdots,(\boldsymbol{d}_K^j)^{\mathrm{T}}\right]$ with $\boldsymbol{d}_k^j = s_k^{j+1}\boldsymbol{u}^{j+1}\in\mathbb{R}^m$ and $\boldsymbol{h}_q\in\mathbb{R}^{mK}$ is formed by concatenating $\boldsymbol{R}_1(:,q),\cdots,\boldsymbol{R}_K(:,q)$. It is seen that the complexities for calculating $\left\{v_q^{j+1}\right\}_{l=1}^n$ are $\mathcal{O}(mnK)$, $\mathcal{O}(mnK\log(mK))$, $\mathcal{O}(mnK\log(1/\epsilon))$, and $\mathcal{O}(m^2nK^2)$ for $p=2$, $p=1$, $p\in(1,2)$, and $p\in(0,1)$, respectively. The per-iteration costs of the ℓ_p-rank-one approximation with $p=1$ and $p\in(0,1)$ are

$$\begin{aligned}&\mathcal{O}(mnK(\log(mn)+\log(nK)+\log(mK)))\\ &=\mathcal{O}\big(mnK\log(m^2n^2k^2)\big)=\mathcal{O}(mnK\log(mnK))\end{aligned}$$

and

$$\begin{aligned}&\mathcal{O}\big(m^2n^2K+mn^2K^2+m^2nK^2\big)\\ &=\mathcal{O}(mnK\max^2(m,n,K))\end{aligned}$$

Table 5.1: Compression ratio and computational complexity.

Method	Compression ratio	Complexity
PCA	$\frac{mnK}{(mn+K+1)r}$	$\mathcal{O}(\max(m^2n^2, K^2)r)$
RPCA	$\frac{mnK}{(mn+K+1)r}$	$\mathcal{O}(\max^3(mn, K)N_{\text{RPCA}})$
2DPCA	$\frac{mnK}{(mK+n)r}$	$\mathcal{O}(mn^2K)$
GLRAM	$\frac{mnK}{(m+n+Kr)r}$	$\mathcal{O}((m+n)^2KrN_{\text{iter}})$
ℓ_2-GP	$\frac{mnK}{(m+n+K)r}$	$\mathcal{O}(mnKrN_{\text{iter}})$
ℓ_1-GP	$\frac{mnK}{(m+n+K)r}$	$\mathcal{O}(mnK\log(mnK)rN_{\text{iter}})$
$p \in (1, 2)$	$\frac{mnK}{(m+n+K)r}$	$\mathcal{O}(mnK\log(1/\epsilon)rN_{\text{iter}})$
$p \in (0, 1)$	$\frac{mnK}{(m+n+K)r}$	$\mathcal{O}(mnK\max^2(m, n, K)rN_{\text{iter}})$

respectively. The algorithm for solving the rank-one fitting of multiple matrices of (5.11) is summarized in Algorithm 11, where N_{iter} is the number of iterations for convergence. From the simulation results, Algorithm 11 converges fast. Typically, several tens of iterations are enough to converge with high accuracy. Furthermore, N_{iter} can be viewed as a constant independent of the dimension. Since (5.11) is nonconvex, the final convergence result relies on the initial values $\boldsymbol{u}^0$ and $\boldsymbol{v}^0$ theoretically. However, we find that random initialization always achieves good performance. Thus, random Gaussian vectors are adopted as the initial value in the numerical experiments.

The compression ratios and computational complexities of the PCA, RPCA, 2DPCA, GLRAM, and ℓ_p-GP with different values of p are compared in Table I. We see that the complexity of the most attractive setting of $p = 1$ is near-linear with the number and dimension of the matrices. Thus, the ℓ_1-GP which exhibits good robustness to outliers is near-scalable to problem size. The N_{RPCA} is the number of iterations of the ALM applied to solving the RPCA for convergence. It is known that the ALM is a first-order method and its convergence rate is generally slow. Thus, N_{RPCA} may be large for attaining a satisfactory solution.

5.3.3 Selection of p

The optimal p relies on the statistical properties of the noise. In the presence of outliers, $p < 2$ will bring a better performance than the most frequently used setting of $p = 2$. Roughly speaking, to select a proper p from $(0, 2)$, we need to consider the following two aspects.

Algorithm 11 ℓ_p-Rank-One Fitting of Multiple Matrices

```
Input: Matrices $\{\boldsymbol{R}_k\}_{k=1}^{K}$.
  Initialization: Set $\boldsymbol{u}^0$ and $\boldsymbol{v}^0$ randomly.
  Form
      $$\boldsymbol{r}_k = \mathrm{vec}(\boldsymbol{R}_k),\ \boldsymbol{g}_l = \begin{bmatrix} \boldsymbol{R}_1^{\mathrm{T}}(:,l) \\ \vdots \\ \boldsymbol{R}_K^{\mathrm{T}}(:,l) \end{bmatrix},\ \boldsymbol{h}_q = \begin{bmatrix} \boldsymbol{R}_1(:,q) \\ \vdots \\ \boldsymbol{R}_K(:,q) \end{bmatrix}$$
  for $k = 1,\cdots,K$, $l = 1,\cdots,m$, and $q = 1,\cdots,n$.
  for $j = 0,1,2,\cdots$ do
      Form $\boldsymbol{b}^j = \mathrm{vec}\big(\boldsymbol{u}^j(\boldsymbol{v}^j)^{\mathrm{T}}\big)$.
      Update $\boldsymbol{s}$:
          $$s_k^{j+1} = \arg\min_{s_k\in\mathbb{R}} \big\| s_k\boldsymbol{b}^j - \boldsymbol{r}_k \big\|_p^p,\ k = 1,\cdots,K.$$
      Form $(\boldsymbol{c}^j)^{\mathrm{T}} = \big[(\boldsymbol{c}_1^j)^{\mathrm{T}},\cdots,(\boldsymbol{c}_K^j)^{\mathrm{T}}\big]$ with $\boldsymbol{c}_k^j = s_k^{j+1}\boldsymbol{v}^j$.
      Update $\boldsymbol{u}$:
          $$u_l^{j+1} = \arg\min_{u_l\in\mathbb{R}} \big\| u_l\boldsymbol{c}^j - \boldsymbol{g}_l \big\|_p^p,\ l = 1,\cdots,m.$$
      Form $(\boldsymbol{d}^j)^{\mathrm{T}} = \big[(\boldsymbol{d}_1^j)^{\mathrm{T}},\cdots,(\boldsymbol{d}_K^j)^{\mathrm{T}}\big]$ with $\boldsymbol{d}_k^j = s_k^{j+1}\boldsymbol{u}^{j+1}$.
      Update $\boldsymbol{v}$:
          $$v_q^{j+1} = \arg\min_{v_q\in\mathbb{R}} \big\| v_q\boldsymbol{d}^j - \boldsymbol{h}_q \big\|_p^p,\ q = 1,\cdots,n.$$
      Stop until convergence satisfies.
  end for
Output: $(\boldsymbol{s}^{j+1},\boldsymbol{u}^{j+1},\boldsymbol{v}^{j+1})$.
```

1) Statistical perspective. The more impulsive the noise is, the smaller value of p is preferred. If the noise is not so impulsive, the choice of $1 \leq p < 2$ is suitable. If the noise is too impulsive and has a more spike-like property, $p < 1$ may be required.

2) Computational perspective. As p decreases to zero, the nonconvexity and nonsmoothness of the ℓ_p-norm becomes stronger, which brings more difficulties in minimization. The computational challenges induced by a very small p includes increased probability of being trapped into local minima far away from the global minimum and slow convergence rate. Therefore, it is not recommended to choose p close to 0.

To summarize, choosing an appropriate p is a trade-off between the statistical and computational aspects. It is preferred to choose $p = 1$ since the resultant subproblems can be efficiently solved based on weighted medians and the ℓ_1-norm is quite robust to outliers. If there is no prior information for the noise, we can resort to cross-validation [31] to determine p.

5.4 Convergence Analysis

5.4.1 ℓ_p-Correlation

We use the concept of ℓ_p-correlation [135] to prove the convergence of the ℓ_p-GP. The normalized ℓ_p-correlation coefficient of two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ is defined as

$$\theta_p(\boldsymbol{a}, \boldsymbol{b}) \triangleq 1 - \frac{\min\limits_{\alpha \in \mathbb{R}} \|\boldsymbol{b} - \alpha \boldsymbol{a}\|_p^p}{\|\boldsymbol{b}\|_p^p} \tag{5.21}$$

which satisfies $0 \leq \theta_p(\boldsymbol{a}, \boldsymbol{b}) \leq 1$. When $\theta_p(\boldsymbol{a}, \boldsymbol{b}) = 0$, $\boldsymbol{a}$ and $\boldsymbol{b}$ are called ℓ_p-orthogonal [135]. When $\theta_p(\boldsymbol{a}, \boldsymbol{b}) = 1$, $\boldsymbol{a}$ and $\boldsymbol{b}$ are colinear, i.e., $\boldsymbol{b} = \beta \boldsymbol{a}$ with $\beta \in \mathbb{R}$. If $\boldsymbol{a}$ or $\boldsymbol{b}$ is random, the probability of $\theta_p(\boldsymbol{a}, \boldsymbol{b}) = 0$ or $\theta_p(\boldsymbol{a}, \boldsymbol{b}) = 1$ is zero. That is, the probability of $0 < \theta_p(\boldsymbol{a}, \boldsymbol{b}) < 1$ is 1 for random $\boldsymbol{a}$ and $\boldsymbol{b}$. When $p = 2$, $\theta_p(\boldsymbol{a}, \boldsymbol{b})$ has the closed-form expression

$$\theta_2(\boldsymbol{a}, \boldsymbol{b}) = \frac{\langle \boldsymbol{a}, \boldsymbol{b} \rangle^2}{\|\boldsymbol{a}\|^2 \|\boldsymbol{b}\|^2}$$

that we are familiar with. The minimum is then formulated as

$$\left\{ \min_{\alpha \in \mathbb{R}} \|\boldsymbol{b} - \alpha \boldsymbol{a}\|_p^p \right\} = (1 - \theta_p(\boldsymbol{a}, \boldsymbol{b})) \|\boldsymbol{b}\|_p^p. \tag{5.22}$$

The larger $\theta_p(\boldsymbol{a}, \boldsymbol{b})$, the more "similar" (correlated) $\boldsymbol{a}$ and $\boldsymbol{b}$ are in the ℓ_p-space and the smaller value $\{\min_\alpha \|\boldsymbol{b} - \alpha\boldsymbol{a}\|_p^p\}$ can attain. Armed with (5.22), we are ready to prove the following descent lemma for convergence analysis. The lemma states that each iteration of Algorithm 11 guarantees a decrease of the objective function.

5.4.2 Descent Lemma

Lemma 3 *At the $(j+1)$th iteration of Algorithm 11 for solving the rank-one approximation to $\left\{\boldsymbol{R}_k^{i-1}\right\}_{k=1}^{K}$, the objective function achieves a decrease of*

$$f(\boldsymbol{s}^{j+1}, \boldsymbol{u}^{j+1}, \boldsymbol{v}^{j+1}) \leq \zeta_{i-1}^{j} \sum_{k=1}^{K} \|\boldsymbol{R}_k^{i-1}\|_p^p \tag{5.23}$$

where $0 < \zeta_{i-1}^{j} < 1$ is the decay factor of the $(j+1)$th iteration, whose subscript $(\cdot)_{i-1}$ implies that its value relates to $\left\{\boldsymbol{R}_k^{i-1}\right\}_{k=1}^{K}$. Algorithm 11 monotonically decreases the objective function indicates that $\left\{\zeta_{i-1}^{j}\right\}$ is a monotonically decreasing sequence. After $j = N_{\text{iter}}$ or $j \to \infty$ iterations, the minimum that Algorithm 11 finds is guaranteed as

$$\begin{aligned}\sum_{k=1}^{K} \|\boldsymbol{R}_k^{i}\|_p^p &= \min_{\boldsymbol{s},\boldsymbol{u},\boldsymbol{v}} f(\boldsymbol{s}, \boldsymbol{u}, \boldsymbol{v}) \\ &\leq \zeta_{i-1} \sum_{k=1}^{K} \|\boldsymbol{R}_k^{i-1}\|_p^p\end{aligned} \tag{5.24}$$

with $\zeta_{i-1} = \zeta_{i-1}^{N_{\text{iter}}}$ or $\zeta_{i-1} = \lim_{j\to\infty} \zeta_{i-1}^{j}$ and $0 < \zeta_{i-1} < 1$.

Proof. It is seen that minimizing f with respect to $\boldsymbol{s}$ at the jth iteration in Algorithm 11 yields

$$\begin{aligned}\left\{\min_{\boldsymbol{s}\in\mathbb{R}^K} f(\boldsymbol{s}, \boldsymbol{u}^j, \boldsymbol{v}^j)\right\} &= \sum_{k=1}^{K} \min_{s_k\in\mathbb{R}} \left\| s_k \boldsymbol{b}^j - \boldsymbol{r}_k \right\|_p^p \\ &= \sum_{k=1}^{K} \left(1 - \theta_p(\boldsymbol{b}^j, \boldsymbol{r}_k)\right) \|\boldsymbol{r}_k\|_p^p \\ &\leq \alpha_p(\boldsymbol{b}^j) \sum_{k=1}^{K} \|\boldsymbol{r}_k\|_p^p \\ &= \alpha_p(\boldsymbol{b}^j) \sum_{k=1}^{K} \|\boldsymbol{R}_k^{i-1}\|_p^p\end{aligned} \tag{5.25}$$

where

$$\alpha_p(\boldsymbol{b}^j) = \max_{1\leq k\leq K} \left(1 - \theta_p(\boldsymbol{b}^j, \boldsymbol{r}_k)\right) \tag{5.26}$$

and $\|\boldsymbol{r}_k\|_p^p = \|\boldsymbol{R}_k^{i-1}\|_p^p$ due to $\boldsymbol{r}_k = \mathrm{vec}(\boldsymbol{R}_k^{i-1})$. Observing that $\boldsymbol{b}^j$ is random since the initialization is random, we have $1 < \theta_p(\boldsymbol{b}^j, \boldsymbol{r}_k) < 1$ and thus, $0 < \alpha_p(\boldsymbol{b}^j) < 1$. This yields a strict decrease of the objective function. Similarly, minimizing f with respect to $\boldsymbol{u}$ leads to

$$\begin{aligned}\left\{\min_{\boldsymbol{u}\in\mathbb{R}^m} f(\boldsymbol{s}^{j+1}, \boldsymbol{u}, \boldsymbol{v}^j)\right\} &= \sum_{l=1}^{m} \min_{u_l\in\mathbb{R}} \left\| u_l \boldsymbol{c}^j - \boldsymbol{g}_l \right\|_p^p \\ &= \sum_{l=1}^{m} \left(1 - \theta_p(\boldsymbol{c}^j, \boldsymbol{g}_l)\right) \|\boldsymbol{g}_l\|_p^p \\ &\le \beta_p(\boldsymbol{c}^j) \sum_{l=1}^{m} \|\boldsymbol{g}_l\|_p^p \\ &= \beta_p(\boldsymbol{c}^j) \sum_{k=1}^{K} \|\boldsymbol{R}_k^{i-1}\|_p^p\end{aligned} \tag{5.27}$$

where

$$\beta_p(\boldsymbol{c}^j) = \max_{1\le l\le m} \left(1 - \theta_p(\boldsymbol{c}^j, \boldsymbol{g}_l)\right) \tag{5.28}$$

and $\sum_{l=1}^m \|\boldsymbol{g}_l\|_p^p = \sum_{k=1}^K \|\boldsymbol{R}_k^{i-1}\|_p^p$ has been used. Again, $0 < \beta_p(\boldsymbol{c}^j) < 1$ leads to a strict decrease of the objective function. Minimizing f with respect to $\boldsymbol{v}$ results in

$$\begin{aligned}\left\{\min_{\boldsymbol{v}\in\mathbb{R}^n} f(\boldsymbol{s}^{j+1}, \boldsymbol{u}^{j+1}, \boldsymbol{v})\right\} &= \sum_{q=1}^{n} \min_{v_q\in\mathbb{R}} \left\| v_q \boldsymbol{d}^j - \boldsymbol{h}_q \right\|_p^p \\ &= \sum_{q=1}^{n} \left(1 - \theta_p(\boldsymbol{d}^j, \boldsymbol{h}_q)\right) \|\boldsymbol{h}_q\|_p^p \\ &\le \gamma_p(\boldsymbol{d}^j) \sum_{q=1}^{n} \|\boldsymbol{h}_q\|_p^p \\ &= \gamma_p(\boldsymbol{d}^j) \sum_{k=1}^{K} \|\boldsymbol{R}_k^{i-1}\|_p^p\end{aligned} \tag{5.29}$$

where

$$\gamma_p(\boldsymbol{d}^j) = \max_{1\le q\le n} \left(1 - \theta_p(\boldsymbol{d}^j, \boldsymbol{h}_q)\right) \tag{5.30}$$

and $\sum_{q=1}^n \|\boldsymbol{h}_q\|_p^p = \sum_{k=1}^K \|\boldsymbol{R}_k^{i-1}\|_p^p$ has been used. Once again, $0 < \gamma_p(\boldsymbol{d}^j) < 1$ strictly decreases of the objective function. Combining (5.25), (5.27), and (5.29), we obtain

$$\begin{aligned}f(\boldsymbol{s}^{j+1}, \boldsymbol{u}^{j+1}, \boldsymbol{v}^{j+1}) &\le \alpha_p(\boldsymbol{b}^j)\beta_p(\boldsymbol{c}^j)\gamma_p(\boldsymbol{d}^j) \sum_{k=1}^{K} \|\boldsymbol{R}_k^{i-1}\|_p^p \\ &= \zeta_{i-1}^j \sum_{k=1}^{K} \|\boldsymbol{R}_k^{i-1}\|_p^p\end{aligned}$$

and hence the upper bound of the minimum in (5.24) by incorporating (5.11). □

Note that the decay ratio $0 < \zeta_{i-1} < 1$ depends on the ℓ_p-correlation of the intermediate variables and the vectorization forms of the residual matrices $\left\{\boldsymbol{R}_k^{i-1}\right\}_{k=1}^{K}$. As the iteration progress, the iterates approximate the common principal component of $\left\{\boldsymbol{R}_k^{i-1}\right\}_{k=1}^{K}$ better as the iteration progress. Hence, the ℓ_p-correlation of the iterates and the principal component is improved and the decay ratio ζ_{i-1} becomes smaller.

5.4.3 Convergence of ℓ_p-GP

Theorem 6 *The objective function of the ℓ_p-GP for RLRAMM in Algorithm 9 decays exponentially:*

$$\sum_{k=1}^{K} \|\boldsymbol{R}_k^i\|_p^p \leq \rho^i \sum_{k=1}^{K} \|\boldsymbol{A}_k\|_p^p \tag{5.31}$$

for the iteration number $i = 0, 1, 2, \cdots$, where $0 < \rho < 1$ is a worst-case bound of the convergence rate.

Proof. Successively applying (5.24), we at once obtain

$$\sum_{k=1}^{K} \|\boldsymbol{R}_k^i\|_p^p \leq \left(\prod_{l=0}^{i-1} \zeta_l\right) \sum_{k=1}^{K} \|\boldsymbol{R}_k^0\|_p^p = \rho^i \sum_{k=1}^{K} \|\boldsymbol{A}_k\|_p^p \tag{5.32}$$

where

$$\rho = \max_{0 \leq l \leq i-1} \zeta_l \tag{5.33}$$

and $\boldsymbol{R}_k^0 = \boldsymbol{A}_k$ has been used. Since the decay ratio satisfies $0 < \rho < 1$, the reconstruction error strictly decreases at each iteration and the GP algorithm converges with a worst decay rate of ρ. ☐

Note that the decay rate $0 < \rho < 1$ depends on the ℓ_p-correlation of the intermediate variables $(\boldsymbol{s}, \boldsymbol{u}, \boldsymbol{v})$ and the vectorization forms of the residual matrices $\left\{\boldsymbol{R}_k^{i-1}\right\}_{k=1}^{K}$. The higher the correlation is, the faster the algorithm converges. By Theorem 6, the reconstruction error approaches zero

$$\lim_{i \to \infty} \sum_{k=1}^{K} \|\boldsymbol{R}_k^i\|_p^p = 0 \tag{5.34}$$

due to $\rho \in (0, 1)$. This implies that the stopping criterion in (5.8) is well defined for any $\delta > 0$. Obviously, (5.34) also indicates

$$\lim_{i \to \infty} \|\boldsymbol{R}_k^i\|_p^p = 0, \text{ and } \lim_{i \to \infty} \boldsymbol{R}_k^i = \boldsymbol{0}, \; k = 1, \cdots, K. \tag{5.35}$$

As a direct conclusion obtained from Theorem 6, the following corollary allows an infinite series expansion for an arbitrary set of matrices $\{\boldsymbol{A}_k\}_{k=1}^{K}$.

Corollary 2 *For any matrix set* $\{\boldsymbol{A}_k\}_{k=1}^K$*, the* ℓ_p*-GP algorithm leads to an infinite series expansion, which is shown as*

$$\boldsymbol{A}_k = \sum_{i=1}^{\infty} s_{k,i}\boldsymbol{u}_i\boldsymbol{v}_i^{\mathrm{T}}, \ k = 1, \cdots, K \tag{5.36}$$

where $(\boldsymbol{u}_i, \boldsymbol{v}_i, \boldsymbol{s}^i)$ *is the result obtained by Algorithm 9 at the ith iteration.*

Proof. Successive application of the residual update formula of (5.10) results in

$$\begin{aligned}\boldsymbol{R}_k^i &= \boldsymbol{R}_k^{i-1} - s_{k,i}\boldsymbol{u}_i\boldsymbol{v}_i^{\mathrm{T}}\\ &= \boldsymbol{R}_k^0 - \sum_{l=1}^{i} s_{k,l}\boldsymbol{u}_l\boldsymbol{v}_l^{\mathrm{T}}\\ &= \boldsymbol{A}_k - \sum_{l=1}^{i} s_{k,l}\boldsymbol{u}_l\boldsymbol{v}_l^{\mathrm{T}}\end{aligned}$$

which is rewritten as

$$\boldsymbol{A}_k = \sum_{l=1}^{i} s_{k,l}\boldsymbol{u}_l\boldsymbol{v}_l^{\mathrm{T}} + \boldsymbol{R}_k^i. \tag{5.37}$$

Exploiting $\lim_{i\to\infty} \boldsymbol{R}_k^i = \boldsymbol{0}$ in (5.35) and taking limits as $i \to \infty$ on both sides of (5.37) yields (5.36). □

In practical applications, mn is usually very large. Generally, a target rank $r \ll mn$ is enough to capture the low-rank structure of natural images and achieves a small reconstruction error.

5.5 Experimental Results

In addition to synthetic random data, the following three real-world databases, including two face datasets, and one object dataset, are used in the experiments.

- ORL face database [141]. It consists of 10 different images of each of 40 distinct subjects for a total of 400 images. The resolution of the gray-scale images is 112×92 and we have $m = 92$, $n = 112$, and $K = 400$.
- Georgia Tech face database [142]. It contains 750 images of 50 individuals. There are 15 images for each individual. The original images are colored and with different sizes. We convert them to gray-scale with the same size of 111×156 so that they can be represented by matrices. We have $m = 156$, $n = 111$, and $K = 750$.

- COIL-20 database [144]. There are 1440 gray-scale images of 20 different objects, which corresponds to 72 images per object. The image resolution is 128×128 and it follows $m = n = 128$ and $K = 1440$.

The normalized mean square error (NMSE) defined by

$$\text{NMSE} = \frac{\sum_{k=1}^{K} \|\boldsymbol{U}\boldsymbol{S}_k\boldsymbol{V}^{\mathrm{T}} - \boldsymbol{A}_k^0\|_{\mathrm{F}}^2}{\sum_{k=1}^{K} \|\boldsymbol{A}_k^0\|_{\mathrm{F}}^2}$$

is adopted as the performance measure, where $\{\boldsymbol{A}_k^0\}_{k=1}^K$ are the true (noiseless) matrices. The noisy observed version is $\boldsymbol{A}_k = \boldsymbol{A}_k^0 + \boldsymbol{N}_k$ where $\boldsymbol{N}_k$ is the noise matrix containing outliers.

5.5.1 Convergence Behaviors

The convergence behaviors are investigated using random data. We set $m = 40$, $n = 50$, and $K = 20$. A set of noise-free matrices of rank 10 is generated by $\boldsymbol{A}_k^0 = \boldsymbol{U}\boldsymbol{S}_k\boldsymbol{V}^{\mathrm{T}},\ k = 1, \cdots, K$, where the entries of $\boldsymbol{U} \in \mathbb{R}^{m\times 5}$ and $\boldsymbol{V} \in \mathbb{R}^{5\times n}$ satisfy the standard Gaussian distribution while the diagonal entries of $\boldsymbol{S}_k$ are uniformly distributed in $[1, 2]$ to avoid any diagonal entry being too close to zero. Then, the noise matrices $\boldsymbol{N}_k$ are added to $\boldsymbol{A}_k^0$ to obtain $\boldsymbol{A}_k$. Each entry of $\boldsymbol{N}_k$ satisfies the two-term zero-mean Gaussian mixture model (GMM) whose probability density function is given by

$$p_\nu(\nu) = \sum_{i=1}^{2} \frac{c_i}{\sqrt{2\pi}\sigma_{\nu_i}} \exp\left(-\frac{\nu^2}{2\sigma_{\nu_i}^2}\right)$$

where $0 \leq c_i \leq 1$ and $\sigma_{\nu_i}^2$ are the probability and variance of the ith term, respectively, with $c_1 + c_2 = 1$. If $\sigma_{\nu_1}^2 \ll \sigma_{\nu_2}^2$ and $c_2 < c_1$ are selected, large noise samples of variance $\sigma_{\nu_2}^2$ occurring with a smaller probability c_2 can be viewed as outliers embedded in Gaussian background noise of variance $\sigma_{\nu_1}^2$. Thus, the GMM well models the phenomenon with both Gaussian noise and outliers. The total noise variance is $\sigma_\nu^2 = \sum_i c_i \sigma_{\nu_i}^2$ and the signal-to-noise ratio (SNR) is defined as $\text{SNR} = \left(\sum_{k=1}^{K} \|\boldsymbol{A}_k^0\|_{\mathrm{F}}^2\right)/(mnK\sigma_v^2)$.

Figures 5.1 and 5.2 plot the normalized objective function defined in (5.8) versus iteration number with $p = 2, 1.5, 1$, and 0.8 at SNR = 3 dB. Note that the normalized objective function is different from the NMSE. The former uses the noisy observations but the latter employs the true matrices. The latter reflects the estimation accuracy while the former does not. As we see, the normalized objective functions for all values of p monotonically decrease, which validates Theorem 6. The NMSE rapidly decreases

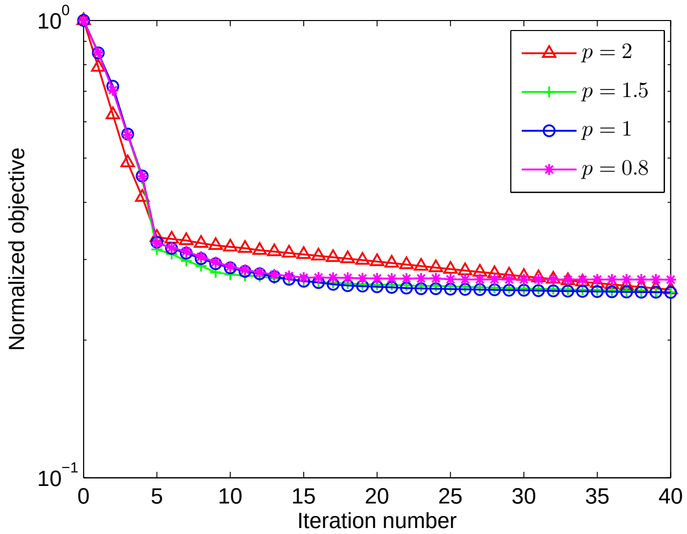


Figure 5.1: Normalized objective function versus iteration number.

when $i \le 5$. For $p = 1$ and $p = 0.8$, the NMSE continues to decrease to a lower value after capturing the dominant low-rank structure, achieving a robust estimation. But for $p = 2$ and $p = 1.5$, the NMSE cannot further decrease. As the iteration progresses, the NMSEs of $p = 2$ and $p = 1.5$ even increases, implying that overfitting is easier to appear for larger p. Therefore, the ℓ_p-GP with larger p is not robust against to outliers while that with smaller p has good robustness.

5.5.2 Results of Robust Image Reconstruction

The performances of image reconstruction of the ℓ_p-GP are compared with the PCA, RPCA, 2DPCA, and GLARM in the presence of outliers. For RPCA, after separating the outlier components, the truncated SVD is performed to calculate principal components to achieve data compression. When processing image data, we first linearly map the pixel values from $[0, 255]$ to $[-0.5, 0.5]$, which is achieved by $\boldsymbol{A}_k \to \boldsymbol{A}_k/255 - 0.5$. The salt-and-pepper noise is used as the outliers and is added to the images. We use the function "`imnoise(`$\boldsymbol{A}_k$`, 'salt & pepper', `σ_n^2`)`" in MATLAB, where the normalized noise intensity is σ_n^2 corresponding to SNR $= 1/\sigma_n^2$, to generate the salt-and-pepper noise. For fair comparison, the NMSEs of the six methods are computed under the same (or close) compression ratios. According to Table I, r_1, r_2, r_3, and r_4, which are the target ranks of PCA/RPCA, 2DPCA, GLARM, and ℓ_p-GP, respectively, should

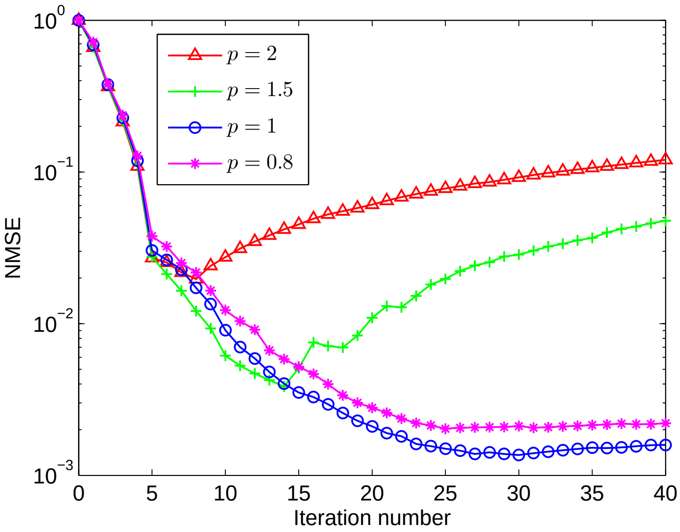


Figure 5.2: NMSE versus iteration number.

satisfy

$$\begin{aligned}(mn+K+1)r_1 &= (mK+n)r_2\\ &= (m+n+Kr_3)r_3 = (m+n+K)r_4.\end{aligned} \tag{5.38}$$

to make the compression ratios of the six methods the same. Noting that $r_1, \cdots, r_4$ are positive integers, (5.38) may not be strictly satisfied. We select the positive integers such that the compression ratios are as close as possible.

We set SNR = 6 dB. Figure 5.3 shows the NMSEs versus the iteration number on the three real-world image databases. It is observed that the ℓ_2-GP and ℓ_1-GP significantly decrease the NMSE at the beginning stage. This implies that these real-world images exhibit several “principal components” and the ℓ_1-GP successfully capture these components although they are not strictly low-rank. The maximum iteration numbers r are set to 149, 231, and 378 for the ORL, Georgia Tech, and COIL-20 databases, respectively. To achieve a sufficiently small NMSE, the maximum iteration number r needs to be larger than $\min(m,n)$ but we still have $r \ll mn$. Again, the ℓ_1-GP is much more robust to outliers than the ℓ_2-GP since the former attains much smaller NMSEs.

The ranks, compression ratios, and NMSEs of the PCA, RPCA, 2DPCA, GLARM, ℓ_2-GP for the three image databases, are listed in Table II. Figures 5.4, 5.5, and 5.6 display several samples of the reconstructed images obtained by the six algorithms on the ORL, Georgia Tech, and COIL-20 databases, respectively. The corresponding samples of the original and noisy images are also shown for comparison. Evidently, the

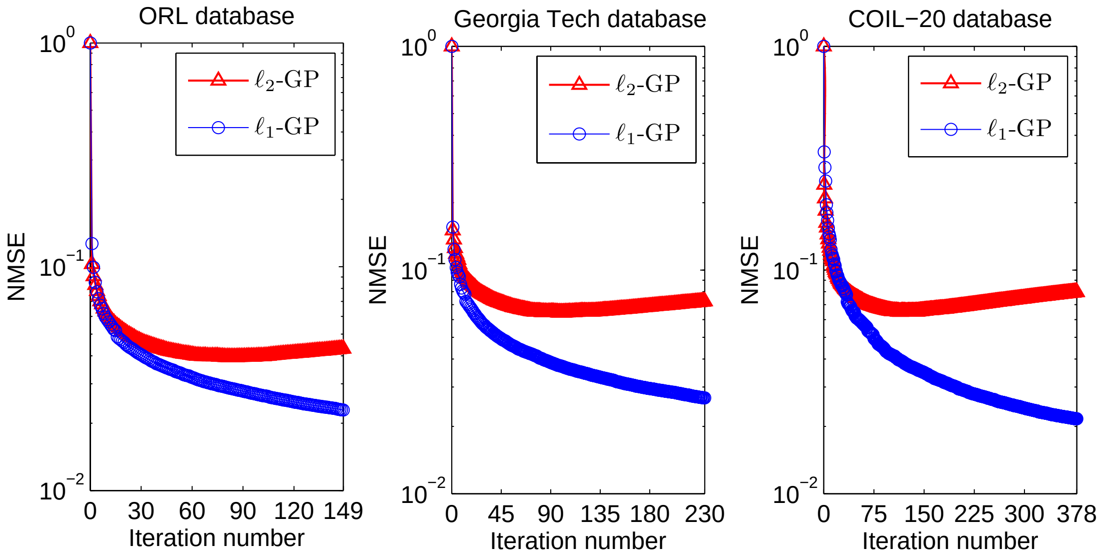


Figure 5.3: NMSEs versus iteration number of ℓ_2-GP and ℓ_1-GP on ORL, Georgia Tech, and COIL-20 databases.

ℓ_1-GP has the best performance in the presence of outliers. Although the RPCA also exhibits robustness against outliers, it is inferior to the ℓ_1-GP. Among the four non-robust methods, the GLRAM and ℓ_2-GP perform better than the PCA and 2DPCA under similar compression ratios.

We then investigate how the NMSE varies with the compression ratio. Figure 5.7 plots the NMSEs of the six methods versus compression ratio on the three image databases. The ℓ_1-GP has the best reconstruction performance for all databases. The ℓ_2-GP loses robustness since the Frobenius norm is sensitive to outliers. The NMSEs of two robust schemes, say, RPCA and ℓ_1-GP monotonically increase with the compression ratio. However, those of the four non-robust schemes do not monotonically increase with the compression ratio. Sometimes, lower compression ratios yields worse NMSEs. This is because the non-robust schemes cannot eliminate the adverse affect induced by the outliers. The principal components found by the four schemes do not capture the true low-rank structure but reflect the outliers. Despite the low computational cost of the 2DPCA, its performance is not satisfactory because it only uses a single side transform resulting limited compression capability. Although the RPCA is more robust than the PCA, it has a large performance gap compared with the ℓ_1-GP. Sometimes it is even inferior to the GLRAM. The advantage of 2D based methods over the vectorization based one is evident when handling multiple matrices.

Table 5.2: Compression ratio (CR) and NMSE.

		PCA	RPCA	2DPCA	GLRAM	ℓ_2-GP	ℓ_1-GP
ORL	rank	8	8	2	15	149	149
	CR	48.1	48.1	45.9	44.3	45.8	45.8
	NMSE (10^{-2})	6.06	4.82	6.50	3.37	4.31	2.29
Georgia Tech	rank	13	13	2	17	230	230
	CR	55.3	55.3	55.4	58.7	55.5	55.5
	NMSE (10^{-2})	9.20	5.75	10.26	5.68	7.24	2.68
COIL-20	rank	22	22	4	22	378	378
	CR	31.3	31.3	32.0	33.3	32.0	32.0
	NMSE (10^{-2})	8.51	3.94	9.61	6.42	7.98	2.16

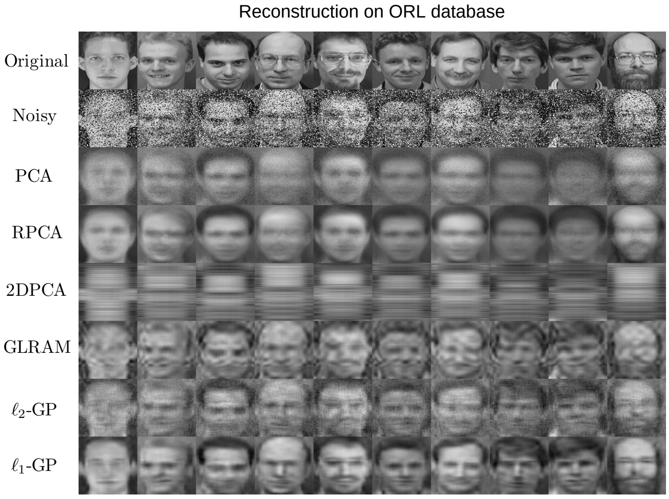


Figure 5.4: Samples of original, noisy, and reconstructed images on ORL face database.

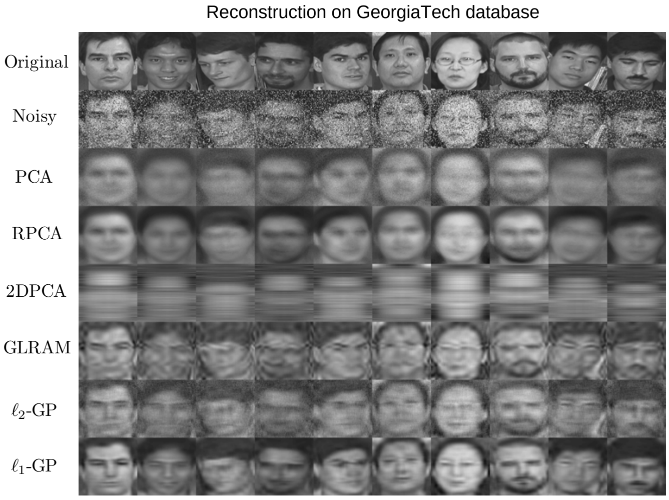


Figure 5.5: Samples of original, noisy, and reconstructed images on Georgia Tech face database. The compression ratio of the six methods is around 55.5, which implies that only 1.8% storage space of the original data is needed after compression.

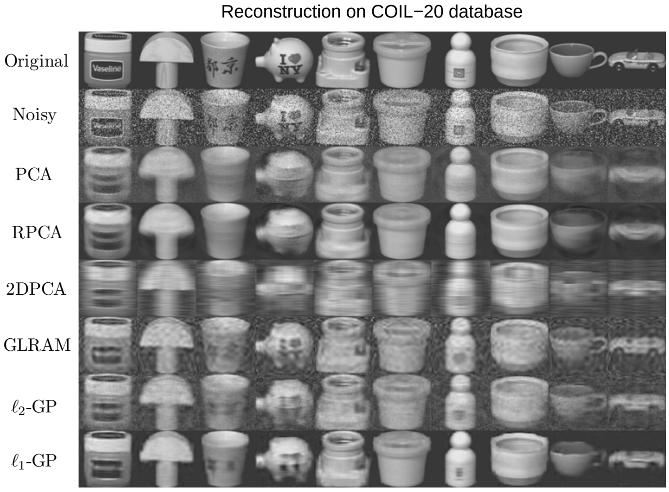


Figure 5.6: Samples of original, noisy, and reconstructed images on COIL-20 database of objectives.

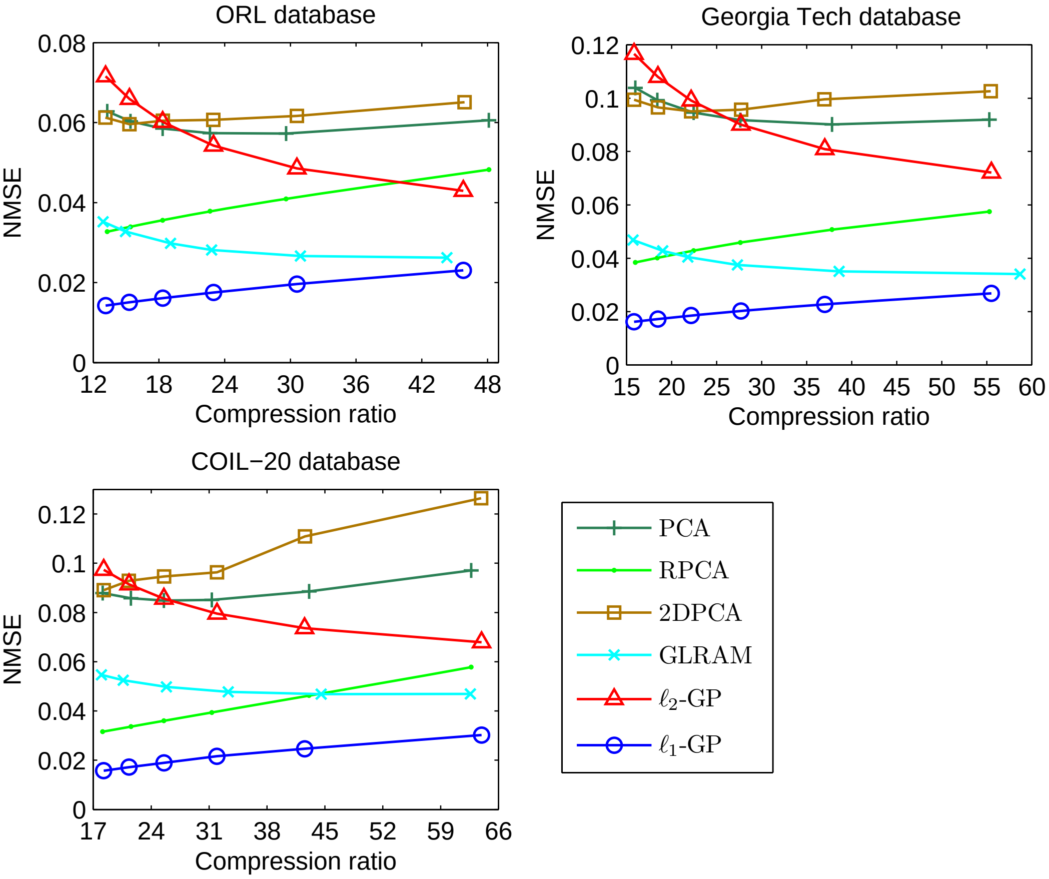


Figure 5.7: NMSEs of PCA, RPCA, 2DPCA, GLARM, ℓ_2-GP, and ℓ_1-GP versus compression ratio on three real-world databases.

5.6 Summary

An ℓ_p-GP algorithm with tunable value of p is devised for RLRAMM. It is near-scalable to the problem size and computationally more efficient than the vectorization based counterpart RPCA. Compared with the RPCA, it provides a different perspective for robust low-rank representation. It has a higher compression ratio and more robust to outliers than the 2DPCA and GLRAM. The convergence theory of the ℓ_p-GP, which is different from that of Chapter 4, is developed. We prove that the sum of the ℓ_p-norms of the residuals converges exponentially. We reveal that the worst-case bound of the convergence rate is related to the ℓ_p-correlation of the residual and the current iterates.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

This dissertation contributes to devising effective and efficient algorithms for outlier-robust low-rank approximation of matrices with theoretical convergence analysis. A variety of applications to signal processing, computer vision and machine learning are also investigated.

In Chapter 1, the background, motivation and organization of the thesis were introduced and an overview of the state-of-the-art techniques for low-rank approximation were given.

Chapter 2 focuses on low-rank approximation of a single matrix whose all entries are known. Using entry-wise ℓ_p-norm minimization, three algorithms are designed to achieve outlier-robustness. The first algorithm is the IR-SVD, where the SVD of a reweighted data matrix is performed at each iteration. The second is the AM, where the objective function is minimized over one factored matrix while the other factor is fixed. Convergence of the AM is proved. Two complex-valued Newton's methods with optimal step size are proposed to solve the resulting ℓ_p-fitting problems. It is revealed that the IRLS is a special case of the pseudo-Newton's method. The third is the ADMM. It casts the difficult nonsmooth ℓ_1-subspace decomposition into an ℓ_2-one, which can be efficiently solved via the truncated SVD with a marginal computational increase of soft-thresholding. Experimental results on random data verify the superior performance of the proposed methodology. Wide applicability of the techniques of this chapter is demonstrated by the application examples to DOA estimation, image demixing and video surveillance.

Chapter 3 addresses the problem of low-rank approximation where there are missing entries, which is referred to as matrix completion. We devise two algorithms for robust matrix completion using low-rank factorization via ℓ_p-minimization. The first method tackles the nonconvex factorization with missing data by iteratively solving multiple independent linear ℓ_p-regressions. On the other hand, the second solution exploits the ADMM for incomplete factorization in ℓ_p-space. Each iteration of the ADMM requires solving a least squares factorization problem and calculating the proximity operator of the ℓ_p-norm. The two algorithms have comparable recovery performance as well as computational efficiency and allow parallel or distributed realization. Their total time complexity is $\mathcal{O}(K|\Omega|r^2)$, where $|\Omega|$ is the number of observations, r is the rank, and K is a fixed constant of several hundreds to thousands. It is lower than the popular schemes employing the nuclear norm and Schatten p-norm minimization that require full SVD. Furthermore, our solutions generalize the conventional matrix factorization based on Frobenius norm minimization. The superiority of the developed algorithms over the SVT, SVP, and AP in terms of implementation complexity, recovery capability and outlier-robustness, is demonstrated using synthetic and real-world data.

Chapter 4 investigates learning the common low-rank structure of multiple matrices, which is an extension of the single matrix case. A main advantage of the LRAMM is that it does not convert matrices into vectors and thus can avoid processing the matrix with much larger size than the original ones. A greedy algorithmic framework including three variants, GP, EGP and OGP is designed for this learning task. The GP works in an iterative manner. At each iteration, it finds a rank-one approximation of the residuals. For GP and OGP, an alternating optimization scheme is devised for the rank-one fitting problem while for the EGP, just an approximate solution is employed to reduce the complexity. To accelerate the convergence, the OGP re-computes the weights of the basis matrices, where least squares orthogonalization is recursively solved. The per-iteration complexity of the three algorithms linearly increases with the number and dimension of the matrices, indicating that they are scalable to problem size. We theoretically prove that the reconstruction error of each algorithm decays exponentially. The lower bound of the convergence rate or decay factor of the GP and EGP is derived. In addition, we prove the finite convergence of the OGP. We also quantitatively show that how much faster the OGP converges than the GP. It is revealed that the acceleration factor of the OGP over GP/EGP is dominated by the angle between the current iterate and the subspace spanned by the previous iterates. Unlike the non-diagonal decompositions of the GLRAM, the proposed methodology achieves a nonorthogonal but joint diagonal decomposition of multiple matrices, which allows a more parsimonious representation and a higher compression ratio. Experimental results on random data and real-world image databases demonstrate the attractiveness of the greedy framework.

Chapter 5 develops the RLRAMM, i.e., a generalization of the LRAMM of Chapter 4 from the ℓ_2-space to the ℓ_p-space, achieving outliers robustness. This generalization is neither trivial nor straightforward since the resultant optimization problems of the ℓ_p-case is more complicated than the ℓ_2-case. The ℓ_p-GP is designed for the robust learning task. It works in an iterative manner. At each iteration, it finds the best rank-one approximation of the residuals based on ℓ_p-norm minimization. A successive optimization scheme is devised for the ℓ_p-rank-one fitting problem. The case of $p = 1$ is particularly attractive since it only requires computation of weighted medians, leading to a near-linear per-iteration complexity and thus, making the ℓ_1-GP near-scalable to problem size. We theoretically prove that the sum of the ℓ_p-norms of the residuals decays exponentially, where the proof is totally different from that of GP/EGP/OGP in Chapter 4. We reveal that the worst-case bound of the convergence rate or decay factor of depends on the ℓ_p-correlation of the residual and the current iterates. Like the GP, EGP and OGP, the ℓ_p-GP also provides a nonorthogonal but joint diagonal decomposition of multiple matrices, which allows a higher compression ratio than the GLRAM. Experimental results on random data and image databases demonstrate the outlier-robustness of the ℓ_p-GP.

6.2 Future Work

There are some key unsolved issues and various related topics worthy of future research. We summarize several open problems as follows.

1. In-depth convergence analysis of the alternating minimization algorithm in Chapter 2 and the iterative ℓ_p-regression algorithm in Chapter 3. We only show that the sequence of the objective function $\{f_p(\boldsymbol{U}^k, \boldsymbol{V}^k)\}$ of the two algorithms converge to a limit point. Further theoretical work is required to solve the following three open problems on the convergence of the two algorithms.
    i) Does the sequence of the argument variable $\{(\boldsymbol{U}^k, \boldsymbol{V}^k)\}$ converge for the nonsmooth case with $p \leq 1$? It is observed that $\{(\boldsymbol{U}^k, \boldsymbol{V}^k)\}$ converges from numerical simulation results. We need to establish this convergence theoretically.
    ii) For the smooth case where $p > 1$, we can show that the limit point that the two algorithms converge to is a stationary point. How about the nonsmooth case with $p \leq 1$?
    iii) Under what condition, will the limit point be a local or even a global optimum point of the nonconvex optimization problem?
2. Convergence analysis of the ADMM for nonconvex optimization. It is known that the convergence of the ADMM has only been proved for convex optimization [81]. Although the convergence of the ADMM for a certain class of nonconvex and nonsmooth optimization problems has been established very recently in [124],

the corresponding results are not applicable to our problem. The nonconvexity of our problem is not only due to the ℓ_p-norm with $p < 1$ but also induced by the matrix product $\boldsymbol{UV}$. These two reasons make the theoretical proof of the convergence of the ADMM for low-rank approximation problems challenging.

3. Performance guarantee of the low-rank approximation in ℓ_p-space. Suppose that data matrix is a superposition of a low-rank component and a sparse component. Under what assumptions, can the ℓ_p-matrix factorization methods in Chapter 2 exactly separate the two components?

4. Performance guarantee of the matrix completion in ℓ_p-space. Suppose that data matrix is a superposition of a low-rank component and a sparse component. Under what conditions, can the ℓ_p-matrix completion methods in Chapter 3 exactly recover the true matrix in the noiseless case? It is also of importance to derive the bound of the estimation error in the presence of outliers.

5. Tighter bound of the convergence rate of the GP for LRAMM. We have derived the worst-case bound of the convergence rate of the GP, which is the same as the EGP. However, in practice, GP always converges faster than the EGP. It is worth finding the tighter bound in average for the GP.

6. Quantitative analysis of the ℓ_p-correlation, which dominates the convergence rate of the ℓ_p-GP for RLRAMM. It is important to study how the ℓ_p-correlation quantitatively varies as the iteration progresses and derive more accurate bounds of the convergence rate.

7. Parallel and distributed implementation of the proposed algorithms, especially the asynchronous parallel implementation.

Appendix

A.1 Polynomial-Time Algorithm for Univariate ℓ_p-Regression

Given $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^n$, consider the following univariate ℓ_p-regression

$$\min_{\alpha \in \mathbb{R}} \|\boldsymbol{b} - \alpha \boldsymbol{a}\|_p^p$$

which is nonconvex and nonsmooth and has multiple local minima when $0 < p < 1$. However, it is not NP-hard and its global minimum can be found with a complexity of $\mathcal{O}(n^2)$. Since the zero element of $\boldsymbol{a}$ has no effect on the minimizer, without loss of generality, we can assume that $\boldsymbol{a}$ does not contain zero elements when finding the minimizer. The univariate ℓ_p-regression is reformulated as

$$\min_{\alpha} f_p(\alpha) := \sum_{i=1}^{n} |a_i|^p \left| \alpha - \frac{b_i}{a_i} \right|^p .$$

Define the sequence $d_i = b_i / a_i$ and assume that $\{d_i\}_{i=1}^n$ has been sorted in ascending order. The function $f_p(\alpha)$ is piecewise with the break points being $\{d_i\}_{i=1}^n$. The domain of $f_p(\alpha)$, is divided into $n+1$ intervals, i.e., $(-\infty, d_1]$, $(d_1, d_2], \cdots,$ $(d_{n-1}, d_n]$, and (d_n, ∞). In each interval, the sign of $\{\alpha - d_i\}_{i=1}^n$ is determined and the absolute operator $|\cdot|$ can be removed. For example, consider $\alpha \in (-\infty, d_1]$, then for all $i = 1, \cdots, n$, we have $|\alpha - d_i|^p = (d_i - \alpha)^p$. Note that $(\alpha - d_i)^p$ or $(d_i - \alpha)^p$ is a *concave* function due to $p < 1$. It turns out that $f_p(\alpha)$ is concave because the non-negative combination preserves concavity. Therefore, the piecewise function $f_p(\alpha)$ is concave in each interval, although it is not concave in the whole domain $\mathbb{R}$. Recall that a concave function attains its the minimum at the boundary points. It is evident that the minimizer of $f_p(\alpha)$ belongs to $\{d_1, \cdots, d_n\}$ since $f_p(-\infty) = f_p(\infty) = \infty$. Then the global minimizer is selected from the n candidates $\min\limits_{1 \le i \le n} f_p(d_i)$. It is obvious that the complexity is $\mathcal{O}(n^2)$.

Figure 1 gives an example of $f_p(\alpha)$ with $p = 0.5$, where the 4-dimensional data are randomly generated as $\boldsymbol{a} = [0.2939, -0.7873, 0.8884, -1.1471]^{\mathrm{T}}$ and $\boldsymbol{b} = [0.4889, 1.0347, 0.7269, -0.3034]^{T}$. To find the global minimum of $f_p(\alpha)$, we only need to compute the function value at the sorted $\boldsymbol{d} = [-1.3143, 0.2645, 0.8182, 1.6636]^{\mathrm{T}}$. We see that $f_p(\alpha)$ attains its minimum at $d_2 = 0.2645$.

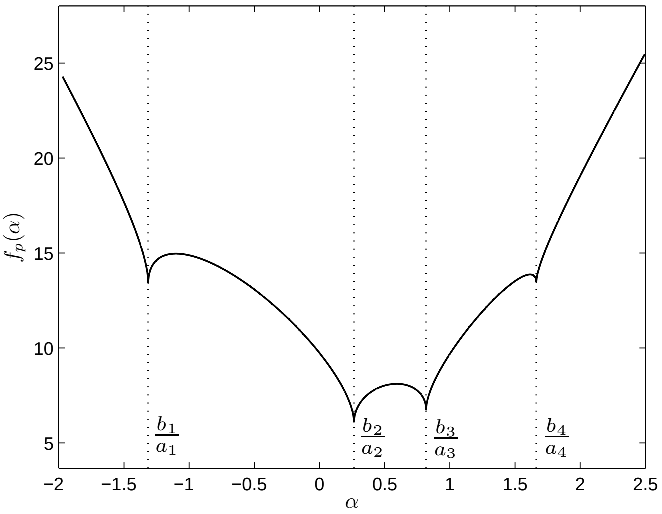


Figure 1: Objective function $\|\boldsymbol{b} - \alpha\boldsymbol{a}\|_p^p$ versus α with $p = 0.5$.

List of Acronyms

2D	two-dimensional
2DPCA	two-dimensional principal component analysis
ADMM	alternating direction method of multipliers
ALM	augmented Lagrange method
AM	alternating minimization
AP	alternating projection
CRB	Cramér-Rao bound
DOA	direction-of-arrival
EGP	economic greedy pursuit
ESPRIT	estimating signal parameters via rotational invariance techniques
EVD	eigenvalue decomposition
FLOM	fractional lower-order moments
FPC	fixed point continuation
GLRAM	generalized low-rank approximations of matrices
GGD	generalized Gaussian distribution
GMM	Gaussian mixture model
GP	greedy pursuit
HOSVD	higher-order singular value decomposition
IHT	iterative hard thresholding
i.i.d.	independent and identically distributed
IRLS	iteratively reweighted least squares

IR-SVD	iteratively reweighted singular value decomposition
IST	iterative soft thresholding
ℓ_p-**GP**	ℓ_p-greedy pursuit
LAD	least absolute deviation
LRAMM	low-rank approximation of multiple matrices
LS	least squares
MAE	mean absolute error
MAPE	mean absolute prediction error
ML	maximum likelihood
MP	matching pursuit
MUSIC	multiple signal classification
NMSE	normalized mean square error
NRE	normalized reconstruction error
OGP	orthogonal greedy pursuit
OMP	orthogonal matching pursuit
PARSuMi	proximal alternating robust subspace minimization
p.d.f.	probability density function
PCA	principal component analysis
RLRAMM	robust low-rank approximation of multiple matrices
RMSE	root mean square error
RMSPE	root mean square prediction error
RPCA	robust principal component analysis
SDP	semi-definite programming

SNR	signal-to-noise ratio
SVD	singular value decomposition
SVP	singular value projection
SVT	singular value thresholding
ULA	uniform linear array
VBMFL_1	variational Bayesian matrix factorization based on L_1-norm
WNNM	weighted nuclear norm minimization

List of Symbols

The following list contains the most important symbols in the dissertation in alphabetical order. The remaining symbols are introduced where they are used.

$(\cdot)^*$	complex conjugate
$(\cdot)^\star$	optimal value or optimal point
$(\cdot)^{-1}$	inverse
$(\cdot)^\dagger$	pseudo-inverse of a matrix
$(\cdot)^\mathrm{H}$	Hermitian transpose
$(\cdot)^\mathrm{T}$	transpose
$\widehat{(\cdot)}$	estimate of a quantity
$\odot$	Hadamard (element-wise) product
$\lvert\cdot\rvert$	cardinality of a set or absolute value of a real number or modulus of a complex number
$\lVert\cdot\rVert$	Euclidean norm of a vector or entry-wise ℓ_2-norm of a matrix
$\lVert\cdot\rVert_2$	spectral norm of a matrix
$\lVert\cdot\rVert_p$	ℓ_p-norm of a vector or entry-wise ℓ_p-norm of a matrix
$\lVert\cdot\rVert_\mathrm{F}$	Frobenius norm
$\lVert\cdot\rVert_*$	nuclear norm
$\lVert\cdot\rVert_{S_p}$	Schatten p-norm
$\langle\cdot,\cdot\rangle$	inner product
$\mathbf{0}$	zero vector or zero matrix
$\mathbf{1}$	vector of all ones
$\mathbb{C}$	set of complex numbers
$\mathbb{R}$	set of real numbers
$\mathbb{R}_+$	set of non-negative real numbers
$\mathbb{Z}$	set of integers
$\mathbb{Z}_+$	set of non-negative integers
$\boldsymbol{I}$	identity matrix
$\imath$	imaginary unit
$\mathrm{E}\{\cdot\}$	expectation
$\mathrm{EV}_{\max}(\cdot)$	unit-norm eigenvector corresponding to maximum eigenvalue
$\mathrm{Im}(\cdot)$	imaginary part of a complex scalar, vector, or matrix
$\lambda_{\max}(\cdot)$	maximum eigenvalue of a matrix

$\mathrm{LSV}_{\max}(\cdot)$	unit-norm left singular vector corresponding to maximum singular value
$\mathrm{range}(\cdot)$	range space
$\mathrm{rank}(\cdot)$	rank of a matrix
$\mathrm{Re}(\cdot)$	real part of a complex scalar, vector, or matrix
$\mathrm{RSV}_{\max}(\cdot)$	unit-norm right singular vector corresponding to maximum singular value
$\sigma_l(\cdot)$	the lth singular value of a matrix
$\sigma_{\max}(\cdot)$	maximum singular value of a matrix
$\mathrm{tr}(\cdot)$	trace of a matrix
$\mathrm{TSVD}_r(\cdot)$	truncated rank-r singular value decomposition of a matrix
$\mathrm{vec}(\cdot)$	concatenating all columns of a matrix into a vector

References

[1] B. Yu, “Let us own data science,” Institute of Mathematical Statistics (IMS) Presidental Address, *IMS Bulletin Online,* ASC-IMS Joint Conference, Sydney, July, 2014. Available: `http://bulletin.imstat.org/2014/10/ims-presidential-address-let-us-own-data-science/`

[2] J. Fan, “Features of big data and sparsest solution in high confidence set,” in *Past, Present and Future of Statistical Science,* X. Lin, *et al* Eds., Chapman & Hall, New York, pp. 507–523, 2014.

[3] V. Larsson and C. Olsson, “Convex low rank approximation,” *Int. J. Comput. Vis.,* vol. 120, no. 2, pp. 194–214, Nov. 2016.

[4] V. Chandrasekaran, S. Sanghavi, P. A. Parrilo, and A. S. Willsky, “Rank-sparsity incoherence for matrix decomposition,” *SIAM J. Opt.,* vol. 21, no. 2, pp. 572–596, Jun. 2011.

[5] E. J. Candès, “Mathematics of sparsity (and a few other things),” in *Proc. International Congress of Mathematicians,* Seoul, South Korea, 2014.

[6] `https://www.netflixprize.com/`

[7] Y. Zhou, D. Wilkinson, R. Schreiber, and R. Pan, “Large-scale parallel collaborative filtering for the Netflix prize,” in *Proc. 4th Int Conf. Algo. Inf. Manag.,* LNCS 5034, pp. 337–348, Springer, 2008.

[8] S. Funk, “Netflix update: Try this at home,” `http://sifter.org/~simon/journal/20061211.html`

[9] E. J. Candès and Y. Plan, “Matrix completion with noise,” *Proc. IEEE,* vol. 98, no. 6, pp. 925–936, Jun. 2010.

[10] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry, “Using collaborative filtering to weave an information tapestry,” *Communications of the ACM,* vol. 35, no. 12, pp. 61–70, Dec. 1992.

[11] E. J. Candès and T. Tao, “The power of convex relaxation: Near-optimal matrix completion,” *IEEE Trans. Inf. Theory,* vol. 56, no. 5, pp. 2053–2080, May 2010.

[12] M. A. Davenport and J. Romberg, “An overview of low-rank matrix recovery from incomplete observations,” *IEEE J. Sel. Top. Signal Process.,* vol. 10, no. 4, pp. 608–622, Jun. 2016.

[13] F. De La Torre and M. J. Black, “A framework for robust subspace learning,” *Int. J. Comput. Vis.,* vol. 54, no. 1–3, pp. 117–142, 2003.

[14] G. Liu, Z. Lin, S. Yan, J. Sun, Y. Yu, and Y. Ma, “Robust recovery of subspace structures by low-rank representation,” *IEEE Trans. Pattern Anal. Mach. Intell.,*, vol. 35, no. 1, pp. 171–184, Jan. 2013.

[15] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester, “Image inpainting,” in *Proc. ACM SIGGRAPH,* New Orleans, USA, pp. 414–424, Jul. 2000.

[16] J. Ye, “Generalized low rank approximations of matrices,” *Mach. Learn.,* vol. 61, no. 1–3, pp. 167–191, 2005.

[17] T. Hastie, R. Mazumder, J. Lee, and Reza Zadeh, “Matrix completion and low-rank SVD via fast alternating least squares,” *J. Mach. Learn. Res.,* vol. 16, pp. 3367–3402, 2015.

[18] A. Eriksson and A. van den Hengel, “Efficient computation of robust weighted low-rank matrix approximations using the L_1 norm,” *IEEE Trans. Pattern Anal. Mach. Intell.,* vol. 34, no. 9, pp. 1681–1690, Sep. 2012.

[19] Y. Peng, A. Ganesh, J. Wright, W. Xu, and Yi Ma, “RASL: Robust Alignment by sparse and low-rank decomposition for linearly correlated images,” *IEEE Trans. Pattern Anal. Mach. Intell.,* vol. 34, no. 11, pp. 2233–2246, Nov. 2012.

[20] J. Chen, J. Liu, and J. Ye, “Learning incoherent sparse and low-rank patterns from multiple tasks,” *ACM Trans. Knowledge Discovery from Data,* vol. 5, issue 4, Article 22, Feb. 2012.

[21] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriengman, “Eigenfaces vs. Fisherfaces: Recognition using class specific linear projection,” *IEEE Trans. Pattern Anal. Mach. Intell.,* vol. 19, no. 7, pp. 711–720, Jul. 1997.

[22] J. P. Haldar and Z.-P. Liang, “Low-rank approximations for dynamic imaging,” in *Proc. IEEE Int. Symp. Biomedical Imaging: From Nano to Macro,* Chicago, USA, Apr. 2011.

[23] H. Kim, G. H. Golub, H. Park, “Missing value estimation for DNA microarray gene expression data: Local least squares imputation,” *Bioinformatics,* vol. 21, no. 2, pp. 187–198, 2005.

[24] L. Liu, D. M. Hawkins, S. Ghosh, and S. S. Young, “Robust singular value decomposition analysis of microarray data,” *Proc. Natl. Acad. Sci.,* vol. 100, no. 23, pp. 13167–13172, Nov. 2003.

[25] J. Huang, F. Nie, H. Huang, Y. Lei, and C. Ding, “Social trust prediction using rank-k matrix recovery,” in *Proc. 23th Int. Joint Conf. Artificial Intelligence (IJCAI),* pp. 2647–2653, Beijing, China, Aug. 2013.

[26] C. Eckart and G. Young, “The approximation of one matrix by another of lower rank,” *Psychometrika*, vol. 1, pp. 211–218, 1936.

[27] K. Pearson, “On lines and planes of closest fit to systems of points in space,” *Philosophical Mag.*, vol. 2, no. 11, pp. 559–572, 1901.

[28] R. Vidal, Y. Ma, and S. Sastry, *Generalized Principal Component Analysis.* New York: Springer-Verlag, 2016.

[29] A. M. Zoubir, V. Koivunen, Y. Chakhchoukh, and M. Muma, “Robust estimation in signal processing: A tutorial-style treatment of fundamental concepts,” *IEEE Signal Process. Mag.,* vol. 29, no. 4, pp. 61–80, Jul. 2012.

[30] F. Nie, J. Yuan, and H. Huang, “Optimal mean robust principal component analysis,” in *Proc. 31st Int. Conf. Machine Learning (ICML),* pp. 1062–1070, Beijing, China, Jun. 2014.

[31] W.-J. Zeng and H. C. So, “Outlier-robust matrix completion via ℓ_p-minimization,” *IEEE Trans. Signal Process.,* vol. 66, no. 5, pp. 1125–1140, Mar. 2018.

[32] E. J. Candès, X. Li, Y. Ma, and J. Wright, “Robust principal component analysis?,” *J. ACM,* vol. 58, no. 3, Article 11, May 2011.

[33] J. Wright, A. Ganesh, K. Min, and Yi Ma, “Compressive prinicpal component pursuit,” *Information and Inference,* vol. 2, no. 1, pp. 32–68, Jun. 2013.

[34] L. Li, W. Huang, I. Gu, and Q. Tian, “Statistical modeling of complex backgrounds for foreground object detection,” *IEEE Trans. Image Process.,* vol. 13, no. 11, pp. 1459C1472, Nov. 2004.

[35] G. H. Golub and C. F. Van Loan, *Matrix Computations.* Baltimore, MD: Johns Hopkins Univ. Press, 1996.

[36] J. Yang, D. Zhang, A. Frangi, and J.-Y. Yang “Two-dimensional PCA: A new approach to appearance-based face representation and recognition,” *IEEE Trans. Pattern Anal. Mach. Intell.,* vol. 26, no. 1, pp. 131–137, Jan. 2004.

[37] Z. Lin, M. Chen, L. Wu, and Y. Ma, “The augmented Lagrange multiplier method for exact recovery of corrupted low-rank matrices,” Tech. Rep. UILU-ENG-09-2215, UIUC, Nov. 2009.

[38] P. Netrapalli, U. Niranjan, S. Sanghavi, A. Anandkumar, and P. Jain, “Provable non-convex robust PCA,” in *Proc. Adv. Neural Inf. Proc. Syst. (NIPS),* pp. 1107–1115, 2014.

[39] E. J. Candès and M. B. Wakin, “An introduction to compressed sampling,” *IEEE Signal Process. Mag.,* vol. 25, no.2, pp. 21–30, Mar. 2008.

[40] R. Tibshirani, “Regression shrinkage and selection via the lasso,” *J. R. Stat. Soc. Ser. B (Methodol.),* vol. 58, no. 1, pp. 267–288, 1996.

[41] S. J. Wright, R. D. Nowak, and M. Figueiredo, “Sparse reconstruction by separable approximation,” *IEEE Trans. Signal Process.,* vol. 57, no. 7, pp. 2479–2493, Jul. 2009.

[42] B. Recht, M. Fazel, and P. A. Parrilo, “Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization,” *SIAM Rev.,* vol. 52, no. 3, pp. 471–501, 2010.

[43] M. Fazel, H. Hindi, and S. Boyd, "A rank minimization heuristic with application to minimum order system approximation," in *Proc. Amer. Control Conf.,* Arlington, Virginia, pp. 4734–4739, Jun. 2001.

[44] Z. Liu and L. Vandenberghe, "Interior-point method for nuclear norm approximation with application to system identification," *SIAM J. Matrix Anal. App.,* vol. 31, no. 3, pp. 1235–1256, Aug. 2009.

[45] J.-F. Cai, E. J. Candès, and Z. Shen, "A singular value thresholding algorithm for matrix completion," *SIAM J. Opt.,* vol. 20, no. 4, pp. 1956–1982, 2010.

[46] S. Ma, D. Goldfarb, and L. Chen, "Fixed point and Bregman iterative methods for matrix rank minimization," *Math. Program. Ser. A,* vol. 128, no. 1, pp. 321–353, 2011.

[47] K. C. Toh and S. W. Yun, "An accelerated proximal gradient algorithm for nuclear norm regularized least squares problems," *Pacific J. Opt.,* vol. 6, pp. 615–640, 2010.

[48] F. Nie, H. Wang, H. Huang, and C. Ding, "Joint Schatten p-norm and ℓ_p-norm robust matrix completion for missing value recovery," *Knowl. Inf. Syst.*, vol. 42, no. 3, pp. 525–544, Mar. 2015.

[49] F. Nie, H. Huang, and C. Ding, "Low-rank matrix recovery via efficient Schatten p-norm minimization," in *Proc. 26th AAAI Conf. Artificial Intelligence,* pp. 655–661, Toronto, Canada, Jul. 2012.

[50] F. Nie, H. Wang, X. Cai, H. Huang, and C. Ding, "Robust matrix completion via joint Schatten p-norm and ℓ_p-norm minimization," in *Proc. 12th Int. Conf. Data Mining (ICDM),* pp. 566–574, Brussels, Belgium, Dec. 2012.

[51] K. Mohan and M. Fazel, "Iterative reweighted algorithms for matrix rank minimization," *J. Machine Learn. Res.,* vol. 13, pp. 3441–3473, Nov. 2012.

[52] G. Marjanovic and V. Solo, "On l_q optimization and matrix completion," *IEEE Trans. Signal Process.,* vol. 60, no. 11, pp. 5714–5724, Nov. 2012.

[53] S. Gu, Q. Xie, D. Meng, W. Zuo, X. Feng, and L. Zhang, "Weighted nuclear norm minimization and its applications to low level vision," *Int. J. Comput. Vision,* vol. 121, no. 2, pp. 183–208, Jan. 2017.

[54] T. Blumensath and M. E. Davies, "Iterative hard thresholding for compressed sensing," *App. Comput. Harmonic Anal.,* vol. 27, no. 3, pp. 265–274, 2009.

[55] P. Jain, R. Meka, and I. S. Dhillon, "Guaranteed rank minimization via singular value projection," in *Adv. Neural Inf. Process. Syst. (NIPS),* pp. 937–945, 2010.

[56] 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.

[57] X. Jiang, Z. Zhong, X. Liu, and H. C. So, "Robust matrix completion via alternating projection," *IEEE Sig. Process. Lett.,* vol. 24, no. 5, pp. 579–583, May 2017.

[58] D. G. Luenberger and Y. Ye, *Linear and Nonlinear Programming (4th Edition).* NY: Springer, 2016.

[59] L. N. Trefethen and D. Bau III. *Numerical Linear Algebra.* SIAM, Philadelphia, 1997.

[60] R. H. Keshavan, A. Montanari, and S. Oh, "Matrix completion from a few entries," *IEEE Trans. Inf. Theory,* vol. 56, no. 6, pp. 2980–2998, Jun. 2010.

[61] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer,* vol. 42, no. 8, pp. 30–37, 2009.

[62] R. Sun and Z.-Q. Luo, "Guaranteed matrix completion via nonconvex factorization," in *Proc. 56th IEEE Annual Sym. Found. Comput. Science (FOCS),* pp. 270–289, Berkeley, CA, Oct. 2015.

[63] M. Hardt, "Understanding alternating minimization for matrix completion," in *Proc. 55th IEEE Annual Sym. Found. Comput. Science (FOCS),* pp. 651–660, Philadelphia, PA, Oct. 2014.

[64] P. Jain, P. Netrapalli, and S. Sanghavi, "Low-rank matrix completion using alternating minimization," in *Proc. 45th Annual ACM Sym. Theory Comput.,* pp. 665–674, Palo Alto, CA, Jun. 2013.

[65] J. Huang, F. Nie, and H. Huang, "Robust discrete matrix completion," in *Proc. 26th AAAI Conf. Artificial Intelligence,* pp. 424–430, Bellevue, USA, Jul. 2013.

[66] H. Zhang, Z. Lin, and C. Zhang, "Completing low-rank matrices with corrupted samples from few coefficients in general basis," *IEEE Inf. Theory,* vol. 62, no. 8, pp. 4748–4768, Aug. 2016.

[67] Y.-X. Wang, C. M. Lee, L.-F. Cheong, and K.-C. Toh, "Practical matrix completion and corruption recovery using proximal alternating robust subspace minimization," *Int. J. Comput. Vision,* vol. 111, no. 3, pp. 315–344, Feb. 2015.

[68] M. Oskarsson, K. Batstone, and K. Åström, "Trust no one: Low rank matrix factorization using hierarchical RANSAC," in *Proc. IEEE Conf. Comput. Vision Patt. Recog. (CVPR),* Las Vegas, NV, USA, pp. 5820–5825, Jun. 2016.

[69] Q. Zhao, D. Meng, Z. Xu, W. Zuo, and Y. Yan, "L_1-Norm low-rank matrix factorization by variational Bayesian method," *IEEE Trans. Neural Netw. Learn. Syst.,* vol. 26, no. 4, pp. 825–839, Apr. 2015.

[70] C. Ding and J. Ye, "2-Dimensional singular value decomposition for 2D maps and images," in *Proc. SIAM Int. Conf. Data Mining (SDM),* pp. 32–43, CA, USA, Apr. 2005.

[71] J. Ye, R. Janardan, and Q. Li, "Two-dimensional linear discriminant analysis," In *Proc. Adv. Neural Inf. Proc. Syst. (NIPS),* pp. 1569–1576, 2004.

[72] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification.* Wiley-Interscience, New York, 2001.

[73] T. G. Kolda and B. W. Bader, “Tensor decompositions and applications,” *SIAM Rev.,* vol. 51, no. 3, pp. 455–500, 2009.

[74] D. Goldfarb and Z. Qin, “Robust low-rank tensor recovery: models and algorithms” *SIAM J. Matrix Anal. & Appl.,* vol. 35, no. 1, pp. 225–253, Jan. 2014.

[75] J Liu, P. Musialski, P. Wonka, and J. Ye, “Tensor completion for estimating missing values in visual data,” *IEEE Trans. Pattern Anal. Mach. Intell.,* vol. 35, no. 1, pp. 208–220, Jan. 2013.

[76] D. P. O’Leary, “Robust regression computation using iteratively reweighted least squares,” *SIAM. J. Matrix Anal. & Appl.,* vol. 11, no. 3, pp. 466–480, 1990.

[77] R. A. Maronna, R. D. Martin, and V. J. Yohai, *Robust Statistics: Theory and Methods.* Wiley Series in Probability and Statistics. John Wiley & Sons, New York, USA, 2006.

[78] P. J. Huber, *Robust Statistics,* New York: Wiley, 2005.

[79] D. Bertsekas, *Nonlinear Programming.* Athena Scientific, Belmont, 1999.

[80] M. R. Hestenes and E. Stiefel, “Methods of conjugate gradients for solving linear systems,” *J. Res. Nat. Bur. Stand.,* vol. 49, no. 6, pp. 409–436, Dec. 1952.

[81] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, “Distributed optimization and statistical learning via the alternating direction method of multipliers,” *Foundations and Trends in Machine Learning,* vol. 3, no. 1, pp. 1–122, 2011.

[82] J. Eckstein, “Augmented Lagrangian and alternating direction methods for convex optimization: A tutorial and some illustrative computational results,” RUTCOR, Res. Rep. RRR 32–2012, Dec. 2012.

[83] R. Glowinski, “On alternating direction methods of multipliers: A historical perspective,” in: *Modeling, Simulation and Optimization for Science and Technology,* pp. 59–82. Springer, 2014.

[84] C. Chen, B. He, Y. Ye, and X. Yuan, “The direct extension of ADMM for multiblock convex minimization problems is not necessarily convergent,” *Math. Program. Ser. A,* vol. 155, no. 1, pp. 57–79, Jan. 2016.

[85] P. L. Combettes and J.-C. Pesquet, “Proximal splitting methods in signal processing,” in: *Fixed-Point Algorithms for Inverse Problems in Science and Engineering.* NY: Springer, 2011.

[86] D. Donoho, “De-noising by soft-thresholding,” *IEEE Trans. Inf. Theory,* vol. 41, no. 3, pp. 613–627, 1995.

[87] S. J. Wright, R. D. Nowak, and M. Figueiredo, “Sparse reconstruction by separable approximation,” *IEEE Trans. Signal Process.,* vol. 57, no. 7, pp. 2479–2493, Jul. 2009.

[88] A. Maleki, L. Anitori, Z. Yang, and R. G. Baraniuk, "Asymptotic analysis of complex LASSO via complex approximate message passing (CAMP)," *IEEE Trans. Inf. Theory,* vol. 59, no. 7, pp. 4290–4308, Jul. 2013.

[89] H. Xu, C. Caramanis, S. Sanghavi, "Robust PCA via outlier pursuit," *IEEE Trans. Inf. Theory,* vol. 58, no. 5, pp. 3047–3064, May 2012.

[90] G. Mateos and G. B. Giannakis, "Robust PCA as bilinear decomposition with outlier sparsity regularization," *IEEE Trans. Signal Process.,* vol. 60, no. 10, pp. 5176–5190, Oct. 2012.

[91] J. Krim and M. Viberg, "Two decades of array signal processing research: The parametric approach," *IEEE Signal Process. Mag.,* vol. 13, no. 3, pp. 67–94, Jul. 1996.

[92] W.-J. Zeng and X.-L. Li, "High-resolution multiple wideband and nonstationary source localization with unknown number of sources," *IEEE Trans. Signal Process.,* vol. 58, no. 6, pp. 3125–3136, Jun. 2010.

[93] H. C. So, *Source Localization: Algorithms and Analysis,* in: S. A. Zekavat and M.Buehrer (Ed.), Handbook of Position Location: Theory, Practice and Advances, Wiley-IEEE Press, 2011.

[94] R. O. Schmidt, "Multiple emitter location and signal parameter estimation," *IEEE Trans. Antennas Propagat.,* vol. AP-34, no. 3, pp. 276–280, Mar. 1986.

[95] M. Viberg and B. Ottersten, "Sensor array processing based on subspace fitting," *IEEE Trans. Signal Process.,* vol. 39, no. 5, pp. 1110–1121, May. 1991.

[96] B. Porat and B. Friedlander, "Analysis of the asymptotic relative efficiency of the MUSIC algorithm," *IEEE Trans. Acoust. Speech Signal Process.,* vol. 36, no. 4, pp. 532–544, Apri. 1988.

[97] P. Stoica and A. Nehorai, "MUSIC, maximum likelihood, and Cramer-Rao bound," *IEEE Trans. Acoust. Speech Signal Process.,* vol. 37, no. 5, pp. 720–741, May 1989.

[98] H. Abeida and J.-P. Delmas, "Efficiency of subspace-based DOA estimators," *Signal Process.,* vol. 87, no. 9, pp. 2075–2084, Sep. 2007.

[99] K. L. Blackard, T. S. Rappaport, and C. W. Bostian, "Measurements and models of radio frequency impulsive noise for indoor wireless communications," *IEEE J. Sele. Areas Commun.,* vol. 11, no. 7, pp. 991–1001, Sep. 1993.

[100] P. L. Brockett, M. Hinich, and G. R. Wilson, "Nonlinear and non-Gaussian ocean noise," *J. Acoust. Soc. Amer.,* vol. 82, no. 4, pp. 1386–1394, Oct. 1987.

[101] D. Middleton, "Non-Gaussian noise models in signal processing for telecommunications: New methods and results for class A and class B noise models," *IEEE Trans. Inform. Theory,* vol. 45, no. 4, pp. 1129–1149, May 1999.

[102] P. Tsakalides and C. L. Nikias, "The robust covariation-based MUSIC (ROC-MUSIC) algorithm for bearing estimation in impulsive noise environments," *IEEE Trans. Signal Process.,* vol. 44, no. 7, pp. 1623–1633, Jul. 1996.

[103] T.-H. Liu and J. M. Mendel, "A subspace-based direction finding algorithm using fractional lower order statistics," *IEEE Trans. Signal Process.,* vol. 49, no. 8, pp. 1605–1613, Aug. 2001.

[104] S. Visuri, H. Oja, and V. Koivunen, "Subspace-based direction-of-arrival estimation using nonparametric statistics," *IEEE Trans. Signal Process.,* vol. 49, no. 9, pp. 2060–2073, Sep. 2001.

[105] R. J. Kozick and B. M. Sadler, "Maximum-likelihood array processing in non-Gaussian noise with Gaussian mixtures," *IEEE Trans. Signal Process.,* vol. 48, no. 12, pp. 3520–3535, Dec. 2000.

[106] A. Swami and B. M. Sadler, "On some detection and estimation problems in heavy-tailed noise," *Signal Process.,* vol. 82, no. 12, pp. 1829–1846, Dec. 2002.

[107] C.-H. Lim, S. C.-M. See, A. M. Zoubir, and B. P. Ng, "Robust adaptive trimming for high-resolution direction finding," *IEEE Signal Process. Lett.,* vol. 16, no. 7, pp. 580–583, Jul. 2009.

[108] M. Muma, Y. Cheng, F. Roemer, M. Haardt, and A. M. Zoubir, "Robust source number enumeration for r-dimensional arrays in case of brief sensor failures," in *Proc. Int. Conf. Acoust., Speech and Signal Process. (ICASSP),* 2012, pp. 3709–3712.

[109] M. Salibian-Barrera, S. Van Aelst, and G. Willems, "Principal components analysis based on multivariate MM Estimators with fast and robust bootstrap," *J. Am. Stat. Assoc.,* vol. 101, no. 475, pp. 1198–1211, Sep. 2006.

[110] W.-J. Zeng, H. C So, and L. Huang, "ℓ_p-MUSIC: Robust direction-of-arrival estimator for impulsive noise environments," *IEEE Trans. Signal Process.,* vol. 61, no. 17, pp. 4296–4308, Sep. 2013.

[111] R. A. Maronna and V. J. Yohai, "Robust low-rank approximation of data matrices with elementwise contamination," *Technometrics,* vol. 50, no. 3, pp. 295–304, Aug. 2008.

[112] Z. Lu, Y. Chakhchoukh, and A. M. Zoubir, "Source number estimation in impulsive noise environments using bootstrap techniques and robust statistics," in *Proc. Int. Conf. Acoust., Speech and Signal Process. (ICASSP),* 2011, pp. 2712–2715.

[113] B. D. Rao and K. V. S. Hari, "Performance analysis of root-MUSIC," *IEEE Trans. Acoust., Speech, Signal Process.,* vol. 37, no. 3, pp. 1939–1949, Dec. 1989.

[114] R. Roy and T. Kailath, "ESPRIT–estimation of signal parameters via rotational invariance techniques," *IEEE Trans. Acoust., Speech, Signal Process.,* vol. 37, pp. 984–995, Jul. 1989.

[115] M. Novey, T. Adali, and A. Roy, "A complex generalized Gaussian distribution–characterization, generation, and estimation," *IEEE Trans. Signal Process.,* vol. 58, no. 3, pp. 1427–1433, Mar. 2010.

[116] M. McCoy. *A geometric analysis of convex demixing,* Ph.D. thesis, California Institute of Technology, Pasadena, USA, Mar. 2014.

[117] Video database for testing change detection algorithms. Online Available: `http://www.changedetection.net/`

[118] P.-S. Laplace, "Sur quelques points du système du monde," *Mémoires de l'Académie des Sciences de Paris,* 1789. Reprinted in *Evres Complétes,* vol. 11, pp. 475–558, Gauthier-Villars, Paris.

[119] P. Bloomfield and W. L. Steiger. *Least Absolute Deviations: Theory, Applications, and Algorithms.* Birkhäuser, Boston, 1983.

[120] S. Portnoy and R. Koenker, "The Gaussian hare and the Laplacian tortoise: Computability of squared-error versus absolute-error estimators," *Statistical Science,* vol. 12, no. 4, pp. 279–300, 1997.

[121] P. Tseng, "Convergence of a block coordinate descent method for nondifferentiable minimization," *J. Optim. Theory Appl.*, vol. 109, no. 3, pp. 475–494, Jun. 2001.

[122] U. M. Ascher and C. Greif. *A First Course in Numerical Methods.* SIAM, Philadelphia, PA, 2011.

[123] W. Zuo, D. Meng, L. Zhang, X. Feng, and D. Zhang, "A generalized iterated shrinkage algorithm for non-convex sparse coding," in *Proc. IEEE Int. Conf. Compu. Vis. (ICCV),* Sydney, Australia, pp. 217–224, Dec. 2013.

[124] Y. Wang, W. Yin, and J. Zeng, "Global convergence of ADMM in nonconvex nonsmooth optimization," Available at `https://arxiv.org/abs/1511.06324`.

[125] G. Claeskens, "Statistical model choice," *Annual Review of Statistics and Its Application,* vol. 3, pp. 233–256, Jun. 2016.

[126] C.-G. Li and R. Vidal, "A structured sparse plus structured low-rank framework for subspace clustering and completion," *IEEE Trans. Signal Process.,* vol. 64, no. 24, pp. 6557–6570, Dec. 2016.

[127] R. Chartrand and W. Yin, "Iteratively reweighted algorithms for compressive sensing," in *Proc. Int. Conf. Acoust. Speech Sig. Process. (ICASSP),* pp. 3869-3872, Las Vegas, NV, USA, March–April, 2008.

[128] `http://www.cs.rochester.edu/~jliu/publications.html`

[129] `https://grouplens.org/datasets/movielens/`

[130] T. H. Gormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms.* Cambridge, MA: MIT Press, 2001.

[131] J. A. Tropp, “Greed is good: Algorithmic results for sparse approximation,” *IEEE Trans. Inf. Theory,* vol. 50, no. 10, pp. 2231–2242, Oct. 2004.

[132] T. Zhang, “On the consistency of feature selection using greedy least squares regression,” *J. Mach. Learn. Res.,* vol. 10, pp. 555–568, 2009.

[133] S. Mallat and Z. Zhang, “Matching pursuits with time-frequency dictionaries,” *IEEE Trans. Signal Process.,* vol. 41, no. 12, pp. 3397–3415, Dec. 1993.

[134] J. A. Tropp and A. C. Gilbert, “Signal recovery from random measurements via orthogonal matching pursuit,” *IEEE Trans. Inf. Theory,* vol. 53, no. 12, pp. 4655–4666, Dec. 2007.

[135] W.-J. Zeng, H. C. So, and X. Jiang, “Outlier-robust greedy pursuit algorithms in ℓ_p-space for sparse approximation,” *IEEE Trans. Signal Process.,* vol. 64, no. 1, pp. 60–75, Jan. 2016.

[136] Z. Wang, M.-J. Lai, Z. Lu, W. Fan, H. Davulcu, and J. Ye, “Rank-one matrix pursuit for matrix completion,” In *Proc. 31st Int. Conf. Mach. Learn. (ICML),* pp. 91–99, Beijing, China, Jun. 2014.

[137] K. Lee and Y. Bresler, “ADMiRA: Atomic decomposition for minimum rank approximation,” *IEEE Trans. Inf. Theory,* vol. 56, no. 9, pp. 4402–4416, Sep. 2010.

[138] Q. Yao and J. T. Kwok, “Greedy learning of generalized low-rank models,” in *Proc. 25th Int. Joint Conf. Artifi. Intell. (IJCAI),* pp. 2294–2300, New York, USA, Jul. 2016.

[139] R. Khanna, E. Elenberg, A. G. Dimakis, and S. Negahban, “On approximation guarantees for greedy low rank optimization,” arXiv:1703.02721v1 [stat.ML], Mar, 2017. Available at: `https://arxiv.org/abs/1703.02721v1`

[140] D. Bernstein, *Matrix Mathematics: Theory, Facts, and Formulas.* Princeton Univ. Press, New Jersey, USA, 2009.

[141] F. Samaria and A. Harter, “Parameterisation of a stochastic model for human face identification,” in *Proc. Second IEEE Workshop App. Comput. Vis.,* 1994.

[142] `http://www.anefian.com/research/face_reco.htm`.

[143] `http://yann.lecun.com/exdb/mnist/`

[144] S. A. Nene, S. K. Nayar, and H. Murase, “Columbia object image library (COIL-20),” Columbia Univ., Technical Report CUCS-005-96, 1996.

[145] Y. Nestrov, *Introductory Lectures on Convex Optimization–A Basic Course.* Springer, New York, 2004.

Curriculum Vitae

Name: Wenjun Zeng

Date of birth: 08.12.1982

Place of birth: Jiangxi, China

Family status: married

Education

Since 08/2017	Technische Universität Darmstadt, Germany Electrical Engineering and Information Technology
08/2006 - 07/2008	Tsinghua University, Beijing, China Control Science and Engineering (M.Sc.) Master thesis: Joint diagonalization based high resolution spectral estimation
09/2000 - 07/2004	Harbin Engineering University, Harbin, China Electronic Engineering (B.Sc.) with highest honor Bachelor thesis: Deconvolution techniques in signal processing
09/1997 - 07/2000	Ningdu Middle School, Jiangxi, China High school degree (Abitur)

Work experience

Since 04/2012	Senior Research Associate at Department of Electronic Engineering City University of Hong Kong, Hong Kong
05/2009 - 10/2011	Faculty Member at Department of Communications Engineering Xiamen University, Xiamen, China
08/2006 - 04/2009	Research Assistant at Department of Automation Tsinghua University, Beijing, China.

Erklärung laut §9 der Promotionsordnung

Ich versichere hiermit, dass ich die vorliegende Dissertation allein und nur unter Verwendung der angegebenen Literatur verfasst habe. Die Arbeit hat bisher noch nicht zu Prüfungszwecken gedient.

Darmstadt, 11. April 2018,