Institute of Information and Communication Technologies,
Electronics and Applied Mathematics

Low-rank matrix and tensor completion using graph-based regularization

Shuyu Dong

Thesis submitted in partial fulfillment of the requirements for the degree of
Docteur en sciences de l'ingénieur

Dissertation committee:

Pierre-Antoine Absil (Université catholique de Louvain, advisor)
Kyle A. Gallivan (Florida State University, advisor)
Jean-Charles Delvenne (Université catholique de Louvain)
Lieven De Lathauwer (Katholieke Universiteit Leuven)
François Glineur (Université catholique de Louvain)
Wen Huang (Xiamen University)
Roland Keunings (Université catholique de Louvain, chair)
Yurii Nesterov (Université catholique de Louvain)

May 2021

Abstract

Matrix and tensor completion are a type of data recovery problem that arises in many different real-world applications related to the inference or acquisition of data. In this thesis, we investigate matrix and tensor completion problems in the aspects of models, algorithms and regularization.

With respect to models and algorithms, we tackle these problems in the framework of low-rank matrix optimization with a least-squares model. These rank-constrained models are known not only for their low computational complexity but also the capability of extracting the most important information in the data that has an intrinsically low rank. By exploiting the manifold structures in the rank-constrained models, we develop Riemannian algorithms and analyze the convergence properties of these algorithms. In particular, we focus on algorithms designed via Riemannian preconditioning and provide novel results that explain their efficiency on the quotient manifold of fixed-rank matrices.

Regularization, in general terms, refers to a mechanism that imposes some additional structure in the problem variable to prevent over-fitting of the model. We investigate the usage of a certain graph-based regularizer in the matrix and tensor completion problems. The penalty function in the graph-based regularization is an extension of the Frobenius norm in the problem variable, in the sense that the entries in the variable are not penalized with uniform weights but with respect to the intensities of some pairwise relations encoded in a graph Laplacian matrix. The graph Laplacian-based penalty function promotes matrix or tensor solutions such that the pairwise similarities among its entries are conform to the underlying graph structure. Besides traditional graph models, we propose a graph learning algorithm for learning a desired graph structure from data.

Acknowledgments

I would like to express my gratitude to everyone who guided and helped me in my research and life during my PhD studies. I enjoyed this journey and find myself very lucky to have been surrounded by people with whom I worked.

I am most grateful to have had Pierre-Antoine Absil and Kyle A. Gallivan as my advisors. I wish to thank Pierre-Antoine and Kyle for their continuous guidance, for the countless discussions that shaped the work in this thesis, and for their support and encouragement in my research.

I would like to thank François Glineur, Jean-Charles Delvenne and Yurii Nesterov for serving on my doctoral committee; the meetings and discussions with them, and all the feedback that I received were important for my thesis and have been invaluable in improving my work. I warmly thank Lieven De Lathauwer, Wen Huang and Roland Keunings for accepting to be part of my thesis jury and for reviewing the thesis.

I am grateful to Pascal Frossard for his guidance through my master thesis and Dorina Thanou for co-supervising my master thesis; their work has been an important and inspiring part in the early stage of my PhD studies. I wish to thank Gabriel Peyré and Xin Liu for hosting my research visits in ENS Paris and AMSS-CAS Beijing, respectively, in different times. These visits have greatly broadened my scientific views. I would also like to thank Laurent Jacques for various interesting discussions in different times.

I am thankful to Bin Gao, for having given me insightful advice during our collaborations and valuable suggestions on one of my thesis chapters, to Yu Guan for our fruitful collaborations and discussions, and to Nicolas Boumal, Bart Vandereycken and Bamdev Mishra, whose works helped introducing me to the main research topic of my PhD studies. I also thank the researchers who gave me great advice or whose company has made my studies enjoyable: Estelle Massart, Emilie Renard, Pierre-Yves Gousenburge, Adrien Taylor, Andersen Man Shun Ang, Loïc Van Hoorebeeck, Guillaume Olikier, Geovani Nunes Grapiglia, Vladimir Gusev, Thanh Son Nguyen and Leonardo G. Gomez. Many thanks go to other researchers in the ICTEAM institute.

I am also grateful to the amazing staff members of ICTEAM for their superb support: thanks to Marie-Christine, Etienne, Pascale, Nathalie, François, Ludovic and Astrid. Merci à toutes et à tous pour leur support tout au long de ma thèse.

I am grateful for the support and love from my friends through all good or bad times. Special thanks to Ming Yu, Chen Liu, Patrick B., Zheming Wang, Lei Hu, Qian Gao, Shivangi S., Yi Huang, Zechuan Chen, Dimitri Kornblum, Kai Feng, Na Y., and Yu Chen.

I would like to thank my mother and father, who give me unconditional love and support throughout my life. This thesis is dedicated to them.

I gratefully acknowledge the ARC and the FNRS for funding my research and travels during the thesis.

Contents

Notation

Variables, vectors and matrices

$\mathbf{1}$	A (column) vector of all ones (of size given by context)
I_p, I	Identity matrix (of size $p \times p$ or given by context)
A^{T}, v^{T}	Transpose of a matrix A and a vector v
A_{ij}	Entry of a matrix A on the row i and column j, (i,j)-th entry of A
δ_{ij}	Kronecker delta, $\delta_{ij} = I_{ij}$
$A_{i,:}$	i-th row of a matrix A
$A_{:,j}$	j-th column of a matrix A
$\mathrm{diag}(A)$	Vector composed of diagonal entries of a matrix A
$\mathrm{Diag}(d_1, \ldots, d_n)$	Diagonal matrix, the (i,i)-th entry is d_i
$\mathrm{Diag}(d)$	Diagonal matrix, the (i,i)-th entry is d_i
$\|v\|_1$, $\|v\|_2$	ℓ_1 and ℓ_2 norms of the vector v
$\|A_{i,:}\|_2$	ℓ_2 norm of the i-th row of the matrix A

Matrix, tensor properties and operations

$\mathrm{vec}(A)$	Vectorization of a matrix A
$\mathrm{vec}_{\mathbb{S}}(A)$	Half-vectorization of a symmetric matrix A
$\mathrm{tr}(A)$	Trace of a matrix A
$\mathrm{rank}(A)$	Rank of a matrix A
$\sigma_{\min}(A)$	Minimal non-zero singular value of a matrix A
$\sigma_i(A)$	i-th singular value of A
$\lambda_i(A)$	i-th eigenvalue of A
$A \succeq 0$, $A \succ 0$	Positive semi-definite matrix, positive definite matrix
$A^{\dagger}$	Moore–Penrose pseudoinverse of a matrix A
$U_{\perp}$	Orthonormal matrix that generates $\mathrm{span}(U)^{\perp}$
$A \star B$	Hadamard product of two matrices A and B
$A \otimes B$	Kronecker product of two matrices A and B
$A \odot B$	Khatri-Rao product of two matrices A and B
$u_1 \circ \cdots \circ u_k$	Outer product of k vectors $u_1, \ldots, u_k$
$\mathcal{T}$	k^{th}-order tensor (multidimensional array), $k \geq 3$

$\mathrm{rank}_{\mathrm{CP}}(\mathcal{T})$	Canonical polyadic rank of a tensor
$\mathrm{rank}_{\mathrm{tc}}(\mathcal{T})$	Tucker rank (multilinear rank) of a tensor
$\mathcal{T}_{(i)}$	Mode-i matricization of the k^{th}-order tensor $\mathcal{T}$, $1 \leq i \leq k$
$\|A\|_{\mathrm{F}}$, $\|\mathcal{T}\|_{\mathrm{F}}$	Frobenius norm of a matrix A, tensor $\mathcal{T}$

Sets and manifolds

$\mathbb{R}^n$	Set of n-dimensional real vectors
$\mathbb{R}^{m\times n}$	Set of $m \times n$ real matrices
$\mathrm{GL}\,(k)$	Set of $k \times k$ invertible matrices
$\mathbb{R}_*^{m\times k}$	Set of $m \times k$ real matrices with full column rank ($k \leq m$)
$\mathrm{St}(m,p)$	Stiefel manifold $\{U \in \mathbb{R}^{m\times p} : U^{\mathrm{T}}U = I_p\}$
$\mathrm{span}(U)$	Subspace spanned by the columns of U
$\mathbb{R}^{m\times k} \times \mathbb{R}^{n\times k}$	Product space of $\mathbb{R}^{m\times k}$ and $\mathbb{R}^{n\times k}$
$\mathcal{M}_{\leq k}$	Set of $m \times n$ real matrices with rank equal or smaller than $k \leq \min(m,n)$
$\mathcal{M}_k$	Set of $m \times n$ real matrices with rank $k \leq \min(m,n)$
$\mathcal{S}^{n-1}$	Unit-sphere in $\mathbb{R}^n$
$\mathcal{S}_+^{n-1}$	Nonnegative orthant of the unit-sphere in $\mathbb{R}^n$

Manifolds and operators

$\mathcal{E}$	Vector space
$\langle u, v\rangle$	Inner product between the vectors u, v
$\mathcal{F}^{\perp} \subset \mathcal{E}$	Orthogonal complement of a subspace $\mathcal{F}$ of $\mathcal{E}$
$\mathcal{E}_1 \times \mathcal{E}_2$	Product space
$\mathcal{M}$	Manifold
$\mathcal{M}_1 \times \mathcal{M}_2$	Product manifold
$x \sim y$	Equivalence relation
$\mathcal{M}/\sim$	Quotient space
$\mathcal{V}_{\bar{x}}$, $\mathcal{H}_{\bar{x}}$	Vertical and horizontal spaces at x to a quotient manifold
$T_x\mathcal{M}$	Tangent space at x to the manifold $\mathcal{M}$
$\mathrm{P}_{T_x\mathcal{M}}(\cdot)$	Orthogonal projection onto the tangent space $T_x\mathcal{M}$
$g_x(\cdot,\cdot)$	Riemannian metric at x
$\|\xi\|_x$	Norm of the tangent vector ξ at x
$\mathrm{grad} f(x)$	Riemannian gradient of f at x

Acronyms

SVD	Singular value decomposition
GL	graph learning
LRMC, LRTC	Low-rank matrix completion, low-rank tensor completion
GRMC	Graph-regularized matrix completion

Flops	Floating point operations
GD, RGD	Gradient descent, Riemannian gradient descent
CG, RCG	Conjugate gradient, Riemannian conjugate gradient
AltMin	Alternating minimization
ADMM	Alternating directions multiplier method
PD	Polyadic decomposition
CP, CPD	Canonical Polyadic, Canonical Polyadic decomposition

Chapter 1

Introduction

Low-rank matrices are often encountered in data science and other applications. Many problems in machine learning, signal processing and big data analysis can be modeled as a problem of learning a low-rank matrix from data. Low-rank models constitute a powerful data representation paradigm that is well-known for its computational and storage efficiency. In addition to matrix optimization problems, low-rank models have also been extended to problems with higher-order multidimensional arrays, known as tensors.

In this thesis, we focus on low-rank models and algorithms for matrix and tensor completion. Matrix completion refers to the task of estimating, or inferring the missing entries of a matrix, given a number of its known entries. Motivated by the application in recommendation systems and especially popularized by the Netflix Prize [BL+07] for movie recommendations, the matrix completion problem has seen quick developments and numerous applications in the last decade. Tensor completion, as the term suggests, is a problem similar to matrix completion but deals with the imputation of missing entries in a *tensor*. Here, the tensor refers to a multidimensional array, which is a widely used data structure for storing data that has multiple (more than two) *modes* such as sounds, images, user profiles, time stamps, and spatial information.

The task of filling the missing entries of a data matrix, without any prior knowledge, is an ill-posed problem. Among the many early attempts for matrix completion, the optimization approach of [KBV09], based on the idea of latent factor models, consists in optimizing a low-rank matrix factorization to fit the sparsely distributed known entries of the data matrix. This approach not only provided impressive completion results through various experimental demonstrations, but also achieved those results with low computational cost. Therefore, low-rank matrix optimization and matrix completion problems have gained much attention from research communities in machine learning and numerical optimization. Our work in this thesis addresses the matrix and tensor completion problems through the two aspects: models and algorithms.

For the modeling of matrix and tensor matrix problems, we take an in-

terest in regularization methods in the framework of low-rank models. In the context of machine learning, regularization refers to a process that imposes an additional structure on the problem variable of a machine learning model to prevent over-fitting of the model. We study so-called graph-based regularization, which exploits relational information in the form of graphs: while it is common approach to use the Frobenius norm (of the matrix variables) as a penalty function of the matrix completion problem, graph-based regularization exploits structural information of the data in a more adaptive manner. More precisely, the index sets of the data matrix are modeled as graphs, in which the graph edges indicate the connections or degrees of similarity between the columns or rows of the matrix. This graph-based modeling can be understood in the example of the movie recommendations application: given a matrix of ratings (e.g., in a scale of 1 to 5) of a number of movies made by a number of users, the matrix row and column index sets are the sets of the users and movies. In addition to the known ratings, the interrelations between the users and/or the movies are important side information; for example, the users (respectively, the movies) form different *clusters*, or closely-connected subgroups, according to similarities in some of their features, interests or profiles. Consequently, given a graph that conforms to the clustering patterns of the set of these users (respectively, movies), the pairwise similarities can be encoded in the adjacency matrix of this graph, and constitute valuable side information in addition to the low-rank matrix completion model. In a larger sense, graphs provide a natural tool for many machine learning problems other than matrix and tensor completion. In particular, advances in graph spectral analysis and graph signal processing [Chu97, Spi12, LK02, SNF$^+$13] lay the ground for extending the notion of smoothness of real-valued functions on a continuous domain to functions defined on the vertices of a graph. A prominent application of these theories is the use of graph Laplacian matrices in machine learning models [ZGL03, Liu06, TSF14].

When one tries to exploit graph information in a certain task like graph-based regularization, important questions to ask include how to get graph information and how to use the graph information in the specific task. To address these questions, we investigate a graph learning method. Also, we explore several ways of building graphs from a given data matrix and then focus on the usage of graph-based regularization for matrix completion. In a previous work about what we refer to as *graph-regularized* matrix completion, Rao et al. [RYRD15] investigated the effect of a graph Laplacian-based regularization for matrix completion and proposed an alternating minimization algorithm for solving the underlying problem. We approach this problem from a Riemannian optimization perspective, which is the second main topic of this thesis.

With respect to optimization, the most important feature of the low-rank models is that the matrix or tensor candidate for the completion task is represented by a tuple of low-rank matrices through matrix or tensor factorization. From the perspective of numerical optimization on manifolds, such a problem with (low) rank constraints can be seen as an unconstrained optimization

problem on a certain manifold or variety of low-rank matrices. Therefore, this thesis takes a view of optimization on matrix manifolds [AMS08]. Following developments in algorithms for low-rank matrix optimization [KO09, CCS10, JMD10, MBS11, MAS12, Van13, AAM14, MMBS14], we focus on a type of algorithms designed through Riemannian preconditioning [MS16] for solving the problems underlying the aforementioned matrix and tensor completion models. The particularity of these algorithms is that their search directions on the matrix manifold are determined with respect to a certain variable metric that is designed according to the cost function of the low-rank model; notice that, in the case of the least-squares loss function, the so-designed metric is a Riemannian metric on the manifold. To further understand the efficiency of these algorithms that we observe through various experiments, we investigate the properties of the Riemannian gradient descent algorithm, designed through Riemannian preconditioning, on the manifold of $m \times n$ matrices with a fixed-rank; the algorithmic framework is built upon the quotient manifold interpretation of the set of fixed-rank matrices. We find a new perspective for the convergence analysis of this algorithm and provide results about the local convergence behavior of the algorithm, which explain the time efficiency and some nice properties of this algorithm that traditional (Euclidean) gradient descent or alternating minimization does not possess for low-rank matrix optimization problems.

The contributions of this thesis come from a collection of research works made by the author and his coauthors, which are listed in the organization of the thesis; see Section 1.1. In the next section, we list some applications of matrix and tensor completion, other than the aforementioned application in movie recommendations.

Applications

Apart from the application to recommendation systems that we will demonstrate in detail, matrix and tensor completion have numerous other applications. In computer vision, matrix and tensor completion are used to restore visual data that has missing values due to limitations of the acquisition or transmission of data or simply technical issues; one can find examples in the following areas.

Biomedical imaging. The recovery of images from limited and noisy measurements is essentially a matrix completion problem and is an important problem in areas of biomedical imaging such as microscopy, magnetic resonance imaging (MRI) and computed tomography; see [HLJ18] for details.

Structure from motion. Structure from motion (SFM) [YJHB14] refers to a photogrammetric range imaging technique for estimating three-dimensional structures from two-dimensional image sequences that may be coupled with

local motion signals; it is a computer vision analogue of the phenomenon in biological vision by which humans (or other living creatures) can recover three-dimensional structure from the projected two-dimensional (retinal) motion field of a three-dimensional scene or moving object. In the SFM problem, it is common to require operations on matrices with missing entries because of occlusion or tracking failures [CS04].

In social computing [SGCH19], matrix and tensor completion are used to estimate missing information concerning the individuals and social interactions within societies; one can find examples in the following areas.

Link prediction. Link prediction aims at predicting missing connections in a network or the probabilities of future node connections. This problem can be formulated as a matrix completion problem based on the existing or observed connections in the network. More recently, link prediction is also modeled as tensor completion problems, following the extension of static networks to networks with dynamic connections over time or networks that contain multiple types of interactions [SGCH19].

Urban computing. Urban computing deals with the human behaviors and mobilities with the help of computing technology to benefit living quality in urban areas. "Traffic and mobility" information is one of the most significant topics in this field. The traffic occupancy data recorded by PeMS (http://pems.dot.ca.gov/) and the New York Taxi trip record data (`https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page`) are notable examples. Based on this large amount of data, the processing of traffic information and mobility recommendation are directly related to matrix and tensor problems.

1.1 Organization of the thesis

Chapter 2: Preliminaries

This chapter presents the background and fundamental definitions and properties about (i) manifolds, Riemannian geometry and optimization with low-rank matrices, (ii) graphs and (iii) tensor computations.

Chapter 3: Graph learning and applications

This chapter presents an optimization approach to the problem of inferring a graph structure from data. The graph learning problem consists of finding a graph adjacency or Laplacian matrix by minimizing a smoothness-promoting function in the graph-related matrix variable. We investigate one of the graph learning formulations and propose a novel graph learning algorithm. The proposed algorithm is applied to graph-dependent machine learning tasks such as

semi-supervised learning and graph-based regularization for matrix completion. This chapter is partly based on the paper [DAG18].

Chapter 4: Graph-regularized matrix completion

Low-rank matrix completion is the problem of recovering the missing entries of a data matrix by using the assumption that the true matrix admits a good low-rank approximation. Much attention has been given recently to exploiting correlations between the column/row entities to improve the matrix completion quality. In this chapter, we propose preconditioned gradient descent algorithms for solving the low-rank matrix completion problem with graph Laplacian-based regularizers and establish global convergence results of these algorithms.

Experiments on synthetic data show that our approach achieves significant speedup compared to an existing method based on alternating minimization. Experimental results on real world data also show that our methods provide low-rank solutions of similar quality in comparable or less time than the state-of-the-art method. This chapter is based on the conference and journal papers [DAG19, DAG20].

Chapter 5: Optimization with fixed-rank matrices

In this chapter, we study a Riemannian gradient descent algorithm on a set of fixed-rank matrices. The set $\mathcal{M}_k$ of $m \times n$ real matrices with a fixed rank k is identified with a quotient manifold of a two-factor product space via matrix factorization. The underlying Riemannian gradient is defined with respect to a metric that induces preconditioning for the matrix factorization problem. We provide novel results about the properties of Riemannian gradient descent on the quotient manifold $\mathcal{M}_k$.

For a class of fixed-rank matrix optimization problems with a quadratic cost function, we analyze the local convergence behavior of the Riemannian gradient descent algorithm. The geometric and convergence results show that the Riemannian algorithm not only has fast convergence behavior but also enjoys desirable invariance properties, in contrast to the Euclidean gradient descent on the matrix product space. Numerical experiments on matrix completion with fixed-rank matrices are provided to demonstrate these properties. This chapter is based on a joint project with P.-A. Absil and K. A. Gallivan. An initial extended abstract of this work was accepted for a talk at MTNS 2020; the conference was postponed and recently canceled.

Chapter 6: Graph-regularized tensor completion

This chapter contains an introduction to the tensor completion problem with an emphasis on the tensor decomposition approach using the polyadic decomposition. The rest of this chapter is based on the paper [GDAG20]. As second author of this paper, my contributions are mainly in the development of the

algorithms and the integration of graph construction methods, in an extended framework of the GRMC project (Chapter 4).

In this work, we start from a low-rank tensor decomposition model for tensor completion and consider an additional graph Laplacian-based function in the model variables. The graph-based function exploits correlations between the rows of the low-rank matrix factors of the polyadic decomposition model. We propose an alternating minimization algorithm for solving the problem that we refer to as graph-regularized tensor completion (GRreg-TC). Furthermore, based on the Kurdyka-Łojasiewicz property, we show that the sequence generated by the proposed algorithm globally converges to a critical point of the objective function. Besides, an alternating direction method of multipliers algorithm is also developed for GRreg-TC. We show the efficiency of the proposed algorithms through extensive numerical experiments on synthetic and real data.

Chapter 7: Riemannian preconditioned algorithms for tensor completion

We propose Riemannian preconditioned algorithms for low-rank tensor completion via the polyadic decomposition of a tensor. These algorithms exploit a non-Euclidean metric on the product space of the factor matrices of the low-rank tensor in polyadic decomposition form. This new metric is designed using an approximation of the diagonal blocks of the Hessian of the tensor completion cost function, thus has a preconditioning effect on these algorithms. We prove that the proposed Riemannian gradient descent algorithm globally converges to a stationary point of the tensor completion problem, with convergence rate estimates using the Łojasiewicz property. Numerical results on both synthetic and real-world data suggest that the proposed algorithms present advantages over existing algorithms in terms of time efficiency under flexible choices of the tensor rank. This chapter is based on the paper [DGGG21].

List of publications

[DAG18] S. Dong, P.-A. Absil, and K. A. Gallivan. Graph learning for regularized low rank matrix completion. *Proceedings of the 23rd International Symposium on Mathematical Theory of Networks and Systems (MTNS)*, pages 460–467, 2018.

[DAG19] S. Dong, P.-A. Absil, and K. A. Gallivan. Preconditioned Conjugate Gradient Algorithms for Graph Regularized Matrix Completion. In *European Symposium on Artificial Neural Networks (ESANN)*, pages 239–244, 2019.

[DAG20] S. Dong, P.-A. Absil, and K. A. Gallivan. Riemannian gradient descent methods for graph-regularized matrix completion. *Linear Algebra and*

its Applications, 2020. doi: https://doi.org/10.1016/j.laa.2020.06.010.

[GDAG20] Y. Guan, S. Dong, P.-A. Absil, and F. Glineur. Alternating minimization algorithms for graph-regularized tensor completion. *arXiv preprint arXiv:2008.12876*, pages 1–30, 2020. URL `https://arxiv.org/pdf/2008.12876.pdf`.

[DGGG21] S. Dong, B. Gao, Y. Guan, and F. Glineur. New Riemannian preconditioned algorithms for tensor completion via polyadic decomposition. *arXiv preprint arXiv:2101.11108*, pages 1–24, 2021. URL `https://arxiv.org/pdf/2101.11108.pdf`.

Chapter 2

Preliminaries

This chapter is composed of the main background knowledge of the topics in this thesis: manifolds and Riemannian geometry, low-rank matrices, graphs and tensors (also known as multidimensional arrays). The first two sections are about the fundamental definitions and properties in the field of numerical optimization on manifolds. In the third and the last sections, brief expositions on the background of graphs and tensors are given.

2.1 Manifolds, Riemannian geometry and optimization

In this section, we give an overview on the essential elements about manifolds and Riemannian optimization that are used in this thesis. The contents of this section relies on [AMS08, Chapters 3–4]. We also refer to [Car92, Lee97, Lee03, O'n83] for the related topics.

A manifold can be informally understood as a set $\mathcal{M}$ that is a collection of patches, or charts, that can be locally identified as open sets of a d-dimensional vector space $\mathbb{R}^d$.

Definition 2.1.1 (Chart)**.** *A chart of a set $\mathcal{M}$ is a pair $(\mathcal{U}, \varphi)$ where $\mathcal{U} \subset \mathcal{M}$ and φ is a bijection between $\mathcal{U}$ and an open set of $\mathbb{R}^d$.*

In the above definition, the chart is d-dimensional and for a *point* $x \in \mathcal{M}$, the elements of $\varphi(x) := (y_1, \ldots, y_d)$ are called the coordinates of x in the chart (U, φ).

Definition 2.1.2 (Atlas)**.** *A collection of charts $(\mathcal{U}_\alpha, \varphi_\alpha)$ is called an ($\mathcal{C}^\infty$) atlas if it satisfies:*

1. $\cup_\alpha \mathcal{U}_\alpha = \mathcal{M}$,
2. *For any $\mathcal{U}_\alpha, \mathcal{U}_\beta$ with $\mathcal{U}_\alpha \cap \mathcal{U}_\beta \neq \emptyset$, the sets $\varphi_\alpha(\mathcal{U}_\alpha \cap \mathcal{U}_\beta)$ and $\varphi_\beta(\mathcal{U}_\alpha \cap \mathcal{U}_\beta)$*

are open sets in $\mathbb{R}^d$ *and the change of coordinates*

$$\varphi_\beta \circ \varphi_\alpha^{-1} : \mathbb{R}^d \mapsto \mathbb{R}^d$$

is ($\mathcal{C}^\infty$) *smooth. In this case,* $\mathcal{U}_\alpha$ *and* $\mathcal{U}_\beta$ *are said to be compatible charts.*

Two atlases $\mathcal{A}_1$ and $\mathcal{A}_2$ are equivalent if $\mathcal{A}_1 \cup \mathcal{A}_2$ is an atlas; this implies that (via Definition 2.1.2) any chart $(\mathcal{U}, \varphi)$ in $\mathcal{A}_1$ is compatible with all charts of $\mathcal{A}_2$ and vice versa. Given an atlas $\mathcal{A}$, let $\mathcal{A}^+$ be *the* set of all charts such that $\mathcal{A} \cup \mathcal{A}^+$ is also an atlas. One can see that $\mathcal{A}^+$ is also an atlas, called *maximal atlas* generated by $\mathcal{A}$. For any atlas $\mathcal{A}$, there exists an unique maximal atlas $\mathcal{A}^+$ generated by $\mathcal{A}$.

We are interested in maximal atlas $\mathcal{A}^+$ such that the topology induced on $\mathcal{M}$ by $\mathcal{A}^+$ is Hausdorff and second countable. In brief, a topological space is *Hausdorff* if two distinct points in this space have disjoint neighborhoods. In view of iterative algorithms, the Hausdorff property rules out the possibility of having a convergent sequence that has distinct limit points. A topological space is *second-countable* if there is a countable collection $\mathcal{B}$ of open sets that generates all open sets of the space by union. Based on such a maximal atlas, we define a manifold in the following classical way.

Definition 2.1.3 (Manifold)**.** *A manifold* $(\mathcal{M}, \mathcal{A}^+)$ *is a set* $\mathcal{M}$ *endowed with a maximal atlas* $\mathcal{A}^+$ *of* M*.*

The manifold $(\mathcal{M}, \mathcal{A}^+)$ is ($\mathcal{C}^\infty$) smooth if the atlas $\mathcal{A}^+$ is ($\mathcal{C}^\infty$) smooth.

Example 2.1.4 (The set $\mathbb{R}^{m\times n}$)**.** *The set of real-valued matrices* $\mathbb{R}^{m\times n}$*, endowed with the chart*

$$\mathbb{R}^{m\times n} \mapsto \mathbb{R}^{mn} : X \mapsto \mathrm{vec}(X),$$

where the matrix entry X_{ij} *is mapped to the* ℓ*-th element of vec*(X) *for* $i \in \{1, \ldots, m\}$*,* $j \in \{1, \ldots, n\}$ *and* $\ell = i + n(j-1)$*, is a manifold, since the chart (via vectorization) forms an atlas.*

Note that the chart through vectorization $\mathrm{vec}(\cdot)$ is unwieldy, as it destroys the matrix structure of its argument; in particular, $\mathrm{vec}(AB)$ cannot be written as a simple expression of $\mathrm{vec}(A)$ and $\mathrm{vec}(B)$. In this thesis, it is the properties (*e.g.* closeness and linearity) of the vector space $\mathbb{R}^{mn}$ that matter; the matrix structure and all matrix computations are preserved.

Example 2.1.5 (The unit 2-sphere)**.** *The unit sphere* $\mathcal{S}^2 = \{w \in \mathbb{R}^3 : \|w\|_2 = 1\}$*, endowed with the two charts (stereographic projections)*

$$\{w = (x, y, z) \in \mathcal{S}^2 : z = \pm 1\} \mapsto \mathbb{R}^2 : w \mapsto \left(\frac{x}{1 \mp z}, \frac{y}{1 \mp z}\right),$$

is a manifold, since these charts form an atlas.

From now, we use the notation $\mathcal{M}$ for the manifold $(\mathcal{M}, \mathcal{A}^+)$ when the atlas

information underlying the manifold structure is not required.

Definition 2.1.6 (Smooth function)**.** *Let $\mathcal{M}$ be a manifold. A function $f : \mathcal{M} \mapsto \mathbb{R}$ is a smooth function (of class $\mathcal{C}^k$) if, for any $x \in \mathcal{M}$, the function*

$$\hat{f} := f \circ \varphi_x^{-1} : \mathbb{R}^d \mapsto \mathbb{R},$$

where $(\mathcal{U}_x, \varphi_x)$ is a chart on $\mathcal{M}$ containing x, is ($\mathcal{C}^k$) smooth.

The set of real-valued smooth functions on a neighborhood of $x \in \mathcal{M}$ is denoted as $\mathcal{F}_x(\mathcal{M})$.

Definition 2.1.7 (Smooth mapping)**.** *Let $\mathcal{M}_1$ and $\mathcal{M}_2$ be two smooth manifolds. A mapping $F : \mathcal{M}_1 \mapsto \mathcal{M}_2$ is of class $\mathcal{C}^k$ if, for all $p \in \mathcal{M}$, there is a chart $(\mathcal{U}, \varphi)$ of $\mathcal{M}$ and a chart $(\mathcal{V}, \psi)$ of $\mathcal{M}_2$ such that $p \in \mathcal{U}, F(\mathcal{U}) \subset \mathcal{V}$ and*

$$\psi \circ F \circ \psi^{-1} : \varphi(\mathcal{U}) \mapsto \psi(\mathcal{V})$$

is of class $\mathcal{C}^k$. The latter is called the local expression of F in the charts $(\mathcal{U}, \varphi)$ and $(\mathcal{V}, \psi)$.

2.1.1 Embedded submanifold

Intuitively, an *embedded submanifold* refers to a subset of a manifold $(\mathcal{M}, \mathcal{A}^+)$ that *inherits* the differential structure of $\mathcal{M}$, and such an *inheritance* is unique; see details in [AMS08, Proposition 3.3.1]. Formally, a subset $\mathcal{N}$ of a manifold $\mathcal{M}$ is a d-dimensional embedded submanifold of $\mathcal{M}$ if and only if, around each point $x \in \mathcal{N}$, there exists a chart $(\mathcal{U}, \varphi)$ of $\mathcal{M}$ such that

$$\mathcal{N} \cap \mathcal{U} = \{x \in \mathcal{U} : \varphi(x) \in \mathbb{R}^d \times \{0\}\}.$$

From an numerical optimization perspective, embedded submanifolds of Euclidean spaces is a special yet important case of embedded submanifolds; in fact, identifying a set $\mathcal{M}$ as an embedded submanifold (of a given Euclidean space) is a common way of recognizing a manifold. The following proposition shows the identification of manifolds through *submersion* (see [AMS08, §3.2.1]) instead of constructing charts explicitly.

Proposition 2.1.8. *Let $F : \mathcal{M}_1 \mapsto \mathcal{M}_2$ be a smooth mapping between two manifolds of dimension d_1 and d_2, $d_1 > d_2$, and let y be a point of $\mathcal{M}_2$. If the rank of F is equal to d_2 at every point of $F^{-1}(y)$, then $F^{-1}(y)$ is a closed embedded submanifold of $\mathcal{M}_1$, and $dim(F^{-1}(y)) = d_1 - d_2$.*

Example 2.1.9 (The unit-sphere in $\mathbb{R}^n$)**.** *Using the mapping $F : \mathbb{R}^n \mapsto \mathbb{R} : x \mapsto \|x\|_2 - 1$, it follows from Proposition 2.1.8 that the sphere $\mathcal{S}^{n-1} = \{x \in \mathbb{R}^n : \|x\|_2 = 1\}$ is a $(n-1)$-dimensional submanifold of $\mathbb{R}^n$.*

Example 2.1.10 (Fixed-rank matrices)**.** *Given integers m, n and $1 \leq k \leq$*

$\min(m, n)$, *the set of* $m \times n$ *real matrices with a fixed rank* k, *i.e.,*

$$\mathcal{M}_k = \{X \in \mathbb{R}^{m \times n} : \operatorname{rank}(X) = k\},$$

is a submanifold embedded in $\mathbb{R}^{m \times n}$; *see Section 2.2 for more details.*

2.1.2 Quotient manifold

Definition 2.1.11 (Equivalence relation)**.** *Let* $\mathcal{M}$ *be a set. An equivalence relation* $\sim$ *on* $\mathcal{M}$ *is a relation that satisfies the following properties: (i) reflexivity:* $x \sim x$ *for all* $x \in \mathcal{M}$, *(ii) symmetry:* $x \sim y$ *if and only if* $y \sim x$, *for all* $x, y \in \mathcal{M}$, *and (iii) transitivity: if* $x \sim y$ *and* $y \sim z$, *then* $x \sim z$, *for all* $x, y, z \in \mathcal{M}$.

Definition 2.1.12 (Quotient set)**.** *Let* $\mathcal{M}$ *be a set and let* $\sim$ *be an equivalence relation on* $\mathcal{M}$. *For a point* $x \in \mathcal{M}$, *the set of points* $y \in \mathcal{M}$ *such that* $y \sim x$, *denoted as* $[x] = \{y \in \mathcal{M} : y \sim x\}$, *is called the equivalence class, or fiber, of* x. *The quotient set associated to the equivalence relation* $\sim$, *denoted as* $\mathcal{M}/\!\sim$, *is the set of all equivalence classes:* $\mathcal{M}/\!\sim := \{[x] : x \in \mathcal{M}\}$.

The operator $\pi : \mathcal{M} \mapsto \mathcal{M}/\!\sim : x \mapsto [x]$ that maps a point to its equivalence class is referred to as the *natural projection* or *canonical projection*. We also use $\pi(x)$ to denote $[x]$, which can be seen as a point in the quotient set $\mathcal{M}/\!\sim$. The set $\mathcal{M}$ is called the *total space* of the quotient set $\mathcal{M}/\!\sim$.

Let $(\mathcal{M}, \mathcal{A}^+)$ be a manifold and $\sim$ be an equivalence relation on $\mathcal{M}$. The quotient set $\mathcal{M}/\!\sim$ admits at most one manifold structure; In this case, there is a unique maximal atlas $\mathcal{B}^+$ such that $(\mathcal{M}/\!\sim, \mathcal{B}^+)$ is a manifold, which we refer to as the *quotient manifold*.

Example 2.1.13 (Fixed-rank matrices)**.** *The set of* $m \times n$ *real matrices with rank* $1 \leq k \leq \min(m, n)$, *i.e.,*

$$\mathcal{M}_k = \{X \in \mathbb{R}^{m \times n} : \operatorname{rank}(X) = k\},$$

is a quotient submanifold in $\mathbb{R}_*^{m \times k} \times \mathbb{R}_*^{n \times k}$, *where* $\mathbb{R}_*^{m \times k}$ *and* $\mathbb{R}_*^{n \times k}$ *are the sets of full column-rank matrices. In fact,* $\mathcal{M}_k \simeq \mathbb{R}_*^{m \times k} \times \mathbb{R}_*^{n \times k}/\!\sim$, *where the equivalence relation is defined as, for any* $(L, R), (L', R') \in \mathbb{R}_*^{m \times k} \times \mathbb{R}_*^{n \times k}$, $(L, R) \sim (L', R')$ *if and only if* $LR^{\mathrm{T}} = L'R'^{\mathrm{T}}$.

Functions on quotient manifolds. A function f on $\mathcal{M}$ is termed *invariant under* $\sim$ if $f(x) = f(y)$ if $x \sim y$, in which case the function f induces a unique function f on $\mathcal{M}/\!\sim$, called the *projection* of f, such that $f = \tilde{f} \circ \pi$.

2.1.3 Tangent vectors and tangent space

The notion of *tangent vector* is interpreted by the velocity of a smooth *curve* at a point on $\mathcal{M}$. For a manifold $\mathcal{M}$ embedded in a vector space $\mathbb{R}^d$, let x be a

point on $\mathcal{M}$ and $\gamma : \mathbb{R} \mapsto \mathcal{M}$ be a smooth mapping, called a curve, on $\mathcal{M}$ such that $\gamma(0) = x$. Then the derivative of the curve at $t = 0$, defined as

$$\gamma'(0) := \frac{\mathrm{d}}{\mathrm{d}t} c(t)\Big|_{t=0} \tag{2.1}$$

characterizes a tangent vector to the manifold $\mathcal{M}$ at $x \in \mathcal{M}$. Note that the above derivative is well-defined since the curve γ on $\mathcal{M}$ is embedded in the vector space $\mathbb{R}^d$. Thus, the tangent space to $\mathcal{M}$ at a point x on $\mathcal{M}$ is defined as follows,

$$T_x\mathcal{M} := \{\gamma'(0) : \text{ such that } \gamma(0) = x\}, \tag{2.2}$$

where γ is a smooth curve on $\mathcal{M}$. The tangent space characterized in (2.2) is a linear subspace of $\mathbb{R}^d$. In fact, the dimension of $T_x\mathcal{M}$ is the dimension of a chart of $\mathcal{M}$ containing x.

In the general sense, the characterization of tangent vector to a manifold $\mathcal{M}$ can be realized through a real-valued function on $\mathcal{M}$ and smooth curves. Let x be a point on $\mathcal{M}$, let γ be a smooth curve passing through x at $t = 0$, and let $\mathcal{F}_x(\mathcal{M})$ denote the set of smooth real-valued functions (Definition 2.1.6) defined on a neighborhood of x. The tangent vector to a curve on $\mathcal{M}$ passing through x is characterized as follows.

Definition 2.1.14 (Tangent vector)**.** *Let $x \in \mathcal{M}$. The tangent vector to a smooth curve γ at x is the mapping $\dot{\gamma}(0)$ from $\mathcal{F}_x(\mathcal{M})$ to $\mathbb{R}$ defined as follows,*

$$\dot{\gamma}(0)f := \frac{\mathrm{d}}{\mathrm{d}t} f(\gamma(t))\Big|_{t=0}, \text{ for } f \in \mathcal{F}_x(\mathcal{M}).$$

Consequently, the *tangent space* to $\mathcal{M}$ at x, denoted as $T_x\mathcal{M}$ is the set of all tangent vectors to $\mathcal{M}$ at x.

Definition 2.1.15 (Directional derivative)**.** *The directional derivative of a smooth real-valued function f on $\mathcal{M}$ at $x \in \mathcal{M}$ along the direction $\xi \in T_x\mathcal{M}$ is the scalar:*

$$\mathrm{D}f(x)[\xi] := \frac{\mathrm{d}}{\mathrm{d}t} f(\gamma(t))\Big|_{t=0} = (f \circ \gamma)'(0),$$

where $\gamma : \mathbb{R} \mapsto \mathcal{M}$ is any smooth curve such that $\gamma(0) = x$ and $\gamma'(0) = \xi$.

Definition 2.1.16 (Tangent bundle)**.** *Let $\mathcal{M}$ be a smooth manifold. The tangent bundle, denoted as $T\mathcal{M}$, is the disjoint union of all tangent spaces:*

$$T\mathcal{M} = \bigcup_{x \in \mathcal{M}} T_x\mathcal{M}.$$

Definition 2.1.17 (Vector field)**.** *A vector field X is a smooth mapping $\xi : \mathcal{M} \mapsto T\mathcal{M}$ such that $\xi(x) = \xi_x$, where $\xi_x \in T_x\mathcal{M}$.*

Tangent space to a vector space

A vector space $\mathcal{E}$ can be seen as a manifold with a linear structure. Through Definition 2.1.14, a tangent vector ξ to $\mathcal{E}$ at a point $x \in \mathcal{E}$ is a mapping

$$\xi : \mathcal{F}_x(\mathcal{E}) \mapsto \mathbb{R} : f \mapsto \xi f = \frac{\mathrm{d}}{\mathrm{d}t} f(\gamma(t))\bigg|_{t=0},$$

where γ is a curve in $\mathcal{E}$ passing through x at $t = 0$. In other words, the tangent vector is characterized via the image $\xi f = \mathrm{D}f(x)[\gamma'(0)]$, for any $f \in \mathcal{F}_x(\mathcal{M})$. Thus, there is a linear one-to-one correspondence $\gamma'(0) \mapsto \xi$, which identifies tangent vectors to $\mathcal{E}$ at x with velocities of smooth curves passing through x. Therefore, the tangent space to $\mathcal{E}$ is identified with $\mathcal{E}$, *i.e.*,

$$T_x\mathcal{E} \simeq \mathcal{E}. \tag{2.3}$$

Since tangent vectors are local objects, if $\mathcal{E}_*$ is an open submanifold of $\mathcal{E}$, then $T_x\mathcal{E}_* \simeq \mathcal{E}$ for all $x \in \mathcal{E}_*$.

Tangent space to an embedded manifold

Let $\mathcal{M}$ be an embedded submanifold of a vector space $\mathcal{E}$. As shown in the beginning of this subsection, the tangent vectors to $\mathcal{M}$ are defined with (2.1) through smooth curves on $\mathcal{M}$ and the tangent space to $\mathcal{M}$ at a point $x \in \mathcal{M}$ is defined in (2.2). In particular, when a manifold $\mathcal{M}$ is (locally or globally) defined through a level set of a constant-rank function $F : \mathcal{E} \mapsto \mathbb{R}^d$, where $\mathcal{E}$ is a vector space, one has

$$T_x\mathcal{M} = \ker(\mathrm{D}F(x));$$

as shown in [AMS08, §3.5.7]. In other words, a tangent vector to $\mathcal{M}$ at x correspond to a vector ξ that satisfies $\mathrm{D}F(x)[\xi] = 0$.

Example 2.1.18 (Tangent space to the sphere)**.** *Consider the unit sphere $\mathcal{S}^{n-1}$ as the level set $F^{-1}(0)$, with $F : \mathbb{R}^n \mapsto \mathbb{R} : z^{\mathrm{T}}z - 1$. Given $x \in \mathcal{S}^{n-1}$, the directional derivative of F at x along $\xi \in \mathbb{R}^n$ is $\mathrm{D}F(x)[\xi] = \xi^{\mathrm{T}}x + x^{\mathrm{T}}\xi$. Hence, the tangent space to $\mathcal{S}^{n-1}$ at x is the set*

$$\ker(\mathrm{D}F(x)) = \{\xi \in \mathbb{R}^n : x^{\mathrm{T}}\xi = 0\}.$$

Tangent space to a quotient manifold

Unlike the tangent vectors to a manifold embedded in a vector space, the notion of tangent vectors to a quotient manifold is more abstract and requires a *representation* that comes from the tangent vectors of the total space (Section 2.1.2).

Let $\overline{\mathcal{M}}$ be a manifold and $\mathcal{M} = \overline{\mathcal{M}}/\!\sim$ be a quotient manifold with canonical projection π. Any tangent vector to the quotient manifold $\mathcal{M}$ can be represented by a tangent vector in the total space $\overline{\mathcal{M}}$ via the canonical projection π;

this representation is not unique, and in fact, infinitely many representations are valid. More precisely, given $x \in \mathcal{M}$ and $\xi \in T_x\mathcal{M}$, let $\bar{x}$ be an element of the equivalence class $\pi^{-1}(x)$. Then, any element $\bar{\xi} \in T_{\bar{x}}\overline{\mathcal{M}}$ that satisfies

$$\mathrm{D}\pi(\bar{x})[\bar{\xi}] = \xi$$

can be considered as a representation of ξ. Indeed, for any smooth function $f : \mathcal{M} \mapsto \mathbb{R}$ and the function $\bar{f} := f \circ \pi : \overline{\mathcal{M}} \mapsto \mathbb{R}$, one has the following identification,

$$\mathrm{D}\bar{f}(\bar{x})[\bar{\xi}] = \mathrm{D}f\left(\pi(\bar{x})\right)\left[\mathrm{D}\pi(\bar{x})[\bar{\xi}]\right] = \mathrm{D}f(X)[\xi].$$

Since π is surjective, the kernel of $\mathrm{D}\pi(\bar{x})$ is non-trivial, thus, the matrix representation of ξ in $T_{\bar{x}}\overline{\mathcal{M}}$ is not unique. Nevertheless, one can find a unique representation of ξ in a subspace of $T_{\bar{x}}\overline{\mathcal{M}}$. This is realized by decomposing the tangent space $T_{\bar{x}}\overline{\mathcal{M}}$ into two complementary subspaces, such that $T_{\bar{x}}\overline{\mathcal{M}} = \mathcal{V}_{\bar{x}} \oplus \mathcal{H}_{\bar{x}}$, where $\mathcal{V}_{\bar{x}}$ is the tangent space at $\bar{x}$ of the equivalence class $[\bar{x}]$, i.e.,

$$\mathcal{V}_{\bar{x}} := T_{\bar{x}}(\pi^{-1}(x)),$$

which is referred to as the *vertical space*; the complement of $\mathcal{V}_{\bar{x}}$, denoted as $\mathcal{H}_{\bar{x}}$, is called the *horizontal space*. It is clear that a tangent vector $\bar{\xi} \in \mathcal{V}_{\bar{x}}$ satisfies $\mathrm{D}\pi(\bar{x})[\bar{\xi}] = 0$. Consequently, for any $x \in \mathcal{M}$ and $\xi \in T_x\mathcal{M}$, there is a unique representation $\bar{\xi} \in \mathcal{H}_{\bar{x}} \subset T_{\bar{x}}\overline{\mathcal{M}}$ of ξ such that $\mathrm{D}\pi(\bar{x})[\bar{\xi}] = \xi$. The tangent vector $\bar{\xi} \in \mathcal{H}_{\bar{x}}$ is called the horizontal lift of ξ.

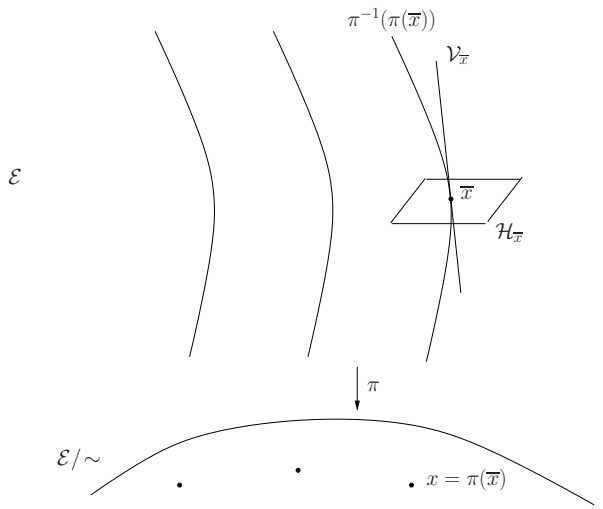


Figure 2.1: Tangent space to a quotient manifold. Figure courtesy of [AMS08].

Figure 2.1 gives an illustration of the tangent space to the total space $\overline{\mathcal{M}}$ and a horizontal lift for the representation of a tangent vector to the quotient

manifold $\mathcal{M}$ at a point $x = [\bar{x}]$.

2.1.4 Riemannian geometry

A manifold can be locally seen as a vector space, thus it can be endowed with a metric to form a metric space, in a similar way as for a vector space. The main difference in defining a metric on $\mathcal{M}$ than a vector space lies in the fact that the tangent space to $\mathcal{M}$ at a point x varies according to the position of x on the manifold.

Definition 2.1.19 (Inner product)**.** *Let $\mathcal{M}$ be a smooth manifold and x be a point in $\mathcal{M}$. An inner product on $T_x\mathcal{M}$, denoted as $\langle \cdot, \cdot \rangle_x$, is a bilinear form satisfying the following properties, for all $\xi, \eta \in T_x\mathcal{M}$,*

1. *bilinearity: $\langle a\xi + b\eta, \zeta \rangle_x = a \langle \xi, \zeta \rangle_x + b \langle \eta, \zeta \rangle_x$, for all $a, b \in \mathbb{R}$;*
2. *symmetry: $\langle \xi, \eta \rangle_x = \langle \eta, \xi \rangle_x$;*
3. *positive definite: $\langle \xi, \xi \rangle_x \geq 0$, where the equality holds if and only if $\xi = 0$.*

The norm of a tangent vector $\xi \in T_x\mathcal{M}$ is $\|\xi\|_x := \sqrt{\langle \xi, \xi \rangle_x}$.

Definition 2.1.20 (Riemannian metric)**.** *Let $\mathcal{M}$ be a manifold, and $x \in \mathcal{M}$. A Riemannian metric is an inner product $g_x : T_x\mathcal{M} \times T_x\mathcal{M} \mapsto \mathbb{R}$, i.e., a bilinear symmetric positive-definite form, defined on each tangent space $T_x\mathcal{M}$, and varying smoothly with x.*

Definition 2.1.21 (Riemannian manifold)**.** *A Riemannian manifold (M, g) is a manifold $\mathcal{M}$ endowed with a Riemannian metric g.*

A Riemannian manifold $(\mathcal{M}, g)$ is often denoted as $\mathcal{M}$ when the metric is given from the context.

For a smooth, real-valued function f defined on a vector space $\mathcal{E}$, the gradients of f form a vector field on $\mathcal{E}$, along which one realizes local *steepest ascents* in the evaluation of the function values. In the case of a function f defined on a Riemannian manifold $\mathcal{M}$, the notion of *gradients* of f depends on both the properties of f and the differential geometry of $\mathcal{M}$, since the quantification of ascents and steepest ascents in function values depends on a measure of "alignment" between tangent vectors, and the degree of alignment is determined by an inner product on the tangent space. The following definition provides a formal characterization of such an ascent direction.

Definition 2.1.22 (Riemannian gradient)**.** *Let $f : \mathcal{M} \mapsto \mathbb{R}$ be a smooth function on $\mathcal{M}$. The Riemannian gradient of f at $x \in \mathcal{M}$, denoted as* $\mathrm{grad} f(x)$*, is defined as the unique element of $T_x\mathcal{M}$ that satisfies*

$$g_x(\mathrm{grad} f(x), \xi) = \mathrm{D} f(x)[\xi], \text{ for all } \xi \in T_x\mathcal{M}. \tag{2.4}$$

Note that in the above definition, the right-hand side of (2.4) is the directional derivative of f along $\xi \in T_x\mathcal{M}$; see Definition 2.1.15. Remarkably, and similarly to the Euclidean case, the gradient defined above is the steepest-ascent vector field and the norm $\|\mathrm{grad} f(x)\|_x$ is the steepest slope of f at x. More precisely,

$$\|\mathrm{grad} f(x)\|_x = \max_{\xi \in T_x\mathcal{M}; \|\xi\|_x = 1} \mathrm{D}f(x)[\xi],$$

for which the maximizer is $\xi = \mathrm{grad} f(x)/\|\mathrm{grad} f(x)\|_x$ achieves the maximum.

Riemannian quotient manifolds

It is shown in Section 2.1.3 that the tangent vectors to a quotient manifold are defined through their representations in the total space, i.e., the horizontal lifts. Now, we consider quotient manifolds $\mathcal{M}$ of a Riemannian manifold, and present how to equip the quotient manifolds with a Riemannian geometry. Let $\overline{\mathcal{M}}$ be a manifold and $\mathcal{M} = \overline{\mathcal{M}}/\sim$ be a quotient manifold with the canonical projection π. Given two tangent vectors $\xi, \eta \in T_x\mathcal{M}$, a metric on the quotient manifold $\mathcal{M}$ can be defined via any metric $\bar{g}$ on $\overline{\mathcal{M}}$ that satisfies the following invariance property:

Definition 2.1.23. *For $X \in \mathcal{M}$ and $\bar{x} \in \pi^{-1}(x)$, let $\bar{\xi}, \bar{\eta} \in T_{\bar{x}}\overline{\mathcal{M}}$ denote the horizontal lifts of ξ and η respectively, a metric $\bar{g}$ in the total space is said to be* invariant *along $\pi^{-1}(x)$ if*

$$\bar{g}_{\bar{x}}(\bar{\xi}, \bar{\eta}) = \bar{g}_{\bar{y}}(\bar{\xi}_{\bar{y}}, \bar{\eta}_{\bar{y}}), \tag{2.5}$$

for any point $\bar{y} \sim \bar{x}$ in $\pi^{-1}(x)$, where $\bar{\xi}_{\bar{y}}, \bar{\eta}_{\bar{y}}$ denote the horizontal lifts of ξ and η at $\bar{y}$ respectively.

Consequently, an inner product $g_x : T_x\mathcal{M} \times T_x\mathcal{M} \mapsto \mathbb{R}$ such that $g_x(\xi, \eta) = \bar{g}_{\bar{x}}(\bar{\xi}_{\bar{x}}, \bar{\eta}_{\bar{x}})$ is a Riemannian metric on the quotient manifold $\mathcal{M}$, provided that $\bar{g}$ satisfies the invariance property as described in (2.5). Endowed with such a Riemannian metric, the quotient manifold $(\mathcal{M}, g)$ is called a *Riemannian quotient manifold*, and the canonical projection π is a *Riemannian submersion*, *i.e.*, the projection $\mathrm{D}\pi$ is surjective everywhere and preserves the inner products in the tangent space of the quotient manifold $\mathcal{M}$.

2.1.5 Distances and geodesic curves

Definition 2.1.24 (Geodesic)**.** *Let $I \subset \mathbb{R}$ be an open interval and let $\gamma : I \mapsto \mathcal{M}$ be a curve on $\mathcal{M}$. γ is a geodesic if and only if its acceleration is zero on I.*

Definition 2.1.25 (Exponential map)**.** *Let $\mathcal{M}$ be a smooth manifold endowed with a connection ∇ and let $x \in \mathcal{M}$. For every $\xi \in T_x\mathcal{M}$, there exists an open interval I containing zero and a unique geodesic $\gamma_{x,\xi} : I \mapsto \mathcal{M}$ such that*

$\gamma(0) = x$ *and* $\dot{\gamma}(0) = \xi$*. The mapping*

$$\mathrm{Exp}_x : T_x\mathcal{M} \mapsto \mathcal{M} : \xi \mapsto \mathrm{Exp}_x(\xi) = \gamma_{x,\xi}(1)$$

is called the exponential map at x*.*

Note that, the geodesic satisfies the homogeneity property, i.e., $\gamma_{x,\xi}(t) = \gamma_{x,\alpha\xi}(\alpha t)$, for any $\alpha \in \mathbb{R}$.

Definition 2.1.26 (Retraction)**.** *Let* $\mathcal{R} : T_X\mathcal{M} \mapsto \mathcal{M}$ *be a smooth mapping, and for* $x \in \mathcal{M}$*, let* $\mathcal{R}_x$ *denote the restriction of* $\mathcal{R}$ *to* $T_x\mathcal{M}$*. The mapping* $\mathcal{R}$ *is a retraction on* $\mathcal{M}$ *if it satisfies the following properties, for all* $x \in \mathcal{M}$*,*

1. $\mathcal{R}_x(0) = x$*, and*
2. *The differential* $\mathrm{D}\mathcal{R}_x(0) : T_x\mathcal{M} \mapsto T_x\mathcal{M}$ *is the identity map.*

Retractions on quotient manifolds

Proposition 2.1.27 ([AMS08, Proposition 4.1.3])**.** *Let* $\mathcal{M} = \overline{\mathcal{M}}/\sim$ *be a quotient manifold with a prescribed horizontal distribution. Let* $\bar{\mathcal{R}}$ *be a retraction on* $\overline{\mathcal{M}}$ *such that for all* $x \in \mathcal{M}$ *and* $\xi \in T_x\mathcal{M}$*,*

$$\pi(\bar{\mathcal{R}}_{\bar{x}}(\bar{\xi}_{\bar{x}})) = \pi(\bar{\mathcal{R}}_{\bar{y}}(\bar{\xi}_{\bar{y}})) \tag{2.6}$$

for all $x, y \in \pi^{-1}(x)$*. Then the mapping* $\mathcal{R}$ *such that*

$$\mathcal{R}_x(\xi) := \pi(\bar{\mathcal{R}}_{\bar{x}}(\bar{\xi}_{\bar{x}}))$$

defines a retraction on $\mathcal{M}$*.*

2.1.6 Optimization on manifolds: techniques and line-search methods

In this subsection, we present techniques for optimization on Riemannian manifolds, which is also referred to as Riemannian optimization. The manifold-related and differential geometric tools used in these techniques are described in the previous part of Section 2.1.

Given a Riemannian manifold $(\mathcal{M}, g)$ and a smooth, real-valued function f defined on $\mathcal{M}$, the optimization of f on $\mathcal{M}$ is formulated as follows,

$$\min_{x \in \mathcal{M}} \; f(x), \tag{2.7}$$

where the manifold characterization of the feasible set usually originates from the nonlinear structure of the variable x.

The formulation (2.7) arises in many problems including the optimization problems in this thesis. In the graph learning problem in Chapter 3, for example, $\mathcal{M}$ is a compact subset of the unit-sphere. In the matrix and tensor

completion problems in Chapters 4, 6 and 7, $\mathcal{M}$ appears as a product manifold of low-rank matrices. In the optimization problems with fixed-rank matrices in Chapter 5, $\mathcal{M}$ takes the form of a quotient manifold. The rest of this subsection, mainly based on [AMS08, §4], reviews some Riemannian optimization techniques for solving problems in the form of (2.7), with a focus on tools for Riemannian line-search methods.

A basic and ubiquitous approach to solving the optimization problem (2.7) is to produce a sequence of points $\{x_t\}_{t\geq 0}$ on the manifold $\mathcal{M}$ according to a series of *descent directions* of f. Intuitively, this scheme is an analogue of classical line-search methods for unconstrained optimization on an Euclidean vector space, but it involves two main challenges, due to fact that the manifold $\mathcal{M}$ is generally a nonlinear space: (i) since the movement of the iterates needs to meet the nonlinear structure of $\mathcal{M}$, the update rule requires a nonlinear mapping to ensure the manifold constraint, and (ii) the *search directions* for the line-search update rule needs to be defined regarding the local geometry of the Riemannian manifold.

Retractions and search directions

For the first challenge, the retraction operator (Definition 2.1.26) provides systematic tool for moving on the manifold along any direction in the tangent space. The underlying update rule is as follows, given a retraction $\mathcal{R}$ and an iterate $x_t \in \mathcal{M}$,

$$x_{t+1} = \mathcal{R}_{x_t}(s\eta_t), \tag{2.8}$$

where $\eta_t \in T_{x_t}\mathcal{M}$ is a search direction and the scalar s a stepsize to be discussed later.

Ideally, the exponential map (Definition 2.1.25) is a natural choice for the retraction operator in (2.8): given an iterate x_t on $\mathcal{M}$ and a search direction η_t in the tangent space $T_{x_t}\mathcal{M}$, the operation $x_{t+1} = \mathrm{Exp}_{x_t}(s\eta_t)$, for a certain stepsize $s \geq 0$, is a valid update rule. However, from a numerical optimization point of view, the computation of the exponential mapping corresponds to solving a nonlinear ordinary differential equation, which leads to significant computational requirements; moreover, it is often computationally intractable. Nevertheless, in many cases, the exponential map is not a necessary option, in the computational point of view. Alternatively, by designing the retraction operator as the first-order (or second-order in certain cases) approximation of the exponential map, one can ensure the manifold constraint with much reduced complexity, without compromising the convergence property of the Riemannian line-search method. Throughout this thesis, we consider the update rule using a retraction with such characterization.

For the second challenge, the search direction in the update rule (2.8) needs to be defined in a way such that the sequence $\{x_t\}_{t\geq 0}$ has a chance to reach a *minimizer* of f on $\mathcal{M}$. The definition of global and local minimizers in numerical optimization on a vector space [NW06, §2.1] applies to the Riemannian optimization problem (2.7). Indeed, if an element $x \in \mathcal{M}$ is a global minimizer

of the problem (2.7), then it is necessarily a local minimizer in an open neighborhood of x in $\mathcal{M}$; also, if x is a local minimizer of the problem (2.7), then x is necessarily a *critical point* of f on $\mathcal{M}$. This motivates the search of critical points of f on the manifold $\mathcal{M}$, and thus, requires the first-order information of f on $\mathcal{M}$; such information is naturally assessed via the Riemannian gradients (Definition 2.1.22) of f. In particular, through the characterization (2.4) of the Riemannian gradient, any element $x \in \mathcal{M}$ satisfying $\mathrm{grad} f(x) = 0$ is a critical point of f on $\mathcal{M}$. Therefore, in the framework of Riemannian line-search methods [AMS08, §4.2], the search directions and stepsizes in (2.8) are selected in relation to the Riemannian gradient of f, as we describe next.

The following definition is useful for determining if a sequence of search directions are sufficiently aligned with the negative Riemannian gradients of f on noncritical points.

Definition 2.1.28 (Gradient-related sequence)**.** *Given a real-valued, smooth function f defined on a Riemannian manifold $\mathcal{M}$, let $\{\eta_t\}$ be a sequence such that $\eta_t \in T_{x_t}\mathcal{M}$, for $\{x_t\}_{t\geq 0} \subset \mathcal{M}$. The sequence $\{\eta_t\}$ is gradient-related if, for any subsequence $\{x_t\}_{t\in\mathcal{K}}$ of $\{x_t\}$ that converges to a noncritical point of f, the corresponding subsequence $\{\eta_t\}_{t\in\mathcal{K}}$ is bounded and satisfies*

$$\lim_{t\to\infty} \sup_{t\in\mathcal{K}} g_{x_t}(\mathrm{grad} f(x_t), \eta_t) < 0.$$

Definition 2.1.29 (Armijo point)**.** *Given a real-valued, smooth function f on $\mathcal{M}$, with retraction $\mathcal{R}$, a point $x \in \mathcal{M}$, a tangent vector $\eta \in T_x\mathcal{M}$, and scalars $\alpha > 0$ (an initial stepsize), $\beta, \sigma \in (0,1)$, the Armijo point is $\eta_\ell := \alpha\beta^\ell\eta \in T_x\mathcal{M}$, where ℓ is the smallest nonnegative integer such that*

$$f(x) - f(\mathcal{R}_x(\alpha\beta^\ell\eta)) \geq -\sigma g_x(\mathrm{grad} f(x), \alpha\beta^\ell\eta).$$

The scalar $s := \alpha\beta^\ell$ is called the Armijo stepsize.

The Armijo line-search criterion above, along with the gradient-relatedness property in Definition 2.1.28—which is shared by many Riemannian gradient-based algorithms—provide conditions for establishing global convergence results of line-search methods with (2.8); see [AMS08, §4.3] for example.

2.2 Low-rank matrices: representations and Riemannian geometries

In this section, we take a look at sets of low-rank real matrices in a geometric point of view. Based on the elements about manifolds and Riemannian geometry given in Section 2.1, we present a manifold description of the set of fixed-rank matrices, along with geometric tools used in some Riemannian optimization methods on this manifold; these contents are mainly based on [Van13, MMBS14, UV19, UV20b] about numerical methods with low-rank

matrices.

Given $m, n, r \in \mathbb{N}_+$ such that $r \leq \min(m, n)$, let $\mathcal{M}_{\leq r}$ be the set of $m \times n$ real matrices with rank equal or lower than r, i.e.,

$$\mathcal{M}_{\leq r} = \{X \in \mathbb{R}^{m\times n} : \operatorname{rank}(X) \leq r\}. \tag{2.9}$$

Since the matrix rank is a nonlinear function on $\mathbb{R}^{m\times n}$, the rank-constrained set $\mathcal{M}_{\leq r}$ is a nonlinear subset of $\mathbb{R}^{m\times n}$. In fact, $\mathcal{M}_{\leq r}$ is a real algebraic variety, and is nonsmooth on matrices (or points) that have a rank strictly smaller than r; see [UV19, UV20b]. The smooth part of $\mathcal{M}_{\leq r}$ is the set of the rank-r matrices

$$\mathcal{M}_r = \{X \in \mathbb{R}^{m\times n} : \operatorname{rank}(X) = r\}, \tag{2.10}$$

which is of particular importance in understanding the numerical methods with low-rank matrices.

In the rest of this section, we describe representations of low-rank matrices, a manifold structure of the set $\mathcal{M}_r$ (2.10) and geometric tools of some existing algorithms on $\mathcal{M}_r$.

2.2.1 Parametrization via matrix factorizations

Because of the low-rank constraints in (2.9) and (2.10), the $m \times n$ real matrices in $\mathcal{M}_r$ and $\mathcal{M}_{\leq r}$ can be parametrized by fewer than mn variables, and this is possible by representing these matrices through matrix factorization. Several different forms of matrix factorization can be used for such parametrization. Next, we list some matrix factorization forms for the parametrization of a matrix X in $\mathcal{M}_r \subset \mathcal{M}_{\leq r}$, followed by some examples of application in numerical methods with low-rank matrices.

- ◇ The *singular-value decomposition* (SVD) of X is

$$X = U\Sigma V^{\mathrm{T}}, \tag{2.11}$$

  where $U \in \operatorname{St}(m, r) = \{U \in \mathbb{R}^{m\times r} : U^{\mathrm{T}}U = I_r\}$, $V \in \operatorname{St}(n, r)$, and $\Sigma = \operatorname{Diag}(\sigma_1, \ldots, \sigma_r) \succ 0$. In particular, with a fixed order of the singular values, e.g., the decreasing order $\sigma_1 \geq \cdots \geq \sigma_r > 0$, the SVD representation (U, Σ, V) of X via (2.11) is unique. The representation (2.11) is called the *compact* SVD (in contrast to the full SVD), as it only involves the non-zero singular values and corresponding singular vectors.

  The SVD representation is used in e.g., [Van13] for low-rank matrix completion.

- ◇ The *UY factorization* of X is as follows,

$$X = UY^{\mathrm{T}}, \tag{2.12}$$

where $U \in \mathrm{St}(m, r)$ is a matrix that contains the left-singular vectors of X—as the factor U defined in (2.11)—and $Y \in \mathbb{R}^{n\times r}$.

The UY representation (2.12) is useful in the variable projection approach for low-rank matrix problems, e.g., in the RTRMC algorithm [BA15] for low-rank matrix completion.

◇ The *two-term factorization* of X is as follows,

$$X = LR^{\mathrm{T}}, \tag{2.13}$$

where L and R are $m \times r$ and $n \times r$ matrices with the full column-rank r respectively, i.e., $(L, R) \in \mathbb{R}_*^{m\times k} \times \mathbb{R}_*^{n\times k}$. An example of (L, R) satisfying (2.13) is $(L, R) = (U\Sigma^{\frac{1}{2}}, V\Sigma^{\frac{1}{2}})$, where U, V, Σ are the SVD factor matrices of X in (2.11). The two-term factorization (2.13) is not unique, and in fact, there are infinite many pairs of rank-r factor matrices satisfying (2.13). It suffices to note that, for any matrix $F \in \mathrm{GL}(k)$, $(L', R') := (LF^{-1}, RF^{\mathrm{T}})$ satisfies (2.13) if $(L, R) \in \mathbb{R}_*^{m\times k} \times \mathbb{R}_*^{n\times k}$ does.

The two-term factorization is used in the graph-regularized matrix completion problem and also in the quotient manifold approach to low-rank matrix problems, which will be discussed in detail in Chapters 4 and 5 respectively.

2.2.2 Geometry of the set of fixed-rank matrices

The set $\mathcal{M}_r$ (2.10) is known to have a smooth manifold structure; see, e.g., [Lee03, Example 8.14], which shows through the construction of a local submersion that $\mathcal{M}_r$ is a smooth embedded submanifold of $\mathbb{R}^{m\times n}$. From another perspective, through matrix factorization (e.g., (2.11)–(2.13)), any matrix X in $\mathcal{M}_r$ can be seen as an equivalence class in the domain of definition of the factor matrices of X, in the sense that two tuples of factor matrices are equivalent if and only if their product matrices in $\mathcal{M}_r$ are identical; see Example 2.1.13. This means that the set $\mathcal{M}_r$ is inherently a quotient space; moreover, it can be interpreted as a Riemannian quotient manifold; see [MBS11, MMBS14, AAM14].

In this subsection, we describe the geometry of $\mathcal{M}_r$ as an embedded submanifold of $\mathbb{R}^{m\times n}$; the quotient manifold structure of $\mathcal{M}_r$ will be presented later, in Chapter 5, in relation to recent and novel developments for Riemannian optimization with fixed-rank matrices.

Based on the first-order perturbation of the SVD (e.g., [Van13, UV19]) of a matrix X in $\mathcal{M}_r$, one has the following characterization for the tangent space to $\mathcal{M}_r$ at X, which involves the full SVD form. Note that, in addition to the factor matrices (U, Σ, V) of the compact SVD (2.11) of X, two orthonormal matrices, $U_\perp \in \mathrm{St}(m, m-r)$ and $V_\perp \in \mathrm{St}(n, n-r)$, corresponding to the orthogonal complements $\mathrm{span}(U)^\perp$ and $\mathrm{span}(V)^\perp$ are used.

Proposition 2.2.1 ([Van13, Proposition 2.1])**.** *The set $\mathcal{M}_r$ (2.10) is a smooth submanifold of dimension $(m+n-r)r$ embedded in $\mathbb{R}^{m\times n}$. Given $X = U\Sigma V^{\mathrm{T}} \in$*

$\mathcal{M}_r$, where (U, Σ, V) is the SVD *representation of X via* (2.11)*, the tangent space at X to $\mathcal{M}_r$ is as follows,*

$$T_X\mathcal{M}_r = \big\{UAV^{\mathrm{T}} + UB^{\mathrm{T}}V_\perp^{\mathrm{T}} + U_\perp CV^{\mathrm{T}} : A \in \mathbb{R}^{r\times r}, B \in \mathbb{R}^{(n-r)\times r}, C \in \mathbb{R}^{(m-r)\times r}\big\} \tag{2.14a}$$

$$= \Big\{UAV^{\mathrm{T}} + UB_p^{\mathrm{T}} + C_pV^{\mathrm{T}} : A \in \mathbb{R}^{r\times r}, B_p \in \mathbb{R}^{n\times r}, C_p \in \mathbb{R}^{m\times r}, U^{\mathrm{T}}C_p = 0, V^{\mathrm{T}}B_p = 0\Big\}. \tag{2.14b}$$

We refer to [UV19, §2.2] and [Van13, §2] for the proof of the proposition above.

From (2.14a) in Proposition 2.2.1, and by the complementarity of span(U) and span$(U_\perp)$ (respectively span(V) and span$(V_\perp)$), the tangent space $T_X\mathcal{M}_r$ can also be written as follows,

$$T_X\mathcal{M}_r = \{U\tilde{B}^{\mathrm{T}} + \tilde{C}V^{\mathrm{T}} : \tilde{B} \in \mathbb{R}^{n\times r}, \tilde{C} \in \mathbb{R}^{m\times r}\}. \tag{2.15}$$

Any matrix in the tangent space $T_X\mathcal{M}_r$, through (2.14a), consists of three components, each corresponding to one of the three mutually orthogonal subspaces. The orthogonal projections onto these three subspaces can be written using $\mathcal{P}_U := UU^{\mathrm{T}}$ and $\mathcal{P}_V := VV^{\mathrm{T}}$, which are the orthogonal projections onto span(U) and span(V) respectively. More precisely, the orthogonal projection operator $\mathrm{P}_{T_X\mathcal{M}_r} : \mathbb{R}^{m\times n} \mapsto T_X\mathcal{M}_r$ is as follows, for any $Z \in \mathbb{R}^{m\times n}$,

$$\mathrm{P}_{T_X\mathcal{M}_r}(Z) = \mathcal{P}_UZ\mathcal{P}_V + \mathcal{P}_UZ(I - \mathcal{P}_V) + (I - \mathcal{P}_U)Z\mathcal{P}_V \tag{2.16a}$$

$$= \mathcal{P}_UZ + Z\mathcal{P}_V - \mathcal{P}_UZ\mathcal{P}_V. \tag{2.16b}$$

Given a matrix $X \in \mathcal{M}_r$ and a matrix ξ in the tangent space $T_X\mathcal{M}_r$, the displacement of X on $\mathcal{M}_r$ along ξ can be realized by the exponential map, which is shown ([Van13, AM12]) to have the following form.

Proposition 2.2.2 (Exponential map on $\mathcal{M}_r$)**.** *Given a matrix $X = U\Sigma V^{\mathrm{T}} \in \mathcal{M}_r$, where (U, Σ, V) is the* SVD *representation of X via* (2.11)*, the exponential map on $\mathcal{M}_r$ is as follows, for any matrix $\xi \in T_X\mathcal{M}_r$,*

$$\mathrm{Exp}_X(\xi) = X + \xi + (I - \mathcal{P}_U)\xi X^\dagger\xi(I - \mathcal{P}_V) + \Delta_X(\xi), \tag{2.17}$$

where $X^\dagger$ denotes the Moore–Penrose pseudoinverse of X and Δ_X is the sum of third- and higher-order terms, in the sense that $\|\Delta_X(\xi)\| \lesssim \|\xi\|^3$.

2.2.3 Tools for Riemannian optimization on $\mathcal{M}_r$

In Section 2.2.2, the set $\mathcal{M}_r$ is described as an embedded submanifold in $\mathbb{R}^{m\times n}$ and the characterization of the tangent space to $\mathcal{M}_r$ is given. In this subsection, we present some geometric tools for Riemannian optimization with fixed-rank

matrices based on the aforementioned manifold structure. Given a smooth, real-valued function f defined on $\mathbb{R}^{m\times n}$, the optimization of f with matrices of a fixed rank r inspires the Riemannian optimization problem (2.7), where the manifold $\mathcal{M}$ takes the form of $\mathcal{M}_r$ (2.10).

The following two elements—retraction and Riemannian metric on $\mathcal{M}_r$—are useful for Riemannian line-search methods (Section 2.1.6) with fixed-rank matrices. In particular, the retraction operator presented below is used in the algorithms such as LRGeomCG [Van13], SVP (singular value projection) [JMD10], IHT (iterative hard thresholding) [GM11], NIHT [TW13] and variants (e.g., CGIHT). The rest of this subsection is mainly based on [Van13, §2].

Retraction on $\mathcal{M}_r$ via metric projection

As mentioned in Section 2.1.6, in Riemannian optimization, the retraction (Definition 2.1.26) is an operator that maps a point in the tangent space to the manifold. Through the characterizations in its definition, any retraction operator is a (at least) first-order approximation of the exponential mapping on the manifold.

Regarding $\mathcal{M}_r$ as a submanifold embedded in $\mathbb{R}^{m\times n}$, one can define the following retraction, $\mathcal{R} : T_X\mathcal{M}_r \mapsto \mathcal{M}_r$, via metric projection onto $\mathcal{M}_r$,

$$\mathcal{R}_X(\xi) = \operatorname*{arg\,min}_{Z\in\mathcal{M}_r} \|X+\xi-Z\|_{\mathrm{F}}, \text{ for all } \xi \in \mathcal{U}_X, \tag{2.18}$$

where $\mathcal{U}_X \subset T_X\mathcal{M}_r$ is a certain neighborhood of the zero matrix. The domain of definition of $\mathcal{R}$ (2.18) needs only be a (sufficiently small) local neighborhood since it suffices to obtain convergence for the Riemannian line-search methods. The retraction mapping (2.18) can be obtained by the truncated SVD (r-SVD) of $X+\xi \in T_X\mathcal{M}_r$, i.e.,

$$\mathcal{R}_X(\xi) = \sum_{i=1}^{r} \sigma_i \tilde{u}_i \tilde{v}_i^{\mathrm{T}}, \tag{2.19}$$

where $(\sigma_i, \tilde{u}_i, \tilde{v}_i)$ are the r leading singular values and vectors of $X+\xi$.

Note in particular that the singular value computations required by (2.19) has a complexity that depends on the rank value r instead of the matrix dimensions (m, n), since the search direction ξ and thus $X+\xi$ are in the tangent space $T_X\mathcal{M}_r$. In fact, the matrix $\xi \in T_X\mathcal{M}_r$ in (2.18) is determined by a $r\times r$ matrix A via the parametrization $\xi = UAV^{\mathrm{T}} + C_pV^{\mathrm{T}} + UB_p^{\mathrm{T}}$ (see (2.14b)), where U, V, C_p and B_p are the matrices in the parametrization of the given matrix X, and it follows that

$$X+\xi = [U \quad U_p] \underbrace{\begin{bmatrix} \Sigma + A & I_r \\ I_r & 0 \end{bmatrix}}_{\tilde{\Sigma}\in\mathbb{R}^{2r\times 2r}} [V \quad V_p]^{\mathrm{T}},$$

which shows that the SVD of $X+\xi$ can be obtained via the SVD of $\tilde{\Sigma} \in \mathbb{R}^{2r\times 2r}$,

which costs only $\mathcal{O}\left(r^3\right)$.

Riemannian geometry on $\mathcal{M}_r$

The Riemannian line-search methods (see Section 2.1.6) require access to search directions in relation with the Riemannian gradients of f. This requires a Riemannian geometry on the manifold $\mathcal{M}_r$. By identifying $\mathcal{M}_r$ as a manifold embedded in $\mathbb{R}^{m\times n}$, and considering the Euclidean inner product $\langle\cdot,\cdot\rangle$ in the tangent space $T_X\mathbb{R}^{m\times n} \simeq \mathbb{R}^{m\times n}$, for a matrix $X \in \mathcal{M}_r$, one can define the following metric g_X through the restriction of $\langle\cdot,\cdot\rangle$ to $T_X\mathcal{M}_r$:

$$g_X(\xi,\eta) = \langle\xi,\eta\rangle := \mathrm{tr}(\xi^{\mathrm{T}}\eta), \tag{2.20}$$

for all $\xi,\eta \in T_X\mathcal{M}_r$. From the definition (2.20), the inner product g_X does not depend on the location of X on $\mathcal{M}_r$, hence, naturally, $(X,\xi,\eta) \mapsto g_X(\xi,\eta)$ varies smoothly on the tangent bundle $T\mathcal{M}_r$. Therefore, $(\mathcal{M}_r, g)$ forms a Riemannian manifold; see Definition 2.1.21. Subsequently, combining Definition 2.1.22 and (2.20), the Riemannian gradient of f is computed through the identification

$$g_X(\mathrm{grad}f(X),\xi) = \mathrm{D}f(X)[\xi], \text{ for all } \xi \in T_X\mathcal{M}_r.$$

Details about the low-rank parametrization and computation of the Riemannian gradient $\mathrm{grad}f(X)$ can be found in [Van13, §3].

2.3 Graphs

Graphs are an omnipresent data structure. Intuitively, a graph is a set of objects that are connected with each other through a certain relation. In this section, we give a formal description of graphs and some fundamental properties. These properties are background knowledge needed in many graph-related problems and are useful for the development of graph-based techniques. In this thesis, the *graph-regularized* matrix and tensor completion problems are examples where the graph data structure is used; see Chapters 4 and 6. We also refer the interested reader to [Ham20, §1-2] for an overview of graph-related machine learning problems.

A graph is a pair of sets, denoted as $\mathcal{G} = (\mathcal{V},\mathcal{E})$, where $\mathcal{V}$ is a set of nodes (or vertices) and $\mathcal{E}$ is a set of edges $\mathcal{E}$ betweens these nodes. Each edge in $\mathcal{E}$ represents the connection between a pair of nodes, $(i,j) \in \mathcal{V}\times\mathcal{V}$. By default, we consider graphs that do not contain *loops*; in other words, the element (i,i), for any $i \in \mathcal{V}$, is not considered as an edge. Therefore, the set of edges $\mathcal{E}$ is such that $\mathcal{E} \subset \mathcal{V}\times\mathcal{V}\setminus\{(i,i) : i \in \mathcal{V}\}$. The edges in $\mathcal{E}$ may be *directed* or *undirected*, depending on whether the elements (i,j) and (j,i) are distinguished or not. If (i,j) and (j,i) are considered as equivalent, then, the edge connecting i and j is undirected and is denoted by (i,j) or equivalently (j,i). The graph $\mathcal{G}$ is

an undirected graph if its edges are undirected. In this thesis, by default, an undirected graph is simply referred to as a graph.

The adjacency matrix of a graph $\mathcal{G}$ of n nodes is a square matrix $A(\mathcal{G})$, or simply A, that represents the set of edges $\mathcal{E}$. For any undirected graph $\mathcal{G}$, the adjacency matrix $A(\mathcal{G})$ is symmetric. The edges of $\mathcal{G}$ are *unweighted* if the elements in $\mathcal{V} \times \mathcal{V}$ are represented by a binary variable. In this case, the adjacency matrix can be represented by a $(0,1)$-valued matrix, such that $A_{ij} = 1$ if and only if $(i,j) \in \mathcal{E}$. On the other hand, *weighted* graph edges are represented by a nonzero real variable. For a graph with nonnegative edge weights, the adjacency matrix A is a real matrix such that $A_{ij} > 0$ if $(i,j) \in \mathcal{E}$, otherwise $A_{ij} = 0$. The edge weight A_{ij}, for $(i,j) \in \mathcal{E}$, represents the intensity of the connection between the nodes i and j; the larger the value of A_{ij}, the stronger the connection between these two vertices.

Definition 2.3.1 (Degree matrix)**.** *Given a graph $\mathcal{G}$ endowed with an adjacency matrix $A(\mathcal{G}) \in \mathbb{R}^{n\times n}$, the degree matrix, denoted as $D(\mathcal{G})$ or simply D, is a diagonal matrix such that $D_{ii} = \sum_{j\in\mathcal{V}} A_{ij}$. The i-th diagonal entry $d_i = D_{ii}$ is referred to as the node degree of i.*

Definition 2.3.2 (Graph Laplacian matrix)**.** *Given a graph $\mathcal{G}$ endowed with an adjacency matrix $A(\mathcal{G}) \in \mathbb{R}^{n\times n}$, the graph Laplacian matrix of $\mathcal{G}$, or simply the graph Laplacian, is denoted and defined as follows,*

$$L = D(\mathcal{G}) - A(\mathcal{G}), \tag{2.21}$$

where $D(\mathcal{G})$ is the degree matrix and $A(\mathcal{G})$ is the adjacency matrix of the graph.

From the graph spectral theory (e.g., [Chu97, Spi12]), any graph Laplacian matrix is positive semi-definite and the smallest eigenvalue of a graph Laplacian matrix is zero. In fact, the graph Laplacian satisfies the following property.

Proposition 2.3.3 (e.g., [Chu97, §1.4])**.** *Given an undirected graph $\mathcal{G}$ of n nodes endowed with a weighted adjacency matrix A, the graph Laplacian matrix $L(\mathcal{G}) \in \mathbb{R}^{n\times n}$ defined in* (2.21) *satisfies*

$$x^T L x = \frac{1}{2} \sum_{i,j=1}^{n} A_{ij} \left(x_i - x_j\right)^2 \tag{2.22}$$

for all $x \in \mathbb{R}^n$.

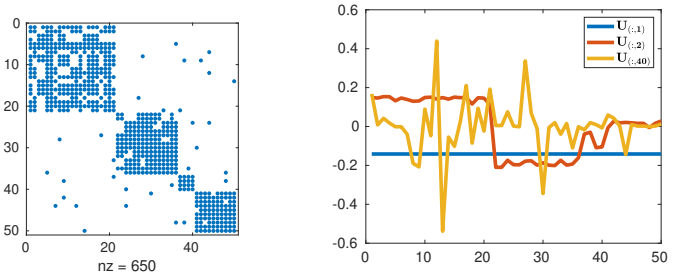
Proof. The degree matrix D of $\mathcal{G}$ is such that $D_{ii} = \sum_{j=1}^{n} A_{ij}$, for $i = 1, \ldots, n$, hence, from (2.21), it holds that $x^{\mathrm{T}} L x = \sum_{i,j=1}^{n} (A_{ij}x_i^2 - A_{ij}x_i x_j)$. It follows that

$$x^{\mathrm{T}} L x = \sum_{i,j=1}^{n} \Big(\frac{1}{2} A_{ij}(x_i^2 + x_j^2) - A_{ij} x_j x_j\Big) = \frac{1}{2} \sum_{i,j=1}^{n} A_{ij}(x_i - x_j)^2, \tag{2.23}$$

where the first equality in (2.23) holds since the adjacency matrix of the undirected graph $\mathcal{G}$ is symmetric, i.e., $A = A^{\mathrm{T}}$. □

From Proposition 2.3.3, any graph Laplacian matrix (Definition 2.3.2) is symmetric and positive semidefinite. By convention, we denote the eigenvalues of a Laplacian matrix $L(\mathcal{G})$ by $\{\lambda_i\}_{i=1,\ldots,|\mathcal{V}|}$ in increasing order, such that $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_{|\mathcal{V}|}$.

Figure 2.2 shows the graph Laplacian matrix of a graph containing several closely connected *clusters* along with some eigenvectors of the graph Laplacian. One can see from the eigenvectors of the smallest eigenvalues (λ_1 and λ_2), in Figure 2.2(b), that they exhibit pairwise similarities, which can be explained, through Proposition 2.3.3, by the fact that the values of the quadratic form $x^{\mathrm{T}}Lx$ is small on such eigenvectors.



(a) (b)

Figure 2.2: Eigenvectors of a graph Laplacian matrix. (a): the adjacency matrix of an unweighted graph $\mathcal{G}$ of 50 nodes; (b): the eigenvectors of the graph Laplacian matrix L of $\mathcal{G}$ corresponding to the eigenvalues λ_1 (blue), λ_2 (red) and λ_{40} (yellow) respectively.

2.4 Tensors: definitions and notation

In this section, we introduce the definitions and notation involved in the tensor operations.

The term *tensor* refers to a multidimensional array. The dimensionality of a tensor is described as its order. A k^{th}-order tensor is a k-way array, also known as a k-mode tensor. We use the term *mode* to describe operations on a specific dimension (e.g., mode-k matricization). A real-valued order-k tensor is defined as $\mathcal{Z} = [z_{\ell_1,\ldots,\ell_k}] \in \mathbb{R}^{m_1 \times m_2 \times \ldots \times m_k}$, where an element $z_{\ell_1,\ldots,\ell_k}$ is accessed via the k-dimensional index $(\ell_1, \ldots, \ell_j, \ldots, \ell_k)$, with $\ell_j \in [\![m_j]\!] := \{1, 2, \ldots, m_j\}$. The following definitions are involved in the tensor computations.

Definition 2.4.1 (Kronecker product $\otimes$)**.** *The Kronecker product of vectors* $\mathbf{u} = [u_\ell] \in \mathbb{R}^{m_1}$ *and* $\mathbf{v} = [v_\ell] \in \mathbb{R}^{m_2}$ *results in a vector* $\mathbf{u} \otimes \mathbf{v} \in \mathbb{R}^{m_1 m_2}$ *defined*

as

$$\mathbf{u} \otimes \mathbf{v} = \begin{bmatrix} u_1\mathbf{v} \\ u_2\mathbf{v} \\ \vdots \\ u_{m_1}\mathbf{v} \end{bmatrix}.$$

More compactly, we have $(\mathbf{u} \otimes \mathbf{v})_{m_2(\ell_1-1)+\ell_2} = u_{\ell_1} v_{\ell_2}$ *for* $\ell_1 \in \{1, \ldots, m_1\}$ *and* $\ell_2 \in \{1, \ldots, m_2\}$.

Definition 2.4.2 (Khatri–Rao product $\odot$)**.** *The Khatri–Rao product* $U \odot V$ *of two matrices* $U = [u_{\ell,r}] \in \mathbb{R}^{m_1 \times R}$ *and* $V = [v_{\ell,r}] \in \mathbb{R}^{m_2 \times R}$ *with the same column number is a matrix of size* $m_1 m_2 \times R$ *whose* r*-th column is* $u_{:,r} \otimes v_{:,r}$.

Definition 2.4.3 (Mode-ℓ product $\times_\ell$)**.** *The mode-*ℓ *product of a given tensor* $\mathcal{G} \in \mathbb{R}^{r_1 \times \cdots \times r_k}$ *with a matrix* $U \in \mathbb{R}^{m \times r_\ell}$*, denoted as* $\mathcal{G} \times_\ell U$*, is a tensor of size* $r_1 \times \cdots \times r_{\ell-1} \times m \times r_{\ell+1} \cdots \times r_k$*, which has entries*

$$[\mathcal{G} \times_\ell U]_{i_1,\ldots,i_{\ell-1},j,i_{\ell+1},\ldots,i_k} = \sum_{p=1}^{r_\ell} U_{j,p} \mathcal{G}_{i_1,\ldots,i_{\ell-1},p,i_{\ell+1},\ldots,i_k}.$$

Definition 2.4.4 (Hadamard product $\star$)**.** *The Hadamard product of two matrices* A *and* B *of the same dimensions, denoted by* $A \star B$*, is a matrix of the same dimensions by entrywise multiplications:* $[A \star B]_{ij} = A_{ij} B_{ij}$.

Definition 2.4.5 (Tensor matricization)**.** *The mode-*i *matricization* $\mathcal{Z}_{(i)}$ *is the unfolding of a tensor* $\mathcal{Z} \in \mathbb{R}^{m_1 \times m_2 \times \ldots \times m_k}$ *along its* i*-th mode of size* $m_i \times \left(\prod_{j \neq i} m_j\right)$*. The tensor element* $z_{\ell_1,\ldots,\ell_k}$ *in* $\mathcal{Z}$ *is identified with the matrix element* $\left[Z_{(i)}\right]_{\ell_i, r_i}$ *in* $\mathcal{Z}_{(i)}$*, where*

$$r_i = 1 + \sum_{\substack{n=1 \\ n \neq i}}^{k} (\ell_n - 1)\, I_n, \quad \text{with} \quad I_n = \prod_{\substack{j=1 \\ j \neq i}}^{n-1} m_j. \tag{2.24}$$

Definition 2.4.6 (Tucker decomposition)**.** *The Tucker decomposition of a tensor [DLDMV00a, DLDMV00b, Tuc66] is defined as an approximation of a core tensor* $\mathcal{C} \in \mathbb{R}^{r_1 \times \cdots \times r_k}$ *multiplied by* k *(orthogonal) factor matrices* $U^{(i)} \in \mathbb{R}^{m_i \times R_i}, i = 1, \ldots, k$ *along each mode, such that*

$$\mathcal{Z} = \mathcal{C} \times_1 U^{(1)} \times_2 \cdots \times_k U^{(k)}. \tag{2.25}$$

Remark 2.4.7. *A tensor* $\mathcal{Z}$ *admitting a Tucker decomposition form as in* (2.25) *can be unfolded along its* i*-th mode as follows,*

$$\mathcal{Z}_{(i)} = U^{(i)} \mathcal{C}_{(i)} \left(U^{(i-1)} \otimes \cdots \otimes U^{(1)} \otimes U^{(k)} \otimes \cdots \otimes U^{(i+1)}\right)^\top,$$

Definition 2.4.8 (Tensor Tucker rank)**.** *Tensor Tucker rank or multilinear rank [KM16], which is introduced earlier by Tucker [Tuc66] is defined as*

$$rank_{TC}\left(\mathcal{Z}\right) = \left(rank\left(\mathcal{Z}_{(1)}\right), \ldots, rank\left(\mathcal{Z}_{(k)}\right)\right),$$

where $rank\left(\mathcal{Z}_{(i)}\right)$ *denotes the matrix rank.*

Definition 2.4.9 (Rank-1 tensor)**.** *A tensor of the form*

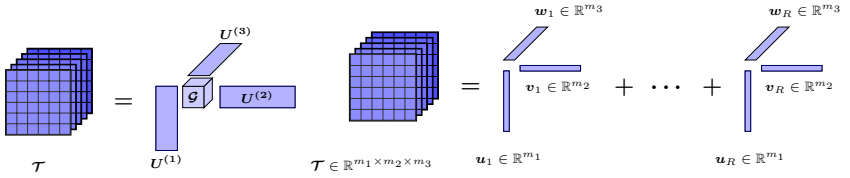
$$\mathcal{Z} = \mathbf{u}^{(1)} \circ \ldots \circ \mathbf{u}^{(k)} = \left[u_{i_1}^{(1)} \ldots u_{i_k}^{(k)}\right]_{i_1,\ldots,i_k},$$

where $\circ$ *denotes the outer product, is said to be of rank one. It is also called a simple tensor [Her10] or decomposable tensor [Hac12].*

Definition 2.4.10 (CP decomposition)**.** *The Canonical Polyadic (CP) decomposition [CC70, FBH03, Har70, Hit27a, Kie00] of a tensor* $\mathcal{Z}$ *is defined as*

$$\mathcal{Z} = [\![U^{(1)}, \ldots, U^{(k)}]\!] = \sum_{r=1}^{R} U_{:,r}^{(1)} \circ \ldots \circ U_{:,r}^{(k)},$$

where $U^{(i)} \in \mathbb{R}^{m_i \times R}$ *for* $i = 1, \ldots, k$.



(a) (b)

Figure 2.3: Tensor decompositions. (a): Tucker decomposition; (b): CP decomposition.

The CP decomposition can be considered as the "diagonalized" version of the Tucker decomposition (Definition 2.4.6), in the sense that a tensor that admits a rank-R CP decomposition also admits a Tucker decomposition with a cubic core tensor of dimension $r = (R, \ldots, R)$, which is composed of nonzero values only on the cubic diagonal entries. The matricizations of a tensor in the CPD form $[\![U^{(1)}, \ldots, U^{(k)}]\!]$ can be written as

$$\mathcal{Z}_{(i)} = U^{(i)} \left(U^{(k)} \odot \ldots \odot U^{(i+1)} \odot U^{(i-1)} \odot \ldots \odot U^{(1)}\right)^{\top} = U^{(i)} \left[(U^{(j)})^{\odot_{j \neq i}}\right]^{\top},$$

where $\mathcal{Z}_{(i)}$ is the mode-i matricization.

Definition 2.4.11 (Tensor CP rank)**.** *The tensor CP rank of a tensor* $\mathcal{Z}$ *is defined as the minimum number of summations of rank-one tensors that generate*

$\mathcal{Z}$ *[Hit27b, Kru77], i.e.,*

$$\operatorname{rank}_{CP}(\mathcal{Z}) = \min\left\{ R \in \mathbb{Z}_+ : \exists \left\{ \mathbf{u}_r^{(i)} \right\}, s.t.\ \mathcal{Z} = \sum_{r=1}^{R} \mathbf{u}_r^{(1)} \circ \ldots \circ \mathbf{u}_r^{(k)} \right\}.$$

Chapter 3

Learning graphs from data

Graphs model the structure of a discrete space, for example, they incorporate correlations or pairwise similarities between objects. Therefore, graphs are a fundamental source of information for the analysis of structured data and are at the basis of many signal processing and machine learning techniques such as semi-supervised learning and spectral clustering. However, in many graph-related problems, the graph structure does not exist in a natural way. Typically, given a set of multivariate data samples, one needs to construct or infer a graph from the data itself. While there is no unique way in defining a graph that conforms to the hidden structures in the data, there are ways to impose restrictions on the graph structure based on certain prior knowledge about the data. A common approach to constructing graphs is to build graph adjacency matrices using affinity graph models [ZCLG14] such as the k-nearest-neighbor (k-NN) model, which can be obtained depending on an affinity matrix associated with the data features.

A recent optimization-based approach, called graph learning [DTFV16, Kal16, EPO16], infers a graph structure by minimizing a smoothness function given a set of multivariate data samples, under graph Laplacian-related constraints. The graph learning problems proposed by [DTFV16] and [Kal16] minimize an objective function with the graph Laplacian or graph adjacency matrix variable as follows,

$$f(L) = \mathrm{tr}(\widehat{C}L) + H(L) \text{ or } f(W) = \mathrm{tr}(\widehat{Z}W) + H(W), \tag{3.1}$$

where $\widehat{C}$ and $\widehat{Z}$ are the empirical covariance and the pairwise distance matrices of the data samples respectively, and the regularization function H is designed according to the prior knowledge about the graph, notably the properties of the node degree distribution. The trace term in the cost function of (3.1) originates from the maximum likelihood principle with the Gaussian Markov Random Fields (GMRF) model [RH05, LW13] and the inverse covariance estimation problem [FHT08, BED08, HBDR12, PHB17]. Given an empirical covariance

matrix $\widehat{C}(X)$, where the entries of X are i.i.d. samples of a GMRF model $\mathcal{N}(\mathbf{0}, \Theta)$, the maximum-likelihood estimator of the inverse covariance matrix Θ can be cast as the following problem (e.g., [FHT08, BED08]),

$$\min_{\Theta \succ 0} \ \mathrm{tr}(\widehat{C}\Theta) - \log(\det(\Theta)) + \lambda \|\Theta\|_1, \tag{3.2}$$

where the ℓ_1 norm $\|\cdot\|_1$ is defined as $\|\Theta\|_1 := \sum_{ij} |\Theta_{ij}|$, and the parameter λ is determined according to a certain model selection criterion. Graph learning is different from inverse covariance estimation in the sense that it searches for a graph Laplacian matrix, and thus requires the variable to satisfy several constraints according to the feasible set of all graph Laplacian matrices; the Laplacian matrix variable is positive semi-definite by construction (see Proposition 2.3.3). As a consequence, the search space of graph learning is substantially different than that of (sparse) inverse covariance estimation. Moreover, depending on specific prior knowledge (mainly on the distribution of node degrees) about the graph Laplacian matrix to be learned, various optimization formulations are proposed.

The rest of this chapter is partly based on the conference paper [DAG18] and is organized as follows. In Section 3.1, we give a description of the basic definitions and notation used in this chapter, and present the background and related work about graph learning. In Section 3.2, we present a new problem formulation for graph learning and describe the proposed algorithms. Numerical experiments with synthetic data and real-world applications are given in Section 3.3. We conclude the chapter in Section 3.4 with discussions.

3.1 Preliminaries

In this section, we present the background of the graph learning problem, definitions and terminology involved in this topic. The notion of graphs and related definitions are given in Section 2.3.

We consider an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ of m nodes and edges with nonnegative weights. The graph adjacency matrix of $\mathcal{G}$ is represented by a real symmetric matrix with nonnegative entries. Moreover, since the graph does not contain loops, by default, the diagonal entries of the adjacency matrix are zeros. Therefore, the graph adjacency matrices of these graphs form a set $\mathcal{W}_+^m$ as follows,

$$\mathcal{W}_+^m := \left\{ W \in \mathbb{R}^{m \times m} : W = W^{\mathrm{T}},\ W \geq 0,\ \mathrm{diag}(W) = 0 \right\}. \tag{3.3}$$

The set $\mathcal{W}_+^m$ (3.3) is the nonnegative cone of the vector space $\mathcal{W}^m$ of real symmetric matrices, $\mathcal{W}^m := \left\{ W : W = W^{\mathrm{T}}, \mathrm{diag}(W) = 0 \right\}$. Note that $\mathcal{W}^m$ is a linear space with dimension $\frac{m(m-1)}{2}$. Indeed, the set $\bar{\mathcal{E}}$ of all possible edges (i, j) between the m nodes in $\mathcal{V}$ has at most $\frac{m(m-1)}{2}$ elements. By convention, (i, j) denotes an *undirected* couple of indices that represents the

edge between the nodes i and j, i.e., (i,j) is equivalent to (j,i) for all $i \neq j \in \{1,\dots,m\}$, since the graph edges are undirected. The set of the $\frac{m(m-1)}{2}$ unit-norm symmetric matrices $\{E_{(i,j)} = \frac{1}{\sqrt{2}}\left(\boldsymbol{e}_i\boldsymbol{e}_j{}^{\mathrm{T}} + \boldsymbol{e}_j\boldsymbol{e}_i{}^{\mathrm{T}}\right) : (i,j) \in \bar{\mathcal{E}}\}$, form an orthonormal basis of $\mathcal{W}^m$, and any graph adjacency matrix $W \in \mathcal{W}^m_+$ can be written as $W = \sum_{(i,j)\in\bar{\mathcal{E}}} \sqrt{2} W_{ij} E_{(i,j)}$. Hence, $W \in \mathcal{W}^m_+$ can be represented by its *half-vectorization* in $\mathbb{R}^{m(m-1)/2}_+$, denoted and defined as

$$\mathrm{vec}_{\mathbb{S}}(W) := \sqrt{2}[W_{1,2},\dots,W_{ij},\dots,W_{m-1,m}]^{\mathrm{T}}_{1\leq i<j\leq m}, \tag{3.4}$$

which is the stacked vector of the strict upper triangle of W (up to the scalar $\sqrt{2}$); the subscript of $\mathrm{vec}_{\mathbb{S}}$ signifies that the vectorization above applies specifically to symmetric matrices. Note that the linear application $\mathrm{vec}_{\mathbb{S}} : \mathcal{W}^m_+ \mapsto \mathbb{R}^{m(m-1)/2}_+$ is bijective, and its inverse, defined on $\mathbb{R}^{m(m-1)/2}_+$, is denoted as $\mathrm{vec}_{\mathbb{S}}^{-1}$.

Given a graph $\mathcal{G} = (\mathcal{V},\mathcal{E})$ endowed with an adjacency matrix $W \in \mathcal{W}^m_+$, the graph Laplacian matrix (see Definition 2.3.2) of $\mathcal{G}$ is $L := D(\mathcal{G}) - W = \mathrm{Diag}(W\mathbf{1}) - W$, where $\mathrm{Diag}(W\mathbf{1})$ denotes the diagonal matrix whose diagonal is composed of the entries of $W\mathbf{1} \in \mathbb{R}^m$. Hence, the graph Laplacian matrices associated with the adjacency matrices in $\mathcal{W}^m_+$ form the following set,

$$\mathcal{L}^m = \{L \in \mathbb{R}^{m\times m} : (\forall i \neq j)\ L_{ij} = L_{ji} \leq 0 \text{ and } L\mathbf{1} = 0\}, \tag{3.5}$$

which is, similar to $\mathcal{W}^m_+$, a closed and convex subset of $\mathbb{R}^{m\times m}$. Any graph Laplacian matrix $L \in \mathcal{L}^m$ is positive semi-definite. In fact, for any vector $x \in \mathbb{R}^m$, the quadratic form $x \mapsto x^{\mathrm{T}}Lx$ satisfies $x^{\mathrm{T}}Lx = \frac{1}{2}\sum_{i,j} W_{ij}(x_i - x_j)^2 \geq 0$, where the equality holds if and only if $x \in \mathrm{span}(\mathbf{1}_m)$; see Proposition 2.3.3. Hence, this quadratic form, denoted as

$$\mathrm{S}_{\mathcal{G}}(x) := x^{\mathrm{T}}Lx = \frac{1}{2}\sum_{i,j=1}^{m} W_{ij}(x_i - x_j)^2, \tag{3.6}$$

is positive semi-definite, and defines the square of a semi-norm on $\mathbb{R}^m$. Let $x \in \mathbb{R}^m$ be a non-zero vector, which can be interpreted as a signal defined on the set $\mathcal{V}$ of m nodes, in the perspective of graph signal processing. In view of Proposition 2.3.3, one has the following remark about $\mathrm{S}_{\mathcal{G}}$.

Remark 3.1.1. The squared semi-norm $\mathrm{S}_{\mathcal{G}}(x)$ (3.6) is a measure of the overall variations of x on the graph $\mathcal{G}$, in the sense that $\mathrm{S}_{\mathcal{G}}(x)$ is small (up to the scale, e.g., $\|x\|_2$) if the values of the signal x on any subset of connected nodes evolve slowly/smoothly. □

From the remark above, if a graph signal $x \in \mathbb{R}^{|\mathcal{V}|}$ has a small squared semi-norm $\mathrm{S}_{\mathcal{G}}(x)$, it is also qualified as a graph signal that exhibits *pairwise similarities* on the graph $\mathcal{G}$. Figure 2.2 (the blue and red signals) shows some examples of graph signals that have pairwise similarities.

3.1.1 Related work.

The problem of learning a graph Laplacian matrix from data samples, introduced in [DTFV16], formulated via the maximum-likelihood of a graph-based Gaussian Markov random fields model, is as follows

$$\min_{L\in\mathcal{L}^m,\ \mathrm{tr}(L)=\theta} \mathrm{tr}(X^{\mathrm{T}}LX) + \beta\|L\|_F^2, \tag{3.7}$$

where X is the matrix of the given multivariate data samples, the feasible set $\mathcal{L}^m$ is defined in (3.5), $\beta \geq 0$ and $\theta > 0$ is a constant parameter. The equality constraint $\mathrm{tr}(L) = \mathbf{1}^{\mathrm{T}}W\mathbf{1} = \theta$ in (3.7) ensures that the sum of all vertex degrees remain constant, while the Frobenius norm of L is a penalty term added to the objective function. The penalty term with $\beta \geq 0$ controls the trade-off between the minimizer of the pure data fitting term (for $\beta = 0$), whose vertex degree distribution may be undesirable, and the fully connected graph with equal weights (for extremely large β).

More recently, [Kal16] proposed to reformulate the graph Laplacian learning problem (3.7) as optimization problems of the weighted adjacency matrix W, by using the following property.

Proposition 3.1.2 ([Kal16, §2])**.** *Given a undirected graph $\mathcal{G}$ of m nodes, endowed with an adjacency matrix W. Then, for any matrix $X \in \mathbb{R}^{m\times n}$, $\mathrm{tr}(X^{\mathrm{T}}LX) = \frac{1}{2}\sum_{i,j=1}^m W_{ij}\|X_{i,:} - X_{j,:}\|_2^2 = \frac{1}{2}\mathrm{tr}(ZW)$, where Z is the Euclidean distance matrix such that $Z_{ij} = \|X_{i,:} - X_{j,:}\|_2^2$ and L is the graph Laplacian matrix of $\mathcal{G}$.*

As a result, the objective function of (3.7) is transformed into a function in W, $f(W) = \frac{1}{2}\mathrm{tr}(ZW) + \beta\|W\|_{\mathrm{F}}^2$. The feasible set of W is $\mathcal{W}_+^m$, which has fewer constraints than the feasible set of graph Laplacian matrices (3.5): while both sets require the $m(m-1)/2$ inequality constraints, the set $\mathcal{L}$ has m additional equality constraints, i.e., $L\mathbf{1} = 0$. The equality constraint $\mathrm{tr}(L) = \theta$ in (3.7) is equivalent to $\mathbf{1}^{\mathrm{T}}W\mathbf{1} = \theta$. Kalofolias [Kal16] considered relaxing this equality constraint using instead a penalty term with the log barrier function $-\sum_{i=1}^m \log(d_i(W))$. The proposed model in [Kal16] is as follows,

$$\min_{W\in\mathcal{W}_+^m} \frac{1}{2}\mathrm{tr}(ZW) - \alpha\mathbf{1}^{\mathrm{T}}\log(W\mathbf{1}) + \beta\|W\|_F^2, \tag{3.8}$$

where $\log(\cdot)$ applies element-wisely to the vector $W\mathbf{1}$ of node degrees. Unlike the equality constraint of (3.7) on $\mathrm{tr}(L)$ (acting on the sum of all node degrees), the log barrier-based penalty term forces all node degrees to be positive. This improves the overall connectivity of the graph without compromising the sparsity of the solution, since only the node degrees are forced to be strictly positive but not the individual coefficients of W.

The graph learning model (3.8) does not have the same statistical interpretation as sparse inverse covariance estimation, and the choice of the regularization parameters (α, β) depends on the application purpose for which the

learned graph is used. Note that, for a given parameter $\alpha > 0$, the model selection of (3.8) in [Kal16] is conducted by searching for a value of $\beta \geq 0$ via grid search. Since all entries of the distance matrix Z are nonnegative, the function $\mathrm{tr}(ZW)$ in (3.8) acts as the sparsity promoting term, thus the smaller $\beta \geq 0$ (for $\beta\|W\|_F^2$) is the more sparse is the solution supposed to be.

3.2 Fixed-scale graph learning

Given a data matrix $X \in \mathbb{R}^{m\times n}$, the goal of graph learning from the data is to learn a graph Laplacian L or adjacency matrix W that represents the pairwise similarities between the rows (or columns) of X. From Proposition 2.3.3, Remark 3.1.1 and Proposition 3.1.2, this goal amounts to the optimization of $\mathrm{tr}(CL)$ or $\mathrm{tr}(ZW)$, given an empirical covariance matrix C or a distance matrix Z associated with the data matrix X.

3.2.1 Problem statement

In the existing graph learning formulations, given the data fitting functions $\mathrm{tr}(CL)$ or $\mathrm{tr}(ZW)$, the graph Laplacian matrix L or adjacency matrix W is learned with respect to a certain restriction on the *scale*, which can be measured by, e.g., $\|W\|_1$, $\mathrm{tr}(L)$ or the Frobenius norms of these matrices. This scale restriction is necessary, since otherwise the minimizer to $\mathrm{tr}(CL)$ or $\mathrm{tr}(ZW)$ is the zero matrix, for any covariance matrix C or distance matrix Z. Indeed, the scale of W (or L), is irrelevant to the the pairwise relations encoded in this matrix. In other words, the relations between the nodes are *invariant* to any dilatation of the matrix coefficients, in the sense that two graph adjacency matrices W' and W are equivalent if there exists $c > 0$ such that $W' = cW$. This equivalence relation suggests that the graph learning problem has an inherent fixed-scale constraint, and we take an interest in determining the fixed-scale constraint using the Frobenius norm of W; the ℓ_1 norm-based constraint, i.e., fixing $\|W\|_1$ to a constant, leads to the minimization of a nonlinear function on the simplex, which does not belong to the focus of this work.

In this section, we propose to learn a graph adjacency matrix W in the intersection of the unit-sphere and the nonnegative cone $\mathcal{W}_+^m$: $\overline{\mathcal{W}}_+^m = \{W \in \mathbb{R}^{m\times m} : W = W^{\mathrm{T}}, W \geq 0, \|W\|_{\mathrm{F}} = 1\}$. Given this search space, we consider minimizing the following objective function, in a manner similar to that used for (3.8),

$$F_\alpha(W) := \frac{1}{2}\,\mathrm{tr}(\tilde{Z}W) - \alpha \mathbf{1}^{\mathrm{T}} \log(W\mathbf{1}), \tag{3.9}$$

where the input matrix $\tilde{Z}$ is a distance matrix, which is to be determined later depending on the data in X. Furthermore, since any matrix $W \in \overline{\mathcal{W}}_+^m$ is symmetric, we deal with the half-vectorization $\mathrm{vec}_{\mathbb{S}}(W)$ (3.4) of W directly. The feasible set of $\mathrm{vec}_{\mathbb{S}}(W)$ for all $W \in \overline{\mathcal{W}}_+^m$ is, by construction, the nonnegative

orthant of the unit-sphere in $\mathbb{R}^{n_m}$,

$$\mathcal{S}_+^{n_m-1} = \{w \in \mathbb{R}^{n_m} : w \geq 0, \|w\|_2 = 1\}, \tag{3.10}$$

where the superscript, $n_m := \frac{m(m-1)}{2}$, denotes the dimension of the half vectorization $\mathrm{vec}_{\mathbb{S}}(W)$. Hence, the fixed-scale graph learning has the following formulation,

$$\min_{w \in \mathcal{S}_+^{n_m-1}} f_\alpha(w) := w^{\mathrm{T}} \mathrm{vec}_{\mathbb{S}}(\tilde{Z}) - \alpha \mathbf{1}^{\mathrm{T}} \log(\mathcal{D}(w)), \tag{3.11}$$

where the objective function f_α is defined such that $f_\alpha(\mathrm{vec}_{\mathbb{S}}(W)) = F_\alpha(W)$ in (3.9), which implies that $\mathcal{D} : \mathbb{R}^{n_m} \mapsto \mathbb{R}^m$ is the linear operator satisfying $\mathcal{D}(w) = W\mathbf{1}$, where $W = \mathrm{vec}_{\mathbb{S}}^{-1}(w) \in \mathcal{W}_+^m$. The feasible set of (3.11) is defined in (3.10).

The rest of this section is organized as follows. Before Section 3.2.2, we present a generalized way of generating the input data $\tilde{Z}$ for the graph learning problem. In Section 3.2.2, we present the proposed algorithm for solving the problem 3.11.

Distance matrix through kernels

Given a data matrix $X \in \mathbb{R}^{m \times n}$, the pairwise distance matrix with $Z_{ij} = \|X_{i,:} - X_{j,:}\|_{\mathrm{F}}^2$ is generalized to a symmetric positive definite matrix depending on a given feature map $\Phi : \mathbb{R}^n \mapsto \mathbb{R}^p$. The feature mapping by Φ is realized implicitly, as in kernel-based methods [Sch01], such that the inner products in the feature space are encoded by a Gram matrix $\tilde{C} \in \mathbb{R}^{m \times m}$, with $\langle \Phi(X_{i,:}), \Phi(X_{j,:}) \rangle := \tilde{C}_{ij}$. The matrix $\tilde{C}$ is defined via a given positive semi-definite function $K : \mathbb{R}^n \times \mathbb{R}^n \mapsto \mathbb{R}$, which is also called a kernel function,

$$\tilde{C}_{ij} = K\big(X_{i,:}, X_{j,:}\big). \tag{3.12}$$

Hence, the properties of the features are determined by the given kernel function K. In particular, the sample covariance estimate XX^{T} corresponds to the case where the kernel function K reduces to the Euclidean inner product $K(x, y) = x^{\mathrm{T}}y$. In a similar way as done for Proposition 3.1.2, the following proposition shows how to construct a generalized distance matrix $\tilde{Z}$ based on a given kernel function K.

Proposition 3.2.1. *Given a undirected graph $\mathcal{G}$ of m nodes endowed with an adjacency matrix W, a Gram matrix $\tilde{C} \in \mathbb{R}^{m \times m}$ generated from a matrix $X \in \mathbb{R}^{m \times n}$ by* (3.12)*, and a matrix $\tilde{Z} \in \mathbb{R}^{m \times m}$ be defined as*

$$\tilde{Z} = \mathrm{diag}(\tilde{C})\mathbf{1}^{\mathrm{T}} + \mathbf{1}\mathrm{diag}(\tilde{C})^{\mathrm{T}} - 2\tilde{C}, \tag{3.13}$$

it holds that $\mathrm{tr}(\tilde{C}L) = \frac{1}{2}\mathrm{tr}(\tilde{Z}W)$, where L is the graph Laplacian matrix of $\mathcal{G}$.

The proof is given in [DAG18, Appendix B].

3.2.2 Constrained optimization on the sphere

In this section, we propose a gradient projection algorithm for solving the problem (3.11) on the unit-sphere.

Interestingly, for the simple differential properties of the sphere, the projection onto this subset $\mathcal{S}_+^{n-1}$ can be obtained using straightforward computations. We consider the projection with respect to the Euclidean distance,

$$\mathrm{P}_{\mathcal{S}_+^{n-1}}(x) = \operatorname*{arg\,min}_{w \in \mathcal{S}_+^{n-1}} \|w - x\|_2, \tag{3.14}$$

for all $x \in \mathbb{R}^n$. The result below provides a closed-form expression for (3.14).

Proposition 3.2.2. *For all $x \in \mathbb{R}^n$,*

$$\mathrm{P}_{\mathcal{S}_+^{n-1}}(x) = \begin{cases} x_+/\|x_+\|_2 & \text{if } x \notin \mathbb{R}^n_- \\ \boldsymbol{e}_i, \text{ with } i = \arg\max\{x_i\} & \text{otherwise,} \end{cases} \tag{3.15}$$

where $x_+ = (\max\{x_i, 0\})_{i=1,\ldots,n}$ and $\mathbb{R}^n_- = \{x \in \mathbb{R}^n : x_i \leq 0, \forall i \in \{1, \ldots, n\}\}$.

Proof. We follow the same proof as for (Lemma 1 of [ZLWZ18]) which solves $\min_{w \in \mathcal{S}_+^{n-1}} b^{\mathrm{T}} w$. It suffices to identify the projection in (3.14) as the case where $b = -x$. □

Although the projection of a point $x \in \mathbb{R}^n$ onto the nonconvex set $\mathcal{S}_+^{n-1}$ may not be unique, the non-uniqueness happens in fact only if $x \in \mathbb{R}^n_-$ and if x admits multiple maximal coefficients. In this last case, one solution is chosen arbitrarily. The computation for (3.15) is light. Based on the explicit form (3.15) for the projection $\mathrm{P}_{\mathcal{S}_+^{n-1}}$, we describe the projected gradient method in Algorithm 3.2.1.

The backtracking line search (line 6) is a simplified form of the nonmonotone line search in [BMR00]. However, for the graph learning problem (3.11), there is no clear guarantee that the line search step always terminate. The initialization step (line 1) corresponds to generating a random graph and initializing w^0 with the graph's edge weights.

Similar to the algorithm of Kalofolias [Kal16], Algorithm 3.2.1 has a per-iteration time complexity of $\mathcal{O}(|\mathcal{V}|^2)$. Note that even without any extra structural priors (other than the sparsity of graph edges) for learning the graph, this basic algorithm has a much lower computational complexity than basic methods (e.g., [FHT08, BED08]) for sparse inverse covariance estimation, which normally amount to $\mathcal{O}(|\mathcal{V}|^3)$.

Due to the special structure of the search space on the unit-sphere, Algorithm 3.2.1 is an instance of nonconvex projected gradient methods whose convergence properties are discussed in recent studies; see [JK17, §3] and [BH18] for details.

Algorithm 3.2.1 Projected gradient on $\mathcal{S}_+^{n-1}$ (GL-SPH).

Input: f_α ($\alpha > 0$); $\sigma, \beta \in (0,1)$, $M, K \geq 1$.
Output: $w \in \mathcal{S}_+^{n-1}$.

1: Initialization: $w = w^0 \in \mathcal{S}_+^{n-1}$.
2: **for** $k \in \{1, \ldots, K\}$ **do**
3: **if** $\| \mathrm{P}_{\mathcal{S}_+^{n-1}}(w^k - \nabla f_\alpha(w^k)) - w^k \|_2 \leq \epsilon$ **then**
4: stop.
5: **end if**
6: Line search:
(6.a) Set the initial stepsize $s_k^0 = 1$,
(6.b) Find the smallest integer $\ell \geq 0$ such that for $s_k = s_k^0 \beta^\ell$ and $w_+ = \mathrm{P}_{\mathcal{S}_+^{n-1}}(w^k - s_k \nabla f_\alpha(w^k))$,

$$f_\alpha(w_+) \leq \max_{0 \leq j \leq \min(k, M-1)} f_\alpha(w^{k-j}) + \sigma \left\langle \nabla f_\alpha(w^k), w_+ - w^k \right\rangle. \quad (3.16)$$

7: Update: $w^{k+1} = w_+$.
8: **end for**

Computational details. In Algorithm 3.2.1, the calculation of the Euclidean gradient $\nabla f_\alpha(w)$ is as follows. Recall that the objective function is $F_\alpha(W) = \mathrm{tr}(\tilde{Z}^{\mathrm{T}} W) - \alpha \sum_{\ell=1}^m \log(\sum_{j=1}^m W_{\ell j}) := \mathrm{tr}(\tilde{Z}^{\mathrm{T}} W) - \alpha H(W)$, for any $W \in \mathcal{W}_+^m$. The gradient of F_α is $\nabla F_\alpha(W) = \tilde{Z} - \alpha \nabla H(W)$, where $\nabla H(W) := \sum_{(i,j) \in \bar{\mathcal{E}}} \partial_{(i,j)} H(W) E_{(i,j)}$, is computed as follows. The partial derivatives in $\nabla H(W)$ with respect to the basis $\{E_{(i,j)} : (i,j) \in \bar{\mathcal{E}}\}$ (see Section 3.1), through the chain rule, can be written as

$$\partial_{(i,j)} H(W) = \sum_{\ell=1}^m \partial_{d_\ell} \left(\log(d_\ell(W)) \right) \partial_{(i,j)} \left(d_\ell(W) \right),$$

where $d_\ell(W) := \sum_{j=1}^m W_{\ell j}$ is the ℓ-th node degree. Note that $\partial_{d_\ell} \left(\log(d_\ell(W)) \right) = 1/d_\ell(W)$ and $\partial_{(i,j)} \left(d_\ell(W) \right)$ reads

$$\partial_{(i,j)} \left[d_\ell(W) \right] := \frac{\mathrm{d}}{\mathrm{d}t} (d_\ell(W + t E_{(i,j)})) \Big|_{t=0} = \sum_{j'=1}^m \frac{\delta_{i\ell}\delta_{jj'} + \delta_{j\ell}\delta_{ij'}}{\sqrt{2}} = \frac{\delta_{i\ell} + \delta_{j\ell}}{\sqrt{2}}.$$

Therefore, we have

$$\partial_{(i,j)} H(W) = \sum_{\ell=1}^m \frac{1}{\sqrt{2} d_\ell(W)} (\delta_{i\ell} + \delta_{j\ell}) = \frac{1}{\sqrt{2}} (d_i(W)^{-1} + d_j(W)^{-1}). \quad (3.17)$$

Through the identification $f_\alpha(\mathrm{vec}_{\mathbb{S}}(W)) = F(W)$ and the linearity of the

half vectorization operator, the gradient $\nabla f_\alpha(\mathrm{vec}_{\mathbb{S}}(W)) = \mathrm{vec}_{\mathbb{S}}(\nabla F_\alpha(W))$.

3.3 Numerical experiments

In the first part of this section, we evaluate the performance of our graph learning method on fully observed synthetic data and compare it with the state-of-the-art algorithm [Kal16]. In the second part, we apply graph learning to several graph-dependent tasks in machine learning: (i) spectral clustering, and (ii) semi-supervised learning.

3.3.1 Graph learning on synthetic data

In this part, a synthetic data matrix is generated using a graph-based model, given a certain randomly generated graph. Therefore, the graph underlying the synthetic data is considered as the ground truth or the reference graph of the graph learning methods, and thus, we evaluate the learned graphs by comparing directly to the reference graph. More precisely, the set of edges of a learned graph matrix W, $\{(i,j) \in \mathcal{V} \times \mathcal{V} : W_{ij} > 0\}$, is considered as a binary classification on $\mathcal{V} \times \mathcal{V}$, and the quality of the learned graph is evaluated by the following methods: (i) the ROC curve [Faw06], which shows the true positive rate against the false positive rate of the learned edges; (ii) the F-measure [vR79] or the F-score, which refers to the harmonic mean of the precision and recall scores of the learned edges $\mathcal{E}$ in comparison with the edges $\mathcal{E}^\star$ of the reference graph. In particular, the F-measure is computed from both the *hard* classification result—with $\{\mathbf{1}_{>0}(W)\}$—and soft classification result—with $\{\mathbf{1}_{\geq\epsilon}(W)\}$, for a small threshold $\epsilon > 0$; and (iii) the relative error, which is defined as $\|\frac{W}{\|W\|_{\mathrm{F}}} - \frac{W^\star}{\|W^\star\|_{\mathrm{F}}}\|_{\mathrm{F}}$ (an error relative to the unit-norm), where $W^\star$ is the adjacency matrix of the reference graph.

Graph-based data model

We generate synthetic data with the following low-rank matrix model, which can be seen as a generalization of the model in [RYRD15, §5.1]. Let $\mathcal{G}^{\mathrm{r}} := (\mathcal{V}^{\mathrm{r}}, \mathcal{E}^{\mathrm{r}}, L^{\mathrm{r}})$ and $\mathcal{G}^{\mathrm{c}} := (\mathcal{V}^{\mathrm{c}}, \mathcal{E}^{\mathrm{c}}, L^{\mathrm{c}})$ be the graphs that model the row-wise and column-wise similarities of a data matrix X and let $(U^{\mathrm{r}}, \Lambda^{\mathrm{r}})$ (resp. $(U^{\mathrm{c}}, \Lambda^{\mathrm{c}})$) denote the pair of matrices containing the eigenvectors and associated eigenvalues of L^{r} (resp. L^{c}), the data matrix X is generated as follows,

$$X = A^{\mathrm{r}} Z (A^{\mathrm{c}})^{\mathrm{T}}, \tag{3.18}$$

where $Z \in \mathbb{R}^{m \times n}$ is a Gaussian matrix and the matrices $A^{\mathrm{r}} \in \mathbb{R}^{m \times m}$ and $A^{\mathrm{c}} \in \mathbb{R}^{n \times n}$ are defined as follows,

$$A^{\mathrm{r}} = U^{\mathrm{r}} g(\Lambda^{\mathrm{r}}) \text{ and } A^{\mathrm{c}} = U^{\mathrm{c}} g(\Lambda^{\mathrm{c}}), \tag{3.19}$$

where the function $g : \mathbb{R} \mapsto \mathbb{R}$ acts elementwise on the diagonal matrices, i.e., $g(\Lambda) = \mathrm{Diag}\,(g(\lambda_1), \ldots, g(\lambda_m))$ for any diagonal matrix Λ. The function g in (3.19) enables one to control the way in which the graph information in L^{r} and L^{c} transforms the random matrix Z. In the literature of graph signal processing [SNF+13], the function g is referred to as a graph spectral filter, which is a graph analogue of filters in signal processing. In our experiments, we set the function g as $g(\lambda) = \lambda^{-p}$ if $\lambda > 0$ and 0 if $\lambda = 0$, with $p \geq 1$. The spectral model (4.46) is a typical example of functions that are monotonically non-increasing over $\mathbb{R}_+^*$, which have the effect of low-pass filters [SNF+13] in the graph spectral domain [SRV16]. Other examples include (i) the Tikhonov filter (*e.g.* [BMN04]) $g_\gamma(\lambda) = 1/\sqrt{1+\gamma\lambda}$, and (ii) the diffusion operator [CLL+05, CM06, ZH08] $g_\tau(\lambda) = e^{-\tau\lambda}$.

The graph Laplacian matrices underlying the data model (3.18)–(3.19) are generated from several common graph models randomly generated using the GSP toolbox [PPS+14]. These graph types are as follows: (i) *Community*, which refers to the type of graphs that have a clear partitioning pattern with closely connected *clusters* or subgraphs. Every node has typically much more connections with nodes of the same cluster than nodes of other clusters, (ii) *Sensor* networks, which refer to the type of graphs where the set $\mathcal{V}$ of nodes is embedded in the 2-dimensional square $[0, 1]^2$ and uniformly distributed. The edge weights are then defined by

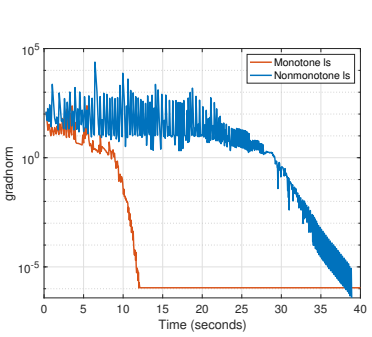
$$W_{ij} = Z_{ij}\mathbf{1}_{\geq\epsilon}(Z_{ij}), \tag{3.20}$$

where $Z_{ij} = \exp\left(-\frac{\|\mathcal{V}_i - \mathcal{V}_j\|_2^2}{2\sigma^2}\right)$ and (iii) the *Erdős-Rényi* random graph [Gil59] model.

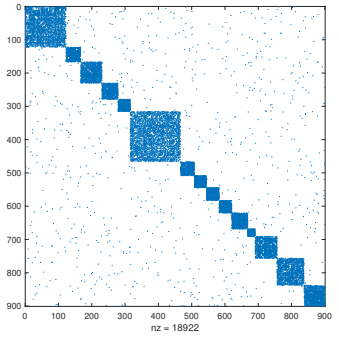
In the following experiments, the graph learning tests are conducted based on input matrices $\tilde{Z}$ described as follows, depending on the size m and the graph type underlying the data model (3.18)–(3.19). Given a data matrix $X \in \mathbb{R}^{m\times n}$ generated by the model (3.18)–(3.19), the input matrix $\tilde{Z}$ for graph learning is a generalized distance matrix associated with X, through (3.12)–(3.13). Note that (3.12) corresponds to the inner products between the rows of X, which is just a convention; it suffices to input X^{T} if one needs to learn a graph on the column index set. We set the kernel function K in (3.12) as the Euclidean inner product.

Experiments and results

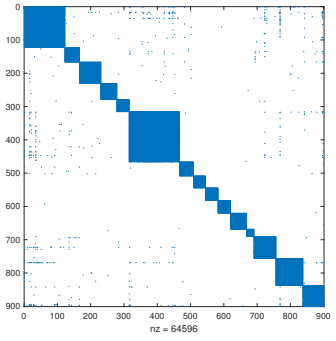
In the following experiment, we compare the performances of Algorithm 3.2.1 (with $M = 10$ in (3.16)) and a variant of it using a more standard line search , i.e., monotone line search, which corresponds to setting the option $M = 1$ in the line search criterion (3.16). We conduct graph learning on the synthetic data matrix (3.18) using the two different line search methods, starting from the same randomly generated initial graph matrix (with a graph edge sparsity set at 0.05).



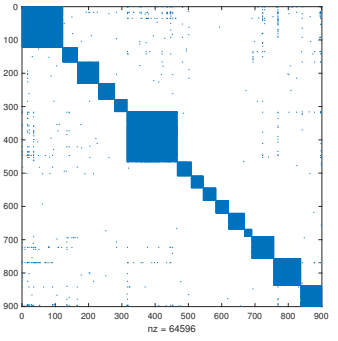
(a) Iteration histories



(b) Reference graph



(c) Nonmonotone line search



(d) Monotone line search

Figure 3.1: Non-monotone vs monotone line search. Graph size: $m = 900$. (c)–(d): graph learning results with the two line search methods.

The numerical results in Figure 3.1 illustrate the main improvement observed in numerous repeated tests: the algorithm using the nonmonotone line search avoids the stagnation issue that the monotone line search encounters, and it terminates successfully in these tests under the sole standard stopping criterion (Algorithm 3.2.1, line 3), for a tolerance parameter $\epsilon = 10^{-8}$ (which corresponds to a gradient norm of f of around 10^{-7}). The algorithm using monotone line search, on the other hand, stagnates earlier than expected and has to be terminated using extra stopping criteria. While the solutions given by these two line search variants are similar, the elapsed time of the algorithm using nonmonotone line search is significantly reduced. Therefore, we adopt the nonmonotone line search procedure, with an option $M = 10$, in all experiments that follow.

Next, we focus on choice of the regularization parameter α of the graph learning problem (3.11), which controls the trade-off between the data fitting term and the regularization function $H(\mathrm{vec}_{\mathbb{S}}(W)) = -\mathbf{1}^{\mathrm{T}} \log(W\mathbf{1})$. In the following tests, we evaluate six trial values of α in $\{0.005, 0.01, 0.015, 0.02, 0.2, 2\}$ using Algorithm 3.2.1, with one (unique) randomly generated initial graph.

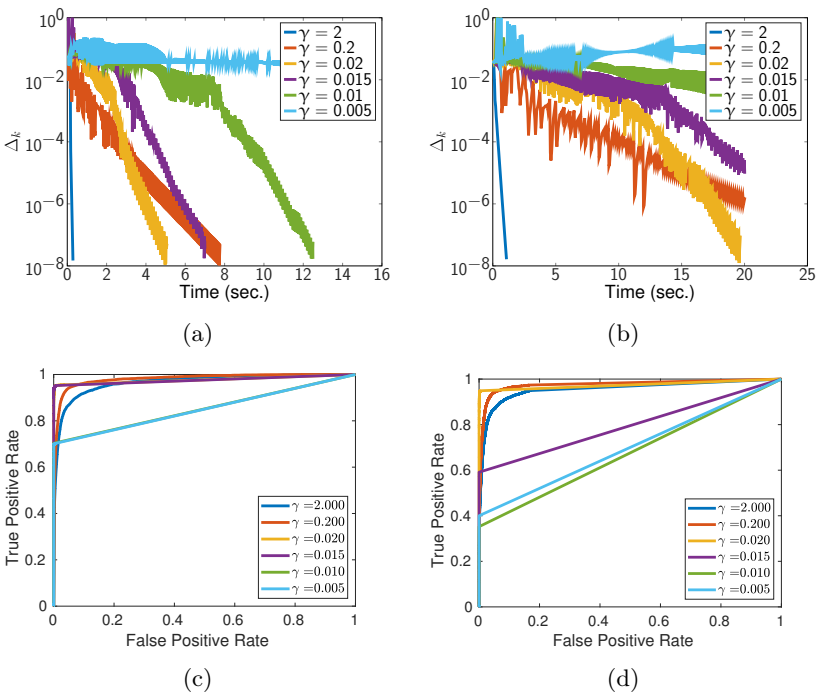


Figure 3.2: Performance of Algorithm 3.2.1 with $\tilde{Z} \in \mathbb{R}^{n \times n}$ from synthetic data. (a)–(b): Iteration history in tests with $n = 200$ and $n = 1000$, respectively. $\Delta_k = \|P_{\mathcal{S}_+^{q-1}}\left(w^k - \nabla f_\gamma(w^k)\right) - w^k\|_2$. (c)–(d): Accuracy of the learned graph edges with respect to the reference graph.

Figure 3.2 shows the graph learning results under several regularization parameters in terms of the ROC curve and time efficiency. We observe that the proposed algorithm has faster convergence behavior when the regularization parameter is larger. On the other hand, from the accuracies of the graph learning results, the optimal regularization parameter, among the tested values, is in the middle of the range of all choices (in the log-scale), and the accuracy decreases once the regularization parameter surpasses a certain magnitude. Such observations conform to the explanation above about the role of the regularization parameter α.

In the following experiment, we evaluate the graph learning performances of the proposed algorithm (Algorithm 3.2.1) in comparison to (i) the state-of-the-art algorithm[1] [Kal16] and (ii) a baseline method using the k-nearest-neighbors (k-NN) graph model, for which the edge weights are defined using the radial basis function kernel (RBF kernel). For all graph types, the number of nodes is set to $m = 300$ and number of samples is $n = 2000$. The scores are obtained by running 20 times the same experiment with parameter values selected through grid search for each method.

Table 3.1 shows the performance of the graph learning results in comparison to the reference graphs. On all three graph-based data types, the two graph learning algorithms achieve similar relative errors (with respect to the reference graph) and they all outperform the k-NN graph model. This is expected since the objective function of (3.11) is not fundamentally different from the one proposed in [Kal16]. The proposed algorithm also performs very well in the edge classification evaluations, with the soft and hard-thresholded F-measures: On two of the graph-based data types (with community graphs and Erdős-Rényi graphs), the proposed algorithm and the state-of-the-art algorithm [Kal16] outperform the k-NN graph model by large margins, and the proposed algorithm has an improvement over the state-of-the-art algorithm by several percentages, in terms of the soft-thresholded F-measure.

More interestingly, in the hard-thresholded F-measures, the proposed algorithm has significant improvements over the state-of-the-art algorithm; we observe that the result of the proposed algorithm (GL-SPH) has less false positives than that of [Kal16]. Since we tested both algorithms with the same number of trial parameter configurations, the improvement of the proposed algorithm (for the model (3.11)) is likely due to fact that it required less parameter trials than the model (3.8) [Kal16] in the parameter selection procedure.

3.3.2 Application to graph-dependent machine learning tasks

In this subsection, we apply the graph learning methods in several real-data applications. We focus on two well-known machine learning tasks that require the knowledge of a graph. First, we consider an application to graph-regularized

[1] See GSPbox [PPS+14].

Table 3.1: Evaluation of graph learning results on synthetic data

Graph type	Evaluation	k-NN	[Kal16]	GL-SPH
Community	Relative error	0.6490	**0.4590**	0.4861
	F-measure (hard)	0.7710	0.1967	**0.8860**
	F-measure (soft)	0.7710	0.8654	**0.8860**
Sensor	Relative Error	0.5919	0.3524	**0.3330**
	F-measure (hard)	**0.8209**	0.2894	0.7399
	F-measure (soft)	**0.8209**	0.5570	0.7399
Erdős-Rényi	Relative Error	1.1371	**0.6863**	0.7249
	F-measure (hard)	0.3497	0.1563	**0.7240**
	F-measure (soft)	0.3497	0.6874	**0.7240**

matrix completion [RYRD15, YRD16], where a graph Laplacian matrix is used as a regularizer to enhance the matrix completion performance. Second, we conduct semi-supervised learning [ZGL03, LWC12, HM14] on a dataset. In this application, we evaluate performances of semi-supervised learning based on several types of graphs, including graphs learned from the given dataset. In both applications, the quality of the graphs learned by the proposed algorithm is measured through the evaluation of the results given by the two graph-dependent tasks respectively.

Semi-supervised learning

In this subsection, we consider the semi-supervised learning problem [ZGL03, LWC12, HM14], which refers to the task of inferring the unknown *labels* of a set of data objects given the known labels of a few other objects. More precisely, assume that a set $\mathcal{V} = \{\mathcal{V}_i : i = 1, \ldots, m\}$ of m objects can be divided into a number of categories, the label of an object $\mathcal{V}_i$, denoted by f_i, refers to the categorical value in which $\mathcal{V}_i$ belongs to. In addition, each object $\mathcal{V}_i$ has some data attribute, or features, which can be represented by a vector $\phi_i \in \mathbb{R}^p$.

In the context of semi-supervised learning, one is faced with a collection of labeled and unlabeled objects $\mathcal{V} = \mathcal{V}_u \cap \mathcal{V}_\ell$, where the subscripts u and ℓ denote the index sets of the "unlabeled" and "labeled" objects respectively. The goal of semi-supervised learning is to infer the labels of the objects in $\mathcal{V}_u$. This is possible under the intuitive reason that objects with similar data features are more likely to have the same label than those that are very different. Therefore, the first step towards semi-supervised learning is to model the similarities of the objects using a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. This step consists in constructing a graph adjacency matrix $W \in \mathbb{R}^{m \times m}$ based on the features $\{\phi_i : i = 1, \ldots, m\}$. In our experiment, this step is similar to the methodologies (including graph learning) described in the previous subsection.

Subsequently, the inference of the unknown labels is realized by solving the

following problem,

$$\min_{y\in\mathbb{R}^m, y_{|\ell}=f_{|\ell}} \frac{1}{2}\sum_{i,j=1}^{m} W_{ij}(y_i-y_j)^2. \tag{3.21}$$

Note that, by the maximum principle of (discrete) harmonic functions [DS84, ZGL03], the problem (3.21) admits an unique solution, which is either a constant vector or a vector satisfying $f_j \in (0,1)$ for all $j \in \mathcal{V}_u$. Furthermore, through Proposition 2.3.3, the solution to (3.21) must satisfy $Ly = 0$ and $y_{|\ell} = f_{|\ell}$, where L is the graph Laplacian matrix associated with W. This entails that the solution y satisfies $y_j = \frac{1}{d_j}\sum_{i\sim j} W_{ij}y_i$, for all j, i.e., the label of any object is an weighted average of the labels of its neighbors. Hence, the solution to (3.21) admits the following expression,

$$\tilde{y}_{|u} = (D_{uu} - W_{uu})^{-1}W_{u\ell}f_{|\ell}, \tag{3.22}$$

where $D := \mathrm{Diag}(W\mathbf{1})$ and the matrix notations with subscripts, A_{uu} and $A_{u\ell}$ for any $A \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$, denote the submatrices $[A_{ij}]_{i\in\mathcal{V}_u, j\in\mathcal{V}_u}$ and $[A_{ij}]_{i\in\mathcal{V}_u, j\in\mathcal{V}_\ell}$ respectively.

Experimental results. We test semi-supervised learning on the USPS handwritten digits [Hul94]. The feature vector of each handwritten image $\mathcal{V}_i$ is the vectorization, $\phi_i \in \mathbb{R}^{256}$, of the gray-scale image itself, and the known labels $f = (f_i)_{i\in\mathcal{V}_\ell}$ $f_{|\ell}$ are sampled from the given labels (digit value in the handwritten image) of the images. We conduct semi-supervised learning on $|\mathcal{V}| = 500$ images containing all 10 classes ("0", "1",..., "9"), among which 20% of the images are labeled. The 10 classes are uniformly distributed in the set of known labels.

We build graphs from the features $\{\phi_i : i = 1,\ldots,|\mathcal{V}|\}$ of all images using the proposed algorithm (Algorithm 3.2.1) and two graph construction methods—the k-NN and the ϵ-NN graph (3.20) models; see Section 3.3.1.

Depending on the different choices of the parameters in each of the three methods: the regularization parameter α for the graph learning algorithm, k for the k-NN graph model and (ϵ,σ) for the ϵ-NN graph model (3.20), the sparsity level of the resulting graphs are different. A fair comparison methodology is to compare the graphs of these three methods according to a given sparsity level. The comparison of the graphs is made through the evaluation of their tests in the same semi-supervised learning problem. Therefore, in each of the obtained graph sparsity levels, three semi-supervised learning tests are conducted, and the evaluation of the label prediction results is measured by the misclassification rate, defined as $S(\tilde{y}_{|\mathcal{V}_u}) := \frac{\#\{\tilde{y}_j \neq f_j : j\in\mathcal{V}_u\}}{|\mathcal{V}_u|}$, where $\tilde{y}$ is the semi-supervised learning solution, according to (3.22), and f is the ground-truth labels of the images.

The comparative results are shown in Figure 3.3. We observe that the

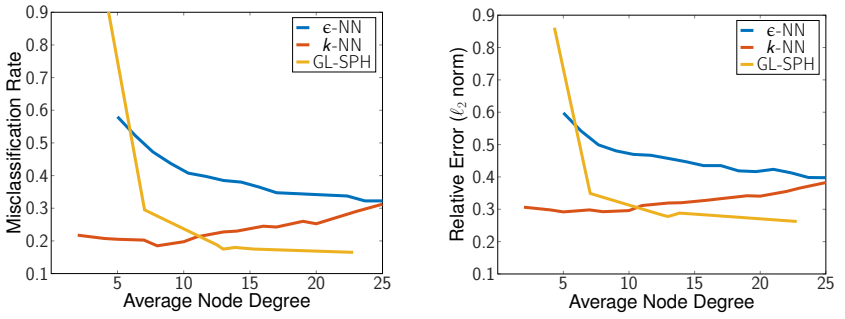


Figure 3.3: Misclassification rates of semi-supervised learning tests based on graphs given by (i) the k-NN model; (ii) the ϵ-NN graph model and (iii) the proposed graph learning algorithm (Algorithm 3.2.1). The x-axis is the average node degree of the graph.

classification results on graphs given by the proposed algorithm outperform the ϵ-NN graph model in most graph sparsity levels and they also outperform the k-NN graph model when the average node degree is larger than 10. Interestingly, this comparison reflects that graph learning is more flexible in terms of node degree distribution than the k-NN model.

Graph-regularized matrix completion

We illustrate the use of graphs for in an application to matrix completion. Given a data matrix $M^\star$ that is known only on a subset of its entries, matrix completion refers to the task of inferring the missing entries based on its known entries. Low-rank matrix algorithms and regularizations are among the most important topics in matrix completion. In brief, low-rank matrix completion can be formulated as an optimization problem with an objective function of the following form,

$$\min_{X \in \mathcal{M}} F(X) := f_\Omega(X) + \psi(X), \tag{3.23}$$

where $\Omega \subset \{1, \ldots, m\} \times \{1, \ldots, n\}$ is the index set of the known entries of $M^\star$, f_Ω is a data fitting function and ψ is a regularization function. The matrix candidate X in the formulation above is constrained in a set $\mathcal{M}$ of low-rank matrices. The interested reader is referred to the literature of low-rank matrix completion for topics related to the low-rank approach; also, the remaining chapters of this thesis also discuss optimization with low-rank matrices in matrix and tensor completion problems.

In this subsection, we focus on a graph-based regularization method, which has received much attention following the recent advances (e.g., [RYRD15, YRD16]) in this topic. In brief, the graph-based regularization refers to the use of the following penalty function,

$$\psi(X) = \mathrm{tr}(X^{\mathrm{T}} L X), \tag{3.24}$$

where X is the matrix completion candidate and L is a given graph Laplacian matrix, which is expected to be conform to the pairwise similarities among the entries of $M^\star$. Since the matrix $M^\star$ is not fully known, the construction of the graph Laplacian matrix L either requires certain side information—as is demonstrated in [RYRD15]—or some prior knowledge about the data matrix. In this experiment, we construct the graph Laplacian matrix using the proposed graph learning algorithm, based on an approximation of the data matrix $M^\star$ itself.

Specifically, as a starting point, we consider applying the regularizer (3.24) to the recent work [CA15] about robust matrix completion (RMC). RMC formulates the data fitting function f_Ω using the ℓ_1 norm-based loss function $\|P_\Omega(X - M)\|_1 := \sum_{(i,j)\in\Omega} |X_{ij} - M_{ij}|$. More precisely, [CA15] proposed to design f_Ω as a smooth approximation to the ℓ_1-norm based loss: $f_\Omega(X) = \sum_{(i,j)\in\Omega} \sqrt{\delta^2 + (X_{ij} - M_{ij})^2}$, for a small $\delta > 0$, and minimize

$$F(X) = \sum_{(i,j)\in\Omega} \sqrt{\delta^2 + (X_{ij} - M_{ij})^2} + \beta \|P_{\Omega^c}(X)\|_{\mathrm{F}}^2 \tag{3.25}$$

on the set of fixed-rank matrices $\mathcal{M}_r = \{X \in \mathbb{R}^{m\times n} : \operatorname{rank}(X) = r\}$. The reason for considering the smooth approximation of the ℓ_1-norm is twofold: (i) to make the data fitting function more robust to *outliers* in $P_\Omega(M^\star)$, as is the ℓ_1 norm-based data fitting function and (ii) to have a smooth and differentiable objective function at the same time. The Riemannian conjugate gradient descent algorithm (LRGeomCG [Van13]) is used to minimize $f_\Omega(X, M^\star)$ over the matrix manifold $\mathcal{M}_r$. We refer to [CA15] for more details about this problem and the algorithm.

By combining the graph-based penalty function (3.24) and the data fitting function (3.25), we have the following graph-regularized matrix completion problem,

$$\min_{X\in\mathcal{M}_r} F(X) := \sum_{(i,j)\in\Omega} \sqrt{\delta^2 + (X_{ij} - M_{ij})^2} + \gamma \operatorname{tr}(X^{\mathrm{T}} L X), \tag{3.26}$$

which can be solved by using the same Riemannian algorithm in [CA15]. Note that, for solving the graph-regularized problem (3.26), the computation of the gradient of the objective function is adjusted according to the additional regularization term (3.24). The graph Laplacian matrix L in the problem (3.26) is obtained via graph learning. More precisely, the graph-regularized matrix completion problem (3.26) is solved following the procedure described in Algorithm 3.3.1.

Experimental results. We conduct graph-regularized matrix completion tests on a PeMS traffic dataset from the UCI repository [Lic13, DG17]. The

Algorithm 3.3.1 Graph learning for graph-regularized matrix completion (GL-RMC)

Input: Ω, $P_\Omega(M^\star)$, rank value r, and parameters α, γ, β.
Output: Matrix completion result X.
1: Initialize X_0 via r-SVD$(P_\Omega(M^\star))$.
2: Get input matrix $\tilde{Z}$ from X_0 via (3.12)–(3.13).
3: Graph learning: $(W, L) \leftarrow$ GL-SPH$(\tilde{Z}, \alpha)$. # Algorithm 3.2.1
4: $X \leftarrow \mathrm{RMC}(P_\Omega(M^\star), \Omega, r, L, \beta, \gamma)$. # [CA15, Algorithm 1]

Traffic dataset[2] contains a $963 \times 10,560$ matrix of traffic occupancy rates (between 0 and 1) recorded by $m = 963$ sensors placed along different car lanes in the San Francisco bay area. The recordings are sampled every 10 minutes covering a period of 15 months. The column index set corresponds to the time domain and the row index set corresponds to the sensors, which is referred to as the spatial domain. We are interested in learning graphs in the spatial domain. Unlike data on social networks or any other kind with useful meta-data, there is no obvious way to find any side information, in the case of the Traffic dataset, that may help constructing a spatial-domain graph. This further motivates the application of graph learning instead of existing graph models.

In the following matrix completion experiment, the data matrix is only observed on a uniformly sampled index set Ω, given a sampling rate $p = \frac{|\Omega|}{mn}$. In addition to the graph-regularized approach (Algorithm 3.3.1), we also test the performance of a graph-agnostic method (a method that does not use any graph-based regularization term), using the original matrix completion algorithm (RMC) [CA15, Algorithm 1]). The regularization parameters involved in the graph learning and matrix completion algorithms, in Algorithm 3.3.1 and [CA15, Algorithm 1], are selected after performing a grid search for (α, β, γ) and β respectively.

Table 3.2: Matrix completion results on the PeMS Traffic data.

Evaluation	$\lvert\Omega\rvert/mn$	RMC[CA15]	GL-RMC
RMSE (on all/test entries)	4%	0.0641 / 0.0653	**0.0278** / **0.0281**
	6%	0.0476 / 0.0489	**0.0254** / **0.0256**
	8%	0.0402 / 0.0415	**0.0247** / **0.0249**
	10%	0.0394 / 0.0410	**0.0243** / **0.0245**
	20%	0.0285 / 0.0298	**0.0236** / **0.0237**
	30%	0.0259 / 0.0269	**0.0235** / **0.0236**
	40%	0.0278 / 0.0305	**0.0234** / **0.0236**

The matrix completion outputs are evaluated by the Root Mean Squared Error (RMSE) in comparison with the ground-truth matrix, on the test and the whole index sets. Table 3.2 shows the RMSE scores of the two matrix completion approaches under a number of sampling rates. We observe that for

[2] `https://archive.ics.uci.edu/ml/datasets/PEMS-SF`

all sampling rates, the graph-regularized matrix completion outperforms the graph-agnostic approach. The improvement in these matrix completion errors is more significant for small sampling rates.

More experiments and results. We make additional experiments, in the application of graph-regularized matrix completion, to compare the performances of graphs via graph learning with those obtained by a more traditional graph construction method.

We evaluate the matrix completion performances on the Traffic dataset, using two variants of Algorithm 3.3.1 (the graph learning and matrix completion procedure), where the input graph Laplacian L (Algorithm 3.3.1, line 4) is now the aforementioned Riemannian conjugate gradient algorithm LRGeomCG [Van13], with the graph-based regularizer (3.24) taken into account, and the graph construction step (Algorithm 3.3.1, line 3) is realized by either of the two following methods: (i) the proposed graph learning algorithm (GL-SPH, Algorithm 3.2.1) or (ii) the ϵ-NN graph model (a type of k-nearest-neighbor graph [CGS09]).

Table 3.3: Matrix completion results on the PeMS Traffic data. "Mean" refers to completing the matrix with the average of the known entries and "MC" refers to the model (3.23) with zero-valued regularization.

	$\lvert\Omega\rvert/mn$	Mean	MC	ϵNN-LRgeomCG	GL-LRgeomCG
RMSE (test)	4%	0.0452	0.0403	0.0316	**0.0312**
	6%	0.0453	0.0334	0.0313	**0.0291**
	8%	0.0452	0.0302	0.0310	**0.0282**
	10%	0.0452	0.0283	0.0308	**0.0271**
	20%	0.0452	0.0255	0.0291	**0.0249**
	30%	0.0452	0.0247	0.0277	**0.0240**
	40%	0.0452	0.0235	0.0265	**0.0233**

The results in test RMSEs are given in Table 3.3. We observe that (i) under all sampling rates tested, the graph-regularized matrix completion models outperform the naive imputations by mean values and the unregularized matrix completion model; and (ii) based on the same optimization algorithm (LRGeomCG), the model using graphs by graph learning (GL) has gained non-negligible improvements over the one using the ϵ-NN graphs. Note that both graph learning and ϵNN graph model require computations for a distance matrix, the extra running time needed for the optimization step of graph learning (by Algorithm 3.2.1) is worth the effort. Especially, in this matrix completion application with a relatively simple and limited parameter selection, it is encouraging to see improvements for all tested sampling rates.

3.4 Conclusion and discussion

In this chapter, we considered the problem of learning a graph matrix to model the pairwise similarities in a given data matrix. The main contribution is the fixed-scale graph learning algorithm. In the proposed fixed-scale approach, the graph learning problem is constrained in nonnegative orthant of the unit sphere. A projected gradient algorithm is proposed to solve this problem. Given a range of appropriate values for the problem parameter, we have observed fast convergence behaviors in experiments. In the experiments with synthetic data, we compared the graph learning performances of the proposed algorithm with one state-of-the-art algorithm and observed improvements in the results given by the proposed algorithm. In the applications to semi-supervised learning and graph-regularized matrix completion, we have observed that the graphs given by the graph learning algorithm resulted in improvements in both applications.

As mentioned in the problem statement, the graph learning problem depends on a given distance matrix. The definition of such a distance matrix calls for tasks such as feature engineering or proper representation of the nodes, which are as important as the optimization problem itself. Another feature of the current graph learning problem is that the computation of the distance matrix and the gradients of the graph learning objective function, both with a complexity of $\mathcal{O}\left(n^2\right)$ for a graph of n nodes, constitute a major bottleneck in the computational cost of graph learning. This limits its application to graph-dependent problems such as the aforementioned graph-regularized matrix completion. Therefore, future research directions to alleviate this computational issue include: (i) developing variant algorithms using sparsity-inducing heuristics and (ii) online learning techniques, i.e., accessing the distance matrix $\tilde{Z}$ incrementally instead of accessing it as a fully stored matrix. For related topics about scalable graph learning algorithms, we refer the interested reader to recent new advances in [KP17].

Chapter 4

Graph-regularized Matrix Completion

Low-rank matrix completion arises in applications such as recommender systems, forecasting and imputation of data; see [NKS19] for a recent survey. Given a data matrix with missing entries, the objective of matrix completion can be formulated as the minimization of an error function of a matrix variable with respect to the data matrix restricted to its revealed entries. In various applications, the data matrix either has a rank much lower than its dimensions or can be approximated by a low-rank matrix. As a consequence, restricting the rank of the matrix variable in the matrix completion objective not only corresponds to a reasonable assumption for successful recovery of the data matrix but also reduces the model complexity.

In certain situations, besides the low-rank constraint, it is useful to add a regularization term to the error function, in order to favor other properties related to the true data matrix. This regularization term can often be built from side information, that is, any information associated with row and column indices of the data matrix. Recent efforts in exploiting side information for matrix completion include inductive low-rank matrix completion [XJZ13, JD13, ZDG18] and graph-regularized matrix completion [ZSBS12, KBBV14, ZZHN15, RYRD15, YRD16]. In particular, graph-regularized matrix completion involves designing the regularization term using graph Laplacian matrices that encode pairwise similarities between the row and/or column entities (see Section 4.5.3 and Appendix 4.B). Depending on the application, the graph information is available through the connections between the data entities or can be inferred from the data itself.

Rao et al. [RYRD15] addressed the task of graph-regularized matrix com-

pletion by a low-rank factorization problem of the following form,

$$\min_{(G,H)\in\mathbb{R}^{m\times k}\times\mathbb{R}^{n\times k}} \frac{1}{2}\sum_{(i,j)\in\Omega}\left(\left(GH^T\right)_{ij} - M^\star_{ij}\right)^2 + \lambda_{\mathrm{r}}\,\mathrm{tr}(G^{\mathrm{T}}\Theta^{\mathrm{r}}G) + \lambda_{\mathrm{c}}\,\mathrm{tr}(H^{\mathrm{T}}\Theta^{\mathrm{c}}H), \tag{4.1}$$

where $M^\star \in \mathbb{R}^{m\times n}$ is the data matrix to be completed, k is the maximal rank of the low-rank model, Ω is the set of revealed entries and $\lambda_{\mathrm{r}} \geq 0$ and $\lambda_{\mathrm{c}} \geq 0$ are parameters. The matrices $\Theta^{\mathrm{r}} := I_m + L^{\mathrm{r}}$ and $\Theta^{\mathrm{c}} := I_n + L^{\mathrm{c}}$ with given graph Laplacian matrices $L^{\mathrm{r}} \in \mathbb{R}^{m\times m}$ and $L^{\mathrm{c}} \in \mathbb{R}^{n\times n}$. The graph Laplacian matrices L^{r} and L^{c} incorporate the pairwise correlations or similarities between the columns or rows of the data matrix $M^\star$. Indeed, observe that the graph Laplacian-based penalty terms in (4.1) in the form of $\mathrm{tr}(F^{\mathrm{T}}LF)$ can be written as

$$\mathrm{tr}(F^{\mathrm{T}}LF) = \sum_{i,j} W_{ij}\|F_{i,:} - F_{j,:}\|_2^2, \tag{4.2}$$

where W is the graph adjacency matrix such that $L = \mathrm{Diag}(W\mathbf{1}) - W$. The right hand-side term above suggests that the graph Laplacian-based penalty term promotes low-rank solutions that show pairwise similarities according to the given graph. Rao et al. [RYRD15] related the graph-based regularization term in (4.1) to a generalized nuclear norm (a weighted atomic norm [CRPW12]) and then found a close connection between the matrix factorization model (4.1) and a convex optimization formulation involving the generalized nuclear norm of the matrix variable $X = GH^{\mathrm{T}} \in \mathbb{R}^{m\times n}$. Moreover, they [RYRD15, §5] derived an error bound for the generalized nuclear-norm minimization problem, which can be smaller than that of the standard nuclear norm minimization problem if the graph Laplacian matrices are sufficiently informative with respect to the pairwise similarities between the columns/rows of $M^\star$. In this previous work, an instance (GRALS) of the alternating minimization method is developed for solving the problem (4.1).

In this chapter, we propose to solve the graph-regularized matrix factorization problem (4.1) by using Riemannian gradient descent and conjugate gradient methods. Our proposed algorithms are motivated by the following consideration. Optimization methods on the matrix product space $\mathbb{R}^{m\times k} \times \mathbb{R}^{n\times k}$ for matrix factorization models have been observed to efficiently provide good quality solutions to matrix completion problems, in spite of the nonconvexity of the cost function. Theoretical support for this observation can be found in [SL16, GJZ17, MWCC18]. Unlike alternating minimization (*e.g.* GRALS [RYRD15]), both Euclidean gradient descent and our proposed algorithms update the two matrix factors simultaneously, and do not require setting stopping criteria for subproblem solvers as in alternating minimization methods. Furthermore, by exploiting non-Euclidean geometries of the set of low-rank matrices in relation to the matrix product space $\mathbb{R}^{m\times k} \times \mathbb{R}^{n\times k}$, our algorithms use descent directions based on what can be seen as scaled gradients [MAS12, NS12] in the matrix product space. Moreover, as in [MAS12],

the particular structure of the objective function makes it possible to resort to exact line minimization along the descent direction. We show that the resulting gradient descent algorithms have an iteration complexity bound akin to the Euclidean gradient method (see Theorem 4.4.5), and that faster convergence behaviors are observed with these proposed algorithms, compared to their counterparts that use the Euclidean geometry (see Section 4.5).

We test the graph-regularized matrix completion model for matrix recovery tasks on both synthetic and real datasets. We compare our proposed algorithms with a state-of-the-art method (GRALS [RYRD15]), a baseline alternating minimization (AltMin) method and Euclidean gradient descent and conjugate gradient methods. We observe that the proposed algorithms enjoy faster or similar convergence behaviors compared to the state-of-the-art method and faster convergence behavior than the rest of the baseline methods tested. Moreover, the convergence behavior of the proposed algorithms is observed to be more robust against balancing issues that may arise with the asymmetric factorization model in (4.1), compared to their counterparts that use the Euclidean geometry. For completeness, we also compare empirically the graph-regularized matrix completion model with two low-rank matrix completion models: the matrix factorization model without regularization and the maximum-margin matrix factorization [SRJ05, RS05] in terms of recovery error. On both synthetic and real datasets, when the graph Laplacian matrices are properly constructed from features of the data matrix (with missing entries), the graph-regularized matrix completion model is found to yield solutions with superior recovery qualities compared to the other two models.

This chapter is based on the conference and journal papers [DAG19, DAG20].

4.1 Related Work

Matrix completion models. The graph-regularized matrix completion problem (4.1) is a generalization of the Maximum-Margin Matrix Factorization (MMMF) problem [SRJ05, RS05],

$$\min_{(G,H)\in\mathbb{R}^{m\times k}\times\mathbb{R}^{n\times k}} \frac{1}{2}\sum_{(i,j)\in\Omega}\left((GH^{\mathrm{T}})_{ij}-M^{\star}_{ij}\right)^2+\frac{\lambda}{2}\left(\|G\|_F^2+\|H\|_F^2\right). \tag{4.3}$$

The MMMF problem (4.3) is related to the nuclear norm-based [CR08, RFP10, CT10] convex program for low-rank matrix completion [MHE+10]

$$\min_{X\in\mathbb{R}^{m\times n}} \frac{1}{2}\sum_{(i,j)\in\Omega}\left(X_{ij}-M^{\star}_{ij}\right)^2+\lambda\|X\|_* \tag{4.4}$$

via the relation

$$\|X\|_*=\min_{G,H:GH^{\mathrm{T}}=X}\frac{1}{2}\left(\|G\|_F^2+\|H\|_F^2\right).$$

As shown by Hastie et al. [HMLZ15], any solution to (4.3) is also solution to the convex program (4.4), provided that $k \geq \operatorname{rank}(M^\star)$. Since the MMMF problem searches for a pair of matrix factors of rank smaller than or equal to k, which is usually much smaller than the matrix dimensions (m and n), its computational cost and memory requirements are significantly reduced compared to the nuclear norm-based convex program (4.4).

Optimization methods in related work. To the best of our knowledge, the state-of-the-art method for graph-regularized matrix completion is the AltMin-type algorithm GRALS [RYRD15]. For an alternating minimization method (*e.g.* [RYRD15]), one must deal with Sylvester-type equations for solving the subproblem with respect to the low-rank factor G (respectively H) when a graph-based regularization term $\operatorname{tr} G^{\mathrm{T}}\Theta^{\mathrm{r}}G$ (respectively $\operatorname{tr} H^{\mathrm{T}}\Theta^{\mathrm{c}}H$) appears in the objective function. Take the fully observed case in [RYRD15] for example, the subproblem of (4.1) with respect to the factor H corresponds to the following Sylvester equation

$$HG^{\mathrm{T}}G + \lambda_c\Theta^{\mathrm{c}}H = M^\star G. \tag{4.5}$$

In the matrix completion scenario, so solving the subproblem of (4.1) with respect to the factor H corresponds to solving an equation similar to (4.5), where the constant matrices involved in the equation depend on the positions of the revealed entries in Ω. Equivalently, the subproblem can be rewritten as a linear least-squares problem (in the form of (4.71) in Appendix 4.A.8) with respect to the vectorization of the factor H^{T} of dimension nk. The Hessian operator of this least-squares problem is not block diagonal due to the fact that the original subproblem corresponds to a Sylvester-type equation. Hence the least-squares problem of each alternating minimization step for (4.1) cannot be decomposed into m or n separate linear systems in dimension k. GRALS [RYRD15] approximately solves each of the two least-squares problems by using a linear conjugate gradient (CG) solver.

AltMin-type algorithms have proven to be very efficient in solving bi-convex problems, such as matrix factorization, nonnegative matrix factorization [WZ12, XY13a], dictionary learning [OF97, MBPS10], low-rank matrix completion, and in particular have also been proven to converge linearly for the low-rank matrix completion problem (without regularization) [JNS13a, Har14]. On the other hand, there are heuristic considerations with AltMin-like algorithms in practice. The parameters that control the stopping criteria of the solver for each alternating least squares problem determines the trade-off between the accuracy of the solution and the time efficiency of the AltMin method, but there is no apparent way to set them to achieve the best trade-off once and for all kinds of data. This can be seen in one of our experiments (see Figure 4.3). GRALS [RYRD15], as an instance of the AltMin method with well-tuned parameters and additional stopping criteria in its subproblem solvers, may suffer a significant drop in efficiency when certain properties of the data matrix change: a change of the "scale" of the data matrix, which can be measured by $\|M^\star\|_{\mathrm{F}}$

for example, changes significantly the performance of GRALS with a fixed set of stopping-criteria parameters.

4.2 Notation, definitions and problem statement

For $m \in \mathbb{N}^*$, we denote the set of integers $\{1, \ldots, m\}$ by $[\![m]\!]$. An undirected graph $\mathcal{G}$, which is determined by a set of nodes, $\mathcal{V}$, a set of (undirected) edges $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ and edge weights $W \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$, is denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, W)$. The graph adjacency matrix W is symmetric because $\mathcal{G}$ is undirected. In addition, we consider adjacency matrices with nonnegative coefficients:

$$W_{i_1,i_2} = W_{i_2,i_1} \begin{cases} > 0 & \text{if } (i_1, i_2) \in \mathcal{E} \\ = 0 & \text{otherwise.} \end{cases} \tag{4.6}$$

Throughout this chapter, the graph Laplacian matrix of a graph $\mathcal{G}$, denoted by L, is defined as

$$L = \operatorname{diag} W\mathbf{1} - W. \tag{4.7}$$

Thus we denote the graph as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, W)$ or $\mathcal{G} = (\mathcal{V}, \mathcal{E}, L)$. The graph Laplacian matrix defined by (4.6)–(4.7) is positive semi-definite (*e.g.* [Chu97, Spi12]). We denote by $\Lambda = \operatorname{Diag}\left(\lambda_1, \ldots, \lambda_{|\mathcal{V}|}\right)$ the diagonal matrix containing the eigenvalues of L in increasing order: $0 = \lambda_1 \leq, \ldots, \leq \lambda_{|\mathcal{V}|}$. For a matrix $M \in \mathbb{R}^{m\times n}$, we model the row index set of M by a graph $\mathcal{G}^{\mathrm{r}} = (\mathcal{V}^{\mathrm{r}}, \mathcal{E}^{\mathrm{r}}, L^{\mathrm{r}})$, where $\mathcal{V}^{\mathrm{r}} = [\![m]\!]$. The superscript r of $\mathcal{G}^{\mathrm{r}}$ signifies that the graph encodes row-wise correlations. Similarly, the graph that models the column-wise correlations of M is denoted as $\mathcal{G}^{\mathrm{c}} = (\mathcal{V}^{\mathrm{c}}, \mathcal{E}^{\mathrm{c}}, L^{\mathrm{c}})$. For a real-valued symmetric matrix Θ, the symbols $\lambda_{\max}(\Theta)$ and $\lambda_{\min}(\Theta)$ denote the largest and smallest eigenvalues. The notation $\Theta \succeq 0$ (respectively $\Theta \succ 0$) signifies that Θ is positive semi-definite (respectively positive definite). The largest and smallest singular values of a matrix X are denoted by $\sigma_{\max}(X)$ and $\sigma_{\min}(X)$ respectively. The Euclidean inner product and norm for the product space $\mathbb{R}^{m\times k} \times \mathbb{R}^{n\times k}$, denoted as $\langle\cdot,\cdot\rangle$ and $\|\cdot\|$ respectively, are defined as

$$\langle x, y\rangle = \operatorname{tr}(G_x^{\mathrm{T}} G_y) + \operatorname{tr}(H_x^{\mathrm{T}} H_y), \tag{4.8a}$$

$$\|x\| = \sqrt{\langle x, x\rangle}, \tag{4.8b}$$

for any pair of points $x = (G_x, H_x)$, $y = (G_y, H_y) \in \mathbb{R}^{m\times k} \times \mathbb{R}^{n\times k}$.

Problem statement. The purpose of this chapter is to solve (4.1), which we reformulate as

$$\min_{(G,H)\in\mathbb{R}^{m\times k}\times\mathbb{R}^{n\times k}} \frac{1}{2}\|P_\Omega(GH^{\mathrm{T}} - M^\star)\|_F^2 + \frac{\alpha}{2}\left(\operatorname{tr}(G^{\mathrm{T}}\Theta^{\mathrm{r}}G) + \operatorname{tr}(H^{\mathrm{T}}\Theta^{\mathrm{c}}H)\right), \tag{4.9}$$

where P_Ω is the projection onto the subspace of sparse matrices with nonzeros restricted to the index set Ω. The first term in (4.9) is an equivalent expression of the first term of (4.1). The second term corresponds to the other terms

of (4.1) with a parameterization that we find more convenient: we set λ_{r} and λ_{c} of (4.1) by one scalar α, and $\Theta^{\mathrm{r}} = I_m + \gamma_{\mathrm{r}} L^{\mathrm{r}}$ and $\Theta^{\mathrm{c}} = I_n + \gamma_{\mathrm{c}} L^{\mathrm{c}}$. This allows for more flexible settings of the weight of the Laplacian-based terms over the Frobenius norms. Setting $\gamma_{\mathrm{r}} = 0$ and $\gamma_{\mathrm{c}} = 0$ turns off the graph regularization, leaving us with the MMMF model (4.3). Setting $\alpha = 0$ turns off all regularization terms, leading to an unregularized matrix factorization problem

$$\min_{(G,H)\in\mathbb{R}^{m\times k}\times\mathbb{R}^{n\times k}} f_\Omega(G,H) := \frac{1}{2}\|P_\Omega(GH^{\mathrm{T}} - M^\star)\|_F^2. \tag{4.10}$$

The choice of the rank parameter k is *a priori* unknown for low-rank matrix approximation problems such as (4.9). Common approaches include model selection via cross-validation and rank adaptive methods [MMBS13, ZHG+16]. In this chapter, we focus on the setting where k is smaller than or equal to an optimal rank.[1] Observe that (4.9) is guaranteed to have a solution whenever $\alpha > 0$ since the objective function is continuous and coercive.

4.3 Optimization on $\mathbb{R}^{m\times k} \times \mathbb{R}^{n\times k}$

In this section, we introduce Riemannian gradient descent and conjugate gradient algorithms for problem (4.9). Since the search space $\mathbb{R}^{m\times k}\times\mathbb{R}^{n\times k}$ is just a vector space, these methods can be interpreted as preconditioned gradient methods. However, we call them "Riemannian" because the preconditioners are inspired from known Riemannian metrics on the set of rank-k m-by-n matrices, as we now explain.

In the main problem model (4.9), the product GH^{T} is an $m \times n$ matrix of rank smaller than or equal to k, and such matrices form the following nonlinear matrix space

$$\mathcal{M}_{\leq k} := \left\{X \in \mathbb{R}^{m\times n} :\ \mathrm{rank}(X) \leq k\right\}.$$

In particular, when the regularization parameter α in (4.9) reduces to 0, the model (4.9) can be directly identified with the following optimization problem on $\mathcal{M}_{\leq k}$ via the matrix factorization model $\mathbb{R}^{m\times k}\times\mathbb{R}^{n\times k} \mapsto \mathcal{M}_{\leq k} : (G,H) \mapsto X = GH^{\mathrm{T}}$,

$$\min_{X\in\mathcal{M}_{\leq k}} \frac{1}{2}\|P_\Omega(X - M^\star)\|_F^2. \tag{4.11}$$

We refer to the model (4.11) and (4.10) as the unregularized matrix completion model, in contrast to the graph-regularized model (4.9). A recent survey [UV20a] provides advances on optimization methods on the low-rank matrix space $\mathcal{M}_{\leq k}$.

[1] In real-world applications, it usually suffices to set up a rank value that is orders of magnitude smaller than m and n in order to work with an "underestimated" rank, that is, smaller than or equal to the optimal rank.

In contrast, when the regularization parameter α in (4.9) is nonzero, the model (4.9) does not induce an optimization problem on the Riemannian quotient manifold $\mathcal{M}_k$ of fixed-rank matrices. This is because the equivalence class of pairs (G, H) that represent the same X is $\{(GF^{\mathrm{T}}, HF^{-1}) : F \in \mathrm{GL}(k)\}$ and it is readily seen that the graph regularization term (4.9) is not constant on the equivalence classes. This does not prevent us from drawing inspiration from known Riemannian metrics on the set of rank-k matrices; see next.

Geometric elements on $\mathbb{R}^{m\times k} \times \mathbb{R}^{n\times k}$. The tangent space at a point $x \in \mathbb{R}^{m\times k} \times \mathbb{R}^{n\times k}$ is the Cartesian product of the tangent spaces of the two element matrix spaces: $T_x\left(\mathbb{R}^{m\times k} \times \mathbb{R}^{n\times k}\right) \simeq \mathbb{R}^{m\times k} \times \mathbb{R}^{n\times k}$.

Given two tangent vectors $\xi, \eta \in T_x\left(\mathbb{R}^{m\times k} \times \mathbb{R}^{n\times k}\right)$ at x, we consider the following two Riemannian metrics on $\mathbb{R}^{m\times k} \times \mathbb{R}^{n\times k}$,

◇ The *right-invariant* metric:

$$g_x(\xi, \eta) = \mathrm{tr}(\xi_G^{\mathrm{T}}\eta_G\big(G^{\mathrm{T}}G + \delta I_k\big)^{-1}) + \mathrm{tr}(\xi_H^{\mathrm{T}}\eta_H\left(H^{\mathrm{T}}H + \delta I_k\right)^{-1}); \quad (4.12)$$

◇ The *preconditioned* metric:

$$g_x(\xi, \eta) = \mathrm{tr}(\xi_G^{\mathrm{T}}\eta_G\left(H^{\mathrm{T}}H + \delta I_k\right)) + \mathrm{tr}(\xi_H^{\mathrm{T}}\eta_H\left(G^{\mathrm{T}}G + \delta I_k\right)), \quad (4.13)$$

where $\delta > 0$ is a constant parameter. Both (4.12) and (4.13), g_x are well-defined inner products. The condition $\delta > 0$ is required for (4.12) and (4.13) to remain well defined and positive definite when G or H does not have full column rank. Note that, if we set $\delta = 0$ and restrict the search space to the product space of full column-rank matrices $\mathbb{R}_*^{m\times k} \times \mathbb{R}_*^{n\times k}$, then (4.12) and (4.13) reduce to known metrics that induce metrics on the Riemannian quotient manifold $\mathcal{M}_k$. Specifically, (4.12) reduces to the right-invariant metric proposed in [MBS11, MMBS14], hence the name "right-invariant". The metric (4.13) reduces to a metric proposed by Mishra et al. [MAS12], which is specially adapted to the matrix factorization loss function (4.10) using the preconditioning technique. To see this, it suffices to note that the diagonal blocks of the Hessian (see Appendix 4.A.9) of f_Ω in (4.10) correspond to the following linear transformation

$$(\xi_G, \xi_H) \quad \mapsto \quad \left(P_\Omega(\xi_G H^{\mathrm{T}})H, P_\Omega(G{\xi_H}^{\mathrm{T}})^{\mathrm{T}}G\right). \quad (4.14)$$

Definition 4.3.1. *For a point $x \in \mathbb{R}^{m\times k} \times \mathbb{R}^{n\times k}$, the gradient of f at x is the unique vector in $T_x\left(\mathbb{R}^{m\times k} \times \mathbb{R}^{n\times k}\right)$, denoted as* $\mathrm{grad} f\,(x)$*, such that*

$$g_x(\xi, \mathrm{grad} f\,(x)) = Df(x)[\xi], \text{ for all } \xi \in T_x\left(\mathbb{R}^{m\times k} \times \mathbb{R}^{n\times k}\right). \quad (4.15)$$

Based on the metric (4.12) and Definition 4.3.1, the gradient $\mathrm{grad} f\,(x)$, denoted as QRIGHTINV, is

$$\mathrm{grad} f\,(G, H) = \left(\partial_G f\,(G, H)\,(G^{\mathrm{T}}G + \delta I_k), \partial_H f\,(G, H)\,(H^{\mathrm{T}}H + \delta I_k)\right). \quad (4.16)$$

Based on the metric (4.13) and Definition (4.3.1), the gradient of $\mathrm{grad}f(x)$, denoted as QPRECON, is

$$\mathrm{grad}f(x) = \left(\partial_G f(G,H)\left(H^{\mathrm{T}}H + \delta I_k\right)^{-1}, \partial_H f(G,H)\left(G^{\mathrm{T}}G + \delta I_k\right)^{-1}\right). \tag{4.17}$$

4.3.1 Algorithms

In this subsection, we introduce our algorithms and their elements such as computation of the gradient of the objective function of (4.9) and stepsize selection. We consider two basic algorithms (Algorithms 4.3.1 and 4.3.2) for optimization on $\mathbb{R}^{m\times k} \times \mathbb{R}^{n\times k}$. Computational details of these algorithms are given in Appendices 4.A.1–4.A.6.

Algorithm 4.3.1 Riemannian Gradient Descent (RGD)

Input: Function $f : \mathbb{R}^{m\times k} \times \mathbb{R}^{n\times k} \mapsto \mathbb{R}$, an initial point $x^0 \in \mathbb{R}^{m\times k} \times \mathbb{R}^{n\times k}$, and tolerance parameter $\epsilon > 0$.
Output: x^t.

1: $t \leftarrow 0$.
2: Compute gradient: $\mathrm{grad}f(x^t)$. # See (4.16) or (4.17)
3: **while** $\|\mathrm{grad}f(x^t)\| > \epsilon$ **do**
4: For $\eta^t = -\mathrm{grad}f(x^t)$, find stepsize s_t such that (s_t, η^t) satisfy (4.22) or (4.23)
See Algorithm 4.A.4 for (4.22)
5: Update: $x^{t+1} = x^t + s_t\eta^t$.
6: $t \leftarrow t + 1$.
7: Compute gradient: $\mathrm{grad}f(x^t)$.
8: **end while**

Initialization. A widely used initialization method is the so-called spectral initialization (*e.g.* [KMO10, KO09, SL16]) to construct the initial low-rank variable x^0. This consists of computing (U_0, S_0, V_0) by the k-SVD of $P_\Omega(M^\star)$, i.e., the matrix with all the unknown entries set to zero (the "zero-filled" matrix) and then defining the initial point $x^0 := (G^0, H^0)$ as follows,

$$(G^0, H^0) = (U_0 S_0^{1/2}, V_0 S_0^{1/2}). \tag{4.18}$$

In the case where the known entries are uniformly distributed, it is shown (e.g., [KMO10]) that this method provides an initial point that is close enough to the true hidden matrix. In other more general cases, more special initialization may be required. One alternative is to fill the unknown entries with the mean value of $P_\Omega(M^\star)$ before computing the k-SVD. Nevertheless, note that the direction that points from the zero-filled matrix to the mean value-filled matrix is likely to be highly aligned with the negative gradient of

Algorithm 4.3.2 Riemannian Conjugate Gradient (RCG)

Input: Function $f : \mathbb{R}^{m\times k} \times \mathbb{R}^{n\times k} \mapsto \mathbb{R}$, an initial point $x^0 \in \mathbb{R}^{m\times k} \times \mathbb{R}^{n\times k}$ and $\epsilon > 0$.
Output: x^t.

1: $t \leftarrow 0$.
2: Compute gradient: $\mathrm{grad} f\left(x^t\right), \eta^t = -\mathrm{grad} f\left(x^t\right)$. # See (4.16) or (4.17)
3: **while** $\|\mathrm{grad} f\left(x^t\right)\| > \epsilon$ **do**
4: Compute the conjugate descent direction η^t by (4.20)
See Algorithm 4.A.3
5: Find step size s_t such that (s_t, η^t) satisfy (4.22) or (4.23)
See Algorithm 4.A.4 for (4.22)
6: Update: $x^{t+1} = x^t + s_t\eta^t$.
7: $t \leftarrow t + 1$.
8: **end while**

$X \mapsto \frac{1}{2}\|P_\Omega(X - M^\star)\|_{\mathrm{F}}^2$, because the mean-valued matrix constitute a major rank-one improvement from the former. Hence, it is imaginable that the mean value-filled matrix be close to the optimization path (between the zero-filled matrix and the solution at termination), and the difference in the two instances with these two different initial points would be indistinguishable after several iterations.

Gradient and descent directions. The Euclidean gradient of the objective function of (4.9) at $x := (G, H) \in \mathbb{R}^{m\times k} \times \mathbb{R}^{n\times k}$ is computed as follows. We have

$$Df(x)[\xi] = \mathrm{tr}({\xi_G}^{\mathrm{T}} SH + {\xi_H}^{\mathrm{T}} S^{\mathrm{T}} G) + \alpha\left(\mathrm{tr}({\xi_G}^{\mathrm{T}}\Theta^{\mathrm{r}} G) + \mathrm{tr}({\xi_H}^{\mathrm{T}}\Theta^{\mathrm{c}} H)\right),$$

where $S = P_\Omega(GH^{\mathrm{T}} - M^\star)$. From the identity

$$Df(x)[\xi] = \mathrm{tr}({\xi_G}^{\mathrm{T}}\partial_G f\left(x\right)) + \mathrm{tr}({\xi_H}^{\mathrm{T}}\partial_H f\left(x\right)),$$

we deduce that the components of the Euclidean gradient $\nabla f(x)$ are

$$\partial_G f\left(x\right) = SH + \alpha\Theta^{\mathrm{r}}\, G, \tag{4.19a}$$
$$\partial_H f\left(x\right) = S^{\mathrm{T}} G + \alpha\Theta^{\mathrm{c}}\, H. \tag{4.19b}$$

Subsequently, the computation of the Riemannian gradient with respect to the metric (4.12) (respectively (4.13)) is based on (4.16) (respectively (4.17)) and (4.19). Algorithms 4.3.1–4.3.2 are later referred to as `Qrightinv RGD/RCG` and `Qprecon RGD/RCG` respectively. Detailed steps for these computations are given in Algorithm 4.A.1.

In Algorithm 4.3.1, the descent direction at iteration t is the negative gradient: $\eta^t = -\mathrm{grad} f\left(x^t\right)$. In Algorithm 4.3.2, the conjugate descent direction is

defined as

$$\eta^t = -\text{grad}f\left(x^t\right) + \beta_t \eta^{t-1} \tag{4.20}$$

for $t \geq 1$, where β_t is determined by a nonlinear CG rule, such as the Fletcher-Reeves rule

$$\beta_t = \frac{g_{x^t}\left(\text{grad}f\left(x^t\right), \text{grad}f\left(x^t\right)\right)}{g_{x^{t-1}}\left(\text{grad}f\left(x^{t-1}\right), \text{grad}f\left(x^{t-1}\right)\right)}, \tag{4.21}$$

depending on the local geometry of $\mathbb{R}^{m\times k} \times \mathbb{R}^{n\times k}$. A full description of the computation of the RCG descent directions is given in Appendix 4.A.1; see Algorithm 4.A.3.

Stepsize selection. In Algorithms 4.3.1–4.3.2, a stepsize s_t must be selected at each iteration for the update step along a descent direction $\eta^t \in T_{x^t}\left(\mathbb{R}^{m\times k} \times \mathbb{R}^{n\times k}\right)$. For this purpose, a common approach is to carry out a line search procedure by using backtracking with respect to the Armijo condition,

$$f(x^t) - f\left(x^t + s\eta^t\right) \geq \sigma s g_{x^t}\left(-\text{grad}f\left(x^t\right), \eta^t\right). \tag{4.22}$$

Alternatively, one can estimate the stepsize s_t via line minimization (*e.g.* [MAS12, Van13]): At a point $x \in \mathbb{R}^{m\times k} \times \mathbb{R}^{n\times k}$, we compute the stepsize defined as follows,

$$s_t^* = \underset{s\geq 0}{\arg\min}\, f(G + s\eta_G, H + s\eta_H), \tag{4.23}$$

for a given *descent* direction $\eta \in T_x\left(\mathbb{R}^{m\times k} \times \mathbb{R}^{n\times k}\right)$. We use the stepsize (4.23) for the numerical experiments in this chapter. The solution s_t^* to (4.23) is obtained by selecting the minimizer from the real positive roots of the derivative of the quartic function of (4.23), which is a polynomial of degree 3 and can be computed easily. The computational cost of this procedure is of the same order as the computation of the Riemannian gradient (4.16) or (4.17). Computational details of (4.23) are given in Appendix 4.A.4. In Section 4.4, we will show convergence properties of Algorithm 4.3.1 using the stepsize (4.23).

4.3.2 The regularization parameters

In this section, we discuss how to choose suitable values for the parameters of problem (4.9). Note that the model (4.9) with $\alpha > 0$ and at least one strictly positive value for γ_{r} and γ_{c} is referred to as a graph-regularized matrix completion (GRMC) model. When $\alpha > 0$ and $(\gamma_{\text{r}}, \gamma_{\text{c}}) = \mathbf{0}$, model (4.9) reduces to a MMMF model (4.3). When $(\alpha, \gamma_{\text{r}}, \gamma_{\text{c}}) = \mathbf{0}$, model (4.9) reduces to the unregularized matrix completion model (4.10), which is referred to as the MC model. Depending on the properties of the data (synthetic and real datasets), and for given graph Laplacian matrices $L^{\text{r}}, L^{\text{c}}$, we have two types of regularization schemes: (i) fixed parameter values and (ii) two-phase regularization scheme.

In the fixed-parameter scheme, we choose the parameter values following a standard cross validation procedure (see, *e.g.*, [JWHT13, §5.1.3]); see Ap-

pendix 4.C for details. We first generate a collection of parameter settings with random samples drawn from a range of values in $I_\alpha \times I_{\gamma_\mathrm{r}} \times I_{\gamma_\mathrm{c}} \subset \mathbb{R}^3_+$ with the uniform distribution in log scale, and then we select the parameter setting that has the best mean validation score (in terms of the RMSE (4.43) on the validation set). This regularization scheme is used later in Sections 4.5.2 and 4.5.3.

Algorithm 4.3.3 Two-phase matrix completion using graph-based regularization (2-phase GRMC)

Input: Parameter $\alpha > 0$; $\gamma_\mathrm{r} > 0$ and/or $\gamma_\mathrm{c} > 0$. Iteration budget $T, S > 0$ for the two phases. An initial point $x^0 \in \mathbb{R}^{m\times k} \times \mathbb{R}^{n\times k}$. A tolerance parameter (for Phase 2) $\epsilon > 0$.
Output: $x^* \in \mathbb{R}^{m\times k} \times \mathbb{R}^{n\times k}$.
1: Initialize with $x^0 \in \mathbb{R}^{m\times k} \times \mathbb{R}^{n\times k}$ using (4.18).
2: Phase 1: $x^*_\alpha =$ `GRMC Solver`(f_α, x^0, ∞) for at most T iterations. # See Algorithm 4.3.1 or 4.3.2.
3: Phase 2: start from x^*_α and find $x^* =$ `GRMC solver`$(f_\Omega, x^*_\alpha, \epsilon)$ for at most S iterations.

In the two-phase regularization scheme, shown in Algorithm 4.3.3, we set α and at least one parameter in $(\gamma_\mathrm{r}, \gamma_\mathrm{c})$ to strictly positive values for Phase 1; for Phase 2, all the regularization parameters are set to zero. In lines 2–3, the `GRMC Solver` is chosen from one of our proposed algorithms, such as `Qprecon RGD` (Algorithm 4.3.1 using the gradient (4.17)). In Phase 1, f_α denotes the objective function of (4.9) with the parameter value α. In Phase 2, the objective function reduces to f_Ω in (4.10). The parameters (for Phase 1) in Algorithm 4.3.3 are chosen in the same way as in the fixed-parameter scheme. This two-phase regularization scheme is designed for sample-efficient exact recovery of low-rank matrices and is used later in Section 4.5.2.

4.4 Convergence analysis

In this section, we analyze the convergence properties of Algorithm 4.3.1 with step sizes selected by line minimization (4.23). We conduct the analysis as follows: First, we show that the objective function of (4.9) is Lipschitz continuously differentiable in the search space $\mathbb{R}^{m\times k} \times \mathbb{R}^{n\times k}$ with respect to the Euclidean geometry. Second, we show that the specially designed non-Euclidean gradient descent directions, defined as QRIGHTINV (4.16) and QPRECON (4.17), ensure sufficient decrease in the function value provided the step sizes are chosen properly depending on the local geometry at each iterate. We show that the line minimization approach (4.23) finds such step sizes. Based on these results, we show the convergence behavior of the proposed RGD algorithm based on a generic convergence result given by Boumal et al. [BAC19]. We assume throughout this section that $\alpha > 0$ in (4.9). Recall that the inner product $\langle \cdot, \cdot \rangle$

and the norm $\|\cdot\|$ throughout this chapter are defined in (4.8) with respect to the Euclidean geometry on $\mathbb{R}^{m\times k}\times\mathbb{R}^{n\times k}$.

Note that the convergence rate estimation given later can be improved under more specific assumptions on the true hidden matrix $M^\star$ and the sample set Ω, which are used in recent works on statistical guarantees for exact recovery via matrix completion [SL16, GLM16, ZL16]. In [SL16], the local convergence behaviors of various algorithms (e.g. AltMin and gradient descent) are studied and the local convergence rate is proved to be linear under assumptions on Ω and $M^\star$. The settings of those matrix recovery results, however, do not apply directly to the graph-regularized problem of this chapter (due to the fact that the graph-based regularization term induces a bias in the solution vis-a-vis the training data $P_\Omega(M^\star)$). Therefore, we only study the global convergence without specific assumptions on Ω or $M^\star$.

In the presence of the regularization term, the Euclidean gradient ∇f in the sublevel set (with respect to a point $x^0\in\mathbb{R}^{m\times k}\times\mathbb{R}^{n\times k}$):

$$\mathcal{S}(x^0)=\left\{x\in\mathbb{R}^{m\times k}\times\mathbb{R}^{n\times k}: f(x)\le f(x^0)\right\} \tag{4.24}$$

is Lipschitz-continuous. We show this in the following lemma, along with a consequence known as the descent lemma.

Lemma 4.4.1. *Given a point $x^0\in\mathbb{R}^{m\times k}\times\mathbb{R}^{n\times k}$, the Euclidean gradient ∇f is Lipschitz continuous in the sublevel set $\mathcal{S}(x^0)$ (4.24). As a corollary, there exists a Lipschitz constant $L_0>0$ such that for any $x,y\in\mathcal{S}(x^0)$,*

$$f(y)-f(x)\le\langle\nabla f(x),y-x\rangle+\frac{L_0}{2}\|y-x\|^2. \tag{4.25}$$

Proof. The objective function of (4.9) is coercive since $\alpha>0$. Hence $\mathcal{S}(x^0)$ is bounded. Let B be a closed ball that contains $\mathcal{S}(x^0)$. Since f is C^∞, it follows that it is Lipschitz continuously differentiable in B. The result follows by a classical argument (see for example [Nes04, Lemma 1.2.3]). □

Based on Lemma 4.4.1, we get the following sufficient decrease property.

Lemma 4.4.2. *At any iterate $x^t=(G^t,H^t)$ produced by Algorithm 4.3.1 before termination, the following sufficient decrease property holds, provided that the step size s satisfies $0<s<2\Sigma_t/L_0$, for a strictly positive value $\Sigma_t>0$,*

$$f(x^{t+1})-f(x^t)\le-C_t(s)\|\mathrm{grad}f\left(x^t\right)\|^2, \tag{4.26}$$

where $C_t(s)=s(\Sigma_t-\frac{L_0 s}{2})>0$. Under the gradient setting QRIGHTINV *(4.16),*

$$\Sigma_t=\min\left(\frac{1}{\delta+\sigma_{\max}^2(G^t)},\frac{1}{\delta+\sigma_{\max}^2(H^t)}\right), \tag{4.27}$$

and under the gradient setting QPRECON (4.17)*,*

$$\Sigma_t = \delta + \min\left(\sigma_{\min}^2(G^t), \sigma_{\min}^2(H^t)\right). \tag{4.28}$$

Proof. At the t-th iteration in Algorithm 4.3.1, the descent direction is $\eta^t = -\mathrm{grad}f\,(x^t)$. Let $s > 0$ denote the step size for producing the next iterate: $x^{t+1} = x^t + s\eta^t$. In the gradient setting QPRECON where $\mathrm{grad}f\,(x)$ is defined by (4.17), the partial differentials are

$$\partial_G f(x) = \eta_G(H^{\mathrm{T}}H + \delta I_k) \text{ and } \partial_H f(x) = \eta_H(G^{\mathrm{T}}G + \delta I_k). \tag{4.29}$$

From Lemma 4.4.1, we have

$$f(x^{t+1}) - f(x^t) \leq \langle \nabla \bar{f}(x^t), x^{t+1} - x^t \rangle + \frac{L_0}{2}\|x^{t+1} - x^t\|^2 \tag{4.30}$$

$$\leq -s\|\eta^t\|^2 \left(\delta + \min\left(\sigma_{\min}^2(G^t), \sigma_{\min}^2(H^t)\right)\right) + \frac{L_0 s^2}{2}\|\eta^t\|^2 \tag{4.31}$$

$$= -C_t(s)\|\mathrm{grad}f\left(x^t\right)\|^2,$$

where $C_t(s) = s(\Sigma_t - \frac{L_0 s}{2})$ and $\Sigma_t = \delta + \min\left(\sigma_{\min}^2(G^t), \sigma_{\min}^2(H^t)\right)$. The inequality (4.31) is obtained by using (4.29) as follows,

$$\begin{aligned}
&\langle \nabla \bar{f}(x^t), x^{t+1} - x^t \rangle = -s\,\langle \nabla \bar{f}(x^t), \mathrm{grad}f\left(x^t\right)\rangle \\
&= -s\left(\mathrm{tr}\,\eta_G^{\mathrm{T}}\eta_G(H^{\mathrm{T}}H + \delta I_k) + \mathrm{tr}\,\eta_H^{\mathrm{T}}\eta_H(G^{\mathrm{T}}G + \delta I_k)\right) \\
&\leq -s\left(\delta\|\eta^t\|^2 + \sigma_{\min}^2(H)\|\eta_G\|_{\mathrm{F}}^2 + \sigma_{\min}^2(G)\|\eta_H\|_{\mathrm{F}}^2\right),
\end{aligned}$$

where the superscript of the element matrices $(G, H) = x^t$ and $(\eta_G, \eta_H) = \eta^t$ are omitted for brevity. Similarly, the same result applies to the gradient setting QPRECON (4.16), with the quantity Σ_t determined by (4.27). □

Next, we prove that Algorithm 4.3.1 with step sizes selected by line minimization (4.23) ensures sufficient decrease at each iteration.

Lemma 4.4.3. *The iterates produced by Algorithm 4.3.1, with step sizes selected by line minimization* (4.23)*, satisfy the following sufficient decrease property,*

$$f(x^{t+1}) - f(x^t) \leq -\left(\Sigma_t^2/2L_0\right)\|\mathrm{grad}f\left(x^t\right)\|^2. \tag{4.32}$$

Proof. In Algorithm 4.3.1, let $\eta = -\mathrm{grad}f\,(x^t)$ denote the Riemannian gradient descent direction at iteration $x^t \in \mathbb{R}^{m\times k} \times \mathbb{R}^{n\times k}$. From Lemma 4.4.1 and Lemma 4.4.2, we have

$$f(x^t + s\eta) \leq f(x^t) - C_t(s)\|\mathrm{grad}f\left(x^t\right)\|^2,$$

for $s \in [0, 2\Sigma_t/L_0]$, with Σ_t defined in (4.28) and (4.27). One the other hand, let $\bar{s}$ be the stepsize determined by (4.23), then by definition, the next iterate

$x^{t+1} = x^t + \bar{s}\eta$ is the minimum of f along the direction η: $f(x^{t+1}) \leq f(x^t + s\eta)$, for all $s \geq 0$. Hence

$$\begin{aligned} f(x^{t+1}) &\leq \min_{s\in[0,2\Sigma_t/L_0]} f(x^t + s\eta) && (4.33)\\ &\leq \min_{s\in[0,2\Sigma_t/L_0]} f(x^t) - C_t(s)\|\mathrm{grad} f\left(x^t\right)\|^2 && (4.34)\\ &= f(x^t) - \left(\Sigma_t^2/2L_0\right)\|\mathrm{grad} f\left(x^t\right)\|^2. && (4.35) \end{aligned}$$

□

In both Lemma 4.4.2 and Lemma 4.4.3, the sufficient decrease quantity depends on the local information Σ_t. The quantity Σ_t is useful only when it is a strictly positive number. We address this point in Proposition 4.4.4 for two gradient settings QRIGHTINV (4.16) and QPRECON (4.17).

Proposition 4.4.4. *Under the same settings as in Lemma 4.4.2 and 4.4.3, there exist positive numerical constants $\Sigma_* > 0$ such that the quantities Σ_t (4.27) and (4.28) are lower-bounded,*

$$\inf_{t\geq 0} \Sigma_t \geq \Sigma_*. \tag{4.36}$$

Proof. In the case of gradient setting QRIGHTINV (4.16),

$$\Sigma_t = \min\left(\frac{1}{\delta + \sigma_{\max}^2(G^t)}, \frac{1}{\delta + \sigma_{\max}^2(H^t)}\right).$$

First, we prove that there exists $D_0 > 0$ such that the norm of the iterate in $\mathbb{R}^{m\times k} \times \mathbb{R}^{n\times k}$ is bounded:

$$\|x^t\| \leq D_0 \tag{4.37}$$

for all $t \geq 0$. It suffices to note that the whole sequence $(x^t)_{t\geq 0}$ belongs to the sublevel set $\mathcal{S}^0$ (4.24) and that f is coercive. Second, for any $x = (G, H) \in \mathbb{R}^{m\times k} \times \mathbb{R}^{n\times k}$, the maximal singular values $\sigma_{\max}(G)$ and $\sigma_{\max}(H^t)$ are bounded by the norm as follows,

$$\|x^t\|^2 = \mathrm{tr}\, G^{\mathrm{T}}G + \mathrm{tr}\, H^{\mathrm{T}}H \geq \sigma_{\max}^2(G) + \sigma_{\max}^2(H). \tag{4.38}$$

The result (4.36) can be deduced by taking the numerical constant $\Sigma_* := 1/(\delta + D_0^2)$ and combining (4.37) and (4.38).

In the case of gradient setting QPRECON (4.17),

$$\Sigma_t = \delta + \min\left(\sigma_{\min}^2(G^t), \sigma_{\min}^2(H^t)\right) \geq \delta > 0,$$

and the result (4.36) can be ensured by $\Sigma_* := \delta$. □

We are now ready to conclude in a manner similar to that in [BAC19, Theorem 2.5]. A minor difference, however, is that the norm in [BAC19] is the

Riemannian norm, whereas our search space is a vector space and we use the Euclidean norm (4.8b). It is possible to use the Riemannian norms induced by (4.12) and (4.13) if the iterates stay on a fixed-rank manifold, but in all cases it suffices to use the Euclidean norm in the development of our results.

Theorem 4.4.5. *Under the problem statement* (4.9)*, for a given initial point* x^0 *and the gradient settings* QRIGHTINV (4.16) *and* QPRECON (4.17)*, the sequence generated by Algorithm 4.3.1 with the stepsize* (4.23) *converges and the decay of the gradient norm satisfies*

$$\|\mathrm{grad} f\left(x^N\right)\| \leq \sqrt{\frac{2L_0(f(x^0)-f^\star)}{\Sigma_* N}} \tag{4.39}$$

after N *iterations, where* $L_0 > 0$ *is the Lipschitz constant mentioned in Lemma 4.4.1, the numerical constant* $\Sigma_* > 0$ *is given in Proposition 4.4.4 and* $f^\star$ *is a lower bound*[2] *of the function value of* (4.9)*.*

Proof. The convergence of the sequence $(x^t)_{t\geq 0}$ is a direct result of the sufficient decrease property (4.26) in Lemma 4.4.2 and the boundedness of the sequence of function values $(f(x^t))_{t\geq 0}$. See Theorem 2.5 of [BAC19].

Let $N \geq 1$ denote the number of iterations needed for getting to an iterate x^N such that $\|\mathrm{grad} f\left(x^N\right)\| \leq \epsilon$, for a tolerance parameter $\epsilon > 0$.

Since our algorithm (Algorithm 4.3.1) does not terminate at $t \leq N-1$, the gradient norms $\|\mathrm{grad} f\left(x^t\right)\| > \epsilon$, for all $t \leq N-1$. By summing the right hand sides of (4.32) for $t = 0, \ldots, N-1$, we have

$$\begin{aligned} f(x^N) - f(x^0) &\leq -\sum_{t=0}^{N-1} (\Sigma_t^2/2L_0)\|\mathrm{grad} f\left(x^t\right)\|^2 && (4.40)\\ &\leq -(\epsilon^2/2L_0)\sum_{t=0}^{N-1}\Sigma_t^2 && (4.41)\\ &= -(\Sigma_*/2L_0)\,\epsilon^2 N && (4.42)\end{aligned}$$

Hence the number of iterations

$$N \leq \frac{2L_0(f(x^0)-f(x^N))}{\Sigma_*\epsilon^2} \leq \frac{2L_0(f(x^0)-f^\star)}{\Sigma_*\epsilon^2}.$$

In other words, the iterate produced by the algorithm after N iterations satisfies

$$\|\mathrm{grad} f\left(x^N\right)\| \leq \sqrt{\frac{2L_0(f(x^0)-f^\star)}{\Sigma_* N}}.$$

□

[2] One can take $f^\star := 0$ since the objective function of (4.9) is nonnegative.

4.5 Numerical Experiments

In this section, we evaluate the performance of the proposed algorithms for solving the graph-regularized matrix completion problem (4.9). The experimental tests are based on both synthetic and real-world datasets. The synthetic data are generated using a low-rank matrix model with graph information, and the graph Laplacian matrices underlying this graph-related data model are then used in the regularization term of (4.9). This synthetic experimental setting corresponds to an ideal situation where the graph Laplacian matrices in the regularization term are perfectly conform with the pairwise similarities between the matrix entries. In the tests on real-world data, a graph Laplacian matrix is constructed using basic graph proximity models based on pairwise distances between the rows (or columns) of an initial estimation of the data matrix.

4.5.1 Preliminaries

On both synthetic and real-world data, we compare the time efficiency of our proposed methods with several baseline methods: Euclidean gradient descent on the product space $\mathbb{R}^{m\times k} \times \mathbb{R}^{n\times k}$, alternating minimization and a state-of-the-art method GRALS [RYRD15] (also an alternating minimization method). Note that by *time efficiency*, we mean the amount of time that an iterative method takes to arrive at an iterate of a certain recovery accuracy, and this mainly depends on the convergence behavior and the computational cost per-iteration of the method. We also compare the graph-regularized (GRMC) model (4.9) with two other matrix completion models in terms of matrix recovery errors: (i) unregularized matrix completion (MC) via the factorization model (4.10) and (ii) the maximum-margin matrix factorization (MMMF) model (4.3). The following list gives detailed description of the methods involved in the experiments.

- ◇ `Qprecon RGD` and `Qprecon RCG` correspond to the Riemannian gradient (Algorithm 4.3.1) and conjugate gradient descent (Algorithm 4.3.2) using the gradient Qprecon (4.17). Similarly, `Qrightinv RGD` and `Qrightinv RCG` correspond to Algorithm 4.3.1 and Algorithm 4.3.2 using the gradient Qrightinv (4.16). By default, the step sizes in all these algorithms are selected via line minimization (4.23), with a label `(linemin)`. Since we focus on the application of (4.9) in the low rank setting, we set the rank parameter k by an underestimated value, that is, smaller or equal to the rank of the true hidden matrix, in all experiments. In such case, we set the parameter δ in the definition of Qprecon (4.17) and Qrightinv (4.16) to zero without any numerical issue (*e.g.* having rank deficient factor matrices). We show in Figure 4.10 (Appendix 4.A.5) that the convergence behavior of the so-tested algorithms is almost the same as their counterparts in the theoretical setting (with a presumably $\delta > 0$) analyzed in Section 4.4.

For consistency, `Euclidean GD` and `Euclidean CG` (see Appendix 4.A.7) stand for the gradient descent and nonlinear conjugate gradient descent algorithms respectively, in which the descent directions (such as the Euclidean gradient) are computed with respect to the Euclidean geometry on $\mathbb{R}^{m\times k} \times \mathbb{R}^{n\times k}$. The step sizes in all these algorithms are selected via line minimization (4.23).

◇ `AltMin`: An alternating minimization algorithm ((4.70a)–(4.70b)), where each of the two graph-regularized least-squares subproblems is solved by the linear CG routine Algorithm 4.A.6. `AltMin1` is an instance of `AltMin` that has accuracy parameter $\epsilon = 10^{-14}$ and $n_{\mathrm{CG}}^{\max} = 500$. `AltMin2` has accuracy parameter $\epsilon = 10^{-6}$ and $n_{\mathrm{CG}}^{\max} = 500$. Note that the parameter $n_{\mathrm{CG}}^{\max}$ can also be set to even larger values: Since each of the two subproblems in the alternating minimization are initialized with the latest iterate (warm-started), the number of iterations required for each subproblem solver to obtain a solution with an accuracy ϵ usually does not exceed $n_{\mathrm{CG}}^{\max}$ preset here. Hence, the active parameter for controlling the stopping behavior of the subproblem solvers in the experiments is ϵ.

◇ `GRALS`: An alternating minimization algorithm implemented by Rao et al. [RYRD15] that is available on line[3]. `GRALS1` denotes the GRALS algorithm with the accuracy parameter $\epsilon = 10^{-10}$ and $n_{\mathrm{CG}}^{\max} = 500$. GRALS differs from `AltMin` in that the linear CG solver for the subproblems (4.70a)–(4.70b) has an additional stopping criterion[4] compared to `AltMin`, which could trigger early termination and hence provide inexact solutions to the subproblems (4.70a)–(4.70b) under certain circumstances.[5]

To assess the approximation performance for the matrix completion task, we use the root mean-squared-error (RMSE). Given $M^\star \in \mathbb{R}^d$ and an index set $\Omega \subset [\![m]\!] \times [\![n]\!]$, the RMSE of $X \in \mathbb{R}^d$ on Ω is defined as

$$\mathrm{RMSE}\,(X;\Omega) = \sqrt{\sum_{(i,j)\in\Omega} (X_{ij} - M^\star_{ij})^2/|\Omega|}. \tag{4.43}$$

All numerical experiments are performed on a workstation with 8-core Intel Core i7-4790 CPUs and 32GB of memory running Ubuntu 16.04 and MATLAB R2015a. The source code is available on `https://gitlab.com/shuyudong.x11/grmc`.

[3]Link: `https://github.com/rofuyu/exp-grmf-nips15`.

[4]A stopping criterion that restricts the subproblem update to a region of radius depending on the norm of the partial gradients of f.

[5]This feature is one of reasons that make the convergence behavior of GRALS different from that of `AltMin`, and in several applications, faster than the latter.

4.5.2 Synthetic Data

We generate synthetic data with the following low-rank matrix model, which is a generalization of the model in [RYRD15, §5.1]. Let $\mathcal{G}^{\mathrm{r}} := (\mathcal{V}^{\mathrm{r}}, \mathcal{E}^{\mathrm{r}}, L^{\mathrm{r}})$ and $\mathcal{G}^{\mathrm{c}} := (\mathcal{V}^{\mathrm{c}}, \mathcal{E}^{\mathrm{c}}, L^{\mathrm{c}})$ be the graphs modeling the row-wise and column-wise similarities of $M^\star$ and let $(U^{\mathrm{r}}, \Lambda^{\mathrm{r}})$ (respectively $(U^{\mathrm{c}}, \Lambda^{\mathrm{c}})$) denote the pair of matrices containing the eigenvectors and associated eigenvalues of L^{r} (respectively L^{c}). A low-rank data matrix $X^\star$ is generated as follows,

$$Z^\star = F^\star Q^{\star T}, \tag{4.44a}$$
$$X^\star = A^{\mathrm{r}} Z^\star (A^{\mathrm{c}})^{\mathrm{T}}, \tag{4.44b}$$

where $(F^\star, Q^\star) \in \mathbb{R}^{m\times r^\star} \times \mathbb{R}^{n\times r^\star}$ are composed of columns that are *i.i.d.* Gaussian vectors and the matrices $A^{\mathrm{r}} \in \mathbb{R}^{m\times m}$ and $A^{\mathrm{c}} \in \mathbb{R}^{n\times n}$ are defined below with respect to a function $g : \mathbb{R} \mapsto \mathbb{R}$ acting element-wisely on a diagonal matrix:

$$A^{\mathrm{r}} = U^{\mathrm{r}} g(\Lambda^{\mathrm{r}}), A^{\mathrm{c}} = U^{\mathrm{c}} g(\Lambda^{\mathrm{c}}). \tag{4.45}$$

More precisely, $g(\Lambda) = \mathrm{Diag}\,(g(\lambda_1), \cdots, g(\lambda_m))$ for any diagonal matrix Λ. The function g in (4.45) enables one to control the way in which the graph information in L^{r} and L^{c} transforms the low-rank random Gaussian matrix $Z^\star$. In the literature of graph signal processing [SNF+13], the function g is referred to as a graph spectral filter, which is a graph analogue of filters in signal processing. In our experiments, the function g is

$$g(\lambda) = \begin{cases} \lambda^{-p} & \text{if } \lambda > 0, \\ 0 & \lambda = 0, \end{cases} \tag{4.46}$$

for $p \geq 1$. The spectral model (4.46) is a typical example of functions that are monotonically non-increasing over $\mathbb{R}_+^*$ and that have the effect of low-pass filters [SNF+13] in the graph spectral domain [SRV16]. Other examples include (i) the Tikhonov filter (*e.g.* [BMN04]) $g_\gamma(\lambda) = 1/\sqrt{1+\gamma\lambda}$, and (ii) the diffusion operator [CLL+05, CM06, ZH08] $g_\tau(\lambda) = e^{-\tau\lambda}$.

Remark 4.5.1. In order for model (4.44) to cover the graph-agnostic setting as a special case, we define by convention that $A^{\mathrm{r}} = I_m$ when $L^{\mathrm{r}} = \mathbf{0}$ and $A^{\mathrm{c}} = I_m$ when $L^{\mathrm{c}} = \mathbf{0}$.

The model (4.44) is of particular interest for experiments on synthetic data because it models a wide range of real data matrices whose entries present pairwise similarities: Due to the non-increasing nature of the function g on $(0, \infty)$ in (4.45), the transformations in (4.44b) with the matrices A^{r} and A^{c} return a data matrix $X^\star$ such that the graph-based regularization terms of (4.9) are reduced compared to that before the transformation (see Appendix 4.B for details). This translates to the observation that the entries of $X^\star$ present pairwise similarities that agree with the graph $(\mathcal{V}^{\mathrm{r}}, L^{\mathrm{r}})$ and/or $(\mathcal{V}^{\mathrm{c}}, L^{\mathrm{c}})$, unlike the structureless entries in $Z^\star$ (4.44a). Figure 4.1 shows the difference between

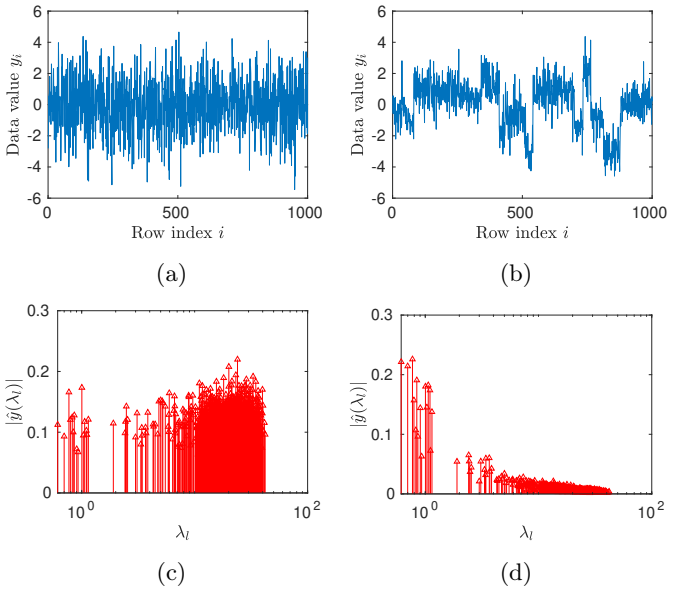


Figure 4.1: A data matrix Z from the Gaussian random model (4.44a) and $X = A^{\mathrm{r}}Z$ from the graph-based model (4.44b). The Laplacian matrix L^{r} involved in (4.44b) is generated with the prototypical graph model `Community` using GSPbox [PPS+14]. Comparison in the data entries and in the graph spectral domain. Top: A randomly chosen column (a): $y = Z_{:,j}$ and (b): $y' = X_{:,j}$. Bottom: Average amplitude of graph Fourier coefficients (c): $\widetilde{\mathbb{E}}|\widehat{y}(\lambda_l)|$ and (d): $\widetilde{\mathbb{E}}|\widehat{y'}(\lambda_l)|$ from low (small eigenvalue $\lambda_l(L^{\mathrm{r}})$) to high graph vertex frequencies.

$Z^\star$ and $X^\star := A^{\mathrm{r}}Z^\star$ regarding this property.

Matrix completion from noiseless observations

In this subsection, the ground-truth data matrix $M^\star$ is generated by (4.44) for $r^\star \ll \min(m,n)$ and is partially observed without any noise. The index of the revealed entries are *i.i.d.* sampled according to the Bernoulli model

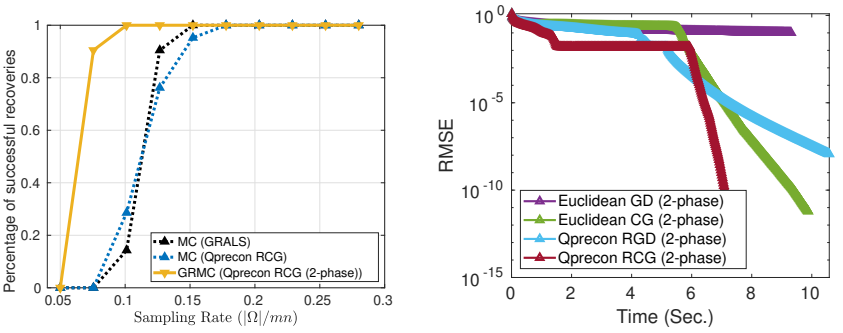
$$(i,j) \in \Omega \text{ with probability } \rho, \text{ for any } (i,j) \in [\![m]\!] \times [\![n]\!]. \tag{4.47}$$

For simplicity, we let $L^{\mathrm{c}} = 0$ such that $A^{\mathrm{c}} = I_n$ (see Remark 4.5.1). Hence the graph information is incorporated in $M^\star$ row-wisely by A^{r} with respect to (4.45). For this purpose, a graph Laplacian matrix L^{r} is generated with the prototypical graph model `Community` using the GSPbox [PPS+14]. The function g in this model is (4.46) with $p = 2$.

For the matrix completion model (4.9), we set the rank parameter by $k := \mathrm{rank}(M^\star)$. Note that in this case, $M^\star$ belongs to $\mathcal{M}_{\leq k}$, and any point $(G^*, H^*) \in \mathbb{R}^{m\times k} \times \mathbb{R}^{n\times k}$ such that $G^* H^{*T} = M^\star$ exactly recovers the hidden matrix $M^\star$. We refer to the search for such a point (G^*, H^*) as exact recovery

of the data matrix. In the literature of matrix completion, exact recovery of a low-rank matrix $M^\star$ by a factorization model such as (4.10) is possible under conditions on the extent of incoherence [CR09a] of the singular subspaces of $M^\star$ and the observation model Ω. Specifically, several sample complexity lower-bounds for $\rho \approx |\Omega|/mn$ are proved with both regularized ([SL16, GJZ17]) and unregularized (implicitly regularized [MWCC18]) matrix factorization models.

In the experiments of this subsection, we carry out tests for recovering the hidden matrix $M^\star$ with our proposed two-phase (2-phase GRMC) regularization scheme (Algorithm 4.3.3). Note that this 2-phase regularization scheme is specially adapted to the exact recovery of the hidden matrix $M^\star$ since it disables the regularization terms in its last phase (avoiding any bias in the solution). The unregularized matrix completion (MC) model (4.10), which corresponds to the special setting of (4.9) for $(\alpha, \gamma^{\mathrm{r}}, \gamma^{\mathrm{c}}) = \mathbf{0}$, is also tested. The label "MC (GRALS)" in Figure 4.2(a) corresponds to the result of unregularized matrix completion using `GRALS`, which reduces to a simple "ALS" algorithm since all regularization parameters are set to zero for the (unregularized) MC model.



(a) (b)

Figure 4.2: Matrix completion from noiseless observations. $M^\star$ is generated with non-trivial graph information with the model (4.46) and is partially observed without any noise. The rank parameter $k := \mathrm{rank}\,(X)\, M^\star$. (a): Percentage of successful recoveries under various sampling rates. The solutions are given by the two different matrix completion models (MC and GRMC). Matrix size $m = 500$, $n = 600$, rank $r^\star = 12$. (b): Results per iteration by 2-phase GRMC (Algorithm 4.3.3) at sampling rate $|\Omega|/mn = 10.0\%$: matrix size $m = 800$, $n = 900$, rank $r^\star = 12$.

First, we compare empirically the sample complexities of (i) the unregularized matrix completion model (MC) and (ii) the graph-regularized matrix completion model (4.9) through the 2-phase GRMC scheme described above. Under the experimental settings described in the beginning of this Section, for $m = 500$, $n = 600$ and $r^\star = 12$, we carry out repeated tests at various sampling rates $|\Omega|/mn$ ranging from 5% to 28%. At each sampling rate, we compute the percentage of successful recoveries among $N_{\mathrm{tests}} = 20$ repeated tests. Each test is counted as successful if the RMSE (4.43) on test entries

is smaller than 10^{-12}.[6] In particular, at each sampling rate, the parameters (α, γ_r) in the GRMC model (4.9) (for Phase 1 of Algorithm 4.3.3) are selected with respect to the test RMSE of the final solution, among `NCONFIGS` $= 5$ randomly generated parameter configurations (see the paragraph of the fixed-parameter scheme of Section 4.3.2 for details). Here the configurations for (α, γ_r) are generated with the uniform distribution (in the log scale) in the 2-dimensional box $[10^{-4}, 1] \times [10^{-2}, 5]$.[7]

As shown in the 2-phase GRMC scheme (Algorithm 4.3.3), the whole algorithm is stopped by either the accuracy parameter ϵ (see Algorithms 4.3.1, 4.3.2), when the iterate becomes an ϵ-stationary point or by the iteration budget parameter S, which is tuned to a sufficiently large value for both successful and unsuccessful recovery scenarios. Experimental results are shown in Figure 4.2(a): These results show empirically that the 2-phase GRMC method has a lower sample complexity than unregularized matrix factorization.

Second, we compare the time efficiency of the proposed algorithms with their counterparts under the Euclidean geometry. Figure 4.2(b) shows results per iteration under a sampling rate that is sufficiently large for 2-phase GRMC.

In particular, when the sampling rate ρ is sufficiently large, it is possible to exactly recover the hidden matrix $M^\star$ without any regularization (see Figure 4.2(a) at sampling rates larger than 15%). Therefore, we test our algorithms for this special case, with the problem parameter α set to zero in (4.9). In this special regime, we compare the time efficiency of our proposed algorithms with the several other methods in two different settings for the initialization point $x^0 \in \mathbb{R}^{m\times k} \times \mathbb{R}^{n\times k}$: In the first test, each method is initialized at a point $x^0 = (G_0, H_0)$ given by (4.18). In this case, the two factors G_0 and H_0 are *balanced*, in the sense that their matrix norms are equal. In the second test, we test the same methods with an unbalanced initial point

$$y^0 = (\lambda G_0, H_0/\lambda), \tag{4.48}$$

for $\lambda = 5$. The comparative results are given in Figure 4.3.

From the results in Figure 4.2 and Figure 4.3, we have the following observations:

- ◇ Our algorithms (`Qprecon RGD, RCG`) are faster than their Euclidean geometry-based counterparts (`Euclidean GD, CG`) in every experimental setting.
- ◇ Our algorithms are faster than the baseline alternating minimization methods `AltMin1, AltMin2`.
- ◇ `Qprecon RCG` is faster than `GRALS1` and this comparison becomes much

[6] This is an attainable accuracy level in the exact recovery scenario, based on preliminary tests.

[7] In the exact recovery scenario, the Phase 1 of our 2-phase algorithms does not need very fine-tuned parameters and the 2-dimensional box was also already narrowed after preliminary tests. A selection from 5 parameter configurations was enough to get the improvements shown in Figure 4.2(a).

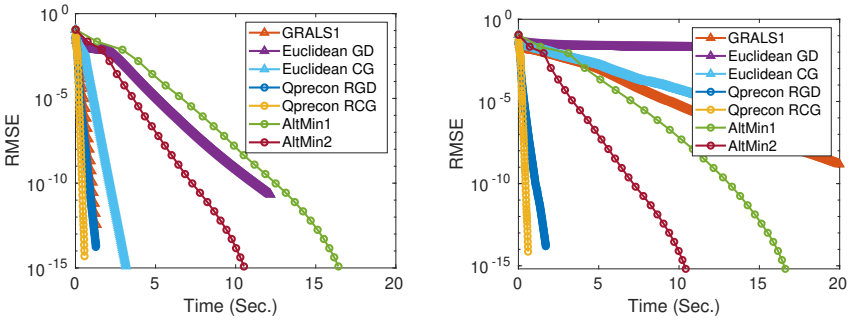


Figure 4.3: Results per iteration. Experimental settings: $m = 1000, n = 900, r = r^\star = 10, |\Omega|/mn = 20.0\%$. $M^\star$ is generated with the model (4.46) and is partially observed without any noise. (a): Each method is initialized at x^0 by (4.18). (b): Each method is initialized at y^0 by (4.48).

more evident when the initialization point is unbalanced than when it is balanced. Similarly, `Qprecon RGD` is as fast as `GRALS1` in the balanced initialization setting and much faster than the latter in the unbalanced case.

- ◇ In relation to the remark above, the baseline methods `Euclidean GD`, `Euclidean CG` and `GRALS1` are significantly slower when the initialization point is unbalanced.

Matrix completion from noisy observations

In this subsection, we assume that the partially observed data matrix $M^\star$ is composed of noisy observations from a low-rank matrix $X^\star$,

$$M^\star = X^\star + E, \tag{4.49}$$

where $E_{ij} \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sigma_N^2)$ for all $(i, j) \in [\![m]\!] \times [\![n]\!]$ and $X^\star \in \mathbb{R}^{m \times n}$ is generated using the model (4.44) with rank $r^\star \ll \min(m, n)$. For simplicity, we let $L^{\mathrm{c}} = 0$ and only incorporate row-wise similarities in $X^\star$ through $L^{\mathrm{r}} \in \mathbb{R}^{m \times m}$ with respect to (4.45). For this purpose, a graph Laplacian matrix L^{r} is generated with the prototypical graph model `Community` using the GSPbox [PPS+14]. The function g in this model is (4.46) with $p = 2$. Figure 4.4 shows the singular values of $M^\star$ generated with (4.49), where the true low-rank matrix $X^\star$ is generated with (4.44), and the noise level of E is determined by a signal-to-ratio parameter SNR $= 20$.

For the matrix completion problem (4.9), the rank parameter k is set to be smaller than $\mathrm{rank}(X^\star)$ and its values will be specified later. We test our algorithms, the baseline algorithms and the state-of-the-art algorithm GRALS

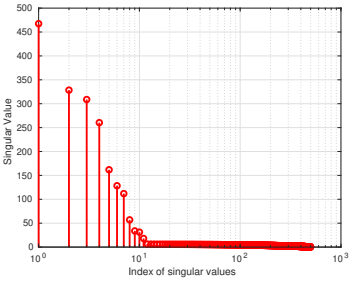


Figure 4.4: Singular values of $M^\star$ generated with (4.44) and (4.49), where the noise level is set by a signal-to-ratio parameter SNR $= 20$.

[RYRD15] for solving the problem (4.9) with fixed parameter values $\alpha \geq 0$ and $\gamma_{\rm r} \geq 0$.

First, we compare the recovery quality of the solutions to the three types of matrix completion models, that is, the unregularized matrix completion (MC), the graph-regularized matrix completion (GRMC) model (4.9) and the maximum-margin matrix completion (MMMF) in (4.3). For the unregularized problem model (MC), the parameter setting is $\alpha = 0$. For the MMMF model ($\alpha > 0$ and $\gamma_{\rm r} = 0$), the parameter setting is selected among a collection of $N_{\rm HP}$ randomly generated values in a reasonable range, $(\alpha^i)_{i=1,\ldots,N_{\rm HP}}$. For the GRMC model ($\alpha > 0$ and $\gamma_{\rm r} > 0$), the parameter setting is selected among a collection of `NCONFIGS` $= 10$ uniformly distributed (in log-scale) values in a 2-dimensional box $[10^{-4}, 1] \times [10^{-2}, 5]$. The criterion for the parameter selection is the test RMSE (4.43). At each sampling rate and once the parameters are selected for MMMF and GRMC, the recovery score of each of the three matrix completion models, using a given algorithm (ours as well as the baseline methods) corresponds to the average score after $N_{\rm tests}$ training instances based on $(M^\star, \Omega_s)_{s=1,\ldots,N_{\rm tests}}$, where the observation sets Ω_s are generated according to (4.47) with the given (fixed) sampling rate. All methods tested are stopped by either the accuracy parameter ϵ (see Algorithms 4.3.1, 4.3.2), when the iterate becomes an ϵ-stationary point or by an iteration budget parameter, which is set to a sufficiently large value for all methods. Figure 4.5 shows the recovery scores of each of the three problem models under different sampling rates. From Figure 4.5, we can see that at various sampling rates, the GRMC model (4.9) provide solutions with superior recovery qualities than the other two graph-agnostic models. Naturally enough, the improvement on recovery qualities via GRMC is significant at small sampling rates.

Second, we compare the time efficiency of the proposed algorithms with the baseline methods. The methods are tested in two slightly different experimental settings. Based on the data generation method described in the beginning of this subsection, the data matrix in each of these two experiments is rescaled with respect to a given constant value: We set the constant scalar such that

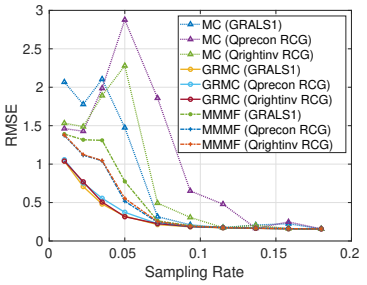


Figure 4.5: Average test RMSE of the three matrix completion models from noisy observations. $M^\star$ is generated with non-trivial graph information with the model (4.46) and is partially observed with additive noise (SNR = 20). The rank parameter $k = 10$. Recovery score of solutions to MC, MMMF and GRMC at various sampling rates. Matrix size: $m = 500$, $n = 600$, rank $r^\star = 12$. The sampling rate $|\Omega|/mn$ ranges from 1.0% to 18.0%.

$\mathbb{E}\left[|M^\star_{ij}|\right] = 1$ in the first setting and $\mathbb{E}\left[|M^\star_{ij}|\right] = 10^{-3}$ in the second one. Figure 4.6 shows the time efficiency of the tested methods in these two experiments in terms of the RMSE score per iteration. Note that the tests for producing Figure 4.5 are conducted under the data setting $\mathbb{E}\left[|M^\star_{ij}|\right] = 1$, without loss of generality. In particular, for the second data setting, with $\mathbb{E}\left[|M^\star_{ij}|\right] = 10^{-3}$, we show in Figure 4.7 the recovery qualities of the iterates under the unregularized (MC) and the graph-regularized (GRMC) models, for a relatively low sampling rate. Given the graph L^{r} underlying the synthetic data model (4.44), the GRMC model corresponds to one randomly generated set of parameters $(\alpha > 0, \gamma_{\mathrm{r}} > 0)$, where α is randomly generated in the range $(10^{-6}, 10^{-3})$ and γ_{r} randomly generated in the range $(10^{-2}, 5)$. We can see that for all the tested methods, the recovery qualities of the iterates under the GRMC model outperforms those of the unregularized matrix completion model.

From the results in Figure 4.6 and Figure 4.7, we have the following observations:

- ◇ Our algorithms (`Qprecon RGD, RCG`) are faster than their counterparts under the Euclidean geometry (`Euclidean GD, CG`).
- ◇ Our algorithms are faster than the baseline alternating minimization methods `AltMin1, AltMin2`.
- ◇ `Qprecon RCG` is either faster than `GRALS1` or as fast as the latter in various settings.
- ◇ In relation to the previous remark. The time efficiency of GRALS changes significantly when there is a simple change in the scale of the data matrix, as shown in Figure 4.6, since it has an additional stopping criterion that restricts the search of the solution to the least-squares subproblem (4.70a) (resp. (4.70b)) to a region of radius $\|\partial_G f(G^{t-1}, H^{t-1})\|$ (respectively $\|\partial_H f(G^t, H^t)\|$). This restricted-region criterion depends

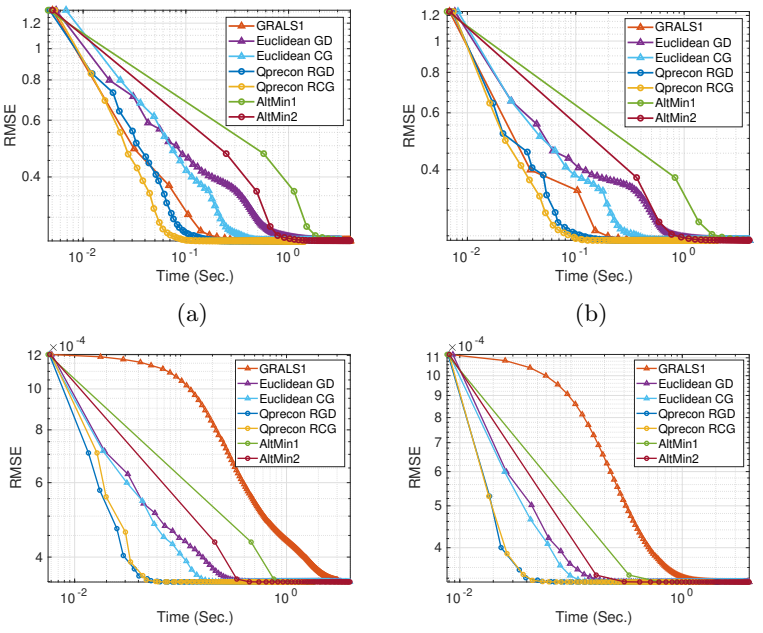


Figure 4.6: Test RMSE per-iteration on sythetic data. The data matrix $M^\star$ is generated via (4.44) and (4.49). Matrix size: $m = 1000$,$n = 900$, rank $r^\star = 12$. The rank parameter $k = 8$. Subplots (a-b): The data matrix $M^\star$ is rescaled by a scalar constant such that $\mathbb{E}\left[|M^\star_{ij}|\right] = 1$; (a): the sampling rate $|\Omega|/mn = 11.5\%$, (b): $|\Omega|/mn = 18.0\%$. Subplots (c-d): The data matrix $M^\star$ is rescaled by a scalar constant such that $\mathbb{E}\left[|M^\star_{ij}|\right] = 10^{-3}$; (c): the sampling rate $|\Omega|/mn = 11.5\%$, (d): $|\Omega|/mn = 18.0\%$.

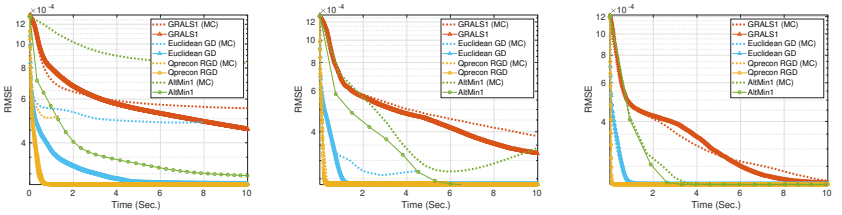


Figure 4.7: Test RMSE per iteration on synthetic data. The data matrix $M^\star$ is rescaled by a scalar constant such that $\mathbb{E}\left[|M^\star_{ij}|\right] = 10^{-3}$. Matrix size: $m = 1000$, $n = 900$, rank $r^\star = 12$. The rank parameter $k = 8$. (a)–(c): the sampling rate $|\Omega|/mn$ is 3.5%, 5% and 10% respectively. The dashed lines with a label "(MC)" corresponds to methods for solving the (unregularized) MC problem.

however, both on properties of the data matrix (such as the scale of the data) and the iterate.

- ◇ As illustrated in Figure 4.5, the recovery performances of all tested methods for GRMC are better and more stable than those for (unregularized) MC, when the sampling rate $|\Omega|/mn$ is insufficient; see Figure 4.7(a)–(b).

4.5.3 Real Data

In this subsection, we conduct experiments on real-world datasets. An essential difference between these experiments and experiments on synthetic data is that there is no reference graph associated with the data matrices in a real-world application. Since the data matrix in a real-world application often present pairwise similarities between its entries, we build the graphs L^{r} and L^{c} based on the given data before the graph-regularized matrix completion task. Subsequently, we conduct tests with the graph-regularized and graph-agnostic matrix completion models using all the methods involved, and compare their time efficiency. The real-world data used for these tests are from the PeMS Traffic occupancy and MovieLens datasets.

Methodology for graph construction

In the existing work on matrix completion using graph-based regularization, there are two main approaches to constructing the graph Laplacian matrices: (i) build the graph Laplacian matrix from the data $M^\star \in \mathbb{R}^d$ itself by using a certain graph node proximity model, *e.g.*, [KBBV14], and (ii) build a similarity graph L^{r} (and/or L^{c}) from side information [RYRD15, YRD16], that is, information related to the entities of the row (and/or column) indices of $M^\star$.

In our experiments, we adopt the first approach. Note that in [KBBV14], the computation of the graph proximity parameters is based on pairwise distances using only the revealed entries in $M^\star$. In contrast, we compute the graph proximity parameters based on a low-rank approximation of the partially revealed matrix. More precisely, we propose to use a rank-r approximation of M_0 as the features for constructing the graph. Let (U_0, S_0, V_0) denote the r-SVD of the zero-filled matrix $M_0 := P_\Omega(M^\star) \in \mathbb{R}^d$ and let $\widetilde{M}_0 := U_0 S_0 V_0^T$. Next, the computation of the graph edge weight parameters based on the given matrix $M := \widetilde{M}_0$ can be realized by using various node proximity methods such as K-Nearest Neighbors (K-NN) and ε-graph models [Cha83, BN03, HN04, CGS09], which boils down to computing a certain distance matrix between the rows (respectively columns) of M. Let $Z^{\mathrm{r}}(M) \in \mathbb{R}^{m\times m}$ denote the row-wise distance matrix of M defined as follows,

$$Z_{ij}(M) = d\left(M_{i,:}, M_{j,:}\right), \text{ for } i, j \in [\![m]\!], \tag{4.50}$$

where $d : \mathbb{R}^n \times \mathbb{R}^n \mapsto \mathbb{R}_+$ is a distance on the n-dimensional vector space. Subsequently, we build a Gaussian ε-graph by computing the node proximity

weights as follows

$$[W_\varepsilon(M)]_{ij} = \exp\left(-Z_{ij}(M)/\varepsilon^2\right), \text{ for } i,j \in [\![m]\!], \tag{4.51}$$

where $\varepsilon \in \mathbb{R}$ is a hyperparameter of the graph model. Furthermore, a sparse graph adjacency matrix is more preferable than a dense in a computational point of view, as the per-iteration cost for computing the gradient (as well as the function value) in (4.19) depends partly on $\mathrm{nnz}(L^{\mathrm{r}})$ and $\mathrm{nnz}(L^{\mathrm{c}})$, hence the sparsity of the row-wise (resp. column-wise) graphs. For simplicity, we sparsify the graph adjacency matrix defined in (4.51) by the following thresholding operation

$$[W_{\varepsilon,\sigma}(M)]_{ij} = \mathbf{1}_{\geq\sigma}\left(\exp\left(-Z_{ij}(M)/\varepsilon^2\right)\right), \text{ for } i,j \in [\![m]\!], \tag{4.52}$$

where $\mathbf{1}_{\geq\sigma}$ is the hard threshold function $\mathbf{1}_{\geq\sigma}(z) = \begin{cases} z & \text{if } z \geq \sigma \\ 0 & \text{otherwise.} \end{cases}$

In the graph model (4.52), the parameter ε is tuned according to the variance of $(Z_{ij})_{i,j=1,\ldots,m}$. In the following experiments, we set $\varepsilon := \mathrm{Var}((Z_{ij})_{ij})/5$ and find that this setting gives satisfactory improvements on the final recovery performances. The parameter σ is chosen according to a preset sparsity level $\mathfrak{s} \ll 1$ for the edge set associated with $W_{\epsilon,\sigma}$ such that $|\mathcal{E}(W_{\epsilon,\sigma})|/m^2 \leq \mathfrak{s}$. We set $\mathfrak{s} := 8\%$ and find that it is an appropriate trade-off between the amount of similarity graph edges and the additional computational cost required by matrix multiplications with the graph Laplacian.

The Traffic Data

The PeMS *Traffic* occupancy data[8] is a matrix with dimensions $963 \times 10\,560$ containing traffic occupancy rates (between 0 and 1) recorded across time by $m = 963$ sensors placed along different car lanes of the San Francisco Bay area freeways. The recordings are sampled every 10 minutes covering a period of 15 months. The column index set corresponds to the time domain and the row index set corresponds to geographical points (sensors), which are referred to as the spatial domain. Unlike the case with data from social networks or any other kind with useful meta-data, there is no straightforward way to find any side information for the *Traffic* dataset that may help constructing a spatial-domain graph. Hence we construct a sparse row-wise similarity graph with the Gaussian ε-graph model (4.52).

The parameter selection is similar to the methodology described in Section 4.5.2. In the following experiment, a set $\mathcal{H} = \{(\alpha^i, \gamma_{\mathrm{r}}^i)\}$ of `NCONFIGS`$= 20$ parameter configurations are sampled in a box region $[10^{-2}, 10] \times [10^{-2}, 2]$ with the uniform distribution (in the log scale). Then, for each sampling rate, the best choice in $\mathcal{H}$ is selected via the 5-fold cross validation (Appendix 4.C) and is shown in Table 4.1. Based on these parameters, we compare the matrix re-

[8] https://archive.ics.uci.edu/ml/datasets/PEMS-SF

covery qualities of GRMC and the other two graph-agnostic matrix completion models. The results are shown in Table 4.2.

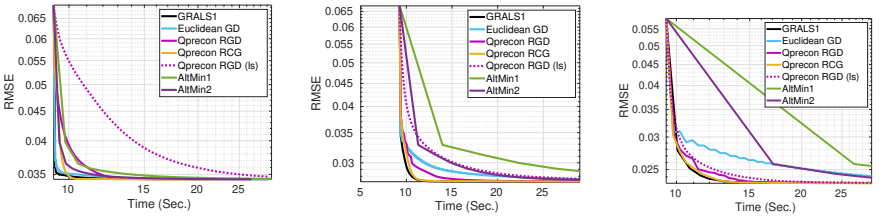
Table 4.1: GRMC tests on the Traffic dataset: selected parameters under sampling rates 1%, 5% and 20%.

	SR=1%		SR=5%		SR=20%	
Algorithm	α	$\alpha\gamma_{\mathrm{r}}$	α	$\alpha\gamma_{\mathrm{r}}$	α	$\alpha\gamma_{\mathrm{r}}$
GRALS1	2.90e-1	3.57e-3	3.74e-1	6.42e-3	6.95e-1	1.34e-1
AltMin1	2.90e-1	3.57e-3	3.74e-1	6.42e-3	6.95e-1	1.34e-1
Qprecon RGD	2.90e-1	3.57e-3	3.74e-1	6.42e-3	6.95e-1	1.34e-1
Qrightinv RGD	2.90e-1	3.57e-3	3.74e-1	6.42e-3	6.95e-1	1.34e-1

Table 4.2: Recovery scores (test RMSE) of the three matrix completion models under various sampling rates (SR): the unregularized matrix completion (MC), maximum-margin matrix factorization (MMMF) and graph-regularized matrix completion (GRMC).

	SR=1%			SR=5%			SR=20%		
Method	MC	MMMF	GRMC	MC	MMMF	GRMC	MC	MMMF	GRMC
GRALS1	0.0453	0.0351	0.0343	0.0778	0.0291	0.0272	0.0371	0.0246	0.0232
AltMin1	0.1218	0.0356	0.0344	0.0455	0.0332	0.0280	0.0317	0.0249	0.0244
AltMin2	0.1217	0.0352	0.0343	0.0455	0.0308	0.0275	0.0317	0.0247	0.0238
Euclidean GD	0.0455	0.0352	0.0343	0.0399	0.0299	0.0276	0.0261	0.0253	0.0241
Euclidean CG	0.0472	0.0350	0.0343	0.1443	0.0289	0.0272	0.0360	0.0246	0.0232
Qprecon RGD	0.0553	0.0351	0.0344	0.0477	0.0293	0.0273	0.0297	0.0246	0.0232
Qprecon RCG	0.0514	0.0350	0.0343	0.1875	0.0291	0.0271	0.0570	0.0246	0.0232
Qrightinv RGD	0.0454	0.0452	0.0349	0.0329	0.0326	0.0326	0.0300	0.0299	0.0300
Qrightinv RCG	0.0454	0.0452	0.0343	0.0812	0.0295	0.0273	0.0261	0.0254	0.0241

Figure 4.8 shows the time efficiency of the methods tested in terms of the RMSE score per iteration.



(a) (b) (c)

Figure 4.8: Test RMSE per iteration on the *Traffic* dataset ($m = 963$, $n = 10560$). The rank chosen is for the model (4.9) is $k = 18$. (a): the sampling rate $|\Omega|/mn = 1.0\%$, (b): $|\Omega|/mn = 5.0\%$, (c): $|\Omega|/mn = 20.0\%$. In particular, the label `(ls)` of the dashed line refers to the method using backtracking-Armijo line search (Algorithm 4.A.4).

MovieLens dataset

The *MovieLens 100K*[9] dataset [HK15] consists of 10 000 ratings (1 to 5) from 943 users on 1 682 movies. Each user has rated at least 20 movies. The data was collected through the MovieLens web site (`movielens.umn.edu`) during the seven-month period from September 19th, 1997 through April 22nd, 1998. This data has been cleaned up—users who had less than 20 ratings or did not have complete demographic information were removed from this data set. For the graph-regularized matrix completion model, we construct a sparse row-wise similarity graph (on the set of users) with the Gaussian ε-graph model (4.52).

Based on the same methodology for parameter selection using K-fold cross validation (with $K = 5$) as described in Section 4.5.2, we compare the matrix recovery qualities of GRMC and the other two graph-agnostic matrix completion models. The results are shown in Table 4.3. The RMSE scores of the GRMC model, returned by the methods tested using the selected parameter setting, are around 0.957, which is close to the RMSE score of 0.945 given by the graph-regularized method in [RYRD15] and is better than the scores of all other methods reported in [RYRD15]. Note that (i) the rank value chosen in the present experiment is the same as that in [RYRD15] and (ii) the graph Laplacian matrix used by Rao et al. [RYRD15] comes from side information, while the graph Laplacian matrix in the present experiment is constructed with the sparse ε-graph model (4.52), and (iii) in our experiment, the training set is 80% of the data entries in the ML100k dataset, while Rao et al. [RYRD15] used 90% of the available data. To achieve even better recovery scores under the GRMC framework, one needs to refine the construction of the graph Laplacian matrix either with models that are more adapted to the features of the data matrix or using more sensible user/movie-related information.

Table 4.3: Matrix completion score (RMSE on test entries) of solutions to the three types of problem models: unregularized matrix completion (MC), Maximum-margin matrix factorization and Graph-regularized matrix completion (GRMC).

Methods	MC	MMMF	GRMC
GRALS1	2.076	0.984	0.957
GRALS2	1.203	0.983	0.957
Euclidean CG	1.411	0.986	0.957
AltMin1	4.069	0.984	0.956
AltMin2	4.018	0.984	0.956
Qprecon RGD	1.083	0.986	0.957
Qprecon RCG	1.917	0.984	0.959

We also compare the time efficiency of the methods tested in terms of the RMSE score per iteration. Results are shown in Figure 4.9.

[9] https://grouplens.org/datasets/movielens/100k/

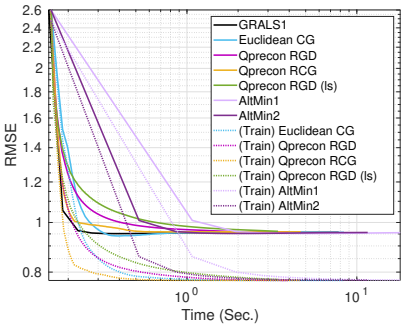


Figure 4.9: RMSE per iteration on the *MovieLens100k* dataset ($m = 943$, $n = 1682$). Rank parameter $k = 10$. The number of revealed entries is 80% of the $100k$ available ratings and the effective sampling rate $|\Omega|/mn \approx 5.05\%$. In particular, the label `(ls)` of the green line refers to the method using backtracking-Armijo line search (Algorithm 4.A.4).

Discussion of real-data experiments

From both Table 4.2 and Table 4.3, we observe that the matrix recovery quality of solutions to the GRMC model (4.9) is superior to those of the other two graph-agnostic matrix completion models. From Figure 4.8 and Figure 4.9, we have the following observations:

- ◇ Our algorithms (`Qprecon RGD, RCG`) are faster than `Euclidean GD` and the baseline alternating minimization methods `AltMin1, AltMin2`.
- ◇ The time efficiency of `Qprecon RCG` and the state-of-the-art method `GRALS1` are similar on the two real datasets tested. Observe that both `Qprecon RCG` and `GRALS1` are considerably faster than the two `AltMin` methods, though `GRALS1` and `AltMin` are based on the same alternating minimization strategy. This can be due to the programming language (C++ for `GRALS1` and MATLAB for `AltMin`) and to GRALS's above-mentioned additional stopping criterion for the subproblem solver.
- ◇ The stepsize by line minimization (4.23) yields faster convergence behavior than backtracking line search (with respect to the Armijo rule, starting from an arbitrary guess $s_0 = 1$ for the initial stepsize).

4.6 Conclusion

In this chapter, we focused on a graph-regularized matrix factorization problem for matrix completion. We proposed efficient algorithms for the underlying optimization problem on the product space $\mathbb{R}^{m\times k} \times \mathbb{R}^{n\times k}$. Our proposed gradient descent and conjugate gradient methods are based on specially designed Riemannian metrics on $\mathbb{R}^{m\times k} \times \mathbb{R}^{n\times k}$ that are inspired from metrics on the Riemannian quotient manifold of fixed-rank matrices. Moreover, we focused

on a stepsize selection method by exact line minimization, which results in a superior time efficiency compared to the approach using back-tracking line search. We provided rigorous theoretical analysis of the convergence property of the proposed Riemannian gradient descent algorithm.

We have investigated the matrix recovery qualities of various matrix completion models under various sampling rates: we found that the graph-based regularization does provide improvement for the matrix recovery quality compared to graph-agnostic matrix completion models, especially for relatively low sampling rates.

We have also conducted extensive experiments on synthetic data: we observed that our approach achieves significant speedup compared to several baseline methods, including a state-of-the-art method (GRALS) using alternating minimization, on various experimental settings. Moreover, we have shown via several tests that the proposed algorithms are much less influenced by changes in the initialization point or the scale of the data matrix. In our experiments on real-world data, we found that our methods produce solutions to the graph-regularized matrix completion model in comparable or less time than the baseline and the state-of-the-art methods.

Appendix

4.A Algorithms

4.A.1 Computation details of Algorithms 4.3.1–4.3.2 with line minimization

Algorithm 4.A.1 Computation of the Riemannian gradient

Input: $x = (G, H) \in \mathbb{R}^{m\times k} \times \mathbb{R}^{n\times k}, P_\Omega(M^\star) \in \mathbb{R}^d, \Omega, \Theta^{\mathrm{r}} \in \mathbb{R}^{m\times m}, \Theta^{\mathrm{c}} \in \mathbb{R}^{n\times n}$, and the parameter α.

Output: Riemannian gradient $\xi = (\xi_G, \xi_H) \in T_x\left(\mathbb{R}^{m\times k} \times \mathbb{R}^{n\times k}\right)$.

1: Compute the residual $S = P_\Omega\left(GH^{\mathrm{T}} - M^\star\right)$. # $(2k+1)|\Omega|$ flops

2: Compute

$$\partial_G f(x) = SH + \alpha\Theta^{\mathrm{r}}G, \quad \partial_H f(x) = S^{\mathrm{T}}G + \alpha\Theta^{\mathrm{c}}H.$$

$4(|\Omega| + \mathrm{nnz}(\Theta^{\mathrm{r}}) + \mathrm{nnz}(\Theta^{\mathrm{c}}))k$ flops

3: Compute

$$\xi = \left(\partial_G f(x)(G^TG + \delta I_k), \partial_H f(x)(H^TH + \delta I_k)\right) \ \textit{w.r.t.}\ (4.16)$$

$4(m+n)k^2$ flops

or

$$\xi = \left(\partial_G f(x)(H^TH + \delta I_k)^{-1}, \partial_H f(x)(G^TG + \delta I_k)^{-1}\right) \ \textit{w.r.t.}\ (4.17)$$

$4(m+n)k^2 + 2C_{\mathrm{chol}}k^3$ flops, see (4.53)

Computing the Riemannian gradient. Detailed steps and their respective computational costs for computing the Riemannian gradient are given in Algorithm 4.A.1. In the case of computing QPRECON (4.17): For the matrix inversion-related computations in the form of AB^{-1}, with $A := \partial_G f(x) \in \mathbb{R}^{m\times k}$ and $B := (G^TG + \delta I_k) \in \mathbb{R}^{k\times k}$, a typical approach is to first take (once) a Cholesky decomposition of B, whose cost is $C_{\mathrm{chol}}k^3$, and then compute the forward-and-backward substitution to get each of the m rows of AB^{-1}, which

costs $2mk^2$. In brief, the flop counts of this line consists of

- ◇ Computing $(G^{\mathrm{T}}G + \delta I_k)$ and $(H^{\mathrm{T}}H + \delta I_k)$: $2(m+n)k^2$ flops,
- ◇ Computing the Cholesky decomposition of $(G^{\mathrm{T}}G+\delta I_k)$ and $(H^{\mathrm{T}}H+\delta I_k)$: $2C_{\mathrm{chol}}k^3$, where $C_{\mathrm{chol}} = 1/3$.
- ◇ Forward-and-backward substitutions: $2(m+n)k^2$,

which sum to

$$4(m+n)k^2 + 2C_{\mathrm{chol}}k^3. \tag{4.53}$$

The dominant term in (4.53) is $4(m+n)k^2$ when $k \ll \min(m,n)$, which is the case for low-rank matrix approximation problems with a small rank parameter k and large data matrices.

The total number of flops needed for Algorithm 4.A.1 is either of the following

$$(6k+1)|\Omega| + 4\mathrm{nnz}\,(\Theta)\,k + 4(m+n)k^2, \tag{4.54a}$$
$$(6k+1)|\Omega| + 4\mathrm{nnz}\,(\Theta)\,k + 4(m+n)k^2 + 2C_{\mathrm{chol}}k^3, \tag{4.54b}$$

where (4.54a) is for computing QRIGHTINV (4.16) and (4.54b) is for computing QPRECON (4.17). The dominant cost in Algorithm 4.A.1 is for the computations in lines 1 and 2, which is

$$\mathcal{O}\left((|\Omega| + \mathrm{nnz}\,(\Theta))k\right),$$

where we use the term $\mathrm{nnz}\,(\Theta) := \mathrm{nnz}\,(\Theta^{\mathrm{r}}) + \mathrm{nnz}\,(\Theta^{\mathrm{c}})$ to denote the sum on the right-hand side, for simplicity. Indeed, when $k \ll \min(m,n)$ and the sampling rate ρ is of the order of 10%, the terms $(m+n)k \leq (m+n)k^2 \ll |\Omega|k = \rho mnk$.

Computing the cost function. For the algorithms with the line search procedure (Algorithm 4.A.4), the evaluation of the cost function is needed.

Algorithm 4.A.2 Computation of the cost function (4.9)

Input: $x = (G,H) \in \mathbb{R}^{m\times k} \times \mathbb{R}^{n\times k}, P_\Omega(M^\star) \in \mathbb{R}^d, \Omega, \Theta^{\mathrm{r}} \in \mathbb{R}^{m\times m}, \Theta^{\mathrm{c}} \in \mathbb{R}^{n\times n}$, and the parameter α.

Output: Function value $f(x)$ in (4.9).

1: Compute the residual $S = P_\Omega\left(GH^{\mathrm{T}} - M^\star\right)$. # $(2k+1)|\Omega|$ flops
2: Compute $f_\Omega(x) := \frac{1}{2}\|S\|_F^2$, # $2|\Omega|$ flops
3: Compute $\mathrm{Reg}(x) := \mathrm{tr}\, G^{\mathrm{T}}\Theta^{\mathrm{r}}G + \mathrm{tr}\, H^{\mathrm{T}}\Theta^{\mathrm{c}}H$, # $2\mathrm{nnz}\,(\Theta)\,k + 2(m+n)k$ flops
4: Return $f(x) = f_\Omega(x) + \frac{\alpha}{2}\mathrm{Reg}(x)$.

Hence, the cost for evaluating once the objective function (4.9) is

$$\mathrm{FLOPS}_{\mathrm{fobj}} = (2k+3)|\Omega| + 2\mathrm{nnz}\,(\Theta)\,k + 2(m+n)k. \tag{4.55}$$

To see the order of magnitude of the total cost: the dominant costs of Algorithm 4.A.2 are $\mathcal{O}\left((|\Omega| + \operatorname{nnz}(\Theta))k\right)$. The total cost of Algorithm 4.A.2 is $\mathcal{O}\left((|\Omega| + \operatorname{nnz}(\Theta))k\right)$.

Computing the conjugate gradient direction. The following Riemannian conjugate gradient schemes, labeled by PR, HS+ and FR respectively, are adapted from the classical nonlinear CG schemes—Polak–Ribière [PR69], Hestenes–Stiefel [HS52] and Fletcher–Reeves [FR64]—for computing the CG step parameter β_t in (4.20):

$$\text{(PR) } \beta = \max\left(0, \frac{g_{x^t}\left(\xi^t - \xi^{t-1}, \xi^t\right)}{g_{x^t}\left(\xi^{t-1}, \xi^{t-1}\right)}\right) \tag{4.56a}$$

$$\text{(HS+) } \beta = \max\left(0, \frac{g_{x^t}\left(\xi^t - \xi^{t-1}, \xi^t\right)}{g_{x^t}\left(\xi^t - \xi^{t-1}, \eta^{t-1}\right)}\right) \tag{4.56b}$$

$$\text{(FR) } \beta = \frac{g_{x^t}\left(\xi^t, \xi^t\right)}{g_{x^t}\left(\xi^{t-1}, \xi^{t-1}\right)}. \tag{4.56c}$$

A survey on nonlinear conjugate gradient can be found in [HZ06]. Implementation of these schemes (4.56) can be found in the Riemannian optimization toolbox Manopt [BMAS14]. In our experiments, we choose the modified Hestenes-Stiefel (HS+) rule. The flop counts for the HS+ rule is $5(m+n)k$.

Algorithm 4.A.3 Computation of the conjugate gradient direction

Input: iterates $x^{t-1}, x^t \in \mathbb{R}^{m\times k} \times \mathbb{R}^{n\times k}$, gradients $\xi^t \in T_x\left(\mathbb{R}^{m\times k} \times \mathbb{R}^{n\times k}\right)$, $\xi^{t-1} \in T_{x^{t-1}}\left(\mathbb{R}^{m\times k} \times \mathbb{R}^{n\times k}\right)$, previous CG direction $\eta^{t-1} \in T_{x^{t-1}}\left(\mathbb{R}^{m\times k} \times \mathbb{R}^{n\times k}\right)$.

Output: CG direction $\eta^t \in T_x\left(\mathbb{R}^{m\times k} \times \mathbb{R}^{n\times k}\right)$.

1: Compute: CG step parameter β_t with one of the schemes in (4.56) and then
$\eta^t = -\xi^t + \beta_t \eta^{t-1}$. # $7(m+n)k$ flops

2: Compute the angle between the CG direction and the gradient:
$\theta = \langle \eta^t, \xi^t \rangle / \|\eta^t\| \|\xi^t\|$. # $6(m+n)k$ flops

3: Reset to gradient if desired: $\eta^t = \xi^t$ if $\theta < 0.1$.

From Algorithm 4.A.3, the total flop counts for computing once the Riemannian CG direction, given two consecutive Riemannian gradients, is

$$13(m+n)k. \tag{4.57}$$

Computational cost of the line minimization (4.23)**.** This corresponds to the computations for $c_1, .., c_4$ in (4.62b)–(4.62d), which sums to $(6k+11)|\Omega| + 2\operatorname{nnz}(\Theta)\, k + 4(m+n)k$. Details are in Appendix 4.A.4.

4.A.2 Computational cost of RGD (linemin)

Each iteration of RGD (linemin) consists of (i) computing the stepsize by line minimization (4.23), the cost of which is in (4.63), (ii) conducting the descent step, the cost of which is $(m+n)k$ flops, (iii) computing the new gradient, the cost of which is in (4.54), and (iv) computing the norm of the gradient, the cost of which is $2(m+n)k$ flops. Note that since the step size s_t is obtained by (4.23), which guarantees a sufficient decrease, there is no need for any additional line search steps. Therefore, the total flop counts for one iteration of Algorithm 4.3.1 is

$$12(k+1)|\Omega| + 6\mathrm{nnz}\left(\Theta\right)k + (m+n)(4k^2+7k). \tag{4.58}$$

4.A.3 Computational cost of RCG (linemin)

`RCG (linemin)` needs to compute the nonlinear CG direction via Algorithm 4.A.3, and its flop counts is larger than that of RGD (linemin) (4.58) by exactly that in (4.57). The total cost is

$$12(k+1)|\Omega| + 6\mathrm{nnz}\left(\Theta\right)k + 4(m+n)(k^2+5k). \tag{4.59}$$

4.A.4 Stepsize computation via line minimization

Computing the stepsize (4.23) requires minimizing

$$f(G+s\eta_G, H+s\eta_H) - f(G,H)$$

for $s \geq 0$.

We have $f(G+s\eta_G, H+s\eta_H) - f(G,H) = A + B$, where

$$A = \frac{1}{2}\|P_\Omega\left(s(G\eta_H{}^{\mathrm{T}} + \eta_G H^{\mathrm{T}}) + s^2\eta_G\eta_H{}^{\mathrm{T}}\right)\|_F^2 + \\ \left\langle P_\Omega(GH^{\mathrm{T}} - M), P_\Omega(s(G\eta_H{}^{\mathrm{T}} + \eta_G H^{\mathrm{T}}) + s^2\eta_G\eta_H{}^{\mathrm{T}})\right\rangle \tag{4.60}$$

and

$$B = \frac{1}{2}\,\mathrm{tr}\Big(\left(sG^{\mathrm{T}}L_r\eta_G + s\eta_G{}^{\mathrm{T}}L_rG + s^2\eta_G{}^{\mathrm{T}}L_r\eta_G\right) + \\ \left(sH^{\mathrm{T}}L_c\eta_H + s\eta_H{}^{\mathrm{T}}L_cH + s^2\eta_H{}^{\mathrm{T}}L_c\eta_H\right)\Big). \tag{4.61}$$

These two equations lead to the following quartic polynomial form $A+B =$

$\sum_{j=1}^{4} c_j s^j$, where

$$
\begin{aligned}
c_1 &= \left\langle P_\Omega(GH^{\mathrm{T}} - M), P_\Omega(G\eta_H{}^{\mathrm{T}} + \eta_G H^{\mathrm{T}}) \right\rangle \\
&\quad + \operatorname{tr}(\eta_G{}^{\mathrm{T}} L_r G + \eta_H{}^{\mathrm{T}} L_c H), && \text{(4.62a)} \\
c_2 &= \frac{1}{2}\|P_\Omega(G\eta_H{}^{\mathrm{T}} + \eta_G H^{\mathrm{T}})\|_F^2 + \left\langle P_\Omega(GH^{\mathrm{T}} - M), P_\Omega(\eta_G \eta_H{}^{\mathrm{T}}) \right\rangle \\
&\quad + \frac{1}{2}\operatorname{tr}(\eta_G{}^{\mathrm{T}} L_r \eta_G + \eta_H{}^{\mathrm{T}} L_c \eta_H), && \text{(4.62b)} \\
c_3 &= \left\langle P_\Omega(G\eta_H{}^{\mathrm{T}} + \eta_G H^{\mathrm{T}}), P_\Omega(\eta_G \eta_H{}^{\mathrm{T}}) \right\rangle, && \text{(4.62c)} \\
c_4 &= \frac{1}{2}\|P_\Omega(\eta_G \eta_H{}^{\mathrm{T}})\|_F^2. && \text{(4.62d)}
\end{aligned}
$$

The solution to s^* is selected from the real positive roots of the derivative of this quartic function, which is the polynomial of degree 3, $(A+B)'(s) = \sum_{j=1}^{4} c_j s^{j-1}$, the roots of which are easily computed.

Computational costs. In Algorithms 4.3.1–4.3.2, whenever the line minimization (4.23) is required, it always follows the computation of a Riemannian gradient, during which we have stored the following intermediate matrices (i) $S = P_\Omega(GH^{\mathrm{T}} - M^\star) \in \mathbb{R}^{m\times n}$ and (ii) $\Theta^{\mathrm{r}} G \in \mathbb{R}^{m\times k}, \Theta^{\mathrm{c}} H \in \mathbb{R}^{n\times k}$. Hence, in the following list of flop counts, the computations related to the items above need not be counted:

- ◇ For c_1 in (4.62b): $(4k+3)|\Omega| + 2(m+n)k$ flops. Information stored:[10] $P_\Omega(G\eta_H{}^{\mathrm{T}})$ and $P_\Omega(\eta_G H^{\mathrm{T}})$.
- ◇ For c_2 in (4.62b): $(2k+4)|\Omega| + 2\mathrm{nnz}(\Theta)\, k + 2(m+n)k$ flops. Information stored: $P_\Omega(\eta_G \eta_H{}^{\mathrm{T}})$.
- ◇ For c_3 in (4.62c): $2|\Omega|$ flops.
- ◇ For c_4 in (4.62d): $2|\Omega|$ flops.

These sum up to

$$(6k+11)|\Omega| + 2\mathrm{nnz}(\Theta)\, k + 4(m+n)k. \tag{4.63}$$

4.A.5 The constant parameter δ in the definition of gradients

In all the experiments in Section 4.5, the gradients defined in (4.16) or (4.17) are used with a parameter $\delta = 0$. In this setting, the underlying metric (4.12) is not guaranteed to be positive definite and the metric (4.17) is not always well-defined at any iterate $x \in \mathbb{R}^{m\times k} \times \mathbb{R}^{n\times k}$. The convergence analysis does not cover the case where $\delta = 0$ in (4.16)–(4.17). Nevertheless, we note that

[10] The information is stored only inside the current iteration.

the convergence behavior of our proposed algorithms so far tested agrees with the theoretical results presented in Section 4.4. In fact, in all our experimental settings, the problem parameters $(\alpha, \gamma_{\mathrm{r}}, \gamma_{\mathrm{c}})$ and the largest rank value k are chosen properly, especially that the rank parameter k is set to an underestimated value. Therefore, we did not observe any singularity in all the results presented in Section 4.5.

Figure 4.10 shows that the iterative results of the proposed RGD algorithms using a strictly positive δ (for δ set to 10^{-4}) and those using $\delta = 0$ are almost the same. The experimental setting for this illustration is the same as in Section 4.5.2.

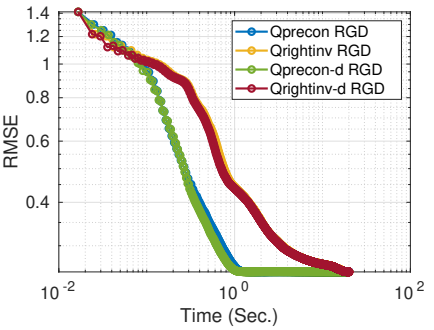


Figure 4.10: RMSE per iteration on synthetic data. The data matrix $M^\star$ is generated via (4.44) and (4.49): Matrix size $m = 1000$, $n = 900$, rank $r^\star = 12$. The rank parameter $k = 8$. The sampling rate $|\Omega|/mn = 5\%$. The results returned by the RGD algorithm using gradients defined by the two metrics with $\delta = 10^{-4}$ (labeled with "-d") and $\delta = 0$ respectively. The convergence behaviors of the algorithms with $\delta = 0$ and with a small $\delta > 0$ are almost same.

4.A.6 Computation details of Algorithms 4.3.1–4.3.2 with Armijo line-search

The line search procedure by backtracking with respect to the Armijo rule (4.22) is given in Algorithm 4.A.4.

Algorithm 4.A.4 Armijo line search

Input: $f : \mathcal{M} \mapsto \mathbb{R}$, a descent direction a retraction $\mathcal{R}$ on $\mathcal{M}$, $x^t \in \mathcal{M}$, initial stepsize $s_t^0 > 0$ and σ, $\beta \in]0, 1[$.

Output: s.

1: Initialize: $s = s_t^0$.
2: **while** $f(x^t) - f(\mathcal{R}_{x^t}(s\eta^t)) < \sigma s \left\langle -\mathrm{grad} f\left(x^t\right), \eta^t \right\rangle$ **do**
3: $\quad s \leftarrow \beta s$.
4: **end while**

Computational cost of `RGD (Armijo)`. `RGD (Armijo)` corresponds to Algorithm 4.3.1 using the backtracking line search with the Armijo condition at each iteration. Computing once the function value and the Riemannian gradient at the same time costs in total

$$\begin{aligned} & \mathrm{FLOPS}_{\mathrm{fobj}} + \mathrm{FLOPS}_{\mathrm{gradf}} - \left[\mathrm{FLOPS}(P_\Omega(GH^{\mathrm{T}})) + \mathrm{FLOPS}(\Theta^{\mathrm{r}}G, \Theta^{\mathrm{c}}H)\right] \\ = \; & (6k+4)|\Omega| + 4\mathrm{nnz}\left(\Theta\right)k + 4(m+n)k^2 + 2(m+n)k. \end{aligned} \tag{4.64}$$

Hence, the computational cost of the t-th iteration is

$$(6k+4)|\Omega| + 4\mathrm{nnz}\left(\Theta\right)k + 4(m+n)k^2 + 2(m+n)k + n_t^{\mathrm{LS}}\mathrm{FLOPS}_{\mathrm{fobj}}, \tag{4.65}$$

where $\mathrm{FLOPS}_{\mathrm{fobj}}$ is defined in (4.55).

Flop counts for `RCG (Armijo)`. `RCG lsArmijo` (Algorithm 4.3.2, with stepsizes chosen via the Armijo line search) needs to compute the nonlinear CG direction via Algorithm 4.A.3, and its per-iteration cost is larger than that of `RGD (Armijo)` by the amount of (4.57). In sum, it is

$$(6k+4)|\Omega| + 4\mathrm{nnz}\left(\Theta\right)k + 12(m+n)k^2 + 17(m+n)k + n_t^{\mathrm{LS}}\mathrm{FLOPS}_{\mathrm{fobj}}. \tag{4.66}$$

for an iteration $t \geq 0$.

4.A.7 Algorithms using the Euclidean gradient

The cost for computing the Euclidean gradient is smaller than the cost of computing the variable metric gradient by the cost of line 3 of Algorithm 4.A.1. Therefore, the computational cost per-iteration of `Euclidean GD (linemin)` is smaller than (4.58) by exactly $4(m+n)k^2$, which is

$$12(k+1)|\Omega| + 6\mathrm{nnz}\left(\Theta\right)k + 7(m+n)k. \tag{4.67}$$

Computing the Euclidean CG step, using the same rule for computing the CG directions, requires the same cost as by (4.57), hence it equals

$$13(m+n)k. \tag{4.68}$$

As a consequence, the computational cost per-iteration of `Euclidean CG (linemin)` is larger than (4.67) by exactly that of (4.68), which is

$$12(k+1)|\Omega| + 6\mathrm{nnz}\left(\Theta\right)k + 20(m+n)k. \tag{4.69}$$

4.A.8 Computation details in GRALS [RYRD15]

This subsection contains a description of the algorithm proposed by Rao et al. [RYRD15] and a detailed list of flop counts for the standard CG steps

involved in this algorithm. GRALS consists of the following two alternating least squares procedures

$$G^t = \arg\min_G f(G, H^{t-1}), \tag{4.70a}$$

$$H^t = \arg\min_H f(G^t, H), \tag{4.70b}$$

for a given initial point.

The quadratic forms of the least-squares systems (4.70a)–(4.70b) have the following structures. The subproblem (4.70a) is a least-squares problem whose objective $f(G) := f(G, H^t)$ can be rewritten as a quadratic form of the vectorization of G^{T} via the identification $f(G) = \tilde{f}_1(\mathrm{vec}(G^{\mathrm{T}}))$,

$$\min_s \tilde{f}_1(s) = \frac{1}{2} s^{\mathrm{T}} A^{(1)} s - \mathrm{vec}(H^{tT} P_\Omega(M^\star)^{\mathrm{T}})^{\mathrm{T}} s, \tag{4.71}$$

where $A^{(1)} \in \mathbb{R}^{km \times km}$ has the following structure,

$$A^{(1)} = \bar{B}^{(1)} + \alpha \Theta^{\mathrm{r}} \otimes I_k, \tag{4.72}$$

where $\bar{B}^{(1)} \in \mathbb{R}^{km \times km}$ is block diagonal with m diagonal blocks $(B_i^{(1)})_{i=1,\ldots,m}$ of size $k \times k$ such that

$$B_i^{(1)} = \sum_{j \in \Omega_i} h_j {h_j}^{\mathrm{T}}, \tag{4.73}$$

for $i \in [\![m]\!]$. Here the index sets $\Omega_i := \{j \in [\![n]\!] : (i,j) \in \Omega\}$ and

$$h_j = [H_{j1}, .., H_{jk}]^{\mathrm{T}} \in \mathbb{R}^k \tag{4.74}$$

is the transpose of the j-th row of H.

Algorithm 4.A.5 Hessian-vector multiplication $A^{(1)}s$ [RYRD15]

Input: Data (known on Ω) $P_\Omega(M^\star) \in \mathbb{R}^{m \times n}, \Omega \subset [\![m]\!] \times [\![n]\!]$. Quadratic form $A^{(1)}$ in (4.72). Vector $s := \mathrm{vec}(G^{\mathrm{T}}) \in \mathbb{R}^{k \times m}$. Laplacian-based matrix $\bar{\Theta}$.
Output: $A^{(1)}s$.

1: **for** $i = 1, .., m$ **do**
2: Get $g_i := [G_{i1}.., G_{ik}]^{\mathrm{T}}$ from s (vectorization of G^{T}).
3: Compute $\tilde{g}_i = \sum_{j \in \Omega_i} h_j ({h_j}^{\mathrm{T}} g_i)$. # See (4.73).
4: **end for**
5: Get G from the vectorization $s = \mathrm{vec}\left(G^{\mathrm{T}}\right)$ and compute $\tilde{G} = \bar{\Theta} G$.
6: Return: $\mathrm{vec}([\tilde{g}_1, .., \tilde{g}_m]) + \mathrm{vec}(\tilde{G}^{\mathrm{T}})$.

Similarly, the subproblem (4.70b) can be solved by the same routines (Algorithm 4.A.5 and 4.A.6) as for (4.70a) by swapping the roles of G and H (and matrices in the regularization terms) in all computations of matrices involved. In the implementation of GRALS, the linear CG routine is used to solve the

two subproblems in the form of (4.71). Algorithm 4.A.6 is a standard CG descent procedure with $A \in \mathbb{R}^{q\times q}$, $b \in \mathbb{R}^q$ as inputs and x^0 as the initial point. Note that GRALS [RYRD15] uses a warm-start scheme: x^0 corresponds to latest iterate G^{t-1} (resp. H^{t-1}) for the t-th step (4.70a) (resp. (4.70b)).

The Hessian-vector multiplication in the linear CG iteration (Algorithm 4.A.6, line 13) is computed via Algorithm 4.A.5.

Algorithm 4.A.6 CG Algorithm for solving (4.70b) (resp. (4.70a))

Input: $A \in \mathbb{R}^{q\times q}$, for $q = nk$ (resp. mk), initial point $x^0 \in \mathbb{R}^q$. Accuracy parameter ϵ, iteration budget n_{CG}.
Output: $x^\star \in \mathbb{R}^q$, $n^\star_{\mathrm{CG}}$

1: Compute: $b = \mathrm{vec}(P_\Omega(M^\star)^{\mathrm{T}}G) \in \mathbb{R}^q$ (resp. $b = \mathrm{vec}(P_\Omega(M^\star)H)$). # $2|\Omega|k$ flops
2: $r_0 = b - Ax^0$.
3: **for** $k = 0, .., n_{\mathrm{CG}}$ **do**
4: Compute: $\|r_k\|$. # $2nk$ (resp. $2mk$) flops
5: **if** $\|r_k\| \le \epsilon\|r_0\|$ **then**
6: Break;
7: **end if**
8: **if** $k = 0$ **then**
9: $p_1 = r_0$.
10: **else**
11: $p_{k+1} = r_k + \frac{\|r_k\|^2}{\|r_{k-1}\|^2}p_k$. # $2nk$ (resp. $2mk$) flops
12: **end if**
13: Compute: $v_{k+1} = Ap_{k+1}$. # $2(|\Omega| + \mathrm{nnz}\,(\Theta))k$ flops
14: Compute: $\beta = \frac{\|r_k\|^2}{p_{k+1}^{\mathrm{T}}v_{k+1}}$. # $2nk$ (resp. $2mk$) flops
15: Compute: $x_{k+1} = x_k + \beta p_{k+1}$, $r_{k+1} = r_k - \beta v_{k+1}$. # $4nk$ (resp. $4mk$) flops
16: **end for**
17: Return $x^\star = x^k, n^\star_{\mathrm{CG}} = k$.

Computational cost of `GRALS`. The number of flops required by Algorithm 4.A.6 is:

$$2(n^\star_{\mathrm{CG}} + 1)|\Omega|k + 2n^\star_{\mathrm{CG}}\mathrm{nnz}\,(\Theta)\,k + 10n^\star_{\mathrm{CG}}nk$$
$$(\text{ resp. } 2(n^\star_{\mathrm{CG}} + 1)|\Omega|k + 2n^\star_{\mathrm{CG}}\mathrm{nnz}\,(\Theta)\,k + 10n^\star_{\mathrm{CG}}mk).$$

During the t-th iteration in GRALS, let n_t^H (respectively n_t^G) denote the number of CG iterations (*i.e.* $n^\star_{\mathrm{CG}}$ returned by this algorithm) required by Algorithm 4.A.6 for solving the subproblem (4.70b) (resp. (4.70a)) at iteration t. Then the number of flops required by GRALS to complete the t-th iteration,

from (G^t, H^t) to (G^{t+1}, H^{t+1}) is

$$2(n_t^G + n_t^H + 2)|\Omega|k + 2(n_t^G + n_t^H)\mathrm{nnz}\,(\Theta)\,k + 10(n_t^G m + n_t^H n)k. \qquad (4.75)$$

Figure 4.11 shows the RMSE per iteration, where the x-axis is represented either by the wall time recorded at each iteration or the cumulative cost (in flops) required by the main computational steps in each of the algorithms at each iteration.

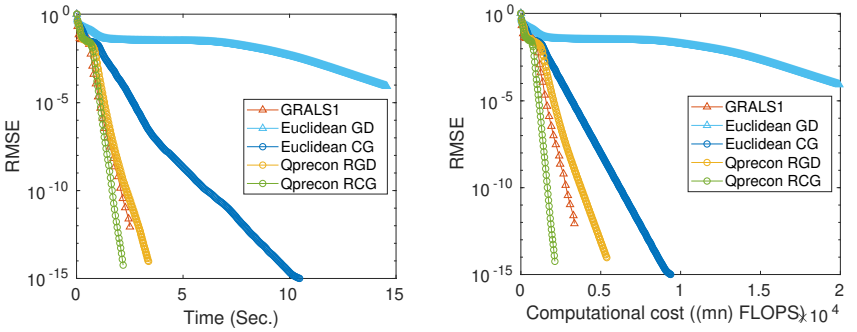


Figure 4.11: RMSE per iteration. Left: the x-axis is wall time at each iteration. Right: the x-axis is the cumulative computational cost at each iteration. Experimental settings: $m = 1000$, $n = 900$, $k = r^\star = 10$, $|\Omega|/mn = 20.0\%$. $M^\star$ is generated with the model (4.46) and is partially observed without any noise.

Our implementation (AltMin)

In addition to GRALS [RYRD15], we implement the alternating minimization method ((4.70a)–(4.70b)) with a linear CG solver that is controlled by the two following parameters,

- $\diamond$ n_{CG}: the iteration budget for each of the two least-squares subproblems.
- $\diamond$ ϵ: tolerance parameter to control the accuracy of the solutions to each of the two subproblems.

The most costly computation in AltMin/GRALS is the computation of $A^{(1)}\mathrm{vec}\left(G^{\mathrm{T}}\right)$ and $A^{(2)}\mathrm{vec}\left(H^{\mathrm{T}}\right)$. Algorithm 4.A.7 avoids searching for indices in the subset Ω_i for each $i \in [\![m]\!]$ by using the following incremental procedure. The notations therein are adapted to the computation of $B^{(1)}\mathrm{vec}\left(G^{\mathrm{T}}\right)$. Note that for the computation of $B^{(2)}\mathrm{vec}\left(H^{\mathrm{T}}\right)$, this algorithm applies by swapping the roles of G and H.

Algorithm 4.A.7 Hessian-vector multiplication $B^{(1)}s$

Input: $\Omega \subset [\![m]\!] \times [\![n]\!]$. $H \in \mathbb{R}^{n\times k}$, vector $s := \mathrm{vec}(G^{\mathrm{T}}) \in \mathbb{R}^{km}$.
Output: $B^{(1)}s$, for $B^{(1)}$ in (4.73).

Initialize the k-dimensional vectors: $y_i = \mathbf{0}$ for $i = 1, .., m$.
for $l = 1, .., |\Omega|$ **do**
  Get (i_l, j_l): the l-th pair of Ω.
  Get h_{j_l} from H. # See (4.74).
  Get $g_{i_l} = [G_{i_l 1}, .., G_{i_l k}]^{\mathrm{T}}$, which is $(s_{i_l k-k+1}, .., s_{i_l k})$.
  Compute $y_{i_l} = y_{i_l} + h_{j_l}(h_{j_l}^{\mathrm{T}} g_{i_l})$.
end for
Return: $\mathrm{vec}\,([y_1, .., y_m]) \in \mathbb{R}^{km}$.

4.A.9 The Hessian of the objective function of (4.9)

The second-order Euclidean directional derivative of f at $x = (G, H) \in \mathbb{R}^{m\times k} \times \mathbb{R}^{n\times k}$ along a direction $\xi = (\xi_G, \xi_H) \in T_x\left(\mathbb{R}^{m\times k} \times \mathbb{R}^{n\times k}\right)$ is defined as

$$\nabla^2 f(x)[\xi] := \frac{d}{dt}\nabla f(x + t\xi)|_{t=0}. \tag{4.76}$$

The gradient vector field has the following expression,

$$\nabla f(x) = \left(SH + \alpha\Theta^{\mathrm{r}}G, S^{\mathrm{T}}G + \alpha\Theta^{\mathrm{c}}H\right), \tag{4.77}$$

where $S := P_\Omega(GH^{\mathrm{T}} - M)$. To simplify notations, we calculate the two matrix components separately,

$$\begin{aligned}\frac{d}{dt}\partial_G f\left(x + t\xi\right)|_{t=0} &= \lim_{t\to 0}\frac{1}{t}\Big[P_\Omega((G + t\xi_G)(H + t\xi_H)^{\mathrm{T}} - M)(H + t\xi_H)\\ &\qquad - P_\Omega(GH^{\mathrm{T}} - M)H + t\alpha\Theta^{\mathrm{r}}\xi_G\Big]\\ &= P_\Omega(G\xi_H^{\mathrm{T}} + \xi_G H^{\mathrm{T}})H + S\xi_H + \alpha\Theta^{\mathrm{r}}\xi_G.\end{aligned}$$

Similarly, $\frac{d}{dt}\partial_H f\left(x + t\xi\right)|_{t=0} = P_\Omega(G\xi_H^{\mathrm{T}} + \xi_G H^{\mathrm{T}})^{\mathrm{T}}G + S^{\mathrm{T}}\xi_G + \alpha\Theta^{\mathrm{c}}\xi_H$. Hence we have

$$\nabla^2 f(x)[\xi] = \begin{pmatrix} P_\Omega(G\xi_H^{\mathrm{T}} + \xi_G H^{\mathrm{T}})H + S\xi_H + \alpha\Theta^{\mathrm{r}}\xi_G \\ P_\Omega(G\xi_H^{\mathrm{T}} + \xi_G H^{\mathrm{T}})^{\mathrm{T}}G + S^{\mathrm{T}}\xi_G + \alpha\Theta^{\mathrm{c}}\xi_H \end{pmatrix}.$$

4.B The matrix model with graph information

The model (4.44) is of particular interest because it models a wide range of real data matrices whose entries present pairwise similarities. Figure 4.1 shows differences between Z (4.44a) and $A^{\mathrm{r}}Z$ (4.44b) in both the data entries and in the graph spectral domain. By using the concept of graph Fourier transforms

(*e.g.* [SRV16]), we illustrate how $(g, A^{\mathrm{r}}, A^{\mathrm{c}})$ in (4.45) transforms a Gaussian random matrix Z into a matrix with more apparent pairwise similarities on the given graphs.

By definition (*e.g.* [SRV16]), the graph Fourier transform of a vector $f \in \mathbb{R}^m$ with respect to the graph Laplacian $L^{\mathrm{r}} = U\Lambda U^{\mathrm{T}}$, is

$$\widehat{f}(\lambda_l) = (Ue_l)^T f, \text{ for all } l \in [\![m]\!].$$

Now we compare the smoothness of the Gaussian low-rank model $Z = FQ^{\mathrm{T}}$ in (4.44a) and the graph-based model $X = A^{\mathrm{r}}Z$ (4.44b) with respect to the row-wise similarity graph L^{r}: For any $j = 1, \ldots, k$, the graph Fourier coefficients of the j-th column of the Gaussian random matrix $F \in \mathbb{R}^{m\times k}$ and transformed matrix $G = A^{\mathrm{r}}F = Ug(\Lambda^{\mathrm{r}})F$ are

$$\widehat{G_{:,j}}(\lambda_l) \quad = \quad (U^T e_l)^T A^{\mathrm{r}} F_{:,j} = \sqrt{g(\lambda_l)} e_l{}^{\mathrm{T}} F_{:,j}, \text{ for all } l \in [\![m]\!],$$

where $F_{:,j} \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sigma_F^2 I_m)$. From basic calculations, the amplitudes of of graph Fourier coefficients of G satisfy

$$\mathbb{E}\left[|\widehat{G_{:,j}}(\lambda_l)|^2\right] \quad = \quad g(\lambda_l)\mathbb{E}\left[\|e_l{}^{\mathrm{T}} F_{:,j}\|_2^2\right] = \sigma_F^2 g(\lambda_l). \tag{4.78}$$

Therefore, when g is decreasing on $(0, +\infty)$, e.g., in the case of (4.46), the expected amplitude of $\widehat{G_{:,j}}(\lambda_l)$ decreases for the increasing eigenvalues $\{\lambda_l\}_{l=2,\ldots,m}$. Note that a small eigenvalue λ corresponds eigenfunctions on the graph with small variations. Hence, in this case, the energy of $G_{:,j}$ in the graph Fourier domain, determined by $(|\widehat{G_{:,j}}(\lambda_l)|)_{1\le l\le m}$, is mostly concentrated on low graph-vertex frequencies.

Indeed, given (4.78), the overall variations of the matrix factor G is related to $\mathrm{tr}(G^{\mathrm{T}}L^{\mathrm{r}}G)$ in the regularizer of our main problem (4.9) as follows,

$$\mathbb{E}\left[\mathrm{tr}(G^{\mathrm{T}}L^{\mathrm{r}}G)\right] = \mathbb{E}\left[\sum_{j=1}^{k}\sum_{l=1}^{m}\lambda_l^2|\widehat{G_{:,j}}(\lambda_l)|^2\right] = k\sigma_F^2\sum_{l=1}^{m}\lambda_l^2 g(\lambda_l).$$

The weighted-sum expression above implies that $\mathrm{tr}(G^{\mathrm{T}}L^{\mathrm{r}}G)$ is small when the amplitudes $(|\widehat{G_{:,j}}(\lambda_l)|)_l$ are more concentrated on low frequencies than on high frequencies. The same property applies to the factor H with respect to L^{c}. This reflects that the graph-based regularizer

$$S_L(x) = \mathrm{tr}(G^{\mathrm{T}}L^{\mathrm{r}}G) + \mathrm{tr}(H^{\mathrm{T}}L^{\mathrm{c}}H),$$

quantifies the smoothness of of the entries of (G, H) on the row and column index sets with respect to the row-wise and column-wise similarity graphs ($\mathcal{G}^{\mathrm{r}}$ and $\mathcal{G}^{\mathrm{c}}$), as explained in (4.2).

4.C Cross validation for parameter selection

A parameter configuration of (4.9) is denoted as $\boldsymbol{\beta} := (\alpha, \gamma_{\mathrm{r}}, \gamma_{\mathrm{c}}) \in \mathbb{R}^3_+$. Given a set $\mathcal{H} = \{\boldsymbol{\beta}_j \in \mathbb{R}^3_+, j = 1, ..., n_{\mathrm{HP}}\}$ of n_{HP} elements, the K-fold cross validation (CV) is used in the experiments to find the best choice in $\mathcal{H}$.

The selection criterion is a validation score that is to be computed for each parameter configuration $\boldsymbol{\beta} \in \mathcal{H}$. Let $\widehat{X}_{\mathcal{S}}(\Omega, \boldsymbol{\beta})$ denote the solution given by a GRMC solver $\mathcal{S}$ using the parameter $\boldsymbol{\beta}$ and the training data on Ω. Then the validation score of $\boldsymbol{\beta}$ is the matrix recovery error of the solution on a set $\Omega_{\mathrm{val}} \subset \Omega^c$ containing some known entries.[11] We use the RMSE (4.43) as the error function. This means the validation score is defined as

$$\mathcal{E}_{\Omega_{\mathrm{val}}}(\widehat{X}_{\mathcal{S}}(\Omega, \boldsymbol{\beta})) := \mathrm{RMSE}(\widehat{X}_{\mathcal{S}}(\Omega, \boldsymbol{\beta}); \Omega_{\mathrm{val}}). \tag{4.79}$$

Next, we proceed the K-fold cross validation for matrix completion with a standard splitting [JWHT13, §5.1.3] of the training and validation sets, given the set of all known entries. Algorithm 4.C.1 describes this procedure specifically for matrix completion.

Algorithm 4.C.1 K-fold cross validation for matrix completion

Input: Union of training and validation sets $\mathcal{U} \subset [\![m]\!] \times [\![n]\!]$ and a test set $\Omega_{\mathrm{te}} \subset \mathcal{U}^c$, a GRMC solver $\mathcal{S} : \mathbb{R}^{m\times k} \times \mathbb{R}^{n\times k} \mapsto \mathbb{R}^{m\times k} \times \mathbb{R}^{n\times k}$, a collection of parameter configurations $\mathcal{H} = \{\boldsymbol{\beta}_j : j = 1, \ldots, n_{\mathrm{HP}}\}$.

Output: $\widehat{\mathrm{CV}}_K(\boldsymbol{\beta})$.

1: **for** $j \in \{1, \ldots, |\mathcal{H}|\}$ **do**
2: **for** $k \in \{1, \ldots, K\}$ **do**
3: Get the k-th partitioning $\mathcal{U} = \Omega^k_{\mathrm{tr}} \cup \Omega^k_{\mathrm{val}}$ such that $|\Omega^k_{\mathrm{tr}}|/|\mathcal{U}| = \frac{K-1}{K}$.
4: Find solution $\widehat{X}_{\mathcal{S}}\left(\Omega^k_{\mathrm{tr}}, \boldsymbol{\beta}_j\right)$ using the solver $\mathcal{S}$.
5: Compute $\mathcal{E}_{\Omega^k_{\mathrm{val}}}(\widehat{X}_{\mathcal{S}}\left(\Omega^k_{\mathrm{tr}}, \boldsymbol{\beta}_j\right))$ # see (4.79)
6: **end for**
7: Compute the K-fold cross validation score $\widehat{\mathrm{CV}}_K(\boldsymbol{\beta}_j)$ as defined in (4.80).
8: **end for**

In the end of this CV procedure (line 7), each parameter setting $\boldsymbol{\beta} \in \mathcal{H}$ is evaluated by the following validation score,

$$\widehat{\mathrm{CV}}_K(\boldsymbol{\beta}) = \frac{1}{K}\sum_{k=1}^{K} \mathcal{E}_{\Omega^k_{\mathrm{val}}}(\widehat{X}_{\mathcal{S}}\left(\Omega^k_{\mathrm{tr}}, \boldsymbol{\beta}\right)). \tag{4.80}$$

Training and validation sets. The training and validation sets during each fold comes from a shuffled splitting of the set of known entries.

[11] Note that this validation set has to be an index set on which the matrix entries are also known, since the matrix recovery error on Ω_{val} needs to be computed.

On synthetic data, the ground-truth matrix $M^\star$ is fully available. In this case, with a given sampling rate ρ, we first generate the union $\mathcal{U}$ of the index set $\Omega_{\text{tr}} \subset [\![m]\!] \times [\![n]\!]$ (of the revealed entries, also called "training data" in a general sense) and the validation set $\Omega_{\text{val}} \subset [\![m]\!] \times [\![n]\!]$. This union is generated according to a certain probabilistic distribution on $[\![m]\!] \times [\![n]\!]$ such that $\mathbb{E}\left[|\mathcal{U}|\right] = \frac{\rho K}{K-1}$. We used the Bernoulli distribution $\mathcal{B}(\rho)$. Subsequently, the test set is either $\mathcal{U}^c$ (the complement of $\mathcal{U}$) or any subset of $\mathcal{U}^c$ with a reasonable size.

For real-world datasets, the data matrix $M^\star$ is often only partially known. Let $\mathcal{O} \subset [\![m]\!] \times [\![n]\!]$ denote the set of all known entries. In this case, we split the available entries into (i) a test set Ω_{te} and (ii) the union of training and validation sets, which is $\mathcal{U} := \Omega_{\text{te}}^c$. Then the training and validation sets are generated in the aforementioned manner based on $\mathcal{U}$.

Chapter 5

Optimization on the set of fixed-rank matrices

Optimization with low-rank matrices is a fundamental problem that arises in signal processing, machine learning and computer vision. In applications such as principal component analysis, matrix recovery and data clustering, the desired solutions to the problem often have a intrinsically low rank [UT19], since the most meaningful information in the data is structured. Moreover, the low-rank constraint reduces significantly the complexity of the problem without compromising the accuracy or utility of the solution. Therefore, low-rank matrix models provide powerful tools for efficient representation, recovery or pattern recognition of the data. One of the approaches to impose the low-rank structure of a m-by-n matrix formulates an optimization with the matrix nuclear norm [CR08, RFP10, CT10], which is a convex relaxation of the matrix rank. Another approach represents the m-by-n matrix X through low-rank matrix factorization $X = AB^{\mathrm{T}}$, where A and B are m-by-k and n-by-k factor matrices. For a search space of dimension $O((m+n)k)$, which is greatly reduced compared to that of the m-by-n matrices within the convex optimization approach, matrix factorization methods present enormous advantage for its low memory requirement and computational cost.

Algorithms for low-rank matrix factorization can be regrouped into two main types, alternating minimization [ZWSP08, JNS13b, Har14] and gradient descent algorithms (e.g., [KO09, KMO10, PKCS18]). Recent advances in matrix recovery problems such as compressed sensing and matrix completion [SL16, GLM16, TBS$^+$16, ZL16] shed light on the absence of spurious local minima in these problems under mild conditions, despite that matrix factorization is a nonconvex problem, and thus they explain formally the success of (Euclidean) gradient descent algorithms for nonconvex matrix recovery. Riemannian algorithms, in a similar spirit as Euclidean gradient descent, exploit manifold structures of a low-rank matrix space, and thus have often been shown

to offer superior performance to Euclidean gradient descent algorithms. As an important example of low-rank matrix spaces, the set $\mathcal{M}_k$ of fixed-rank matrices

$$\mathcal{M}_k = \left\{X \in \mathbb{R}^{m\times n} : \operatorname{rank}(X) = k\right\} \tag{5.1}$$

is a smooth Riemannian manifold (e.g. [Van13]) and can be characterized through the factorization of X in various ways. Depending on the matrix factorization form, a rank-k matrix X is identified with a point in the product space of the factor matrices of X; the Riemannian first-order (and/or second-order) information of the objective function on the product space is computed according to a given metric defined on the manifold $\mathcal{M}_k$. Vandereycken [Van13] used the rank-k singular value decomposition (SVD) to identify a matrix $X \in \mathcal{M}_k$ with the point (U, Σ, V), where U, Σ and V are the factor matrices of the rank-k SVD of X, and considered $\mathcal{M}_k$ as an embedded submanifold of $\mathbb{R}^{m\times n}$, using the Euclidean metric. In recent works about low-rank matrix recovery, Wei et al. [WCCL16, TW16] proposed several variants of the iterative hard thresholding (IHT) algorithm to address the fixed-rank constraint; IHT refers to the projection of a matrix with a rank larger than k onto the set $\mathcal{M}_k$ with respect to the Euclidean metric on $\mathbb{R}^{m\times n}$, in a similar way as the projection-like retraction used in [Van13]. We also refer to [MMBS14] for an overview of matrix factorizations and related Riemannian algorithms for optimization with fixed-rank matrices.

In this chapter, we focus on a quotient manifold-based approach to optimization with fixed-rank matrices. First, we revisit the quotient manifold structure of $\mathcal{M}_k$ based on a metric defined through Riemannian preconditioning [MAS12, MS16] and then investigate the properties of the underlying Riemannian gradient descent (RGD) algorithm. Naturally, the nonlinear descent step of this RGD algorithm is defined according the quotient manifold structure of $\mathcal{M}_k$, instead of an embedded manifold structure. Hence, a notable difference of this RGD algorithm from many other algorithms on the set $\mathcal{M}_k$ is that it is *projection-free*, in the sense that it does not need metric projection (see Section 2.2.3) to maintain the fixed-rank constraint. We show that the RGD algorithm induces a sequence of fixed-rank matrices that enjoys an invariance property regarding the pair of factor matrices of the matrix factorization. Second, we propose novel results about the properties of the Riemannian gradient descent algorithm under the so-called *restricted positive definiteness* (RPD) property [UV20b] of a class of low-rank matrix optimization problems. Through the RPD property of the objective function around a given critical point, we demonstrate the existence of a *region of attraction* on $\mathcal{M}_k$, in which the landscape of the objective function in the ambient space $\mathbb{R}^{m\times n}$ is preserved. From these discoveries, we give a result about the local convergence rate of the RGD algorithm. To the best our knowledge, this is the first convergence analysis of projection-free algorithms on the manifold of fixed-rank matrices.

The convergence results can be applied to matrix recovery problems. Specifically, an application to matrix completion with fixed-rank matrices is dis-

cussed. We highlight the aforementioned properties through numerical experiments: we show that the quotient manifold-based algorithms not only enjoys the benefits of low-rank matrix factorization but also present desirable *invariance* properties that the Euclidean gradient descent does not possess. In particular, its convergence behavior does not vary with changes in the *balancing* between the factor matrices, while the performance of Euclidean gradient descent is easily deteriorated due to unbalanced factor matrices.

In summary, the main contributions in this chapter are as follows.

- ◇ We show formally that the Riemannian gradient descent algorithm under the aforementioned quotient geometric setting enjoys desirable invariance properties. A new result about this quotient manifold-based RGD algorithm is given; see Section 5.2.

- ◇ For a class of low-rank matrix optimization problems, we provide new results about the geometric properties of the RGD algorithm around critical points, under the restricted positive definiteness property. Based on these new results, a convergence rate analysis on $\mathcal{M}_k$ is given; see Section 5.3.

Organization. The rest of this chapter is organized as follows. In Section 5.2, we present the Riemannian quotient manifold structure of $\mathcal{M}_k$ and the quotient manifold-based RGD algorithm, followed by new results about this RGD algorithm. The main results about the RGD algorithm for a class of fixed-rank matrix optimization problems are given in Section 5.3. Numerical experiments and results are presented in Section 5.5. We conclude the chapter in Section 5.6.

5.1 Notation

Given two integers $m, n \geq 1$ and a rank value $k \leq \min(m, n)$, we denote by $\mathbb{R}_*^{m\times k} \times \mathbb{R}_*^{n\times k}$ the product manifold of real $m \times k$ and $n \times k$ matrices with full column-ranks. A point in $\mathbb{R}_*^{m\times k} \times \mathbb{R}_*^{n\times k}$ is denote by $\bar{x} = (G_{\bar{x}}, H_{\bar{x}})$, (G, H) or simply $\bar{x}$ indifferently. By default, the symbols G and H signify the $m \times k$ and $n \times k$ matrices of $\bar{x}$ respectively; they also constitute a pair of left and right factor matrices of $X \in \mathcal{M}_k$ for $X = GH^{\mathrm{T}}$.

A tangent vector to the product space $\overline{\mathcal{M}_k}$ has two matrix components and is denoted as $\bar{\xi} = (\bar{\xi}^{(1)}, \bar{\xi}^{(2)})$. Given a matrix $X \in \mathbb{R}^{m\times n}$, the Euclidean metric on the tangent space $T_X\mathbb{R}^{m\times n} \simeq \mathbb{R}^{m\times n}$, also called the Frobenius inner product, is defined and denoted by $\langle V, W\rangle := \mathrm{tr}(V^{\mathrm{T}}W)$, for $V, W \in T_X\mathbb{R}^{m\times n}$. Given a matrix $X \in \mathcal{M}_k$, the k-th largest singular value of X, i.e., the minimal non-zero singular value, is denoted as $\sigma_{\min}(X)$ by default. The spectral norm of a symmetric positive semidefinite matrix A and the operator norm of the linear operator $A : X \mapsto AX$ are denoted by $\|A\|_2$.

5.2 Optimization on the set of fixed-rank matrices $\mathcal{M}_k$

In this section, we exploit the geometric structure of $\mathcal{M}_k$ (5.1) as a quotient submanifold of the following product manifold (e.g., [MBS11, MMBS14, AAM14])

$$\overline{\mathcal{M}_k} := \mathbb{R}_*^{m\times k} \times \mathbb{R}_*^{n\times k}. \tag{5.2}$$

Based on a metric on $\overline{\mathcal{M}_k}$ that induces a metric on the quotient manifold $\mathcal{M}_k$, we investigate the Riemannian gradient descent algorithm and show that it induces a RGD sequence on $\mathcal{M}_k$.

This section is organized as follows. We first give an overview of the geometric properties—the tangent vectors and metrics—of $\mathcal{M}_k$ as a quotient space of $\overline{\mathcal{M}_k}$ in Section 5.2.1. Then in Section 5.2.2, we explain how an optimization problem on $\mathcal{M}_k$ is related to a problem defined on the product space $\overline{\mathcal{M}_k}$ and give an explicit characterization of a product space-based (or factorization-based) Riemannian gradient descent algorithm regarding its relation with optimization on the quotient space $\mathcal{M}_k$. Such a characterization reveals some interesting invariance properties of all algorithms that are qualified for optimizing a function defined on the quotient space. A detailed description of the algorithm is given in Section 5.2.2.

5.2.1 Geometry of the quotient space $\mathcal{M}_k$

The identification of $\mathcal{M}_k$ as a quotient submanifold of $\overline{\mathcal{M}_k} = \mathbb{R}_*^{m\times k} \times \mathbb{R}_*^{n\times k}$ follows the standard definition of a quotient manifold through canonical projection; see [AMS08, §3.4] for more generic descriptions of this topic. Hereafter, we focus on the geometry of $\mathcal{M}_k$ as a quotient submanifold of $\overline{\mathcal{M}_k}$ [MBS11, MMBS14, AAM14].

Let $\sim$ denote an equivalence relation (see Definition 2.1.11) on the product space $\overline{\mathcal{M}_k}$ such that, for any pair of points $(G,H),(G',H') \in \overline{\mathcal{M}_k}$, $(G,H) \sim (G',H')$ if and only if $GH^{\mathrm{T}} = G'H'^{\mathrm{T}}$. Since the equivalent classes of $\sim$ are the fibers of the matrix multiplication

$$\pi : \overline{\mathcal{M}_k} \mapsto \mathbb{R}^{m\times n}, (G,H) \mapsto GH^{\mathrm{T}},$$

and because $\mathcal{M}_k \subset \mathbb{R}^{m\times n}$ is the image of π, the mapping π induces an one-to-one correspondence between $\mathcal{M}_k$ and $\overline{\mathcal{M}_k}/\!\sim$.

Let π denote the canonical projection $\overline{\mathcal{M}_k} \mapsto \overline{\mathcal{M}_k}/\!\sim : (G,H) \mapsto [(G,H)]$, where $[(G,H)]$ denotes the equivalence class $\{(G',H') \in \overline{\mathcal{M}_k} : G'H'^{\mathrm{T}} = GH^{\mathrm{T}}\}$. The structure of the fibers can be characterized by the linear group $\mathrm{GL}(k)$: For any $X \in \mathcal{M}_k$, the set of all pairs of full column-rank matrices $(G,H) \in \overline{\mathcal{M}_k}$ satisfying $\pi((G,H)) = X$ is the equivalent class

$$\pi^{-1}(X) := \left\{(GF^{-1}, HF^{\mathrm{T}}) : F \in \mathrm{GL}(k)\right\}. \tag{5.3}$$

In fact, the operations $(G, H) \mapsto (GF^{-1}, HF^{\mathrm{T}}), F \in \mathrm{GL}(k)$ correspond to all possible transformations that leave the matrix product $X = GH^{\mathrm{T}} \in \mathcal{M}_k$ unchanged. Hence the identification $\mathcal{M}_k \simeq \overline{\mathcal{M}_k}/\mathrm{GL}(k)$. One can see that the quotient space $\mathcal{M}_k$ is a quotient submanifold of $\overline{\mathcal{M}_k}$ [AMS08, MBS11, MMBS14, AAM14]. The product space $\overline{\mathcal{M}_k}$ is referred to as the *total space*.

Tangent space and metrics. The tangent vectors to the quotient manifold $\mathcal{M}_k$ can be represented by tangent vectors to the total space $\overline{\mathcal{M}_k}$. In fact, given a matrix $X \in \mathcal{M}_k$ and a tangent vector $\xi \in T_X\mathcal{M}_k$, the mapping $\pi : \overline{\mathcal{M}_k} \mapsto \mathcal{M}_k$ actually induces infinitely many representations of ξ, due to the fact that it is a projection. Let $\bar{x}$ be an element of the equivalence class $\pi^{-1}(X)$. Any element $\bar{\xi} \in T_{\bar{x}}\overline{\mathcal{M}_k}$ that satisfies $\mathrm{D}\pi(\bar{x})[\bar{\xi}] = \xi$ can be considered as a representation of ξ. Indeed, for any smooth function $f : \mathcal{M}_k \mapsto \mathbb{R}$, the function $\bar{f} := f \circ \pi : \overline{\mathcal{M}_k} \mapsto \mathbb{R}$, one has the following identification,

$$\mathrm{D}\bar{f}(\bar{x})[\bar{\xi}] = \mathrm{D}f(\pi(\bar{x}))[\mathrm{D}\pi(\bar{x})[\bar{\xi}]] = \mathrm{D}f(X)[\xi].$$

Since $\pi : \overline{\mathcal{M}_k} \mapsto \mathcal{M}_k$ is surjective, the kernel of $\mathrm{D}\pi(\bar{x}) : T_{\bar{x}}\overline{\mathcal{M}_k} \mapsto T_X\mathcal{M}_k$ is non-trivial, thus, the matrix representation of ξ in $T_{\bar{x}}\overline{\mathcal{M}_k}$ is not unique. Nevertheless, one can find a unique representation of ξ in a subspace of $T_{\bar{x}}\overline{\mathcal{M}_k}$. This is realized by decomposing the tangent space

$$T_{\bar{x}}\overline{\mathcal{M}_k} \simeq \mathbb{R}^{m\times k} \times \mathbb{R}^{n\times k}$$

into two complementary subspaces (see Section 2.1.3) as follows: $T_{\bar{x}}\overline{\mathcal{M}_k} = \mathcal{V}_{\bar{x}} \oplus \mathcal{H}_{\bar{x}}$, where $\mathcal{V}_{\bar{x}}$, called the *vertical space*, is the tangent space at $\bar{x}$ of the equivalence class $[\bar{x}]$, i.e.,

$$\mathcal{V}_{\bar{x}} := T_{\bar{x}}(\pi^{-1}(X)), \tag{5.4}$$

and $\mathcal{H}_{\bar{x}}$, called the *horizontal space*, is a subspace of $T_{\bar{x}}\overline{\mathcal{M}_k}$ complementary to $\mathcal{V}_{\bar{x}}$. One can see that a tangent vector $\bar{\xi} \in \mathcal{V}_{\bar{x}}$ satisfies $\mathrm{D}\pi(\bar{x})[\bar{\xi}] = 0$. Consequently, for any $X \in \mathcal{M}_k$ and $\xi \in T_X$, there is a unique representation $\bar{\xi} \in \mathcal{H}_{\bar{x}} \subset T_{\bar{x}}\overline{\mathcal{M}_k}$ of ξ such that

$$\mathrm{D}\pi(\bar{x})[\bar{\xi}] = \xi.$$

The tangent vector $\bar{\xi} \in \mathcal{H}_{\bar{x}}$ is called the horizontal lift of ξ.

Given the horizontal lifts as the matrix representation of tangent vectors to the quotient manifold $\mathcal{M}_k$, any metric $\bar{g}$ on the total space that satisfies following invariance property induces a metric on $\mathcal{M}_k$.

Definition 5.2.1 (e.g., [AAM14, §3]). *For $X \in \mathcal{M}_k$ and $\bar{x} \in \pi^{-1}(X)$, let $\bar{\xi}, \bar{\eta} \in T_{\bar{x}}\overline{\mathcal{M}_k}$ denote the horizontal lifts of ξ and η respectively, a metric $\bar{g}$ in the total space is said to be* invariant *along $\pi^{-1}(X)$ if*

$$\bar{g}_{\bar{x}}(\bar{\xi}, \bar{\eta}) = \bar{g}_{\bar{y}}(\bar{\xi}_{\bar{y}}, \bar{\eta}_{\bar{y}}), \tag{5.5}$$

for any point $\bar{y} \sim \bar{x}$ in $\pi^{-1}(X)$, where $\bar{\xi}_{\bar{y}}, \bar{\eta}_{\bar{y}}$ denote the horizontal lifts of ξ and η at $\bar{y}$ respectively.

Indeed, given a metric $\bar{g}$ that satisfies the invariance property in Definition 5.2.1, the inner product $g_X : T_X\mathcal{M}_k \times T_X\mathcal{M}_k \mapsto \mathbb{R}$ such that $g_X(\xi, \eta) = \bar{g}_{\bar{x}}(\bar{\xi}_{\bar{x}}, \bar{\eta}_{\bar{x}})$ is a metric on the quotient manifold $\mathcal{M}_k$. In Section 5.2.2, we give an explicit example of a metric $\bar{g}$ on $\overline{\mathcal{M}_k}$ that induces a metric on $\mathcal{M}_k$.

5.2.2 Metrics and algorithms on $\overline{\mathcal{M}_k}$

Based on the relations between the points and tangent vectors on the total space and those on the quotient space, we establish links between the optimization with fixed-rank matrices and matrix factorization in this subsection.

Given a function $f : \mathcal{M}_k \mapsto \mathbb{R}$, we consider $\bar{f} := f \circ \pi$. By construction, $\bar{f}(\bar{x}) = f(X)$ for any $\bar{x} \in \pi^{-1}(X)$. Naturally, the optimization of f on $\mathcal{M}_k$:

$$\min_{X \in \mathcal{M}_k} f(X) \tag{5.6}$$

can be dealt with by minimizing $\bar{f}$ on the total space $\overline{\mathcal{M}_k}$.

Example 5.2.2 (Low-rank matrix completion)**.** *In the context of low-rank matrix completion, let $M^\star$ be a given matrix observed only on an index set Ω and let $\psi : \mathbb{R}^{m\times n} \mapsto \mathbb{R}$ be a regularizer. The optimization on $\mathcal{M}_k$ of $f(X) := \frac{1}{2}\|P_\Omega(X - M^\star)\|_{\mathrm{F}}^2 + \psi(X)$ can be realized by solving the following matrix factorization problem,*

$$\min_{(G,H)\in\mathbb{R}_*^{m\times k}\times\mathbb{R}_*^{n\times k}} \bar{f}(G, H) := \frac{1}{2p}\|P_\Omega(GH^{\mathrm{T}} - M^\star)\|_F^2 + \psi(GH^{\mathrm{T}}). \tag{5.7}$$

Indeed, the function $\bar{f}$ is invariant along the equivalence classes (5.3)*.*

Example 5.2.3 (Low-rank matrix approximation)**.** *Given a matrix $A \in \mathbb{R}^{m\times n}$. The problem of finding the best rank-k approximation of A can be formulated by minimizing $f(X) = \frac{1}{2}\|X - A\|_{\mathrm{F}}^2$ on $\mathcal{M}_k$. Then $\bar{f} = f \circ \pi : \overline{\mathcal{M}_k} \mapsto \mathbb{R}$ has the following form*

$$\bar{f}(G, H) = \frac{1}{2}\|GH^{\mathrm{T}} - A\|_{\mathrm{F}}^2. \tag{5.8}$$

The graph of $\bar{f}$ (5.8) can be visualized in the example with the smallest dimensions. Figure 5.1 shows the landscape of $\bar{f} : \mathbb{R}_* \times \mathbb{R}_* \mapsto \mathbb{R} : (x, y) \mapsto \frac{1}{2}(xy - A)^2$ on the 2-dimensional plane, for $A = 1$.

Optimizing $\bar{f}$ on $\overline{\mathcal{M}_k}$ instead of (5.6) is essentially a matrix factorization approach since it deals with the rank-k factor matrices of the matrix variable in $\mathcal{M}_k$. Through the quotient structure of $\mathcal{M}_k$, the function $\bar{f}$ is invariant along the equivalence classes in $\overline{\mathcal{M}_k}$, i.e., $f(\bar{x}) = f(\bar{x}')$, for any $\bar{x}' \sim \bar{x}$. This poses potential issues to algorithms for solving the problem on $\overline{\mathcal{M}_k}$. For example, due to the invariance of $\bar{f}$ in equivalence classes, any (isolated) local minimum of f on $\mathcal{M}_k$ corresponds to a whole equivalence class in $\overline{\mathcal{M}_k}$, which contains an

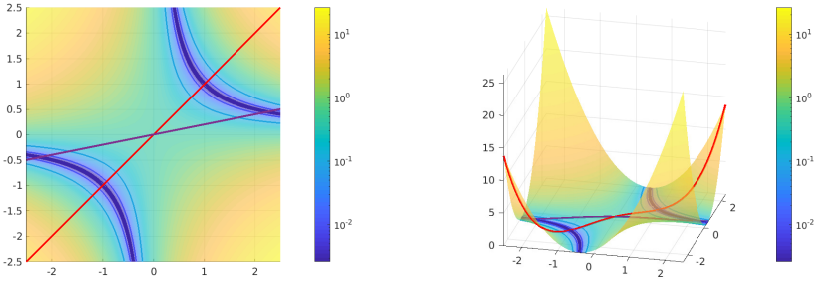


Figure 5.1: Landscape of the cost function of matrix factorization on the 2D plane.

infinite number of nondegenerate minima. Similarly, since the landscape of $\bar{f}$ on $\overline{\mathcal{M}_k}$ is different and actually more complicated (the search space being the total space $\overline{\mathcal{M}_k}$) than f, the optimization path of an algorithm on $\overline{\mathcal{M}_k}$ generally depends on the location of its current iterate in its equivalence class. To address these issues, we consider algorithms using a certain type of Riemannian gradient on $\overline{\mathcal{M}_k}$ that enjoys invariance properties regarding the location of its iterates in the equivalence classes. For this purpose, we first exploit the quotient structure of $\mathcal{M}_k$ and consider a Riemannian metric on $\overline{\mathcal{M}_k}$ that induces a metric on the quotient manifold $\mathcal{M}_k$. Therefore, in the remaining part of this subsection, we introduce one example of such metrics on $\overline{\mathcal{M}_k}$ and present a Riemannian gradient descent algorithm under this metric. Then, in the next subsection, we investigate the properties of the underlying Riemannian gradient and the RGD algorithm.

As mentioned in the last subsection, any metric in the total space that is invariant along equivalence classes induces a metric in the quotient space. Such a metric provides a way to relate the Riemannian gradients of $\bar{f}$ to gradients of f. Next, we study a metric that satisfies this invariance property. This particular metric is interesting not only because it induces a metric on the quotient space but also it is adapted to Frobenius norm-based loss functions in the sense of Riemannian preconditioning.

A metric on $\overline{\mathcal{M}_k}$ through Riemannian preconditioning

We introduce a metric on $\overline{\mathcal{M}_k}$ designed through Riemannian preconditioning that has been shown to have nice properties in the optimization of Frobenius norm-based loss functions (e.g., Example 5.2.2) with fixed-rank matrices.

Definition 5.2.4 ([MAS12]). *Given $\bar{x} := (G, H) \in \overline{\mathcal{M}_k}$, let $\bar{g}_{\bar{x}} : T_{\bar{x}}\overline{\mathcal{M}_k} \times T_{\bar{x}}\overline{\mathcal{M}_k}$ denote an inner product defined as follows,*

$$\bar{g}_{\bar{x}}(\bar{\xi}, \bar{\eta}) = \mathrm{tr}(\bar{\xi}^{(1)T}\bar{\eta}^{(1)}(H^{\mathrm{T}}H)) + \mathrm{tr}((\bar{\xi}^{(2)})^{\mathrm{T}}\bar{\eta}^{(2)}(G^{\mathrm{T}}G)), \tag{5.9}$$

for $\bar{\xi}, \bar{\eta} \in T_{\bar{x}}\overline{\mathcal{M}_k}$.

It can be shown that $\bar{g}_{\bar{x}}(\cdot,\cdot)$ is a Riemannian metric since it is symmetric and positive-definite at any point $\bar{x}$ and it is a smooth-varying bilinear form on $\overline{\mathcal{M}_k}$.

From Definition 2.1.22, the Riemannian gradient of a function $\bar{f}$ at $\bar{x} \in \overline{\mathcal{M}_k}$ is the unique vector, denoted as $\mathrm{grad}\bar{f}\,(\bar{x}) \in T_{\bar{x}}\overline{\mathcal{M}_k}$, such that $\bar{g}_{\bar{x}}(\bar{\xi}, \mathrm{grad}\bar{f}\,(\bar{x})) = \mathrm{D}\bar{f}(\bar{x})[\bar{\xi}]$, $\forall \bar{\xi} \in T_{\bar{x}}\overline{\mathcal{M}_k}$. Therefore, given the metric (5.9), the Riemannian gradient of $\bar{f}$ has the following form,

$$\mathrm{grad}\bar{f}\,(\bar{x}) = \left(\partial_G \bar{f}\,(\bar{x})\,(H^{\mathrm{T}}H)^{-1}, \partial_H \bar{f}\,(\bar{x})\,(G^{\mathrm{T}}G)^{-1}\right), \tag{5.10}$$

where $\partial_G \bar{f}\,(\bar{x})$ and $\partial_H \bar{f}\,(\bar{x})$ are the two partial differentials of $\bar{f}$.

The metric $\bar{g}$ defined in (5.9) was proposed by [MAS12] in the context of matrix completion with fixed-rank matrices, and it can be seen as a specially adapted metric for the two-factor matrix factorization problems. Indeed, note that the second-order partial differentials of $\bar{f}(G,H) := \frac{1}{2}\|GH^{\mathrm{T}} - M^{\star}\|_F^2$ have the following forms:

$$\partial_G^2 \bar{f}(G,H)[\bar{\xi}] = \bar{\xi}^{(1)}H^{\mathrm{T}}H \text{ and } \partial_H^2 \bar{f}(G,H)[\bar{\xi}] = \bar{\xi}^{(2)}G^{\mathrm{T}}G.$$

Let $\mathcal{H} : T_{(G,H)}\overline{\mathcal{M}_k} \mapsto T_{(G,H)}\overline{\mathcal{M}_k}$ denote an operator that maps the tangent vector $\bar{\xi}$ to the right hand-side terms above, that is, $\mathcal{H}\bar{\xi} = (\bar{\xi}^{(1)}H^{\mathrm{T}}H, \bar{\xi}^{(2)}G^{\mathrm{T}}G)$. Then one can see that the Riemannian gradient (5.10), denoted as $\mathrm{grad}\bar{f}\,(\bar{x}) := (\bar{\xi}^{(1)}, \bar{\xi}^{(2)})$, is the solution to the following secant equation $\mathcal{H}\bar{\xi} = (\partial_G \bar{f}(\bar{x}), \partial_H \bar{f}(\bar{x}))$, which shows that the Riemannian gradient is an approximation of the Newton direction of $\bar{f}$ at (G,H). Hence, the metric $\bar{g}$ (5.9) is also referred to as the *preconditioned* metric on the two-factor product space $\mathbb{R}_*^{m\times k} \times \mathbb{R}_*^{n\times k}$. We refer to [MS16] for a more thorough view about metric selection with Riemannian preconditioning.

The following proposition shows that the metric (5.9) induces a metric on the quotient space $\mathcal{M}_k$.

Proposition 5.2.5. *For any matrix $X \in \mathcal{M}_k$, the preconditioned metric* (5.9) *satisfies the invariance property as in Definition 5.2.1, that is, for two horizontal lifts $\bar{x} \in \overline{\mathcal{M}_k}$ and $\bar{\xi}, \bar{\eta} \in T_{\bar{x}}\overline{\mathcal{M}_k}$, $\bar{g}_{\bar{x}}(\bar{\xi}, \bar{\eta}) = \bar{g}_{\bar{x}'}(\bar{\xi}', \bar{\eta}')$, for any $\bar{x}' \sim \bar{x}$, where $\bar{\xi}', \bar{\eta}' \in T_{\bar{x}'}\overline{\mathcal{M}_k}$ are the horizontal lifts at $\bar{x}'$ such that $\mathrm{D}\pi(\bar{x}')[\bar{\xi}'] = \mathrm{D}\pi(\bar{x})[\bar{\xi}]$ and $\mathrm{D}\pi(\bar{x}')[\bar{\eta}'] = \mathrm{D}\pi(\bar{x})[\bar{\eta}]$.*

Proof. For any $\bar{x} := (G,H) \in \overline{\mathcal{M}_k}$ and $\bar{x}' \sim \bar{x}$, there exists an invertible matrix $F \in \mathrm{GL}\,(k)$, such that $\bar{x}' = (GF^{\mathrm{T}}, HF^{-1})$. Since the tangent vector $\bar{\xi}'$ must satisfy $\mathrm{D}\pi(\bar{x}')[\bar{\xi}'] = \mathrm{D}\pi(\bar{x})[\bar{\xi}]$, and note that $\mathrm{D}\pi(\bar{x})[\bar{\xi}] = G(\bar{\xi}^{(2)})^{\mathrm{T}} + \bar{\xi}^{(1)}H^{\mathrm{T}}$ for $\bar{x} = (G,H)$, $\bar{\xi}'$ and F must satisfy

$$G(\bar{\xi}^{(2)})^{\mathrm{T}} + \bar{\xi}^{(1)}H^{\mathrm{T}} = GF^{\mathrm{T}}(\bar{\xi'}^{(2)})^{\mathrm{T}} + \bar{\xi'}^{(1)}(HF^{-1})^{\mathrm{T}},$$

for any $\bar{\xi} \in T_{\bar{x}}\overline{\mathcal{M}_k}$, which yields

$$\bar{\xi}'^{(1)} = \bar{\xi}^{(1)} F^{\mathrm{T}} \text{ and } \bar{\xi}'^{(2)} = \bar{\xi}^{(2)} F^{-1}.$$

The same relation holds for $\bar{\eta}, \bar{\eta}'$ and F. Applying these equalities into the expression of $\bar{g}_{\bar{x}'}(\bar{\xi}', \bar{\eta}')$, where $\bar{g}$ is the preconditioned metric (5.9), we recover immediately the expression of $\bar{g}_{\bar{x}}(\bar{\xi}, \bar{\eta})$. □

Riemannian gradient descent algorithms. Under the quotient geometric settings of $\mathcal{M}_k$, we focus on algorithms for optimization on $\overline{\mathcal{M}_k}$ in relation with the main problem (5.6). We introduce two Riemannian algorithms on the product space $\overline{\mathcal{M}_k}$ that require only the first-order information of the objective function, followed by computational details.

Algorithm 5.2.1 Riemannian Gradient Descent based on the preconditioned metric (Qprecon RGD)

Input: Initial point $\bar{x}_0 \in \overline{\mathcal{M}_k}$, parameters $\beta, \sigma \in (0,1)$, and $\epsilon > 0$; $t = 0$.
Output: $\bar{x}_t \in \overline{\mathcal{M}_k}$.

1: **while** $\|\mathrm{grad}\bar{f}\,(\bar{x}_t)\| > \epsilon$ **do**
2: Set $\bar{\eta}_t = -\mathrm{grad}\bar{f}\,(\bar{x}_t)$.
3: Backtracking line search: given θ_t^0, find the smallest integer $\ell \geq 0$ such that, for $\theta_t := \theta_t^0 \beta^\ell$,

$$\bar{f}(\bar{x}_t) - \bar{f}\,(\bar{x}_t + \theta_t \bar{\eta}_t) \geq \sigma \theta_t \bar{g}_{\bar{x}_t}(-\mathrm{grad}\bar{f}\,(\bar{x}_t), \bar{\eta}_t). \tag{5.11}$$

4: Update: $\bar{x}_{t+1} = \bar{x}_t + \theta_t \bar{\eta}_t$; $t \leftarrow t+1$.
5: **end while**

In Algorithm 5.2.1, the details are as follows. The search direction is the negative Riemannian gradient defined in (5.10). The operation needed for the gradient descent update step (line 4) on $\overline{\mathcal{M}_k}$ is chosen to be as the identity map, which is a valid retraction operator on the $\overline{\mathcal{M}_k}$. The stepsize θ_t in each iteration is obtained following a backtracking line search procedure with respect to the line search condition (5.11). In this backtracking procedure, the initial trial stepsize θ_t^0 (line 3) is important to the time efficiency of algorithm. We consider the following methods for setting the initial trial stepsize, including notably (i) exact line minimization; see (4.23) in Chapter 4; and (ii) Riemannian Barzilai–Borwein stepsize rules [IP18], which have been shown to be an efficient stepsize method for Riemannian gradient methods. The following two variants are considered,

$$\theta_t^{\mathrm{BB1}} := \frac{\bar{g}_{\bar{x}_t}(\bar{z}_{t-1}, \bar{z}_{t-1})}{|\bar{g}_{\bar{x}_t}(\bar{z}_{t-1}, \bar{y}_{t-1})|}, \qquad \theta_t^{\mathrm{BB2}} := \frac{|\bar{g}_{\bar{x}_t}(\bar{z}_{t-1}, \bar{y}_{t-1})|}{\bar{g}_{\bar{x}_t}(\bar{y}_{t-1}, \bar{y}_{t-1})}, \tag{5.12}$$

where $\bar{z}_{t-1} = \bar{x}_t - \bar{x}_{t-1}$ and $\bar{y}_{t-1} = \mathrm{grad}\bar{f}\,(\bar{x}) - \mathrm{grad}\bar{f}\,(\bar{x}_{t-1})$.

Riemannian conjugate gradient descent. Based on the same computational elements as for Algorithm 5.2.1, we also consider a Riemannian conjugate gradient (Qprecon RCG) algorithm on the total space $\overline{\mathcal{M}_k}$. With the same Riemannian gradient definition, the search direction $\bar{\eta}$ of the RCG algorithm at the t-th iteration is defined as $\bar{\eta}_t = \bar{\xi}_t + \beta_t \bar{\eta}^{t-1}$, with $\bar{\xi}_t := -\mathrm{grad}\bar{f}\,(\bar{x}_t))$, where β_t is computed using a Riemannian version of one of the nonlinear conjugate gradient rules. Details about the RCG search direction are presented in the similar algorithm (Algorithm 4.3.2) in Chapter 4. In the numerical experiments, we choose the Riemannian version of the following modified Hestenes-Stiefel rule [HS52] (HS+)

$$\beta = \max\left(0, \frac{\bar{g}_{\bar{x}_t}\left(\bar{\xi}_t - \bar{\xi}_{t-1}, \bar{\xi}_t\right)}{\bar{g}_{\bar{x}_t}\left(\bar{\xi}_t - \bar{\xi}_{t-1}, \bar{\eta}_{t-1}\right)}\right).$$

5.2.3 Induced sequence in the quotient manifold

In this subsection, we investigate the relation between the RGD algorithm (Algorithm 5.2.1) on $\overline{\mathcal{M}_k}$ and the optimization problem (5.6) on $\mathcal{M}_k$. First, through the Riemannian gradient of $\bar{f} = f \circ \pi : \overline{\mathcal{M}_k} \mapsto \mathbb{R}$ under the preconditioned metric (5.9), we propose an explicit form of the induced Riemannian gradient of f. Second, we study the image under the projection π of the sequence $\{\bar{x}_t\}_{t\geq 0} \subset \overline{\mathcal{M}_k}$ produced by the RGD algorithm; the sequence $\{\pi(\bar{x}_t)\}_{t\geq 0}$ in $\mathcal{M}_k$ is referred to as the *induced* sequence. Then, we propose a result about the induced sequence of the quotient-manifold based RGD algorithm.

Under the preconditioned metric $\bar{g}$ (5.9), the gradient vector field of $\bar{f}$ induces a Riemannian gradient vector field of f in the tangent bundle of $\mathcal{M}_k$. Indeed, from Proposition 5.2.5, the metric $\bar{g}$ (5.9) is invariant along the equivalence classes and therefore induces a metric g in $\mathcal{M}_k$ such that, for any $X \in \mathcal{M}_k$ and $\xi, \eta \in T_X\mathcal{M}_k$,

$$g_X(\eta, \xi) = g_{\bar{x}}(\bar{\eta}, \bar{\xi}), \tag{5.13}$$

where $\bar{x}$ is an element in $\pi^{-1}(X)$ and $\bar{\eta}$ and $\bar{\xi}$ are the horizontal lifts (at $\bar{x}$) of η and ξ respectively. Note that with $\bar{\eta} := \mathrm{grad}\bar{f}\,(\bar{x})$,

$$\begin{aligned} \bar{g}_{\bar{x}}(\bar{\eta}, \bar{\xi}) &= \mathrm{D}\bar{f}(\bar{x})[\bar{\xi}] = D(f \circ \pi)(\bar{x})[\bar{\xi}] = \mathrm{D}f(\pi(\bar{x}))[\mathrm{D}\pi(\bar{\xi})] \\ &\equiv g_{\pi(\bar{x})}(\xi, \mathrm{grad}f\,(\pi(\bar{x}))), \end{aligned}$$

for any $\bar{\xi} \in T_{\bar{x}}\overline{\mathcal{M}_k}$. Through the identification (5.13), the horizontal component of $\eta = \mathrm{grad}\bar{f}\,(\bar{x})$ is the horizontal lift of $\mathrm{grad}f\,(X)$ at $\bar{x} \in \pi^{-1}(X)$. Also note that $\mathrm{grad}\bar{f}\,(\bar{x})$ belongs to the horizontal space $\mathcal{H}_{\bar{x}}$: since $\bar{f}$ is invariant along the equivalence classes, it is constant on $\pi^{-1}(X)$, which entails that for any $\bar{\xi} \in \mathcal{V}_{\bar{x}} = T_{\bar{x}}(\pi^{-1}(X))$, $\mathrm{D}\bar{f}(\bar{x})[\bar{\xi}] = \bar{g}_{\bar{x}}(\bar{\xi}, \mathrm{grad}\bar{f}\,(\bar{x})) = 0$. Hence, $\mathrm{grad}\bar{f}\,(\bar{x}) \in (\mathcal{V}_{\bar{x}})^{\perp} = \mathcal{H}_{\bar{x}}$. Therefore, $\mathrm{grad}\bar{f}\,(\bar{x})$ is the horizontal lift at $\bar{x}$ of $\mathrm{grad}f\,(X)$.

Moreover, the horizontal lifts $\mathrm{grad}\bar{f}\,(\bar{x})$, for all $\bar{x} \in \pi^{-1}(X)$, are equivalent, i.e., the induced Riemannian gradient $\mathrm{grad}f\,(X)$ is invariant to the location

of $\bar{x}$ in the equivalence class $\pi^{-1}(X)$. In addition to this remark, we give an explicit form of $\mathrm{grad} f(X)$ in the following proposition.

Proposition 5.2.6. *Given a matrix $X \in \mathcal{M}_k$, let $X = U\Sigma V^{\mathrm{T}}$ denote its* SVD *and let $\bar{x} := (G, H) \in \pi^{-1}(X)$ be an element of $\overline{\mathcal{M}_k}$. Under the metric induced by the preconditioned metric* (5.9)*, the Riemannian gradient of f on the quotient manifold $\mathcal{M}_k$ satisfies*

$$\mathrm{grad} f(X) = \mathcal{G}_X(\nabla f(X)), \tag{5.14}$$

where $\nabla f(X)$ is the Euclidean gradient of f in $\mathbb{R}^{m\times n}$ and $\mathcal{G}_X : \mathbb{R}^{m\times n} \mapsto T_X\mathcal{M}_k$ is a linear operator defined as follows,

$$\mathcal{G}_X(Z) = \mathcal{P}_U Z + Z\mathcal{P}_V, \tag{5.15}$$

for all $Z \in \mathbb{R}^{m\times n}$, where $\mathcal{P}_U := UU^{\mathrm{T}}$ and $\mathcal{P}_V := VV^{\mathrm{T}}$ are the orthogonal projections onto the column and row subspaces of X respectively.

Proof. Let $\mathrm{grad}\bar{f}(\bar{x}) = (\bar{\eta}^{(1)}, \bar{\eta}^{(2)})$ denote the two components of the gradient (5.10) of $\bar{f}$. With $\bar{x} = (G, H) \in \pi^{-1}(X)$, where $X := U\Sigma V^{\mathrm{T}} \in \mathcal{M}_k$, we have

$$\bar{\eta}^{(1)} = \partial_G \bar{f}(\bar{x})\,(H^{\mathrm{T}}H)^{-1} \text{ and } \bar{\eta}^{(2)} = \partial_H \bar{f}(\bar{x})\,(G^{\mathrm{T}}G)^{-1},$$

where $\partial_G \bar{f}(\bar{x}) = \nabla f(X)H$ and $\partial_H \bar{f}(\bar{x}) = \nabla f(X)^{\mathrm{T}}G$. Since $\mathrm{grad}\bar{f}(\bar{x})$ is the horizontal lift of $\mathrm{grad} f(X)$, by definition, $\mathrm{grad} f(X) = \mathrm{D}\pi(X)[\mathrm{grad}\bar{f}(\bar{x})]$. Therefore, we have

$$\begin{aligned}
\mathrm{grad} f(X) &= \mathrm{D}\pi(X)[\mathrm{grad}\bar{f}(\bar{x})] \\
&= G(\bar{\eta}^{(2)})^{\mathrm{T}} + \bar{\eta}^{(1)}H^{\mathrm{T}} \\
&= G(G^{\mathrm{T}}G)^{-1}G^{\mathrm{T}}\nabla f(X) + \nabla f(X)H(H^{\mathrm{T}}H)^{-1}H \\
&= \mathcal{P}_U\nabla f(X) + \nabla f(X)\mathcal{P}_V,
\end{aligned}$$

where $\mathcal{P}_U$ and $\mathcal{P}_V$ are the matrices defined in the statement. □

Remark 5.2.7. In the notation of (5.15), $\mathcal{P}_U$ and $\mathcal{P}_V$ are the matrices of the orthogonal projections. We also denote by abuse the action of these projection operators as $\mathcal{P}_U(Z) = \mathcal{P}_U Z$ and $\mathcal{P}_V(Z) = Z\mathcal{P}_V$ respectively. The action of $\mathcal{G}_X$ (5.15) can be rewritten as $\mathcal{G}_X(Z) = \mathcal{P}_U Z\mathcal{P}_V^{\mathrm{T}} + \mathcal{P}_U{}^{\mathrm{T}} Z\mathcal{P}_V + 2\mathcal{P}_U Z\mathcal{P}_V$. This implies that the operator $\mathcal{G}_X$ is related to the orthogonal projection $\mathrm{P}_{T_X\mathcal{M}_k}$ (2.16a) (i.e., (2.5) in [Van13]) through

$$\mathcal{G}_X(Z) = \mathrm{P}_{T_X\mathcal{M}_k}(Z) + \mathcal{P}_U Z\mathcal{P}_V, \text{ for all } Z \in \mathbb{R}^{m\times n}. \tag{5.16}$$

This relation is used in the proof of a main lemma (Lemma 5.3.11) later. □

Next, let $\{\bar{x}_t\}_{t\geq 0}$ denote the sequence generated by Algorithm 5.2.1. We take an interest in the image of the sequence under the projective mapping $\pi : \overline{\mathcal{M}_k} \mapsto \mathcal{M}_k$. Let $\{X_t\}_{t\geq 0} \subset \mathcal{M}_k$ be the sequence associated with $\{\bar{x}_t\}_{t\geq 0}$

such that $X_t = \pi(\bar{x}_t)$. The following lemma describes how the induced sequence $\{X_t\}_{t\geq 0}$ is related to the update rule of Algorithm 5.2.1.

Lemma 5.2.8. *Let $(\bar{x}_t)_{t\geq 0} \in \overline{\mathcal{M}_k}$ denote the sequence generated by Algorithm 5.2.1 and let $\{X_t\}_{t\geq 0}$ be the induced sequence in $\mathcal{M}_k$. Then the iterates satisfy*

$$X_{t+1} = X_t - \theta_t \mathrm{grad} f(X) + \theta_t^2 \nabla f(X_t) X_t^\dagger \nabla f(X_t), \tag{5.17}$$

where $X_t^\dagger \in \mathbb{R}^{n\times m}$ denotes the Moore–Penrose pseudoinverse of X_t.

Proof. To simplify the notations, we omit the subscript t of all terms related to the t-th iterate (G_t, $\bar{\eta}_t$, stepsize θ_t) and denote X_{t+1} by X_+ in the following equations. Let $\bar{\eta} := \mathrm{grad}\bar{f}\,(\bar{x})$ denote the Riemannian gradient of $\bar{f}$ on the current iterate $\bar{x} := (G, H)$. From the update step in Algorithm 5.2.1, we have

$$\begin{aligned}
X^+ := \pi(\bar{x} - \theta \mathrm{grad}\bar{f}\,(\bar{x})) &= \pi(\bar{x}) - \theta\left(G(\bar{\eta}^{(2)})^{\mathrm{T}} + \bar{\eta}^{(1)} H^{\mathrm{T}}\right) + \theta^2 \bar{\eta}^{(1)}(\bar{\eta}^{(2)})^{\mathrm{T}} \\
&= X - \theta \mathrm{grad} f(X) + \theta^2 \partial_G \bar{f}(\bar{x})(H^{\mathrm{T}}H)^{-1}(G^{\mathrm{T}}G)^{-1}\left(\partial_H \bar{f}(\bar{x})\right)^{\mathrm{T}} && (5.18) \\
&= X - \theta \mathrm{grad} f(X) + \theta^2 \underbrace{\nabla f(X) H (H^{\mathrm{T}}H)^{-1}(G^{\mathrm{T}}G)^{-1} G^{\mathrm{T}} \nabla f(X)}_{\phi_1} && (5.19) \\
&= X - \theta \mathrm{grad} f(X) + \theta^2 \nabla f(X) X^\dagger \nabla f(X), && (5.20)
\end{aligned}$$

which proves (5.17). The equation 5.18 is obtained by identifying $(G(\bar{\eta}^{(2)})^{\mathrm{T}} + \bar{\eta}^{(1)} H^{\mathrm{T}})$ with

$$\mathrm{D}\pi[(G,H)](\bar{\eta}) := \mathrm{D}\pi[(G,H)](\mathrm{grad}\bar{f}\,(\bar{x})) = \mathrm{grad} f\,(X)\,.$$

The equation (5.20) is obtained as follows. Let $X := U\Sigma V^{\mathrm{T}}$ denote the SVD of $X \in \mathcal{M}_k$, where where $U \in \mathrm{St}(m,k)$ and $V \in \mathrm{St}(n,k)$ and $\Sigma \in \mathbb{R}^{k\times k}$. Then we use the fact that there exists $F \in \mathrm{GL}\,(k)$, for any $(G,H) \in \pi^{-1}(X)$, such that $G = U\Sigma_G F^{\mathrm{T}}$ and $H = V\Sigma_H F^{-1}$, where Σ_G and Σ_H are k-by-k diagonal matrices such that $\Sigma_G\Sigma_H = \Sigma$. Therefore, ϕ_1 in the last term of the right-hand side of (5.19) reads:

$$\begin{aligned}
\phi_1 &= \nabla f(X) V \Sigma_H F^{-1}(F\Sigma_H^{-2}F^{\mathrm{T}})(F^{-T}\Sigma_G^{-2}F^{-1})F\Sigma_G U^{\mathrm{T}}\nabla f(X) \\
&= \nabla f(X) V \Sigma_H (\Sigma_H^{-2}\Sigma_G^{-2})\Sigma_G U^{\mathrm{T}}\nabla f(X) \\
&= \nabla f(X) V \Sigma_H^{-1}\Sigma_G^{-1} U^{\mathrm{T}}\nabla f(X) \\
&= \nabla f(X) X^\dagger \nabla f(X).
\end{aligned}$$

□

The characterization in Proposition 5.2.6 is *not* a computational component of the RGD algorithm (see Algorithm 5.2.1) but is an update rule on $\mathcal{M}_k$ induced by the algorithm. This quotient manifold-based RGD algorithm produces iterates in the total space $\overline{\mathcal{M}_k}$. More interestingly, the result above shows explicitly that the algorithm is invariant on equivalence classes (5.3),

and does not rely on metric projection to maintain the rank constraint.

The invariance property revealed in Proposition 5.2.6 and Lemma 5.2.8 enables us to qualify these algorithms as Riemannian descent algorithms on the quotient space. The explicit form in Lemma 5.2.8 about the induced sequence of this algorithm allows for the local convergence analysis on $\mathcal{M}_k$, which is substantially different and simpler than with matrix factorization on the total space $\overline{\mathcal{M}_k}$; see the next section.

5.3 Main results for fixed-rank matrix optimization

In this section, we investigate the quotient manifold-based algorithm for solving low-rank matrix optimization problems. We focus on a class of problems with a quadratic objective function. Due to the rank constraint, these problems are nonconvex. However, the quadratic objective function of these problems usually possess nice properties around critical points on the rank-constrained search space. Such nonconvex problems appear in various forms in applications like compressed sensing (e.g., [WCCL16]), matrix completion (e.g., [SL16]), trace regression and phase retrieval [LHLZ21].

Starting from recent advances about quadratic functions satisfying a certain restricted positive definiteness (RPD) property [UV20b], we exploit the RPD properties of quadratic objective functions in the context of fixed-rank optimization (5.6), and then propose novel results about the local convergence behavior of Algorithm 5.2.1.

5.3.1 Low-rank matrix optimization problems

Let $\mathcal{A} : \mathbb{R}^{m\times n} \mapsto \mathbb{R}^{m\times n}$ be a linear operator that is symmetric and positive semidefinite, and let $B^\star = \mathcal{A}(M^\star)$ be a reference data matrix, where $M^\star$ is a (partially) hidden real-valued $m \times n$ matrix. We define f as the following quadratic function

$$f(X) := \frac{1}{2}\langle X, \mathcal{A}(X)\rangle - \langle B^\star, X\rangle, \tag{5.21}$$

where $\langle\cdot,\cdot\rangle$ denotes the Frobenius inner product of two matrices.

This function appears as the objective function of many matrix recovery problems [D$^+$06, CR09b, FCRP08, SL16, CWW17], as in the examples below. In these matrix recovery problems, the given matrix $B^\star$ corresponds to the data acquired from a hidden (or partially hidden) matrix $M^\star$, through an observation system modeled by the linear operator $\mathcal{A}$.

Example 5.3.1 (Matrix sensing)**.** *Compressed sensing refers to the problem of recovering a data matrix $M^\star$ from its observations $b^\star$ through a matrix sensing operator, $\Phi : \mathbb{R}^{m\times n} \mapsto \mathbb{R}^d$, where $b^\star := \Phi(M^\star)$. This matrix recovery problem*

can be formulated as the minimization of the following quadratic function

$$f(X) = \frac{1}{2}(\|\Phi(X) - b^\star\|_2^2 - \|b^\star\|_2^2),$$

which can be written in the form of (5.21)*, with* $\mathcal{A} = \Phi^*\Phi$ *and* $B^\star = \Phi^* b^\star$.

Example 5.3.2 (Matrix completion)**.** *For matrix completion, e.g., in Example 5.2.2, the recovery of a hidden matrix from a few of its entries can be formulated by minimizing the following quadratic function*

$$f(X) = \frac{1}{2p}(\|P_\Omega(X - M^\star)\|_{\mathrm{F}}^2 - \|P_\Omega(M^\star)\|_{\mathrm{F}}^2), \tag{5.22}$$

where Ω *is the index set of the known entries,* $p = |\Omega|/mn$*, and* $P_\Omega : \mathbb{R}^{m\times n} \mapsto \mathbb{R}^{m\times n}$ *is the projection operator that retains only the entries on* Ω *and projects all other entries to zero. In this case,* $\mathcal{A} = \frac{1}{p}P_\Omega$ *and* $B^\star = \frac{1}{p}P_\Omega(M^\star)$.

When the hidden matrix $M^\star$ has a low rank that equals k, then naturally f admits $M^\star$ as a global minimum on the low-rank manifold $\mathcal{M}_k$. This scenario is often referred to as the *noiseless case* in the low-rank matrix recovery problems.

The following definition (e.g., [UV20b]), depending on a rank parameter k and a bounded positive constant, reveals an important property of the quadratic function f that can be satisfied by the objective functions of some aforementioned matrix recovery problems.

Definition 5.3.3 (RPD property [UV20b, Definition 3.1])**.** *Let* $\mathcal{A} : \mathbb{R}^{m\times n} \mapsto \mathbb{R}^{m\times n}$ *be a linear operator and let* r *be an integer such that* $0 < r \leq \min(m, n)$*. The operator* $\mathcal{A}$ *satisfies the* (β, r)-RPD *property on* $\mathcal{M}_{\leq r}$ *if there exists* $0 \leq \beta < 1$ *such that*

$$(1 - \beta)\|Z\|_{\mathrm{F}}^2 \leq \langle Z, \mathcal{A}(Z)\rangle \leq (1 + \beta)\|Z\|_{\mathrm{F}}^2, \tag{5.23}$$

for all matrices $Z \in \mathcal{M}_{\leq r}$*. The smallest nonnegative number* β *satisfying the property above is called the* RPD constant.

In the literature of compressed sensing, the RPD property appears under the name of the RIP condition, and this condition can be satisfied with overwhelmingly high probability for a large family of random measurement matrices, for example, the normalized Gaussian and Bernoulli matrices [CP11, RFP10, WCCL16].

Critical points of low-rank matrix problems

A critical point $X^\star$ of the quadratic function f (5.21) in the low-rank space $\mathcal{M}_{\leq k}$ is characterized given in [SU15] and [UV20b, Proposition 2.4]: either $X^\star$ is a critical point of f on $\mathcal{M}_k$, or $X^\star$ is a solution to the optimality condition $\mathcal{A}(X) - B^\star = 0$.

In the general cases where $\mathcal{A}$ is a positive semidefinite operator with a

nontrivial kernel, i.e., an operator such that the matrix equation $\mathcal{A}(X) = B^\star$ corresponds to a linear subspace of $\mathbb{R}^{m\times n}$, there are possibly more than one point in $\mathcal{M}_{\leq k}$ that satisfy either of the above two characterizations. Interestingly, recent advances in (low-rank) matrix recovery theories have proven uniqueness results about the matrix $M^\star$: for example, [UV20b, Theorem 3.5] shows that under certain RPD properties (requiring an upper bound on the RPD constant): (i) the optimality equation $\mathcal{A}(X) = B^\star$ admits $M^\star$ as the unique solution (ii) $M^\star$ is the unique global minimum of f and (iii) any critical point of f on $\mathcal{M}_k$ different than $M^\star$ is either a saddle point or local minimum. More precisely, this theorem uses the quantification of the lower spectral bound of the Riemannian Hessian of f around its critical points on $\mathcal{M}_k$.

Note that the *uniqueness* result about the rank-k matrix $M^\star$ [UV20b, Theorem 3.5] implies, in particular, that one can use Riemannian descent algorithms on $\mathcal{M}_k$ to find the global minimum $M^\star$, among the critical points of f on $\mathcal{M}_k$. One can also deduce local convergence results around $M^\star$ from this uniqueness result: given that $M^\star$ is the unique global minimum hence a local minimum on $\mathcal{M}_k$, the local convergence rate of Riemannian gradient descent to $M^\star$ is linear [AMS08, Theorem 4.5.6].

We make the point above clearer next (in Section 5.3.2) and in a less strict sense. More precisely, we take an interest in a question other than the uniqueness of $M^\star$ as a local minimum of f on $\mathcal{M}_k$. Instead, we focus on the properties of f around $M^\star$ without requiring it to be *the unique* local minimum. For the purpose of local convergence analysis, we formally describe a property of f (5.6) in the following assumption in terms of the spectral bounds of the operator $\mathcal{A}$ centered on a matrix $M^\star \in \mathcal{M}_k$, which is related to but different than the RPD property.

Assumption 5.3.4. *Given $M^\star \in \mathcal{M}_k$ and $0 \leq \beta < 1$, the inequality*

$$(1-\beta)\|X - M^\star\|_{\mathrm{F}}^2 \leq \langle X - M^\star, \mathcal{A}(X - M^\star)\rangle \leq (1+\beta)\|X - M^\star\|_{\mathrm{F}}^2 \quad (5.24)$$

holds for all matrices $X \in \mathcal{M}_k$. The smallest number $\beta \geq 0$ satisfying (5.24) *is referred to as the* RPD *constant of* $(\mathcal{A}, M^\star)$.

The assumption above can be seen as a consequence of the RPD property (5.23) with suitable rank parameters. The proposition below provides a sufficient condition for Assumption 5.3.4 to hold in view of the RPD property of $\mathcal{A}$:

Proposition 5.3.5. *Assumption 5.3.4 holds for any $M^\star \in \mathcal{M}_k$ if $\mathcal{A}$ satisfies the $(\beta, 2k)$-*RPD *property.*

Proof. This is because $(X - M^\star) \in \mathcal{M}_{\leq 2k}$ for any X and $M^\star$ in $\mathcal{M}_k$. □

In the proposition above, the $(\beta, 2k)$-RPD property is strong enough for Assumption 5.3.4 to hold without any requirement on the properties of $M^\star$ (other than $M^\star \in \mathcal{M}_k$).

In the development of our main results (Section 5.3.2), we use Assumption 5.3.4 (i.e., (5.24)) instead of the RPD property because Assumption 5.3.4 offers a condition that is more straightforward for the local convergence analysis. Note again that in contrast to the result in [UV20b, Theorem 3.5], our local convergence analysis does not investigate nor require the uniqueness of $M^\star$ as a local minimum of f on $\mathcal{M}_k$.

From Proposition 5.3.5, Assumption 5.3.4 can be satisfied in the compressed sensing problem, as mentioned before. In the context of matrix completion, where $\mathcal{A} = P_\Omega$ (see Example 5.3.2), the inequality (5.24) in Assumption 5.3.4 is generally not satisfied for all $X \in \mathcal{M}_k$ but a slightly more restrictive version of it is an important intermediate result for establishing exact recovery and convergence results of some matrix factorization-based algorithms. In fact, the sampling operator P_Ω generally does not satisfy the RPD property (Definition 5.3.3); see [CR09b, §1.1.1] for examples of very sparse (rank-1) matrices that cannot be recovered from a sample of its entries, and there is little chance that the inequality (5.24) is satisfied around such matrices. However, it is also shown in [CR09b] that the inequality (5.24) can be satisfied if $M^\star$ is sufficiently *incoherent*. Since the incoherence is a natural feature of data matrices in many practical problems, the inequality (5.24) is a reasonable and useful property; see Section 5.4 for more details, where a more restricted version of Assumption 5.3.4 is validated.

5.3.2 Local convergence analysis

In this subsection, we propose some key results about properties of the quadratic function f around the hidden matrix $M^\star$ based on Assumption 5.3.4. When Assumption 5.3.4 holds (with $0 \leq \beta < 1$), the quadratic function f (5.21) is well-conditioned and has a similar landscape as the function $X \mapsto \frac{1}{2}(\|X - B^\star\|_{\mathrm{F}}^2 - \|B^\star\|_{\mathrm{F}}^2)$, which is strongly convex in the ambient matrix space $\mathbb{R}^{m\times n}$. Interestingly, we show that some nice properties of f in the ambient space $\mathbb{R}^{m\times n}$ are preserved on the fixed-rank manifold. Based on these properties, we provide results about local convergence properties of the proposed algorithm and describe a region of attraction for the algorithm.

The following lemma is a basic property of the Euclidean gradient of f on the vector space $\mathbb{R}^{m\times n}$.

Lemma 5.3.6. *The quadratic function f defined in* (5.21) *has Lipschitz continuous gradient on $\mathcal{M}_{\leq k}$, with a Lipschitz constant $L > 0$. In particular, for matrices $X, Y \in \mathcal{M}_k$,*

$$\|\nabla f(X) - \nabla f(Y)\|_{\mathrm{F}} \leq L\|X - Y\|_{\mathrm{F}}. \tag{5.25}$$

Proof. The inequality (5.25) holds since f (5.21) has Lipschitz-continuous gradient in $\mathbb{R}^{m\times n}$, which is true since it is twice-differentiable and is composed of quadratic terms. □

The magnitude of L in Lemma 5.3.6 depends on $\mathcal{A}$ and k. For example, given that $\mathcal{A}$ is symmetric and positive semidefinite, if $\mathcal{A}$ satisfies the $(\beta_{2k}, 2k)$-RPD property (5.23), it follows that

$$\|\nabla f(X) - \nabla f(Y)\|_{\mathrm{F}} = \sqrt{\langle \mathcal{A}^2(X-Y), X-Y\rangle} \leq (1+\beta_{2k})\|X-Y\|_{\mathrm{F}}, \quad (5.26)$$

which entails that the Lipschitz constant $L \leq 1 + \beta_{2k}$.

Next, in the noiseless case, i.e., the hidden matrix $M^\star$ has a low rank k, we investigate the Riemannian Hessian of f (5.21) at $M^\star$.

Lemma 5.3.7. *Let f be a quadratic function defined in* (5.21)*, where the hidden matrix $M^\star$ has a low rank k. If $M^\star$ satisfies Assumption 5.3.4, i.e., the* RPD *inequality* (5.24)

$$(1-\beta)\|X-M^\star\|_{\mathrm{F}}^2 \leq \langle \mathcal{A}(X-M^\star), X-M^\star\rangle \leq (1+\beta)\|X-M^\star\|_{\mathrm{F}}^2$$

holds for all $X \in \mathcal{M}_k$, with a RPD *constant $0 \leq \beta < 1$, then the Riemannian Hessian of f at $M^\star$ is positive definite:*

$$\lambda_{\min}(f) := \min_{W \in T_{M^\star}\mathcal{M}_k} \left(\frac{\langle \mathrm{Hess} f(M^\star)[W], W\rangle}{\|W\|_{\mathrm{F}}^2} \right) \geq 1 - \beta > 0. \quad (5.27)$$

The proof of this lemma is given in Section 5.3.3.

The result above is very useful in determining the local convergence rate of a Riemannian algorithm that deals with fixed-rank matrices. Let $\{\bar{x}_t\}_{t\geq 0}$ be a sequence generated by Algorithm 5.2.1 and let $\{X_t\}_{t\geq 0}$ denote the induced sequence such that $X_t = \pi(\bar{x}_t)$.

Theorem 5.3.8. *Let f be a quadratic function defined in* (5.21)*, where the hidden matrix $M^\star$ has a low rank k. Suppose that $M^\star$ satisfies Assumption 5.3.4. Then, if the sequence $\{X_t\}_{t\geq 0}$ induced by Algorithm 5.2.1 converges to $M^\star$, the local convergence rate of $\{X_t\}_{t\geq 0}$ is linear.*

Proof. Under Assumption 5.3.4, the result of Lemma 5.3.7 holds, i.e., the Riemannian Hessian of f at $M^\star$ is positive definite; see (5.27). Therefore, according to [AMS08, Theorem 4.5.6], if $\{X_t\}_{t\geq 0}$ converges to $M^\star$, the convergence rate of $\{X_t\}_{t\geq 0}$ is linear. □

To state the convergence rate above, we need to assume that the sequence converges to $M^\star$, because the sole Assumption 5.3.4 does not rule out the existence of other critical points of f on $\mathcal{M}_k$ different than $M^\star$, hence, one cannot rule out the case where the sequence $\{X_t\}_{t\geq 0}$ converges to a different point. Also, it is possible that the sequence does not admit an accumulation point, due to the openness of $\mathcal{M}_k$. The similar assumption is also seen in the related work, e.g., [SU15, 2.4.4].

Therefore, we take an interest in the question of uniqueness of $M^\star$ as a critical point of f in a certain neighborhood of it on $\mathcal{M}_k$. To approach this question, still in the noiseless case, we investigate the Riemannian gradient of

f (as in (5.14)) in a neighborhood of $M^\star$ defined as follows,

$$\mathcal{B}^\star(\delta) = \{X \in \mathcal{M}_k : \|X - M^\star\|_{\mathrm{F}} \leq \delta\}. \tag{5.28}$$

In the following proposition and corollary, we provide a sufficient condition under which $M^\star$ is the unique critical point in $\mathcal{B}^\star(\delta)$. Moreover, we will show that $\mathcal{B}^\star(\delta)$ is a region of contraction regarding the RGD update rule in Algorithm 5.2.1. The proof strategy of Corollary 5.3.10 is inspired by [Nes04, Theorem 2.1.14], which explains the linear convergence of gradient descent for the minimization of a strongly convex function. In the case of this section, we study the function f on $\mathcal{M}_k$ under Assumption 5.3.4, which yields properties similar to "strong convexity", in a weaker sense in $\mathcal{B}^\star(\delta)$. We clarify this property in the next proposition.

Proposition 5.3.9. *Let f be a quadratic function defined in* (5.21)*, where the hidden matrix $M^\star$ has a low rank k. Suppose that $M^\star$ satisfies Assumption 5.3.4, i.e., the RPD inequality* (5.24) *holds for all $X \in \mathcal{M}_k$, with a RPD constant $0 \leq \beta < 1$. Then for any constant δ satisfying*

$$0 \leq \delta < (1 - \beta)\frac{\sigma^\star_{\min}}{L}, \tag{5.29}$$

it holds that

$$\langle \mathrm{grad} f(X), X - M^\star \rangle \geq \tilde{\nu}\|X - M^\star\|_{\mathrm{F}}^2, \tag{5.30}$$

for all $X \in \mathcal{B}^\star(\delta)$, with the constant $\tilde{\nu} = 2\left(1 - \beta - \frac{\delta L}{\sigma^\star_{\min}}\right) > 0$, where L is the Lipschitz constant of f defined in Lemma 5.3.6 and the constant $\sigma^\star_{\min} := \sigma_{\min}(M^\star) > 0$.

The proof of this proposition is given in Section 5.3.3. The technique of the proof is based on the novel results of this chapter (see Section 5.2.3 and Lemma 5.3.11) and a useful lemma ([WCCL16, Lemma 4.1]) about the orthogonal projection (2.16a) on $\mathcal{M}_k$.

Proposition 5.3.9 reveals that the growth of f around $M^\star$ is similar to that of the Euclidean distance function $X \mapsto \|X - M^\star\|_{\mathrm{F}}^2$ in the region $\mathcal{B}^\star(\delta)$ around $M^\star$, provided that Assumption 5.3.4 holds. In view of this remark, we have the following corollary.

Corollary 5.3.10. *Under the statement of Proposition 5.3.9: (i) The hidden matrix $M^\star$ is the unique critical point of f in the region $\mathcal{B}^\star(\delta)$* (5.28)*, for any radius δ satisfying* (5.29)*.*
(ii) For any radius δ satisfying (5.29)*, the region $\mathcal{B}^\star(\delta)$ is* stable *by the RGD update rule with the Riemannian gradient* (5.14)*, given a bounded stepsize: for any $t \geq 1$, if $X_t \in \mathcal{B}^\star(\delta)$* (5.28)*, then the induced RGD update X_{t+1}* (5.17) *satisfies*

$$\|X_{t+1} - M^\star\|_{\mathrm{F}} \leq \sqrt{\kappa(\theta)}\|X_t - M^\star\|_{\mathrm{F}}, \tag{5.31}$$

with $\kappa(\theta) = 1 - \theta(2\tilde{\nu} - \tilde{C}_\delta\theta) \in (0,1)$ *for any stepsize* $0 < \theta < \min(1, 2\tilde{\nu}\tilde{C}_{\delta,t}^{-1})$, *where* $\tilde{C}_{\delta,t} = 4L^2 + \delta\frac{L^2(4L+2+\delta)}{\sigma_{\min}(X_t)} > 0$ *and* $\tilde{\nu} > 0$ *is the constant given in Proposition 5.3.9.*

Proof. (i): First, $M^\star$ is a critical point (and also a global minimum) of f in $\mathcal{B}^\star(\delta)$. Moreover, if δ satisfies (5.29), the inequality (5.30) in Proposition 5.3.9 holds for all $X \in \mathcal{B}^\star(\delta)$. Now assume that there exists another critical point $X' \in \mathcal{B}^\star(\delta)$ and $X' \neq M^\star$, then from the inequality (5.30), one has

$$\langle \mathrm{grad} f\,(X')\,, X' - M^\star\rangle \geq \tilde{\nu}\|X' - M^\star\|_{\mathrm{F}}^2 > 0.$$

But the left hand-side of the above inequality equals 0 (since X' being a critical point of f on $\mathcal{M}_k$ implies $\mathrm{grad} f\,(X') = 0$), which entails a contradiction. Therefore, $M^\star$ is the unique critical point in $\mathcal{B}^\star(\delta)$.

(ii): Let (5.17) be denoted as $X_{t+1} = X_t - \theta\mathrm{grad} f\,(X_t) + \theta^2\Gamma_{X_t}$. We have

$$\begin{aligned}&\|\mathrm{grad} f\,(X_t)\,\|_{\mathrm{F}} \leq C_1\|X_t - M^\star\|_{\mathrm{F}} \text{ and } \|\Gamma_{X_t}\|_{\mathrm{F}} \leq C_2\|X_t - M^\star\|_{\mathrm{F}}^2\\ &\text{for } C_1 = 2L > 0 \text{ and } C_2 = L^2\,\sigma_{\min}(X_t)^{-1} > 0.\end{aligned} \tag{5.32}$$

Indeed, the Riemannian gradient of f (5.21) $\mathrm{grad} f\,(X) = \mathcal{G}_X(\nabla f(X)) = \mathcal{G}_X(\mathcal{A}(X - M^\star))$ satisfies $\|\mathrm{grad} f\,(X)\,\|_{\mathrm{F}} = \|\mathcal{G}_X(\mathcal{A}(X - M^\star))\|_{\mathrm{F}} \leq 2\|\,\mathcal{A}(X - M^\star)\|_{\mathrm{F}} \leq 2L\|X - M^\star\|_{\mathrm{F}}$, according to Lemma 5.3.11 (spectral bounds of $\mathcal{G}_X$) and Lemma 5.3.6. Similarly, $\|\Gamma_X\|_{\mathrm{F}} \leq \|X^\dagger\|_2\|\nabla f(X)\|_{\mathrm{F}}^2 \leq \sigma_{\min}(X)^{-1}L^2\|X - M^\star\|_{\mathrm{F}}^2 \leq C_2\|X - M^\star\|_{\mathrm{F}}^2$.

Based on Proposition 5.3.9 and (5.32), it holds that, for $X_t \in \mathcal{B}^\star(\delta)$ and any stepsize $\theta \in (0,1)$,

$$\begin{aligned}\|X_{t+1} - M^\star\|_{\mathrm{F}}^2 &= \|X_t - M^\star\|_{\mathrm{F}}^2 - 2\theta\,\langle \mathrm{grad} f\,(X_t)\,, X_t - M^\star\rangle + 2\theta^2\,\langle \Gamma_{X_t}, X_t - M^\star\rangle\\ &\qquad + \theta^2\|\mathrm{grad} f\,(X_t) - \theta\Gamma_{X_t}\|_{\mathrm{F}}^2\\ &\leq \|X_t - M^\star\|_{\mathrm{F}}^2 - 2\theta\tilde{\nu}\|X_t - M^\star\|_{\mathrm{F}}^2 + 2\theta^2C_2\|X_t - M^\star\|_{\mathrm{F}}^3\\ &\quad + \theta^2(C_1^2\|X_t - M^\star\|_{\mathrm{F}}^2 + C_2^2\theta^2\|X_t - M^\star\|_{\mathrm{F}}^4 + 2C_1C_2\theta\|X_t - M^\star\|_{\mathrm{F}}^3) \quad (5.33)\\ &\leq \left(1 - 2\tilde{\nu}\theta + \theta^2(C_1^2 + 2C_2\delta + 2\theta C_1C_2\delta + \theta^2C_2\delta^2)\right)\|X_t - M^\star\|_{\mathrm{F}}^2, \quad (5.34)\\ &\leq \left(1 - 2\tilde{\nu}\theta + \theta^2(C_1^2 + 2C_2\delta + 2C_1C_2\delta + C_2\delta^2)\right)\|X_t - M^\star\|_{\mathrm{F}}^2, \quad (5.35)\end{aligned}$$

where the second term (with $\tilde{\nu}$) on the right-hand side of (5.33) is obtained through (5.30) in Proposition 5.3.9, and the third and last terms obtained through the Cauchy–Schwarz inequality and (5.32). The terms in δ in (5.34) come directly from the bound $\|X_t - M^\star\|_{\mathrm{F}} \leq \delta$ and the coefficients with θ^2 in (5.35) are simplified from (5.34) in view of the setting $\theta \in (0,1)$ above.

Let the coefficient in (5.35) be denoted as $\kappa(\theta) := 1 - \theta\left(2\tilde{\nu} - \tilde{C}_\delta\theta\right)$, where

$$\tilde{C}_\delta := C_1^2 + \delta(2C_2(1 + C_1) + C_2\delta) \underset{(5.32)}{=} 4L^2 + \delta\frac{L^2(4L + 2 + \delta)}{\sigma_{\min}(X_t)} > 0.$$

We have $0 < \kappa(\theta) < 1$ if and only if $0 < \theta(2\tilde{\nu} - \tilde{C}_\delta\theta) < 1$, which is satisfied when $0 < 2\tilde{\nu} - \tilde{C}_\delta\theta \leq 1$ given that $\theta \in (0, 1)$, hence the conclusion provided that $0 < \theta < \min(1, 2\tilde{\nu}\tilde{C}_\delta^{-1})$. □

The application of the geometrical and local convergence results above are also discussed in Section 5.4, in the specific case of matrix completion. Based on recent advances on recovery guarantees for matrix completion, we characterize a subset of $\mathcal{M}_k$ in which the inequality (5.24) holds and discuss the regularization schemes under which these results are applicable. Numerical experiments for matrix completion in Section 5.5 are given as demonstrations.

5.3.3 Technical lemmas and proofs of the results

We give proofs of the results in Section 5.3.2.

Proof of Lemma 5.3.7.

Proof. Recall that the Riemannian Hessian of f at $Y \in \mathcal{M}_k$, by definition [AMS08, Proposition 5.5.4], satisfies

$$\langle \mathrm{Hess} f(Y)[Z], Z\rangle = \frac{\mathrm{d}^2}{\mathrm{d}t^2}(f(\mathrm{Exp}_Y(tZ)))\Big|_{t=0}, \tag{5.36}$$

where $\mathrm{Exp}_Y : T_Y\mathcal{M}_k \mapsto \mathcal{M}_k$ is the exponential map at Y. We prove the spectral lower bound of $\mathrm{Hess} f(M^\star)$ as follows. First, the exponential map at $Y := M^\star$ has the following expression (e.g., [AM12], [Van13, Proposition A1], [UV20b, Appendix A]),

$$\mathrm{Exp}_{M^\star}(Z) = M^\star + Z + \Delta_{M^\star}(Z), \tag{5.37}$$

where $\Delta_{M^\star}(Z) := (I - \mathcal{P}_{U^\star})ZM^{\star\dagger}Z(I - \mathcal{P}_{V^\star}) + o(\|Z\|_{\mathrm{F}}^3)$ satisfies $\|\Delta_{M^\star}(W)\|_{\mathrm{F}} \lesssim \|Z\|_{\mathrm{F}}^2$. From (5.36)–(5.37), it follows that the quadratic function f (5.21) satisfies, for any $Z \in T_{M^\star}\mathcal{M}_k$,

$$\langle \mathrm{Hess} f(M^\star)[Z], Z\rangle = \langle \mathcal{A}(Z), Z\rangle . \tag{5.38}$$

Next, we bound (5.38) using Assumption 5.3.4 (about $\mathcal{A}$ and $M^\star$) via the quadratic function $\tilde{f} : X \mapsto f(X) - f(M^\star)$. In fact, from the definition of f (5.21), we have

$$\begin{aligned}
\tilde{f}(X) &= \frac{1}{2}\langle \mathcal{A}(X), X\rangle - \langle B^\star, X\rangle - f(M^\star) \\
&= \frac{1}{2}\langle \mathcal{A}(X), X\rangle - \langle \mathcal{A}(M^\star), X\rangle - (-\frac{1}{2}\langle \mathcal{A}(M^\star), M^\star\rangle) \qquad (5.39)\\
&= \frac{1}{2}\langle \mathcal{A}(X - M^\star), X - M^\star\rangle \geq \frac{1-\beta}{2}\|X - M^\star\|_{\mathrm{F}}^2, \qquad (5.40)
\end{aligned}$$

where the equality (5.39) holds due to the fact that $B^\star = \mathcal{A}(M^\star)$ by definition (5.21), the equality in (5.40) holds since $\mathcal{A}$ is a symmetric operator, and the last inequality is a direct result of the inequality (5.24) in Assumption 5.3.4. It follows that, for any $Z \in T_{M^\star}\mathcal{M}_k$ and $X := \mathrm{Exp}_{M^\star}(Z) \in \mathcal{M}_k$,

$$\tilde{f}(\mathrm{Exp}_{M^\star}(Z)) \geq \frac{1-\beta}{2} \| \mathrm{Exp}_{M^\star}(Z) - M^\star \|_{\mathrm{F}}^2, \tag{5.41}$$

which can be rewritten via the expression (5.37) of $\mathrm{Exp}_{M^\star}(Z)$ as follows,

$$\tilde{f}(\mathrm{Exp}_{M^\star}(Z)) = \langle \mathcal{A}(Z), Z \rangle + \varphi_{\tilde{f}}(Z) \geq (1-\beta)\|Z\|_{\mathrm{F}}^2 + \varphi_F(Z), \tag{5.42}$$

where $\varphi_{\tilde{f}}(Z)$ and $\varphi_F(Z)$ are the sums of third- and higher-order terms of Z on the two sides of (5.41), i.e., $|\varphi_{\tilde{f}}(Z)| \leq C_1\|Z\|_{\mathrm{F}}^3$ and $|\varphi_F(Z)| \leq C_2\|Z\|_{\mathrm{F}}^3$, for constants $C_1 > 0$ and $C_2 > 0$.

Finally, combining (5.42) and (5.38), we have, for any $Z \in T_{M^\star}\mathcal{M}_k$,

$$\begin{aligned}\lambda_f(Z) &:= \frac{\langle \mathrm{Hess} f(M^\star)[Z], Z\rangle}{\|Z\|_{\mathrm{F}}^2} = \frac{\langle \mathcal{A}(Z), Z\rangle}{\|Z\|_{\mathrm{F}}^2} \geq (1-\beta) - \frac{|\varphi_{\tilde{f}}(Z)| + |\varphi_F(Z)|}{\|Z\|_{\mathrm{F}}^2} \\ &\geq (1-\beta) - (C_1 + C_2)\,\|Z\|_{\mathrm{F}},\end{aligned}$$

which entails that

$$\lambda_f(Z) \geq \sup_{Z \in T_{M^\star}\mathcal{M}_k} (1 - \beta - (C_1 + C_2)\|Z\|_{\mathrm{F}}) = 1 - \beta, \tag{5.43}$$

where the supremum of the right-hand side is obtained with $W \in T_{M^\star}\mathcal{M}_k$ in a certain direction, such that $\|W\|_{\mathrm{F}} \to 0$. □

The following two lemmas provide some useful properties of the operator $\mathcal{G}_X$ that relates the Euclidean gradient of f to its Riemannian gradient through (5.15). They will be used in the proof of Proposition 5.3.9 that follows.

Lemma 5.3.11. *Let X be a matrix in $\mathcal{M}_k$ and let $X := U\Sigma V^{\mathrm{T}}$ denotes its rank-k SVD. The linear operator $\mathcal{G}_X : \mathbb{R}^{m\times n} \mapsto T_X\mathcal{M}_k$ defined in (5.15) satisfies the following properties.*

(i): $\mathcal{G}_X$ is symmetric and positive semidefinite. In particular, $\mathcal{G}_X(Z) = 2Z$ if $Z \in T_X\mathcal{M}_k$, and $\mathcal{G}_X(Z) = 0$ if $Z \in (T_X\mathcal{M}_k)^\perp$.

(ii): Let X, Y be two matrices in $\mathcal{M}_k$, and let $(\mathcal{G}_X - 2\,\mathrm{I})(Z) := \mathcal{G}_X(Z) - 2Z$ denote the evaluation of the operator $(\mathcal{G}_X - 2\,\mathrm{I})$ at $Z \in \mathbb{R}^{m\times n}$. It holds that

$$(\mathcal{G}_X - 2\,\mathrm{I})(X - Y) = 2(\mathrm{I} - \mathrm{P}_{T_X\mathcal{M}_k})(Y). \tag{5.44}$$

Proof. (i): From the definition (5.15), $\mathcal{G}_X(Z) = \mathcal{P}_U(Z) + \mathcal{P}_V(Z)$, we deduce that $\mathcal{G}_X$ is a symmetric operator, since the orthogonal projections $\mathcal{P}_U(\cdot)$ and $\mathcal{P}_V(\cdot)$ are symmetric operators. We prove the remaining claims as follows. From (5.16) in Remark 5.2.7, we deduce that $\mathcal{G}_X(Z) = \mathrm{P}_{T_X\mathcal{M}_k}(Z) +$

$UU^{\mathrm{T}}ZVV^{\mathrm{T}} = 2Z$ if $Z \in T_X\mathcal{M}_k$. If $Z' \in (T_X\mathcal{M}_k)^{\perp}$, then there exists $Y \in \mathbb{R}^{m\times n}$ such that $Z' = (\mathrm{I} - \mathrm{P}_{T_X\mathcal{M}_k})(Y)$, and consequently

$$\mathcal{G}_X(Z') = \mathrm{P}_{T_Y\mathcal{M}_k}((\mathrm{I} - \mathrm{P}_{T_X\mathcal{M}_k})(Y)) + UU^{\mathrm{T}}((\mathrm{I} - \mathrm{P}_{T_X\mathcal{M}_k})(Y))VV^{\mathrm{T}} = 0.$$

Therefore, for any $Z \in \mathbb{R}^{m\times n}$, $\mathcal{G}_X(Z) = \mathcal{G}_X(\mathrm{P}_{T_X\mathcal{M}_k}(Z)) = 2\,\mathrm{P}_{T_X\mathcal{M}_k}(Z)$, which entails that $0 \leq 2\|\,\mathrm{P}_{T_X\mathcal{M}_k}(Z)\|_{\mathrm{F}}^2 = \langle \mathcal{G}_X(Z), Z\rangle \leq 2\|Z\|_{\mathrm{F}}^2$.

(ii): We prove the equality (5.44) as follows. The matrix $Z := Y - X \in \mathbb{R}^{m\times n}$ can be decomposed as $Z = \tilde{Z} + \Delta_Z$, where $\tilde{Z} := \mathrm{P}_{T_X\mathcal{M}_k}(Z)$ and $\Delta_Z := (\mathrm{I} - \mathrm{P}_{T_X\mathcal{M}_k})(Z) = (\mathrm{I} - \mathrm{P}_{T_X\mathcal{M}_k})(Y)$, where the last equality holds since $(\mathrm{I} - \mathrm{P}_{T_X\mathcal{M}_k})(X) = 0$. Therefore, we have

$$\begin{aligned}(\mathcal{G}_X - 2\,\mathrm{I})(X - Y) &= -(\mathcal{G}_X - 2\,\mathrm{I})(\tilde{Z} + \Delta_Z)\\ &= (2\,\mathrm{I} - \mathcal{G}_X)(\Delta_Z) = 2\Delta_Z := 2(\mathrm{I} - \mathrm{P}_{T_X\mathcal{M}_k})(Y),\end{aligned} \tag{5.45}$$

where the equalities in (5.45) are obtained by using the fact that $\tilde{Z} \in T_X\mathcal{M}_k$ and $\Delta_Z \in (T_X\mathcal{M}_k)^{\perp}$ and the properties $\mathcal{G}_{X\,|T_X\mathcal{M}_k} = 2\,\mathrm{I}$ and $\mathcal{G}_{X\,|(T_X\mathcal{M}_k)^{\perp}} = 0$, proven in (i). □

The properties of $\mathcal{G}_X$ listed above are related to the orthogonal projection operator $\mathrm{P}_{T_X\mathcal{M}_k}$. The following lemma in the related work about this projection operator will be used in the proof of Proposition 5.3.9.

Lemma 5.3.12 ([WCCL16], Lemma 4.1)**.** *Let X and Y be two matrices in $\mathcal{M}_k$. Then it holds that*

$$\|(\mathrm{I} - \mathrm{P}_{T_Y\mathcal{M}_k})(X)\|_{\mathrm{F}} \leq \frac{1}{\sigma_{\min}(X)}\|X - Y\|_{\mathrm{F}}^2. \tag{5.46}$$

We refer to [WCCL16, §4.1] for the proof of the lemma above.

Proof of Proposition 5.3.9.

Proof. We prove the inequality (5.30) as follows. First, through Proposition 5.2.6, the Riemannian gradient of f (5.21) is

$$\mathrm{grad} f\,(X) = \mathcal{G}_X(\nabla f(X)) = \mathcal{G}_X(\mathcal{A}(X - M^{\star})), \tag{5.47}$$

for all $X \in \mathcal{M}_k$. Therefore, we have

$$\begin{aligned}&\langle \mathrm{grad} f(X), X - M^{\star}\rangle = \langle \mathcal{G}_X(\mathcal{A}(X - M^{\star})), X - M^{\star}\rangle\\ &= \langle \mathcal{A}(X - M^{\star}), \mathcal{G}_X(X - M^{\star})\rangle &(5.48)\\ &= 2\,\langle \mathcal{A}(X - M^{\star}), X - M^{\star}\rangle + \langle \mathcal{A}(X - M^{\star}), (\mathcal{G}_X - 2\,\mathrm{I})(X - M^{\star})\rangle\\ &= 2\,\langle \mathcal{A}(X - M^{\star}), X - M^{\star}\rangle + 2\,\langle \mathcal{A}(X - M^{\star}), (\mathrm{I} - \mathrm{P}_{T_X\mathcal{M}_k})(M^{\star})\rangle &(5.49)\\ &\geq 2\,\underbrace{\langle \mathcal{A}(X - M^{\star}), X - M^{\star}\rangle}_{a_1} - 2\,\underbrace{\|\,\mathcal{A}(X - M^{\star})\|_{\mathrm{F}}}_{a_2}\,\underbrace{\|(\mathrm{I} - \mathrm{P}_{T_X\mathcal{M}_k})(M^{\star})\|_{\mathrm{F}}}_{a_3}, &(5.50)\end{aligned}$$

where (5.48) holds since $\mathcal{G}_X$ is a symmetric operator, (5.49) is obtained by noticing that $(\mathcal{G}_X - 2\,\mathrm{I})(X - M^\star) = 2(\mathrm{I} - \mathrm{P}_{T_X\mathcal{M}_k})(M^\star)$; see (5.44) in Lemma 5.3.11. The terms a_1 and a_2 in (5.50) have the following bounds,

$$a_1 := \langle \mathcal{A}(X - M^\star), X - M^\star \rangle \geq (1-\beta)\|X - M^\star\|_{\mathrm{F}}^2, \tag{5.51}$$
$$a_2 := \|\,\mathcal{A}(X - M^\star)\|_{\mathrm{F}} \leq L\|X - M^\star\|_{\mathrm{F}}, \tag{5.52}$$

where (5.51) is the result of (5.24), since Assumption 5.3.4 holds for $M^\star$, and (5.52) holds according to Lemma 5.3.6 (recall that $\mathcal{A}(X - M^\star) = \nabla f(X) = \nabla f(X) - \nabla f(M^\star)$). From (5.46) in Lemma 5.3.12, the term a_3 in (5.50) has the following bound,

$$a_3 := \|(\mathrm{I} - \mathrm{P}_{T_X\mathcal{M}_k})(M^\star)\|_{\mathrm{F}} \leq \frac{1}{\sigma_{\min}^\star}\|X - M^\star\|_{\mathrm{F}}^2, \tag{5.53}$$

where $\sigma_{\min}^\star := \sigma_{\min}(M^\star)$. Applying (5.51)–(5.53) to (5.50), we have

$$\langle \mathrm{grad} f(X), X - M^\star \rangle \geq 2\Big((1-\beta) - \frac{L}{\sigma_{\min}^\star}\|X - M^\star\|_{\mathrm{F}}\Big)\|X - M^\star\|_{\mathrm{F}}^2. \tag{5.54}$$

Note that the coefficient in the right-hand side of (5.54) is strictly positive if $X \in \mathcal{M}_k$ satisfies $\|X - M^\star\|_{\mathrm{F}} \leq \delta$, where δ is a radius satisfying $0 \leq \delta < (1-\beta)\frac{\sigma_{\min}^\star}{L}$, which proves the condition (5.29). In conclusion, for any radius δ satisfying (5.29),

$$\begin{aligned}\langle \mathrm{grad} f(X), X - M^\star \rangle &\geq 2\Big((1-\beta) - \frac{L}{\sigma_{\min}^\star}\|X - M^\star\|_{\mathrm{F}}\Big)\|X - M^\star\|_{\mathrm{F}}^2 \\ &\geq 2\left(1 - \beta - \frac{\delta L}{\sigma_{\min}^\star}\right)\|X - M^\star\|_{\mathrm{F}}^2,\end{aligned}$$

for all $X \in \mathcal{B}^\star(\delta)$, which concludes the proof. □

5.4 Extension to low-rank matrix completion and discussions

As introduced in Example 5.3.2, the low-rank matrix completion problem can be formulated as an instance of the low-rank quadratic problem. In this section, we summarize a class of matrix completion problem formulations and discuss the conditions on the problem setting—in relation with the matrix subsampling operator and regularization methods—under which the new results (in Section 5.3) are applicable. Numerical experiments are presented to demonstrate the convergence behavior of the proposed algorithms in comparison with existing Euclidean-based and Riemannian algorithms.

5.4.1 Problem setting

Let $M^\star \in \mathbb{R}^{m\times n}$ be a matrix that is known on a part of its entries and let Ω be the index set of the known entries of $M^\star$. We consider the optimization on $\mathcal{M}_k$ of the following matrix completion model,

$$\min_{X\in\mathcal{M}_k} F(X) := f(X) + \psi(X), \tag{5.55}$$

where f is the data fitting function defined in (5.22) and $\psi : \mathbb{R}^{m\times n} \mapsto \mathbb{R}_+$ is a smooth regularization function (an instance of ψ is specified later in (5.59)). The mask operator $P_\Omega : \mathbb{R}^{m\times n} \mapsto \mathbb{R}^{m\times n}$ in (5.22) is an orthogonal projection such that the (i,j)-th entry of $P_\Omega(X)$ is X_{ij} if $(i,j) \in \Omega$ and zero otherwise.

As in Example 5.3.2, the data fitting function f is a quadratic function that fits the definition (5.21) up to a constant scalar $\frac{1}{2p}\|P_\Omega(M^\star)\|_{\mathrm{F}}^2$. The linear operator $\mathcal{A}$ in (5.21) takes the form $\mathcal{A} = \frac{1}{p}P_\Omega{}^* P_\Omega = \frac{1}{p}P_\Omega$.

It is well understood that the subsampling operator P_Ω can be badly conditioned with unbalanced sampling pattern or on matrices that are highly *coherent* [CR09b]. Thus, we consider the standard assumption that $M^\star$ is μ-incoherent, which guarantees that the subsampling operator P_Ω is well-conditioned around $M^\star$.

Definition 5.4.1 (Incoherence [CR09b])**.** *A matrix $Z = U\Sigma V^{\mathrm{T}} \in \mathbb{R}^{m\times n}$ is* μ-incoherent *if*

$$\|U_{i,:}\|_2 \le \sqrt{\frac{\mu k}{m}}, \|V_{j,:}\|_2 \le \sqrt{\frac{\mu k}{n}}, \tag{5.56}$$

for all $(i,j) \in [\![m]\!] \times [\![n]\!]$.

The incoherence constant μ is a bound that measures the maximal row norms of U and V. Since the SVD factor matrix U (respectively V) is orthonormal, such that $\|U\|_{\mathrm{F}}^2 = \sum_{i=1}^m \|U_{i,:}\|_2^2 \equiv k$, the incoherence constant μ is small if the variations of the entries in U and V are moderate; on the contrary, μ is large if U and V has columns with *spikes*. It can be shown that the smallest possible incoherence constant is $\mu = 1$.

5.4.2 The restricted positive definite inequality

As mentioned in Section 5.3.1, the subsampling operator P_Ω generally does not satisfy the RPD property (Definition 5.3.3). However, under certain conditions on the subsampling pattern, the inequality (5.24) in Assumption 5.3.4 can be satisfied in some neighborhood of the hidden matrix $M^\star$, provided that $M^\star$ is sufficiently incoherent. The following set, adapted from [SL16, §3],

$$\mathcal{B}^\star(\delta,\mu) = \mathcal{B}^\star(\delta) \cap \{X \in \mathcal{M}_{\le k} : X \text{ is } \mu\text{-incoherent}\}, \tag{5.57}$$

where $\mathcal{B}^\star(\delta)$ is defined in (5.28), is a neighborhood of $M^\star$ in which the inequality (5.24) holds. More precisely, the following proposition, adapted from a key

result in [SL16], validates the inequality (5.24) on $\mathcal{B}^\star(\delta, \mu)$.

Proposition 5.4.2 ([SL16, Claim 3.1, Lemma 3.1])**.** *Assume that a rank-k matrix $M^\star \in \mathbb{R}^{m\times n}$ is μ-incoherent. Suppose the condition number of $M^\star$ is κ and $\alpha = m/n \geq 1$. Then there exists a numerical constant C_0 such that: if the indices in Ω are uniformly generated with size*

$$|\Omega| \geq C_0 \alpha k \kappa^2 \max(\mu \log(n), \sqrt{\alpha} k^6 \mu^2 \kappa^4),$$

then with probability at least $1 - 2n^{-4}$,

$$\frac{1}{3}\|X - M^\star\|_{\mathrm{F}}^2 \leq \frac{1}{p}\|P_\Omega(X - M^\star)\|_{\mathrm{F}}^2 \leq 2\|X - M^\star\|_{\mathrm{F}}^2, \tag{5.58}$$

for any $X \in \mathcal{B}^\star(\delta, \mu)$ defined in (5.57)*.*

The proof of this proposition is given in [SL16, Appendix J].

As a consequence, the results in Proposition 5.3.9 applies to the quadratic data fitting function f on the set $\mathcal{B}^\star(\delta, \mu) \cap \{X \in \mathcal{M}_k : \sigma_{\min}(X) \geq \bar{\sigma} > 0\}$, for a certain $\delta > 0$ with a given constant $0 < \bar{\sigma} \leq \sigma_{\min}(M^\star)$. Intuitively, $\mathcal{B}^\star(\delta, \mu)$ is a region of attraction around the critical point $M^\star$. Therefore, if the induced sequence $\{X_t\}_{t\geq 0}$ of Algorithm 5.2.1 converges to $M^\star \in \mathcal{M}_k$, its convergence rate is linear, as stated in Theorem 5.3.8.

Regularization and discussions

The conclusion above about the convergence rate using Theorem 5.3.8 is established based on the assumption that the algorithm converges to $M^\star$. The validation of this assumption, in the context of the matrix completion problem setting, is realized by ensuring that the iterates produced by an algorithm remain confined in the set $\mathcal{B}^\star(\delta, \mu)$ defined in (5.57), once entering this neighborhood. It is however not obvious to ensure this since a mere decrease (from one iterate to the next) in the value of f through the RGD update does not necessarily imply that the μ-incoherence condition is preserved.

Let $\mathcal{L}_0 := \{X \in \mathcal{M}_k : f(X) \leq f(X_0)\}$ be a sublevel set, for an initial point $X^0 \in \mathcal{M}_k$. The following criteria about the sublevel set of the regularized objective function F (5.55) ensure that a monotonically decreasing algorithm produces a sequence that belongs to $\mathcal{B}^\star(\delta, \mu)$ (5.57), and therefore provide a way to design the regularization function ψ: (i) there exists δ_0 such that all points in $\mathcal{L}_0$ are δ_0-close to $M^\star$; (ii) for any matrix X in the sublevel set $\mathcal{L}_0$, the matrix $Z = X - M^\star$ is μ-incoherent; (iii) the regularizer satisfies $\psi(X) = 0$ for all X that is δ_1-close to $M^\star$, for some fixed parameter $\delta_1 > 0$.

We refer to [SL16, GLM16] for a thorough view regarding the regularization function ψ. In addition to the regularization schemes in the aforementioned works about recovery theories for matrix completion, we consider an adaptation of the graph Laplacian-based regularization in Chapter 4.

In the same context of Section 4.2 (in the previous chapter), let $(\mathcal{G}^{\mathrm{r}}, \mathcal{G}^{\mathrm{c}})$ be

a given pair of row-wise and column-wise similarity graphs associated with the matrix $M^\star$. We define the regularizer ψ as

$$\psi(X) = \frac{\alpha}{4(mn)^2}\left(T_{\delta_\mathrm{r}}\left(\mathrm{tr}(X^\mathrm{T}\Theta^\mathrm{r}X)\right) + T_{\delta_\mathrm{c}}\left(\mathrm{tr}(X\Theta^\mathrm{c}X^\mathrm{T})\right)\right), \tag{5.59}$$

where $T_{\delta_\mathrm{r}}, T_{\delta_\mathrm{c}}$ are hard-thresholding operators and Θ^r and Θ^c are graph Laplacian-based matrices defined as follows,

$$T_\delta(z) = \max(0, z-\delta)^2, \tag{5.60a}$$

$$\Theta^\mathrm{r} = I_m + \gamma_r L^\mathrm{r} \text{ and } \Theta^\mathrm{c} := I_n + \gamma_c L^\mathrm{c}. \tag{5.60b}$$

As in the related work [SL16, GLM16], the purpose of the hard thresholding function T_δ is to enforce the regularization term only when the associated penalty function exceeds a prescribed value, and to avoid distorting the critical point of f (5.22) from the global minimizer $M^\star$.

In the next section, we apply the regularization scheme (5.59) to (5.55) in numerical experiments.

5.5 Numerical experiments

In this section, we conduct numerical experiments on matrix completion using the proposed algorithms. The matrix completion model is defined according to (5.55). Depending on whether the regularizer ψ in the model 5.55 is active or not ($\psi \equiv 0$), the matrix completion model is referred to as a regularized model or a *non-regularized* model. For the regularized model, we use the regularization scheme in (5.59), such that the regularization term ψ is only effective if the matrix candidate is far from the desired solution.

We carry out matrix completion tasks with both the non-regularized and regularized versions of (5.55) using the proposed algorithms (Algorithm 5.2.1). Details of these algorithms and their Euclidean counterparts are as follows. Several state-of-the-art algorithms for optimization on $\mathcal{M}_k$ are also tested, with details in Section 5.5.2.

The proposed algorithms: Algorithm 5.2.1 is labeled as Qprecon RGD and the RCG algorithm is labeled as Qprecon RCG. The trial stepsizes (Algorithm 5.2.1, line 5.11) are selected by default using line minimization, and the other stepsize rules that are also tested are (i) (Armijo) for the Armijo line search method, and (ii) (RBB) for the Barzilai–Borwein stepsize (5.12).

Euclidean gradient descent (Euclidean GD) and nonlinear conjugate gradient (Euclidean CG) algorithms refer to the GD and CG algorithms using the Euclidean metric on $\overline{\mathcal{M}}_k$ in the definition of the search directions on $\overline{\mathcal{M}}_k$. The stepsize selection rules are the same as the proposed algorithms; these algorithms are implemented along with the proposed algorithms in the source code.

5.5.1 Invariant optimization path on $\mathcal{M}_k$

As shown in Lemma 5.2.8, the algorithm Qprecon RGD on $\overline{\mathcal{M}_k}$ induces a sequence $\{X_t\}_{t\geq 0}$ that moves on $\mathcal{M}_k$ along the Riemannian gradient descent directions in $T\mathcal{M}_k$. This means that the sequence does not depend on the locations of the iterates $\{\bar{x}_t\}_{t\geq 0}$ in the vertical space $\mathcal{V}_{\bar{x}}$. In the following experiment, we demonstrate this remark by observing the iteration histories of Qprecon RGD in matrix completion tests, in comparison with Euclidean GD.

A synthetic low-rank matrix is generated as follows: $M^\star = A^\star B^\star$, where $A \in \mathbb{R}^{m\times k}$ and $B \in \mathbb{R}^{m\times k}$ are composed of entries drawn from the Gaussian distribution $\mathcal{N}(0,1)$, for a rank parameter k that is much smaller than the matrix dimensions. The index set of the observed entries (training data) consists of indices sampled from the Bernoulli distribution $\mathcal{B}(p)$: for $(i,j) \in [\![m]\!] \times [\![n]\!]$,

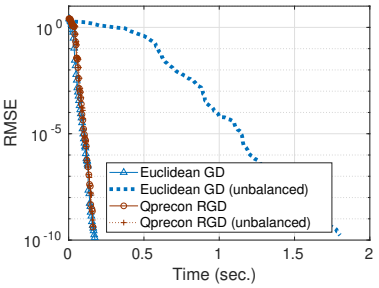
$$(i,j) \in \Omega, \text{ with probability } p. \tag{5.61}$$

Using the non-regularized model, we carry out matrix completion tests on noiseless observations of $M^\star$ on Ω, with a sampling rate $p = 0.8$. Qprecon RGD and Euclidean GD are tested with two different initial points x_0, $x_0' \in \overline{\mathcal{M}_k}$ that belong to the same equivalence class (in the sense that $\pi(x_0) = \pi(x_0')$): In the first test, each algorithm is initialized with $x_0 = (G_0, H_0)$ using the (balanced) spectral initialization method (4.18). In this case, the two factors G_0 and H_0 are balanced, i.e., having equal matrix norms. In the second test, the initial point is defined as $x_0' = (\lambda G_0, H_0/\lambda)$, for $\lambda = 5$. The comparative results are given in Figure 5.2.

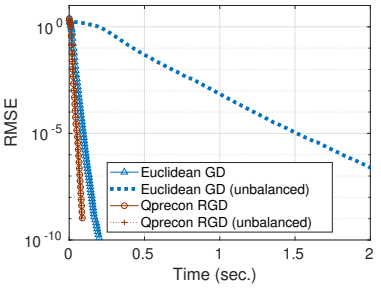
From the results in Figure 5.2, we observe that the two sequences of Qprecon RGD overlap, which shows indeed that the path of the sequence generated by Qprecon RGD does not vary with the change in the initial point. We also observe that these overlapping sequences converge linearly, with much faster speed than Euclidean GD with the unbalanced initial point. In fact, one can see from the figure that the convergence of Euclidean GD is significantly slowed down with the unbalanced initial point x_0' compared to the case with x_0.

5.5.2 Matrix completion performances

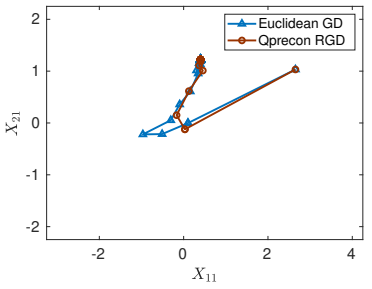
In this subsection, a synthetic matrix $M^\star$ is generated using the graph-based matrix model (4.44) with $r^\star \ll \min(m,n)$. As introduced in Chapter 4, this graph-based model is designed to simulate low-rank matrices in real-world applications in the following sense: the matrix rows (respectively columns) of $M^\star$ present pairwise similarities according to a given graph structure defined on the index set of the matrix rows (respectively columns). For this purpose, a graph Laplacian matrix L^{r} is generated with the prototypical graph model `Community` using the GSPbox [PPS+14]. The function g in this model is chosen as (4.46) with $p = 2$. The data matrix is observed on an index set Ω is that follows the Bernoulli distribution (5.61) without any noise.



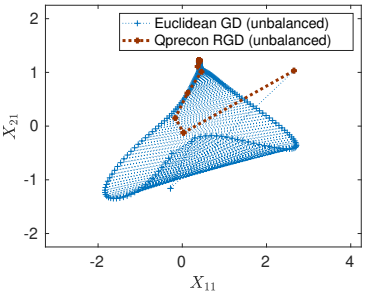
(a) Armijo line search



(b) Line minimization



(c) Balanced initialization



(d) Unbalanced initialization

Figure 5.2: Iteration histories of the algorithms with a balanced and an unbalanced initial point. The size of $M^\star$ is 100×200, with rank $r^\star = 3$. The sampling rate $p = 0.8$. (a)–(b): test RMSEs by time. (c)–(d): path of the matrix entries $([X_t]_{1,1}, [X_t]_{2,1})$ of the iterates $\{\pi(\bar{x}_t)\}$.

We evaluate the matrix recovery performances of the low-rank model (5.55) with the graph-based regularizer (5.59) in comparison with the non-regularized model. In the construction of the graph-based regularizer (5.59), the graph Laplacian matrices are set to be the same as the ones used for generating $M^\star$, as described above. This is of course an ideal choice. Nevertheless, we refer to Chapter 4 for realistic choices, which have also been shown to be effectively helpful for the matrix completion task. The regularization parameters $(\alpha, \gamma_r, \gamma_c)$ are selected from a few randomly generated values in $(0, 1)$ and $(\delta_{\mathrm{r}}, \delta_{\mathrm{c}})$ are estimated based on the norm of $P_\Omega(M^\star)$ and the sampling rate p.

We also compare the time efficiency of Qprecon RGD/RCG with their Euclidean counterparts and several state-of-the-art algorithms in the aforementioned matrix completion tests.

From the experimental results in Figure 5.3, we have the following observations:

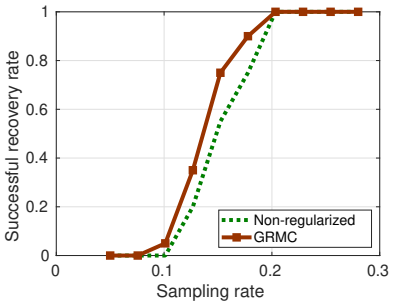
- ◇ In Figure 5.3: (a)-(b), the successful recovery rates with the graph-regularized model is much higher than those with the non-regularized model. This shows that the graph-regularized model already has a smaller sample complexity requirement than the non-regularized model, despite the current parameter search for $(\alpha, \delta_{\mathrm{r}}, \delta_{\mathrm{c}})$ is insufficient.

- ◇ In Figure 5.3: (e)-(f), under a critically small sampling rate, the time efficiency of Qprecon RGD and Qprecon RCG is enhanced when the graph-based regularization scheme (5.59) is used. In particular, with the graph-regularized model, the convergence of these algorithms are seldom slowed down during the iterations, and the convergence rate is linear with a quasi constant rate.

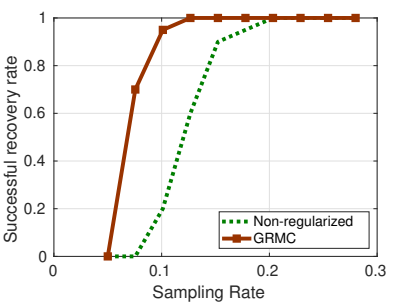
- ◇ In Figure 5.3: (c)-(d), also under a critically small sampling rate, and with both the regularized and non-regularized models, the time efficiency of Qprecon RGD and Qprecon RCG outperform their Euclidean counterparts by at least 8 times or even orders of magnitude.

Comparisons with existing algorithms. Furthermore, we compare the performances of Qprecon RGD, Qprecon RCG with some existing and state-of-the-art algorithms.

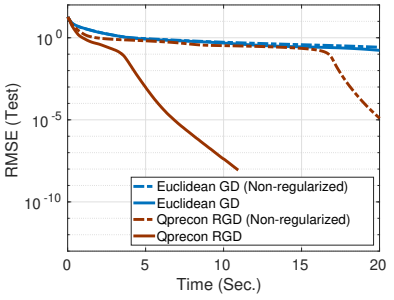
Figure 5.4 shows the iteration histories of Qprecon RGD and the 2-phase GRMC algorithm (Algorithm 4.3.3) in Chapter 4. From this figure, we observe that the iteration histories of these two algorithms are similar or even overlapping, over multiple runs of random tests, and they all show that the matrix recoveries are successful. This suggests that the RGD algorithm (Algorithm 4.3.1) on $\mathcal{M}_{\leq k}$ using the preconditioned metric produces a similar sequence as Qprecon RGD, given an appropriate rank k; see Section 5.3.1. This observation agrees with the discussion about the location of critical points of f (5.21) on $\mathcal{M}_k \subset \mathcal{M}_{\leq k}$.



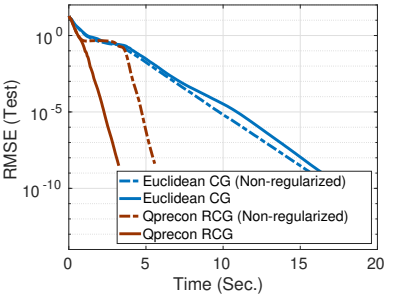
(a) Recovery performances of RGD



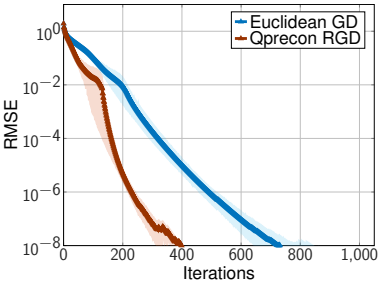
(b) Recovery performances of RCG



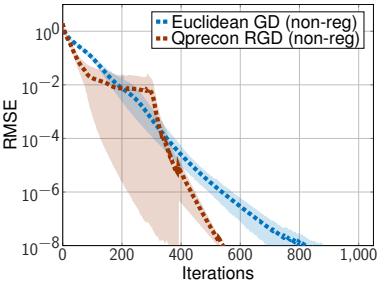
(c) Time efficiency of RGD and GD



(d) Time efficiency of RCG and CG



(e) Graph-regularized model



(f) Non-regularized model

Figure 5.3: Performances of the matrix completion algorithms. The size of $M^\star$ is 500×600, with rank $r^\star = 10$. (a–(b): percentage of successful recoveries (test RMSE below 10^{-8}) with and without the graph-based regularization. (c–(d): test RMSEs by time of Qprecon and Euclidean algorithms under the sampling rate $p = 10\%$ (in a successfully recovered instance). (e–(f): average iteration histories of Qprecon RGD and Euclidean GD under the sampling rate $p = 12.6\%$ (in case of successful recoveries).

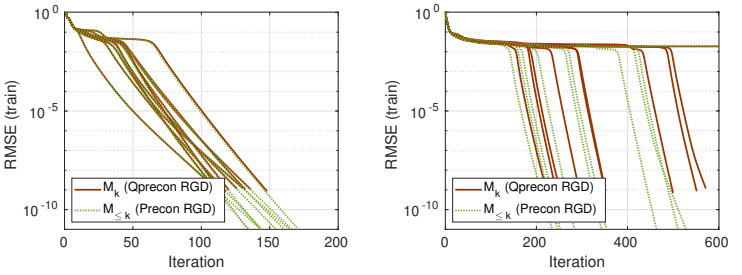
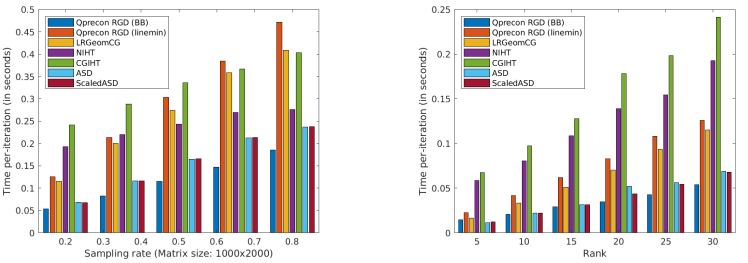


Figure 5.4: Iterative results of the tested algorithms. The y-axis shows the RMSE on the revealed entries (training data). $M^\star$ is generated with non-trivial graph information with the model (4.46) (for $p = 2$) and is partially observed without any noise. Matrix size $m = 500$, $n = 600$, rank $r^\star = 10$. The rank parameter k is set to be equal to $r^\star$ for both algorithms. The sampling rate is set to 0.6 for all tests. The "phase-switching" criterion of `Precon RGD` ($\mathcal{M}_{\leq k}$ `2-phase`)(originally Algorithm 3 of [DAG20]) is if the relative error (on training data) falls below 10^{-2}. (a)–(b): two instances of repeated tests.

For time efficiency comparisons, we test the following algorithms [Van13, WCCL16, TW16] for optimization with fixed-rank matrices. Precisely, LRGeomCG [Van13], NIHT and CGIHT [WCCL16] use the same manifold structure of $\mathcal{M}_k$ and the same retraction operator for their respective update rules on $\mathcal{M}_k$. These algorithms produce iterates the matrix product space $\mathrm{St}(m,k) \times \mathrm{St}(n,k) \times \mathbb{R}^{k\times k}$, on which the Euclidean metric is used for the definition of the gradient. A retraction is needed to ensure the fixed-rank constraint, that is, $\pi(\bar{x}) := U\Sigma V^{\mathrm{T}} \in \mathcal{M}_k$. The projection-like retraction [AM12] (also called the IHT for "iterative hard thresholding") is used. ASD and ScaledASD [TW16] are algorithms using alternating steepest descent on the two-factor product space $\overline{\mathcal{M}_k}$.

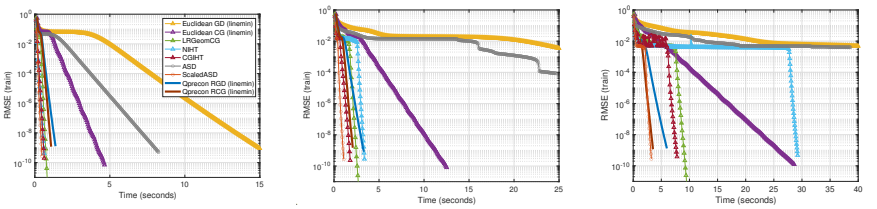
Note that the computation environment of the above algorithms are similar: (i) All these algorithms, as well as the proposed ones, are implemented in MATLAB; (ii) under the assumption that $\max(m,n) \ll |\Omega| = \rho mn$, the most costly part of each of these algorithms is the computation of the computing the residual matrix $S := P_\Omega(X - M^\star) \in \mathbb{R}^{m\times n}$, where X depends on the factorization form underlying the structure of the manifold (or simply Euclidean domain for ASD) of each algorithm, whose costs are at the order $O(|\Omega|k)$ for a rank parameter k given. For this computational step, all the algorithms use a MEX interface to a function implemented in C.

In Figure 5.5, we show the time efficiency of the tested algorithms for recovering the same type of low-rank matrix as the above tests. From these results, we can observe that the proposed algorithms Qprecon RGD and Qprecon RCG perform similarly as ScaledASD and they are comparable or faster than the rest of the algorithms. In Figure 5.5, the average costs of time of the algorithms are given based on tests on a 1000×2000 matrix, under various rank choices $k \in \{5, 10, \ldots, 30\}$ and sampling rates $p \in \{0.2, 0.3, \ldots, 0.8\}$. We observe that



(a) (b)

Figure 5.5: Average costs of time per-iteration. The matrix size is 1000×2000.



(a) $r^\star = 10$ (b) $r^\star = 20$ (c) $r^\star = 30$

Figure 5.6: Iterative results of the tested algorithms. The y-axis shows the RMSE on the revealed entries (training data). $M^\star$ is generated with non-trivial graph information with the model (4.46) (for $p = 2$) and is partially observed without any noise. The matrix size is 800×900, with ranks $r^\star \in \{20, 30\}$. The sampling rate $p = 0.6$ for all tests. The rank parameter $k = r^\star$ in each setting.

Qprecon RGD (RBB), ASD and Scaled ASD are three most efficient algorithms in terms of per-iteration cost of time, for all settings of (p, k). In particular, these three algorithms have much better scalability in k than the rest of the algorithms.

5.6 Conclusion

We investigated methods for the matrix completion problem with a fixed-rank constraint and focused on a Riemannian gradient descent algorithm in the framework of optimization on the quotient manifold of fixed-rank matrices. We showed that the Riemannian gradient descent algorithm under the aforementioned quotient geometric setting not only enjoys the advantage of matrix factorization but can also be analyzed in a more convenient way than existing matrix factorization methods. We developed novel results for analyzing the quotient manifold-based algorithm and proved that this algorithm solves the fixed-rank matrix completion problem with a linear convergence rate. Moreover, the convergence property of the algorithm has desirable invariance properties in contrast to Euclidean gradient descent algorithms. Because of the efficient iteration efficiency and its light per-iteration cost, the time efficiency of this algorithm is also shown to be much faster than the Euclidean gradient descent algorithms and is faster than many other Riemannian algorithms on the set of fixed-rank matrices. Through the convergence analysis, we also provided a novel understanding of the graph-based regularization in the theoretical framework of matrix completion.

Chapter 6

Tensor completion using graph-based regularization

Tensor completion refers to the task of recovering missing values of a multidimensional array and can be seen as a generalization of the matrix completion problem. Similar to the approximation of a matrix with low-rank models, the approximation of a tensor can be formulated by a *low-rank* tensor model. Starting from this idea, low-rank tensor completion consists in finding a low-rank approximation of a tensor based on a given subset of its entries. Applications of low-rank tensor completion can be found in many areas, e.g., signal processing for EEG (brain signals) data [MHH+06] and MRI (magnetic resonance imaging) [BAH16], and image and video inpainting [BSCB00, LMWY12, KR07].

In the literature on low-rank matrix completion, the matrix nuclear norm is proven [CR09b, RFP10] to be a convex relaxation of the matrix rank that guarantees exact solutions to the corresponding rank-constrained problem in some specific circumstances and has been used in various matrix rank-constrained problems [MHE+10] such as matrix completion. Generalized from this matrix relaxation, several works [LMWY09, GRY11, LMWY12, SHZL13, YHS13, SDDS14] extended the nuclear norm-based regularization to the completion of partially observed tensors (also known as multidimensional arrays). In [LMWY12], the nuclear norm of a tensor is defined as a convex combination of nuclear norms of its unfolding matrices, and the tensor completion model is as follows,

$$\min_{\mathcal{Z}\in\mathbb{R}^{m_1\times\cdots\times m_k}} \quad \frac{1}{2}\|\mathcal{P}_\Omega(\mathcal{T}-\mathcal{Z})\|_F^2 + \sum_{i=1}^{k}\lambda_i\|\mathcal{Z}_{(i)}\|_*, \tag{6.1}$$

where $\mathcal{T}\in\mathbb{R}^{m_1\times\ldots\times m_k}$ is a given tensor that is only partially known, and $\mathcal{P}_\Omega$ is the projection operator that retains the known entries of $\mathcal{T}$ recorded in the index set $\Omega\subset[\![m_1]\!]\times\cdots\times[\![m_k]\!]$, and $\|\mathcal{Z}_{(i)}\|_*$ denotes the matrix *nuclear norm* of the mode-i matricization (Definition 2.4.5) of tensor $\mathcal{Z}$. The penalty

terms $\|\mathcal{Z}_{(i)}\|_*$ in (6.1) promote solutions such that the matricizations of the tensor variable $\mathcal{Z}$ have a low rank.

However, the problem (6.1) requires handling the full tensor variable (in its matricization forms), hence has very high memory requirements when the tensor dimensions are large. In contrast, low-rank tensor decomposition offers an alternative for its significant savings in memory. Previous work of low-rank decomposition of tensors with missing data include [AB98, CG09, KABO10, PFS17, TB05, SD19].

Depending on different tensor decomposition forms, there are several ways to define the rank of a tensor. Low-rank tensor decompositions provide a useful tool for tensor representation and are widely used in tensor completion [AB98, CG09, KABO10, ADKM11, PFS16, TB05, KB09]. The low-rank tensor decomposition paradigm allows for extracting the most meaningful and informative latent structures of a tensor, which usually contain heterogeneous and multi-aspect data. The Canonical Polyadic (CP) decomposition [Hit27b, Kru77, KB09], the Tucker and multilinear decomposition (MLSVD) [Tuc66, DLDMV00a, DLDMV00b], and the tensor-train (TT) decomposition [Ose11, GKK15, PTBD16] are among the most fundamental tensor decomposition forms. Other variants include hierarchical tensor representations [DSH13, RSS15, RSS17] and PARAFAC2 models [RSS15]. We refer to [SDF+17] for a thorough view on these tensor decompositions in the context of signal processing and machine learning.

The rest of this chapter is based on [GDAG20]. The notation of tensor and matrix operations and definitions of the tensor decompositions used in the following part are given in Section 2.4.

We focus on the CP decomposition model for tensor completion [ADKM11, Bro97, Bro98, KTBB99, Kro83, TB05, SD19]. Besides the CP decomposition model, graph Laplacian-based regularization has also been applied to tensor completion, see [NHTK12, GCZS16, GZA+18]. Similar approaches [BMG13, LNSS16, ZZC15a, ZZC15b] also exploit auxiliary information of inter-relations between the tensor entries with a probabilistic perspective.

In this work, we consider the following model using CP decomposition and a graph Laplacian-based regularization:

$$\min_{U^{(1)},\ldots,U^{(k)}} \frac{1}{2}\|\mathcal{P}_\Omega(\mathcal{T} - [\![U^{(1)},\cdots,U^{(k)}]\!])\|_F^2 + \sum_{i=1}^{k} \frac{\lambda_i}{2}\langle U^{(i)}(U^{(i)})^{\mathrm{T}}, L^{(i)}\rangle + \sum_{i=1}^{k} \frac{\lambda_i}{2}\|(U^{(j)})^{\odot_{j\neq i}}\|_F^2, \tag{6.2}$$

where $\Omega \subset [\![m_1]\!] \times \cdots \times [\![m_k]\!]$ is the index set of the revealed entries and $\mathcal{T}$ is the ground truth tensor that is known only on Ω. The proportion of the known entries $|\Omega|/(m_1 \ldots m_k)$, or its expectation $\mathbb{E}[|\Omega|]/(m_1 \ldots m_k)$, is referred to as the

sampling rate ρ. The shifted graph Laplacian $L^{(i)}$ is defined as

$$L^{(i)} = \lambda_L \mathbf{Lap}^{(i)} + I_{m_i}, \tag{6.3}$$

where $\mathbf{Lap}^{(i)}$ is the graph Laplacian matrix which is assumed to be known. The parameters $\lambda_i \geq 0$ and $\lambda_L \geq 0$ control the trade-off between the training error function and the regularization term. In particular, when $\lambda_L = 0$, problem (6.2) reduces to a graph-agnostic tensor completion model. The search space of (6.2) is $\mathbb{R}^{m_1 \times R} \times \cdots \times \mathbb{R}^{m_k \times R}$. Throughout this chapter, the graphs and the graph Laplacian matrices involved in the regularization term are described in Section 2.3. The model (6.2) can be seen as an extension of (4.9) in Chapter 4.

We use the CP decomposition in the problem setting above because it is the most natural extension of the two-term matrix factorization in the graph-regularized matrix completion model. Other decomposition forms such as Tucker or MLSVD appear less convenient because the core tensor dimensions are different than those of the graph matrices (which equals the tensor dimensions).

Besides considerations for a more data-adaptive model, the graph-based regularization term plays a role like other regularizers in limiting the actual search space of the problem to a closed sublevel set, for many algorithms. It is worth noting that, in the absence of such regularizers, the low-rank tensor approximation problem underlying (6.2) can be ill-posed, in the sense that the best approximation with bounded CP rank (depending on R) does not exist. This ill-posedness can be seen in the well-known example with *border tensors* [DSL08], where the approximation of such tensors can be arbitrarily well but it does not have a minimum (despite the existence of an infimum).

Contributions and organization

The main contributions of this chapter are as follows.

- ◇ We provide an alternating minimization algorithm for solving the graph-regularized tensor completion problem. An efficient Hessian-vector multiplication scheme is used in the linear conjugate gradient (CG) algorithm for solving the subproblems in this alternating minimization framework. The computational framework of the linear CG algorithm is inspired by Rao et al. [RYRD15] for graph-regularized matrix completion. In addition, an alternating direction method of multipliers (ADMM) algorithm is also proposed; see Section 6.1.

- ◇ We provide a proof for the convergence of iterates of the proposed AltMin algorithm to a critical point of the objective function according to the Kurdyka-Łojasiewicz (KŁ) property; see Section 6.2.

- ◇ We show through experiments (Section 6.3) on both synthetic and real data that our algorithms produce tensor completion results with good recovery accuracies and are time efficient compared to several baseline methods.

The rest of this chapter is organized as follows. In Section 6.1, an alternating minimization (AltMin) algorithm using linear CG for solving the subproblems is proposed; an ADMM algorithm is also developed for solving the graph-regularized LRTC problem. Convergence analysis of the AltMin algorithm is given in Section 6.2. Numerical experiments together with some interesting observations are presented in Section 6.3. Conclusion is given in Section 6.4.

6.1 Algorithms using alternating minimization

In this section, we introduce an alternating minimization (AltMin) algorithm and an ADMM algorithm for solving the LRTC problem (6.2).

To minimize the objective function of (6.2), defined on the product space of tensor factors $\mathbb{R}^{m_1 \times R} \times \cdots \times \mathbb{R}^{m_k \times R}$, alternating minimization (also referred to as block coordinate descent) consists in minimizing the function cyclically over each factor matrix among $(U^{(1)}, \ldots, U^{(k)})$ while keeping the remaining variables fixed at their last updated values.

Let $f(\cdot)$ denote the objective function of (6.2) and $U_t^{(i)}$ denote the t-th iterate of $U^{(i)}$ for $t \geq 0$. Let $f_{t+1}^{(i)}$ denote the objective function of the subproblem in $U^{(i)}$ as follows:

$$f_{t+1}^{(i)}(U^{(i)}) \triangleq f(U_{t+1}^{(1)}, \ldots, U_{t+1}^{(i-1)}, U^{(i)}, U_t^{(i+1)}, \ldots, U_t^{(k)}). \tag{6.4}$$

Algorithm 6.1.1 Alternating minimization

Input: Data (known on Ω) $\mathcal{P}_\Omega(\mathcal{T}) \in \mathbb{R}^{m_1 \times \ldots \times m_k}$, observed set Ω. Objective function f

Output: $(U_t^{(i)})_{i=1,\ldots,k}$

Initialization: $U_0^{(1)}, \ldots, U_0^{(k)}$
for $t = 0, 1, 2, \ldots$ **do**
  if stopping criterion is satisfied **then**
    return;
  end if
  for $i = 1, \ldots, k$ **do**
    $U_{t+1}^{(i)} = \arg\min_{U \in \mathbb{R}^{m_i \times R}} f_{t+1}^{(i)}(U)$
  end for
end for

During the $(t+1)$-th iteration and for $i \in [\![k]\!]$, subproblem (6.4) has the

following expression[1]:

$$\min_{U^{(i)}\in\mathbb{R}^{m_i\times R}} \frac{1}{2}\|\mathcal{P}_{\Omega^{(i)}}(\mathcal{T}_{(i)} - U^{(i)}[(U^{(j)})^{\odot j\neq i}]^{\mathrm{T}})\|_F^2 + \frac{\lambda_i}{2}\langle U^{(i)}(U^{(i)})^{\mathrm{T}}, L^{(i)}\rangle + \sum_{\substack{j=1\\ j\neq i}}^{k} \frac{\lambda_j}{2}\|(U^{(n)})^{\odot n\neq j}\|_F^2, \tag{6.5}$$

where $\Omega^{(i)}$ is the set of 2-dimensional indices, in the form of $(\ell_i, r_i) \in \Omega^{(i)}$, corresponding to $(\ell_1, \ldots, \ell_k) \in \Omega$ with respect to the tensor matricization (Definition 2.4.5). The one-to-one map satisfies (2.24).

Due to the graph Laplacian-based regularization term, the major challenge in solving (6.2) by the alternating minimization procedure is the structure of each subproblem, which is different from those of an unregularized tensor decomposition problem. We explain this in detail as follows. For clarity and practical reasons, we show this by first converting the matrix variable $U^{(i)}$ into its vectorization $\mathbf{x} := \mathrm{vec}\big((U^{(i)})^{\mathrm{T}}\big) \in \mathbb{R}^{m_i R}$ and then studying $g^{(i)}(\mathbf{x}) := f_{t+1}^{(i)}(U^{(i)})$. The function g is a quadratic function of the following form,

$$g^{(i)}(\mathbf{x}) := \frac{1}{2}\mathbf{x}^{\mathrm{T}} M^{(i)}\mathbf{x} - \mathrm{vec}(Q^{(i)})^{\mathrm{T}}\mathbf{x}, \quad \mathbf{x}\in\mathbb{R}^{m_i R}, \tag{6.6}$$

where

$$M^{(i)} = A^{(i)} + \lambda_i L^{(i)} \otimes I_R + I_{m_i} \otimes C^{(i)} \in \mathbb{R}^{m_i R\times m_i R}, \tag{6.7a}$$

$$Q^{(i)} = (\mathcal{P}_{\Omega^{(i)}}\mathcal{T}_{(i)})(U^{(j)})^{\odot j\neq i} \in \mathbb{R}^{m_i\times R}. \tag{6.7b}$$

The components $A^{(i)}$ and $C^{(i)}$ in (6.7a) are defined and computed as follows. Let $A^{(i)} \in \mathbb{R}^{m_i R\times m_i R}$ be the matrix of the following quadratic form $\mathbf{x}^T A^{(i)}\mathbf{x} := \|\mathcal{P}_{\Omega}(U^{(i)}[(U^{(j)})^{\odot j\neq i}]^{\mathrm{T}})\|_F^2$ in $\mathbf{x} = \mathrm{vec}\big((U^{(i)})^{\mathrm{T}}\big)$. We have

$$\begin{aligned}
&\|\mathcal{P}_{\Omega^{(i)}}(U^{(i)}[(U^{(j)})^{\odot j\neq i}]^{\mathrm{T}})\|_F^2 = \langle U^{(i)}(U^{(-i)})^{\mathrm{T}}, P_{\Omega^{(i)}}(U^{(i)}(U^{(-i)})^{\mathrm{T}})\rangle \\
&= \mathrm{tr}(U^{(-i)}(U^{(i)})^{\mathrm{T}} P_{\Omega^{(i)}}(U^{(i)}(U^{(-i)})^{\mathrm{T}})) = \sum_{s=1}^{m_i} \mathrm{tr}(U_{s,:}{}^{\mathrm{T}} P_{\Omega_s^{(i)}}(U_{s,:}(U^{(-i)})^{\mathrm{T}})U^{(-i)}) \\
&= \sum_{s=1}^{m_i} U_{s,:}\left(\sum_{\ell\in\Omega_s^{(i)}} (U_{\ell,:}^{(-i)})^{\mathrm{T}} U_{\ell,:}^{(-i)}\right)(U_{s,:})^{\mathrm{T}},
\end{aligned} \tag{6.8}$$

where $U_{s,:}$ denotes the s-th row of $U^{(i)}$, $U^{(-i)}$ denotes $(U^{(j)})^{\odot j\neq i} \in \mathbb{R}^{m_{(-i)}\times R}$ and $m_{(-i)}$ denotes the number $\prod_{j\neq i} m_j$ for brevity. In the line above (6.8), we

[1]For convenience, we ignore the subscript $t+1$ or t in the variables $U^{(j)}$ for all $j = 1, \ldots, k$, and omit constant terms in the objective.

decomposed $P_{\Omega^{(i)}}$ by rows with $P_{\Omega_s^{(i)}}$ (defined on $\mathbb{R}^{m_{(-i)}}$), for $s = 1, \dots, m_i$, which applies to the s-th row $U_{s,:}(U^{(-i)})^T$. Therefore, $A^{(i)} \in \mathbb{R}^{m_i R \times m_i R}$ is a block diagonal matrix with m_i diagonal blocks and each block has the form

$$A_s^{(i)} = \sum_{\ell \in \Omega_s^{(i)}} (U_{\ell,:}^{(-i)})^{\mathrm{T}} U_{\ell,:}^{(-i)} \in \mathbb{R}^{R \times R}, \tag{6.9}$$

where $\Omega_s^{(i)} = \{\ell : (s, \ell) \in \Omega^{(i)}\}$.

The component $I_{m_i} \otimes C^{(i)}$ denotes the matrix related to the quadratic form

$$q(U^{(i)}) := \sum_{j \neq i} \lambda_j \|(U^{(n)})^{\odot_{n \neq j}}\|_F^2 \tag{6.10}$$

in (6.5). Now we verify that

$$C^{(i)} = \sum_{\substack{j=1 \\ j \neq i}}^{k} \lambda_j \mathrm{diag}[(\|U_{:,\ell}^{(-i,-j)}\|^2)_{\ell=1,\dots,R}] \in \mathbb{R}^{R \times R}, \tag{6.11}$$

where $U^{(-i,-j)} := (U^{(n)})^{\odot_{n \neq i,j}}$ denotes the Khatri-Rao product of $U^{(n)}$'s excluding $U^{(i)}$ and $U^{(j)}$. Indeed, the function (6.10) writes

$$\begin{aligned} q(U^{(i)}) &= \sum_{j \neq i} \lambda_j \sum_{\ell=1}^{R} \|U_{:,\ell}^{(-i,-j)} \otimes U_{:,\ell}^{(i)}\|_2^2 \\ &= \sum_{j \neq i} \lambda_j \sum_{\ell=1}^{R} \underbrace{\|U_{:,\ell}^{(-i,-j)}\|_2^2}_{C_{\ell\ell}^{(i,j)}} \mathrm{tr}\left(U_{:,\ell}^{(i)} (U_{:,\ell}^{(i)})^{\mathrm{T}}\right) \\ &= \sum_{j \neq i} \lambda_j \,\mathrm{tr}\left(U^{(i)} C^{(i,j)} (U^{(i)})^{\mathrm{T}}\right) = \mathrm{tr}\left(U^{(i)} (\sum_{j \neq i} \lambda_j C^{(i,j)}) (U^{(i)})^{\mathrm{T}}\right). \end{aligned}$$

By recalling that $\mathrm{tr}(X^{\mathrm{T}} C X) = \mathrm{vec}(X)^{\mathrm{T}} (I \otimes C) \mathrm{vec}(X)$, the formula (6.11) of $C^{(i)}$ yields the identification $q(U^{(i)}) = \mathbf{x}^{\mathrm{T}} (I_{m_i} \otimes C^{(i)}) \mathbf{x}$, with $\mathbf{x} = \mathrm{vec}\big((U^{(i)})^{\mathrm{T}}\big)$.

The function $g^{(i)}$ in (6.6), and equivalently $f_{t+1}^{(i)}$ of (6.5), is strongly convex (see Theorem 6.2.7 point 2) provided that $\lambda_n > 0$ for $n = 1, \dots, k$. As a consequence, the update step (6.5) consists of finding the graph-regularized least squares solution

$$M^{(i)} \mathbf{x}^* = \mathrm{vec}(Q^{(i)}). \tag{6.12}$$

Note that the main computational challenge in finding the least-squares solution (6.12) is the presence of graph Laplacian-based regularization terms in (6.6). The similar difficulty can be found in the graph-regularized least squares problem in [RYRD15]. More precisely, the matrix $M^{(i)} \in \mathbb{R}^{m_i R \times m_i R}$ in (6.6) is not block diagonal because of the component $L^{(i)} \otimes I_R$. Therefore,

the least squares problem with (6.6) cannot be decomposed into m_i separable least squares problems in $\mathbb{R}^R$. In the next subsection, we use linear CG to solve each subproblem with respect to its vectorized form. We also consider (in Section 6.1.2) an alternating direction method of multipliers (ADMM) as an alternative way to address the difficulty with these nonseparable least squares problems.

The stopping criterion in Algorithm 6.1.1 (line 3) is satisfied if either of the following conditions is satisfied: (i) the wall time used for producing the latest iterate is larger than a time budget parameter $\tau_{\max}$ (which is potentially set to ∞); (ii) the progress of the iterate $(U_t^{(i)})_{i=1,\dots,k}$, measured by a heuristic difference function Δ_t, is smaller than a (global) tolerance parameter ϵ. Here we define Δ_t as follows,

$$\Delta_t := |E(U_t; \Omega_{\text{tr}}) - E(U_{t-1}; \Omega_{\text{tr}})|, \tag{6.13}$$

where $E(U; \Omega_{\text{tr}}) := \frac{\|P_{\Omega_{\text{tr}}}([\![U^{(1)},\dots,U^{(k)}]\!]-\mathcal{T})\|_F}{\|P_{\Omega_{\text{tr}}}(\mathcal{T})\|_F}$ is the relative error restricted to the index set $\Omega_{\text{tr}} \subset [\![m_1]\!] \times \cdots \times [\![m_k]\!]$ of the revealed entries.[2]

In the following subsections, we consider linear CG for solving the high-dimensional least-squares problem (6.6).

6.1.1 The linear CG solver

Algorithm 6.1.2 shows an instance of AltMin using linear CG as the subproblem solver. Detailed steps for the linear CG algorithm are given in Algorithm 6.1.3.

Algorithm 6.1.2 AltMin-CG for solving (6.2)

Input: Observed tensor $\mathcal{P}_\Omega(\mathcal{T})$, graph Laplacian $\mathbf{Lap}^{(1)}, \dots, \mathbf{Lap}^{(k)}$, observed set Ω, parameters $\lambda_1, \dots, \lambda_k$ and λ_L

Output: $(U_t^{(i)})_{i=1,\dots,k}$

1: Initialization: $U_0^{(1)}, \dots, U_0^{(k)}$
2: **for** $t = 0, 1, 2, \dots$ **do**
3: **if** stopping criterion is satisfied **then**
4: return;
5: **end if**
6: **for** $i = 1, \dots, k$ **do**
7: Compute: $C^{(i)}, Q^{(i)}$ defined in (6.11), (6.7b) and $(U^{(j)})^{\odot_{j\neq i}}$
8: $\mathbf{x}_{t+1}^{(i)} := \arg\min_{\mathbf{x}} g^{(i)}(\mathbf{x})$ by Algorithm 6.1.3
9: $U_{t+1}^{(i)} = \text{unvec}(\mathbf{x}_{t+1}^{(i)})$
10: **end for**
11: **end for**

[2]The subscript "tr", indicating the "training set", refers to the revealed entries.

Note that $M^{(i)}$ has a special form which contains Kronecker products and its size is very large, hence we compute it in a more efficient way by a special Hessian-vector multiplication in the CG method. By the relation $(B^{\mathrm{T}} \otimes A)\mathrm{vec}(X) = \mathrm{vec}(AXB)$, it follows that

$$\begin{aligned}(L^{(i)} \otimes I_R)\mathbf{x} &= \mathrm{vec}\big((U^{(i)})^{\mathrm{T}} L^{(i)}\big),\\ (I_{m_i} \otimes C^{(i)})\mathbf{x} &= \mathrm{vec}\big(C^{(i)}(U^{(i)})^{\mathrm{T}}\big),\end{aligned}$$

where $\mathbf{x} = \mathrm{vec}\big((U^{(i)})^{\mathrm{T}}\big)$. Thus the Hessian-vector multiplication can be implemented by a series of matrix multiplications as follows

$$M^{(i)}\mathbf{x} = \mathrm{vec}\big(\lambda_i (U^{(i)})^{\mathrm{T}} L^{(i)} + C^{(i)}(U^{(i)})^{\mathrm{T}}\big) + A^{(i)}\mathbf{x}. \tag{6.14}$$

Define $\mathrm{vec}(N^{(i)}) = A^{(i)}\mathbf{x}$ with $N^{(i)}_{:,j} = A^{(i)}_j (U^{(i)}_{j,:})^{\mathrm{T}}$. In view of (6.9), and similar to [RYRD15], $A^{(i)}_j (U^{(i)}_{j,:})^{\mathrm{T}}$ can be computed in the following way

$$N^{(i)}_{:,j} = A^{(i)}_j (U^{(i)}_{j,:})^{\mathrm{T}} = \sum_{\ell \in \Omega^{(i)}_s} U^{(-i)}_{\ell,:} (U^{(i)}_{j,:})^{\mathrm{T}} (U^{(-i)}_{\ell,:})^{\mathrm{T}}. \tag{6.15}$$

Details to compute this Hessian-vector product in the CG method are listed in Algorithm 6.1.4.

Computational cost of AltMin-CG The computational cost for each alternating step (6.5) corresponds to the procedure required by line 7–line 9 of Algorithm 6.1.2.

The cost of forming $(U^{(j)})^{\odot_{j\neq i}}$ is $O(\frac{|\Omega|R}{\rho m_i})$, where ρ denotes the sampling rate. The cost of computing $Q^{(i)}$ in (6.7b) is $O(|\Omega|R)$ with access to $(U^{(j)})^{\odot_{j\neq i}}$. The cost of forming $C^{(i)}$ in (6.11) is $O(\frac{|\Omega|R}{\rho m_i m_j})$.

The major cost in Algorithm 6.1.2 corresponds to line 8, which involves (inner) iterations of the linear CG. The total cost of line 8 is n_{CG} times the per-iteration cost of the linear CG algorithm (Algorithm 6.1.3), where n_{CG} denotes the number of iterations required by the CG solver (Algorithm 6.1.3) for producing $\mathbf{x}^{(i)}_{t+1}$. The per-iteration cost of Algorithm 6.1.3 is mainly composed of the following components.

- ◇ Cost of computing $A^{(i)}\mathbf{x}$: $O(|\Omega|R)$, since the cost of computing $A^{(i)}_j (U^{(i)}_{j,:})^{\mathrm{T}}$ in (6.15) is $O(|\Omega^{(i)}_j|R)$ for $j = 1, ..., m_i$ and $\sum_{j=1,...,m_i} |\Omega^{(i)}_j| = |\Omega|$;
- ◇ Cost of computing $(L^{(i)} \otimes I_R)\mathbf{x}$: $O(\mathrm{nnz}(L^{(i)})R)$;
- ◇ Cost of computing $(I_{m_i} \otimes C^{(i)})\mathbf{x}$: $O(m_i R)$.

Algorithm 6.1.3 Linear CG for solving $\min g^{(i)}(\mathbf{x})$ defined in (6.6)

Input: $M^{(i)} \in \mathbb{R}^{m_i R \times m_i R}$, $Q^{(i)} \in \mathbb{R}^{m_i \times R}$, initial point $\mathbf{x}_0 \in \mathbb{R}^{m_i R}$, accuracy parameter ϵ, iteration budget $T_{\max}$

Output: $x^* \in \mathbb{R}^{m_i R}$

$\mathbf{r}_0 = \mathrm{vec}(Q^{(i)}) - M^{(i)}\mathbf{x}_0$
for $t = 0, \ldots, T_{\max}$ **do**
  Compute: $\|\mathbf{r}_t\|$
  if $\|\mathbf{r}_t\| \leq \epsilon \|\mathbf{r}_0\|$ **then**
    Break
  end if
  if $t = 0$ **then**
    $\mathbf{p}_1 = \mathbf{r}_0$
  else
    $\mathbf{p}_{t+1} = \mathbf{r}_t + \frac{\|\mathbf{r}_t\|^2}{\|\mathbf{r}_{t-1}\|^2}\mathbf{p}_t$
  end if
  Compute: $\mathbf{v}_{t+1} = M^{(i)}\mathbf{p}_{t+1}$ # see Algorithm 6.1.4
  Compute: $\alpha = \frac{\|\mathbf{r}_t\|^2}{\mathbf{p}_{t+1}{}^{\mathrm{T}}\mathbf{v}_{t+1}}$
  Compute: $\mathbf{x}_{t+1} = \mathbf{x}_t + \alpha\mathbf{p}_{t+1}$, $\mathbf{r}_{t+1} = \mathbf{r}_t - \alpha\mathbf{v}_{t+1}$
end for
return $\mathbf{x}^* = \mathbf{x}_t$.

Algorithm 6.1.4 Hessian-vector multiplication $M^{(i)}x$ in the CG method

Input: $L^{(i)} \in \mathbb{R}^{m_i \times m_i}$, $\Omega_j^{(i)}$, $C^{(i)} \in \mathbb{R}^{R \times R}$, $D^{(i)} := (U^{(j)})^{\odot_{j \neq i}} \in \mathbb{R}^{m_{(-i)} \times R}$, $\mathbf{x} := \mathrm{vec}\big((U^{(i)})^{\mathrm{T}}\big) \in \mathbb{R}^{m_i R}$, $\lambda_i \geq 0$.

Output: $M^{(i)}\mathbf{x}$

for $i = 1, \ldots, k$ **do**
  $X = \mathrm{unvec}(\mathbf{x}) \in \mathbb{R}^{R \times m_i}$
  Compute: $N_{:,j}^{(i)} = \sum_{\ell \in \Omega_j^{(i)}} (D_{\ell,:} x_{:,j}) D_{\ell,:}{}^{\mathrm{T}}$
  Compute: $M^{(i)}\mathbf{x} = \mathrm{vec}(C^{(i)}X + \lambda_i X L^{(i)}) + \mathrm{vec}(N^{(i)})$ defined in (6.14)
end for

Overall, the cost of computing the Hessian-vector multiplication $M^{(i)}\mathbf{x}$ is

$$O(\text{nnz}(L^{(i)})R + |\Omega|R).$$

Therefore, the dominant term in the per-iteration cost of Algorithm 6.1.2 is

$$n_{\text{CG}}O(\text{nnz}(L^{(i)})R + |\Omega|R). \tag{6.16}$$

Discussion Alternating minimization (AltMin) methods have been applied to various low-rank tensor completion problems [LYZY10, LMWY12, XY13b] and are known for many advantages. AltMin is a desirable choice for the graph-regularized tensor decomposition problem (6.2) for the following reasons:

(i) It is easy to implement as there is no need to tune optimization parameters like step sizes; (ii) Each of the subproblems (6.5) is convex and easy to solve. In fact, we have shown in the previous subsection that each subproblem is equivalent to a least-squares problem in the $m_i R$-dimensional vector space; and (iii) global convergence properties of alternating minimization methods in matrix and tensor decomposition-related problems are also well known.

6.1.2 ADMM

Besides the AltMin-CG method, we also consider ADMM to solve the LRTC problem (6.2). ADMM has become a very popular approach to solving a broad variety of optimization problems in signal, image processing and sparse representation problems [ABDF10, BPC+11, GO09] for several reasons, and one notable advantage of ADMM is that it provides a simple way to handle optimization problems with complicated objective functions for its nature of decomposing a problem into sequences of simpler subproblems. Moreover, its decoupling of problem variables usually enables simple implementation and easy parallelization. The ADMM framework is used in [LS15], for example, for designing parallel algorithms for nonnegative tensor factorization and with encouraging results, is shown to have high potential for large-scale applications.

For the graph-regularized tensor problem (6.2), the aforementioned advantages are also appealing. As is shown in the description of AltMin (Algorithm 6.1.1), the graph-based regularization term in this problem model induces a major computational challenge (especially for large-scale applications), since it turns the least-squares subproblems harder to solve (in centralized as well as parallel settings) due to the fact that it breaks the (pure) block-diagonal pattern of the Hessian (see (6.7a)) of the least-squares subproblems.

To remedy this, we address the problem (6.2) in the following spirit: we introduce an auxiliary variable $(B^{(1)}, \ldots, B^{(k)}) \in \mathbb{R}^{m_1 \times R} \times \cdots \times \mathbb{R}^{m_k \times R}$ to replace the role of $(U^{(i)}, \ldots, U^{(k)}$ in the graph-based regularization term and then impose equality constraints $B^{(i)} = U^{(i)}$ for all i. Subsequently, we apply ADMM to the reformulation.

The reformulation with auxiliary variables is:

$$\begin{aligned}\min_{U^{(1)},\ldots,U^{(k)}} & \frac{1}{2}\|\mathcal{P}_\Omega(\mathcal{T}-[\![U^{(1)},\cdots,U^{(k)}]\!])\|_F^2+\sum_{i=1}^{k}\frac{\lambda_i}{2}\langle B^{(i)}(B^{(i)})^{\mathrm{T}},L^{(i)}\rangle \\ & +\sum_{i=1}^{k}\frac{\lambda_i}{2}\|(U^{(j)})^{\odot_{j\neq i}}\|_F^2, \\ & \text{subject to} \quad U^{(i)}=B^{(i)}, i=1,\ldots,k. \end{aligned} \tag{6.17}$$

The augmented Lagrangian for (6.17) is

$$\mathcal{L}_\eta(U,B,Y)=f(U,B)+\sum_{i=1}^{k}\langle Y^{(i)},B^{(i)}-U^{(i)}\rangle+\sum_{i=1}^{k}\frac{\eta}{2}\|B^{(i)}-U^{(i)}\|_F^2, \tag{6.18}$$

where $f(U,B)$ denotes the objective function of (6.17) and $Y^{(i)}\in\mathbb{R}^{m_i\times R}$ is the matrix of Lagrange multiplier and $\eta>0$ is a penalty parameter. We apply the ADMM iterative scheme to minimize $\mathcal{L}_\eta$ over $\{U^{(1)},\ldots,U^{(k)}\}$ and $\{B^{(1)},\ldots,B^{(k)}\}$ as follows

$$\{U_{t+1}^{(i)}\}_{i=1}^{k}=\underset{\{U^{(i)}\}_{i=1}^{k}}{\arg\min}\,\mathcal{L}_{\eta_t}(U^{(1)},\ldots,U^{(k)},B_t^{(1)},\ldots,B_t^{(k)},Y_t^{(1)},\ldots,Y_t^{(k)}) \tag{6.19}$$

$$\{B_{t+1}^{(i)}\}_{i=1}^{k}=\underset{\{B^{(i)}\}_{i=1}^{k}}{\arg\min}\,\mathcal{L}_{\eta_t}(U_{t+1}^{(1)},\ldots,U_{t+1}^{(k)},B^{(1)},\ldots,B^{(k)},Y_t^{(1)},\ldots,Y_t^{(k)}) \tag{6.20}$$

$$Y_{t+1}^{(i)}=Y_t^{(i)}+\eta_t(B_{t+1}^{(i)}-U_{t+1}^{(i)}),\quad i=1,\ldots,k. \tag{6.21}$$

Updating $\{U_{t+1}^{(1)},\ldots,U_{t+1}^{(k)}\}$. The optimization problem (6.19) can be rewritten as follows when updating $\{U_{t+1}^{(1)},\ldots,U_{t+1}^{(k)}\}$

$$\begin{aligned}\min_{U^{(1)},\ldots,U^{(k)}} & \frac{1}{2}\ \|\mathcal{P}_\Omega(\mathcal{T}-[\![U^{(1)},\ldots,U^{(k)}]\!])\|_F^2+\sum_{i=1}^{k}\frac{\lambda_i}{2}\|(U^{(j)})^{\odot_{j\neq i}}\|_F^2 \\ & +\sum_{i=1}^{k}\frac{\eta_t}{2}\|U^{(i)}-B_t^{(i)}-(1/\eta_t)Y_t^{(i)}\|_F^2. \end{aligned} \tag{6.22}$$

We apply the alternating minimization method to update each $U^{(i)}$ for $i=1,\ldots,k$, while fixing the other variables. Then problem (6.22) becomes a quadratic optimization problem. For convenience, we ignore the subscript in the fixed $U^{(j)}$ for $j\neq i$, and the resulting subproblem with respect to $U^{(i)}$ is

formulated as

$$\min_{U^{(i)}} \frac{1}{2}\|\mathcal{P}_{\Omega^{(i)}}(\mathcal{T}_{(i)} - U^{(i)}[(U^{(j)})^{\odot_{j\neq i}}]^{\mathrm{T}})\|_F^2 + \sum_{\substack{j=1\\ j\neq i}}^{k} \frac{\lambda_j}{2}\|(U^{(n)})^{\odot_{n\neq j}}\|_F^2$$

$$+ \frac{\eta_t}{2}\|U^{(i)} - B_t^{(i)} - \frac{Y_t^{(i)}}{\eta_t}\|_F^2, \tag{6.23}$$

which is separable by rows of $U^{(i)}$. Thus, each row of the new iterate $U_{t+1}^{(i)}$ is the solution to the following linear equation in $\mathbb{R}^R$,

$$(A_j^{(i)} + \eta_t I_R + C^{(i)})(U_{j,:}^{(i)})^{\mathrm{T}} = [(\mathcal{P}_{\Omega^{(i)}}\mathcal{T}_{(i)})(U^{(j)})^{\odot_{j\neq i}} + \eta_t B_t^{(i)} + Y_t^{(i)}]_{j,:}^{\mathrm{T}}, \tag{6.24}$$

where $A_j^{(i)}$ and $C^{(i)}$ are defined in (6.9) and (6.11) respectively, for $j = 1, \ldots, m_i$.

Updating $\{B_{t+1}^{(1)}, \ldots, B_{t+1}^{(k)}\}$: By alternating minimization method, the optimization problem (6.20) can be reformulated as follows when updating the variables $\{B_{t+1}^{(1)}, \ldots, B_{t+1}^{(k)}\}$,

$$\min_{B^{(i)}} \frac{\lambda_i}{2}\langle B^{(i)}(B^{(i)})^{\mathrm{T}}, L^{(i)}\rangle + \frac{\eta_t}{2}\|U_{t+1}^{(i)} - B^{(i)} - (1/\eta_t)Y_t^{(i)}\|_F^2. \tag{6.25}$$

which boils down to solving

$$(\eta_t I_{m_i} + \lambda_i L^{(i)})B^{(i)} = \eta_t U_{t+1}^{(i)} - Y_t^{(i)}. \tag{6.26}$$

Followed by the above standard procedure of ADMM, it concludes in Algorithm 6.1.5. Here we adopt the CG method combined with the Hessian-vector product defined in the previous subsection to update $B_{t+1}^{(i)}$ in (6.26) (also corresponding to line 10 of Algorithm 6.1.5). The iterate $U_{t+1}^{(i)}$ in (6.24) is updated by rows, therefore a simple CG solver is used.

Computational cost of ADMM. We analyze the computational cost for each alternating step of the augmented Lagrangian step (6.18) corresponding to the procedure required by line 6–line 12 of Algorithm 6.1.5. The costs of forming $(U^{(j)})^{\odot_{j\neq i}}$, $C^{(i)}$ and $Q^{(i)}$ are computed in the complexity analysis part of Section 6.1.1. Let n_{ADMM} denote the maximal number of iterations required for solving the linear equations (6.24) and (6.26). Then the dominant costs are $n_{\mathrm{ADMM}}O(m_i R + |\Omega|R)$ and $n_{\mathrm{ADMM}}O(\mathrm{nnz}(L^{(i)})R)$ respectively. Therefore, the total complexity of Algorithm 6.1.5 (ADMM) is $n_{\mathrm{ADMM}}O(\mathrm{nnz}(L^{(i)})R + |\Omega|R)$, which is of the same order as Algorithm 6.1.2 (AltMin-CG).

Algorithm 6.1.5 ADMM for problem (6.2)

Input: Observed tensor $\mathcal{P}_\Omega(\mathcal{T})$, graph Laplacian $\mathbf{Lap}^{(1)}, \ldots, \mathbf{Lap}^{(k)}$, observed set Ω, parameters $\lambda_1, \ldots, \lambda_k$ and λ_L

Output: $(U_t^{(i)})_{i=1,\ldots,k}$

Initialization: $U_0^{(1)}, \ldots, U_0^{(k)}, \eta_0$
for $t = 0, 1, 2, \ldots,$ **do**
  if stopping criterion is satisfied **then**
    return;
  end if
  for $i = 1, \ldots, k$ **do**
    for $j = 1, \ldots, m_i$ **do**
      Update the j-th row of $U_{t+1}^{(i)}$ by (6.24)
    end for
    Update $B_{t+1}^{(i)}$ by (6.26)
    $Y_{t+1}^{(i)} = Y_t^{(i)} + \eta_t^{(i)}(B_{t+1}^{(i)} - U_{t+1}^{(i)})$
  end for
  Update $\eta_{t+1} = \gamma \eta_t$
end for

6.2 Convergence analysis

In this section, we will show the global convergence of iterates $\{U^{(1)}, \ldots, U^{(k)}\}$ generated by Algorithm 6.1.2 (AltMin-CG) to a critical point.

6.2.1 Preliminaries

The following definitions and lemmas are used for the convergence analysis in the next subsection.

Definition 6.2.1 ([RW09],[ABRS10, Definition 1]). *Let $f : \mathbb{R}^m \mapsto \mathbb{R} \cup \{+\infty\}$ be proper and lower semicontinuous.*
1) The domain of f is defined and denoted by $domf := \{x \in \mathbb{R}^m : f(x) < +\infty\}$.
2) For each $x \in domf$, the Fréchet subdifferential of f at x, denoted as $\hat{\partial} f(x)$, is

$$\hat{\partial} f(x) = \left\{ \xi \in \mathbb{R}^m : \liminf_{\substack{y \neq x \\ y \to x}} \frac{f(y) - f(x) - \langle \xi, y - x \rangle}{\|y - x\|} \geq 0 \right\}.$$

If $x \notin domf$, then $\hat{\partial} f(x) = \emptyset$.
3) The limiting subdifferential of f at $x \in domf$, denoted as $\partial f(x)$, is defined as follows

$$\partial f(x) := \{\xi^* \in \mathbb{R}^m : \exists (x_n)_{n \geq 0}, x_n \to x, f(x_n) \to f(x) \text{ and } \xi_n \in \hat{\partial} f(x_n) \to \xi^*\}.$$

Definition 6.2.2 (KŁ function)**.** *[XY13b, Definition 2.5] A function $f(x)$ satisfies the Kurdyka-Łojasiewicz (KŁ) property at point $\bar{x} \in dom(\partial f)$ if, in a certain neighborhood $\mathcal{U}$ of $\bar{x}$, there exists $\psi(s) = cs^{1-\theta}$ for some $c > 0$ and $\theta \in [0, 1)$ such that the KŁ inequality below holds:*

$$\psi'(f(x)-f(x^*))dist(0, \partial f(x)) \geq 1, \text{ for any } x \in \mathcal{U} \cap dom(\partial f) \text{ and } f(x) \neq f(x^*), \tag{6.27}$$

where $dom(\partial f) = \{x : \partial f(x) \neq \emptyset\}$ and $dist(0, \partial f(x)) = \min\{\|y\| : y \in \partial f(x)\}$. If f satisfies the KŁ property at each point of $dom(f)$, f is called a KŁ function.

Definition 6.2.3 (Strong convexity)**.** *A differentiable function $f : domf \mapsto \mathbb{R}$ is strongly convex if and only if*

$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle + \frac{\mu}{2}\|y - x\|^2 \tag{6.28}$$

holds for some $\mu > 0$ and all $x, y \in domf$.

Definition 6.2.4 (Coercivity)**.** *A real-valued function $f : \mathbb{R}^m \to \mathbb{R}$ is called coercive if and only if*

$$f(x) \to +\infty \quad as \quad \|x\| \to +\infty. \tag{6.29}$$

The following two lemmas follow directly from Theorem 2.8 and Theorem 2.9 of [XY13b] and are used for proving the convergence of the proposed alternating minimization method.

Lemma 6.2.5. *Assume f satisfies the KŁ property and ∇f is Lipschitz continuous on any bounded subset of its domain. Let $(U_0^{(1)}, \ldots, U_0^{(k)})$ be any initialization and $(U_t^{(1)}, \ldots, U_t^{(k)})$ be the sequence generated by Algorithm 6.1.1, where each subproblem $f_t^{(i)}(U^{(i)})$ (line 7 of Algorithm 6.1.1) is strongly convex and is solved exactly. If the sequence $(U_t^{(1)}, \ldots, U_t^{(k)})$ is bounded and there exists a finite limit point $(U_*^{(1)}, \ldots, U_*^{(k)})$, then it converges to $(U_*^{(1)}, \ldots, U_*^{(k)})$, which is a critical point of f.*

The convergence rate of the sequence is as follows.

Lemma 6.2.6. *Assume ∇f is Lipschitz continuous on any bounded set and suppose that $U_t^{(i)}$ converges to a critical point $U_*^{(i)}$ for $i = 1, \ldots, k$, at which f satisfies the KŁ inequality with $\psi(s) = cs^{1-\theta}$ for constants $c > 0$ and $\theta \in [0, 1)$. Then:*
1. If $\theta = 0$, $U_t^{(i)}$ converges to $U_*^{(i)}$ in a finite number of iterations;
2. If $\theta \in (0, \frac{1}{2}]$, $\|U_t^{(i)} - U_*^{(i)}\| \leq \beta\tau^t$, $\forall t \geq t_0$ for certain $t_0 > 0$, $\beta > 0$, $\tau \in [0, 1)$;
3. If $\theta \in (\frac{1}{2}, 1)$, $\|U_t^{(i)} - U_*^{(i)}\| \leq \beta t^{-(1-\theta)/(2\theta-1)}$, $\forall t \geq t_0$ for certain $t_0 > 0$, $\beta > 0$.
Part 1, 2 and 3 correspond to finite convergence, linear convergence, and sublinear convergence, respectively.

6.2.2 Convergence properties of AltMin

We show that the iterates generated by Algorithm 6.1.1 (AltMin) converge to a stationary point in the following theorem. Note that this theorem applies to Algorithm 6.1.2 (AltMin-CG), provided that the updated iterate of each of the subproblems (line 8 in Algorithm 6.1.2) is the exact minimizer of the corresponding (graph-regularized) least-squares problem. In practice, this requires setting a sufficiently low tolerance parameter ϵ for the subproblem solver (Algorithm 6.1.3).

Theorem 6.2.7. *The iterates $(U_t^{(1)}, \ldots, U_t^{(k)})$ generated by Algorithm 6.1.1 (AltMin) from any initialization converge globally to a critical point of f in* (6.2)*. Moreover, linear convergence and sublinear convergence in parts 2 and 3 of Lemma 6.2.6 apply depending on θ in KŁ property of f.*

Proof. According to Lemma 6.2.5, we need to check whether all the assumptions satisfied.

1) Function f in (6.2) is a KŁ function with $\theta \in [1/2, 1)$ as it is a combination of polynomials which are one kind of real analytic functions (see [KP02, Definition 1.1.5]). The real analytic function itself and the finite sum or product of real analytic functions are KŁ functions, see [ABRS10, section 4] and [XY13b, section 2.2].

2) Gradient ∇f is Lipschitz continuous on any bounded subset of domain since f is a C^∞ function.

3) For $i = 1, \ldots, k$, $f^{(i)}$ in (6.5) is strongly convex by Definition 6.2.3 since $L^{(i)}$ in (6.3) is positive definite. Therefore, the quadratic form $g^{(i)}$ of the subproblems (6.6) is strongly convex through the identification $g^{(i)}\big(\mathrm{vec}\big((U^{(i)})^{\mathrm{T}}\big)\big) = f^{(i)}(U^{(i)})$. Moreover, the solution for each $g^{(i)}$ corresponds to the exact minimizer.

4) Notice that since f is coercive as defined in Definition 6.2.4 and real analytic, it is guaranteed to produce a bounded sequence $(U_t^{(1)}, \ldots, U_t^{(k)})$, thus it has a critical point $(U_*^{(1)}, \ldots, U_*^{(k)})$.

Lemma 6.2.5 then implies that the sequence generated by Algorithm 6.1.2 from any initial point converges to a critical point $(U_*^{(1)}, \ldots, U_*^{(k)})$ of f. Moreover, the asymptotic convergence rates in parts 2 and 3 of Lemma 6.2.6 apply as $\theta \in [1/2, 1)$. □

6.3 Experiments

In this section, we carry out some numerical experiments to demonstrate the working of our proposed algorithms Algorithm 6.1.2 (AltMin-CG) and Algorithm 6.1.5 (ADMM) on the LRTC model (6.2) with $k = 3$. All numerical experiments were performed on a Macbook Pro with a 2.3 GHz Intel Core i7 CPU,

16GB RAM and MATLAB R2015a with Tensor Toolbox version 2.5 [BK+12]. The source code is made available online.[3]

We focus on the following two tasks:

1. To test and verify the effect of the graph Laplacian-based regularizer in our model (6.2), we compare the recovery quality of solutions given by the graph-regularized tensor completion model with two graph-agnostic tensor completion models.

2. To validate the effectiveness and efficiency of the proposed methods Algorithm 6.1.2 and Algorithm 6.1.5 when applied to solving model 6.2, we compare with other baseline methods on both synthetic data and real data.

In our experiments, we evaluate the quality of a tensor approximation with the following error functions, for a given index set Ω' that contains only the known entries of $\mathcal{T}^\star$: (i) the relative error (RE) of the tensor candidate $\mathcal{T} = [\![U^{(1)}, \cdots, U^{(k)}]\!]$ against $\mathcal{T}^\star$ on Ω' in the Frobenius norm, and (ii) the root mean squared error (RMSE) of $\mathcal{T}$ restricted on Ω'. The training and test RMSEs refer to the RMSE on the training and test set respectively.

Initialization We initialize both our proposed methods and other methods with a point $U_0 \in \mathbb{R}^{m_1 \times R} \times \mathbb{R}^{m_2 \times R} \times \mathbb{R}^{m_3 \times R}$ where each factor matrix $U_0^{(i)}$ is a Gaussian matrix such that $[U_0^{(i)}]_{jr} \sim \mathcal{N}(0, 1)$.

Experimental methodology Based on a ground truth tensor $\mathcal{T} \in \mathbb{R}^{m_1 \times m_2 \times m_3}$ and for a fixed sampling rate ρ, we generate N_{test} training instances $(\Omega_\ell)_{\ell=1,\ldots,N_{\text{test}}}$ under the same sampling rate ρ. For each training instance $(\mathcal{T}, \Omega_\ell)$, N_{init} initial points $(U_{0,(\ell,j)})$, for $j = 1, .., N_{\text{init}}$, are generated. Let $\hat{\mathcal{T}}(U_{0,(\ell,j)}; \Omega_\ell)$ denote the solution of the j-th test based on the training instance $(\mathcal{T}, \Omega_\ell)$ with the initial point $U_{0,(\ell,j)}$, and $E(\hat{\mathcal{T}})$ the error (e.g., RE, RMSE) of the candidate tensor $\hat{\mathcal{T}}$ w.r.t. $\mathcal{T}$. Then each method is evaluated by the following score

$$\text{Err} = \frac{1}{N_{\text{test}} N_{\text{init}}} \sum_{\ell=1}^{N_{\text{test}}} \sum_{j=1}^{N_{\text{init}}} E(\hat{\mathcal{T}}(U_{0,(\ell,j)}); \Omega_\ell). \tag{6.30}$$

In all experiments we select $N_{\text{test}} = 5$ and $N_{\text{init}} = 10$.

Parameter selection In all experiments, the problem-related parameters (λ_i, λ_L) in (6.2)–(6.3) are generated randomly with the uniform distribution in the log scale. Then the parameter is chosen among all generated parameter settings through K-fold cross validation (for $K = 3$).

[3] `https://gitlab.com/ricky7guanyu/tensor-completion-with-regularization-term`.

6.3.1 Graph-regularized and graph-agnostic tensor completion models

In this subsection, we conduct tests to compare the graph-regularized tensor completion model with graph-agnostic models. Hereafter, we refer to our graph-regularized tensor completion model (6.2) as GREG-TC, when λ_i and λ_L are nonzero. The model (6.2) is referred to as NUCLREG-TC when the problem parameter for the graph Laplacian-based regularization term is reduced to zero, i.e., $\lambda_L = 0$, but the other regularizers are kept active (λ_i nonzero). The model (6.2) is referred to as UNREG-TC when all regularization parameters are reduced to zero, i.e., $\lambda_i = \lambda_L = 0$.

In the experiments of this subsection, the stopping criteria consist of (i) a large iteration budget parameter $T_{\max}$; and (ii) a global tolerance parameter ϵ for (6.13).

Experiment on synthetic data with graph information

As our model (6.2) has a graph regularizer term, it is tempting to generate a synthetic low rank tensor $\mathcal{T}$ that features some graph information inside.

Here is how we construct such a synthetic tensor. First, we generate the graph Laplacian matrix $\mathbf{Lap}^{(1)} \in \mathbb{R}^{100\times 100}$ of $U^{(1)}$ which exhibits row-wise similarities by the prototypical graph model "Community" using GSP-box [PPS+14]. Let $\mathbf{Lap}^{(1)} = \bar{U}\bar{\Lambda}\bar{U}^{\mathrm{T}}$ be the eigenvalue value decomposition of a given graph Laplacian matrix. We consider the following tensor model,

$$\mathcal{T} = [\![\tilde{A}U^{(1)}, U^{(2)}, U^{(3)}]\!] + \mathcal{E}, \tag{6.31}$$

where $\tilde{A} = \bar{U}\bar{\Lambda}^{-1}$ and $(U^{(1)}, U^{(2)}, U^{(3)}) \in \mathbb{R}^{m_i \times R}$ are random matrices whose columns are i.i.d. Gaussian vectors. The tensor $\mathcal{E}$ contains additive noise such that $\mathcal{E}_{\ell_1\ell_2\ell_3} \sim \mathcal{N}(0, \sigma)$, where σ is set according to a given signal-to-noise ratio (SNR). In this experiment, we set SNR to 20 dB. The graph information is incorporated in the tensor model as follows. By construction, the first factor matrix belongs to an eigensubspace of a given graph Laplacian matrix with a prescribed order of "preferences" for each of the eigenvectors (here the preferences are given by $(1/\bar{\lambda}_i)_{i=1,\ldots,R}$). Here, the model enables building a tensor where the entries along the first dimension exhibit pairwise similarities according to the connections of the given graph. This model can be seen in the related work on graph-based regularization for matrix completion [RYRD15, DAG20].

We test the tensor completion performances on the synthetic tensor (6.31), for $R = 10$. The sample rates (SR) tested are $\{0.3\%, 0.5\%, 0.7\%, 1\%\}$ and the rank parameter R is set to 10 for all the models tested. The average relative error (RE) are given in Table 6.1. Figure 6.1 shows the histogram of the relative errors for $SR = 0.3\%$. We see that on this experiment, our GREG-TC model outperforms both other models, for both tested methods.

Table 6.1: Average recovery accuracies (relative error) of the three models on synthetic data: GReg-TC (with graph Laplacian), NuclReg-TC (without graph Laplacian), unreg-TC (no regularizer).

Sample Ratio	Algorithm	GReg-TC	NuclReg-TC	unreg-TC
0.3%	AltMin-CG	0.8198	1.0083	7.1110
	ADMM	0.8246	1.0054	4.4393
0.5%	AltMin-CG	0.4418	0.9201	9.9010
	ADMM	0.4486	0.9236	7.6692
0.7%	AltMin-CG	0.3106	0.7561	9.9548
	ADMM	0.3000	0.7487	8.4821
1%	AltMin-CG	0.1380	0.4772	13.5740
	ADMM	0.1439	0.4513	10.9230

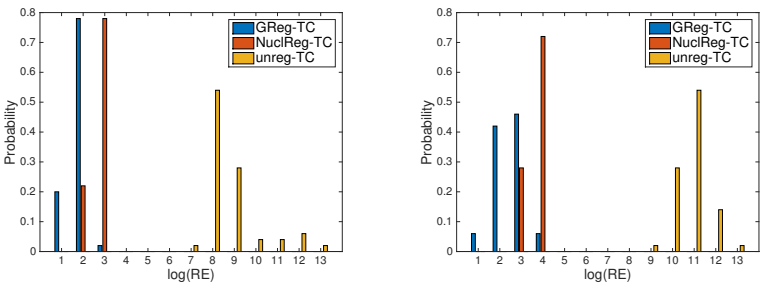


Figure 6.1: Recovery accuracies of the three tensor completion models on synthetic data. The sampling rate $SR = 0.3\%$.

Experiments on real data

Next, we consider experimenting from two real-world datasets. An essential difference between these experiments and the experiments on synthetic data is that there is no ground-truth similarity graphs associated with the data tensor. While we assume that real life data often present pairwise similarities between its entries, we need to build the graph Laplacians $\mathbf{Lap}^{(i)}$.

MovieLens. We now evaluate the models on a movie rating dataset called MovieLens dataset 100k,[4] which consists of 100,000 movie ratings from 943 users on 1682 movies during a seven-month period from September 19th, 1997 through April 22nd, 1998. Each movie rating in this dataset has a time stamp. Therefore, we obtain a tensor $\mathcal{T}$ of size $943 \times 1682 \times 7$ (*i.e.*, time period is split into 7 parts). We randomly select 80% of the known ratings as training set.

In this experiment, we construct a movie-wise similarity graph based on the data matrix itself (with missing entries). Let $M^\star$ denote the MovieLens data matrix with missing entries. We compute the graph proximity parameters based on a low-rank approximation of the partially revealed matrix. More precisely, we use a rank-r approximation of the zero-filled matrix $M_0 := P_\Omega(M^\star) \in \mathbb{R}^{m\times n}$ as the features for constructing the graph. Let (U_0, S_0, V_0) denote the r-SVD of M_0 and let $\widetilde{M_0} := U_0 S_0 V_0^T$.

Next, the computation of the graph edge weight parameters based on the given matrix $M := \widetilde{M_0}$ can be realized using various node proximity methods such as K-Nearest Neighbors (K-NN) and ε-graph models [Cha83, BN03, HN04, CGS09], which boil down to computing a certain distance matrix between the rows (resp. columns) of M. Let $Z^{\mathrm{r}}(M) \in \mathbb{R}^{m\times m}$ denote the row-wise distance matrix of M defined as $Z_{ij}(M) = d(M_{i,:}, M_{j,:})$, for $i, j \in [\![m]\!]$, where $d : \mathbb{R}^n \times \mathbb{R}^n \mapsto \mathbb{R}_+$ is a distance on the n-dimensional vector space. Subsequently, we build a Gaussian ε-graph by computing the node proximity weights as follows

$$[W_\varepsilon(M)]_{ij} = \exp\left(-Z_{ij}(M)/\varepsilon^2\right), \text{ for } i, j \in [\![m]\!], \tag{6.32}$$

where $\varepsilon \in \mathbb{R}$ is a hyperparameter of the graph model. Furthermore, a sparse graph adjacency matrix is preferable to a dense one from a computational point of view, as the per-iteration cost of the proposed algorithms (*e.g.* Algorithm 6.1.2) depends partly on $\mathrm{nnz}(L^{\mathrm{r}})$ and $\mathrm{nnz}(L^{\mathrm{c}})$. For simplicity, we sparsify the graph adjacency matrix defined in (6.32) with the following thresholding operation

$$[W_{\varepsilon,\sigma}(M)]_{ij} = 1_{\geq\sigma}\left(\exp\left(-Z_{ij}(M)/\varepsilon^2\right)\right), \text{ for } i, j \in [\![m]\!], \tag{6.33}$$

where $1_{\geq\sigma}$ is the hard threshold function $1_{\geq\sigma}(z) = \begin{cases} z & \text{if } z \geq \sigma \\ 0 & \text{otherwise.} \end{cases}$

In the graph model (6.33), parameter ε is tuned according to the variance

[4] https://grouplens.org/datasets/movielens/100k/

Table 6.2: RE of the three models on the MovieLens dataset: GReg-TC (with graph Laplacian), NuclReg-TC (without graph Laplacian), unreg-TC (no regularizer).

Method	GReg-TC	NuclReg-TC	unreg-TC
AltMin-CG	0.2233	0.2233	1.4526
ADMM	0.2193	0.2234	1.1205

of $(Z_{ij})_{i,j=1,..,m}$ and threshold σ is chosen according to a preset sparsity level $\mathfrak{s} \ll 1$ for the edge set associated with $W_{\epsilon,\sigma}$ such that $|\mathcal{E}(W_{\epsilon,\sigma})|/m^2 \leq \mathfrak{s}$.

Given the graphs described above, we test the tensor completion performances using a rank parameter $R = 10$. The results are obtained after 50 repeated tests with random initialization; see Table 6.2, while the histogram of these results is presented in Figure 6.2. These results show that the solutions to the graph-regularized model (6.2) (labeled GReg-TC) and NuclReg-TC model have better recovery performance (in terms of RMSE) than unreg-TC model. The gain of recovery performance induced by the graph learned from the data may be considered marginal; however, on movie rating data, it is notoriously difficult to improve the RMSE score much beyond basic methods [BL+07].

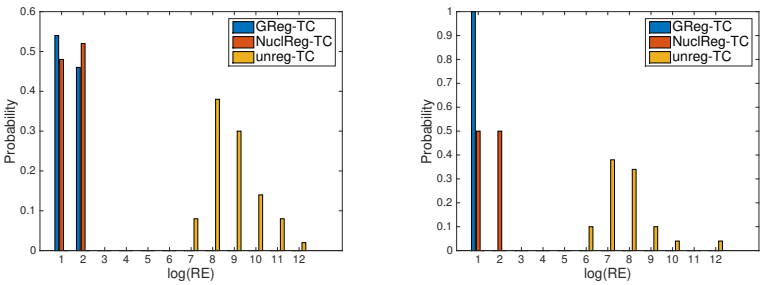


Figure 6.2: Recovery accuracies (relative error) of the three tensor completion models on the MovieLens dataset.

The FIA dataset. In this experiment, we use the "rank-deficient spectral FIA dataset".[5] This dataset consists of results of flow injection analysis on 12 different chemical substances. The represented tensor is of size 12(substances)$\times$ 100(wavelengths) $\times$ 89(reaction times). We construct the adjacency matrices following the ideas in [NHTK12, Section 4.1]. For 12 chemical substances, we build the adjacency matrix between two substances as the inverse of Euclidean distance between their feature vectors. For wavelengths and reaction times, the adjacency matrices are built using the simple chain graph model since these quantities varies in a smooth manner in their respective domains of definition.

[5] http://www.models.life.ku.dk/datasets

Table 6.3: RE of the three models on the FIA dataset: GReg-TC (with graph Laplacian), NuclReg-TC (without graph Laplacian), unreg-TC (no regularizer).

Sample Ratio	Algorithm	GReg-TC	NuclReg-TC	unreg-TC
1%	AltMin-CG	0.1012	1.1096	6.6597
	ADMM	0.2396	1.1385	1.5985
5%	AltMin-CG	0.0146	0.3470	3.9619
	ADMM	0.0209	0.2131	0.2131
7%	AltMin-CG	0.0126	0.2087	0.3837
	ADMM	0.0180	0.0572	0.0572

The sample ratio varies the fraction of observed entries as 1%, 5% and 7%. We present in Table 6.3 average results after running 50 times; the histogram for $SR = 1\%$ is shown in Figure 6.3. Similar to the comparative results on synthetic data, these results show that the solutions of the GReg-TC model have smaller recovery errors compared to those of the other two models.

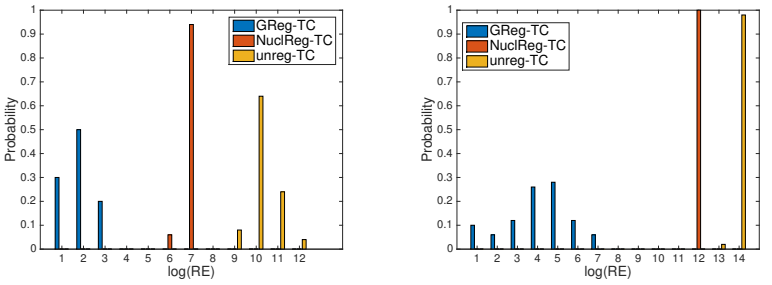


Figure 6.3: Recovery accuracies (relative error) of the three tensor completion models on the FIA dataset. The sampling rate $SR = 1\%$.

Results and observations. The results in Tables 6.1–6.3 show that the solutions of the graph-regularized model (6.2) (labeled by GReg-TC) have smaller recovery errors than those of the two graph-agnostic models, NuclReg-TC and unreg-TC. This observation is encouraging since it indicates that incorporating graph auxiliary information in the tensor completion model leads to more robust recovery results, especially when the proportion (sampling rate) of the revealed entries is very low.

The graph Laplacians in synthetic data are comparably easy to build using the synthetic data itself, which always has the ground truth. In comparison, for real data, we need to explore and construct the adjacency matrix from the observed data, which is only a small portion of the data. Moreover, sometimes the data itself is not well collected. Notice that in the MovieLens experiment, we build the graph Laplacians only in user mode without metadata about the background information of users. Therefore, the GReg-TC and NuclReg-TC models perform relatively similar, and both demonstrate lower relative

errors. With the increase of the sample ratio, the effect of graph Laplacians is getting weaker as shown in Table 6.3.

Moreover, from histograms, we find that the proposed GReg-TC model and NuclReg-TC model display better consistency than the unreg-TC model, as the results of 50 runs are quite close, implying that the proposed model is more robust on predicting the missing data. The last values of log(RE) in the GReg-TC model always stay at the same bar or at the adjacent bar, and are located left of those of the NuclReg-TC model and unreg-TC model.

6.3.2 Time efficiency comparisons

In this subsection, we focus on evaluating the time efficiency of our proposed algorithms under the same experimental settings described in the last subsection. In each of the following experiments, iteration information for our proposed algorithms is recorded. At each iteration, the recovery quality of the current iterate is evaluated in terms of the RMSE on test entries.

Besides our proposed algorithms, a representative selection of state-of-art methods are also tested, based on their codes that are publicly available or made available to us: (i) INDAFAC is a damped Gauss-Newton method proposed by Tomasi and Bro [TB05] for solving the following tensor completion model

$$\min_{\mathcal{T},U} \|\mathcal{T}_{(1)} - U^{(1)}(U^{(3)} \odot U^{(2)})^{\mathrm{T}}\|_F^2 \text{ s.t } \quad \mathcal{P}_\Omega(\mathcal{T} - \mathcal{Z}) = 0,$$

where $\mathcal{Z}$ is only known on Ω; (ii) CP-WOPT [ADKM11] is an algorithm for solving the CPD-based problem $\min_U \|P_\Omega([\![U^{(1)}, U^{(2)}, U^{(3)}]\!] - \mathcal{T}^\star)\|_F^2$ and is available in the Tensor Toolbox [BK+12]; (iii) BPTF is a Bayesian probabilistic tensor CPD algorithm [XCH+10] and solves a regularized problem

$$\min_U \|P_\Omega([\![U^{(1)}, U^{(2)}, U^{(3)}]\!] - \mathcal{T}^\star)\|_F^2 + \psi(U)$$

where the regularizer ψ is composed of Frobenius norms of the factors of U and a ℓ_2 norm-based function imposing columnwise smoothness of $U^{(3)}$; (iv) TFAI [NHTK12] is an algorithm for tensor completion with auxiliary information. We adopt the within-mode auxiliary information method with the graph Laplacians shown as

$$\min_{\mathcal{T},U} \|\mathcal{T} - [\![U^{(1)}, U^{(2)}, U^{(3)}]\!]\|_F^2 + \psi(U) \text{ s.t } \quad \mathcal{P}_\Omega(\mathcal{T}) = \mathcal{P}_\Omega(\mathcal{T}^\star),$$

where ψ is a graph Laplacian-based regularizer as in (6.2); (v) TNCP [LSJ+14] is an ADMM algorithm for solving the following matrix trace-norm regularized problem

$$\min_{\mathcal{T},U} \frac{\lambda}{2}\|\mathcal{T} - [\![U^{(1)}, U^{(2)}, U^{(3)}]\!]\|_F^2 + \sum_{i=1}^{3} \alpha_i \|U^{(i)}\|_* \text{ s.t } \quad \mathcal{P}_\Omega(\mathcal{T}) = \mathcal{P}_\Omega(\mathcal{T}^\star);$$

and (vi) AirCP [GCZS16] is an ADMM algorithm for solving the CP-based tensor completion using auxilary graph information

$$\min_{\mathcal{T},U,X} \frac{\lambda}{2}\|\mathcal{T}-[\![U^{(1)},U^{(2)},U^{(3)}]\!]\|_F^2+\psi(U,X) \text{ s.t } \quad \mathcal{P}_\Omega(\mathcal{T})=\mathcal{P}_\Omega(\mathcal{T}^\star), U^{(i)}=X^{(i)},$$

$\psi(U,X)$ is composed of the sum of Frobenius norms of $U^{(i)}$ and the graph Laplacian-based norms of $X^{(i)}$.

Remark 6.3.1. *The parameters involved in the models of TFAI, TNCP and AirCP are chosen after cross validation.*

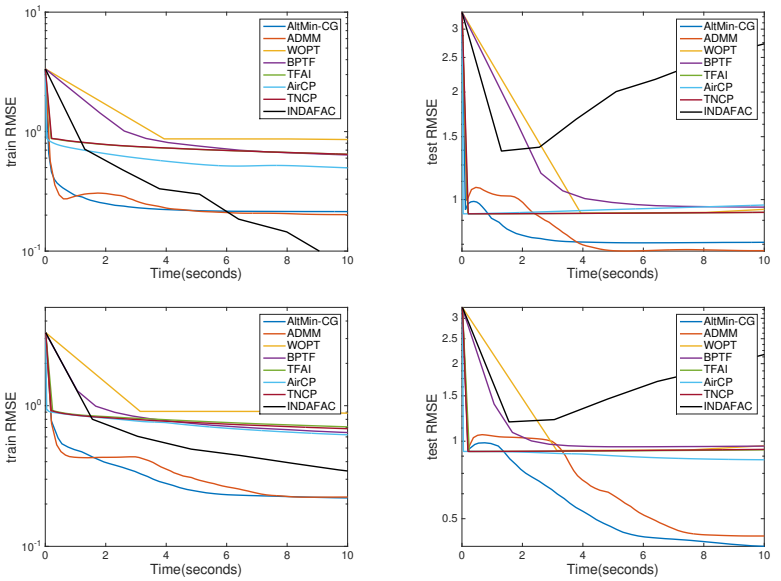


Figure 6.4: Iterative results of the tested algorithms on synthetic data: RMSE on training and test sets by accumulative time per-iteration. The sampling rates tested are 0.3% and 0.5%.

Tests on synthetic data. Under the same experimental settings as for Table 6.1, the RMSEs of the iterates given by the tested methods are shown in Figure 6.4. The iterative results in the figures are taken from one test randomly chosen from the repeated tests. Figure 6.4 shows the results under the sampling rates $SR = 0.3\%$ and 0.5%, in which we observe that proposed algorithms, AltMin-CG and ADMM, have a better time efficiency than the rest of the tested methods. Detailed observations on these results are given in Section 6.3.2.

Tests on the MovieLens dataset. Under the same experimental settings on the MovieLens dataset and with the same graph construction method as in the previous subsection, we show the RMSEs of the iterates given by the tested methods in Figure 6.5. The iterative results in the figure are taken from one test randomly chosen from the repeated tests. In particular, the labels "AltMin-CG1" and "ADMM1" represent the results of these algorithms under the graph-agnostic, nuclear norm-based model (NuclReg-TC). The results therein shows that the AltMin-CG and ADMM methods remain competitive on this dataset, in particular their time efficiency is comparable to AirCP.

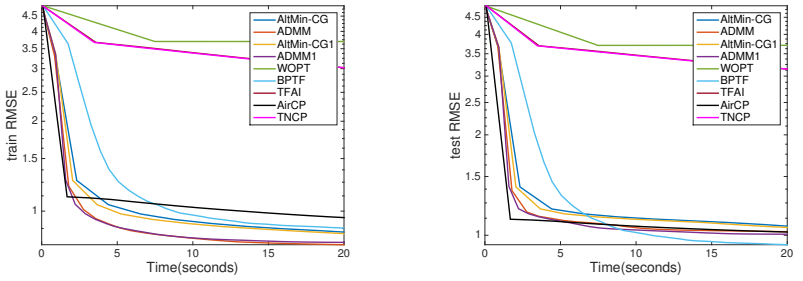


Figure 6.5: Iterative results of the tested algorithms on the MovieLens dataset: RMSE on training and test sets by accumulative time per-iteration.

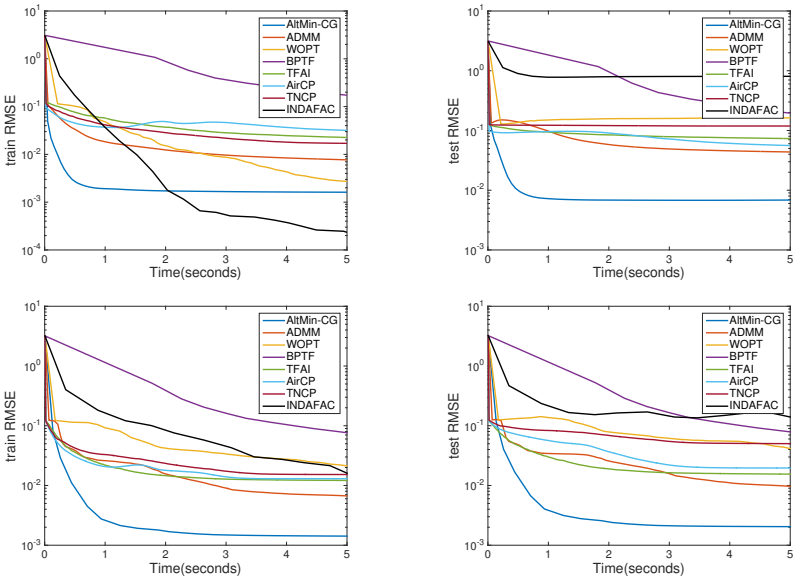


Figure 6.6: Iterative results of the tested algorithms on the FIA dataset: RMSE on training and test sets by accumulative time per-iteration.

Tests on the FIA dataset. Under the same experimental settings on the FIA dataset and with the same graph construction method as in the previous subsection, we show the RMSEs of the iterates given by the tested methods in Figure 6.6. The iterative results in the figure are taken from one test randomly chosen from the repeated tests. These results show that the proposed algorithms AltMin-CG and ADMM yield the best recovery performances (in terms of test RMSE).

Results and observations

From Figure 6.4, we observe that most of the methods perform comparably in time efficiency. In particular, the proposed AltMin-CG and ADMM methods, followed by AirCP, TFAI, TNCP, are the fastest in terms of training RMSE. Also, AltMin-CG and ADMM achieve the lowest test RMSEs, under low sampling rates. This result is encouraging since it shows that our methods and model for tensor completion can achieve robust recovery where only a small fraction of data is available.

For real data in Figure 6.5, AirCP, TFAI, AltMin-CG and ADMM have comparable time efficiencies and are the fastest among all tested algorithms in terms of training error. BPTF is slower than the aforementioned methods but obtains the lowest test RMSE. In Figure 6.6, we also observe that the proposed AltMin-CG and ADMM methods outperform most other methods both in efficiency and accuracy with the given training data that correspond to low sampling rates.

6.4 Conclusion

Built on the ideas of a nuclear norm based approach, CP decomposition and graph Laplacian regularizers, we constructed a new low-rank tensor completion model (6.2), also motivated by results in [RYRD15] on matrix completion with graph Laplacian. The advantage of the proposed model is twofold: (i) the CP-based decomposition enables a memory-efficient model for low-rank tensors and (ii) the use of the graph Laplacian-based regularizer yields completion results with higher recovery accuracy than several classical, graph-agnostic tensor completion models. An alternating minimization (AltMin) method is proposed. The minimization of each of the subproblems is done by a linear CG routine, in which an efficient Hessian-vector multiplication is used. Besides, the alternating directions method of multipliers (ADMM) method is also applied to the LRTC model which shows competitive performance. Moreover, the convergence analysis for the proposed AltMin algorithm is given. From the results of various numerical experiments, we verified that the graph-regularized tensor completion model (6.2) produces solutions with better recovery performances compared to graph-agnostic tensor completion models. This observation is especially significant when the sample rate is small. Our proposed algorithms

for solving the graph-regularized tensor completion problem, AltMin-CG (Algorithm 6.1.2) and ADMM (Algorithm 6.1.5), have been shown to be time efficient compared with other state-of-art methods.

Chapter 7

Riemannian preconditioned algorithms for tensor completion

In this chapter, we focus on the low-rank tensor decomposition approach to tensor completion and develop new algorithms using Riemannian preconditioning. This chapter is based on the paper [DGGG21].

Related work

For the tensor completion problem using the CP decomposition (CPD), Tomasi and Bro [TB05] proposed to used the Levenberg–Marquardt (modified Gauss-Newton) method for third-order tensors, in which the rank-R tensor candidate is represented by the vectorization of all three factor matrices of the CPD; Acar et al. [ADKM11] proposed to use a nonlinear conjugate gradient algorithm; Jain et al. [JO14] proposed an alternating minimization algorithm, which uses a special initialization by the Robust Tensor Power Method [AGH$^+$14] allowing for a guaranteed tensor recovery with a sample complexity lower bound. The Tucker decomposition, as a closely related decomposition format, is also widely used for tensor completion, with more or less different application purposes; see [CMD$^+$15] about the relations and differences between the Tucker and the CP decompositions. Kressner et al. [KSV14] exploited the Riemannian geometry of the rank-constrained search space with the Tucker decomposition and proposed a Riemannian conjugate gradient (RCG) algorithm for the tensor completion problem. Kasai and Mishra [KM16] paid attention to the data fitting function of tensor completion and introduced a preconditioned metric on the quotient space of the rank-constrained search space with the Tucker decomposition; they used a Riemannian conjugate gradient algorithm with respect to the proposed metric. Vervliet and De Lathauwer [VD19] developed

efficient algorithms using the Gauss-Newton method for CPD-based data fusion problems, which also include problems with missing entries. We refer to [SVBD13, SVBD15, SD19] for more recent works about CP decomposition with fully observed or missing entries.

For a $m_1 \times \cdots \times m_k$ tensor that is only partially observed, low-rank tensor completion using CP decomposition can be modeled by approximating the given partially observed tensor with a rank-R tensor candidate in the CP decomposition form $\mathcal{T} = [\![U^{(1)}, \ldots, U^{(k)}]\!]$, where $U^{(i)}$ are $m_i \times R$ matrices with full column-rank and $[\![U^{(1)}, \ldots, U^{(k)}]\!]$ denotes the product of the CP decomposition, which is the sum of the outer products of the respective columns of $U^{(i)}$. In such problem formulations, the CP decomposition not only provides a powerful data representation, but also has an advantageously low memory requirement—which scales as $O((m_1 + \cdots + m_k)R)$ for a given CP rank R—compared to other types of models (e.g., [LMWY12]) that involve otherwise a full dense tensor variable (requiring $O(m_1 m_2 \ldots m_k)$ memory). However, the fixed-rank CP decomposition, as well as other tensor decomposition models with a fixed-rank, requires the choice of an appropriate rank value. Since an optimal rank choice is usually unknown in practice, fixing the tensor rank in the CP decomposition-based (as well as Tucker-based) approaches is not an ideal strategy. Unfortunately, the search or estimation of the optimal rank is also hard [Kru89]. Therefore, the approach with a fixed CP rank limits the applicability of the completion model, and it is natural to study CP decompositions with a rank upper bound (e.g., [LSC+14]). For example, a rank-increasing approach is used (e.g., [YZC16]) during the optimization process for the exploration of an optimal rank.

Contributions and organization of the chapter

In this chapter, we consider the tensor completion problem by modeling low-rank tensors using the polyadic decomposition (PD). More precisely, the k^{th}-order tensor variable is represented by k factor matrices via polyadic decomposition. Thus, we formulate the tensor completion problem as follows,

$$\underset{U:=(U^{(1)},\ldots,U^{(k)})\in\mathcal{M}}{\text{minimize}} \quad \frac{1}{2}\|P_\Omega([\![U^{(1)}, \ldots, U^{(k)}]\!] - \mathcal{T}^\star)\|_{\text{F}}^2 + \psi(U), \tag{7.1}$$

where $[\![U^{(1)}, ..., U^{(k)}]\!]$ denotes the product defined in Definition 2.4.10, $\mathcal{T}^\star$ is the partially observed tensor, Ω is an index set indicating the observed entries of $\mathcal{T}^\star$, P_Ω denotes the orthogonal projector such that the $(i_1, \ldots, i_k)$-th entry of $P_\Omega(Z)$ is equal to $Z_{i_1,\ldots,i_k}$ if $(i_1, \ldots, i_k) \in \Omega$ and zero otherwise, and the search space $\mathcal{M}$ is a product space defined as

$$\mathcal{M} = \mathbb{R}^{m_1 \times R} \times \cdots \times \mathbb{R}^{m_k \times R} \tag{7.2}$$

with a rank parameter R, and $\psi : \mathcal{M} \mapsto \mathbb{R}$ is a regularization function.

Note that the search space of (7.1) includes points such that the actual

CP rank of the product $[\![U^{(1)}, \ldots, U^{(k)}]\!]$ is smaller than the rank parameter R. Hence, the PD representation of the tensor candidate by $U \in \mathcal{M}$ is not necessarily *canonical*. In the context of low-rank tensor completion, the optimal rank of the tensor candidate is unknown, and thus, it is important to choose appropriate parameters of the low-rank model in question. One of the approaches (using low-rank tensor decomposition) in previous work is to explore and adjust the rank parameter R sequentially [YZC16]. In the case of the model using the Tucker decomposition with a fixed Tucker rank [KM16], the most natural way to choosing the Tucker rank is to explore among a range of (k-dimensional) trials.

In this work, we are interested in choosing large enough rank parameters (while maintaining it in the range of values that are much lower than the tensor dimensions) for the model (7.1). To alleviate issues where R is overestimated compared to the rank of the desired solution, we design algorithms that tolerate rank-deficient factor matrices (or *almost* rank-deficient, with close-to-zero tailing singular values). The proposed algorithms are shown in numerical results to be able to reach optimal solutions to (7.1), given a large enough value of R, and thus, avoid lengthy sequential rank explorations.

The main contributions of this chapter are as follows. We design a *preconditioned* metric on the search space $\mathcal{M}$ of the polyadic decomposition model (7.1) and propose Riemannian gradient descent (RGD) and Riemannian conjugate gradient (RCG) algorithms to solve the tensor completion problem.

We prove that the sequence of iterates generated by the RGD algorithm converges to a critical point of the objective function and provide estimates of the convergence rate using the Łojasiewicz property.

We test on synthetic data for recovering a partially observed tensor with and without additive noise. The numerical results show that our algorithms outperform the several existing algorithms. On the real-world dataset (MovieLens 1M), we find that the proposed algorithms are also faster than the other algorithms under various rank choices. Moreover, the tensor recovery performance of our algorithms is not sensitive to the choice of the rank parameter, in contrast to algorithms based on a fixed-rank model.

Organization. We give the definitions and notation for the tensor operations and state the main problem in Section 7.1. The proposed algorithms and convergence analysis are presented in Sections 7.2–7.3. Numerical results are reported in Section 7.4. We conclude the chapter in Section 7.5.

7.1 Problem statement and notation

In this section, we introduce the main problem (7.1). The definitions and notation involved in the tensor operations are given in Section 2.4.

In addition, for a strictly positive integer n, we denote the index set $\{1, \ldots, n\}$ as $[\![n]\!]$. The set of n-dimensional real-valued vectors is denoted by $\mathbb{R}^n$. For

$\ell \in [\![n]\!]$, we denote by $\mathbf{e}_\ell$ the (n-dimensional) vector that indicates 1 in the ℓ-th entry: $[\mathbf{e}_\ell]_\ell = 1$ and $[\mathbf{e}_\ell]_j = 0$ for all $j \neq \ell$. The set of k^{th}-order real-valued tensors is denoted by $\mathbb{R}^{m_1 \times \cdots \times m_k}$. The i-th row and the j-th column of a matrix A are denoted by $A_{i,:}$ and $A_{:,j}$ respectively. An entry of a real-valued k^{th}-order tensor $\mathcal{Z} \in \mathbb{R}^{m_1 \times \cdots \times m_k}$ is accessed via an k-dimensional index $[\ell_j]_{j=1,\ldots,k}$, with $\ell_j \in [\![m_j]\!]$, and is denoted as $\mathcal{Z}_{\ell_1,\ldots,\ell_k}$. The inner product of two tensors $\mathcal{Z}^{(1)}, \mathcal{Z}^{(2)} \in \mathbb{R}^{m_1 \times \cdots \times m_k}$ is defined as follows $\left\langle \mathcal{Z}^{(1)}, \mathcal{Z}^{(2)} \right\rangle = \sum_{i_1=1}^{m_1} \cdots \sum_{i_k=1}^{m_k} \mathcal{Z}^{(1)}_{i_1,\ldots,i_k} \mathcal{Z}^{(2)}_{i_1,\ldots,i_k}$. The Frobenius norm of a tensor $\mathcal{Z}$ is defined as $\|\mathcal{Z}\|_{\text{F}} = \sqrt{\langle \mathcal{Z}, \mathcal{Z} \rangle}$.

A point in the search space $\mathcal{M} = \mathbb{R}^{m_1 \times R} \times \cdots \mathbb{R}^{m_k \times R}$ of (7.1) is denoted by $U = (U^{(1)}, \ldots, U^{(k)})$, where $U^{(i)} \in \mathbb{R}^{m_i \times R}$ is the i-th *factor matrix* of U. In fact, $\mathcal{M}$ is a smooth manifold, and the tangent space to $\mathcal{M}$ at any point $U \in \mathcal{M}$ is $\mathcal{T}_U\mathcal{M} = \mathbb{R}^{m_1 \times R} \times \cdots \mathbb{R}^{m_k \times R}$. Therefore, a tangent vector in $\mathcal{T}_U\mathcal{M}$ is also a tuple of k factor matrices, denoted as $\xi = (\xi^{(1)}, \ldots, \xi^{(k)})$, where $\xi^{(i)} \in \mathbb{R}^{m_i \times R}$, for $i = 1, \ldots, k$. The Euclidean metric at U is defined and denoted as, for all $\xi, \eta \in \mathcal{T}_U\mathcal{M}$, $\langle \xi, \eta \rangle = \sum_{i=1}^{k} \operatorname{tr}({\xi^{(i)}}^T \eta^{(i)})$, where $\operatorname{tr}(\cdot)$ is the trace of a matrix.

Let $\mathcal{M}$ be endowed with a Riemannian metric g, then the Riemannian gradient of a real-valued smooth function f at $U \in \mathcal{M}$, denoted as $\operatorname{grad} f(U)$, is the unique element in $\mathcal{T}_U\mathcal{M}$ that satisfies, for all $\xi \in \mathcal{T}_U\mathcal{M}$, $g_U(\operatorname{grad} f(U), \xi) = \mathrm{D}f(U)[\xi]$, where $\mathrm{D}f(U)$ denotes the first-order differential of f at U.

Problem statement

By setting the regularizer as $\psi(U) = \frac{\lambda}{2} \sum_{i=1}^{k} \|U^{(i)}\|_{\text{F}}^2$, in a similar way as in Maximum-margin Matrix Factorization [SRJ05], we specify the polyadic decomposition-based model (7.1) as follows,

$$\underset{U \in \mathcal{M} = \mathbb{R}^{m_1 \times R} \times \cdots \mathbb{R}^{m_k \times R}}{\text{minimize}} \quad f(U) := f_\Omega(U) + \frac{\lambda}{2} \sum_{i=1}^{k} \|U^{(i)}\|_{\text{F}}^2, \tag{7.3}$$

where the first term $f_\Omega(U) := \frac{1}{2p} \|P_\Omega([\![U^{(1)}, \cdots, U^{(k)}]\!] - \mathcal{T}^\star)\|_{\text{F}}^2$ is referred to as the data fitting function of the problem, and $p = |\Omega|/(m_1 \cdots m_k)$ is a constant called the *sampling rate*. Note that when the regularization parameter $\lambda > 0$, the regularizer ψ has an effect of keeping the variable U in a compact subset of $\mathcal{M}$.

7.2 Algorithms

In this section, we propose a new metric on the manifold $\mathcal{M} = \mathbb{R}^{m_1 \times R} \times \cdots \mathbb{R}^{m_k \times R}$. Under the proposed metric, we develop Riemannian gradient descent and Riemannian conjugate gradient algorithms on $\mathcal{M}$.

7.2.1 A preconditioned metric

The Riemannian preconditioned algorithms in [MAS12, KM16] on the product space of full-column rank matrices were proposed to improve the Euclidean gradient descent method. In these algorithms, a Riemannian metric was defined on the search space according to the differential properties of the cost function. Specifically, it is designed based on an operator that approximates the "diagonal blocks" of the second-order differential of the cost function. The chosen metric plays a role of preconditioning in the optimization algorithms such as Riemannian gradient descent. We refer to [MS16] for a more general view on this topic of Riemannian preconditioning.

Starting from the idea of Riemannian preconditioning, we design a metric for the polyadic decomposition-based problem (7.1), in which the rank constraints are more relaxed than the fixed Tucker-rank constraint in [KM16]. First, we construct an operator $\mathcal{H}(U)$ using the "diagonal blocks" of the second-order differential of the data fitting function f_Ω in (7.3). Roughly speaking, such an operator satisfies

$$\langle \mathcal{H}(U)[\xi], \eta \rangle \approx \nabla^2 f_\Omega(U)[\xi, \eta], \tag{7.4}$$

for all $\xi, \eta \in \mathcal{T}_U\mathcal{M}$, where $\nabla^2 f_\Omega(U)$ denotes the second-order differential of f_Ω, and $\langle \cdot, \cdot \rangle$ is the Euclidean metric. Then, we design a metric that behaves locally like $(\xi, \eta) \mapsto \langle \mathcal{H}(U)[\xi], \eta \rangle$. Assume that the operator $\mathcal{H}(U)$ is invertible and that $\tilde{g} : (\xi, \eta) \mapsto \langle \mathcal{H}(U)[\xi], \eta \rangle$ forms a Riemannian metric on $\mathcal{M}$, then the Riemannian gradient $\mathrm{grad}[f_\Omega](U)$ satisfies, by definition, $\tilde{g}_U(\mathrm{grad}[f_\Omega](U), \xi) = \mathrm{D}f_\Omega(U)[\xi] = \langle \nabla f_\Omega(U), \xi \rangle$ for all $\xi \in \mathcal{T}_U\mathcal{M}$, where $\nabla f_\Omega(U)$ denotes the Euclidean gradient of f_Ω. Hence, $\mathrm{grad}[f_\Omega](U) = \mathcal{H}(U)^{-1}[\nabla f_\Omega(U)]$. In view of $\mathcal{H}(U)$ as in (7.4), $\mathrm{grad}[f_\Omega](U)$ is an approximation of the Newton direction of f_Ω, which results in a much improved convergence behavior than the Euclidean gradient descent.

The main problematic in designing such a metric using $\mathcal{H}(U)$ as in (7.4) is two-fold: (i) we need to find an appropriate approximation to the second-order differential $\nabla^2 f_\Omega(U)$ and (ii) the sought operator $\mathcal{H}(U) : \mathcal{T}_U\mathcal{M} \mapsto \mathcal{T}_U\mathcal{M}$ may not be invertible. Concretely, we address these two problematics as follows.

First, we deduce the second-order partial derivatives of f_Ω. Let U be a point in $\mathcal{M}$. The gradient of f_Ω at $U \in \mathcal{M}$ under the Euclidean metric is a vector in $\mathcal{T}_U\mathcal{M}$ as follows:

$$\nabla f_\Omega(U) = (\partial_{U^{(1)}} f_\Omega(U), \ldots, \partial_{U^{(k)}} f_\Omega(U)), \tag{7.5}$$

where the partial derivatives have the following element-wise expression, for $i = 1, \ldots, k$,

$$[\partial_{U^{(i)}} f_\Omega(U)]_{\ell,r} := \frac{\mathrm{d}}{\mathrm{d}t} f_\Omega(U^{(1)}, \ldots, U^{(i)} + tE_{\ell,r}, \ldots, U^{(k)})\Big|_{t=0}, \tag{7.6}$$

where $E_{\ell,r} = \mathbf{e}_\ell \mathbf{e}_r^T$. The right-hand side of (7.4) can be written in terms of the following second-order partial derivatives,

$$\nabla^2 f_\Omega(U)[\xi,\eta] := \sum_{i=1}^{k} \langle \partial_{ii}^2 f_\Omega(U)[\xi],\eta\rangle + \sum_{i=1,j'\neq i}^{k} \langle \partial_{ij'}^2 f_\Omega(U)[\xi],\eta\rangle\,,$$

where

$$\partial_{ij}^2 f_\Omega(U)[\xi] := \frac{\mathrm{d}}{\mathrm{d}t}\left(\partial_{U^{(i)}} f_\Omega(U^{(1)},\ldots,U^{(j)}+t\xi^{(j)},\ldots,U^{(k)})\right)\Big|_{t=0} \tag{7.7}$$

for $i,j=1,\ldots,k$. Subsequently, we construct $\mathcal{H}(U)$ as an operator on $\mathcal{T}_U\mathcal{M}$ based on the action of the aforementioned "diagonal blocks" of $\nabla^2 f_\Omega(U)$, i.e., $\partial_{ii}^2 f_\Omega(U)[\xi]$ for $i=1,\ldots,k$. More accurately, to design a metric that works for all tensor completion problems with a certain subsampling pattern Ω, we use the expectation of these terms over the subsampling operator. Therefore, we define $\mathcal{H}(U):\mathcal{T}_U\mathcal{M}\mapsto\mathcal{T}_U\mathcal{M}$ as follows,

$$\mathcal{H}(U)[\xi] = \left(\mathbb{E}_\Omega\left[\partial_{11}^2 f_\Omega(U)[\xi]\right],\ldots,\mathbb{E}_\Omega\left[\partial_{kk}^2 f_\Omega(U)[\xi]\right]\right). \tag{7.8}$$

Given the polyadic decomposition-based data fitting function f_Ω in (7.3), f_Ω can be rewritten in terms of $U^{(i)}$ and $(U^{(j)})^{\odot_{j\neq i}}$ through the mode-i tensor matricizations (2.24): $f_\Omega(U) = \frac{1}{2p}\left\|P_{\Omega_{(i)}}\left(U^{(i)}((U^{(j)})^{\odot_{j\neq i}})^T - \mathcal{T}^\star_{(i)}\right)\right\|_F^2$, where $\Omega_{(i)}$ is the mode-i matricization of Ω. Therefore, from (7.6), the first-order derivatives have the following expression,

$$\partial_{U^{(i)}} f_\Omega(U) = \frac{1}{p}\mathcal{S}_{(i)}(U^{(j)})^{\odot_{j\neq i}}, \tag{7.9}$$

where $\mathcal{S}_{(i)}$ is the mode-i matricization of the *residual tensor*

$$\mathcal{S} := P_\Omega\left(\llbracket U^{(1)},\cdots,U^{(k)}\rrbracket - \mathcal{T}^\star\right). \tag{7.10}$$

Combining (7.7) and (7.9), it follows that

$$\partial_{ii}^2 f_\Omega(U)[\xi] = \frac{1}{p}P_{\Omega_{(i)}}\left(\xi^{(i)}((U^{(j)})^{\odot_{j\neq i}})^T\right)(U^{(j)})^{\odot_{j\neq i}}. \tag{7.11}$$

Let Ω be a random index set of entries that are i.i.d. samples of the Bernoulli distribution: $(i_1,\ldots,i_k)\in\Omega$, with probability p, for $(i_1,\ldots,i_k)\in[\![m_1]\!]\times\cdots\times[\![m_k]\!]$. Then by taking the expectation over the index set Ω, (7.11) has the following approximation $\mathbb{E}_\Omega\left[\partial_{ii}^2 f_\Omega(U)[\xi]\right] = \xi^{(i)}((U^{(j)})^{\odot_{j\neq i}})^T(U^{(j)})^{\odot_{j\neq i}}$.

Therefore, the operator $\mathcal{H}(U)$ as defined in (7.8) reads

$$\mathcal{H}(U)[\xi] = \left(\xi^{(1)}(U^{(j)})^{\odot_{j\neq 1}})^T(U^{(j)})^{\odot_{j\neq 1}}, \ldots, \xi^{(k)}(U^{(j)})^{\odot_{j\neq k}})^T(U^{(j)})^{\odot_{j\neq k}}\right). \tag{7.12}$$

The second problematic is that the operator $\mathcal{H}(U)$ in (7.12) may not be invertible, since the R-by-R symmetric matrices in the form of $((U^{(j)})^{\odot_{j\neq i}})^T(U^{(j)})^{\odot_{j\neq i}}$ are not necessarily positive definite. To this end, we propose to regularize $\mathcal{H}(U)$ with the identity operator on $\mathcal{T}_U\mathcal{M}$. This is done by *shifting* the aforementioned matrices by adding a constant diagonal matrix δI_R, where I_R denotes the R-by-R identity matrix and $\delta > 0$ is a relatively small parameter. Consequently, we define the following inner product.

Definition 7.2.1. *Given $U = (U^{(1)}, \ldots, U^{(k)}) \in \mathcal{M}$, let g_U be an inner product in $\mathcal{T}_U\mathcal{M}$ as follows,*

$$g_U(\xi, \eta) = \sum_{i=1}^{k} \operatorname{tr}\left(\xi^{(i)} H_{i,U} (\eta^{(i)})^T\right), \text{ for } \xi, \eta \in \mathcal{T}_U\mathcal{M}, \tag{7.13}$$

where $H_{i,U}$ is a R-by-R matrix defined as

$$H_{i,U} = ((U^{(j)})^{\odot_{j\neq i}})^T(U^{(j)})^{\odot_{j\neq i}} + \delta I_R \tag{7.14}$$

and $\delta > 0$ is a constant parameter.

Note that $H_{i,U}$ is positive definite even when there is a rank-deficient factor matrix among $\{U^{(i)}\}$. Moreover, g is smooth on $\mathcal{M}$, and is therefore a Riemannian metric.

Now, we consider $\mathcal{M}$ as a manifold endowed with the Riemannian metric (7.13). The associated norm of a tangent vector $\xi \in T_U\mathcal{M}$ is defined and denoted by $\|\xi\|_U = \sqrt{g_U(\xi, \xi)}$. Based on the Euclidean gradient of f in (7.5), the Riemannian gradient of f is, by definition,

$$\operatorname{grad} f(U) = \left(\partial_{U^{(1)}} f(U) H_{1,U}^{-1}, \cdots, \partial_{U^{(k)}} f(U) H_{k,U}^{-1}\right). \tag{7.15}$$

As mentioned in the beginning of this subsection, the Riemannian gradient (7.15) can be seen as the result of preconditioning of the Euclidean gradient $\nabla f(U)$ with the operator $\mathcal{H}(U)$ (upto a "rescaling" with δI_R). We refer to the new metric (7.13) as the *preconditioned metric* on $\mathcal{M}$.

7.2.2 The Riemannian preconditioned algorithms

With the gradient defined in (7.15) using Riemannian preconditioning, we adapt Riemannian gradient descent and Riemannian conjugate gradient algorithms (e.g., [AMS08]) to solve the problem (7.3).

The Riemannian gradient algorithm is given in Algorithm 7.2.1, which consists mainly of setting the descent direction as the negative Riemannian gra-

Algorithm 7.2.1 Riemannian Gradient Descent (RGD)

Input: $f : \mathcal{M} \mapsto \mathbb{R}$, $x_0 \in \mathcal{M}$, tolerance $\epsilon > 0$; $t = 0$.
Output: x_t.
1: **while** $\|\text{grad} f(x_t)\| > \epsilon$ **do**
2: Set $\eta_t = -\text{grad} f(x_t)$. # See (7.15)
3: Set stepsize s_t through one of the rules (7.16), (7.17) or (7.18).
4: Update: $x_{t+1} = x_t + s_t \eta_t$; $t \leftarrow t + 1$.
5: **end while**

dient and selecting the stepsizes. Note that since the search space is $\mathcal{M} = \mathbb{R}^{m_1 \times R} \times \cdots \mathbb{R}^{m_k \times R}$, the retraction map in this algorithm (line 4) is chosen as the identity map. In line 3, given an iterate $x_t \in \mathcal{M}$ and $\eta_t \in \mathcal{T}_{x_t}\mathcal{M}$, the stepsize s_t is chosen by one of the following three methods.

Stepsize by line minimization. The line minimization consists in computing a stepsize as follows,

$$s_t = \arg\min_{s>0} h(s) := f(x_t + s\eta_t). \tag{7.16}$$

With third-order tensors ($k = 3$), the solution can be obtained numerically by selecting from the roots of the derivative $h'(s)$, which is a polynomial of degree 5.

Backtracking line search with the Armijo condition. We first set up a trial stepsize s_t^0 using the classical strategy [NW06, §3.4]: (i) when $t \leq 1$, $s_t^0 = 1$, (ii) when $t \geq 2$, $s_t^0 = 2(f(x_{t-1}) - f(x_{t-2}))/g_{x_{t-1}}(\eta_{t-1}, \text{grad} f(x_{t-1}))$; then the stepsize s_t is returned by a backtracking procedure, i.e., finding the smallest integer $\ell \geq 0$ such that

$$f(x_t) - f\left(x_t + s_t\eta_t\right) \geq \sigma s_t g_{x_t}(-\text{grad} f(x_t), \eta_t), \tag{7.17}$$

for $s_t := \max(s_t^0 \beta^\ell, s_{\min})$ with a constant parameter $s_{\min} > 0$. The backtracking parameters are fixed with $\sigma, \beta \in (0, 1)$.

The Barzilai–Borwein (BB) stepsize. Recently, the Riemannian Barzilai–Borwein (RBB) stepsize [IP18] has proven to be an efficient stepsize rule for Riemannian gradient methods. Hence, we choose the stepsize as

$$s_t^{\text{RBB1}} := \frac{\|z_{t-1}\|_{x_t}^2}{|g_{x_t}(z_{t-1}, y_{t-1})|}, \quad \text{or} \quad s_t^{\text{RBB2}} := \frac{|g_{x_t}(z_{t-1}, y_{t-1})|}{\|y_{t-1}\|_{x_t}^2}, \tag{7.18}$$

where $z_{t-1} = x_t - x_{t-1}$ and $y_{t-1} = \text{grad} f(x_t) - \text{grad} f(x_{t-1})$.

The Riemannian conjugate gradient algorithm is similar to RGD (Algorithm 7.2.1) in terms of the stepsize selection (line 3) and the update step (line

4). The RCG algorithm differs with RGD in the choice of the search direction (line 2). More specifically, the search direction of RCG is defined as

$$\eta_t = -\mathrm{grad} f(x_t) + \beta_t \eta_{t-1},$$

where β_t is the CG parameter. In the numerical experiments, we choose the Riemannian version [BMAS14] of the modified Hestenes–Stiefel rule [HS52] (HS+) as follows,

$$\beta_t = \max\left(0, \frac{g_{x_t}(\xi_t - \xi_{t-1}, \xi_t)}{g_{x^t}(\xi_t - \xi_{t-1}, \eta_{t-1})}\right).$$

The vector transport operation involved in the computation of β_t is chosen to be the identity map.

In both algorithms, the cost for calculating the Riemannian gradient (7.15) is a dominant term of the total cost. Therefore, we propose an efficient method for evaluating the Riemannian gradient in the next section.

7.2.3 Computation of the gradient

We focus on the computation of the Riemannian gradient of f in (7.3). From the definition (7.15), the computation of $\mathrm{grad} f(U) = (\eta^{(1)}, \ldots, \eta^{(k)})$ consists of two parts: (i) computing the partial derivatives $D_i := \partial_{U^{(i)}} f(U) \in \mathbb{R}^{m_i \times R}$ and (ii) computing the matrix multiplications $\eta^{(i)} = D_i H_{i,U}^{-1}$. From the expressions (7.9) and (7.14), we have

$$D_i = \frac{1}{p} \underbrace{\mathcal{S}_{(i)}(U^{(j)})^{\odot_{j\neq i}}}_{\tilde{D}_i} + \lambda U^{(i)}, \tag{7.19}$$

$$\eta^{(i)} = D_i(\underbrace{((U^{(j)})^{\odot_{j\neq i}})^T (U^{(j)})^{\odot_{j\neq i}} + \delta I_R}_{H_{i,U}})^{-1}. \tag{7.20}$$

In a straightforward manner, these two parts require mainly the following operations:

1. Computing the sparse tensor $\mathcal{S}$ as in (7.10), which requires $2|\Omega|R$ flops;
2. Computing $(U^{(j)})^{\odot_{j\neq i}}$ for $i = 1, \ldots, k$, which requires $2\sum_{i=1}^{k} m_{-i} R$ flops, where $m_{-i} := \prod_{j\neq i} m_j$;
3. Forming the sparse matricizations $\mathcal{S}_{(i)}$ for $i = 1, \ldots, k$, which takes some extra time for input/output with the sparse tensor $\mathcal{S}$ and the matricizations of the index set Ω.
4. Computing the sparse-dense matrix products $\tilde{D}_i := \mathcal{S}_{(i)}(U^{(j)})^{\odot_{j\neq i}}$, for $i = 1, \ldots, k$, which require $2k|\Omega|R$ flops. Then, one has access to $\{D_i\}_{i=1,\ldots,k}$ after the matrix additions with $\lambda U^{(i)}$ (whose cost is not listed out since it is fixed regardless of the computational method).

5. Computing the R-by-R matrix $H_{i,U}$ (7.14) based on the matrix $(U^{(j)})^{\odot_{j\neq i}}$ (obtained in step 2), which consists of a dense-dense matrix multiplication of sizes $R \times m_{-i}$ and $m_{-i} \times R$, for $i = 1, \ldots, k$, which mainly requires $2\sum_{i=1}^{k} m_{-i}R^2$.

6. Computing $D_i H_{i,U}^{-1}$ given D_i (obtained in step 4) and $H_{i,U}$ (obtained in step 5), through Cholesky decomposition of $H_{i,U}$, for $i = 1, \ldots, k$, which requires $\sum_{i=1}^{k} 2m_i R^2 + C_{\text{chol}} R^3$.

The sum of the flops counted in the above list of operations is

$$2(k+1)|\Omega|R + (\sum_{i=1}^{k} 2m_{-i}(R^2+R)) + (\sum_{i=1}^{k} 2m_i R^2 + C_{\text{chol}} R^3). \tag{7.21}$$

An efficient computational method. We propose a computational method that avoids the matricizations of the residual tensor $\mathcal{S}$ and the computations of the Khatri–Rao products $(U^{(j)})^{\odot_{j\neq i}}$.

Given the residual tensor $\mathcal{S}$ after the step 1 above, we propose to compute $\tilde{D}_i$ in (7.19) without passing through the steps 2 and 3. In fact, the computation of $\tilde{D}_i = \mathcal{S}_{(i)}(U^{(j)})^{\odot_{j\neq i}}$ corresponds to the Matricized tensor times Khatri–Rao product (MTTKRP), which is a common routine in the tensor computations. Through basic tensor computations, the entrywise expression of this MTTKRP does not require forming the matricizations of $\mathcal{S}$ explicitly. For brevity, we demonstrate these relations concretely in the case of third-order tensors ($k = 3$), knowing that their extension to higher-order tensors is straightforward. The matrix $\tilde{D}_1 = \mathcal{S}_{(1)}(U^{(j)})^{\odot_{j\neq 1}} \in \mathbb{R}^{m_1 \times R}$ in (7.19) has the entrywise expression below,

$$\left[\tilde{D}_1\right]_{i_1 \ell} = \sum_{i_2=1}^{m_2} \sum_{i_3=1}^{m_3} \mathcal{S}_{i_1 i_2 i_3} U^{(3)}_{i_3 \ell} U^{(2)}_{i_2 \ell}, \tag{7.22}$$

for $(i_1, i_2, i_3) \in [\![m_1]\!] \times \cdots \times [\![m_3]\!]$ and $\ell = 1, \ldots, R$. Based on (7.22), Algorithm 7.2.2 presents an efficient way to compute the MTTKRPs of (7.19), with a sparse residual tensor $\mathcal{S}$ as input. Note that the computations of $\tilde{D}_2$ and $\tilde{D}_3$ correspond to the same equation (7.22) but with the indices (i_1, i_2, i_3) swapped via the rotations $(1, 2, 3; 2, 3, 1)$ and $(1, 2, 3; 3, 1, 2)$ respectively.

Subsequently, we propose to compute $H_{i,U}$ (7.14) without large matrix multiplications. In fact, the large matrix multiplications with $(U^{(j)})^{\odot_{j\neq i}}$ in (7.14) can be decomposed into smaller ones. Note that these matrix multiplications satisfy the following identity:

$$\left((U^{(j)})^{\odot_{j\neq i}}\right)^T \left((U^{(j)})^{\odot_{j\neq i}}\right) = G_k \star \cdots G_{i+1} \star G_{i-1} \star \cdots \star G_1, \tag{7.23}$$

where $G_j := {U^{(j)}}^T U^{(j)}$ for $j \neq i$ and the product by $\star$ denotes the Hadamard product. Using this property, the computation of $H_{i,U}$ reduces to computing

Algorithm 7.2.2 Sparse MTTKRP

Input: The index sets by axis $I_\Omega := \{i : (i,j,k) \in \Omega\}$, J_Ω and K_Ω. The sparse tensor $\mathcal{S}$ in the form of a $|\Omega|$-by-1 array of observed entries $\{S_p = \mathcal{S}_{i_p,j_p,k_p} : (i_p,j_p,k_p) \in \Omega\}$. The factor matrices $(U^{(i)})_{i=1,2,3}$.

Output: $\tilde{D} := \mathcal{S}_{(1)}(U^{(3)} \odot U^{(2)})$.

```
1: D̃ = 0.
2: for p = 1, ..., |Ω| do
3:    for ℓ = 1, ..., R do
4:       D̃_{i_p ℓ} = D̃_{i_p ℓ} + S_p U^(3)_{k_p ℓ} U^(2)_{j_p ℓ}.
5:    end for
6: end for
```

$G_j = {U^{(j)}}^T U^{(j)}$, and then the entrywise multiplications between the (small) R-by-R matrices $\{G_j\}$, which require only $\sum_{i=1}^{k}(2m_i + k - 1)R^2$ flops, since the computation of the matrices G_j cost $2\sum_{i=1}^{k} m_i R^2$ and the entrywise multiplications between G_j cost $(k-1)R^2$.

In summary, the operations reduce to the following steps.

a. Computing the sparse tensor $\mathcal{S}$ as in (7.10). This is identical to the step 1 above, which requires $2|\Omega|R$ flops;

b. Computing $\tilde{D}_i := \mathcal{S}_{(i)}(U^{(j)})^{\odot_{j\neq i}}$, for $i = 1, \ldots, k$, using $\mathcal{S}$ (obtained in step a) and U; see Algorithm 7.2.2. The computational cost of this step is $2k|\Omega|R$. Then, one has access to $\{D_i\}_{i=1,\ldots,k}$ after the matrix additions with $\lambda U^{(i)}$.

c. Computing the R-by-R matrix $H_{i,U}$ (7.14) using U (input data); see (7.23). The computational cost of this step is $\sum_{i=1}^{k}(2m_i + k - 1)R^2$.

d. Computing $D_i H_{i,U}^{-1}$ given D_i (obtained in step b) and $H_{i,U}$ (obtained in step c), through Cholesky decomposition of $H_{i,U}$, for $i = 1, \ldots, k$. This is identical to the step 6 above, which requires $\sum_{i=1}^{k} 2m_i R^2 + C_{\text{chol}} R^3$ flops.

Therefore, the total cost of the above steps is

$$2(k+1)|\Omega|R + \sum_{i=1}^{k}(4m_i + k - 1)R^2 + C_{\text{chol}}R^3,$$

which is significantly reduced compared to the cost (7.21) of the naive method. In particular, for third-order (or a bit higher-order) tensors ($k \ll m_i$) with a low rank parameter R, the dominant term in (7.21) is $2(k+1)|\Omega|R + \sum_{i=1}^{k}(2m_{-i} + 2m_i)R^2$, while the dominant term in the cost of the proposed method is $2(k+1)|\Omega|R + \sum_{i=1}^{k} 4m_i R^2$. The reduction in the cost can be seen from the fact

that $m_i \ll m_{-i} = \prod_{j \neq i} m_j$ and $m_i \ll |\Omega| = pm_1 \dots m_k$. Note that on top of the above reduction in flops, the time efficiency is further improved as the matricizations of the residual tensor $\mathcal{S}$ are not needed. In particular, speedups related to the step b (instead of steps 2–4 in the naive method) are demonstrated in Table 7.A.1.

7.3 Convergence analysis

In this section, we analyze the convergence behavior of Algorithm 7.2.1. Let $\{x_t\}_{t\geq 0}$ denote the sequence generated by this algorithm. First, we demonstrate in Proposition 7.3.3 that every accumulation point of $\{x_t\}_{t\geq 0}$ is a stationary point. Second, we analyze the iterate convergence property of the algorithm in Theorem 7.3.5.

The following lemma generalizes the class of functions with Lipschitz-continuous gradient to functions defined on Riemannian manifolds, and will be used in Lemma 7.3.2.

Lemma 7.3.1 ([BAC19, Lemma 2.7])**.** *Let $\mathcal{M}' \subset \mathcal{M}$ be a compact Riemannian submanifold. Let $\mathcal{R}_x : \mathcal{T}_x\mathcal{M}' \mapsto \mathcal{M}'$. If $f : \mathcal{M}' \mapsto \mathbb{R}$ has Lipschitz continuous gradient in the convex hull of $\mathcal{S}$. Then there exists $L > 0$ such that, for all $x \in \mathcal{M}'$ and $\xi \in \mathcal{T}_x\mathcal{M}'$,*

$$|f(\mathcal{R}_x(\xi)) - (f(x) + g_x(\xi, gradf(x))| \leq \frac{L}{2}\|\xi\|_x^2. \tag{7.24}$$

Starting from the above lemma, we show that the proposed algorithm ensures a sufficient decrease property at each iteration.

Lemma 7.3.2. *Let $\{x_t\}_{t\geq 0}$ be the sequence generated by Algorithm 7.2.1 (RGD), in which the step sizes are chosen by either line minimization* (7.16) *or Armijo line search* (7.17)*. For all $t \geq 0$, there exists $\bar{C} > 0$ such that*

$$f(x_t) - f(x_{t+1}) \geq \bar{C}\|gradf(x_t)\|_{x_t}^2. \tag{7.25}$$

In particular, with the step sizes chosen by line minimization (7.16)*, there exists a Lipschitz-like constant $L_0 > 0$ such that* (7.25) *holds for $\bar{C} = \frac{1}{2L_0}$.*

Proof. Due to the fact that the objective function is coercive (because of the Frobenius norm-based terms), the sublevel set $\mathcal{S} = \{x \in \mathcal{M} : f(x) \leq f(x_0)\}$ is a closed and bounded subset of $\mathcal{M}$. Due to the boundedness of $\mathcal{S}$, the convex hull of $\mathcal{S}$, denoted as $\bar{\mathcal{S}}$, is bounded. Therefore, f has Lipschitz continuous gradient in $\bar{\mathcal{S}}$ since $f \in C^2(\mathcal{M})$. From Lemma 7.3.1, there exists a Lipschitz-like constant $L_0 > 0$ such that (7.24) holds. The inequality (7.24) ensures an upper bound of $f(\mathcal{R}_x(s\xi)) = f(x+s\xi)$ as follows, $f(x+s\xi) \leq f(x)+g_x(s\xi, \mathrm{grad} f(x))+\frac{L_0}{2}\|s\xi\|_x^2$, for all $s \geq 0$. Consequently, when the stepsize $s_t = s^*$ is selected by line

minimization (7.16), we have

$$f(x_{t+1}) = f(x_t - s^* \mathrm{grad} f(x_t)) \\ \leq \min_{s\geq 0} \left(f(x) - s(1 - \frac{L_0 s}{2}) \|\mathrm{grad} f(x_t)\|_{x_t}^2 \right) = f(x_t) - \bar{C} \|\mathrm{grad} f(x_t)\|_{x_t}^2,$$

with $\bar{C} = \frac{1}{2L_0}$. When the stepsize s_t is selected using Armijo line search, the new iterate $x_{t+1} = x_t - s_t \mathrm{grad} f(x_t)$ is an Armijo point, where $s_t \geq s_{\min} > 0$, by construction of the line search procedure (with the parameter of lower bound of stepsizes $s_{\min}$). Hence, through (7.17), we have

$$f(x_t) - f(x_{t+1}) \geq \sigma s_t \|\mathrm{grad} f(x_t)\|_{x_t}^2 \geq \bar{C} \|\mathrm{grad} f(x_t)\|_{x_t}^2,$$

where $\bar{C} = \sigma s_{\min} > 0$, with the line search parameter $\sigma \in (0, 1)$.

In conclusion, the sufficient decrease property is satisfied with the two stepsize selection methods in the statement. □

Proposition 7.3.3. *The sequence $\{x_t\}_{t\geq 0}$ generated by Algorithm 7.2.1, with step sizes chosen by either line minimization* (7.16) *or Armijo line search* (7.17)*, satisfies the following convergence properties: (i) Every accumulation point is a stationary point; (ii) The algorithm needs at most $\left\lceil \frac{(f^* - f(x_0))}{\bar{C}} \frac{1}{\epsilon^2} \right\rceil$ iterations to reach an ϵ-stationary solution, for a constant $\bar{C} > 0$.*

Proof. (i) Let $x_* \in \mathcal{M}$ be an accumulation point, then there exists a subsequence $(x_{k(t)})_{t\geq 0}$, where $\{k(t) : t \geq 0\} \subset \mathbb{N}$, such that $\lim_{t\to\infty}(f(x_{k(t)}) - f(x_*)) = 0$. This entails that $\sum_{t=0}^{\infty} f(x_{k(t)}) - f(x_{k(t+1)}) = f(x_{k(0)}) - f(x_*) < \infty$. Applying the sufficient decrease property (7.25) of Lemma 7.3.2 to this previous inequality, we have

$$\sum_{t=0}^{\infty} \bar{C} \|\mathrm{grad} f(x_{k(t)})\|_{x_{k(t)}}^2 \leq \sum_{t=0}^{\infty} f(x_{k(t)}) - f(x_{k(t+1)}) < \infty, \tag{7.26}$$

for a constant $\bar{C} > 0$. Therefore, $\lim_{t\to\infty} \|\mathrm{grad} f(x_{k(t)})\|_{x_{k(t)}} = 0$. (ii) Suppose that the algorithm does not attain an ϵ-stationary point (a point on which the gradient norm is bounded by ϵ) at iteration $T-1$, then $\|\mathrm{grad} f(x_t)\| > \epsilon$, for all $0 \leq t \leq T-1$. Using (7.25), we have $f(x_0) - f(x_T) \geq \bar{C} \sum_{t=0}^{T-1} \|\mathrm{grad} f(x_t)\|_{x_t}^2 \geq \bar{C}\epsilon^2 T$. Therefore $T \leq \frac{f(x_0) - f(x_*)}{\bar{C}} \frac{1}{\epsilon^2}$. □

Next, we prove the iterate convergence of the RGD algorithm in Theorem 7.3.5 using the Łojasiewicz property. We first give the definition of the Łojasiewicz inequality for functions defined on a Riemannian manifold [SU15].

Definition 7.3.4 (Łojasiewicz inequality [SU15, Definition 2.1])**.** *Let $\mathcal{M} \subset \mathbb{R}^n$ be a Riemannian submanifold of $\mathbb{R}^n$. The function $f : \mathcal{M} \mapsto \mathbb{R}$ satisfies a*

Łojasiewicz gradient inequality at a point $x \in \mathcal{M}$, if there exists $\delta > 0$, $\sigma > 0$ and $\theta \in (0, 1/2]$ such that for all $y \in \mathcal{M}$ with $\|y - x\| \leq \delta$, it holds that

$$|f(x) - f(y)|^{1-\theta} \leq \sigma \|gradf(y)\|, \tag{7.27}$$

where θ is called the Łojasiewicz exponent.

The Proposition 2.2 of [SU15] guarantees that (7.27) is satisfied for real analytic functions defined on an analytic manifold. Since the objective function of (7.3) is indeed real analytic and that the search space $\mathcal{M}$ is an analytic manifold, the Łojasiewicz inequality (7.27) holds. Consequently, we have the following iterate convergence result.

Theorem 7.3.5. *Let $\{x_t\}_{t\geq 0}$ be the sequence generated by Algorithm 7.2.1 with stepsizes chosen by either line minimization* (7.16) *or Armijo line search* (7.17)*. Then $\{x_t\}_{t\geq 0}$ converges to a stationary point $x_* \in \mathcal{M}$. Moreover, the local convergence rate of $\{x_t\}_{t\geq 0}$ follows:*

$$\|x_t - x_*\| \leq C \begin{cases} e^{-ct} & \text{if } \theta = 1/2, \\ t^{-\theta/(1-2\theta)} & \text{otherwise}, \end{cases}$$

with the Łojasiewicz exponent $\theta \in (0, 1/2]$ and constants $c > 0$ and $C > 0$.

Proof. The inequality (7.25) ensures that the sequence $\{x_t\}_{t\geq 0}$ is monotonically decreasing. Hence, the RGD algorithm satisfies the conditions in [SU15, Theorem 2.3]. More precisely, it follows from (7.25) of Lemma 7.3.2 that

$$\begin{aligned} |f(x_{t+1}) - f(x_t)| &\geq \bar{C}\|\mathrm{grad} f(x_t)\|_{x_t}^2 = (\bar{C}/s_t)\|x_{t+1} - x_t\|_{x_t}\|\mathrm{grad} f(x_t)\|_{x_t} \\ &\geq \kappa_0 \|x_{t+1} - x_t\|_{x_t}\|\mathrm{grad} f(x_t)\|_{x_t}, \end{aligned} \tag{7.28}$$

where $\kappa_0 > 0$, since with line minimization (7.16), $s_t = s^* > 0$, for all $t \geq 0$, is chosen from a finite number of numerical solutions; and with Armijo line search (7.17), $0 < s_{\min} \leq s_t \leq s_t^0$, where $s_t^0 > 0$ is the initial stepsize before backtracking. In addition, the RGD update rule ensures that

$$\|x_{t+1} - x_t\|_{x_t} = s_t \|\mathrm{grad} f(x_t)\|_{x_t} \geq \kappa \|\mathrm{grad} f(x_t)\|_{x_t}, \tag{7.29}$$

for $\kappa > 0$. The result of the theorem is obtained by combining (7.28), (7.29) and the Łojasiewicz inequality (7.27) and using [SU15, Theorem 2.3]. □

Note that although the convergence properties of the RGD algorithm (Algorithm 7.2.1) with the RBB stepsize (7.18) are not given, one can easily use this RBB stepsize as the initial trial stepsize in a backtracking line search procedure. Note that the convergence results in this section can be proved for the RGD algorithm using line search with RBB (7.18) as the initial stepsize. Interested readers are referred to [IP18] for details.

7.4 Experiments

In this section, we carry out numerical experiments for tensor completion using the proposed algorithms and several existing algorithms in the related work. Details of these algorithms are as follows.

The proposed algorithms: Algorithm 7.2.1 is labeled as Precon RGD and the RCG algorithm is labeled as Precon RCG. Depending on the stepsize selection method, these algorithms are labeled with a descriptor in terms of (i) (linemin) for the stepsize rule (7.16), (ii) (Armijo) for the Armijo line search method (7.17), and (iii) (RBB) for the Barzilai–Borwein stepsize (7.18).

Euclidean gradient descent (Euclidean GD) and nonlinear conjugate gradient (Euclidean CG) algorithms refer to the algorithms using the Euclidean gradient (7.5) in the definition of the search directions on $\mathcal{M}$. The stepsize selection rules are the same as the proposed algorithms; these algorithms are implemented along with the proposed algorithms in the source code. INDAFAC is a damped Gauss-Newton method for CPD-based tensor completion proposed by Tomasi and Bro [TB05]. CP-WOPT [ADKM11] is a nonlinear conjugate gradient algorithm for CPD-based tensor completion. AltMin [GDAG20] is an alternating minimization algorithm for CPD-based tensor completion, which uses the linear CG for each of the least squares subproblems.

KM16 refers to a Riemannian optimization algorithm proposed by Kasai and Mishra [KM16] for tensor completion with a fixed Tucker rank. The Riemannian gradient in this algorithm is defined under a metric selected through Riemannian preconditioning on the manifold corresponding to a (fixed-rank) Tucker decomposition. In this algorithm, the tensor candidate is represented by a tuple of factor matrices and a core tensor via the Tucker decomposition. In our experiments on third-order tensors, this algorithm is labeled as KM16 (r_1, r_2, r_3), according to the Tucker rank (r_1, r_2, r_3) with which it is tested. Note that the dimension of the search space of KM16 is $\sum_{i=1}^{k} \left(m_i r_i - r_i^2\right) + \prod_{i=1}^{k} r_i$, which is different than the dimension of $\mathcal{M}$ (search space of the CPD/PD-based algorithms); In particular, the difference in these dimensions is marginal when $r_i \approx R$ for $i = 1, 2, 3$, with $R \ll \min(m_1, \ldots, m_k)$.

All the CPD/PD-based algorithms are initialized with a same randomly generated point on $\mathcal{M}$ and the Tucker decomposition-based algorithm (KM16) is initialized with a point such that its tensor representation is close enough to that of the initial point of the other algorithms; see Appendix 7.B for details.

All numerical experiments were performed on a workstation with 8-core Intel Core i7-4790 CPUs and 32GB of memory running Ubuntu 16.04 and MATLAB R2019. The source code is available at `https://gitlab.com/shuyudong.x11/tcprecon/`. Implementations of the existing algorithms are also publicly available.

7.4.1 Synthetic data

Tensor model. We consider a low-rank tensor model that is composed of a low Tucker-rank tensor and independent additive noises: with a given Tucker rank parameter $r^\star = (r_1^\star, r_2^\star, r_3^\star)$, we generate such a tensor $\mathcal{T}^\star$ using the following procedure,

$$\mathcal{T}^\star = \mathbb{T}_{r^\star}(\mathcal{T}) + \mathcal{E}, \tag{7.30}$$

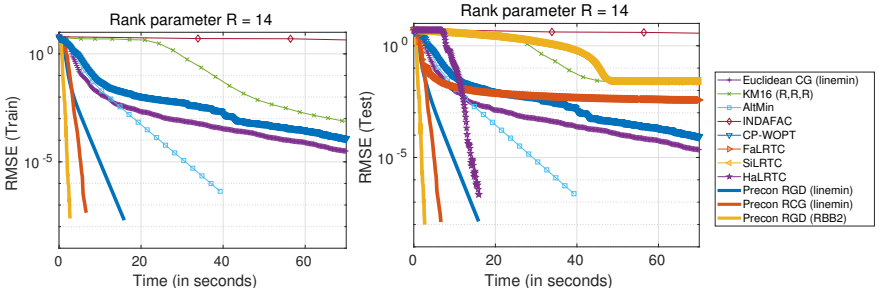
where $T \in \mathbb{R}^{m_1 \times m_2 \times m_3}$ is a third-order tensor composed of i.i.d. Gaussian entries, that is, $T_{ijk} \sim \mathcal{N}(0,1)$, the operator in the form of $\mathbb{T}_r(\cdot)$ is a Tucker-rank (rank_{tc}) truncation operator defined as the best Tucker rank-r approximation of $\mathcal{T}$. The truncation $\mathbb{T}_r(\cdot)$ can be obtained using existing implementations that are available in state-of-the-art tensor toolboxes (e.g., Tensor Toolbox [BK+19] and Tensorlab [VDS+16]). Here we use the function `tucker_als.m` in the MATLAB Tensor Toolbox. In the scenario of noiseless observations, $\mathcal{E} = 0$; otherwise $\mathcal{E} \in \mathbb{R}^{m_1 \times m_2 \times m_3}$ contains independent noises such that $\mathcal{E}_{\ell_1 \ell_2 \ell_3} \sim \mathcal{N}(0, \sigma)$, where σ is set according to a given signal-to-noise ratio (SNR); see Appendix 7.B.

Low-rank tensor recovery from partial, noiseless observations. A synthetic tensor $\mathcal{T}^\star$ is generated with the model (7.30) without noise. The tensor $\mathcal{T}^\star$ is only observed on an index set Ω, which is composed of indices drawn from the Bernoulli distribution: $(i,j,k) \in \Omega$ with probability $p \in (0,1)$, for all $(i,j,k) \in [\![m_1]\!] \times [\![m_2]\!] \times [\![m_3]\!]$.

For the problem model (7.3), we set the regularization parameter λ to zero, which allows for recovering the low-rank tensor $\mathcal{T}^\star$ without any bias, provided that the sampling rate p is sufficient. Then we test the aforementioned algorithms with a given rank parameter R, assuming that the rank (CP or Tucker rank) of the hidden tensor $\mathcal{T}^\star$ is unknown to all the algorithms. For the CPD and PD-based algorithms, we set the rank parameter R to an arbitrary value such that $R \geq \max(r_1^\star, r_2^\star, r_3^\star)$. Since the optimal CP rank of the tensor candidate is unknown, a larger-than-expected rank parameter is interesting because it allows for searching solutions in a fairly large tensor space, so that there is better chance that optimal solutions are in the search space $\mathcal{M}$.

The termination of the proposed algorithms (Precon RGD and Precon RCG) is controlled by a tolerance parameter ($\epsilon = 10^{-7}$ in this experiment) against the norm of the gradient; KM16 uses a Riemannian CG algorithm with Armijo line search and default stopping criteria. On top of their respective stopping criteria, all the algorithms are tested within a heuristic iteration budget maxiter = 1000. Note that for all tested algorithms except AltMin, one iteration corresponds to one pass over the whole training data $P_\Omega(\mathcal{T}^\star)$; for AltMin, one iteration corresponds to multiple passes over the training data, since each of its iteration has a number of inner iterations for solving the underlying alternating subproblem. Table 7.1 shows the performances of the tested algorithms in terms of recovery errors and time, for rank parameters $R \in \{7, 14, 21\}$. The iteration histories of these algorithms (including KM16) with $R = 14$ are

shown in Figure 7.1. Specifically, for the fixed-Tucker rank algorithm, KM16, we also test several other rank parameters than $r = (R, R, R)$; its tensor recovery performances along with those of the proposed algorithms are presented in Table 7.2. In Table 7.2, "#variables" indicates the dimension of the search space of each of algorithms, depending on the rank parameters.



(a) Training RMSEs (b) Test RMSEs

Figure 7.1: Tensor completion from noiseless observations. The size of $\mathcal{T}^\star$ is $(100, 100, 100)$ with a Tucker rank $r^\star = (3, 5, 7)$. The sampling rate is 0.3. The rank parameter is set as $R = 14$.

From the results shown in Figure 7.1 and Tables 7.1–7.2, we have the following observations: (i) For all three values of the rank parameter R that are larger than $\max(r_1^\star, r_2^\star, r_3^\star)$, the proposed algorithms and HaLRTC succeeded in recovering exactly the true hidden tensor $\mathcal{T}^\star$ (with a test RMSE lower than 10^{-6}). AltMin, CP-WOPT and Euclidean CG succeeded exact recoveries only with one or two of the rank parameter choices, and their convergences are slower than the proposed algorithms by orders of magnitude. The test error of KM16 stagnated at a certain level as the core tensor dimensions chosen are not exactly the same as the Tucker rank of $\mathcal{T}^\star$; (ii) Among the algorithms that successfully recovered the true hidden tensor, the proposed algorithms (Precon RGD and Precon RCG) outperform AltMin in time with a speedup of around 10 times, and they achieve speedups between 2 and 6 times compared to HaLRTC; Especially, Precon RGD (RBB2) has the fastest convergence behavior. (iii) For KM16 specifically, the time efficiency and the recovery performance of KM16 (r, r, r) improves significantly when the core tensor dimensions (r, r, r) decrease (and get closer to $r^\star$). In particular, when r is only $1/2$ of the rank parameter $R = 14$, the time efficiency of KM16 (r, r, r) gets close to those of the proposed algorithms. Note that when $r \approx R/2$, the dimensions of the search space of KM16 (r, r, r) is much smaller than that of the proposed algorithms; see Table 7.2. These comparisons can be explained by the fact that the per-iteration cost of KM16 is much larger than the proposed algorithms, even when the dimensions of its search space is close or smaller than that of the proposed algorithms; see Figure 7.2 for detailed comparisons of their average per-iteration time. The high computational cost of KM16 lies in its computation of the Riemannian gradient, which scales poorly with the Tucker rank as it

involves computing the Gram matrix of the matricizations of the core tensor—with a cost of $O(r_1 r_2 r_3 (r_1 + r_2 + r_3))$—and solving Lyapunov equations (for $k = 3$) of the $r_i \times r_i$ matrices.

Table 7.1: Tensor completion with noiseless observations. The size of $\mathcal{T}^\star$ is $(100, 100, 200)$ with a Tucker rank $r^\star = (3, 5, 7)$. The sampling rate is 0.3. The rank parameters R tested are $\{12, 14, 16\}$.

Algorithm	R	iter	time (s)	RMSE (test)	RMSE (train)
Euclidean CG (linemin)	12	592	100.03	1.57e-12	2.03e-12
AltMin	12	67	100.97	8.05e-06	8.09e-06
INDAFAC	12	7	129.91	3.97e-01	4.57e-01
CP-WOPT	12	405	29.96	5.74e-07	6.70e-07
HaLRTC	12	142	15.81	2.19e-07	–
Precon RGD (linemin)	12	96	16.25	3.84e-08	4.79e-08
Precon RCG (linemin)	12	48	8.23	1.96e-08	3.58e-08
Precon RGD (RBB2)	12	65	**3.91**	9.52e-09	4.74e-07
Euclidean CG (linemin)	14	518	100.11	2.20e-06	3.10e-06
AltMin	14	19	39.27	2.39e-07	4.35e-07
INDAFAC	14	5	125.77	1.35e+00	2.60e+00
CP-WOPT	14	1561	94.29	2.40e-06	3.35e-06
HaLRTC	14	142	15.97	2.19e-07	–
Precon RGD (linemin)	14	82	15.90	1.38e-08	1.99e-08
Precon RCG (linemin)	14	34	6.65	1.29e-08	4.39e-08
Precon RGD (RBB2)	14	39	**2.69**	9.84e-09	2.38e-08
Euclidean CG (linemin)	16	456	100.01	1.37e-07	1.53e-07
AltMin	16	8	31.79	2.67e-08	3.32e-08
INDAFAC	16	5	154.34	9.78e-01	1.19e+00
CP-WOPT	16	1766	99.02	5.23e-06	7.00e-06
HaLRTC	16	142	16.34	2.19e-07	–
Precon RGD (linemin)	16	40	8.82	1.56e-09	2.51e-09
Precon RCG (linemin)	16	50	11.09	2.59e-09	5.08e-09
Precon RGD (RBB2)	16	39	**3.03**	1.53e-10	3.14e-10

Low-rank tensor recovery from partial, noisy observations. In the following experiments, we conduct tensor completion tests under the same settings as in the previous experiment, except that the revealed tensor entries are observed with additive noise, and the noise level σ in the model (7.30) is set according to a given signal-to-noise ratio (SNR) of 40 dB.

In this experiment, the regularization parameter λ of the problem (7.3) is selected from $\{0, 1/p, 10^{1/2}/p, 10/p\}$ (where the scalar p is the sampling rate) for a rank parameter $R = 14$ and the selected value of λ is $10^{1/2}/p$. All the tested algorithms are terminated if the relative change of the training error attains a given tolerance value:

$$\texttt{relchg} = \frac{|E(U^{t+1}) - E(U^t)|}{|E(U^t)|} \leq \texttt{tol}, \tag{7.31}$$

where E denotes the training RMSE. Also, the algorithms terminates if a heuristic time budget $T_{\max}$ is attained. We set the tolerance parameter as

Table 7.2: Tensor completion with the noiseless observations. Proposed algorithms vs KM16 (r, r, r) with different choices of r. The size of $\mathcal{T}^\star$ is (100, 100, 200), with a Tucker rank $r^\star = (3, 5, 7)$.

Algorithm	R	#variables	iter	Time (sec.)	RMSE (test)
KM16 (R, R, R)	–	7756	29	102.20	2.70e-2
KM16 (12, 12, 12)	–	6096	30	88.32	9.00e-4
KM16 (9, 9, 9)	–	4086	21	29.63	2.70e-05
KM16 (7, 7, 7)	–	2996	22	14.39	2.99e-05
Precon RGD (linemin)	14	5600	82	15.90	1.38e-08
Precon RCG (linemin)	14	5600	34	6.65	1.29e-08
Precon RGD (RBB2)	14	5600	39	**2.69**	9.84e-09

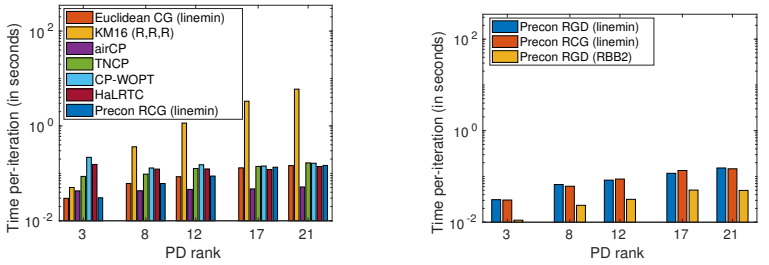


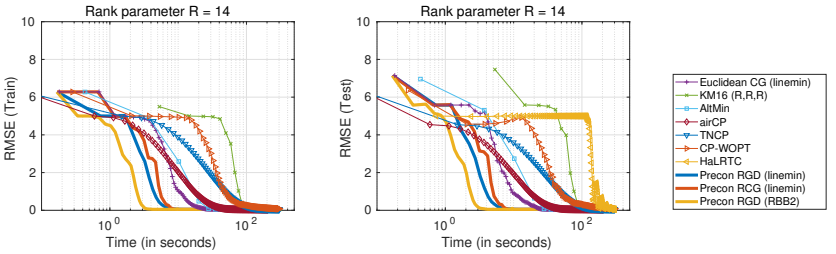
Figure 7.2: Average cost of time per-iteration for various rank parameters. The size of $\mathcal{T}^\star$ is $(100, 100, 200)$ with a Tucker rank $r^\star = (3, 7, 5)$. The sampling rate is 12%.

$\texttt{tol} = 10^{-6}$ and the time budget as $T_{\max} = 300$s (seconds).

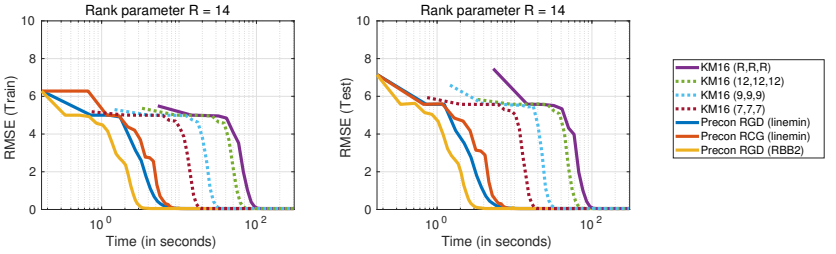
The performances of the tested algorithms are shown in Figure 7.3. From these results, we have similar observations as in the previous experiments: (i) For a polyadic decomposition rank R that is larger than $\max(r_1^\star, r_2^\star, r_3^\star)$, all algorithms achieve a recovery performance of the same level (with a test RMSE around 0.050); (ii) the proposed algorithms (Precon RGD and Precon RCG) outperform the rest of the algorithms in time with speedups between 4 and 15 times at the several accuracy levels near their respective termination point; (iii) the time efficiency of KM16 (r, r, r) improves significantly when the core tensor dimensions (r, r, r) decrease but it remains inferior to those of the proposed algorithms; see Figure 7.3(d)–7.3(e).

7.4.2 Real data

In this subsection, we conduct experiments on a real-world dataset. We focus on evaluating the time efficiency of the proposed algorithms under various choices of the rank parameter R. To ensure a good generalization performance of the tensor completion model, we activate the regularization terms of the problem (7.3) with a regularization parameter $\lambda > 0$.



(a) Training RMSEs of all algorithms (b) Test RMSEs of all algorithms



(c) Comparison with KM16 (d) Comparison with KM16

Figure 7.3: Tensor completion from noisy observations. The size of $\mathcal{T}^\star$ is $(300, 500, 200)$ with rank $r^\star = (3, 7, 5)$. The sampling rate is 5%. The rank parameter $R = 14$.

Dataset and algorithms. The tensor completion tests are conducted on the MovieLens 1M dataset[1], which consists of 1 million movie ratings from 6040 users on 3952 movies and seven-month period from September 19th, 1997 through April 22nd, 1998. Each movie rating in this dataset has a time stamp, which is the number of week during which a movie rating was given. Therefore, this dataset has a tensor form $\mathcal{T}^\star$ of size $6040 \times 3952 \times 150$, where the first two indices are the user and movie identities and the third order index is the time stamp. The dataset contains over 10^6 ratings, which correspond to the known entries of the data tensor $\mathcal{T}^\star$. For the tensor completion tasks, we randomly select 80% of the known ratings as the training set. Note that in this case, the absolute sampling rate $p = |\Omega|/(m_1 m_2 m_3)$ is 2.23%.

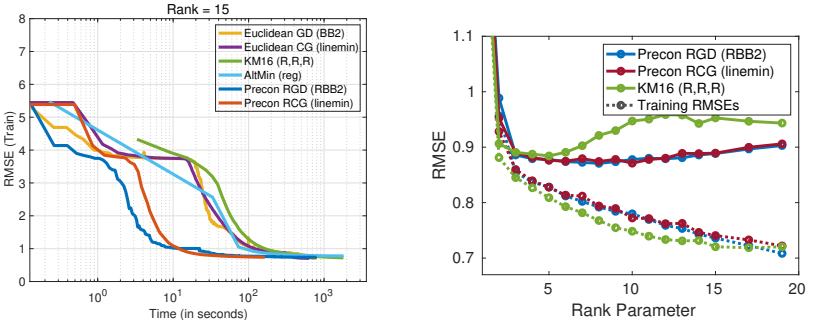
Due to the large dimensions of the data tensor of MovieLens 1M, several aforementioned algorithms in the related work were not tested on this dataset due to excessive memory requirements. In the implementation of these algorithms, the tensor index filtering operations such as accessing $\mathcal{T}^\star_{|\Omega}$ or $\mathcal{T}_{|\Omega}$ in iterations, a dense tensor format is used for the index set Ω, which—for the same size as $\mathcal{T}^\star$—requires the storage of over 3.5×10^9 entries in total; Such a data format poses a memory requirement bottleneck that blocks the test. On the other hand, the proposed algorithms and Euclidean GD/CG, KM16 and AltMin can be run without the memory issue since they use the

[1] https://grouplens.org/datasets/movielens/1m/

COO format for the training set Ω, which corresponds to only 10^6 entries on the same dataset. Note that for all tensor decomposition-based algorithms, the memory requirement for the decomposition variables (or factor matrices) is $O((m_1+m_2+m_3)R)$—which is bounded by 10^5 for any rank parameter $R < 100$ in the experiments on MovieLens-1M—is also memory-efficient. Therefore, the algorithms that are tested are: the proposed algorithms (Precon RGD/RCG), Euclidean GD/CG, KM16 and AltMin.

Experiments and results. Given the data tensor $\mathcal{T}^\star$ and the index set Ω as the training set, we conduct performance evaluations using various choices for the rank parameter R, after selection of the regularization parameter λ. The parameter λ for the CPD-based algorithms is selected among $\{0,\ 1/p,\ 10^{1/3}/p,\ 10^{2/3}/p,\ 10/p,\ 10^{4/3}/p\}$ via 3-fold cross validation using the Euclidean GD algorithm (instead of the proposed ones), where the rank parameter R is set to be 5, 10 and 15 respectively; and the values selected by these cross validation procedures are $10^{2/3}/p$ (when $R = 5$ and 10) and $10^{4/3}/p$ (when $R = 15$). For the Tucker decomposition-based algorithm—KM16—with the Tucker rank $r = (R, R, R)$, for $R \in \{5, 10, 15\}$, the values of λ selected after the same cross validation procedure are 0. Subsequently, we test all algorithms using the selected parameters. Similar to previous tests, the stopping criteria for all the tested algorithms use the relative change of training errors, i.e., `relchg` in (7.31), and a large enough time budget $T_{\max}$. We set the tolerance parameter for `relchg` (7.31) as `tol` $= 10^{-5}$ and the maximal time budget as $T_{\max} = 1800$s.

We present the iteration histories of all algorithms under the rank parameter $R = 15$ in Figure 7.4(a). We also observe the recovery performances of the algorithms under a series of different rank parameters—for $R \in \{1, \ldots, 15, 17, 19\}$ and $\lambda = 10^{2/3}/p$; see Figure 7.4(b).



(a) Training RMSE by time. (b) Training and test RMSEs.

Figure 7.4: Tensor completion on the MovieLens 1M dataset. Recovery performances with various rank parameters and comparisons of the algorithms (with $R = 15$) in time.

Moreover, Figure 7.5 shows the iteration histories of the proposed algorithm

Precon RGD (using the BB stepsize) in comparison with KM16, with the rank parameters $R \in \{3, 6, 15\}$.

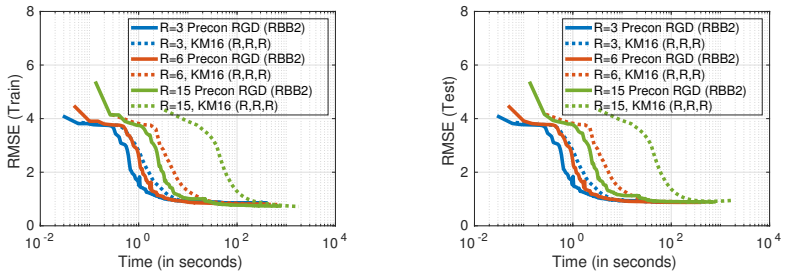


Figure 7.5: Proposed algorithms and KM16 [KM16]. Dataset: MovieLens 1M. The rank parameters R are $\{3, 6, 15\}$. Left: training RMSE by time; Right: test RMSE by time.

From Figure 7.4(a), we observe that (i) the two proposed algorithms have faster convergence behaviors than all the rest of the tested algorithms, and (ii) in particular, the proposed algorithms achieve the same recovery error with speedups of around 10 times compared to KM16, and 6 times compared to Euclidean CG (linemin). The results in Figure 7.4(b) give an overview of the performance of our algorithms and KM16 in terms of their trade-offs between the model complexity (for the minimization of the fitting error) and the generalization of the model for the prediction of missing entries. From these results, we can see that (under the several randomly generated regularization parameters so-far explored), the rank choice of $R = 8$ provides the best recovery error on (unknown) test entries. Moreover, for rank choices that are larger than $R = 8$, the decrease in the recovery performances of the proposed algorithms (compared to the case with $R = 8$) is much smaller than that of KM16. This can be explained by the fact that the search space of our algorithms under larger rank parameters contain those with smaller ranks, while the search space of KM16 corresponds to matrix spaces of a fixed column-rank.

7.5 Conclusion

We proposed a new class of Riemannian preconditioned first-order algorithms for tensor completion through low-rank polyadic decomposition. We have analyzed the convergence properties of Riemannian gradient descent using the proposed Riemannian preconditioning. The main feature of the proposed algorithms stems from a new Riemannian metric defined on the product space of the factor matrices of polyadic decomposition. This metric induces a local preconditioning on the Euclidean gradient descent direction of the PD-based objective function; the underlying preconditioner has the form of an approximated inverse of the diagonal blocks of the Hessian of the objective function.

These Riemannian preconditioned algorithms share some characteristics

with the related work [KM16], which deals with tensor completion with a fixed Tucker rank. They differ however in the following sense: the polyadic decomposition model allows for finding a low-rank tensor candidate within a range of CP ranks, while the algorithm of [KM16] searches a tensor solution with a fixed (Tucker) rank.

Because of the more flexible decomposition modeling, our algorithms perform well with various arbitrary choices of the rank parameter in the tensor completion tasks on both synthetic and the MovieLens 1M datasets. Moreover, we have observed that the proposed algorithms provide significant speedup over several state-of-the-art algorithms for CPD-based tensor completion, while providing comparable or better tensor recovery quality.

Appendix

7.A Algorithmic details

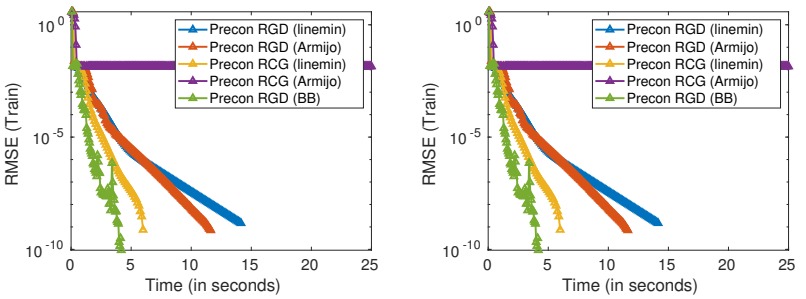
Speed-up by using C/MEX-based MTTKRP. Algorithm 7.2.2 is implemented in a mexfunction and has shown significant speedup over the so-far implemented computations (explicit sparse matricizations times the explicitly computed Khatri–Rao products). Table 7.A.1 shows comparative results on the MovieLens 1M dataset, "Naive" corresponds to the implementation where the gradient computations involve (i) forming sparse matricizations of the residual tensor, (ii) computing the Khatri–Rao products and (iii) sparse-dense matrix multiplication. "Proposed" corresponds to the results of the implementation using sparse MTTKRP (Algorithm 7.2.2).

Table 7.A.1: Speedups of the efficient computational method. The tensor dimensions are $6040 \times 3952 \times 150$.

iter	Time (s)		Average	RMSE
	Naive	Proposed	speedup	Naive/Proposed
1	0.568	0.067	–	4.778 / 4.778
101	71.848	16.199	4.435	0.795 / 0.795
201	142.697	32.333	4.413	0.765 / 0.765
301	213.897	48.508	4.410	0.759 / 0.759
401	285.117	64.609	4.413	0.759 / 0.759
501	356.405	80.712	4.416	0.759 / 0.759

Performance of the three trial stepsize methods. Under the settings of Section 7.4.1, we show in Figure 7.6 the iterative histories of the proposed algorithms in RMSE using the three trial stepsize methods for the line search procedure. The results in the top row is conducted with a low-rank tensor $\mathcal{T}^\star$ generated from (7.30) and rescaled such that $\tilde{\mathbb{E}}[\mathcal{T}^\star_{ijk}] = 1$ while those in the bottom row is based on a tensor under the same settings except that it is rescaled such that $\tilde{\mathbb{E}}[\mathcal{T}^\star_{ijk}] = 10^{-2}$. From the empirical iteration and time performance in these results, we observe that (i) the iteration performance of our algorithms using exact line minimization (7.16) is stable and comparable to the rest of the two stepsize rules but it has the largest per-iteration time

cost; (ii) the iteration and time performance with Armijo backtracking are comparable to the BB stepsize rule but is not stable under changes in the "scale" of the tensor entries. From these observations, we choose to represent our proposed algorithms by Precon RGD (RBB) and Precon RCG (linemin).



(a) Tensor rescaled with $\tilde{\mathbb{E}}[\mathcal{T}^\star_{ijk}] = 1$ (b) Tensor rescaled with $\tilde{\mathbb{E}}[\mathcal{T}^\star_{ijk}] = 10^{-2}$

Figure 7.6: Comparisons between the stepsize selection methods. $\mathcal{T}^\star$ is generated with the model (7.30) and is partially observed without any noise: Tensor size $(100, 100, 100)$, Tucker rank $r^\star = (10, 10, 10)$. The rank parameter $R = 15$. The sampling rate is set to 0.2.

7.B Experimental details

Synthetic model. The noise level in the synthetic tensor model (7.30) is determined according to the signal-to-noise ratio (SNR): $\text{SNR} = \mathbb{E}[T^2]/\mathbb{E}[E^2]$, where T and E are the random variables that represent the tensor entries of the low-rank tensor $\mathcal{T}$ and the noise tensor $\mathcal{E}$ in (7.30). In the experiments with $T \sim \mathcal{N}(0, 1)$ and $E \sim \mathcal{N}(0, \sigma_N)$, the parameter σ_N is computed for a given SNR. The SNR expressed the logarithmic decibel scale (dB) is defined as $\text{SNR (dB)} = 10 \log_{10}(\text{SNR})$.

Initialization. The initial point of all CPD and polyadic decomposition-based algorithms is a tuple $U_0 = (U_0^{(1)}, \ldots, U_0^{(k)})$, where the $m_i \times R$ factor matrices are random Gaussian matrices: $[U_0^{(i)}]_{\ell r} \sim \mathcal{N}(0, 1)$. For the Tucker decomposition-based algorithm (KM16), we choose to construct an initial point that is close enough to $U_0 \in \mathcal{M}$ for fair comparisons. For a Tucker rank $(r_1, \ldots, r_k)$, we initialize KM16 $(r_1, \ldots, r_k)$ with a point in Tucker decomposition form $(G; \tilde{U}^{(1)}, \ldots, \tilde{U}^{(k)})$, such that its tensor representation is close enough to $[\![U_0^{(1)}, \cdots, U_0^{(k)}]\!]$. For this purpose, we set $\tilde{U}_0^{(i)}$ as random Gaussian matrices of size $m_i \times r_i$ with $\tilde{U}_0^{(i)} \sim \mathcal{N}(0, 1)$, and set the core tensor G of size $r_1 \times \cdots \times r_k$ as a random Gaussian matrix with $G_{i_1, \ldots, i_k} \sim \mathcal{N}(0, \sigma)$, where $\sigma = \sqrt{R/r_1 \ldots r_k}$. The choice of this variance parameter is based on the observation that the CPD form can be seen as a special Tucker with diagonal core

tensor $D = \mathrm{diag}(1, ..., 1) \in \mathbb{R}^{R \times \cdots \times R}$. Restricting G to have the same Frobenius norm as D requires that the variance parameter $\sigma = \sqrt{R/r_1 ... r_k}$. In particular, in the case where $r_i = R$, we set $\tilde{U}_0^{(i)} = U_0^{(i)}$, for $i = 1, \ldots, k$.

Performance evaluation. In the experiments, we evaluate the quality of tensor completion with the root-mean-square error (RMSE), for a tensor candidate $\mathcal{T}$ and a given index set Ω', $\mathrm{RMSE}(\Omega') = \|\mathcal{P}_{\Omega'}(\mathcal{T} - \mathcal{T}^\star)\|_{\mathrm{F}} / \sqrt{|\Omega'|}$. The training and test RMSE refer to $\mathrm{RMSE}(\Omega)$ and $\mathrm{RMSE}(\tilde{\Omega}^c)$ respectively, where Ω is the (training) index set of the observed entries used in the definition of the data fitting function f_Ω in (7.3) and the test set $\tilde{\Omega}^c$ is the complementary of Ω in the set of all available entries. In some of the experiments, $\tilde{\Omega}^c$ is a subset of the complementary set (with uniformly distributed indices) such that $|\tilde{\Omega}^c| = (1/4)|\Omega|$ in order to reduce the time for evaluating the test RMSE, if the whole complementary set is overwhelmingly large (2.10^6).

Chapter 8

Conclusion

In this thesis, we focused on low-rank models and algorithms for matrix and tensor completion. With respect to modeling, we explored the usage of graph information in the regularization of matrix and tensor completion problems, and with respect to optimization, we investigated algorithms on Riemannian manifolds of low-rank matrices.

Summary of contributions

In the studies about graph-regularized matrix completion, we took an interest in a problem formulation using low-rank matrix factorization and graph Laplacian-based regularization terms. While the graph-based regularization has been shown, in a previous work by Rao et al. [RYRD15], to be a promising way to enhancing the matrix completion performance, we found that the graph Laplacian-based regularization unavoidably complicates the matrix factorization problem; one way to see this is through the fact that it makes the traditional alternating minimization more costly in this problem. More precisely, in the alternating minimization framework for this problem, each of the two alternating least-squares problems requires solving a Sylvester equation, instead of a simpler and embarrassingly parallelizable linear problem. To alleviate this burden, we went beyond the alternating minimization framework and proposed gradient descent and conjugate gradient algorithms on the product manifold of the low-rank factor matrices. The proposed algorithms differ from alternating minimization in the sense that they update the low-rank factor matrices simultaneously; furthermore, their search directions (on the product manifold) are defined through Riemannian conditioning, which is a recently developed technique that designs a valid Riemannian metric according to the cost function on the manifold. We provided global convergence analysis of the proposed Riemannian gradient descent algorithm.

Through extensive experiments on synthetic and real-world data, we ob-

served that our approach achieves significant speedup compared to the aforementioned alternating minimization algorithm. We also evaluated the matrix recovery qualities of various matrix completion models under different sampling rates. For example, on the Traffic dataset, we observed that, under the samplings ranging from 1% to 40%, graph-regularized models always result in an improvement (Tables 3.2, 3.3 and 4.2) in the completion score over the Frobenius norm-based model (MMMF) and the unregularized model. Such improvements depend naturally on the fact that the given graph Laplacian matrices (in the regularization term) are conformal to the (partially) hidden matrix.

In view of the different algorithms considered in this thesis: GL-RMC, GL-LRGeomCG, ϵNN-LRGeomCG in Chapter 3, and ϵNN-GRMC in Chapter 4, it is natural to ask which algorithm to choose to obtain a better completion result. To answer this: we note that all these algorithms are a hybrid two-stage procedure, which consists of constructing or learning the graph Laplacian matrices first and conducting the graph-regularized matrix completion step subsequently, and among these two consecutive stages, the graph construction or graph learning procedure is the determinant step that defines the graph-regularized matrix completion model. Indeed, we observe that, from Tables 4.2 and 4.3 (column GRMC), once the graph Laplacian matrices are fixed, the different subsequent algorithms (GRALS, Euclidean GD, Qprecon-RGD) produce almost indistinguishable results. Hence the choice of the graph construction method is key to the performance of the two-stage algorithms.

From the results in Table 3.3, we observe that the graph learning approach (GL-LRGeomCG) outperforms the ϵNN graph model (ϵNN-LRGeomCG) in the same experimental setting (sample set Ω, initial point for the matrix completion step), under all sampling rates tested. Hence graph learning is a more favorable choice. This comes with a compromise, however, in the computation time, since the graph learning approach requires a heavier computational effort than computing directly a ϵNN graph, as mentioned in Chapter 3. On the other hand, the time performance comparisons between the matrix completion algorithms are discussed and shown in the experiments in Chapter 4 and 5.

With respect to algorithms, we have focused on gradient descent on a product space of low-rank factor matrices and the manifold of fixed-rank matrices using Riemannian preconditioning. This preconditioning technique exploits a closed-form approximation to the second-order information of the optimization problem. To understand more about this technique, we investigated a gradient descent algorithm, designed with a particular Riemannian preconditioned metric, on the manifold $\mathcal{M}_k$ of fixed-rank matrices. We developed novel results for analyzing this algorithm and showed that it not only enjoys the same advantages of classical low-rank matrix factorization algorithms (compared to convex optimization algorithms in the $m \times n$ matrix space) but can also be analyzed in a more convenient way. In particular, we proved that this algorithm minimizes a class of quadratic functions on $\mathcal{M}_k$ with a local linear convergence rate, under mild conditions. Moreover, the convergence property of the algorithm has

desirable invariance properties in contrast to Euclidean gradient descent algorithms. Because of the efficient iteration efficiency and its light per-iteration cost, the time efficiency of this algorithm is also shown to be much faster than the Euclidean gradient descent algorithms under various rank choices and is faster than many other Riemannian algorithms with fixed-rank matrices.

In addition to matrix completion, we extend the graph-based regularization to low-rank tensor completion, through a CP decomposition-based formulation. In the investigation of the so-called graph-regularized tensor completion, we developed an alternating minimization algorithm for the underlying tensor decomposition problem and also found improvements in the tensor recovery performances when comparing the graph-regularized model to graph-agnostic ones. Furthermore, we also extend the aforementioned Riemannian preconditioned algorithms to the tensor completion problem and discovered significant improvements in time efficiency in comparison with several state-of-the-art algorithms. Another nice feature of the proposed algorithm is that the tensor recovery performances can be ensured under various different choices of the tensor rank parameter; this feature enables a more flexible and lighter parameter selection procedure, compared to several other tensor decomposition-based algorithms, such as the ones developed with a fixed tensor rank.

Perspectives and future work

The graph-regularized models for matrix and tensor completion in Chapters 4 and 6 require the knowledge of graph Laplacian matrices that are related to the partially hidden data in a certain way. However, such relationships remain obscure. From experimental observations, a graph Laplacian matrix that guarantees a relatively small value in the semi-norm of the form (4.2) yields an improvement in the matrix recovery performance. Rao et al. [RYRD15] proposed an error bound of the graph-regularized matrix completion model and the improvement in the upper bound is related to the constants that depend on the graph Laplacian matrices used in the regularization term. Subsequently, they showed, also through numerical simulations, that a number of graph models ensure that the sought constants are better (i.e., smaller) than the counterparts of the graph-agnostic models. The question of how the magnitude of these graph-based constants are related to the graph Laplacian matrices, or whether these constants ensure an improvement over the graph-agnostic model, remains unanswered. It is therefore interesting to look into this question and develop more clear criteria for selecting graphs in the matrix and tensor completion tasks.

In Chapter 5, we investigated convergence properties of quotient manifold-based algorithms for a class of matrix recovery problems. This analysis depends on the global or local properties of critical points of the matrix recovery problem on the manifold $\mathcal{M}_k$. While several recent results made advances in this line of research specifically about the matrix sensing problem, questions re-

main for the matrix completion problem. The main challenge in the case of matrix completion is that the subsampling operator of the matrix completion problem does not ensure the same conditions as the operator underlying the matrix sensing problem. Since the subsampling operator usually yields much smaller computational cost than dense matrix sensing operators, it is interesting to continue exploring the boundaries of matrix recovery performances of Riemannian algorithms for matrix completion, in both the noiseless and noisy scenarios. It is also worth noting that we have focused on gradient descent algorithms designed through Riemannian preconditioning, which exploits a closed-form approximation to the second-order information of the objective function. This approach is particularly appealing with the matrix and tensor completion models using the least squares loss function. Furthermore, since the objective function of such models is in principle a polynomial function of order $2k$, in the case of problems with a k^{th} order tensor (for $k \geq 2$), it would be interesting to explore higher-order information of the matrix and tensor factorization problems; algorithms using higher-order information (e.g., [NP06]) has been shown in previous work to have very good iteration efficiency and perform well in escaping saddle points of nonconvex problems.

Another direction of future work is related to Riemannian algorithms: it is interesting to develop Riemannian gradient descent algorithms with acceleration, and investigate its properties in both its applications and convergence behaviors; it is also interesting to explore randomized variants, or approximate Riemannian algorithms using randomization techniques for large-scale matrix optimization problems.

Bibliography

[AAM14] P.-A. Absil, L. Amodei, and G. Meyer. Two newton methods on the manifold of fixed-rank matrices endowed with riemannian quotient geometries. *Computational Statistics*, 29(3-4):569–590, 2014.

[AB98] C. A. Andersson and R. Bro. Improving the speed of multi-way algorithms:: Part i. tucker3. *Chemometrics and intelligent laboratory systems*, 42(1-2):93–103, 1998.

[ABDF10] M. V. Afonso, J. M. Bioucas-Dias, and M. A. Figueiredo. An augmented lagrangian approach to the constrained optimization formulation of imaging inverse problems. *IEEE Transactions on Image Processing*, 20(3):681–695, 2010.

[ABRS10] H. Attouch, J. Bolte, P. Redont, and A. Soubeyran. Proximal alternating minimization and projection methods for nonconvex problems: An approach based on the kurdyka-łojasiewicz inequality. *Mathematics of Operations Research*, 35(2):438–457, 2010.

[ADKM11] E. Acar, D. M. Dunlavy, T. G. Kolda, and M. Mørup. Scalable tensor factorizations for incomplete data. *Chemometrics and Intelligent Laboratory Systems*, 106(1):41–56, 2011.

[AGH+14] A. Anandkumar, R. Ge, D. Hsu, S. M. Kakade, and M. Telgarsky. Tensor decompositions for learning latent variable models. *Journal of Machine Learning Research*, 15:2773–2832, 2014.

[AM12] P.-A. Absil and J. Malick. Projection-like retractions on matrix manifolds. *SIAM Journal on Optimization*, 22(1):22(1):135—-158, 2012. doi: 10.1137/100802529.

[AMS08] P.-A. Absil, R. Mahony, and R. Sepulchre. *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, Princeton, NJ, 2008.

[BA15] N. Boumal and P. A. Absil. Low-rank matrix completion via preconditioned optimization on the Grassmann manifold. *Linear Algebra and Its Applications*, 475:200–239, 2015. doi: 10.1016/j.laa.2015.02.027.

[BAC19] N. Boumal, P. A. Absil, and C. Cartis. Global rates of convergence for nonconvex optimization on manifolds. *IMA Journal of Numerical Analysis*, 39(1):1–33, 2019, 1605.08101. doi: 10.1093/imanum/drx080.

[BAH16] D. Banco, S. Aeron, and W. S. Hoge. Sampling and recovery of mri data using low rank tensor models. In *2016 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 448–452. IEEE, 2016.

[BED08] O. Banerjee, L. El Ghaoui, and A. D'Aspremont. Model Selection Through Sparse Maximum Likelihood Estimation for Multivariate Gaussian or Binary Data. *The Journal of Machine Learning Research*, 9:485–516, 2008, 0707.0704v1. URL `http://dl.acm.org/citation.cfm?id=1390681.1390696{%}5Cnpapers2://publication/uuid/239BBC05-EDE0-4F1A-B010-4C9768E7D145`.

[BH18] R. F. Barber and W. Ha. Gradient descent with non-convex constraints: local concavity determines convergence. *Information and Inference: A Journal of the IMA*, 7(4):755–806, 2018.

[BK+12] B. W. Bader, T. G. Kolda, et al. Matlab tensor toolbox version 2.5. *Available online, January*, 7, 2012.

[BK+19] B. W. Bader, T. G. Kolda, et al. Matlab tensor toolbox version 3.1. Available online, June 2019. URL `https://www.tensortoolbox.org`.

[BL+07] J. Bennett, S. Lanning, et al. The netflix prize. In *Proceedings of KDD cup and workshop*, volume 2007, page 35. New York, NY, USA., 2007.

[BMAS14] N. Boumal, B. Mishra, P.-A. Absil, and R. Sepulchre. Manopt, a Matlab toolbox for optimization on manifolds. *Journal of Machine Learning Research*, 15:1455–1459, 2014. URL `http://www.manopt.org`.

[BMG13] J. A. Bazerque, G. Mateos, and G. B. Giannakis. Rank regularization and bayesian inference for tensor completion and extrapolation. *IEEE transactions on signal processing*, 61(22):5689–5703, 2013.

[BMN04] M. Belkin, I. Matveeva, and P. Niyogi. Tikhonov regularization and semi-supervised learning on large graphs. In *2004 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 3, pages iii–1000. IEEE, 2004.

[BMR00] E. G. Birgin, J. M. Martínez, and M. Raydan. Nonmonotone Spectral Projected Gradient Methods on Convex Sets. *SIAM Journal on Optimization*, 10(4):1196–1211, 2000. doi: doi:10.1137/S1052623497330963.

[BN03] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, 15(6):1373–1396, 2003.

[BPC+11] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 3(1):1–122, 2011.

[Bro97] R. Bro. Parafac. tutorial and applications. *Chemometrics and intelligent laboratory systems*, 38(2):149–171, 1997.

[Bro98] R. Bro. Multi-way analysis in the food industry-models, algorithms, and applications. In *MRI, EPG and EMA," Proc ICSLP 2000*. Citeseer, 1998.

[BSCB00] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester. Image inpainting. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 417–424. ACM Press/Addison-Wesley Publishing Co., 2000.

[CA15] L. Cambier and P.-A. Absil. Robust Low-Rank Matrix Completion by Riemannian Optimization. *SISC, Siam Journal on Scientific Computing*, 38(5):1–25, 2015. doi: 10.1137/15M1025153.

[Car92] M. P. d. Carmo. *Riemannian geometry*. Birkhäuser, 1992.

[CC70] J. Carroll and J.-J. Chang. Analysis of individual differences in multidimensional scaling via an n-way generalization of Eckart-Young decomposition. *Psychometrika*, 35:283–319, 1970. URL `http://dx.doi.org/10.1007/BF02310791`. 10.1007/BF02310791.

[CCS10] J.-F. Cai, E. J. Candès, and Z. Shen. A singular value thresholding algorithm for matrix completion. *SIAM Journal on optimization*, 20(4):1956–1982, 2010.

[CG09] W. Chu and Z. Ghahramani. Probabilistic models for incomplete multi-dimensional arrays. In *Artificial Intelligence and Statistics*, pages 89–96, 2009.

[CGS09] J. Chen, H. A. Gov, and Y. Saad. Fast Approximate kNN Graph Construction for High Dimensional Data via Recursive Lanczos Bisection Haw-ren Fang. *Journal of Machine Learning Research*, 10, 2009. URL `http://www.jmlr.org/papers/volume10/chen09b/chen09b.pdf`.

[Cha83] B. Chazelle. An improved algorithm for the fixed-radius neighbor problem. *Information Processing Letters*, 16(4):193–198, 1983.

[Chu97] F. R. K. Chung. *Spectral Graph Theory*. American Mathematical Society, 1997.

[CLL+05] R. R. Coifman, S. Lafon, A. B. Lee, M. Maggioni, B. Nadler, F. Warner, and S. W. Zucker. Geometric diffusions as a tool for harmonic analysis and structure definition of data: Diffusion maps. *Proceedings of the national academy of sciences*, 102(21):7426–7431, 2005.

[CM06] R. R. Coifman and M. Maggioni. Diffusion wavelets. *Applied and Computational Harmonic Analysis*, 21(1):53–94, 2006.

[CMD+15] A. Cichocki, D. Mandic, L. De Lathauwer, G. Zhou, Q. Zhao, C. Caiafa, and H. A. Phan. Tensor decompositions for signal processing applications: From two-way to multiway component analysis. *IEEE signal processing magazine*, 32(2):145–163, 2015.

[CP11] E. J. Candès and Y. Plan. Tight oracle inequalities for low-rank matrix recovery from a minimal number of noisy random measurements. *IEEE Transactions on Information Theory*, 57(4):2342–2359, 2011. doi: 10.1109/TIT.2011.2111771.

[CR08] E. Candès and B. Recht. Exact low-rank matrix completion via convex optimization. *2008 46th Annual Allerton Conference on Communication, Control, and Computing*, (m):1–49, 2008, arXiv:0805.4471v1. doi: 10.1109/ALLERTON.2008.4797640.

[CR09a] E. J. Candès and B. Recht. Exact Matrix Completion via Convex Optimization. *Foundations of Computational Mathematics*, 9(6):717–772, 2009. doi: 10.1007/s10208-009-9045-5.

[CR09b] E. J. Candès and B. Recht. Exact matrix completion via convex optimization. *Foundations of Computational mathematics*, 9(6):717, 2009.

[CRPW12] V. Chandrasekaran, B. Recht, P. A. Parrilo, and A. S. Willsky. The Convex Geometry of Linear Inverse Problems. *Foundations of Computational Mathematics*, 12(6):805–849, 2012, 1012.0621. doi: 10.1007/s10208-012-9135-7.

[CS04] P. Chen and D. Suter. Recovering the missing components in a large noisy low-rank matrix: application to SFM. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(8):1051–1063, 2004. doi: 10.1109/TPAMI.2004.52.

[CT10] E. J. Candès and T. Tao. The power of convex relaxation: Near-optimal matrix completion. *IEEE Transactions on Information Theory*, 56(5):2053–2080, 2010, 0903.1476. doi: 10.1109/TIT.2010.2044061.

[CWW17] J.-F. Cai, T. Wang, and K. Wei. Spectral Compressed Sensing via Projected Gradient Descent. *arxiv:1707.09726*, 2017. URL `http://arxiv.org/abs/1707.09726`.

[D+06] D. L. Donoho et al. Compressed sensing. *IEEE Transactions on information theory*, 52(4):1289–1306, 2006.

[DAG18] S. Dong, P.-A. Absil, and K. A. Gallivan. Graph learning for regularized low rank matrix completion. *Proceedings of the 23rd International Symposium on Mathematical Theory of Networks and Systems (MTNS)*, pages 460–467, 2018.

[DAG19] S. Dong, P.-A. Absil, and K. A. Gallivan. Preconditioned Conjugate Gradient Algorithms for Graph Regularized Matrix Completion. In *European Symposium on Artificial Neural Networks (ESANN)*, pages 239–244, 2019.

[DAG20] S. Dong, P.-A. Absil, and K. A. Gallivan. Riemannian gradient descent methods for graph-regularized matrix completion. *Linear Algebra and its Applications*, 2020. doi: https://doi.org/10.1016/j.laa.2020.06.010.

[DG17] D. Dua and C. Graff. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`.

[DGGG21] S. Dong, B. Gao, Y. Guan, and F. Glineur. New Riemannian preconditioned algorithms for tensor completion via polyadic decomposition. *arXiv preprint arXiv:2101.11108*, pages 1–24, 2021. URL `https://arxiv.org/pdf/2101.11108.pdf`.

[DLDMV00a] L. De Lathauwer, B. De Moor, and J. Vandewalle. A multilinear singular value decomposition. *SIAM J. Matrix Anal. Appl.*, 21(4):1253–1278 (electronic), 2000. doi: 10.1137/S0895479896305696.

[DLDMV00b] L. De Lathauwer, B. De Moor, and J. Vandewalle. On the best rank-1 and rank-$(R_1, R_2, \cdots, R_N)$ approximation of higher-order tensors. *SIAM J. Matrix Anal. Appl.*, 21(4):1324–1342 (electronic), 2000. doi: 10.1137/S0895479898346995.

[DS84] P. G. Doyle and J. L. Snell. *Random walks and electric networks*, volume 22. American Mathematical Soc., 1984.

[DSH13] C. Da Silva and F. Herrmann. Hierarchical tucker tensor optimization-applications to tensor completion. sampta 2013. In *10th International Conference on Sampling Theory and Application, Jacobs University Bremen*, 2013.

[DSL08] V. De Silva and L.-H. Lim. Tensor rank and the ill-posedness of the best low-rank approximation problem. *SIAM Journal on Matrix Analysis and Applications*, 30(3):1084–1127, 2008.

[DTFV16] X. Dong, D. Thanou, P. Frossard, and P. Vandergheynst. Learning Laplacian Matrix in Smooth Graph Signal Representations. *IEEE Transactions on Signal Processing*, 64(23):6160–6173, 2016, 1406.7842. doi: 10.1109/TSP.2016.2602809.

[EPO16] H. E. Egilmez, E. Pavez, and A. Ortega. Graph Learning from Data under Structural and Laplacian Constraints. 2016, 1611.05181. doi: 10.1109/JSTSP.2017.2726975.

[Faw06] T. Fawcett. An introduction to roc analysis. *Pattern recognition letters*, 27(8):861–874, 2006.

[FBH03] N. K. M. Faber, R. Bro, and P. K. Hopke. Recent developments in candecomp/parafac algorithms: a critical review. *Chemometrics and Intelligent Laboratory Systems*, 65(1):119 – 137, 2003. doi: 10.1016/S0169-7439(02)00089-8.

[FCRP08] M. Fazel, E. Candès, B. Recht, and P. Parrilo. Compressed sensing and robust recovery of low rank matrices. In *Conference Record - Asilomar Conference on Signals, Systems and Computers*, pages 1043–1047, 2008. doi: 10.1109/ACSSC.2008.5074571.

[FHT08] J. Friedman, T. Hastie, and R. Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441, 2008, 0708.3517. doi: 10.1093/biostatistics/kxm045.

[FR64] R. Fletcher and C. M. Reeves. Function minimization by conjugate gradients. *The Computer Journal*, 7(2):149–154, 1964. doi: 10.1093/comjnl/7.2.149.

[GCZS16] H. Ge, J. Caverlee, N. Zhang, and A. Squicciarini. Uncovering the spatio-temporal dynamics of memes in the presence of incomplete information. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 1493–1502. ACM, 2016.

[GDAG20] Y. Guan, S. Dong, P.-A. Absil, and F. Glineur. Alternating minimization algorithms for graph-regularized tensor completion. *arXiv preprint arXiv:2008.12876*, pages 1–30, 2020. URL `https://arxiv.org/pdf/2008.12876.pdf`.

[Gil59] E. N. Gilbert. Random Graphs. *The Annals of Mathematical Statistics*, 30(4):1141–1144, 1959. doi: 10.1214/aoms/1177706098.

[GJZ17] R. Ge, C. Jin, and Y. Zheng. No spurious local minima in nonconvex low rank problems: A unified geometric analysis. *34th International Conference on Machine Learning, ICML 2017*, 3:1990–2028, 2017, 1704.00708. URL `http://arxiv.org/abs/1704.00708`.

[GKK15] L. Grasedyck, M. Kluge, and S. Krämer. Alternating least squares tensor completion in the tt-format. *arXiv preprint arXiv:1509.00311*, 2015.

[GLM16] R. Ge, J. D. Lee, and T. Ma. Matrix completion has no spurious local minimum. In *Advances in Neural Information Processing Systems*, pages 2973–2981, 2016.

[GM11] D. Goldfarb and S. Ma. Convergence of fixed-point continuation algorithms for matrix rank minimization. *Foundations of Computational Mathematics*, 11(2):183–210, 2011.

[GO09] T. Goldstein and S. Osher. The split bregman method for l1-regularized problems. *SIAM journal on imaging sciences*, 2(2):323–343, 2009.

[GRY11] S. Gandy, B. Recht, and I. Yamada. Tensor completion and low-n-rank tensor recovery via convex optimization. *Inverse Problems*, 27(2):025010, 2011.

[GZA+18] H. Ge, K. Zhang, M. Alfifi, X. Hu, and J. Caverlee. Distenc: A distributed algorithm for scalable tensor completion on spark. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*, pages 137–148. IEEE, 2018.

[Hac12] W. Hackbusch. *Tensor spaces and numerical tensor calculus*, volume 42. Springer, 2012.

[Ham20] W. L. Hamilton. Graph representation learning. *Synthesis Lectures on Artifical Intelligence and Machine Learning*, 14(3):1–159, 2020.

[Har70] R. Harshman. Foundations of the parafac procedure: Models and conditions for an "explanatory" multi-modal factor analysis. *UCLA Working Papers in Phonetics*, 16, 1970.

[Har14] M. Hardt. Understanding alternating minimization for matrix completion. *Proceedings - Annual IEEE Symposium on Foundations of Computer Science, FOCS*, pages 651–660, 2014, arXiv:1312.0925v3. doi: 10.1109/FOCS.2014.75.

[HBDR12] C.-j. Hsieh, A. Banerjee, I. Dhillon, and P. Ravikumar. A divide-and-conquer method for sparse inverse covariance estimation. In *Advances in Neural Information Processing Systems*, volume 25, 2012. URL `https://proceedings.neurips.cc/paper/2012/file/184260348236f9554fe9375772ff966e-Paper.pdf`.

[Her10] D. Hernandez. Simple tensor products. *Inventiones mathematicae*, 181(3):649–675, 2010.

[Hit27a] F. Hitchcock. *The Expression of a Tensor Or a Polyadic as a Sum of Products*. Contributions from the Department of Mathematics. sn., 1927. URL `http://books.google.com/books?id=G7VOHAAACAAJ`.

[Hit27b] F. L. Hitchcock. The expression of a tensor or a polyadic as a sum of products. *Journal of Mathematics and Physics*, 6(1-4):164–189, 1927.

[HK15] F. M. Harper and J. A. Konstan. The MovieLens Datasets: History and Context. *ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4, Article 19 (December 2015), 19 pages.*, 2015. doi: http://dx.doi.org/10.1145/2827872.

[HLJ18] Y. Hu, X. Liu, and M. Jacob. A generalized structured low-rank matrix completion algorithm for MR image recovery. *IEEE transactions on medical imaging*, 38(8):1841–1851, 2018.

[HM14] T. J. Hansen and M. W. Mahoney. Semi-supervised eigenvectors for large-scale locally-biased learning. *The Journal of Machine Learning Research*, 15(1):3691–3734, 2014.

[HMLZ15] T. Hastie, R. Mazumder, J. D. Lee, and R. Zadeh. Matrix completion and low-rank SVD via fast alternating least squares. *The Journal of Machine Learning Research*, 16(1):3367–3402, 2015.

[HN04] X. He and P. Niyogi. Locality preserving projections. In *Advances in neural information processing systems*, pages 153–160, 2004.

[HS52] M. R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of research of the National Bureau of Standards*, 49:409–436, 1952.

[Hul94] J. J. Hull. A database for handwritten text recognition research. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(5):550–554, 1994. doi: 10.1109/34.291440.

[HZ06] W. W. Hager and H. Zhang. A survey of nonlinear conjugate gradient methods. *Pacific journal of Optimization*, 2(1):35–58, 2006.

[IP18] B. Iannazzo and M. Porcelli. The Riemannian Barzilai–Borwein method with nonmonotone line search and the matrix geometric mean computation. *IMA J. Numer. Anal.*, 38(1):495–517, 2018. doi: 10.1093/imanum/drx015.

[JD13] P. Jain and I. S. Dhillon. Provable Inductive Matrix Completion. Technical report, 2013, 1306.0626. URL `http://arxiv.org/abs/1306.0626`.

[JK17] P. Jain and P. Kar. Non-convex optimization for machine learning. *Foundations and Trends® in Machine Learning*, 10(3-4):142–363, 2017. doi: 10.1561/2200000058.

[JMD10] P. Jain, R. Meka, and I. S. Dhillon. Guaranteed rank minimization via singular value projection. In *Advances in Neural Information Processing Systems*, pages 937–945, 2010.

[JNS13a] P. Jain, P. Netrapalli, and S. Sanghavi. Low-rank matrix completion using alternating minimization. *Proceedings of the 45th annual ACM symposium on Symposium on theory of computing - STOC '13*, page 665, 2013, 1212.0467. doi: 10.1145/2488608.2488693.

[JNS13b] P. Jain, P. Netrapalli, and S. Sanghavi. Low-rank matrix completion using alternating minimization. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 665–674, 2013.

[JO14] P. Jain and S. Oh. Provable tensor factorization with missing data. In *Advances in Neural Information Processing Systems*, pages 1431–1439, 2014.

[JWHT13] G. James, D. Witten, T. Hastie, and R. Tibshirani. *An introduction to statistical learning*, volume 112. Springer, 2013.

[KABO10] A. Karatzoglou, X. Amatriain, L. Baltrunas, and N. Oliver. Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 79–86. ACM, 2010.

[Kal16] V. Kalofolias. How to Learn a Graph from Smooth Signals. In A. Gretton and C. C. Robert, editors, *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, volume 51 of *Proceedings of Machine Learning Research*, pages 920–929, Cadiz, Spain, 2016. PMLR. URL `http://proceedings.mlr.press/v51/kalofolias16.html`.

[KB09] T. G. Kolda and B. W. Bader. Tensor decompositions and applications. *SIAM review*, 51(3):455–500, 2009.

[KBBV14] V. Kalofolias, X. Bresson, M. Bronstein, and P. Vandergheynst. Matrix Completion on Graphs. In *NIPS2014 - Robustness in High Dimension*, 2014, 1408.1717. URL `http://arxiv.org/abs/1408.1717`.

[KBV09] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009. doi: 10.1109/MC.2009.263.

[Kie00] H. A. L. Kiers. Towards a standardized notation and terminology in multiway analysis. *Journal of Chemometrics*, 14(3):105–122, 2000. doi: 10.1002/1099-128X(200005/06)14:3<105::AID-CEM582>3.0.CO;2-I.

[KM16] H. Kasai and B. Mishra. Low-rank tensor completion: a Riemannian manifold preconditioning approach. In *International Conference on Machine Learning*, pages 1012–1021, 2016.

[KMO10] R. H. Keshavan, A. Montanari, and S. Oh. Matrix completion from a few entries. *IEEE transactions on information theory*, 56(6):2980–2998, 2010.

[KO09] R. H. Keshavan and S. Oh. OptSpace : A Gradient Descent Algorithm on the Grassman Manifold for Matrix Completion. 2009, arXiv:0910.5260v2. URL `https://arxiv.org/pdf/0910.5260.pdf`.

[KP02] S. G. Krantz and H. R. Parks. *A primer of real analytic functions*. Springer Science & Business Media, 2002.

[KP17] V. Kalofolias and N. Perraudin. Large Scale Graph Learning from Smooth Signals. 2017, 1710.05654. doi: 10.1111/obr.12551.

[KR07] T. Korah and C. Rasmussen. Spatiotemporal inpainting for recovering texture maps of occluded building facades. *IEEE Transactions on Image Processing*, 16(9):2262–2271, 2007.

[Kro83] P. M. Kroonenberg. *Three-mode principal component analysis: Theory and applications*, volume 2. DSWO press, 1983.

[Kru77] J. B. Kruskal. Three-way arrays: rank and uniqueness of trilinear decompositions, with application to arithmetic complexity and statistics. *Linear algebra and its applications*, 18(2):95–138, 1977.

[Kru89] J. B. Kruskal. Rank, decomposition, and uniqueness for 3-way and n-way arrays. *Multiway data analysis*, pages 7–18, 1989.

[KSV14] D. Kressner, M. Steinlechner, and B. Vandereycken. Low-rank tensor completion by riemannian optimization. *BIT Numerical Mathematics*, 54(2):447–468, 2014.

[KTBB99] H. A. Kiers, J. M. Ten Berge, and R. Bro. Parafac2—part i. a direct fitting algorithm for the parafac2 model. *Journal of Chemometrics: A Journal of the Chemometrics Society*, 13(3-4):275–294, 1999.

[Lee97] J. M. Lee. Riemannian manifolds, volume 176 of Graduate Texts in Mathematics, 1997.

[Lee03] J. M. Lee. Smooth manifolds. In *Introduction to Smooth Manifolds*. Springer-Verlag, New York, 2003.

[LHLZ21] Y. Luo, W. Huang, X. Li, and A. R. Zhang. Recursive importance sketching for rank constrained least squares: Algorithms and high-order convergence, 2021, 2011.08360.

[Lic13] M. Lichman. UCI Machine Learning Repository, 2013. URL `https://archive.ics.uci.edu/ml/`.

[Liu06] Y. Liu. Graph-based learning models for information retrieval: A survey. 2006. URL `http://www.cse.msu.edu/{~}rongjin/semisupervised/graph.pdf`.

[LK02] J. Lafferty and R. I. Kondor. Diffusion Kernels on Graphs and Other Discrete Input Spaces. *ICML '02 Proceedings of the Nineteenth International Conference on Machine Learning*, pages 315—-322, 2002. doi: 10.1.1.57.7612.

[LMWY09] J. Liu, P. Musialski, P. Wonka, and J. Ye. Tensor completion for estimating missing values in visual data. *ICCV, 2009*, pages 2114–2121, 2009.

[LMWY12] J. Liu, P. Musialski, P. Wonka, and J. Ye. Tensor completion for estimating missing values in visual data. *IEEE transactions on pattern analysis and machine intelligence*, 35(1):208–220, 2012.

[LNSS16] H. Lamba, V. Nagarajan, K. Shin, and N. Shajarisales. Incorporating side information in tensor completion. In *Proceedings of the 25th International Conference Companion on World Wide Web*, pages 65–66. International World Wide Web Conferences Steering Committee, 2016.

[LS15] A. P. Liavas and N. D. Sidiropoulos. Parallel algorithms for constrained tensor factorization via alternating direction method of multipliers. *IEEE Transactions on Signal Processing*, 63(20):5450–5463, 2015.

[LSC+14] Y. Liu, F. Shang, H. Cheng, J. Cheng, and H. Tong. Factor matrix trace norm minimization for low-rank tensor completion. In *Proceedings of the 2014 SIAM International Conference on Data Mining*, pages 866–874. SIAM, 2014.

[LSJ+14] Y. Liu, F. Shang, L. Jiao, J. Cheng, and H. Cheng. Trace norm regularized candecomp/parafac decomposition with missing data. *IEEE transactions on cybernetics*, 45(11):2437–2448, 2014.

[LW13] P.-L. Loh and M. J. Wainwright. Structure Estimation for Discrete Graphical Models: Generalized Covariance Matrices and Their Inverses. *The Annals of Statistics*, 41(6):3022–3049, 2013, arXiv:1212.0478v2. doi: 10.1214/13-AOS1162.

[LWC12] W. Liu, J. Wang, and S. F. Chang. Robust and scalable graph-based semisupervised learning. *Proceedings of the IEEE*, 100(9):2624–2638, 2012. doi: 10.1109/JPROC.2012.2197809.

[LYZY10] Y. Li, J. Yan, Y. Zhou, and J. Yang. Optimum subspace learning and error correction for tensors. In *European Conference on Computer Vision*, pages 790–803. Springer, 2010.

[MAS12] B. Mishra, K. A. Apuroop, and R. Sepulchre. A Riemannian geometry for low-rank matrix completion. *arXiv preprint arXiv:1211.1550*, 2012, 1211.1550. URL `http://arxiv.org/abs/1211.1550`.

[MBPS10] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online learning for matrix factorization and sparse coding. *Journal of Machine Learning Research*, 11(Jan):19–60, 2010.

[MBS11] G. Meyer, S. Bonnabel, and R. Sepulchre. Linear regression under fixed-rank constraints: a Riemannian approach. In *Proceedings of the 28th international conference on machine learning*, 2011.

[MHE+10] R. Mazumder, T. Hastie, H. Edu, R. Tibshirani, T. Edu, and T. Jaakkola. Spectral Regularization Algorithms for Learning Large Incomplete Matrices. *Journal of Machine Learning Research*, 11:2287–2322, 2010. URL `https://web.stanford.edu/{~}hastie/Papers/mazumder10a.pdf`.

[MHH+06] M. Mørup, L. K. Hansen, C. S. Herrmann, J. Parnas, and S. M. Arnfred. Parallel factor analysis as an exploratory tool for wavelet transformed event-related eeg. *NeuroImage*, 29(3):938–947, 2006.

[MMBS13] B. Mishra, G. Meyer, F. Bach, and R. Sepulchre. Low-rank optimization with trace norm penalty. *SIAM Journal on Optimization, Society for Industrial and Applied Mathematics*, 23(4):2124–2149, 2013. doi: 10.1137/110859646.

[MMBS14] B. Mishra, G. Meyer, S. Bonnabel, and R. Sepulchre. Fixed-rank matrix factorizations and Riemannian low-rank optimization. *Computational Statistics*, 29(3-4):591–621, 2014, arXiv:1209.0430v2. doi: 10.1007/s00180-013-0464-z.

[MS16] B. Mishra and R. Sepulchre. Riemannian preconditioning. *SIAM Journal on Optimization*, 26(1):635–660, 2016.

[MWCC18] C. Ma, K. Wang, Y. Chi, and Y. Chen. Implicit regularization in nonconvex statistical estimation: Gradient descent converges linearly for phase retrieval and matrix completion. In *35th International Conference on Machine Learning, ICML 2018*, volume 8, pages 5264–5331, 2018, 1711.10467. URL `http://arxiv.org/abs/1711.10467`.

[Nes04] Y. Nesterov. *Introductory Lectures on Convex Optimization*, volume 87. Springer Publishing Company, Incorporated, 1 edition, 2004. doi: 10.1007/978-1-4419-8853-9.

[NHTK12] A. Narita, K. Hayashi, R. Tomioka, and H. Kashima. Tensor factorization using auxiliary information. *Data Mining and Knowledge Discovery*, 25(2):298–324, 2012.

[NKS19] L. T. Nguyen, J. Kim, and B. Shim. Low-Rank Matrix Completion: A Contemporary Survey. *IEEE Access*, 7:94215–94237, jul 2019. doi: 10.1109/access.2019.2928130.

[NP06] Y. Nesterov and B. T. Polyak. Cubic regularization of Newton method and its global performance. *Mathematical Programming*, 2006. doi: 10.1007/s10107-006-0706-8.

[NS12] T. Ngo and Y. Saad. Scaled gradients on Grassmann manifolds for matrix completion. In *Advances in Neural Information Processing Systems*, pages 1412–1420, 2012.

[NW06] J. Nocedal and S. Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

[OF97] B. A. Olshausen and D. J. Field. Sparse coding with an overcomplete basis set: A strategy employed by V1? *Vision Research*, 37(23):3311–3325, 1997. doi: 10.1016/S0042-6989(97)00169-7.

[O'n83] B. O'neill. *Semi-Riemannian geometry with applications to relativity*. Academic press, 1983.

[Ose11] I. V. Oseledets. Tensor-train decomposition. *SIAM Journal on Scientific Computing*, 33(5):2295–2317, 2011.

[PFS16] E. E. Papalexakis, C. Faloutsos, and N. D. Sidiropoulos. Tensors for data mining and data fusion: Models, applications, and scalable algorithms. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 8(2):1–44, 2016.

[PFS17] E. E. Papalexakis, C. Faloutsos, and N. D. Sidiropoulos. Tensors for data mining and data fusion: Models, applications, and scalable algorithms. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 8(2):16, 2017.

[PHB17] Y. Park, D. Hallac, and S. Boyd. Learning the Network Structure of Heterogeneous Data via Pairwise Exponential Markov Random Fields. In *Proceedings of the 20th International Conference on Artifi- cial Intelligence and Statistics (AISTATS) 2017*, volume 54, pages 1302–1310, 2017. URL `http://www-cs.stanford.edu/people/jure/pubs/pemrf-aistats17.pdf`.

[PKCS18] D. Park, A. Kyrillidis, C. Caramanis, and S. Sanghavi. Finding low-rank solutions via nonconvex matrix factorization, efficiently and provably. *SIAM Journal on Imaging Sciences*, 11(4):2165–2204, 2018.

[PPS+14] N. Perraudin, J. Paratte, D. Shuman, L. Martin, V. Kalofolias, P. Vandergheynst, and D. K. Hammond. GSPBOX: A toolbox for signal processing on graphs. *ArXiv e-prints*, aug 2014, 1408.5781. URL `http://arxiv.org/abs/1408.5781`.

[PR69] E. Polak and G. Ribiere. Note sur la convergence de méthodes de directions conjuguées. *Rev. Francaise Informat Recherche Opertionelle*, pages 35–43, 1969. URL `http://www.numdam.org/legal.php`.

[PTBD16] H. N. Phien, H. D. Tuan, J. A. Bengua, and M. N. Do. Efficient tensor completion: Low-rank tensor train. *arXiv preprint arXiv:1601.01083*, 2016.

[RFP10] B. Recht, M. Fazel, and P. A. Parrilo. Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. *SIAM Review*, 52(3):471–501, 2010, 0706.4138. doi: 10.1137/070697835.

[RH05] H. Rue and L. Held. *Gaussian Markov random fields: theory and applications*. CRC press, 2005.

[RS05] J. D. M. Rennie and N. Srebro. Fast maximum margin matrix factorization for collaborative prediction. In *Proceedings of the 22nd international conference on Machine learning - ICML '05*, pages 713–719, New York, New York, USA, 2005. ACM Press. doi: 10.1145/1102351.1102441.

[RSS15] H. Rauhut, R. Schneider, and Ž. Stojanac. Tensor completion in hierarchical tensor representations. In *Compressed Sensing and its Applications*, pages 419–450. Springer, 2015.

[RSS17] H. Rauhut, R. Schneider, and Ž. Stojanac. Low rank tensor recovery via iterative hard thresholding. *Linear Algebra and its Applications*, 523:220–262, 2017.

[RW09] R. T. Rockafellar and R. J.-B. Wets. *Variational analysis*, volume 317. Springer Science & Business Media, 2009.

[RYRD15] N. Rao, H.-F. Yu, P. K. Ravikumar, and I. S. Dhillon. Collaborative filtering with graph information: Consistency and scalable methods. In *Advances in neural information processing systems*, pages 2107–2115, 2015.

[Sch01] B. Schölkopf. The kernel trick for distances. In *Advances in neural information processing systems*, pages 301–307, 2001.

[SD19] M. Sørensen and L. De Lathauwer. Fiber sampling approach to canonical polyadic decomposition and application to tensor completion. *SIAM Journal on Matrix Analysis and Applications*, 40(3):888–917, 2019.

[SDDS14] M. Signoretto, Q. T. Dinh, L. De Lathauwer, and J. A. Suykens. Learning with tensors: a framework based on convex optimization and spectral regularization. *Machine Learning*, 94(3):303–351, 2014.

[SDF+17] N. D. Sidiropoulos, L. De Lathauwer, X. Fu, K. Huang, E. E. Papalexakis, and C. Faloutsos. Tensor decomposition for signal processing and machine learning. *IEEE Transactions on Signal Processing*, 65(13):3551–3582, 2017.

[SGCH19] Q. Song, H. Ge, J. Caverlee, and X. Hu. Tensor completion algorithms in big data analytics. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 13(1):1–48, 2019.

[SHZL13] Z. Shi, J. Han, T. Zheng, and J. Li. Guarantees of augmented trace norm models in tensor recovery. In *Twenty-Third International Joint Conference on Artificial Intelligence*, 2013.

[SL16] R. Sun and Z. Q. Luo. Guaranteed Matrix Completion via Non-Convex Factorization. *IEEE Transactions on Information Theory*, 62(11):6535–6579, 2016, 1411.8003. doi: 10.1109/TIT.2016.2598574.

[SNF+13] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst. The emerging field of signal processing on graphs. *IEEE Signal Processing Magazine*, 30(3):83–98, 2013, 1211.0053. doi: 10.1109/MSP.2012.2235192.

[Spi12] D. Spielman. *Spectral Graph Theory*. Combinatorial Scientific Computing. Chapman and Hall/CRC Press, 2012.

[SRJ05] N. Srebro, J. D. Rennie, and T. S. Jaakkola. Maximum-margin matrix factorization. *Advances in Neural Information Processing Systems*, 17:1329–1336, 2005. doi: 10.1.1.59.118.

[SRV16] D. I. Shuman, B. Ricaud, and P. Vandergheynst. Vertex-frequency analysis on graphs. *Applied and Computational Harmonic Analysis*, 40(2):260–291, mar 2016. doi: 10.1016/j.acha.2015.02.005.

[SU15] R. Schneider and A. Uschmajew. Convergence results for projected line-search methods on varieties of low-rank matrices via łojasiewicz inequality. *SIAM Journal on Optimization*, 25(1):622–646, 2015.

[SVBD13] L. Sorber, M. Van Barel, and L. De Lathauwer. Optimization-based algorithms for tensor decompositions: Canonical polyadic decomposition, decomposition in rank-(l_r,l_r,1) terms, and a new generalization. *SIAM Journal on Optimization*, 23(2):695–720, 2013.

[SVBD15] L. Sorber, M. Van Barel, and L. De Lathauwer. Structured data fusion. *IEEE Journal of Selected Topics in Signal Processing*, 9(4):586–600, 2015.

[TB05] G. Tomasi and R. Bro. Parafac and missing values. *Chemometrics and Intelligent Laboratory Systems*, 75(2):163–180, 2005.

[TBS+16] S. Tu, R. Boczar, M. Simchowitz, M. Soltanolkotabi, and B. Recht. Low-rank solutions of linear matrix equations via procrustes flow. In *International Conference on Machine Learning*, pages 964–973. PMLR, 2016.

[TSF14] D. Thanou, D. I. Shuman, and P. Frossard. Learning parametric dictionaries for signals on graphs. *IEEE Transactions on Signal Processing*, 62(15):3849–3862, 2014, 1401.0887. doi: 10.1109/TSP.2014.2332441.

[Tuc66] L. Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31:279–311, 1966. URL `http://dx.doi.org/10.1007/BF02289464`.

[TW13] J. Tanner and K. Wei. Normalized iterative hard thresholding for matrix completion. *SIAM Journal on Scientific Computing*, 35(5):S104–S125, 2013.

[TW16] J. Tanner and K. Wei. Low rank matrix completion by alternating steepest descent methods. *Applied and Computational Harmonic Analysis*, 40(2):417–429, 2016.

[UT19] M. Udell and A. Townsend. Why are big data matrices approximately low rank? *SIAM Journal on Mathematics of Data Science*, 1(1):144–160, 2019.

[UV19] A. Uschmajew and B. Vandereycken. Geometric methods on low-rank matrix and tensor manifolds. Preprint, 2019.

[UV20a] A. Uschmajew and B. Vandereycken. Geometric methods on low-rank matrix and tensor manifolds. In P. Grohs, M. Holler, and A. Weinmann, editors, *Variational methods for nonlinear geometric data and applications*. Springer, 2020. doi: 10.1007/978-3-030-31351-7_9.

[UV20b] A. Uschmajew and B. Vandereycken. On critical points of quadratic low-rank matrix optimization problems. *IMA Journal of Numerical Analysis*, 03 2020, https://academic.oup.com/imajna/advance-article-pdf/doi/10.1093/imanum/drz061/33046976/drz061.pdf. doi: 10.1093/imanum/drz061.

[Van13] B. Vandereycken. Low-Rank Matrix Completion by Riemannian Optimization. *SIAM Journal on Optimization*, 23(2):1214–1236, 2013, arXiv:1209.3834v1. doi: 10.1137/110845768.

[VD19] N. Vervliet and L. De Lathauwer. Numerical optimization-based algorithms for data fusion. In *Data Handling in Science and Technology*, volume 31, pages 81–128. Elsevier, 2019.

[VDS+16] N. Vervliet, O. Debals, L. Sorber, M. V. Barel, and L. D. Lathauwer. Tensorlab 3.0. Available online, Mar. 2016. URL `http://www.tensorlab.net`.

[vR79] C. J. van Rijsbergen. *Information retrieval*. Butterworths, London, 2 edition, 1979. URL `http://www.dcs.gla.ac.uk/Keith/Preface.html`.

[WCCL16] K. Wei, J.-F. Cai, T. F. Chan, and S. Leung. Guarantees of Riemannian optimization for low rank matrix recovery. *SIAM Journal on Matrix Analysis and Applications*, 37(3):1198–1222, 2016.

[WZ12] Y.-X. Wang and Y.-J. Zhang. Nonnegative matrix factorization: A comprehensive review. *IEEE Transactions on Knowledge and Data Engineering*, 25(6):1336–1353, 2012.

[XCH+10] L. Xiong, X. Chen, T.-K. Huang, J. Schneider, and J. G. Carbonell. Temporal collaborative filtering with bayesian probabilistic tensor factorization. In *Proceedings of the 2010 SIAM international conference on data mining*, pages 211–222. SIAM, 2010.

[XJZ13] M. Xu, R. Jin, and Z.-H. Zhou. Speedup matrix completion with side information: Application to multi-label learning. In *Advances in neural information processing systems*, pages 2301–2309, 2013.

[XY13a] Y. Xu and W. Yin. A Block Coordinate Descent Method for Regularized Multiconvex Optimization with Applications to Nonnegative Tensor Factorization and Completion. *SIAM Journal on Imaging Sciences*, 6(3):1758–1789, 2013. doi: 10.1137/120887795.

[XY13b] Y. Xu and W. Yin. A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion. *SIAM Journal on imaging sciences*, 6(3):1758–1789, 2013.

[YHS13] L. Yang, Z.-H. Huang, and X. Shi. A fixed point iterative method for low n-rank tensor pursuit. *IEEE Transactions on Signal Processing*, 61(11):2952–2962, 2013.

[YJHB14] G. Yi, L. Jianxin, Q. Hangping, and W. Bo. Survey of structure from motion. In *Proceedings of 2014 International Conference on Cloud Computing and Internet of Things*, pages 72–76. IEEE, 2014.

[YRD16] H.-F. Yu, N. Rao, and I. S. Dhillon. Temporal Regularized Matrix Factorization for High-dimensional Time Series Prediction. In *Advances in Neural Information Processing Systems 29*, pages 847–855, 2016. URL `http://www.cs.utexas.edu/{~}rofuyu/papers/tr-mf-nips.pdf`.

[YZC16] T. Yokota, Q. Zhao, and A. Cichocki. Smooth parafac decomposition for tensor completion. *IEEE Transactions on Signal Processing*, 64(20):5423–5436, 2016.

[ZCLG14] X. Zhu, C. Change Loy, and S. Gong. Constructing robust affinity graphs for spectral clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1450–1457, 2014.

[ZDG18] X. Zhang, S. Du, and Q. Gu. Fast and Sample Efficient Inductive Matrix Completion via Multi-Phase Procrustes Flow. In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 5756–5765, Stockholmsmässan, Stockholm Sweden, 2018. PMLR. URL `http://proceedings.mlr.press/v80/zhang18b.html`.

[ZGL03] X. Zhu, Z. Ghahramani, and J. D. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the 20th International conference on Machine learning (ICML-03)*, pages 912–919, 2003.

[ZH08] F. Zhang and E. R. Hancock. Graph spectral image smoothing using the heat kernel. *Pattern Recognition*, 41(11):3328–3342, 2008.

[ZHG+16] G. Zhou, W. Huang, K. A. Gallivan, P. Van Dooren, and P. Absil. A Riemannian rank-adaptive method for low-rank optimization. *Neurocomputing*, 192:72–80, jun 2016. doi: 10.1016/j.neucom.2016.02.030.

[ZL16] Q. Zheng and J. Lafferty. Convergence Analysis for Rectangular Matrix Completion Using Burer-Monteiro Factorization and Gradient Descent. Technical report, 2016, 1605.07051. URL `http://arxiv.org/abs/1605.07051`.

[ZLWZ18] J. Zhang, H. Liu, Z. Wen, and S. Zhang. A sparse completely positive relaxation of the modularity maximization for community detection. *SIAM Journal on Scientific Computing*, 40(5):A3091–A3120, 2018. URL `https://epubs.siam.org/doi/10.1137/17M1141904`.

[ZSBS12] T. Zhou, H. Shan, A. Banerjee, and G. Sapiro. Kernelized probabilistic matrix factorization: Exploiting graphs and side information. In *Proceedings of the 2012 SIAM international Conference on Data mining*, pages 403–414. SIAM, 2012.

[ZWSP08] Y. Zhou, D. Wilkinson, R. Schreiber, and R. Pan. Large-scale parallel collaborative filtering for the Netflix prize. In *International conference on algorithmic applications in management*, pages 337–348. Springer, 2008.

[ZZC15a] Q. Zhao, L. Zhang, and A. Cichocki. Bayesian cp factorization of incomplete tensors with automatic rank determination. *IEEE transactions on pattern analysis and machine intelligence*, 37(9):1751–1763, 2015.

[ZZC15b] Q. Zhao, L. Zhang, and A. Cichocki. Bayesian sparse tucker models for dimension reduction and tensor completion. *arXiv preprint arXiv:1505.02343*, 2015.

[ZZHN15] Z. Zhao, L. Zhang, X. He, and W. Ng. Expert Finding for Question Answering via Graph Regularized Matrix Completion. *IEEE Transactions on Knowledge and Data Engineering*, 27(4):993–1004, 2015. doi: 10.1109/TKDE.2014.2356461.