ABSTRACT

ZHANG, MIN. Structured Smooth Optimization in Statistical Learning. (Under the direction of Eric Chi).

Convex and nonconvex optimization problems are frequently encountered in many research fields. In this thesis, we study several models with different model structure assumptions under the smooth optimization framework.

In many applications, data are two-dimensional and represented as data matrices where each row represents an observation, whereas each column represents a feature of each observation. In Chapter 2, we propose a general method that can simultaneously exploit the correlations among both the rows and columns of the data matrix to efficiently compute the affinities in a missing data setting. We introduce a multi-scale metric that captures the geometry of the complete data matrix and represents the row and column similarities under an optimization framework. The multi-scale metric can serve as affinity weights in many applications even when data are incomplete. We present experimental results to illustrate the effectiveness of the method on common machine learning problems and show the metric is general and can be easily adapted to other applications.

In Chapter 3, we consider the problem of low-rank plus sparse matrix decomposition. It is proved that this problem can be solved via heuristically solving a convex relaxation problem where the nuclear norm and ℓ_1 norm are utilized to induce low-rank and sparsity. Many algorithms exploit first-order methods that involve the proximal operator of the nuclear norm and require conducting the singular value decomposition (SVD). In each iteration, the dominant cost is computing an SVD of the same size as the input data, which has a superlinear cost.The need to compute a sequence of full or partial SVD severely limits the applicability of these algorithms to large-scale problems. To remedy this, we introduce the low-rank bi-factorization structure and characterize the nuclear norm in terms of the Frobenius norm. With this low-rank bi-factorization, we develop a more scalable algorithm that avoids the computational burden of SVD and establish the convergence properties. Our method has a faster convergence speed in practice and the per iteration cost is greatly reduced.

In Chapter 4, we shift our attention to tensor-valued data that arise in many applications such as image genetics. A tensor is a multidimensional array. Practitioners often deal with tensor-valued data by reshaping tensors into vectors first. Naively transforming the tensor data into high-dimensional vectors will result in unsatisfactory results as vectorization

destroys the structural information within the tensor structure such as shapes and spatial correlations. It also increases the dimensionality of the problem as well as the computational complexity. Therefore, it is of interest to develop a framework for tensor data instead of this unsophisticated transformation. We study the problem of canonical correlation analysis in the context of tensor-valued data. We present an estimation algorithm for solving tensor canonical correlation analysis (TCCA) and apply the method to several real datasets. We tackle some problems that are not covered in the previous paper and fill a knowledge gap on tensor analysis by bringing new insights. The results illustrate the effectiveness and efficiency of the method and exhibit that the method can be used to extract useful and interpretable representations from the data. We consider several practical issues that arise in real data analysis and demonstrate new ways of how to use the method, which extends the scope of the existing framework.

Structured Smooth Optimization in Statistical Learning

by
Min Zhang

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

Statistics

Raleigh, North Carolina
2021

APPROVED BY:

Arvind Krishna Saibaba	Minh Tang
Jung-Ying Tzeng	Eric Chi Chair of Advisory Committee

DEDICATION

To my beloved family.

BIOGRAPHY

Min Zhang received her B.S. degree in Statistics from Wuhan University, Wuhan, China, in 2017. Later, Min was enrolled in the Ph.D. program of the Department of Statistics at North Carolina State University. She worked under the guidance of Dr. Eric Chi. She will graduate with a Ph.D. in Statistics in 2021.

ACKNOWLEDGEMENTS

First of all, I would like to express my deepest gratitude to my advisor, Dr. Eric Chi, for his support and guidance during these years. I feel lucky to work with Dr. Chi throughout my Ph.D. study. He has inspired me immensely not only in research topics but also in my growth.

I am also very grateful to my committee members, for their valuable suggestions. They provided many insights and helped me improve my projects.

I would like to thank my parents for their unconditional love and support, thank all my friends for their wonderful company throughout this journey, and thank my cat Doupi for bringing me warmth and happiness.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER

1

INTRODUCTION

In optimization problems, we are looking for the best solution that minimizes the objective function from all feasible solutions. With different model assumptions and constraints, different strategies and algorithms could be designed accordingly. Under the smooth optimization framework, we study several models with different model structure assumptions in this thesis.

In Chapter 2, we study the problem of affinity metric learning when missing data is presented from an optimization perspective. Affinity is a metric that quantifies the pairwise similarity between observations or features in a dataset. The choice of weights can dramatically impact the effectiveness of the algorithm. Existing methods fail to learn the geometry of both rows and columns simultaneously. Additionally, these constructed metrics are based on the complete data and do not address the critical problem of missing data. In these cases, we seek a method that can exploit the correlations among both the rows and columns of the data matrix to efficiently compute the affinities in a missing data setting. We propose a general method that can simultaneously exploit the correlations among both the rows and columns of the data matrix to efficiently compute the affinities in a missing data setting. In this method, a multi-scale metric is introduced and it captures the geometry of the complete data matrix and represents the row and column similarities under an optimization

framework. The multi-scale metric can serve as affinity weights in many applications even when data are incomplete. Experimental results are presented to illustrate the effectiveness of the method on common machine learning problems. Furthermore, the metric is general and can be easily adapted to other applications.

In Chapter 3, we consider the problem of low-rank plus sparse matrix decomposition, where the goal is to recover a low-rank and a sparse component from the given observation. Robust Principal Component Analysis is a representative example. This problem can be solved via heuristically solving a convex relaxation problem where the nuclear norm and ℓ_1 norm are utilized to induce low-rank and sparsity [Wright et al. (2012); Candès and Recht (2009)]. Many existing algorithms are based on first-order methods and involve the proximal operator of the nuclear norm. These algorithms require conducting the singular value decomposition (SVD) in each iteration. The need to compute a sequence of full or partial SVD of the same size as the input data severely limits the applicability of these algorithms to large-scale problems. In order to resolve this, we introduce the low-rank bi-factorization structure and decompose the low-rank matrices as the matrix product of two factor matrices. Matrices of rank at most k are those that can be factored into $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{V}$, $\boldsymbol{U} \in \mathbb{R}^{n\times k}, \boldsymbol{V} \in \mathbb{R}^{m\times k}$, and so seeking a low-rank matrix is equivalent to seeking a low-dimensional factorization [Srebro et al. (2004)]. This factorization allows us to characterize the nuclear norm in terms of the Frobenius norm. With this low-rank bi-factorization, we develop a more scalable algorithm that avoids the computational burden of SVD. The proposed method has a faster convergence speed in practice and the per iteration cost is greatly reduced. The convergence property of the proposed algorithm is also established.

While in Chapter 2 and Chapter 3, we study the optimization problem when data are two-dimensional and represented as data matrices where each row represents an observation, whereas each column represents a feature of each observation. In Chapter 4, we shift our attention to tensor-valued data that arise in many applications such as image genetics. A tensor is a multidimensional array. Previous practitioners often deal with tensor-valued data by reshaping tensors into vectors first. This naive transformation of tensors into high-dimensional vectors will result in unsatisfactory results as vectorization destroys the structural information within the tensor structure such as shapes and spatial correlations. Additionally, it increases the dimensionality of the problem as well as the computational complexity. Therefore, it is of great interest to develop a framework for tensor data instead of this unsophisticated transformation. We consider the problem of canonical correlation analysis in the context of tensor-valued data. By building off an existing estimation algorithm for solving tensor canonical correlation analysis (TCCA), we

made several new contributions. We address several practical issues that arise in real data analysis and apply the method to several real datasets. We tackle some problems that are not covered in the previous paper and fill a knowledge gap on tensor analysis by bringing new insights. The results illustrate the effectiveness and efficiency of the method and exhibit that the method can be used to extract useful and interpretable representations from the data. Furthermore, we consider the high-dimensional scenario when data are presented as data matrices and demonstrate new ways of using the TCCA method by tensorizing matrices first. We also provide practical guidance in this direction, which extends the scope of the existing framework.

CHAPTER

2

MULTI-SCALE AFFINITIES WITH MISSING DATA: ESTIMATION AND APPLICATIONS

2.1 Introduction

In many applications, data are two-dimensional and represented as data matrices where each row represents an observation, whereas each column represents a feature of each observation. Weights, or affinities, quantifying pairwise similarity between observations or features in a dataset are widely used in many machine learning problems. The choice of weights can dramatically affect the effectiveness of the algorithm. In unsupervised learning, the success of clustering techniques depends on the choice of the similarity measure between data points being clustered. Such pairwise similarity measures or pairwise distances of data points can be used to construct a graph on the data. Spectral clustering [Ng et al. (2001); von Luxburg (2007)], treating data points as nodes of a graph, makes use of the eigenvectors of a graph affinity matrix as a new representation space in which to

partition data into disjoint meaningful groups. An ideal affinity graph gives a perfect clustering result [von Luxburg (2007)]. In convex clustering [Hocking et al. (2011); Lindsten et al. (2011); Pelckmans et al. (2005)], a proper choice of weights will ensure the construction of an entire well-nested hierarchical partition tree [Chi and Steinerberger (2019)]. In supervised learning, kernel regression [Nadaraya (1964); Watson (1964)] is a non-parametric estimation technique that uses a kernel function to weight the observations of the learning sample, depending on their "distance" from the predicted observation. In the k-nearest neighbors (k-NN) algorithm, which can be used for both classification and regression, a useful technique is to assign weights to the contributions of the neighbors so that closer neighbors contribute more to the average than the more distant ones. Setting the weights appropriately can dramatically improve the generalization of the k-NN algorithm [Ling and Wang (1997)].

When there is no missing data, the most common practice to quantify similarities as weights between pairs of rows and pairs of columns of the data matrix is to use Gaussian kernel affinities. When data are missing, however, computing affinity weights becomes non-trivial. For example, kernel-based manifold learning methods rely on calculating a similarity matrix between observations to yield a new embedding of the data through an eigendecomposition [Belkin and Niyogi (2003); Coifman and Lafon (2006)]. Naively ignoring missing values can distort the distances between data points and sabotage efforts to learn representative embeddings. Recently, Gilbert and Sonthalia [Gilbert and Sonthalia (2018)] proposed the MR-MISSING algorithm and used a graph metric repair strategy to learn metrics and metric embeddings in datasets with missing data. They first estimated an initial distance matrix from the incomplete data. Then they used the Increase Only Metric Repair (IOMR) [Gilbert and Jain (2017)] method to fix the distance matrix so that it can be used as the metric to compute low dimensional representations. Methods like MR-MISSING, however, account for similarities along either only the rows (observations) or only the columns (features) of a data matrix and does not account for any potential coupled structure of the rows and the columns.

Yet in many applications, for example, gene expression analysis [Chi et al. (2017)], neuroscience [Mishne et al. (2016)] and recommendation systems [Bennett and Lanning (2007)], there is an underlying geometry to both the rows (observations) and the columns (features) of the data matrix [Chi et al. (2017); Coifman and Gavish (2011); Gavish and Coifman (2012); Mishne et al. (2018, 2016); Rabin and Fishelov (2019)]. In gene expression data, subsets of samples (observations) have similar genetic profiles, and subsets of genes (features) have similar expressions across groups of samples. The relationships between

the rows may be informed by the relationships between the columns, and vice versa.

Recent work [Ankenman (2014); Coifman and Gavish (2011); Gavish and Coifman (2012); Mishne et al. (2018, 2016)] exploit this coupled correlation structure of both rows and columns to co-organize matrices. Gavish and Coifman [Gavish and Coifman (2012)] introduced an an approach for matrix structured datasets to recover the smooth joint organization of the features and observations. The organization of the data relies on the construction of a pair of hierarchical partition trees on the observations and on the features. Mishne et al. [Mishne et al. (2018)] proposed multi-scale data-driven transforms and metrics based on trees that are smooth with respect to an underlying geometric structure in the data. None of these methods, however, learns the geometry of both rows *and* columns simultaneously. Additionally, these constructed metrics are based on the complete data and do not address the critical problem of missing data. If we are given those metrics as prior knowledge, we can reliably recover the underlying coupled geometry [Coifman and Gavish (2011)]. Yet how to construct them in the presence of missing data remains an open question. In these cases, we seek a method that can exploit the correlations among both the rows and columns of the data matrix to efficiently compute the affinities in a missing data setting.

In this chapter, we propose a flexible framework to compute affinity weights that simultaneously account for the coupled structure of the rows and columns in the presence of missing data. We present a multi-scale metric that captures the geometry of the complete data matrix and represents the row and column similarities. This metric can be used to calculate the affinity weights in many applications where data is often missing. Mishne et al. [Mishne et al. (2019)] exploited the multi-scale metric to learn low-dimensional embeddings of both the rows and columns of a data matrix. By applying diffusion maps [Coifman et al. (2005)], a dimension reduction technique on the multi-scale distances, local connections found in the data are integrated into a global representation. We will show how this affinity construction strategy can address a wider range of machine learning problems beyond learning low-dimensional embeddings.

In the multi-scale approach, we estimate a collection of complete matrix approximations of a partially observed data matrix that have been smoothed along their rows and columns to different degrees. Row and column multi-scale metrics are calculated based on the collection of estimated completed matrices to encode the affinities between pairs of rows and columns. We offer the following contributions:

- We propose a general method to simultaneously construct row and column affinities of the data matrix when the data is only partially observed. This method is distinct

from other related methods in that our ultimate goal is not to perform specific tasks such as manifold learning or clustering. We aim at presenting a general framework that integrates the task of encoding similarity structure as hyperparameters in many real applications and providing better solutions to those applications through our multi-scale procedure.

- We present a multi-scale metric that leverages both row and column smoothness between pairs of rows and pairs of columns under an optimization framework. By exploiting correlations that exist among both rows and columns, the new metric introduces a coupling between the rows and the columns.
- The estimation runs at multiple scales to encode different levels of smoothness instead of determining a single scale of the solution as in [Chi et al. (2017)]. We aggregate solutions at different scales to estimate the underlying geometry both locally and globally. Consequently, our approach eliminates the need to identify a single "ideal" scale at which to fill in the points.
- We present experimental results to illustrate the effectiveness of the method on common machine learning problems, and show the metric can be easily adapted to other applications.

The rest of the chapter is organized as follows. In Section 2.2, we present an optimization framework obtaining smooth estimates of a partially observed data matrix that will be combined to calculate row and column multi-scale metrics. In Sections 2.3, 2.4, and 2.5, we show a sampling of the breadth of how our multi-scale affinities can be used in common problems in supervised and unsupervised learning to demonstrate its effectiveness and flexibility. We apply the new metric in different applications and compare the performance of the proposed methods through experimental results on different tasks.

2.2 Preliminaries

Let $\boldsymbol{X} \in \mathbb{R}^{n \times p}$ be a partially observed data matrix, where $\Theta \subset [n] \times [p]$ is the subset of indices for which x_{ij} is observed and $[n]$ denotes the set of indices $\{1, \ldots, n\}$. Let $\mathscr{P}_\Theta$ denote the projection operator of $n \times p$ matrices onto the index set Θ such that $[\mathscr{P}_\Theta(\boldsymbol{X})]_{ij}$ is x_{ij} if $(i, j) \in \Theta$ and is 0 otherwise. The ith row and jth column of the matrix $\boldsymbol{X}$ are denoted by $\boldsymbol{X}_{i\cdot}$ and $\boldsymbol{X}_{\cdot j}$ respectively. Let $\mathscr{G}_r = (\mathscr{V}_r, \mathscr{E}_r, \boldsymbol{W}_r)$ denote an undirected weighted row graph

with a vertex set $\mathcal{V}_r = [n]$ and an edge set $\mathcal{E}_r = \mathcal{V}_r \times \mathcal{V}_r$ where $(i, i') \in \mathcal{E}_r$ has an edge weight $w_{ii'}$ defined by the ii'th entry of a nonnegative symmetric weight matrix $\boldsymbol{W}_r \in \mathbb{R}^{n \times n}$. The column graph $\mathcal{G}_c = (\mathcal{V}_c, \mathcal{E}_c, \boldsymbol{W}_c)$ is defined analogously.

We construct affinity weights through a multi-scale procedure which requires computing a collection of smooth estimates of the incomplete data matrix at different levels of row and column smoothness. A new multi-scale metric is presented to estimate a coupled row and column geometry of the complete data matrix both locally and globally. Our approach consists of two stages. In the first stage, given the partially observed data matrix, we solve a collection of co-clustering problems to obtain a smooth estimate of the observed data matrix and a filled-in data matrix. Then a weighted distance between pairs of rows and pairs of columns is calculated based on the filled-in data matrix. Multiple weighted distances are computed for different combinations of row and column smoothness. In the second stage, new row and column multi-scale metrics are obtained by taking a weighted average of distances computed across different smoothness scales.

Thus, in our multi-scale distance approach, we estimate a collection of complete matrix approximations of a partially observed data matrix that have been smoothed along their rows and columns to different degrees. Row and column multi-scale metrics are calculated based on the collection of estimated completed matrices to encode the affinities between pairs of rows and columns. This multi-scale metric captures the geometry of the complete data matrix both locally and globally, and encodes the row and column similarities.

2.2.1 Smooth estimates

We use a variation on the co-clustering method proposed in [Chi et al. (2017)] to estimate the complete matrix from a partially observed matrix. To recover the smooth estimate of the incomplete data matrix $\boldsymbol{X}$ along both the rows and columns, we seek a minimizer $\boldsymbol{U}(\gamma_r, \gamma_c)$ of the objective function described below:

$$f(\boldsymbol{U}; \gamma_r, \gamma_c) = \frac{1}{2} ||\mathscr{P}_\Theta(\boldsymbol{X}) - \mathscr{P}_\Theta(\boldsymbol{U})||_{\mathrm{F}}^2 + \gamma_r J_r(\boldsymbol{U}) + \gamma_c J_c(\boldsymbol{U}). \tag{2.1}$$

Here γ_r and γ_c are nonnegative tuning parameters, and $J_r(\boldsymbol{U})$ and $J_c(\boldsymbol{U})$ are regularization terms that impose smoothness in $\boldsymbol{U}$ along its rows and columns. By varying the penalty parameters γ_r and γ_c, we can trade off how well the estimate $\boldsymbol{U}$ agrees with $\boldsymbol{X}$ over the observed indices Θ against how smooth $\boldsymbol{U}$ is along its rows and column. Smaller γ_r and γ_c enforce less smoothness on rows and columns of the data matrix.

Following [Mishne et al. (2019)], we employ the following regularization terms in (2.1)

$$J_r(\boldsymbol{U}) = \sum_{(i,j)\in\mathcal{E}_r} \Omega(||\boldsymbol{U}_{i\cdot} - \boldsymbol{U}_{j\cdot}||_2) \quad \text{and} \quad J_c(\boldsymbol{U}) = \sum_{(i,j)\in\mathcal{E}_c} \Omega(||\boldsymbol{U}_{\cdot i} - \boldsymbol{U}_{\cdot j}||_2),$$

where Ω is a folded concave penalty Fan and Li (2001); Zhang (2010), which will induce sparsity in differences between pairs of rows and pairs of columns in $\boldsymbol{U}$. This sparsity will be useful for determining which $\boldsymbol{U}(\gamma_r, \gamma_c)$ to use in our multi-scale affinities. In this chapter, Ω is an approximate snowflake metric:

$$\Omega(z) = \frac{1}{2}\int_0^z \frac{1}{\sqrt{u}+\epsilon}\, d\,u,$$

where ϵ is a small positive number. As ϵ tends to zero, $\Omega(||\boldsymbol{U}_{i\cdot} - \boldsymbol{U}_{j\cdot}||_2)$ converges to a snowflake metric $d(\boldsymbol{U}_{i\cdot}, \boldsymbol{U}_{j\cdot}) = \sqrt{||\boldsymbol{U}_{i\cdot} - \boldsymbol{U}_{j\cdot}||_2}$. As a result, small differences between rows and columns are penalized significantly more than larger differences. We refer readers to [Mishne et al. (2019)] for more detailed discussion about this choice. The graphs $\mathcal{E}_r$ and $\mathcal{E}_c$ quantify the similarities between pairs of rows and pairs of columns of the data matrix. When the data matrix is fully observed $\mathcal{E}_r$ and $\mathcal{E}_c$ are typically computed using a k-NN graph based on the observed values [von Luxburg (2007)]. Since we do not observe a complete matrix, however, a distance based on the observed values, used in related work for image inpainting [Ram et al. (2013)], is used to calculate the k-NN graph. We use the CO-CLUSTER-MISSING algorithm proposed by Mishne et al. [Mishne et al. (2019)] to solve the minimization problem in (2.1).

2.2.2 Multi-scale affinities

After obtaining a smooth estimate $\boldsymbol{U}(\gamma_r, \gamma_c)$, we fill in the data matrix as $\widetilde{\boldsymbol{X}} = \mathscr{P}_\Theta(\boldsymbol{X}) + \mathscr{P}_{\Theta^c}(\boldsymbol{U}(\gamma_r, \gamma_c))$. We repeat the co-clustering procedure with multiple pairs of parameters (γ_r, γ_c) to encode different scales of the row and column smoothness. Then we leverage those estimates of $\boldsymbol{X}$ to calculate a new multi-scale metric. This metric takes full advantage of the coupling between both modes by taking into account all joint scales of the data as the estimate $\boldsymbol{U}$ is smoothed across rows and columns simultaneously.

Instead of determining an optimal single scale of the solution, namely a single pair of (γ_r, γ_c), we aggregate solutions over a wide range of different scales to better estimate the underlying geometry. This eliminates the need to identify a single "ideal" scale at which to fill in the points, as different points in the matrix may have different optimal scales. We

create a collection of pairs of (γ_r, γ_c) as follows. We first pick small values of γ_r and γ_c. By solving the optimization problem (2.1), each pair of the cost parameters yields a smooth estimate $\boldsymbol{U}(\gamma_r, \gamma_c)$, a filled-in matrix $\widetilde{\boldsymbol{X}}$, and the numbers of distinct row and column clusters denoted by n_r and n_c respectively. We increase γ_r and γ_c along a log-linear scale of $\gamma_r = 2^l$, $\gamma_c = 2^k$, until both n_r and n_c shrink to 1 [Chi and Steinerberger (2019)]. In the end, we obtain the collection $\{\widetilde{\boldsymbol{X}}^{(l,k)}\}_{l,k}$ with each $\widetilde{\boldsymbol{X}}^{(l,k)}$ at different smoothing levels ranging from coarse to fine. Here l and k denote the power of 2 taken for specific row and column cost parameters (γ_c, γ_r) in the solution.

Based on this collection, a new multi-scale metric for both rows and columns using the filled-in matrices at multiple scales is defined. This new metric estimates both local and global geometry of the complete data matrix. Figure 2.1 shows the smooth estimates of the data matrix at different scales for different combinations of γ_r and γ_c. The red and yellow lines represent the ith column and jth column of the matrix respectively. At each joint scale, we calculate the Euclidean distance between columns for the filled-in matrix and weight it by the product of γ_r and γ_c raised to a parameter α:

$$d\left(\widetilde{X}_{\cdot i}^{(l,k)}, \widetilde{X}_{\cdot j}^{(l,k)}\right) = (\gamma_r \gamma_c)^{\alpha} ||\widetilde{X}_{\cdot i}^{(l,k)} - \widetilde{X}_{\cdot j}^{(l,k)}||_2. \tag{2.2}$$

The parameter α can be chosen to emphasize local or global structure. Negative values of α favor local over global structure and positive values of α favor global over local structure. The decision to emphasize local structure over global structure or vice-versa is application dependent.

After solving the joint optimization for multiple pairs from the solution surface at different scales, we obtain the multi-scale distance for pairwise columns by summing over the distances at different joint scales:

$$d_c(i, j) = \sum_{l,k} d\left(\widetilde{X}_{\cdot i}^{(l,k)}, \widetilde{X}_{\cdot j}^{(l,k)}\right). \tag{2.3}$$

As noted earlier, the choice of α depends on the application but typically we will take α to be large in order to emphasize differences at the coarser scales. If two columns are very similar at all smoothing levels, their multi-scale distance will be small. If two columns are different at all but the most smoothed scale, they will be far apart in the multi-scale distance. In this way, small differences between pairs of columns will be washed out, whereas material differences in pairs of columns will persist. The multi-scale distance for pairwise rows is calculated in a similar way. This computed distance matrix adheres to a metric that

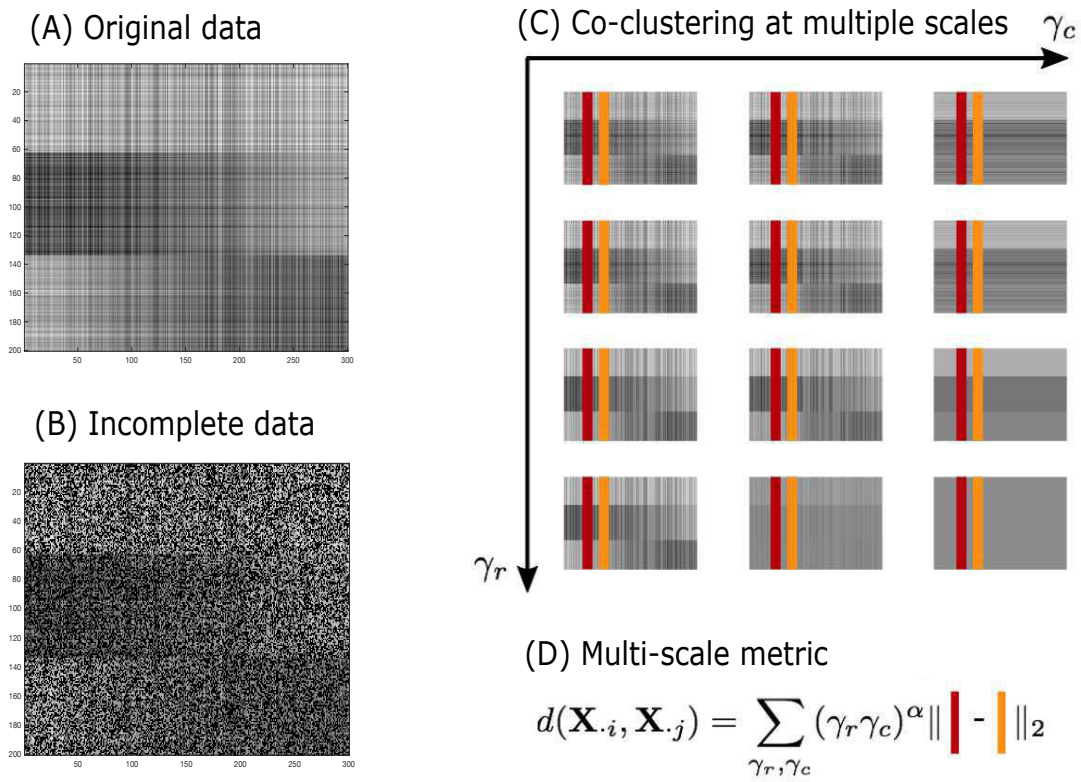


Figure 2.1: Multi-scale affinity calculation in the presence of missing data by leveraging smooth estimates of the data matrix at multiple scales via co-clustering. Ideally, we would have at our disposal a completely observed data matrix (A), but we may instead only have on hand an incompletely observed matrix (B). In this example, the entries are missing completely at random. Given the incomplete data (B), we obtain a collection of complete matrix approximations of the incomplete data matrix by performing co-clustering at multiple scales. The co-clustering problem is posed as an optimization problem in (2.1) with trade-off parameters γ_r and γ_c controlling the smoothness level along rows and columns. Having solved the co-clustering problem for multiple pairs of the trade-off parameters γ_r and γ_c to obtain a collection of smooth estimates, we calculate the multi-scale metric based on those smooth estimates (D). The red and yellow lines represent $\widetilde{X}_{\cdot i}$ and $\widetilde{X}_{\cdot j}$, the ith column and jth column of the matrix respectively. For a given pair of parameters γ_r, γ_c, we calculate pairwise distance between two columns in (2.2) and then aggregate these distances by taking their weighted sum across multiple scales of the smooth estimates (2.3). For pairs of rows the multi-scale metric is computed in an analogous way.

Algorithm 1: Multi-scale Affinities with Missing Data

Input: $\mathscr{E}_r$ and $\mathscr{E}_c$
Set $d_r(i,j)=0$ and $d_c(i,j)=0$;
Set $n_r=m$, $n_c=n$, $k=k_0$ and $l=l_0$
while $n_r > 1$ **do**
  while $n_c > 1$ **do**
    $\left\{U^{(l,k)}, \widetilde{X}^{(l,k)}, n_r, n_c\right\} \leftarrow$ CO-CLUSTER-MISSING$\left(\mathscr{P}_\Theta(X), \gamma_r = 2^l, \gamma_c = 2^k\right)$
    $d(\widetilde{X}_{i\cdot}^{(l,k)}, \widetilde{X}_{j\cdot}^{(l,k)}) \leftarrow (\gamma_r\gamma_c)^\alpha ||\widetilde{X}_{i\cdot}^{(l,k)} - \widetilde{X}_{j\cdot}^{(l,k)}||_2$
    $d(\widetilde{X}_{\cdot i}^{(l,k)}, \widetilde{X}_{\cdot j}^{(l,k)}) \leftarrow (\gamma_r\gamma_c)^\alpha ||\widetilde{X}_{\cdot i}^{(l,k)} - \widetilde{X}_{\cdot j}^{(l,k)}||_2$
    Update row distances: $d_r(i,j)$ += $d(\widetilde{X}_{i\cdot}^{(l,k)}, \widetilde{X}_{j\cdot}^{(l,k)})$
    Update column distances: $d_c(i,j)$ += $d(\widetilde{X}_{\cdot i}^{(l,k)}, \widetilde{X}_{\cdot j}^{(l,k)})$
    $k \leftarrow k+1$
  $l \leftarrow l+1$
Output: $d_r(i,j)$ and $d_c(i,j)$

quantifies data affinities.

Algorithm 1 provides a detailed summary of how multi-scale row and column affinities are computed. For all examples in the chapter, we set l_0 and k_0 to be -6. Figure 2.1 provides a higher level overview of our approach. In many cases, one does not observe a full data matrix (A), but rather an incompletely sampled matrix (B). Smooth estimates of the data matrix at different scales are computed by co-clustering for different combinations of the trade-off parameters γ_r and γ_c. These control the level of row and column smoothing respectively. We construct multi-scale affinities in the presence of missing data (D) by leveraging this collection of smooth estimates of the matrix at multiple scales through formulas (2.2) and (2.3). In our experiments, given a full data matrix we remove a sub-sample of the entries at random.

2.3 Data imputation

Missing data presents a challenge for machine learning algorithms that require completely observed data. Consequently, imputation of missing data is often performed as a preprocessing step for downstream tasks. Commonly used non-parametric imputation methods for missing response values include kernel imputation, which depends on the “distance" between data points. Usually if you have access to good distance measurements, estimation by interpolation is straightforward.

The inverse distance weighting (IDW) method is an interpolation approach to estimate the unknown value at a location using some known values with corresponding weighted values. The IDW method is widely applied because of its low computational cost and easy implementation. The classical IDW is essentially a zeroth-order local Nadaraya–Watson kernel regression [Nadaraya (1964); Watson (1964)] method with an inverse distance weight function. To predict x_{ij} for all $(i,j) \in \Theta^c$, a general form of finding an interpolated value $\hat{x}_{ij}$ at a given location based on observed samples using IDW is given as:

$$\hat{x}_{ij} = \sum_{(s,t)\in\Theta} \frac{v_{ij}(s,t)}{\sum_{(s,t)\in\Theta} v_{ij}(s,t)} x_{st},$$

where $v_{ij} : [n] \times [p] \mapsto \mathbb{R}_+$ and $v_{ij}(s,t) \geq v_{ij}(\widetilde{s},t)$ for all $l \in [p]$ if $d_r(i,s) \leq d_r(i,\widetilde{s})$ and likewise $v_{ij}(s,t) \geq v_{ij}(s,\widetilde{t})$ for all $s \in [n]$ if $d_c(j,t) \leq d_c(j,\widetilde{t})$.

The weight is a function of the distances between pairs of points that measures the similarity between them. The underlying assumption is that data points near the target points carry a larger weight than those further away. Larger weight means the point has a closer relationship to the estimated one and thus should be given more importance. To reflect the correlations and similarities of those data points, it is natural to employ our multi-scale distances in computing weights of IDW. The row and column multi-scale distances can serve to calculate the IDW weights by taking the form:

$$v_{ij}(s,t) = \frac{\exp(-d_r(i,s))}{\sum_{s'=1}^{n} \exp(-d_r(i,s'))} \frac{\exp(-d_c(j,t))}{\sum_{t'=1}^{p} \exp(-d_c(j,t'))}.$$

When the multi-scale distance is smaller, we put more weights on those data points.

IDW has the advantage of being intuitive and is popular for its simplicity, computational speed and good empirical results. We demonstrate the utility of our affinities learned through the multi-scale procedure in imputing missing entries by conducting numerical experiments on different datasets. We compare a simple IDW approach using our multi-scale row and column affinities with the two-directional Laplacian pyramid (2D Pyds) imputation method proposed in [Rabin and Fishelov (2019)], which is also a multi-scale approach based on the pairwise distances between rows and columns of the known matrix. We also include in our comparison standard techniques that replace the missing values in each column by its mean (Mean) and replace the missing values by the most frequent value (Freq).

We follow the simulations in [Rabin and Fishelov (2019)] and test our methods on two

public datasets from the UCI repository (http://archive.ics.uci.edu/ml/datasets). The data is normalized such that each column has mean 0 and standard deviation 1. The Mice Protein Expression Data [Higuera et al. (2015)] contains expression levels of 77 proteins and a total of 1080 measurements per protein. Each measurement can be considered as an independent sample/mouse. While mice of the same class may have similar protein expression levels, at the same time similar protein expression levels are likely to be in the same class. The original dataset has many missing values. We extract a smaller, complete dataset $\boldsymbol{X}$, of size $M \times N = 1000 \times 66$ from the original data in order to evaluate the results. The voice rehabilitation dataset [Tsanas et al. (2014)] contains data for 126 patients and 309 features. Each feature corresponds to the application of a speech signal processing algorithm including wavelet-based, frequency-based, and non-linear time-series algorithms that aim to objectively characterize the signal. Consequently, there is likely correlation between rows of this dataset as well as between its columns [Rabin and Fishelov (2019)].

Table 2.1 and Table 2.2 summarize the imputation performance measured by the average root mean square errors (RMSE) for ten replicates of the four methods. Results for 2D Pyds, Mean and Freq are reported in Rabin and Fishelov (2019). IDW refers to the proposed procedure of estimating missing entries by IDW with weights derived from the multi-scale distances. The performances are evaluated under different settings with 20%, 50% and 80% missing entries, respectively. For each mode, each method is repeated ten times and each time the missing data locations are chosen at random. The average RMSE for the ten replicates is computed. We observe that our method outperforms the multi-scale 2D Pyds approach in terms of reconstruction errors. By solving the joint optimization for both rows and columns, this multi-scale distance accounts for the coupling connections found in the data, and thus provides a good formulation of those weights.

Note that the simple IDW method can suffer from an underestimation bias when the missing percentage is high, as seen in Table 2.1 when the missing fraction is 80%. Since missing entries are computed as an average over observed values in IDW, by construction all imputed values will always lie within the range of the observed data. In short, IDW can never impute a value with a magnitude larger than that observed within the data. Despite the intrinsic limitation of IDW, our emphasis here is that once we obtain good affinities, a simple method such as IDW can still have good performance. These affinity weights can be also employed in other more sophisticated imputation methods for future work.

Table 2.1: RMSE for the Mice dataset

% missing	IDW	2D Pyds	Mean	Freq
20%	0.3596	0.381	1.0024	3.0918
50%	0.4392	0.5198	0.9999	3.0890
80%	0.9341	0.7697	1.0028	2.8001

Table 2.2: RMSE for the Voice dataset

% missing	IDW	2D Pyds	Mean	Freq
20%	0.5872	0.7586	0.9952	3.2704
50%	0.6753	0.8207	1.0086	2.9594
80%	0.8840	0.9002	1.0205	2.2059

2.4 Clustering

Clustering is the task of dividing a collection of objects into groups so that objects in the same group are more similar to each other than to those in other groups. The quality of clustering relies on the similarity criterion between points. Missing values can complicate the application of clustering algorithms as similarity criteria are usually computed between completely observed data points. To deal with missing values in the context of clustering, it is a common practice to impute the missing values first and then apply the clustering algorithm on the completed data [Boluki et al. (2019)].

The multi-scale affinity approach learns the underlying geometry that can be exploited to impute missing values. Depending on the downstream task for which weights are used, the best single pair of parameters for one task may not be optimal for another. Specifically, the optimal pair of cost parameters at a single scale for imputation and clustering are not necessarily the same. Moreover, it is unclear whether a single ideal scale exists. Our multi-scale approach enjoys the property of not requiring the identification of the ideal scale at which to fill in missing values. Consequently, our multi-scale approach eliminates the need for extensively tuning and picking a single pair of the cost parameters and can fully take advantage of the estimates at different smoothing levels.

Lung500 is a real-world dataset that contains 56 lung cancer patients and their gene expressions across 500 genes with the greatest sample variance from the original collection of 12,625 genes [Lee et al. (2010)]. Patients belong to one of four subgroups; they are either normal subjects (Normal) or have been diagnosed with one of three types of cancers: pul-

monary carcinoid tumors (Carcinoid), colon metastases (Colon), and small cell carcinoma (Small Cell).

Figure 2.2 shows the Adjusted Rand Index (ARI) [Hubert and Arabie (1985)] with respect to the ground-truth labels of the 4 cancer types for the Lung500 dataset, comparing our approach to competing methods. The ARI measures the agreement between the clustering results and the ground truth labels in a way that higher values indicate better clustering quality. Four techniques are used to process data of different missing percentages. We evaluate the clustering result using k-means on data imputed by IDW using the multi-scale affinity weights, as well as k-means on the co-manifold embeddings [Mishne et al. (2019)] for lower dimension representation based on these multi-scale distances. For the purpose of clustering, rows and columns are more smoothed in IDW compared with the imputation task. The k-POD algorithm [Chi et al. (2016)] is a method of performing k-means clustering on partially observed data. It identifies a cluster that is in accord with the observed data even when the missingness mechanism is unknown and when external information is unavailable. Fast Robust PCA on Graphs [Shahid et al. (2016)] (FRPCAG) is a fast dimensionality reduction algorithm for mining clusters from high dimensional and large low-rank datasets. It also introduces graph smoothness on both rows and columns of the data matrix and handles corruption in the data. FRPCAG targets an approximate recovery of low-rank signals that exploits the linear coupled geometry and may fail when the data lie on a nonlinear manifold or suffers from high percentage of missing values, as it assumes these are sparse in the data.

The multi-scale metric takes the coupled structure of the genes and the samples into account and gives the best clustering result among those methods. When data is not terribly corrupted, the performance is unaffected by increasing the missing fractions of data. FRPCAG aims to recover an approximate low-rank matrix with dual-graph regularization and requires tuning the cost parameters for row graph and column graph. However, it did not address the problem of how to choose those regularization parameters and it targets solely the recovery of the low-rank matrix which makes data-driven approaches to search for optimal parameters infeasible. We pick two choices of cost parameters ($\gamma = 0.01$ and $\gamma = 1$) and observed they have similar performance and observe that FRPCAG's performance is more affected by data corruption. In contrast, our approach eliminates this parameter selection step, and our approach's performance begins to significantly degrade only at 90% missing values.

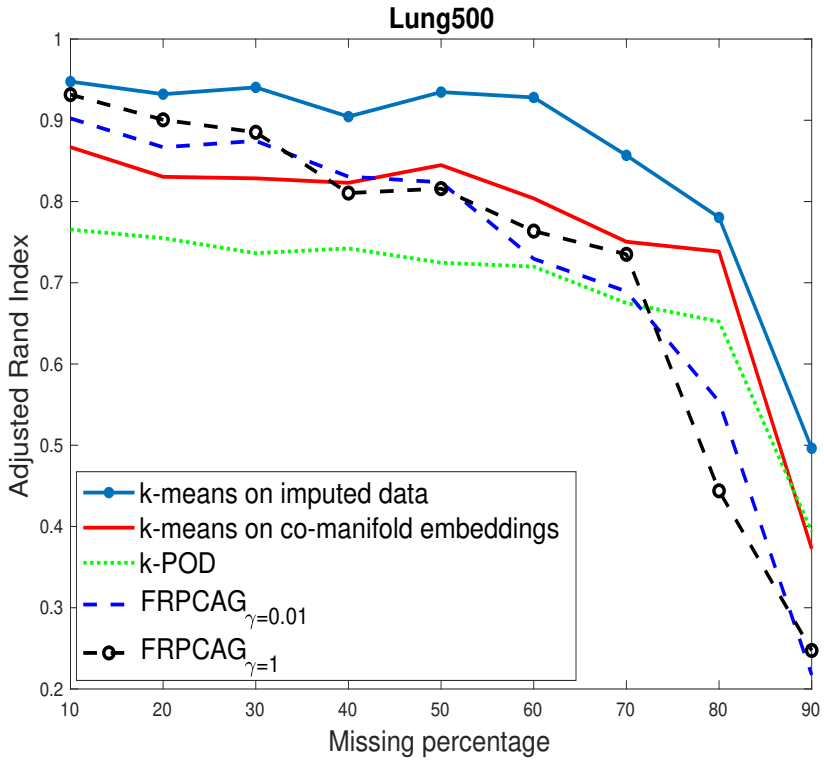


Figure 2.2: Comparing k-means clustering applied to data processed by four techniques. Higher ARI indicates better agreement between two clusterings.

2.5 Matrix completion

The goal of matrix completion is to estimate missing entries of a partially observed matrix, a task that bears some similarity to missing data imputation. Indeed, the matrix completion problem can be cast as a special case of the missing data analysis (MDA) problem [Chi and Li (2019)] where one of the inference tasks in MDA is missing data imputation. While matrix completion shares some goals with missing data imputation, matrix completion problems differs from standard missing data imputation problems in nontrivial ways. The MDA problem assumes more general models and the missing mechanism can be more complex. While the matrix completion problem assumes data to be missing completely at random (MCAR), the missing mechanism can depend on the data in the MDA problem. The missing proportion in matrix completion is significantly higher than that in MDA. When no missing data is present, the MDA problem becomes the standard problem with repeated measurements for the same model parameters. In matrix completion, we seek to complete the missing entries of a partially observed matrix with one sample for each observation. The problem of recovering the full matrix from incomplete observation, however, is ill-posed and underdetermined without any assumptions or restrictions on the completed matrix. The most common assumption is that the unknown matrix is low-rank or approximately low-rank.

Candès and Recht [Candès and Recht (2009)] proved that most low-rank matrix matrices can be completed accurately with high probability by solving a convex optimization problem. Mazumder et al. [Mazumder et al. (2010)] considered the scenarios when the observations are noisy and proposed the softImpute algorithm using convex relaxation techniques to solve a nuclear norm regularized problem. It is pointed out in [Kalofolias et al. (2014)] that the standard low-rank matrix recovery problem can be further improved by using similarity information about rows and columns. They borrow ideas from the field of manifold learning and force the solution to be smooth on the manifolds of users and movies through graph regularizations. A similar idea was exploited in [Chi et al. (2018)] and they considered the problem of performing matrix completion with side information on row-by-row and column-by-column similarities under a structural assumption that is closely related to the low-rank assumption.

In this section, we focus on the matrix completion on graphs (MCG) model, where the row and column structures are simultaneously taken into consideration. In [Kalofolias et al. (2014)], the authors show that the standard low-rank matrix recovery problem can be further improved using similarity information about rows and columns. We evaluate the

proximity structure encoded in the multi-scale metric and show the effectiveness of these row and column affinities when data is missing.

Let $\boldsymbol{Z} \in \mathbb{R}^{n\times p}$ be the matrix that we want to recover. The MCG problem is formulated as:

$$\min \frac{1}{2}||\mathscr{P}_\Theta(\boldsymbol{X})-\mathscr{P}_\Theta(\boldsymbol{Z})||_{\mathrm{F}}^2+\gamma_n||\boldsymbol{Z}||_*+\frac{\gamma_r}{2}\mathrm{tr}(\boldsymbol{Z}\boldsymbol{L}_r\boldsymbol{Z})+\frac{\gamma_c}{2}\mathrm{tr}(\boldsymbol{Z}\boldsymbol{L}_c\boldsymbol{Z}), \tag{2.4}$$

where $\boldsymbol{L}_r$ is the Laplacian of the row graph given by

$$l_{ij}^r = \begin{cases} \sum\limits_{(i,i')\in\mathscr{E}_r} w_{ii'}^r & \text{if } i=j \\ -w_{ij}^r & \text{otherwise,} \end{cases}$$

and $\boldsymbol{L}_c$ is the Laplacian of the column graph defined in the same way. If γ_r and γ_c are both 0, problem (2.4) solves the same problem as in [Mazumder et al. (2010)]. If γ_n is 0, problem (2.4) reduces to the biclustered matrix completion (BMC) problem in [Chi et al. (2018)].

The biggest challenge in [Kalofolias et al. (2014)] is to construct the graphs of rows and columns that well represent the similarity structure in the presence of missing data. The multi-scale distances encode coupling proximity information about rows and columns, and this information can be taken advantage of by introducing structures via graphs. Row and column graph Laplacians based on the row and column multi-scale distance are constructed by graph algorithms such as k-NN graph and passed into the MCG algorithm. In this way, we incorporate the additional row and column structures into the matrix completion problem.

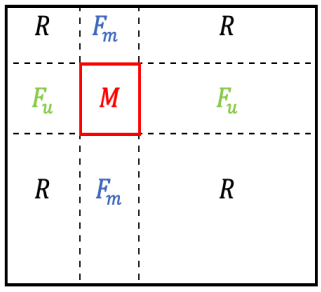


Figure 2.3: For the Movielens 10M dataset, in original MCG model graphs are constructed by leveraging prior information about $\boldsymbol{F_u}$ and $\boldsymbol{F_m}$. The blocks F_u and F_m are used to construct the movie and user graphs. The submatrix $\boldsymbol{M}$ of $\boldsymbol{A}$ is used for training and testing.

The MovieLens 10M dataset [Harper and Konstan (2015)] contains ratings ('stars') from

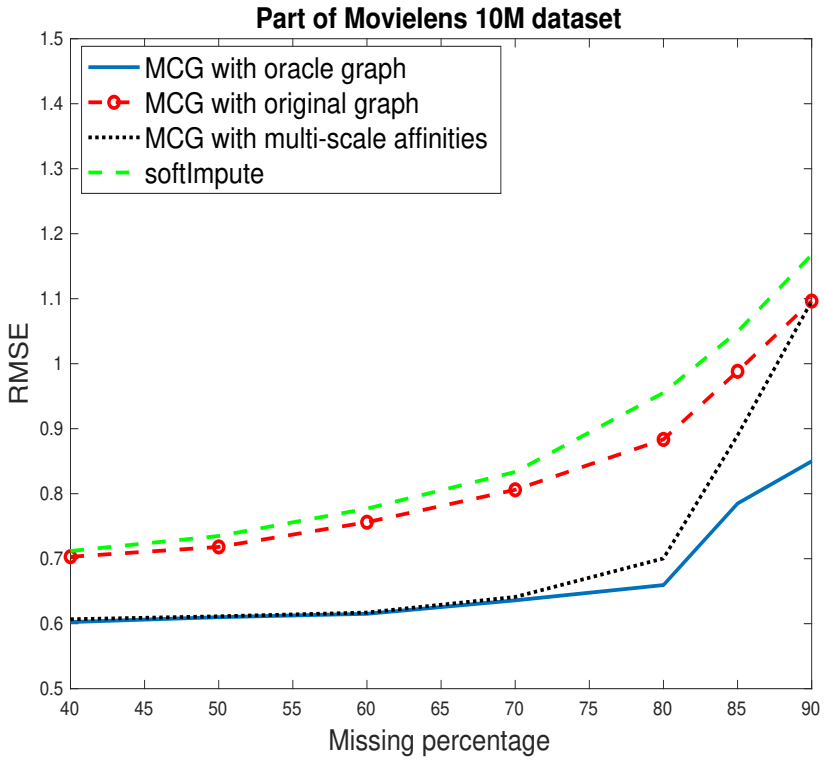


Figure 2.4: Reconstruction error of experiments on the complete subset of Movielens 10M. MCG with oracle graph means generating graphs from the multi-scale affinities algorithm that is initialized by the oracle graph. MCG with oracle graph is the procedure mentioned in [Kalofolias et al. (2014)] where additional information about users and movies are used to build the graph. MCG with multi-scale affinities uses the k-NN graphs computed from the multi-scale affinity matrix. softImpute solves the problem using only nuclear norm regularization.

1 to 5 (increments of 1) given by 71,567 users for 10,677 movies. In the original MCG paper [Kalofolias et al. (2014)], the authors use information *outside* of the subset matrix as features to construct the row and column graphs. Figure 2.3 illustrates how ratings outside of the complete submatrix are used as features to construct the column and row graphs. The rows (users) and columns (movies) are sorted by order of increasing sampling frequency. After a row and column permutation, the original MovieLens 10M matrix $\boldsymbol{A}$ is partitioned in blocks $A = [\boldsymbol{M}, \boldsymbol{F_u}; \boldsymbol{F_m}, \boldsymbol{R}]$, where $\boldsymbol{M}$ is the 100×200 complete data matrix we use in the experiment, and $\boldsymbol{F_u}$ is used as the users feature matrix and $\boldsymbol{F_m}$ is used as the movies feature matrix. For comparison, we consider MCG using row and column graphs constructed using $\boldsymbol{F_u}$ and $\boldsymbol{F_m}$ as described in [Kalofolias et al. (2014)].

Figure 2.4 shows the prediction error as a function of missing percentage for softImpute [Mazumder et al. (2010)], which leverages only low-rank structure, and MCG using three different ways of generating graphs: oracle graphs, graphs described in [Kalofolias et al. (2014)], and graphs constructed using multi-scale affinities. Although different methods might call for slightly different graph parameters for optimal results, for the given dataset, we use the same graph parameters for those methods to ensure a fair comparison. The oracle graphs are computed as the k-NN graphs based on true complete data. We see the multi-scale affinities can be used to construct graphs that capture nearly as much coupling similarity structure along both rows and columns as the oracle graphs even when a large fraction of the entries are missing.

2.6 Conclusion

In this chapter, we presented a new method for learning pairwise affinities of both the rows and columns of a matrix with missing data. We seek a collection of estimated complete matrices at multiple scales of smoothness by solving a family of optimization problems with different regularization parameters, which encode a smoothness scale of the estimate along the rows and columns. We combine these multi-scale estimates into a new metric that captures the joint row and column geometry of the complete data matrix and represents similarities among rows and columns when data is partially observed. It can serve as affinity weights in many applications even when data are incomplete. The metric presented in this chapter is general and may be adapted to other tasks. In future work, we can further broaden the scope of this framework by extending it to more applications.

CHAPTER

3

A SCALABLE ALGORITHM FOR ESTIMATING A LOW-RANK PLUS SPARSE MATRIX

3.1 Introduction

The problem of low-rank plus sparse matrix decomposition for big data arises in many applications such as background extraction in video surveillance [Min et al. (2010)], shadow and specularity removal from face images [Candès et al. (2011); Zhang et al. (2013)], and link prediction in social networks [Richard et al. (2012)]. Recovery of low-rank and sparse components of a given matrix is a nondeterministic polynomial-time hard (NP-hard) task. It is proved that this problem can be solved via heuristically solving a convex relaxation problem where the nuclear norm and ℓ_1 norm are utilized to induce low-rank and sparsity [Candès et al. (2011)]. The nuclear norm, which is the sum of singular values of a matrix, is an effective surrogate in rank minimization problems [Recht et al. (2010)]. The ℓ_1 norm, which is the sum of magnitudes, has been well studied as a surrogate for counting the

number of non-zero entries for finding sparse representations of signals. In theory, under certain conditions, the convex relaxation problem can be solved in polynomial time.

One representative example is the Robust Principal Component Analysis, where the goal is to recover a low-rank matrix L_0 from grossly corrupted observations $Y = L_0 + S_0$. This task can be solved both theoretically and numerically via a convex relaxation problem, called Compressive Principal Component Pursuit (CPCP) [Wright et al. (2012)]:

$$\min_{S,L} \frac{1}{2}\|Y - S - L\|_F^2 + \lambda_S \|S\|_1 + \lambda_L \|L\|_*. \tag{3.1}$$

Here λ_S and λ_L are regularization parameters. Inspired by these theoretical results, many provably convergent algorithms are developed to solve PCP and its variants [Aybat et al. (2011); Lin et al. (2010, 2009); Aybat et al. (2014); Tao and Yuan (2011); Yuan and Yang (2013); Beck and Teboulle (2009)]. These algorithms exploit first-order methods that involve the proximal operator of the nuclear norm and require conducting the singular value decomposition (SVD). In each iteration, the dominant cost is computing an SVD of the same size as the input data, which has a superlinear cost.

The need to compute a sequence of full or partial SVD severely limits the applicability of these algorithms to large-scale problems. Motivated by this, a more scalable algorithm called Frank-Wolfe Thresholding (FW-T) that leverages Frank-Wolfe iterations and proximal steps is proposed in [Mu et al. (2016)]. It computes only a rank-one SVD in each iteration. Thus, it is more scalable than proximal-based first-order methods such as ISTA and FISTA [Beck and Teboulle (2009)] and has much faster practical convergence than the Frank-Wolfe method.

Although the per iteration cost of FW-T is significantly cheaper because of the rank-one update, it requires more iterations to reach an accurate solution due to the limitation of the Frank-Wolfe method. Furthermore, the computation of SVD is still a bottleneck in large datasets. To remedy this, we introduce the low-rank bi-factorization structure and characterize the nuclear norm in terms of the Frobenius norm. With this low-rank bi-factorization, we develop a more scalable algorithm that avoids the computational burden of SVD. It has a faster convergence speed in practice and the per iteration cost is greatly reduced.

The rest of the chapter is organized as follows. In section 3.2, we provide some preliminaries that are useful for the subsequent analysis. In section 3.3, we reform the problem into a more structured form, which inspires our algorithm to be developed, and present our algorithm. In section 3.4, we establish the convergence property of the algorithm. Numerical

results are reported in section 3.5 to verify the justification of the model and the efficiency of the proposing algorithm. In section 3.6, we make some conclusions and discuss some topics for future work.

3.2 Preliminaries

3.2.1 Notations

Throughout the chapter, scalar variables, column vectors, matrices, and sets are denoted by lower-case letters, italic boldface lower-case letters, upper-case letters, and calligraphy letters, respectively. For example, x is a scalar variable, $\boldsymbol{x}$ is a vector, $\boldsymbol{X}$ is a matrix and $\mathscr{X}$ is a set. The elements of vectors and matrices are denoted by subscripts. For a matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$, vec($\boldsymbol{A}$) refers to the vector in $\mathbb{R}^{mn}$ obtained by stacking the columns of the matrix $\boldsymbol{A}$ on bottom of one another.

3.2.2 Frank-Wolfe method

The Frank-Wolfe (FW) method [Frank and Wolfe (1956)] is a conditional gradient method [Levitin and Polyak (1966)] that can be applied to solve the general problem of minimizing a differentiable convex function h over a compact, convex domain $\mathscr{D} \in \mathbb{R}^n$:

$$\text{minimize } h(\boldsymbol{x}) \quad \text{subject to } \boldsymbol{x} \in \mathscr{D}.$$

Here, ∇h is assumed to be L-Lipschitz continuous. A general version of Frank-Wolfe with updating scheme that converges at a rate of $\mathscr{O}(1/k)$ [Mu et al. (2016)] is given in Algorithm 2. At a current position $\boldsymbol{x}$, the algorithm considers the linearization of the objective function and moves towards a minimizer of this linear function (taken over the same domain).

Algorithm 2: Frank-Wolfe method with general updating scheme

Input: $\boldsymbol{x}^0 \in \mathscr{D}, T$
for $k = 0, 1, 2, \ldots, T$ **do**
  $\boldsymbol{v}^k \in \operatorname{argmin}_{\boldsymbol{v}\in\mathscr{D}} \langle \boldsymbol{v}, \nabla h(\boldsymbol{x}^k)\rangle$;
  $\gamma = \frac{2}{k+2}$;
  Update $\boldsymbol{x}^{k+1} \in \mathscr{D}$ such that $h(\boldsymbol{x}^{k+1}) \leq h(\boldsymbol{x}^k + \gamma(\boldsymbol{v}^k - \boldsymbol{x}^k))$

3.2.3 L_1 optimization using Hadamard product parametrization

The problem of finding the Lasso estimator is shown to be equivalent to a double ridge regression by Hadamard product reparametrization(HPP)[Hoff (2017)]. Consider the estimation for the normal linear regression model $\boldsymbol{y} \sim N_n(\boldsymbol{X}\boldsymbol{\beta}, \sigma_2 \boldsymbol{I})$, where $\boldsymbol{X} \in \mathbb{R}^{n\times p}$ is a matrix of predictor variables and $\boldsymbol{\beta} \in \mathbb{R}^p$ is the vector of regression coefficients to be estimated. By introducing $\boldsymbol{Q} = \boldsymbol{X}\boldsymbol{X}^T$ and $\boldsymbol{l} = \boldsymbol{X}^T\boldsymbol{y}$, the least-squares estimate is the minimizer of $h(\boldsymbol{\beta}) = \boldsymbol{\beta}^T\boldsymbol{Q}\boldsymbol{\beta} - 2\boldsymbol{\beta}^T\boldsymbol{l}$. The lasso estimate is the minimizer of $f(\boldsymbol{\beta}) = \boldsymbol{\beta}^T\boldsymbol{Q}\boldsymbol{\beta} - 2\boldsymbol{\beta}^T\boldsymbol{l} + \lambda\|\boldsymbol{\beta}\|_1$. We reparametrize the model so that $\boldsymbol{\beta} = \boldsymbol{u} \circ \boldsymbol{v}$, where $\circ$ is the Hadamard (element-wise) product.

Lemma 3.2.1. *Let* $f(\boldsymbol{\beta}) = h(\boldsymbol{\beta}) + \lambda\|\boldsymbol{\beta}\|_1$ *and* $g(\boldsymbol{u}, \boldsymbol{v}) = h(\boldsymbol{u} \circ \boldsymbol{v}) + \lambda(\boldsymbol{u}^T\boldsymbol{u} + \boldsymbol{v}^T\boldsymbol{v})/2$. *Then*
1. $\inf_{\boldsymbol{\beta}} f(\boldsymbol{\beta}) = \inf_{\boldsymbol{u},\boldsymbol{v}} g(\boldsymbol{u}, \boldsymbol{v})$;
2. if $(\hat{\boldsymbol{u}}, \hat{\boldsymbol{v}})$ *is a local minimum of* g, *then* $\hat{\boldsymbol{\beta}} = \hat{\boldsymbol{u}} \circ \hat{\boldsymbol{v}}$ *is a local minimum of* f.

Since all local minimizers of f are global minimizers [Tibshirani et al. (2013)], any local minimizer $(\hat{\boldsymbol{u}}, \hat{\boldsymbol{v}})$ of g provides a global optimizer $\hat{\boldsymbol{\beta}} = \hat{\boldsymbol{u}} \circ \hat{\boldsymbol{v}}$ of the lasso objective function. In other words, lasso estimates can be obtained from local minimizers of g.

3.3 Methods

Problem (3.1) has a non-differentiable objective function and an unbounded feasible set. To apply the Frank-Wolfe method, a two-step reformulation is exploited. First, the epigraph reformulation [Harchaoui et al. (2015)] of Problem (3.1) is obtained by introducing auxiliary variables t_L and t_S. Then a trivial feasible solution to the epigraph reformulation is chosen and used as an upper bound on the optimal values. Thus we could apply the Frank-Wolfe method to solve the equivalent problem:

$$\begin{aligned} &\min g(\boldsymbol{L}, \boldsymbol{S}, t_L, t_S) = \frac{1}{2}\|\boldsymbol{L} + \boldsymbol{S} - \boldsymbol{Y}\|_F^2 + \lambda_L t_L + \lambda_S t_S \\ &s.t. \|\boldsymbol{L}\|_* \le t_L \le U_L, \|\boldsymbol{S}\|_1 \le t_S \le U_S. \end{aligned} \tag{3.2}$$

By applying the Frank-Wolfe method directly to the reformulated Problem (3.2), linear per-iteration cost and an $\mathcal{O}(1/k)$ convergence rate in the function value can be achieved [Mu et al. (2016)]. However, it updates the sparse term one element at a time, so it converges very slowly on typical numerical examples. To resolve this issue, a new algorithm called Frank-Wolfe-Thresholding (FW-T) that combines a modified Frank-Wolfe step with a proximal

gradient step is introduced in [Mu et al. (2016)]. The FW-T method essentially exploits the Frank-Wolfe step to handle the nuclear norm and a proximal gradient step to handle the l_1-norm, while keeping iteration cost low and retaining convergence guarantees.

Although the FW-T algorithm is efficient at the beginning, it becomes less competitive in reaching solutions of high accuracy due to the nature of Frank-Wolfe. Since the FW-T method only conducts a rank-one update on the low-rank component in each iteration, it requires more iterations to reach an accurate solution [Mu et al. (2016)]. To achieve a primal convergence result that only weakly depends on the tightness of U_L and U_S, the exact line search updates the iterate by solving a two-dimensional quadratic, which has a polynomial cost. Although the Frank-Wolfe step to solve for D_L and D_S has explicit forms, the nuclear norm optimization requires a singular value decomposition (SVD) of a $m \times n$ matrix at each iteration, and the cost is $\mathcal{O}(n^3)$ per iteration if m and n are of the same order. This decomposition prohibits the computation in large-scale applications.

To address these computational problems in practice, we specify the low-rank matrix to have an explicit low-rank bi-factorization structure such that $\boldsymbol{L} = \boldsymbol{W}\boldsymbol{Z}$, where $\boldsymbol{W} \in \mathbb{R}^{m\times r}, \boldsymbol{Z} \in \mathbb{R}^{r\times n}$. This formulation yields a much smaller number of $r(m+n)$ parameters to be estimated instead of $m\times n$ parameters when $r \ll \min(m,n)$. A variational form [Majindar (1963)] of the nuclear norm through this bi-factorization structure can be exploited. The following lemma allows us to characterize the nuclear norm in terms of the Frobenius norm.

Lemma 3.3.1. $\|\boldsymbol{X}\|_* = \min_{\boldsymbol{X}=\boldsymbol{U}\boldsymbol{V}^T} \|\boldsymbol{U}\|_F \|\boldsymbol{V}\|_F = \min_{\boldsymbol{X}=\boldsymbol{U}\boldsymbol{V}^T} \frac{1}{2}\left(\|\boldsymbol{U}\|_F^2 + \|\boldsymbol{V}\|_F^2\right)$

We propose to solve this equivalent problem:

$$\min_{\boldsymbol{S},\boldsymbol{W},\boldsymbol{Z}} f(\boldsymbol{S},\boldsymbol{W},\boldsymbol{Z}) = \frac{1}{2}\|\boldsymbol{Y}-\boldsymbol{S}-\boldsymbol{W}\boldsymbol{Z}\|_F^2 + \lambda_S\|\boldsymbol{S}\|_1 + \frac{\lambda_L}{2}\left[\|\boldsymbol{W}\|_F^2 + \|\boldsymbol{Z}\|_F^2\right]. \tag{3.3}$$

The original formulation (3.1) is convex. While the downside of solving the above problem is its non-convexity, the advantage is that the optimization problem (3.3) above is differentiable. Besides, the objective function is convex in one variable when the other two variables are fixed. For example, it is convex in $\boldsymbol{S}$ when $(\boldsymbol{W},\boldsymbol{Z})$ are fixed. Thus, we could use an alternating updating procedure to update the variables:

$$\boldsymbol{S}^{k+1} = \arg\min_{\boldsymbol{S}} f(\boldsymbol{S}, \boldsymbol{W}^k, \boldsymbol{Z}^k) = \arg\min \frac{1}{2}\|\boldsymbol{Y} - \boldsymbol{W}^k\boldsymbol{Z}^k - \boldsymbol{S}\|_F^2 + \lambda_S\|\boldsymbol{S}\|_1$$
$$\boldsymbol{W}^{k+1} = \arg\min_{\boldsymbol{W}} f(\boldsymbol{S}^{k+1}, \boldsymbol{W}, \boldsymbol{Z}^k) = \arg\min \frac{1}{2}\|\boldsymbol{Y} - \boldsymbol{S}^{k+1} - \boldsymbol{W}\boldsymbol{Z}^k\|_F^2 + \frac{\lambda_L}{2}\|\boldsymbol{W}\|_F^2$$
$$\boldsymbol{Z}^{k+1} = \arg\min_{\boldsymbol{Z}} f(\boldsymbol{S}^{k+1}, \boldsymbol{W}^{k+1}, \boldsymbol{Z}) = \arg\min \frac{1}{2}\|\boldsymbol{Y} - \boldsymbol{S}^{k+1} - \boldsymbol{W}^{k+1}\boldsymbol{Z}\|_F^2 + \frac{\lambda_L}{2}\|\boldsymbol{Z}\|_F^2$$

Through this formulation, at each iteration, we can have an explicit form to update those variables. $\boldsymbol{S}^{k+1}$ can be obtained by a soft-thresholding operator:

$$\boldsymbol{S}^{k+1} = \mathscr{T}_{\lambda_S}(\boldsymbol{Y} - \boldsymbol{W}^k\boldsymbol{Z}^k).$$

For $\boldsymbol{W}$ and $\boldsymbol{Z}$, we can take derivatives of the objective function and obtain the minimizers by solving the linear systems:

$$\boldsymbol{W}(\boldsymbol{Z}^k\boldsymbol{Z}^{k\,T} + \lambda_L\boldsymbol{I}) = (\boldsymbol{Y} - \boldsymbol{S}^k)\boldsymbol{Z}^{k\,T}$$
$$(\boldsymbol{W}^{k+1\,T}\boldsymbol{W}^{k+1} + \lambda_L\boldsymbol{I})\boldsymbol{Z} = \boldsymbol{W}^{k+1\,T}(\boldsymbol{Y} - \boldsymbol{S}^{k+1}).$$

The resulting algorithm is presented as Algorithm 3 below. We call this method the LBF algorithm as it uses the low-rank bi-factorization structure.

Algorithm 3: LBF method for Problem (3.3)

Input: $\boldsymbol{U}^0, \boldsymbol{V}^0, \boldsymbol{W}^0, \boldsymbol{Z}^0, T$
for $k = 0, 1, 2, ..., T$ **do**
  $\boldsymbol{S}^{k+1} = \mathscr{T}_{\lambda_S}(\boldsymbol{Y} - \boldsymbol{W}^k\boldsymbol{Z}^k)$
  $\boldsymbol{W}^{k+1} = (\boldsymbol{Y} - \boldsymbol{S}^k)\boldsymbol{Z}^{k\,T}(\boldsymbol{Z}^k\boldsymbol{Z}^{k\,T} + \lambda_L\boldsymbol{I})^{-1}$
  $\boldsymbol{Z}^{k+1} = (\boldsymbol{W}^{k+1\,T}\boldsymbol{W}^{k+1} + \lambda_L\boldsymbol{I})^{-1}\boldsymbol{W}^{k+1\,T}(\boldsymbol{Y} - \boldsymbol{S}^{k+1})$

To solve the linear system $(\boldsymbol{W}^T\boldsymbol{W} + \lambda_L\boldsymbol{I})\boldsymbol{Z} = \boldsymbol{W}^T(\boldsymbol{Y} - \boldsymbol{S})$, using the QR decomposition will take $\mathscr{O}(r^3)$ where $(\boldsymbol{W}^T\boldsymbol{W} + \lambda_L\boldsymbol{I}) = \boldsymbol{Q}\boldsymbol{R}$ and $\boldsymbol{Q} \in \mathbb{R}^{r\times r}$, $\boldsymbol{Q}^T\boldsymbol{Q} = \boldsymbol{I}$ and $\boldsymbol{R} \in \mathbb{R}^{r\times r}$ is triangular. The linear system to be solved becomes $\boldsymbol{Q}\boldsymbol{R}\boldsymbol{Z} = \boldsymbol{W}^T(\boldsymbol{Y} - \boldsymbol{S})$. Then we can obtain $\boldsymbol{Z}$ by calculating $\boldsymbol{Z} = \boldsymbol{R}^{-1}\boldsymbol{Q}^T\boldsymbol{W}^T(\boldsymbol{Y} - \boldsymbol{S})$, which takes $\mathscr{O}(r^2) + \mathscr{O}(mr^2) + \mathscr{O}(mnr)$ and can be written as $\mathscr{O}(r^2) + \mathscr{O}(mnr)$. Similarly, solving $\boldsymbol{W}(\boldsymbol{Z}\boldsymbol{Z}^T + \lambda\boldsymbol{I}) = (\boldsymbol{Y} - \boldsymbol{S})\boldsymbol{Z}^T$ takes $\mathscr{O}(r^3) + \mathscr{O}(mnr)$.

In addition to the variational form of the nuclear norm, the Hadamard product parametrization(HPP) technique is developed [Hoff (2017)] for obtaining L_1-penalized regression estimates. We propose to solve this equivalent problem:

$$f(\boldsymbol{U},\boldsymbol{V},\boldsymbol{W},\boldsymbol{Z})=\frac{1}{2}\|\boldsymbol{Y}-\boldsymbol{U}\circ\boldsymbol{V}-\boldsymbol{W}\boldsymbol{Z}\|_F^2+\frac{\lambda_S}{2}\left[\|\boldsymbol{U}\|_F^2+\|\boldsymbol{V}\|_F^2\right]+\frac{\lambda_L}{2}\left[\|\boldsymbol{W}\|_F^2+\|\boldsymbol{Z}\|_F^2\right] \quad (3.4)$$

The optimization problem in (3.4) can be solved through an alternating procedure by coordinate descent as well. We start with an initial guess $(\boldsymbol{U}^0,\boldsymbol{V}^0,\boldsymbol{W}^0,\boldsymbol{Z}^0)$ and repeat until convergence for $k=0,1,2,\ldots,$:

$$\begin{aligned}
\boldsymbol{U}^{k+1}&=\arg\min_{\boldsymbol{U}}\frac{1}{2}\|\boldsymbol{Y}-\boldsymbol{U}\circ\boldsymbol{V}^k-\boldsymbol{W}^k\boldsymbol{Z}^k\|_F^2+\frac{\lambda_S}{2}\|\boldsymbol{U}\|_F^2,\\
\boldsymbol{V}^{k+1}&=\arg\min_{\boldsymbol{V}}\frac{1}{2}\|\boldsymbol{Y}-\boldsymbol{U}^{k+1}\circ\boldsymbol{V}-\boldsymbol{W}^k\boldsymbol{Z}^k\|_F^2+\frac{\lambda_S}{2}\|\boldsymbol{V}\|_F^2,\\
\boldsymbol{W}^{k+1}&=\arg\min_{\boldsymbol{W}}\frac{1}{2}\|\boldsymbol{Y}-\boldsymbol{U}^{k+1}\circ\boldsymbol{V}^{k+1}-\boldsymbol{W}\boldsymbol{Z}^k\|_F^2+\frac{\lambda_L}{2}\|\boldsymbol{W}\|_F^2,\\
\boldsymbol{Z}^{k+1}&=\arg\min_{\boldsymbol{Z}}\frac{1}{2}\|\boldsymbol{Y}-\boldsymbol{U}^{k+1}\circ\boldsymbol{V}^{k+1}-\boldsymbol{W}^{k+1}\boldsymbol{Z}\|_F^2+\frac{\lambda_L}{2}\|\boldsymbol{Z}\|_F^2.
\end{aligned}$$

The objective function in each block is differentiable. Similar to Algorithm 3, $\boldsymbol{W}$ and $\boldsymbol{Z}$ can be updated via solving linear systems $\boldsymbol{W}(\boldsymbol{Z}\boldsymbol{Z}^T+\lambda\boldsymbol{I})=(\boldsymbol{Y}-\boldsymbol{U}\circ\boldsymbol{V})\boldsymbol{Z}^T$ and $(\boldsymbol{W}^T\boldsymbol{W}+\lambda_L\boldsymbol{I})\boldsymbol{Z}=\boldsymbol{W}^T(\boldsymbol{Y}-\boldsymbol{U}\circ\boldsymbol{V})$. To solve for $\boldsymbol{U}$ and $\boldsymbol{V}$, we apply the Barzilai-Borwein (BB) method [Barzilai and Borwein (1988)]. It is a gradient method with modified step sizes, which is motivated by Newton's method but does not involve any Hessian matrices. At nearly no extra cost, the method often significantly improves the performance of a standard gradient method. The method is used along with non-monotone line search as a safeguard. Pairing with non-monotone linear search, Barzilai-Borwein gradient methods work very well on general unconstrained differentiable problems. We call the method the LSBF algorithm since both the low-rank bi-factorization structure and sparsity encouraged bi-factorization of the ℓ_1 penalty are used.

3.4 Convergence analysis

In this section, we present the convergence analysis of the proposed LBF algorithm. The LBF algorithm is an instance of block coordinate descent, where each block update performs a minimization step over the corresponding coordinate hyperplane while fixing all other

coordinate blocks. The convergence of the iterate sequence follows from the application of Meyer's monotone convergence theorem [Meyer (1976)] stated in Theorem 3.4.1 and The LBF algorithm has the following convergence guarantee.

Proposition 1. *The LBF algorithm generates a sequence of iterates that has at least one limit point, and the limit points are stationary points of the objective function in (3.3).*

We prove this in two stages. First, we show that all limit points of the LBF algorithm are fixed points of the algorithm map, which follows from the continuity of the algorithm map and Meyer's theorem. Second, we show that fixed points of the LBF algorithm are stationary points of f in (3.3).

Consider

$$\min_{\boldsymbol{S},\boldsymbol{W},\boldsymbol{Z}} \frac{1}{2}\|\boldsymbol{Y}-\boldsymbol{S}-\boldsymbol{W}\boldsymbol{Z}\|_F^2+\lambda_S\|\boldsymbol{S}\|_1+\frac{\lambda_L}{2}\left[\|\boldsymbol{W}\|_F^2+\|\boldsymbol{Z}\|_F^2\right]$$

In each block of the coordinate descent, the updating scheme is given by solving the minimization problems:

$$\boldsymbol{S}^+=\arg\min\frac{1}{2}\|\boldsymbol{Y}-\boldsymbol{W}\boldsymbol{Z}-\boldsymbol{S}\|_F^2+\lambda_S\|\boldsymbol{S}\|$$

$$\boldsymbol{W}^+=\arg\min\frac{1}{2}\|\boldsymbol{Y}-\boldsymbol{S}-\boldsymbol{W}\boldsymbol{Z}\|_F^2+\frac{\lambda_L}{2}\|\boldsymbol{W}\|_F^2$$

$$\boldsymbol{Z}^+=\arg\min\frac{1}{2}\|\boldsymbol{Y}-\boldsymbol{S}-\boldsymbol{W}\boldsymbol{Z}\|_F^2+\frac{\lambda_L}{2}\|\boldsymbol{Z}\|_F^2$$

Let $\boldsymbol{\theta}=(\mathrm{vec}(\boldsymbol{W}),\mathrm{vec}(\boldsymbol{Z}),\mathrm{vec}(\boldsymbol{S}))$ and let $\boldsymbol{W}^+,\boldsymbol{Z}^+,\boldsymbol{S}^+$ denote the $\boldsymbol{W},\boldsymbol{Z},\boldsymbol{S}$ updates such that $\psi_W(\boldsymbol{\theta})=\big(\mathrm{vec}(\boldsymbol{W}^+),\mathrm{vec}(\boldsymbol{Z}),\mathrm{vec}(\boldsymbol{S})\big)$, $\psi_Z(\boldsymbol{\theta})=\big(\mathrm{vec}(\boldsymbol{W}),\mathrm{vec}(\boldsymbol{Z}^+),\mathrm{vec}(\boldsymbol{S})\big)$, and $\psi_S(\boldsymbol{\theta})=\big(\mathrm{vec}(\boldsymbol{W}),\mathrm{vec}(\boldsymbol{Z}),\mathrm{vec}(\boldsymbol{S}^+)\big)$. Let $\psi(\boldsymbol{\theta})=\psi_S\circ\psi_Z\circ\psi_W(\boldsymbol{\theta})$ represents a complete round of block updates.

3.4.1 Continuity of the algorithm map

Proposition 2. *The minimizer $\boldsymbol{W}^+$ of $\frac{1}{2}\|\boldsymbol{Y}-\boldsymbol{S}-\boldsymbol{W}\boldsymbol{Z}\|_F^2+\frac{\lambda_L}{2}\|\boldsymbol{W}\|_F^2$ is jointly continuous in $(\boldsymbol{W},\boldsymbol{Z},\boldsymbol{S})$.*

Proof. Without loss of generality, we denote other variables as $\boldsymbol{\zeta}=(\mathrm{vec}(\boldsymbol{Z}),\mathrm{vec}(\boldsymbol{S}))$. Thus,we can check if the solution $\boldsymbol{W}^+$ is continuous in the variable $\boldsymbol{\zeta}$.

Let $f(\boldsymbol{W};\boldsymbol{\zeta})=\frac{1}{2}\|\boldsymbol{Y}-\boldsymbol{S}-\boldsymbol{W}\boldsymbol{Z}\|_F^2+\lambda_S\|\boldsymbol{S}\|_1+\frac{\lambda_L}{2}\big(\|\boldsymbol{W}\|_F^2+\|\boldsymbol{Z}\|_F^2\big)$. It is easy to know that f is jointly continuous in $\boldsymbol{W}$ and $\boldsymbol{\zeta}$. Let $\boldsymbol{W}^+(\zeta)=\arg\min_{\boldsymbol{W}} f(\boldsymbol{W};\zeta)$. We will prove the continuity by contradiction.

Suppose $\boldsymbol{W}^+(\boldsymbol{\zeta})$ is not continuous at a point $\boldsymbol{\zeta}$, then there exists an $\epsilon > 0$ and a sequence $\{\boldsymbol{\zeta}^{(m)}\}$ converging to $\boldsymbol{\zeta}$ such that $\|\boldsymbol{W}^{(m)} - \boldsymbol{W}^+(\boldsymbol{\zeta})\|_F \geq \epsilon$ for all m where

$$\boldsymbol{W}^{(m)} = \arg\min_{\boldsymbol{W}} f(\boldsymbol{W}; \boldsymbol{\zeta}^{(m)}).$$

Since $\lambda_L > 0$, $f(\boldsymbol{W}; \boldsymbol{\zeta})$ is strongly convex in $\boldsymbol{W}$, so the minimizers $\boldsymbol{W}^{(m)}$ and $\boldsymbol{W}^+(\boldsymbol{\zeta})$ exist and are unique. Without loss of generality we can assume $\|\boldsymbol{\zeta}^{(m)} - \boldsymbol{\zeta}\|_F \leq 1$ since the sequence $\{\boldsymbol{W}^{(m)}\}_{m\in\mathbb{N}}$ converges to $\boldsymbol{\zeta}$.

If $\{\boldsymbol{W}^{(m)}\}_{m\in\mathbb{N}}$ is a bounded sequence, then we can pass to a convergent subsequence with limit $\boldsymbol{W}^*$. For any fixed arbitrary point $\tilde{\boldsymbol{W}}$, we have

$$f(W^{(m)}; \boldsymbol{\zeta}^{(m)}) \leq f(\tilde{\boldsymbol{W}}; \boldsymbol{\zeta}^{(m)}) \tag{3.5}$$

for all m. Since f is continuous in $(\boldsymbol{W}, \boldsymbol{\zeta})$, taking limits on both sides of (3.5) gives us the inequality:

$$f(\boldsymbol{W}^*; \boldsymbol{\zeta}) \leq f(\tilde{\boldsymbol{W}}; \boldsymbol{\zeta}). \tag{3.6}$$

Since $\tilde{\boldsymbol{W}}$ was selected arbitrarily, it follows that $\boldsymbol{W}^* = \boldsymbol{W}^+(\boldsymbol{\zeta})$, which is a contradiction.

Now it remains to show that the sequence $\boldsymbol{W}^{(m)}$ is bounded.

Consider the function

$$g(\boldsymbol{W}) = \sup_{\tilde{\boldsymbol{\zeta}}:\|\tilde{\boldsymbol{\zeta}} - \boldsymbol{\zeta}\|_F \leq 1} \frac{1}{2}\|\tilde{\boldsymbol{Y}} - \tilde{\boldsymbol{S}} - \boldsymbol{W}\tilde{\boldsymbol{Z}}\|_F^2 + \frac{\lambda_L}{2}\|\boldsymbol{W}\|_F^2.$$

We know that g is convex, since it is the point-wise supremum of a collection of convex functions. Since $f(\boldsymbol{W}; \boldsymbol{\zeta}^{(m)}) \leq g(\boldsymbol{W})$ and f is strongly convex in $\boldsymbol{W}$, it follows that $g(\boldsymbol{W})$ is also strongly convex and therefore has a unique global minimizer $\boldsymbol{W}^+$ such that $g(\boldsymbol{W}^+) < \infty$. It also follows that

$$\frac{\lambda_L}{2}\|\boldsymbol{W}^{(m)}\|_F^2 \leq f(\boldsymbol{W}^{(m)}; \boldsymbol{\zeta}^{(m)}) \leq f(\boldsymbol{W}^+; \boldsymbol{\zeta}^{(m)}) \leq g(\boldsymbol{W}^+) < \infty$$

for all m.

Suppose the sequence $\{\boldsymbol{W}^{(m)}\}_{m\in\mathbb{N}}$ is unbounded, namely $\|\boldsymbol{W}^{(m)}\|_F \longrightarrow \infty$, the left hand side must diverge. Thus, we arrive at a contradiction if $\{\boldsymbol{W}^{(m)}\}_{m\in\mathbb{N}}$ is unbounded. □

Similarly, $\boldsymbol{Z}^+$ is jointly continuous in $(\boldsymbol{W}, \boldsymbol{S})$. We conclude that ψ_W and ψ_Z are continuous.

Continuity of ψ_S follows from the fact that proximal maps are continuous; they are in fact 1-Lipschitz or nonexpansive. Therefore ψ is continuous.

3.4.2 Meyer's theorem

After showing the continuity of the algorithm map ψ, we derive that all limit points of the LBF algorithm are fixed points of the algorithm map using Meyer's monotone convergence theorem.

Theorem 3.4.1. *Let f be a continuous function on a domain $\mathscr{D}$, and let ψ be a continuous iterative map from $\mathscr{D}$ into $\mathscr{D}$ such that $f(\psi(\boldsymbol{\theta})) < f(\boldsymbol{\theta})$ for all $\boldsymbol{\theta} \in \mathscr{D}$ with $\psi(\boldsymbol{\theta}) \neq \boldsymbol{\theta}$. Suppose there is a $\boldsymbol{\theta}^{(0)}$ such that the set $\mathscr{L}_f(\boldsymbol{\theta}^{(0)}) \equiv \{\boldsymbol{\theta} \in \mathscr{D} : f(\boldsymbol{\theta}) \leq f(\boldsymbol{\theta}^{(0)})\}$ is compact. Define $\boldsymbol{\theta}^{(k+1)} = \psi(\boldsymbol{\theta}^{(k)})$ for $k = 0, 1, \ldots$, then*

1. *the sequence of iterates $\{\boldsymbol{\theta}^{(k)}\}_{k\in\mathbb{N}}$ has at least one limit point, and all its limit points are fixed points of ψ;*
2. *the distance between successive iterates converges to 0, that is, $\|\boldsymbol{\theta}^{(k+1)} - \boldsymbol{\theta}^{(k)}\| \longrightarrow 0$*

We first check those assumptions of Theorem 3.4.1 hold.

- The objective function is continuous.
- The mapping ψ is continuous.
- The objective function is convex in $\boldsymbol{W}$ and $\boldsymbol{Z}$, respectively. ψ_W and ψ_Z is the stationary point in their updating block. The objective function is strongly convex with respect to block variable $\boldsymbol{S}$. Therefore the block updates ensure that $f(\psi(\boldsymbol{\theta})) < f(\boldsymbol{\theta})$ for all $\boldsymbol{\theta} \in \mathscr{D}$ with $\psi(\boldsymbol{\theta}) \neq \boldsymbol{\theta}$.
- The objective function is coercive in $\boldsymbol{\theta}$, namely, if any of the block variables $\boldsymbol{W}$, $\boldsymbol{Z}$, or $\boldsymbol{S}$ diverges in magnitude, the objective function will diverge. Consequently, for any $\boldsymbol{\theta}^{(0)}$, the set $\mathscr{L}_f(\boldsymbol{\theta}^{(0)}) \equiv \{\boldsymbol{\theta} \in \mathscr{D} : f(\boldsymbol{\theta}) \leq f(\boldsymbol{\theta}^{(0)})\}$ is compact.

Therefore, it follows that the iterate sequence $\{\boldsymbol{\theta}^{(k)}\}_{k\in\mathbb{N}}$ has at least one limit point and all its limit points are fixed points of ψ. We also have $\|\boldsymbol{\theta}^{(k+1)} - \boldsymbol{\theta}^{(k)}\| \longrightarrow 0$.

3.4.3 Fixed points are stationary points

A point $\boldsymbol{u}$ is a stationary point of f, if for all direction vectors $\boldsymbol{v}$,

$$0 \leq f'(\tilde{\boldsymbol{\theta}}; \boldsymbol{v}),$$

where $f'(\tilde{\boldsymbol{\theta}}; \boldsymbol{v})$ is the directional derivative of f at $\tilde{\boldsymbol{\theta}}$ in the direction $\boldsymbol{v}$.

Let $\tilde{\boldsymbol{\theta}}$ be a fixed point of ψ, then it is also a fixed point of mapping ψ_W, ψ_Z and ψ_S. This follows from the fact that ψ_W, ψ_Z, ψ_S only updates a block of $\tilde{\boldsymbol{\theta}}$. Suppose $\psi_W(\tilde{\boldsymbol{\theta}}) = \big(\text{vec}(\boldsymbol{W}^+), \text{vec}(\tilde{\boldsymbol{Z}}), \text{vec}(\tilde{\boldsymbol{S}})\big)$ where $\boldsymbol{W}^+ \neq \tilde{\boldsymbol{W}}$, then $\psi(\tilde{\boldsymbol{\theta}}) = \psi_S \circ \psi_Z \circ \psi_W(\tilde{\boldsymbol{\theta}}) \neq \tilde{\boldsymbol{\theta}}$, which is a contradiction.

Since

$$\begin{aligned}\tilde{\boldsymbol{W}} &= \arg\min_{\boldsymbol{W}} f(\boldsymbol{W}; \tilde{\boldsymbol{Z}}, \tilde{\boldsymbol{S}}) \\ &= \arg\min_{\boldsymbol{W}} \frac{1}{2}\|\boldsymbol{Y} - \tilde{\boldsymbol{S}} - \boldsymbol{W}\tilde{\boldsymbol{Z}}\|_F^2 + \lambda_S\|\tilde{\boldsymbol{S}}\|_1 + \frac{\lambda_L}{2}\big(\|\boldsymbol{W}\|_F^2 + \|\tilde{\boldsymbol{Z}}\|_F^2\big),\end{aligned}$$

we have

$$\boldsymbol{0} \in \partial_{\boldsymbol{W}} f(\tilde{\boldsymbol{W}}; \tilde{\boldsymbol{Z}}, \tilde{\boldsymbol{S}}).$$

That is,

$$\boldsymbol{0} \in \big\{\big[\tilde{\boldsymbol{W}}\tilde{\boldsymbol{Z}} - (\boldsymbol{Y} - \tilde{\boldsymbol{S}})\big]\tilde{\boldsymbol{Z}}^T + \lambda_L\tilde{\boldsymbol{W}}\big\}. \tag{3.7}$$

Similarly, we have

$$\boldsymbol{0} \in \big\{\tilde{\boldsymbol{W}}^T\big[\tilde{\boldsymbol{W}}\tilde{\boldsymbol{Z}} - (\boldsymbol{Y} - \tilde{\boldsymbol{S}})\big] + \lambda_L\tilde{\boldsymbol{Z}}\big\}, \tag{3.8}$$

and

$$\boldsymbol{0} \in \big\{\tilde{\boldsymbol{W}}\tilde{\boldsymbol{Z}} + \tilde{\boldsymbol{S}} - \boldsymbol{Y}\big\} + \lambda_S\partial\|\tilde{\boldsymbol{S}}\|_1. \tag{3.9}$$

Let $f(\boldsymbol{\theta}) = h_1(\boldsymbol{\theta}) + h_2(\boldsymbol{\theta})$, where

$$h_1(\boldsymbol{\theta}) = \frac{1}{2}\|\boldsymbol{Y} - \boldsymbol{S} - \boldsymbol{W}\boldsymbol{Z}\|_F^2$$

and

$$h_2(\boldsymbol{\theta}) = \lambda_S\|\boldsymbol{S}\|_1 + \frac{\lambda_L}{2}\big(\|\boldsymbol{W}\|_F^2 + \|\boldsymbol{Z}\|_F^2\big).$$

We have

$$\partial h_2(\tilde{\boldsymbol{\theta}}) = \begin{pmatrix} \lambda_L \operatorname{vec}(\tilde{\boldsymbol{W}}) \\ \lambda_L \operatorname{vec}(\tilde{\boldsymbol{Z}}) \\ \lambda_S \partial \|\operatorname{vec}(\tilde{\boldsymbol{S}})\|_1 \end{pmatrix}, \partial h_1(\tilde{\boldsymbol{\theta}}) = \begin{pmatrix} \operatorname{vec}(\left[\tilde{\boldsymbol{W}}\tilde{\boldsymbol{Z}} - (\boldsymbol{Y} - \tilde{\boldsymbol{S}})\right]\tilde{\boldsymbol{Z}}^T) \\ \operatorname{vec}(\tilde{\boldsymbol{W}}^T\left[\tilde{\boldsymbol{W}}\tilde{\boldsymbol{Z}} - (\boldsymbol{Y} - \tilde{\boldsymbol{S}})\right]) \\ \operatorname{vec}(\tilde{\boldsymbol{W}}\tilde{\boldsymbol{Z}} + \tilde{\boldsymbol{S}} - \boldsymbol{Y}) \end{pmatrix}.$$

Combining Equation (3.7), Equation (3.8), Equation (3.9), we have

$$\mathbf{0} \in \partial h_1(\tilde{\boldsymbol{\theta}}) + \partial h_2(\tilde{\boldsymbol{\theta}}). \tag{3.10}$$

As a result,

$$\mathbf{0} \in \partial (h_1 + h_2)(\tilde{\boldsymbol{\theta}}).$$

Therefore,

$$\begin{aligned} \forall\, \boldsymbol{y}, \langle \boldsymbol{y}, \mathbf{0} \rangle &= \langle (\tilde{\boldsymbol{\theta}} + \alpha \boldsymbol{y}) - \tilde{\boldsymbol{\theta}}, \mathbf{0} \rangle / \alpha \\ &\leq \left[f(\tilde{\boldsymbol{\theta}} + \alpha \boldsymbol{y}) - f(\tilde{\boldsymbol{\theta}}) \right] / \alpha. \end{aligned}$$

Taking the limit as α goes to 0, we have

$$\langle \boldsymbol{y}, \mathbf{0} \rangle \leq f'(\tilde{\boldsymbol{\theta}}; \boldsymbol{y}) \tag{3.11}$$

That is, for all direction vector $\boldsymbol{v}$,

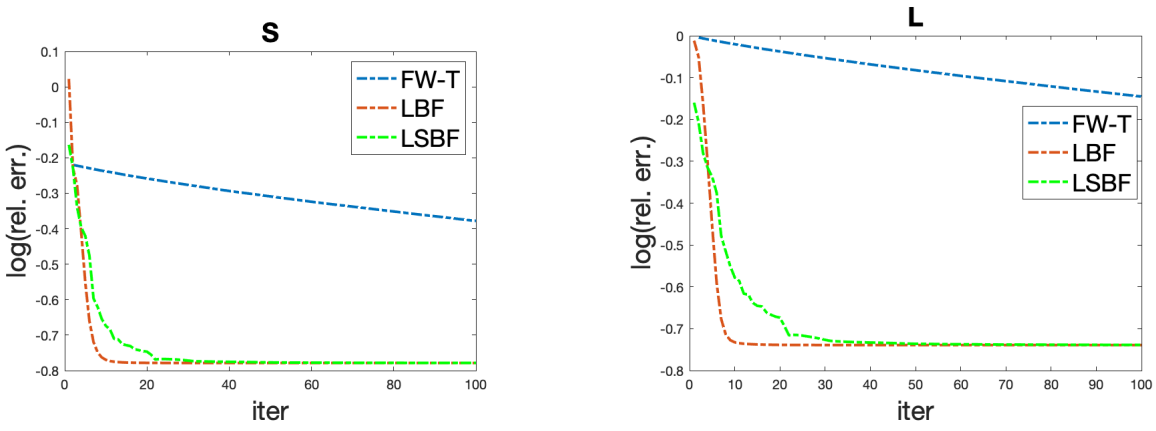
$$0 = \langle \boldsymbol{v}, \mathbf{0} \rangle \leq f'(\tilde{\boldsymbol{\theta}}; \boldsymbol{v}).$$

Consequently, all fixed points of ψ are also stationary points of f.

3.5 Simulation results

In this section, we present the numerical results obtained by applying our LBF method (Algorithm 3) to solve Problem (3.3) and LSBF method to solve Problem (3.4), and compare them with the FW-T method to Problem (3.1). The tuning parameters λ_L and λ_S are kept the same for three methods and chosen to be

$$\lambda_L = \delta \|\boldsymbol{Y}\|_F \quad \text{and} \quad \lambda_S = \delta \|\boldsymbol{Y}\|_F / \sqrt{\max(m, n)}$$



(a) S^k (b) L^K

Figure 3.1: Comparisons on synthetic data. The left figure plots $\log_{10}(\|\boldsymbol{S}^k - \boldsymbol{S}_0\|_F / \|\boldsymbol{S}_0\|_F)$ versus the iteration number k. The right figure plots $\log_{10}(\|\boldsymbol{L}^k - \boldsymbol{L}_0\|_F / \|\boldsymbol{L}_0\|_F)$ versus k.

as suggested in [Candès et al. (2011); Mu et al. (2016)].

We generate a synthetic dataset of size 1000×1000, with the rank of the target low-rank matrix $\boldsymbol{L}_0$ to be 100 and 10% of the entries in the target sparse matrix $\boldsymbol{S}_0$ are nonzero. Figure 3.1 shows the practical convergence of these algorithms. As shown, the progress of FW-T is slower and less efficient compared to LBF and LSBF in recovering $\boldsymbol{L}_0$ and $\boldsymbol{S}_0$.

In Table 3.1, we summarize the recovery result for $\boldsymbol{L}_0$ and $\boldsymbol{S}_0$ by comparing the relative errors of the recovered $\boldsymbol{L}$ and $\boldsymbol{S}$ through these methods. With the same stopping criterion, LBF and LSBF have better recovery results because they progress faster than FW-T.

Table 3.1: Comparisons of the relative error of the recovered $\boldsymbol{S}$ and $\boldsymbol{L}$.

Dataset			$\boldsymbol{S}$			$\boldsymbol{L}$		
m	r	card	FW-T	LBF	LSBF	FW-T	LBF	LSBF
1000	100	0.1	0.3928	0.1802	0.1816	0.6489	0.1910	0.1945
1000	50	0.1	0.1969	0.1550	0.1557	0.3054	0.1279	0.1303
1000	100	0.05	0.5167	0.1500	0.1509	0.6159	0.1433	0.1460
1000	100	0.2	0.3723	0.2316	0.2374	0.7929	0.2905	0.3068
5000	100	0.1	0.3345	0.3186	0.3191	0.4005	0.1770	0.1767

In Figure 3.2, we plot the run time of these three methods on synthetic datasets against the size of the matrix. FW-T algorithm will be terminated if $g(\boldsymbol{x}^{k+1}) - g(\boldsymbol{x}^k))/g(\boldsymbol{x}^k) \leq \epsilon$ for

five consecutive iterations as suggested in [Mu et al. (2016)]. We set $\epsilon = 10^{-3}$ as the stopping criterion for FW-T and terminated LBF and LSBF whenever they reached the objective value returned by the FW-T method. In these examples, m and n are set to be the same. The true rank of $\boldsymbol{L}_0$ is fixed at 100 and 10% of the entries in the target sparse matrix $\boldsymbol{S}_0$ are nonzero. In Figure 3.3, the computational time for different ranks of the target low-rank matrix and different sparsity levels of the target sparse matrix is also compared. As the rank increases, FW-T takes a longer time as it requires more iterations to update the matrices. LBF and LSBF are more steady when the sparsity and rank change. The LBF method is much more efficient than FW-T. LSBF requires updating four variables each time, which adds complexity to the method, and thus is less efficient when the size of the data matrix becomes large.

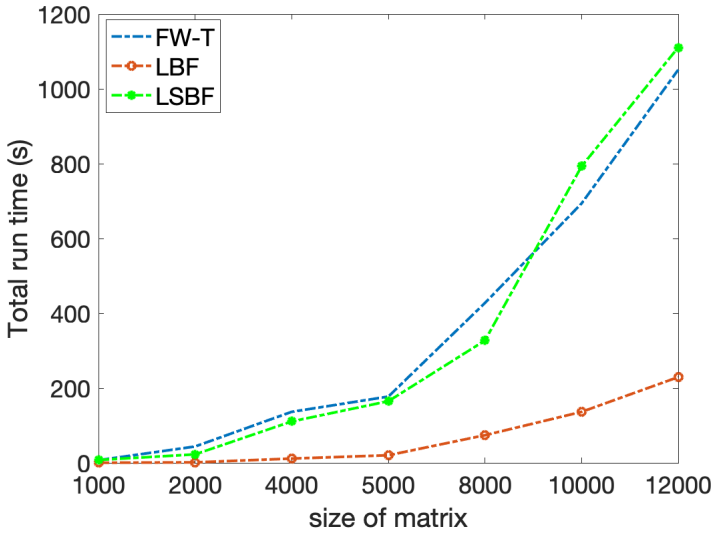
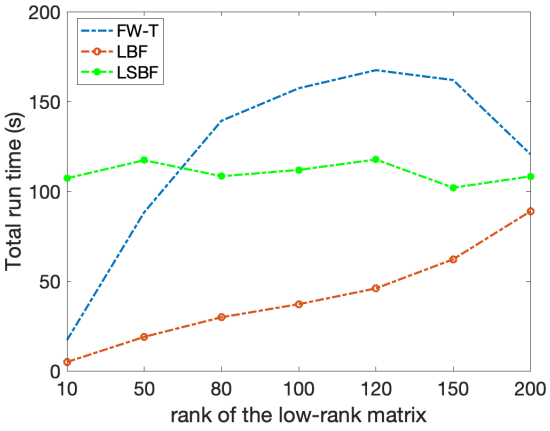


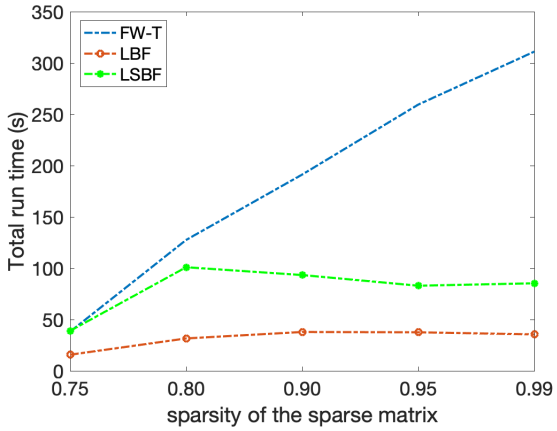
Figure 3.2: Total computational time against data size.

3.6 Discussion

In this chapter, we proposed two algorithms called LBF and LSBF that aim at solving nonconvex variants of PCP. We exploited the idea of characterizing nuclear norm in terms of the Frobenius norm and optimizing ℓ_1 norm using Hadamard product parametrization. Compared with the popular state-of-art FW-T method, our methods converge faster in



(a) 2000×2000 matrix with fixed 10% nonzero entries



(b) 2000×2000 matrix with fixed rank 100

Figure 3.3: Comparing computational time against rank of the target low rank matrix and nonzero proportions of target sparse matrix.

practice. The LBF algorithm avoids the expensive computation of SVD and takes much less cost per iteration, which greatly reduces the total computation time. The promising numerical experiments demonstrated the great potential of our methods to deal with large-scale problems.

CHAPTER

4

TENSOR CANONICAL CORRELATION ANALYSIS

4.1 Introduction

Canonical correlation analysis (CCA) [Hotelling (1936)] is a multivariate analysis technique for identifying associations between two sets of measurements. CCA and its variants, such as Kernel CCA (KCCA) [Zheng et al. (2006)], Locality Preserving CCA (LPCCA) [Sun and Chen (2007)], and sparse CCA (SCCA) [Hardoon and Shawe-Taylor (2011); Mai and Zhang (2019); Chen et al. (2017); Witten and Tibshirani (2009)] have been used in a wide range of research areas.

In many applications, such as imaging genetics where we aim at identifying genetic variants that can best capture and explain phenotypic variations in brain function and structure [Stein et al. (2010)], data produced by modern technologies are presented in the form of multidimensional arrays or tensors. Classic CCA is not appropriate for handling tensor data due to the multidimensional structure and ultrahigh dimensionality of such data. Practitioners often apply CCA to such data by first converting them into vectors. Yet naively

transforming the tensor data into high-dimensional vectors will result in unsatisfactory results. Vectorization destroys the structural information within the tensor structure, such as shapes and spatial correlations, and increases the dimensionality of the problem as well as the computational complexity. For example, turning a $p \times q \times n$ tensor into a $pq \times n$ matrix generates unmanageably high dimensionality even when p and q are not large. The two-dimensional CCA (2DCCA) [Lee and Choi (2007)] is the first work along this line to analyze relations between two sets of image data without reshaping them into vectors. However, the 2DCCA problem is nonconvex, thus there are no guarantees that it converges to a global maximum [Lee and Choi (2007); Chen et al. (2021)].

Therefore, instead of an ad-hoc way of transforming data to vectors, we are interested in developing a framework for tensor data. Min et al. (2019) introduced a novel statistical model for tensor CCA (TCCA), which extends the probabilistic interpretation of CCA by Bach and Jordan (2005) to TCCA. They propose a TCCA population model that enforces model parsimony and enables efficient estimation by imposing the CANDECOMP/PARAFAC (CP) decomposition [Carroll and Chang (1970); Harshman et al. (1970)] structure on canonical tensors. They develop efficient estimation algorithms with global convergence guarantees of the estimation algorithms under modest regularity conditions based on block coordinate ascent. A parallel work by Chen et al. (2021) formulates a tensor extension of 2DCCA and uses the power method [Kolda and Bader (2009); De Lathauwer et al. (2000)] to solve the TCCA problem.

We investigate the TCCA problem and make several contributions in this article:

- We improve the existing framework by building off the algorithms proposed by [Min et al. (2019)]. We tackle some problems that have not been covered in the previous paper and fill a knowledge gap on tensor analysis by bringing new insights.

- We consider several practical issues that arise in real data analysis. We provide an effective initialization scheme that helps the algorithm converge faster and reduces the probability of getting trapped in a poor local optimum. The Ledoit-Wolf covariance estimator is applied here to avoid computational issues that arise in estimating the sample covariance and solving the generalized eigendecomposition problem. Moreover, we utilize the sparse tensor structure to reduce the memory, storage, and computational time cost, which facilitates the application in large-scale data.

- We evaluate the method on several real datasets, including gene expression data, image data, and electricity demands data. The results illustrate the effectiveness and

efficiency of our method, even in the extremely high-dimensional setting. Furthermore, we also exhibit that the method can be used to extract useful and interpretable representations from the data.

- We conduct numerical experiments on several applications and demonstrate new ways of how to use the method, which extends the scope of the existing framework. We also provide practical guidance on using the method.

The remainder of the chapter is organized as follows. In Section 4.2, we review tensor notations and basic operations used in this chapter and introduce CCA. In Section 4.3, we present the TCCA algorithm and discuss several practical considerations when applying the method. Numerical experiment results on both real datasets and simulated datasets are described in Section 4.4. Section 4.5 concludes the discussion on TCCA.

4.2 Preliminaries

4.2.1 Notations

To start with, we briefly introduce the notations and concepts in linear algebra needed for our analysis. For a more detailed overview, please refer to the survey by Kolda and Bader (2009).

A *tensor* is a multidimensional array. The *order* of a tensor is the number of dimensions, also known as ways or modes. Throughout this chapter, vectors (tensors of order one) are denoted by boldface lowercase letters, e.g., $\boldsymbol{a}$. Matrices (tensors of order two) are denoted by boldface capital letters, e.g., $\boldsymbol{A}$. Higher-order tensors (order three or higher) are denoted by boldface Euler script letters, e.g., $\mathscr{X}$. Scalars are denoted by lowercase letters, e.g., a.

Fibers are the higher-order analog of matrix rows and columns. *Matricization*, also known as *unfolding* or *flattening*, is the process of reordering the elements of an N-way array into a matrix. The mode-n matricization of a tensor $\mathscr{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is denoted by $\boldsymbol{X}_{(n)}$ and arranges the mode-n fibers to be the columns of the resulting matrix. On the contrary, *tensorization* is defined as the transformation or mapping of lower-order data to higher-order data. For example, the low-order data can be a vector, and the tensorized result is a matrix, a third-order tensor or a higher-order tensor.

An N-way tensor $\mathscr{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is *rank one* if it can be written as the outer product of N vectors, i.e.,

$$\mathscr{X} = \boldsymbol{a}^{(1)} \circ \boldsymbol{a}^{(2)} \circ \cdots \circ \boldsymbol{a}^{(N)},$$

where $\circ$ is the vector outer product. Elementwise, we have:

$$x_{i_1 i_2 \ldots i_N} = a^{(1)}_{i_1} a^{(2)}_{i_2} \cdots a^{(N)}_{i_N} \text{ for all } 1 \le i_i \le I_n.$$

A *rank-R* tensor can be written as the sum of of R rank-1 tensors, namely,

$$\mathcal{X} = [\![\boldsymbol{A}_1, \ldots, \boldsymbol{A}_N]\!] = \sum_{r=1}^{R} \boldsymbol{a}^{(1)}_r \circ \boldsymbol{a}^{(2)}_r \circ \cdots \circ \boldsymbol{a}^{(N)}_r,$$

where $\boldsymbol{A}_i = (\boldsymbol{a}^{(i)}_1 \;\; \boldsymbol{a}^{(i)}_2 \;\; \cdots \;\; \boldsymbol{a}^{(i)}_R) \in \mathbb{R}^{I_i \times R}$. We use the Kruskal notation $[\![\ldots]\!]$ to concisely summarize the sum. The CANDECOMP/PARAFAC decomposition factorizes a tensor into a sum of component rank-one tensors. For example, given a third-order tensor $\mathcal{X} \in \mathbb{R}^{I \times J}$, we wish to write it as

$$\mathcal{X} \approx \sum_{r=1}^{R} \boldsymbol{a}_r \circ \boldsymbol{b}_r \circ \boldsymbol{c}_r,$$

where R is a positive integer and $\boldsymbol{a}_r \in \mathbb{R}^I$, $\boldsymbol{b}_r \in \mathbb{R}^J$, and $\boldsymbol{c}_r \in \mathbb{R}^K$, for $r = 1, \ldots, R$.

The *inner product* of two tensors of the same dimensions $\mathcal{X}, \mathcal{Y} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is the sum of the product of their entries, i.e.,

$$\langle \mathcal{X}, \mathcal{Y} \rangle = \sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} \cdots \sum_{i_N=1}^{I_N} x_{i_1 i_2 \cdots i_N} y_{i_1 i_2 \cdots i_N}.$$

The *n-mode (matrix)* product of a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ with a matrix $U \in \mathbb{R}^{J \times I_n}$ is denoted by $\mathcal{X} \times_n U$ and is the tensor of size $I_1 \times \cdots \times I_{n-1} \times J \times I_{n+1} \times \cdots \times I_N$ with elements

$$(\mathcal{X} \times_n U)_{i_1 \cdots i_{n-1} j i_{n+1} \cdots i_N} = \sum_{i_n=1}^{I_n} x_{i_1 i_2 \cdots i_N} u_{j i_n}.$$

The *Kronecker product* of matrices $\boldsymbol{A} \in \mathbb{R}^{p_1 \times p_2}$ and $\boldsymbol{B} \in \mathbb{R}^{q_1 \times q_2}$ is a matrix of size $p_1 q_1 - b\,y - p_2 q_2$ and denoted by $A \otimes B$,

$$\boldsymbol{A} \otimes \boldsymbol{B} = \begin{pmatrix} a_{11}\boldsymbol{B} & a_{12}\boldsymbol{B} & \cdots & a_{1p_2}\boldsymbol{B} \\ a_{21}\boldsymbol{B} & a_{22}\boldsymbol{B} & \cdots & a_{2p_2}\boldsymbol{B} \\ \vdots & \vdots & \ddots & \vdots \\ a_{p_1 1}\boldsymbol{B} & a_{p_1 2}\boldsymbol{B} & \cdots & a_{p_1 p_2}\boldsymbol{B} \end{pmatrix}.$$

The *Khatri–Rao product* [Smilde et al. (2004)] of two matrices of the same number of

columns $\boldsymbol{A} \in \mathbb{R}^{p_1 \times p_2}$ and $\boldsymbol{B} \in \mathbb{R}^{q_1 \times p_2}$ is a matrix of size $p_1 q_1 - b\,y - p_2$ and denoted by $\boldsymbol{A} \odot \boldsymbol{B}$. It is a column-wise Kronecker product of $\boldsymbol{A}$ and $\boldsymbol{B}$ and defined by

$$\boldsymbol{A} \odot \boldsymbol{B} = \begin{pmatrix} a_1 \otimes b_1 & a_2 \otimes b_2 & \dots & a_{p_2} \otimes b_{p_2} \end{pmatrix}.$$

The *Hadamard product* of two matrices A and B of the same size $I \times J$ is a matrix of size $I \times J$ and denoted by $\boldsymbol{A} * \boldsymbol{B}$. It is the element-wise product of A and B, i.e.,

$$\boldsymbol{A} * \boldsymbol{B} = \begin{pmatrix} a_{ij} b_{ij} \end{pmatrix}.$$

4.2.2 Canonical correlation analysis

Canonical correlation analysis can be defined as the problem of finding two sets of basis vectors, one for $\boldsymbol{x}$ and the other for $\boldsymbol{y}$, such that the correlations between the projections of the variables onto these basis vectors are mutually maximized.

Formally, let $\boldsymbol{x} \in \mathbb{R}^p$ and $\boldsymbol{y} \in \mathbb{R}^q$ be two random vectors with a joint probability distribution $P(\boldsymbol{x}, \boldsymbol{y})$. The objective of CCA in the population setting is to find $\boldsymbol{v} \in \mathbb{R}^p$ and $\boldsymbol{w} \in \mathbb{R}^q$ such that projections of the random variable onto these directions are maximally correlated:

$$\max_{\boldsymbol{v},\boldsymbol{w}} \frac{\mathbb{E}\left[(\boldsymbol{v}^T \boldsymbol{x})(\boldsymbol{w}^T \boldsymbol{y})\right]}{\sqrt{\mathbb{E}[(\boldsymbol{v}^T \boldsymbol{x})^2]}\sqrt{\mathbb{E}\left[(\boldsymbol{w}^T \boldsymbol{y})^2\right]}}. \tag{4.1}$$

This objective can be written in the equivalent constrained form:

$$\begin{aligned} \max_{\boldsymbol{v},\boldsymbol{w}} \quad & \boldsymbol{v}^T \mathbb{E}_{xy} \boldsymbol{w} \\ \text{s.t.} \quad & \boldsymbol{v}^T \mathbb{E}_{xx} \boldsymbol{v} = \boldsymbol{w}^T \mathbb{E}_{yy} \boldsymbol{w} = 1. \end{aligned}$$

In practice, we do not have access to the population covariance matrices but observe samples pairs $(\boldsymbol{x_1}, \boldsymbol{y_1}), \dots, (\boldsymbol{x_N}, \boldsymbol{y_N})$ drawn from $P(\boldsymbol{x}, \boldsymbol{y})$. We write the data matrices as $\boldsymbol{X} = (\boldsymbol{x_1}, \dots, \boldsymbol{x_N}) \in \mathbb{R}^{p \times N}$, $\boldsymbol{Y} = \begin{pmatrix} \boldsymbol{y_1}, \dots, \boldsymbol{y_N} \end{pmatrix} \in \mathbb{R}^{q \times N}$. Without loss of generality, we assume that $\boldsymbol{X}$ and $\boldsymbol{Y}$ are centered. Define the sample covariances $\hat{\Sigma}_{XX} = \frac{1}{N} \boldsymbol{X}\boldsymbol{X}^T$, $\hat{\Sigma}_{YY} = \frac{1}{N} \boldsymbol{Y}\boldsymbol{Y}^T$ and $\hat{\Sigma}_{XY} = \frac{1}{N} \boldsymbol{X}\boldsymbol{Y}^T$. CCA estimates projection vectors, or canonical vectors, $\boldsymbol{v}$ and $\boldsymbol{w}$ by solving the optimization problem:

$$\text{maximize} \quad \rho = \text{corr}(\boldsymbol{v}^T \boldsymbol{x}, \boldsymbol{w}^T \boldsymbol{y}) = \frac{\boldsymbol{v}^T \boldsymbol{X}\boldsymbol{Y}^T \boldsymbol{w}}{\sqrt{\left(\boldsymbol{v}^T \boldsymbol{X}\boldsymbol{X}^T \boldsymbol{v}\right)\left(\boldsymbol{w}^T \boldsymbol{Y}\boldsymbol{Y}^T \boldsymbol{w}\right)}}. \tag{4.2}$$

The random variables $\boldsymbol{v}^T\boldsymbol{x}$ and $\boldsymbol{w}^T\boldsymbol{y}$ are the canonical variates. For a positive integer $r < \min\{p,q\}$, CCA finds finds r projection vectors $\{\boldsymbol{v}_l^*, \boldsymbol{w}_l^*\}_{l=1}^r$ as canonical directions that sequentially maximize the correlation between $(\boldsymbol{v}_l^*)^T\boldsymbol{X}$ and $(\boldsymbol{w}_l^*)^T\boldsymbol{Y}$. Given the first $(r-1)$ pairs $\{\boldsymbol{v}_l^*, \boldsymbol{w}_l^*\}_{l=1}^{r-1}$, the r-th pair is

$$\begin{aligned}(\boldsymbol{v}_r^*, \boldsymbol{w}_r^*) = \underset{\boldsymbol{v}_r, \boldsymbol{w}_r}{\arg\max}\, & \boldsymbol{v}_r^T \Sigma_{YX} \boldsymbol{w}_r \\ \text{s. t. } & \boldsymbol{v}_r^T \Sigma_{XX} \boldsymbol{v}_r = 1 \\ & \boldsymbol{w}_r^T \Sigma_{YY} \boldsymbol{w}_r = 1 \\ & \boldsymbol{v}_r^T \Sigma_{XX} \boldsymbol{w}_l^* = 0 \text{ for } l < r \\ & \boldsymbol{w}_r^T \Sigma_{YY} \boldsymbol{w}_l^* = 0 \text{ for } l < r.\end{aligned}$$

Multiple projections of CCA can be computed simultaneously by solving the following problem:

$$\begin{aligned}\max\ & tr(\boldsymbol{V}^T\boldsymbol{X}\boldsymbol{Y}^T\boldsymbol{W}) \\ \text{subject to } & \boldsymbol{V}^T\boldsymbol{X}\boldsymbol{X}^T\boldsymbol{W} = \boldsymbol{I}, \boldsymbol{W}^T\boldsymbol{Y}\boldsymbol{Y}^T\boldsymbol{W} = \boldsymbol{I},\end{aligned}$$

where each column of $\boldsymbol{V} \in \mathbb{R}^{p\times r}$ and $\boldsymbol{W} \in \mathbb{R}^{q\times r}$ responds to a projection vector.

4.3 Tensor canonical correlation analysis

Let $\mathcal{X} \in \mathbb{R}^{p_1\times\cdots\times p_{D_x}}$ and $\mathcal{Y} \in \mathbb{R}^{q_1\times\cdots\times q_{D_y}}$ be two random tensors of order D_x and D_y. The vectorizations of $\mathcal{X}$ and $\mathcal{Y}$ are denoted by $\boldsymbol{x}$ and $\boldsymbol{y}$, respectively. We denote the covariances of $\boldsymbol{x}$, the covariances of $\boldsymbol{y}$, and covariance between $\boldsymbol{x}$ and $\boldsymbol{y}$ by Σ_x, Σ_y and Σ_{xy}, respectively. And $\hat{\Sigma}_x$, $\hat{\Sigma}_y$, and $\hat{\Sigma}_{x,y}$ are sample estimates of the corresponding covariances. Let $\mathcal{V} \in \mathbb{R}^{p_1\times\cdots\times p_{D_x}}$ and $\mathcal{W} \in \mathbb{R}^{q_1\times\cdots\times q_{D_y}}$ be constant tensors, and let $\rho(\mathcal{V}, \mathcal{W})$ denote the correlation between the two linear combinations $(\mathcal{X}, \mathcal{V})$ and $(\mathcal{Y}, \mathcal{W})$. The optimization problem for TCCA is given by:

$$\text{maximize } \rho(\mathcal{V}, \mathcal{W}) = \frac{\boldsymbol{v}^T\Sigma_{x,y}\boldsymbol{w}}{\sqrt{(\boldsymbol{v}^T\Sigma_x\boldsymbol{v})(\boldsymbol{w}^T\Sigma_y\boldsymbol{w})}}. \tag{4.3}$$

Suppose we observe N pairs of $i.i.d.$ tensor data $(\mathcal{X}_n, \mathcal{Y}_n)$, and based on sample covari-

Algorithm 4: TCCA with assumption of CP structure on canonical tensors

Input: $\boldsymbol{V}_i^{(0)}$ and $\boldsymbol{W}_j^{(0)}$, for $i \in [D_x]$ and $j \in [D_y]$
$t \leftarrow 0$
while $\rho^{(t)}$ *not converging* **do**
  Select $(i, j) \in [D_x] \times [D_y]$
  $\boldsymbol{V}_{(-i)}^{(t)} \leftarrow \left[\boldsymbol{V}_{D_x}^{(t)} \odot \cdots \odot \boldsymbol{V}_{i+1}^{(t)} \odot \boldsymbol{V}_{i-1}^{(t)} \odot \cdots \odot \boldsymbol{V}_1^{(t)}\right] \otimes \boldsymbol{I}_{p_i}$
  $\boldsymbol{W}_{(-j)}^{(t)} \leftarrow \left[\boldsymbol{W}_{D_y}^{(t)} \odot \cdots \odot \boldsymbol{W}_{j+1}^{(t)} \odot \boldsymbol{W}_{j-1}^{(t)} \odot \cdots \odot \boldsymbol{W}_1^{(t)}\right] \otimes \boldsymbol{I}_{q_j}$
  $\boldsymbol{C}_x^{(t)} \leftarrow \boldsymbol{V}_{(-i)}^{(t)\,T} \hat{\Sigma}_{\boldsymbol{x}_{(i)}} \boldsymbol{V}_{(-i)}^{(t)}$
  $\boldsymbol{C}_y^{(t)} \leftarrow \boldsymbol{W}_{(-j)}^{(t)\,T} \hat{\Sigma}_{\boldsymbol{y}_{(i)}} \boldsymbol{W}_{(-i)}^{(t)}$
  $\boldsymbol{C}_{x,y}^{(t)} \leftarrow \boldsymbol{V}_{(-i)}^{(t)\,T} \hat{\Sigma}_{\boldsymbol{x}_{(i)},\boldsymbol{y}_{(j)}} \boldsymbol{W}_{(-j)}^{(t)}$
  Generalized eigen-decomposition:

$$\begin{pmatrix} \boldsymbol{0} & \boldsymbol{C}_{x,y}^{(t)} \\ \boldsymbol{C}_{x,y}^{(t)\,T} & \boldsymbol{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{w}_i^{(t+1)} \\ \boldsymbol{w}_j^{(t+1)} \end{pmatrix} = \rho^{(t+1)} \begin{pmatrix} \boldsymbol{C}_x^{(t)} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{C}_y^{(t)} \end{pmatrix} \begin{pmatrix} \boldsymbol{w}_i^{(t+1)} \\ \boldsymbol{w}_j^{(t+1)} \end{pmatrix}$$

  t $\leftarrow t+1$
Output: $\mathscr{V} = [\![\boldsymbol{V}_1, \ldots, \boldsymbol{V}_{D_x}]\!]$, $\mathscr{W} = [\![\boldsymbol{W}_1, \ldots, \boldsymbol{W}_{D_y}]\!]$, ρ

ance matrices we estimate $\mathscr{V}$ and $\mathscr{W}$ by solving the optimization problem:

$$\text{maximize } \hat{\rho}(\mathscr{V}, \mathscr{W}) = \frac{\boldsymbol{v}^T \hat{\Sigma}_{x,y} \boldsymbol{w}}{\sqrt{\left(\boldsymbol{v}^T \hat{\Sigma}_x \boldsymbol{v}\right)\left(\boldsymbol{w}^T \hat{\Sigma}_y \boldsymbol{w}\right)}}. \tag{4.4}$$

The pair $(\mathscr{V}, \mathscr{W})$ that maximizes ρ are the canonical tensors, and the optimal ρ is the canonical coefficient.

4.3.1 Algorithm

The high dimensionality of optimization variables $\mathscr{V}$ and $\mathscr{W}$ presents challenges for maximizing the objective in (4.4). To address this, the assumption of CANDECOMP/PARAFAC (CP), or Kruskal, structure on the canonical tensors is imposed by Min et al. (2019). The CP representation generalizes the idea of representing a matrix as the sum of rank-1 matrices to representing a tensor as the sum of rank-1 tensors. Under the CP structure assumption, the parameter search space is limited to tensors of rank-R_x and rank-R_y, instead of searching

over the space of all order-D_x and order-D_y tensor pairs $(\mathscr{V}, \mathscr{W})$:

$$\mathscr{V} = [\![\boldsymbol{V}_1, \ldots, \boldsymbol{V}_{D_x}]\!], \boldsymbol{V}_i \in \mathbb{R}^{p_i \times R_x}, i \in [D_x],$$
$$\mathscr{W} = [\![\boldsymbol{W}_1, \ldots, \boldsymbol{W}_{D_y}]\!], \boldsymbol{W}_i \in \mathbb{R}^{q_i \times R_y}, j \in [D_y].$$

This parameterization reduces the dimensionality from an exponential number of parameters $\prod_{i=1}^{D_x} p_i + \prod_{j=1}^{D_y} q_j$ to a linear number of parameters $R_x \sum_{i=1}^{D_x} p_i + R_y \sum_{j=1}^{D_y} q_j$, as well as alleviating the burden of estimating a huge covariance matrix. In this way, the multiway information is preserved. Algorithm 4.3.1 summarizes the estimation procedure and it has provable convergence guarantees under modest regularity conditions [Min et al. (2019)].

4.3.2 Sparse tensor structure

One of the obstacles in applying TCCA to large-scale data is the storage and computational cost involved in tensor operations, which is computationally and memory intensive as the size of the tensor grows. Various techniques are employed to more efficiently store and compute these tensors by taking advantage of specific tensor structures. Sparse tensors exist in many naturally occurring datasets [Tew (2016)]. Tensors are considered sparse if they hold a majority of zero values. By only storing and operating on the non-zero values, we can reduce the storage requirements of a tensor and eliminate unnecessary computations involving zeros.

We utilize the **sptensor** class [Bader and Kolda (2008)] in Matlab Tensor Toolbox that stores the data in coordinate format. Let $P = \prod_{i=1}^{D_x} p_i, Q = \prod_{j=1}^{D_y} q_j$ and $nnz(\mathscr{X})$ be the number of non-zero values of $\mathscr{X}$. Table 4.1 summarizes the flop counts of regular tensor and sparse tensor in major steps of Algorithm 4.3.1. By taking advantage of the sparse tensor structure, we could reduce the computation complexity.

In Table 4.2, we compare the runtime and memory usage with different tensor structures. It demonstrates that taking advantage of sparse tensor structure can not only reduce memory cost but also save time.

4.3.3 Effective initialization

It is proved that the limit points of the iterate sequence generated by Algorithm 4.3.1 are canonical tensors of the sample TCCA problem (4.4) provided that $\hat{\Sigma}_x$ and $\hat{\Sigma}_y$ are nonsingular [Min et al. (2019)]. Thus a good initial point is important.

Table 4.1: Flops calculation in major steps

Step	Flops (regular tensor)	Flops (sparse tensor)
Update $\boldsymbol{C}_x$	$\mathcal{O}(R_x p_i P^2)$	$\mathcal{O}(R_x p_i nnz(\mathcal{X})) + \mathcal{O}(R_x p_i N^2) + \mathcal{O}((R_x p_i)^2 N)$
Update $\boldsymbol{C}_y$	$\mathcal{O}(R_y q_j Q^2)$	$\mathcal{O}(R_y q_j nnz(\mathcal{Y})) + \mathcal{O}(R_y q_j N^2) + \mathcal{O}((R_y q_j)^2 N)$
Update $\boldsymbol{C}_{xy}$	$\min\{\mathcal{O}(R_x p_i PQ) + \mathcal{O}(R_x p_i Q R_y q_j), \mathcal{O}(R_y q_j PQ) + \mathcal{O}(R_x p_i P R_y q_j)\}$	$\mathcal{O}(R_x p_i nnz(\mathcal{X})) + \mathcal{O}(R_y q_j nnz(\mathcal{Y})) + \mathcal{O}(\min\{R_x p_x N^2, R_y q_j N^2\}) + \mathcal{O}(R_x p_i N R_y q_j)$

Table 4.2: Runtime and memory comparison

Size	Structure	Time (s)	Memory (kb)
$\mathcal{X}$: (100, 200, 1000), cardinality = 0.01	regular	90 (35 iter, $R_x = 2$, $R_y = 1$)	8880870
$\mathcal{Y}$: (100, 1000), cardinality = 0.01	sparse	10 (35 iter, $R_x = 2$, $R_y = 1$)	5169013
$\mathcal{X}$: (100, 200, 1000), cardinality = 0.05	regular	91 (37 iter, $R_x = 2$, $R_y = 1$)	9003944
$\mathcal{Y}$: (100, 1000), cardinality = 0.05	sparse	15 (37 iter, $R_x = 2$, $R_y = 1$)	5420239
$\mathcal{X}$: (100, 200, 1000), cardinality = 0.1	regular	106 (43 iter, $R_x = 2$, $R_y = 1$)	9356393
$\mathcal{Y}$: (100, 1000), cardinality = 0.1	sparse	23 (43 iter, $R_x = 2$, $R_y = 1$)	6397610

To this end, we propose an effective initialization procedure that helps the algorithm converge in fewer iterations to initialize $\mathcal{V}$ and $\mathcal{W}$. We first perform SVD on the covariance matrix $\Sigma_{\boldsymbol{x}_{(N)},\boldsymbol{y}_{(N)}}$. After obtaining the left and right singular matrices, we reshape them back into compatible tensors and perform CP decomposition on these tensors. The factor matrices that refer to the combination of the vectors from the rank-one components of CP decomposition are used as initial points for $\mathcal{V}$ and $\mathcal{W}$, respectively:

$$\mathcal{U}_{-N} \approx [\![\boldsymbol{V}_1^{(0)}, \ldots, \boldsymbol{V}_{D_x}^{(0)}]\!],$$
$$\mathcal{V}_{-N} \approx [\![\boldsymbol{W}_1^{(0)}, \ldots, \boldsymbol{W}_{D_y}^{(0)}]\!],$$

where $\mathcal{U}_{-N}$ and $\mathcal{V}_{-N}$ are obtained by reshaping the left singular vector $\boldsymbol{u}$ and the right singular vector $\boldsymbol{v}$ that correspond to the largest singular value of $\Sigma_{\boldsymbol{x}_{(N)},\boldsymbol{y}_{(N)}}$ into tensors of size $p_1 \times \cdots \times p_{D_x}$ and $q_1 \times \cdots \times q_{D_y}$, respectively.

In Figure 4.1, we plot the correlation in each iteration with different initialization methods on the same toy dataset. This figure reveals that the proposed initialization method has identical paths of correlation and the same convergence property.

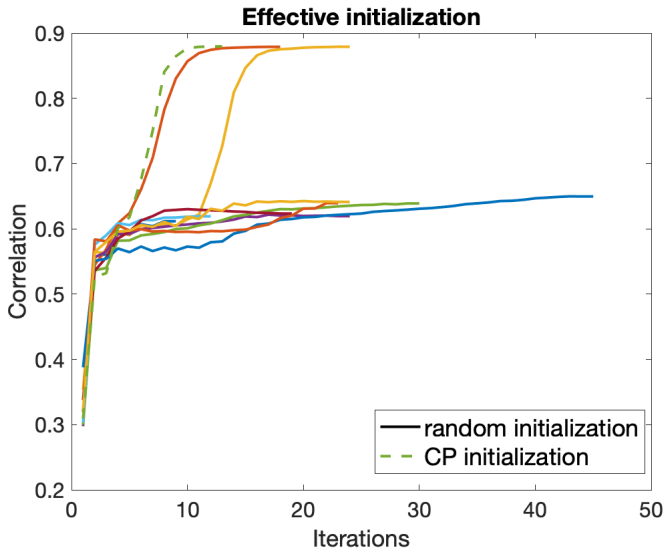


Figure 4.1: Correlation in each iteration with 100 random initializations and CP initialization on the same data set. This figure reveals that the proposed initialization method has identical paths of correlation and the same convergence property.

4.3.4 Ledoit-Wolf covariance estimator

Algorithm 4.3.1 relies on the sample estimates of separable covariances $\hat{\Sigma}_{\boldsymbol{x}_{(i)}}$ and $\hat{\Sigma}_{\boldsymbol{y}_{(j)}}$, and the cross-covariance $\hat{\Sigma}_{\boldsymbol{x}_{(i)},\boldsymbol{y}_{(j)}}$. The usual approach is to use a maximum likelihood estimator, which is unbiased and converges to the true (population) covariance when given many observations. However, for large-dimensional covariance matrices, the usual estimator — the sample covariance matrix is typically not well-conditioned and may not even be invertible. When the matrix dimension p is larger than the number n of observations available, the sample covariance matrix is not even invertible. When the ratio p/n is less than one but not negligible, the sample covariance matrix is invertible but numerically ill-conditioned. In this regard, it is important to develop a well-conditioned estimator for large-dimensional covariance matrices. Ledoit and Wolf (2004b) introduce an estimator that is both well-conditioned and asymptotically more accurate than the sample covariance matrix:

$$\hat{\Sigma}_{Shrink} = (1-\delta)\hat{\Sigma} + \delta \frac{Tr(\hat{\Sigma})}{p} \boldsymbol{I}, \tag{4.5}$$

where δ is the shrinkage constant.

The estimator is the asymptotically optimal convex linear combination of the sample covariance matrix with the identity matrix. Ledoit and Wolf (2004a) derives a formula for

estimating the optimal δ^* with regard to the criterion that minimizes the expected distance between the shrinkage estimator and the true covariance matrix.

4.4 Numerical experiments

In this section, we carefully evaluate the TCCA algorithm and demonstrate new directions of possible applications that are not discussed in [Min et al. (2019)] through simulation studies and empirical data analysis. Comparisons with other popular methods are also included.

4.4.1 Correlation analysis

Chen et al. (2021) present a power method that can solve the two-dimensional CCA (2DCCA) problem and present a natural extension of 2DCCA to tensor-valued data. To distinguish from our method, we call it 2DTCCA in this chapter. We evaluate our method using real tensor data used in [Chen et al. (2021)] for comparison.

Table 4.3: The summary of two methods on three datasets.

		TCCA	2DTCCA
Gene expression	corr (train)	0.999	0.979
	corr (test)	0.889	0.833
	time (sec)	0.886	0.089
MNIST	corr (train)	0.964	0.939
	corr (test)	0.940	0.943
	time (sec)	0.011	1.181
Adelaide	corr (train)	0.983	0.972
	corr (test)	0.976	0.969
	time (sec)	0.018	0.097

The gene expression dataset uses 318 individuals with genotype data from 1000 genomes phase 1 and corresponding RNA-seq data from GEUVADIS in four populations, GBR, FIN, YRI, and TSI. They follow the same procedure given in [Brown et al. (2018)] to extract expression data and remove the confounding, and randomly select 5000 features of gene expression and genomics for reducing dimension. The datasets are separated by a training

set and a testing set to test generalization. MNIST [Deng (2012)] is a database of handwritten digits. It consists of 28×28 gray-scale digit images. Labels of the handwritten digits are processed using one-hot encoding. The Adelaide example investigates the relationship between electricity demands in Adelaide, Australia, and temperatures measured at Kent Town from Sunday to Saturday between 7/6/1997 and 3/31/2007. The demands and temperatures are measured every half-hour. The data are represented as two 508 by 48 by 7 tensors.

The result is presented in Table 4.3. Although the problem formulation is different for the two methods, they achieve comparable performance on real data sets with TCCA being slightly better in terms of test correlation.

To better illustrate the canonical representations obtained by TCCA, we perform principal component analysis (PCA) [Pearson (1901)] on those canonical representations. Figure 4.2 shows populations are well-separated by the first two PCA components. It demonstrates that TCCA can be used to extract useful and interpretable representations from the data.

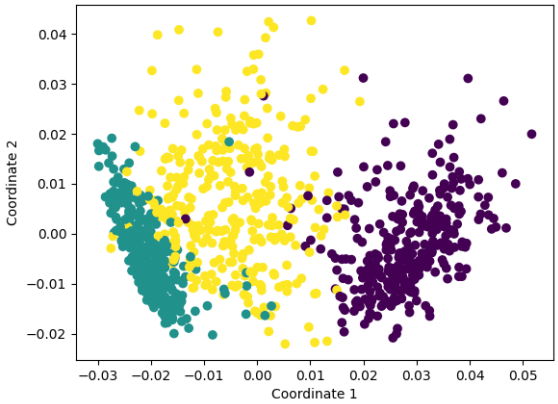


Figure 4.2: PCA on MNIST canonical representations.

4.4.2 Multi-class classification

In many applications, such as classification of images or videos, the tensor data contain spatial correlation among neighbouring pixels. Considering the computational and undersampling issues caused by tensor vectorization , it is of interest to develop a framework

for tensor data instead of an ad-hoc way of transforming data to vectors. Under the TCCA framework, we could preserve this multiway information and encode useful signals in the canonical representation with much fewer training samples and much less training time.

The EMNIST [Cohen et al. (2017)] dataset is a set of handwritten character digits derived from the NIST Special Database 19 [Grother (1995)] and converted to a 28×28 pixel image format and dataset structure that directly matches the MNIST dataset. We compared our method with 2DTCCA, matrix LDA [Hu et al. (2020)] and a simple 3-layer neural network(NN) on the EMNIST dataset. Matrix LDA extends Fisher linear discriminant analysis (LDA) [McLachlan (2004); Fisher (1936)] to matrix-valued predictors. It formulates the binary classification problem as a nuclear norm penalized least-squares problem, and efficiently exploits the low-rank structure of the two-dimensional discriminant direction matrix. To apply matrix LDA in this multi-class classification problem, we use the widely used One-vs-Rest strategy that splits a multi-class classification into one binary classification problem per class. The neural network [Liu et al. (2017); Rawat and Wang (2017); Wang et al. (2019)] is a computational model that works in a similar way to the neurons in the human brain. Each neuron takes an input, performs some operations then passes the output to the following neuron. Recent years have witnessed the success of deep neural networks in many research areas, including image classification.

Figure 4.3 plots the classification accuracy of four methods and their runtime against the number of training samples used. Although none of these methods beat the simple neural network when a large amount of training samples is used, the TCCA algorithm still achieves higher accuracy compared with another two methods, and its computational time is low. Nonetheless, this suggests that when we do not have enough training data or when we have limited computational budgets, the TCCA algorithm could still be a good choice to perform classification.

4.4.3 Support recovery performance

Driven by applications in genomics, imaging research, etc., many recent studies generalize CCA to high-dimensional settings where the number of variables exceeds the sample size or settings when the variables are highly correlated. However, most of them either rely on strong assumptions on covariance matrices. Mai and Zhang (2019) proposed a new sparse CCA (SCCA) method that recasts high-dimensional CCA as an iterative penalized least squares problem. With properly chosen penalty functions, it performs structured variable selection and incorporates prior information. It does not impose any sparsity assumptions

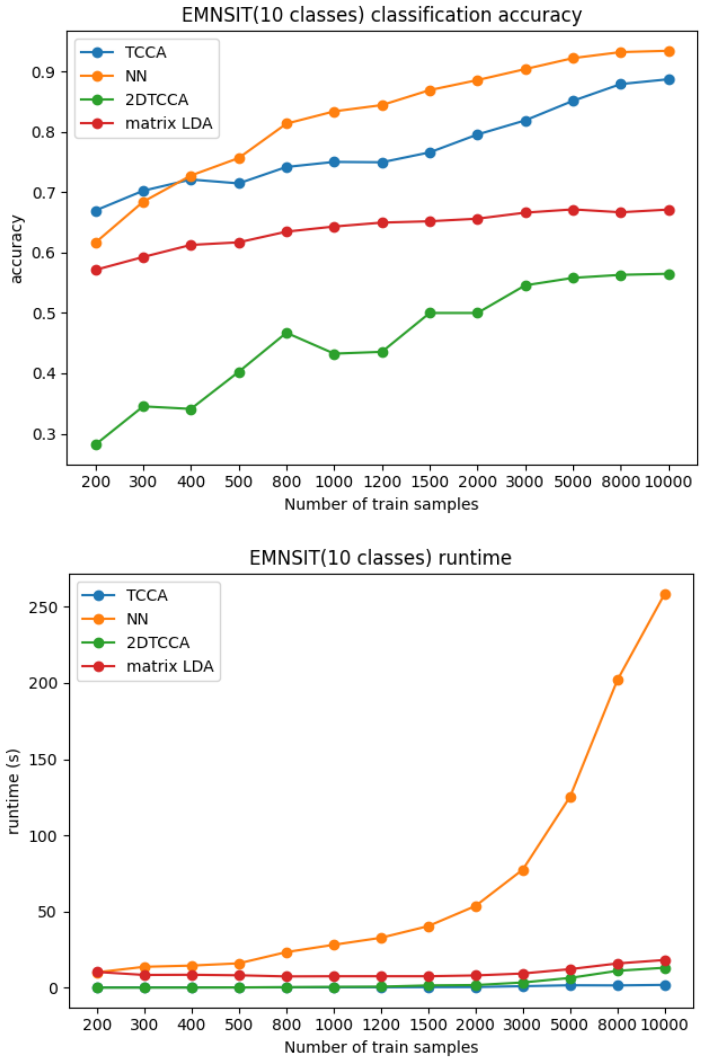


Figure 4.3: Comparing the classification accuracy on EMNIST dataset with 10 classes for four methods.

on the covariance matrices and produces nested solutions.

We follow the simulation setup in [Mai and Zhang (2019)] and consider the problem in (4.2). In each simulation setting, we specify $\mathbf{\Sigma}_{XX}$, $\mathbf{\Sigma}_{YY}$ and $\{\boldsymbol{v}_l^*, \boldsymbol{w}_l^*\}_{l=1}^r$. Then we simulate $(\boldsymbol{X}, \boldsymbol{Y}) \sim N(0, \mathbf{\Sigma})$, where

$$\mathbf{\Sigma} = \begin{pmatrix} \mathbf{\Sigma}_{XX} & \mathbf{\Sigma}_{YX} \\ \mathbf{\Sigma}_{YX}^T & \mathbf{\Sigma}_{YY} \end{pmatrix}$$

and $\mathbf{\Sigma}_{YX} = \mathbf{\Sigma}_{XX}\{\sum_{l=1}^r \rho_l^* \boldsymbol{v}_l^* (\boldsymbol{w}_l^*)^T\}\mathbf{\Sigma}_{YY}$.

We consider the model used in [Mai and Zhang (2019)]: $\mathbf{\Sigma}_{XX} = \mathbf{\Sigma}_{YY} = AR(0.8)$, where $\mathbf{\Sigma} = AR(s)$ if $\mathbf{\Sigma}_{ij} = s^{|i-j|}$ for any positive definite matrix $\mathbf{\Sigma}$ and any positive number s, and there is one canonical pair $\boldsymbol{v}_1^* = \boldsymbol{w}_1^* \propto (\mathbf{1}_{nnz}, \mathbf{0}_{p-nnz})$. We fix $n = 500, p = q = 2000$, and vary the sparsity level of the canonical vectors. We monitor the support recovery performance [Rothman et al. (2010)] using the true-positive rate and the true-negative rate. The true-positive rate measures the probability of including true relevant variables detected by the estimation procedure. The true-negative rate measures the hit rate of excluding unimportant variables from the canonical tensors. The results on sparsity recognition performance for the sparse simulation designs between SCCA and TCCA are reported in Table 4.4. We observe that when the number of relevant variables to be detected is rare, SCCA is doing well at recovering them. As the number increases, SCCA is doing worse. TCCA has more consistent performances, and both the probability of including relevant variables and the probability of excluding relevant variables are high.

4.4.4 High dimensional data matrices

In many modern data analysis problems, such as imaging genetics, each observation may consist more generally of a pair of multidimensional arrays or tensors. A naïve approach used in classic CCA is to transform tensor-valued data into vectors. However, this is not appropriate due to the multidimensional structure and ultrahigh dimensionality of such modern data. After vectorization, structural information in tensors is discarded. And a prohibitively large number of parameters needs to be estimated in the resulting vectors.

Not only tensor vectorization has many drawbacks, reshaping vectors into tensors even if the data are not presented as tensors also introduces many benefits in TCCA, especially within the small n, large p framework. With the CP assumption on tensor structures, transforming vectors to tensors will greatly reduce the number of parameters in the search space. By introducing some bias, tensorization actually reduces the variances and could prevent the algorithm from overfitting.

Table 4.4: Support recovery performance: true-positive rate and true-negative rate for canonical vectors

Sparsity	Model	TPR($\mathcal{V}$)	TNR($\mathcal{V}$)	TPR($\mathcal{W}$)	TNR($\mathcal{W}$)
0.005	TCCA	1.0000	1.0000	1.0000	1.0000
	SCCA	1.0000	0.9869	1.0000	0.9844
0.01	TCCA	1.0000	1.0000	1.0000	0.9949
	SCCA	1.0000	0.9990	1.0000	0.9990
0.02	TCCA	1.0000	1.0000	1.0000	0.9949
	SCCA	0.8000	0.9888	0.8750	0.9913
0.03	TCCA	1.0000	0.9948	1.0000	0.9948
	SCCA	0.7333	0.9964	0.7167	0.9943
0.05	TCCA	1.0000	1.0000	1.0000	1.0000
	SCCA	0.5800	0.9921	0.5900	0.9900
0.1	TCCA	1.0000	1.0000	1.0000	1.0000
	SCCA	0.4250	0.9761	0.4250	0.9739
0.2	TCCA	1.0000	1.0000	1.0000	1.0000
	SCCA	0.2825	0.9525	0.2350	0.9575

We demonstrate the applicability of TCCA on simulated high-dimensional data matrices. For any matrix $\boldsymbol{M}$, let $\mathscr{P}_{\boldsymbol{M}}$ be the projection matrix onto the column space of $\boldsymbol{M}$. We write the true parameters as $\boldsymbol{V}^* = (\boldsymbol{v}_1^*, \ldots, \boldsymbol{v}_r^*)$ and $\boldsymbol{W}^* = (\boldsymbol{w}_1^*, \ldots, \boldsymbol{w}_r^*)$, and denote $\hat{\boldsymbol{V}} = (\hat{\boldsymbol{v}}_1, \ldots, \hat{\boldsymbol{v}}_r)$ and $\hat{\boldsymbol{W}} = (\hat{\boldsymbol{w}}_1, \ldots, \hat{\boldsymbol{w}}_r)$ as the estimates. To evaluate the methods, we compute $\text{err}(\hat{\boldsymbol{V}}) = \|\mathscr{P}_{\hat{\boldsymbol{V}}} - \mathscr{P}_{\boldsymbol{V}^*}\|_F$ and $\text{err}(\hat{\boldsymbol{W}}) = \|\mathscr{P}_{\hat{\boldsymbol{W}}} - \mathscr{P}_{\boldsymbol{W}^*}\|_F$. We generate an independent validation set with the same size as the training set to choose the tuning parameters for SCCA and TCCA.

Figure 4.4 evaluates the effect of covariance structure for SCCA and TCCA. The simulation results are obtained over 10 replicates. Varying s from small to large results in the covariance matrices becoming far away from centering on the diagonal lines. As the covariance becomes more complicated, TCCA is able to capture the structural information while SCCA performs worse because they lose the information.

We also illustrate the computational efficiency and scalability of our algorithm. We consider the simulations with $n = 500$ and dimension $p = q$ varying from 1000 to 8000. Figure 4.5 shows the reconstruction errors on data sets of different dimensions by changing p/n ratio. As the feature dimension increases, our method is consistently doing better and the performance does not get much hurt by this change. The averaged computing time elapsed for each method on data of different sizes is reported in Figure 4.6. As the feature dimension increases, the runtime of SCCA grows much faster than our method, and TCCA

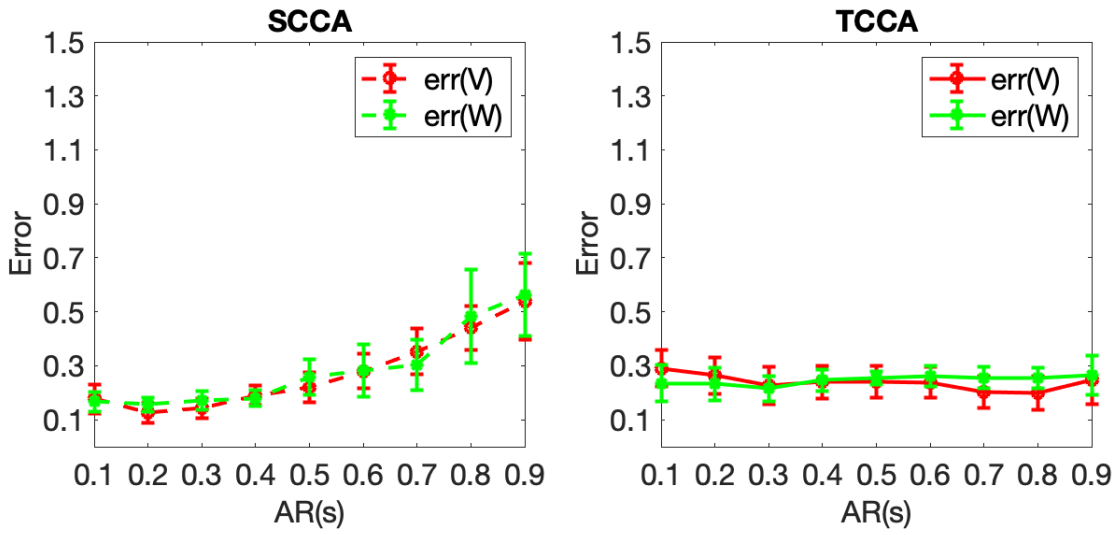


Figure 4.4: Simulation results for $n = 500$, $p = q = 1500$. We plot the mean with the standard error over 10 replicates.

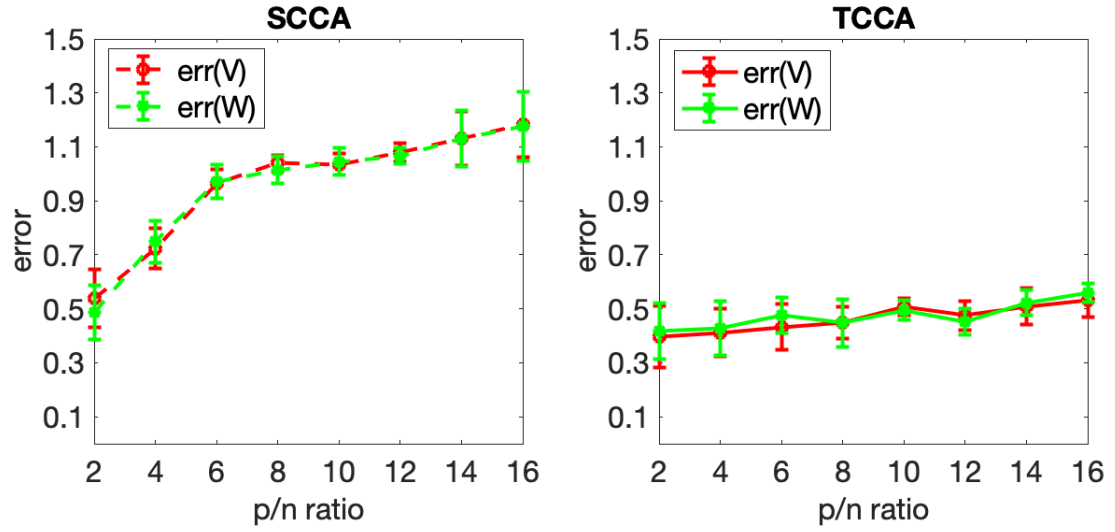


Figure 4.5: Simulation results for $n = 500$, $p = q$, with sparsity level 0.01.

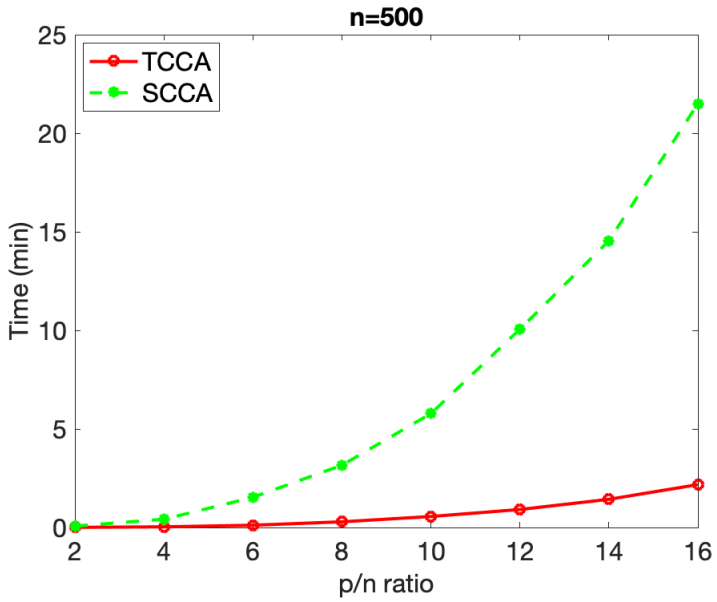


Figure 4.6: Runtime elapsed for $n = 500$, $p = q$, with sparsity level 0.01.

is much more scalable.

The question of interest is whether and how transforming high-dimensional matrices into tensors contribute to the analysis. The above results illustrate the advantages of reshaping high dimensional matrices into tensors and then performing TCCA. In this small n, large p high-dimensional setting, using TCCA is implicitly imposing regularization and enjoys many benefits as demonstrated in our simulations.

Now we evaluate the effect of how we perform tensorization. In Table 4.5, we present results by reshaping data into tensors with the same mode but different orders. Permutation in the dimensions of tensor modes does not affect the performance of TCCA.

Table 4.5: Reshaping tensors with different orders. The reported numbers are the means and standard errors (in parentheses) over 10 replicates.

$p = q$	Tensor size	# params	err($\mathcal{V}$)	err($\mathcal{W}$)	ρ_{test}	ρ_{train}
$p = 3000$	(3, 10, 10, 10, 500)	33	0.2070 (0.0300)	0.2138 (0.0336)	0.8931	0.9073
	(10, 3, 10, 10, 500)	33	0.1869 (0.0323)	0.1866 (0.0273)	0.8906	0.9076
	(10, 10, 3, 10, 500)	33	0.2041 (0.0310)	0.1995 (0.0339)	0.8872	0.9077
	(10, 10, 10, 3, 500)	33	0.1854 (0.0295)	0.2040 (0.0179)	0.8949	0.9068

We also evaluate the effects of tensor sizes to be used and summarize the result in Table 4.6. It is advantageous to increase the number of modes when the dimensionalities of data matrices are high. Reshaping the tensor is one way of reparametrization. This changes the number of parameters to be estimated in the model. By adding more modes, we enforce more assumptions on the model and thus introduce more bias. At first, as the number of model parameters to be estimated decreases, the performance will improve because the reduction in variance is worth the extra bias. When the introduced bias is too strong, the performance will decrease as the model is now underfitting. Figure 4.7 plots errors for different models and displays the trend of the bias-variance trade-off.

Table 4.6: Results by reshaping tensors into different sizes for $n = 50$, $p = q = 300$ and sparsity level = 0.5. The reported numbers are the means and standard errors (in parentheses) over 10 replicates.

Tensor size	# params	err($\mathcal{V}$)	err($\mathcal{W}$)	ρ_{test}	ρ_{train}
(300, 50)	300	1.1045 (0.1011)	1.1279 (0.1159)	0.5206	0.9848
(10, 30, 50)	40	0.8982 (0.1619)	0.9240 (0.1431)	0.6618	0.9786
(3, 10, 10, 50)	23	0.5811 (0.0930)	0.5543 (0.1148)	0.8482	0.9460
(3, 2, 5, 10, 50)	20	0.5111 (0.1168)	0.5459 (0.1229)	0.8326	0.9505
(3, 2, 5, 2, 5, 50)	17	0.6509 (0.1309)	0.6537 (0.1067)	0.7919	0.8931

4.5 Discussion

In this chapter, we investigate the topic of canonical correlation analysis within the context of tensor-valued data. We provide a deeper understanding of an existing method TCCA from several aspects. Real datasets are analyzed and the results demonstrate the method is efficient and effective in many applications. The benefits of applying TCCA on data that are presented as or transformed to tensor are also justified by simulations with different models and parameters.

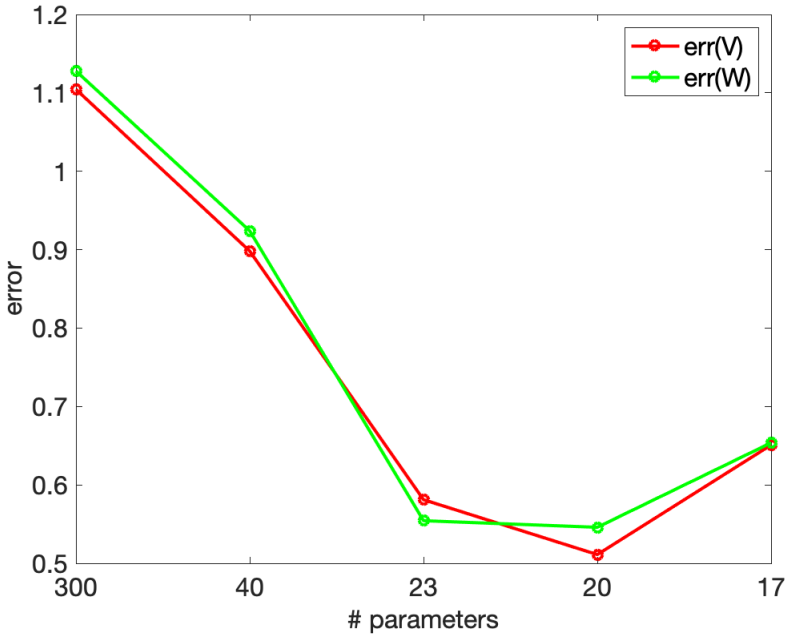


Figure 4.7: Results for $n = 50$, $p = 300$

CHAPTER

5

CONCLUSION

In this thesis, we focus on some smooth optimization problems under different assumptions and constraints. In Chapter 2 and Chapter 4, we explore new directions of existing methods and further extend the scope by tackling the problems that have not been addressed before. In Chapter 2, we present a multi-scale metric that exploits the correlations among both the rows and columns of the data matrix to efficiently compute the affinities in a missing data setting. In Chapter 4, we introduce an algorithm for solving the problem of tensor canonical correlation analysis and fill a knowledge gap on tensor analysis. In Chapter 3, we develop a new algorithm for estimating a low-rank and a sparse matrix from given observations. We demonstrate that it has a faster convergence rate, a cheaper computational cost, and better recovery results than the state-of-art method. The convergence property is also established in Chapter 3.

For future work, it is of interest to develop a more principled approach for tensor rank selection in Chapter 4.

REFERENCES

Ankenman, J. I. (2014). *Geometry and Analysis of Dual Networks on Questionnaires*. PhD thesis, Yale University.

Aybat, N. S., Goldfarb, D., and Iyengar, G. (2011). Fast first-order methods for stable principal component pursuit. *arXiv preprint arXiv:1105.2126*.

Aybat, N. S., Goldfarb, D., and Ma, S. (2014). Efficient algorithms for robust and stable principal component pursuit problems. *Computational Optimization and Applications*, 58(1):1–29.

Bach, F. R. and Jordan, M. I. (2005). A probabilistic interpretation of canonical correlation analysis.

Bader, B. W. and Kolda, T. G. (2008). Efficient matlab computations with sparse and factored tensors. *SIAM Journal on Scientific Computing*, 30(1):205–231.

Barzilai, J. and Borwein, J. M. (1988). Two-point step size gradient methods. *IMA Journal of Numerical Analysis*, 8(1):141–148.

Beck, A. and Teboulle, M. (2009). A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202.

Belkin, M. and Niyogi, P. (2003). Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396.

Bennett, J. and Lanning, S. (2007). The Netflix Prize. In *In KDD Cup and Workshop in conjunction with KDD*.

Boluki, S., Zamani Dadaneh, S., Qian, X., and Dougherty, E. R. (2019). Optimal clustering with missing values. *BMC Bioinformatics*, 20(S12):321.

Brown, B. C., Bray, N. L., and Pachter, L. (2018). Expression reflects population structure. *PLoS genetics*, 14(12):e1007841.

Candès, E. J., Li, X., Ma, Y., and Wright, J. (2011). Robust principal component analysis? *J. ACM*, 58(3).

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

Carroll, J. D. and Chang, J.-J. (1970). Analysis of individual differences in multidimensional scaling via an n-way generalization of "eckart-young" decomposition. *Psychometrika*, 35(3):283–319.

Chen, M., Gao, C., Ren, Z., and Zhou, H. H. (2017). Sparse cca via pre- cision adjusted iterative thresholding. *Proceedings of International Congress of Chinese Mathematicians*.

Chen, Y.-L., Kolar, M., and Tsay, R. S. (2021). Tensor canonical correlation analysis with convergence and statistical guarantees. *Journal of Computational and Graphical Statistics*, 30(3):728–744.

Chi, E. C., Allen, G. I., and Baraniuk, R. G. (2017). Convex biclustering. *Biometrics*, 73(1):10–19.

Chi, E. C., Hu, L., Saibaba, A. K., and Rao, A. U. K. (2018). Going off the grid: Iterative model selection for biclustered matrix completion. *Journal of Computational and Graphical Statistics*, 28(1):36–47.

Chi, E. C. and Li, T. (2019). Matrix completion from a computational statistics perspective. *Wiley Interdisciplinary Reviews: Computational Statistics*, 11(5):e1469.

Chi, E. C. and Steinerberger, S. (2019). Recovering trees with convex clustering. *SIAM Journal on Mathematics of Data Science*, 1(3):383–407.

Chi, J. T., Chi, E. C., and Baraniuk, R. G. (2016). k-pod: A method for k-means clustering of missing data. *The American Statistician*, 70(1):91–99.

Cohen, G., Afshar, S., Tapson, J., and van Schaik, A. (2017). Emnist: an extension of mnist to handwritten letters.

Coifman, R. R. and Gavish, M. (2011). Harmonic analysis of digital data bases. In *Wavelets and Multiscale analysis*, pages 161–197. Springer.

Coifman, R. R. and Lafon, S. (2006). Diffusion maps. *Applied and Computational Harmonic Analysis*, 21(1):5 – 30. Special Issue: Diffusion Maps and Wavelets.

Coifman, R. R., Lafon, S., Lee, A. B., Maggioni, M., Nadler, B., Warner, F., and Zucker, S. W. (2005). 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.

De Lathauwer, L., De Moor, B., and Vandewalle, J. (2000). A multilinear singular value decomposition. *SIAM journal on Matrix Analysis and Applications*, 21(4):1253–1278.

Deng, L. (2012). The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142.

Fan, J. and Li, R. (2001). Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American Statistical Association*, 96(456):1348–1360.

Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2):179–188.

Frank, M. and Wolfe, P. (1956). An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3(1-2):95–110.

Gavish, M. and Coifman, R. R. (2012). Sampling, denoising and compression of matrices by coherent matrix organization. *Applied and Computational Harmonic Analysis*, 33(3):354 – 369.

Gilbert, A. C. and Jain, L. (2017). If it ain't broke, don't fix it: Sparse metric repair. *2017 55th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*.

Gilbert, A. C. and Sonthalia, R. (2018). Unsupervised metric learning in presence of missing data. *2018 56th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*.

Grother, P. J. (1995). Nist special database 19-hand-printed forms and characters database. *Technical Report, National Institute of Standards and Technology*.

Harchaoui, Z., Juditsky, A., and Nemirovski, A. (2015). Conditional gradient algorithms for norm-regularized smooth convex optimization. *Mathematical Programming*, 152(1–2):75–112.

Hardoon, D. R. and Shawe-Taylor, J. (2011). Sparse canonical correlation analysis. *Machine Learning*, 83(3):331–353.

Harper, F. M. and Konstan, J. A. (2015). The movielens datasets: History and context. *ACM transactions on interactive intelligent systems (tiis)*, 5(4):1–19.

Harshman, R. A. et al. (1970). Foundations of the parafac procedure: Models and conditions for an" explanatory" multimodal factor analysis.

Higuera, C., Gardiner, K. J., and Cios, K. J. (2015). Self-organizing feature maps identify proteins critical to learning in a mouse model of down syndrome. *PLOS ONE*, 10(6):1–28.

Hocking, T. D., Joulin, A., Bach, F., and Vert, J.-P. (2011). Clusterpath: An algorithm for clustering using convex fusion penalties. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, page 745–752, Madison, WI, USA. Omnipress.

Hoff, P. D. (2017). Lasso, fractional norm and structured sparse estimation using a hadamard product parametrization. *Computational Statistics Data Analysis*, 115:186–198.

Hotelling, H. (1936). Relations between two sets of variates. *Biometrika*, 28(3/4):321–377.

Hu, W., Shen, W., Zhou, H., and Kong, D. (2020). Matrix linear discriminant analysis. *Technometrics*, 62(2):196–205.

Hubert, L. and Arabie, P. (1985). Comparing partitions. *Journal of Classification*, 2(1):193–218.

Kalofolias, V., Bresson, X., Bronstein, M., and Vandergheynst, P. (2014). Matrix Completion on Graphs. *arXiv e-prints*, page arXiv:1408.1717.

Kolda, T. G. and Bader, B. W. (2009). Tensor decompositions and applications. *SIAM Review*, 51(3):455–500.

Ledoit, O. and Wolf, M. (2004a). Honey, i shrunk the sample covariance matrix. *The Journal of Portfolio Management*, 30(4):110–119.

Ledoit, O. and Wolf, M. (2004b). A well-conditioned estimator for large-dimensional covariance matrices. *Journal of Multivariate Analysis*, 88(2):365–411.

Lee, M., Shen, H., Huang, J. Z., and Marron, J. S. (2010). Biclustering via sparse singular value decomposition. *Biometrics*, 66(4):1087–1095.

Lee, S. H. and Choi, S. (2007). Two-dimensional canonical correlation analysis. *IEEE Signal Processing Letters*, 14(10):735–738.

Levitin, E. and Polyak, B. (1966). Constrained minimization methods. *USSR Computational Mathematics and Mathematical Physics*, 6(5):1–50.

Lin, Z., Chen, M., and Ma, Y. (2010). The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices. *arXiv preprint arXiv:1009.5055*.

Lin, Z., Ganesh, A., Wright, J., Wu, L., Chen, M., and Ma, Y. (2009). Fast convex optimization algorithms for exact recovery of a corrupted low-rank matrix. *Coordinated Science Laboratory Report no. UILU-ENG-09-2214, DC-246*.

Lindsten, F., Ohlsson, H., and Ljung, L. (2011). *Just relax and come clustering!: A convexification of k-means clustering*. Linköping University Electronic Press.

Ling, C. X. and Wang, H. (1997). Computing optimal attribute weight settings for nearest neighbor algorithms. *Artificial Intelligence Review*, 11(1):255–272.

Liu, W., Wang, Z., Liu, X., Zeng, N., Liu, Y., and Alsaadi, F. E. (2017). A survey of deep neural network architectures and their applications. *Neurocomputing*, 234:11–26.

Mai, Q. and Zhang, X. (2019). An iterative penalized least squares approach to sparse canonical correlation analysis. *Biometrics*, 75(3):734–744.

Majindar, K. N. (1963). On a factorisation of positive definite matrices. *Canadian Mathematical Bulletin*, 6(3):405–407.

Mazumder, R., Hastie, T., and Tibshirani, R. (2010). Spectral regularization algorithms for learning large incomplete matrices. *Journal of Machine Learning Research*, 11(80):2287–2322.

McLachlan, G. J. (2004). *Discriminant analysis and statistical pattern recognition*, volume 544. John Wiley & Sons.

Meyer, R. R. (1976). Sufficient conditions for the convergence of monotonic mathematical-programming algorithms. *Journal of computer and system sciences*, 12(1):108–121.

Min, E. J., Chi, E. C., and Zhou, H. (2019). Tensor canonical correlation analysis. *Stat*, 8(1):e253.

Min, K., Zhang, Z., Wright, J., and Ma, Y. (2010). Decomposing background topics from keywords by principal component pursuit. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 269–278.

Mishne, G., Chi, E., and Coifman, R. (2019). Co-manifold learning with missing data. In *Proceedings of the 36th International Conference on Machine Learning*, pages 4605–4614, Long Beach, California, USA. PMLR.

Mishne, G., Talmon, R., Cohen, I., Coifman, R. R., and Kluger, Y. (2018). Data-driven tree transforms and metrics. *IEEE Transactions on Signal and Information Processing over Networks*, 4(3):451–466.

Mishne, G., Talmon, R., Meir, R., Schiller, J., Lavzin, M., Dubin, U., and Coifman, R. R. (2016). Hierarchical coupled-geometry analysis for neuronal structure and activity pattern discovery. *IEEE Journal of Selected Topics in Signal Processing*, 10(7):1238–1253.

Mu, C., Zhang, Y., Wright, J., and Goldfarb, D. (2016). Scalable robust matrix recovery: Frank–wolfe meets proximal methods. *SIAM Journal on Scientific Computing*, 38(5):A3291–A3317.

Nadaraya, E. A. (1964). On estimating regression. *Theory of Probability & Its Applications*, 9(1):141–142.

Ng, A. Y., Jordan, M. I., and Weiss, Y. (2001). On spectral clustering: Analysis and an algorithm. In *Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic*, NIPS'01, page 849–856, Cambridge, MA, USA. MIT Press.

Pearson, K. (1901). Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin philosophical magazine and journal of science*, 2(11):559–572.

Pelckmans, K., De Brabanter, J., Suykens, J. A., and De Moor, B. L. (2005). Convex clustering shrinkage. In *PASCAL Workshop on Statistics and Optimization of Clustering Workshop*.

Rabin, N. and Fishelov, D. (2019). Two directional Laplacian pyramids with application to data imputation. *Advances in Computational Mathematics*, 45(4):2123–2146.

Ram, I., Elad, M., and Cohen, I. (2013). Image processing using smooth ordering of its patches. *IEEE Transactions on Image Processing*, 22(7):2764–2774.

Rawat, W. and Wang, Z. (2017). Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review. *Neural Computation*, 29(9):2352–2449.

Recht, B., Fazel, M., and Parrilo, P. A. (2010). Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. *SIAM review*, 52(3):471–501.

Richard, E., Savalle, P.-A., and Vayatis, N. (2012). Estimation of simultaneously sparse and low rank matrices. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, ICML'12, page 51–58.

Rothman, A. J., Levina, E., and Zhu, J. (2010). Sparse multivariate regression with covariance estimation. *Journal of Computational and Graphical Statistics*, 19(4):947–962.

Shahid, N., Perraudin, N., Kalofolias, V., Puy, G., and Vandergheynst, P. (2016). Fast robust PCA on graphs. *IEEE Journal of Selected Topics in Signal Processing*, 10(4):740–756.

Smilde, A., Bro, R., and Geladi, P. (2004). Multi way analysis — applications in chemical sciences.

Srebro, N., Rennie, J. D. M., and Jaakkola, T. S. (2004). Maximum-margin matrix factorization. In *Proceedings of the 17th International Conference on Neural Information Processing Systems*, NIPS'04, page 1329–1336, Cambridge, MA, USA. MIT Press.

Stein, J. L., Hua, X., Lee, S., Ho, A. J., Leow, A. D., Toga, A. W., Saykin, A. J., Shen, L., Foroud, T., Pankratz, N., et al. (2010). Voxelwise genome-wide association study (vgwas). *neuroimage*, 53(3):1160–1174.

Sun, T. and Chen, S. (2007). Locality preserving cca with applications to data visualization and pose estimation. *Image and Vision Computing*, 25(5):531–543.

Tao, M. and Yuan, X. (2011). Recovering low-rank and sparse components of matrices from incomplete and noisy observations. *SIAM Journal on Optimization*, 21(1):57–81.

Tew, P. A. (2016). *An investigation of sparse tensor formats for tensor libraries*. PhD thesis, Massachusetts Institute of Technology.

Tibshirani, R. J. et al. (2013). The lasso problem and uniqueness. *Electronic Journal of statistics*, 7:1456–1490.

Tsanas, A., Little, M. A., Fox, C., and Ramig, L. O. (2014). Objective automatic assessment of rehabilitative speech treatment in parkinson's disease. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 22(1):181–190.

von Luxburg, U. (2007). A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416.

Wang, W., Yang, Y., Wang, X., Wang, W., and Li, J. (2019). Development of convolutional neural network and its application in image classification: a survey. *Optical Engineering*, 58(4):040901.

Watson, G. S. (1964). Smooth regression analysis. *Sankhyā: The Indian Journal of Statistics, Series A (1961-2002)*, 26(4):359–372.

Witten, D. M. and Tibshirani, R. J. (2009). Extensions of sparse canonical correlation analysis with applications to genomic data. *Statistical applications in genetics and molecular biology*, 8(1).

Wright, J., Ganesh, A., Min, K., and Ma, Y. (2012). Compressive principal component pursuit. In *2012 IEEE International Symposium on Information Theory Proceedings*, pages 1276–1280.

Yuan, X. and Yang, J. (2013). Sparse and low-rank matrix decomposition via alternating direction method. *Pacific Journal of Optimization*, 9:167.

Zhang, C.-H. (2010). Nearly unbiased variable selection under minimax concave penalty. *The Annals of Statistics*, 38(2):894–942.

Zhang, Y., Mu, C., Kuo, H.-W., and Wright, J. (2013). Toward guaranteed illumination models for non-convex objects. In *2013 IEEE International Conference on Computer Vision*, pages 937–944.

Zheng, W., Zhou, X., Zou, C., and Zhao, L. (2006). Facial expression recognition using kernel canonical correlation analysis (kcca). *IEEE Transactions on Neural Networks*, 17(1):233–238.