NONNEGATIVE MATRIX AND TENSOR FACTORIZATIONS, LEAST SQUARES PROBLEMS, AND APPLICATIONS

A Dissertation
Presented to
The Academic Faculty

by

Jingu Kim

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in
Computer Science

School of Computational Science and Engineering
Georgia Institute of Technology
December 2011

NONNEGATIVE MATRIX AND TENSOR FACTORIZATIONS, LEAST SQUARES PROBLEMS, AND APPLICATIONS

To my family

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my advisor, Professor Haesun Park. Looking back the time when I started a Ph.D. program, Dr. Park's warm greeting was very relieving. I joined her new group at Georgia Tech, and, in the first fall semester, I learned the fundamentals of computational mathematics from her course on numerical linear algebra. Since then, she guided me into challenging research problems, encouraged me to explore various ideas, and patiently helped me improve how I do research. She led me to be an independent thinker and to own my research problem, but her advices have been helpful in deciding which road to take. Beyond academic aspects, I have also learned much from the way she communicates with me and other people. I feel very fortunate to be advised by Dr. Park.

What I learned from my committee members are invaluable. I would like to thank Professor Renato Monteiro for numerous research discussions I had with him. He shared his expertise on optimization and provided insightful comments. Dr. Monteiro's convex analysis class helped me develop understandings on the interesting subject. Interactions with Professor Hongyuan Zha were very constructive: He always pinpointed the core of research problem and suggested alternative views. Dr. Zha's course on numerical methods turned into important background for this dissertation work. Professor Alexander Gray and Professor Guy Lebanon have been my teachers on machine learning area. They not only guided me into the field with excellent lectures but also helped me think scientifically and calibrate my work for strong impacts. In addition, I thank Dr. Gray and Dr. Lebanon for allowing me to have access to their computing resources.

The School of CSE provided a pleasant environment for students to learn and work.

I thank Professor Richard Fujimoto for his energetic support to graduate students. I thank Arlene, Carolyn, Lometa, and Michael for their support and guidance.

I am also grateful to all friends and collaborators. With Jaegul and Dongryeol, I shared all adventures as a graduate student. Jaegul has been the nearest neighbor in the corner of lab, and it was enjoyable to talk about research problems or just chat about everyday lives. Dongryeol has been a fun friend and a brilliant colleague, and he taught me much about kernel machines. Nishant is a great man: Talking with him is fun and always rewarding. Daniel and Hanseung were good neighbors whom I enjoyed talking to. I thank Da and Yunlong for their friendship and collaboration. Discussions with Dr. Mariya Ishteva were fruitful and fun. I thank Seungyeon and Joonseok for their kindness and help. Discussions and collaboration with Ryder were very enjoyable. I also thank all KACB 1305 members and KSA friends for their warm friendship.

Above all, this dissertation would have been never possible without the unwavering support of my lovely wife, Jisoo, and my family in Korea. Sharing all joys and struggles with me, Jisoo patiently supported my study and work, listened to me whenever I needed, and cheered me up when I felt depressed. She has been my best friend and the best colleague, not to mention the best wife. My parents and brother, Mingu, encouraged me to pursue my passion and supported my study in every possible way. I also much appreciate my parents-in-laws for their warm supports and encouragements. Finally, I would like to dedicate this dissertation to my grandmother and my sister, Minju.

I also thank the Samsung Foundation of Culture for generous financial support as fellowship.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ANLS	Alternating nonnegative least square, p. 25.
BCD	Block coordinate descent, p. 20.
BLCP	Linear complementarity problem with bounds, p. 85.
HALS	Hierarchical alternating least squares, p. 24.
KKT	Karush-Kuhn-Tucker, p. 19.
LARS	Least angle regression, p. 84.
LCP	Linear complementarity problem, p. 43.
NCP	Nonnegative CANDECOMP/PARAFAC, p. 6.
NLS	Nonnegativity-constrained least squares, p. 5.
NMF	Nonnegative matrix factorization, p. 1.
NTF	Nonnegative tensor factorization, p. 6.
RRI	Rank-one residue iteration, p. 27.
SVD	Singular value decomposition, p. 13.

SUMMARY

Nonnegative matrix factorization (NMF) is a useful dimension reduction method that has been investigated and applied in various areas. NMF is considered for high-dimensional data in which each element has a nonnegative value, and it provides a low-rank approximation formed by factors whose elements are also nonnegative. The nonnegativity constraints imposed on the low-rank factors not only enable natural interpretation but also reveal the hidden structure of data. Extending the benefits of NMF to multidimensional arrays, nonnegative tensor factorization (NTF) has been shown to be successful in analyzing complicated data sets. Despite the success, NMF and NTF have been actively developed only in the recent decade, and algorithmic strategies for computing NMF and NTF have not been fully studied. In this thesis, computational challenges regarding NMF, NTF, and related least squares problems are addressed.

First, efficient algorithms of NMF and NTF are investigated based on a connection from the NMF and the NTF problems to the nonnegativity-constrained least squares (NLS) problems. A key strategy is to observe typical structure of the NLS problems arising in the NMF and the NTF computation and design a fast algorithm utilizing the structure. We propose an accelerated block principal pivoting method to solve the NLS problems, thereby significantly speeding up the NMF and NTF computation. Implementation results with synthetic and real-world data sets validate the efficiency of the proposed method.

In addition, a theoretical result on the classical active-set method for rank-deficient NLS problems is presented. Although the block principal pivoting method appears generally more efficient than the active-set method for the NLS problems, it is not applicable for rank-deficient cases. We show that the active-set method with a proper starting vector can actually solve the rank-deficient NLS problems without ever running into rank-deficient least squares problems during iterations.

Going beyond the NLS problems, it is presented that a block principal pivoting strategy can also be applied to the l_1-regularized linear regression. The l_1-regularized linear regression, also known as the Lasso, has been very popular due to its ability to promote sparse solutions. Solving this problem is difficult because the l_1-regularization term is not differentiable. A block principal pivoting method and its variant, which overcome a limitation of previous active-set methods, are proposed for this problem with successful experimental results.

Finally, a group-sparsity regularization method for NMF is presented. A recent challenge in data analysis for science and engineering is that data are often represented in a structured way. In particular, many data mining tasks have to deal with group-structured prior information, where features or data items are organized into groups. Motivated by an observation that features or data items that belong to a group are expected to share the same sparsity pattern in their latent factor representations, We propose mixed-norm regularization to promote group-level sparsity. Efficient convex optimization methods for dealing with the regularization terms are presented along with computational comparisons between them. Application examples of the proposed method in factor recovery, semi-supervised clustering, and multilingual text analysis are presented.

CHAPTER I

INTRODUCTION

Matrix factorization has been one of the most fundamental tools in machine learning, data mining, and other areas of computational science and engineering. Singular value decomposition (SVD) and principal component analysis (PCA), for example, have been standard factorization methods that have played a pivotal role. Low-rank approximations based on SVD or PCA provide a compact representation of a large matrix, and the compact representation not only enables data compression and dimension reduction but also discovers latent structure from multivariate statistical data. The latent structure has been utilized in numerous applications: For example, the low-rank approximation of term-document matrices based on SVD has been known as latent semantic indexing, which significantly improves information retrieval and text categorization.

Recently, nonnegative matrix factorization (NMF) emerged as a useful factorization method. NMF was earlier introduced by Paatero and Tapper [85] as positive matrix factorization and subsequently popularized with a seminal paper by Lee and Seung [64]. A distinguishing feature of NMF is the requirement of nonnegativity: NMF is considered for high-dimensional and large scale data in which the representation of each element is inherently nonnegative, and it seek low-rank factor matrices that are constrained to have only nonnegative elements. There are many examples of data with nonnegative representation. In a standard term-frequency encoding [77], a text document is represented as a vector of nonnegative numbers since each element represents the number of appearances of each term in the document. In image processing, digital images are represented by pixel intensities, which can be only nonnegative. In sciences, chemical concentrations or gene expression levels are nonnegatively represented.

To introduce the main idea of NMF, let us consider a matrix $\mathbf{A} \in \mathbb{R}^{M \times N}$, in which the rows represent features and the columns represent data items. Suppose a low-rank approximation if $\mathbf{A}$ is given by two factor matrices $\mathbf{W} \in \mathbb{R}^{M \times K}$ and $\mathbf{H} \in \mathbb{R}^{N \times K}$ with a small integer $K < \min\{M, N\}$:

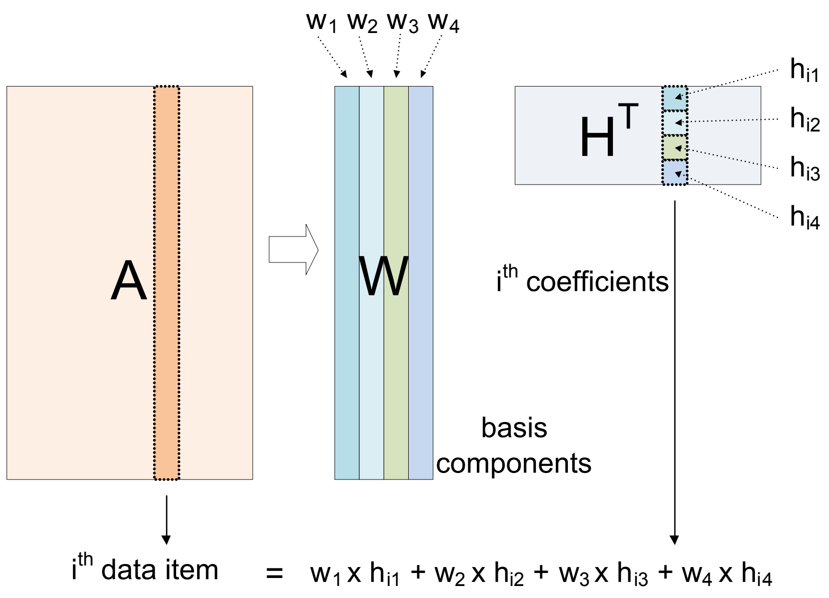
$$\mathbf{A} \approx \mathbf{W}\mathbf{H}^T. \tag{1.1}$$

An illustration of this factorization model is given in Figure 1.1. Matrices $\mathbf{W}$ and $\mathbf{H}$ are commonly interpreted as *basis vectors* and reconstruction *coefficients*, respectively. With this interpretation, each data item is understood as an approximation given as

$$\mathbf{a}_i \approx \sum_{k=1}^{K} \mathbf{w}_k h_{ik},$$

where $\mathbf{a}_i$, $\mathbf{w}_k$, and h_{ik} denote the i^{th} column of $\mathbf{A}$, the k^{th} column of $\mathbf{W}$, and the $(i,k)^{th}$ element of $\mathbf{H}$, respectively. That is, the i^{th} data item represented by $\mathbf{a}_i$ is composed of a linear combination of basis components $\mathbf{w}_1, \cdots, \mathbf{w}_K$ with coefficients $h_{i1}, \cdots, h_{iK}$.

Now, for a $\mathbf{A} \in \mathbb{R}^{M \times N}$ that contain only nonnegative elements, such as text documents or images with pixel intensities, a key idea of NMF is to take advantage of the inherent nonnegativity by enforcing that low-rank factor matrices are themselves nonnegative. The fact that $\mathbf{W}$ and $\mathbf{H}$ are element-wise nonnegative enables natural interpretations of the approximation model in Eq. (1.1). First, the nonnegativity of basis factor $\mathbf{W}$ enforces that each basis component, which is each column of $\mathbf{W}$, is a *physically meaningful instance* of original data type. If $\mathbf{w}_k$ contains a nonnegative element, it does not represent a text document or a digital image any more. In addition, the nonnegativity of $\mathbf{H}$ implies that each data item can be explained by an *additive* linear combination of basis components, as opposed to an additive and subtractive combination. The additive combination naturally represents the actual



w_1 w_2 w_3 w_4
A
W
H^T
h_{i1}
h_{i2}
h_{i3}
h_{i4}
i^{th} coefficients
basis components
i^{th} data item = $w_1 \times h_{i1} + w_2 \times h_{i2} + w_3 \times h_{i3} + w_4 \times h_{i4}$

Figure 1.1: Matrix factorization, basis components, and reconstruction model

interaction of real-world objects, in which a subtraction does not have a direct interpretation. Combining the two advantages, Lee and Seung [64] reported that a *part-based representation* can be discovered with NMF. The nonnegativity of **W** ensures that its column is a meaningful data type, that can be interpreted as a 'part'. The nonnegativity of **H** ensures that the parts can only be combined additively without subtractions.

The benefits of the nonnegative latent factor matrices have been recognized in many applications. In text mining [88, 103, 92], NMF was shown to detect latent topics from a large text corpus, and the discovered topics improved text categorization and topic tracking. In computer vision [68, 48], NMF was used to analyze a collection of facial images for feature extraction. In bioinformatics [16, 27, 55] NMF has been used to analyze gene and protein expression microarray data. In signal processing, NMF has been shown successful for blind source separation [26] and music analysis [33]. In data mining, the use of NMF for clustering has been actively investigated [29, 58].

NMF has become an indispensable tool in numerous data analysis problems. On the computational side, the computation of NMF is more demanding than that of conventional factorization methods such as SVD. Since NMF has been developed only for a decade, studies on its theoretical properties and algorithmic strategies have been only in beginning. In this thesis, algorithmic and computational challenges of NMF, its extensions, and constrained least squares problems are addressed. Mathematical treatments of NMF, problem formulations, and the contributions of this thesis are summarized in the following. Connections to the main body of this thesis are made whenever needed.

1.1 Nonnegative Matrix Factorization

A mathematical formulation of NMF is given as follows. Suppose a nonnegative matrix $\mathbf{A} \in \mathbb{R}^{M \times N}$ is given. With an integer $K < \min\{M, N\}$, the goal of NMF is to find two nonnegative factors $\mathbf{W} \in \mathbb{R}^{M \times K}$ and $\mathbf{H} \in \mathbb{R}^{N \times K}$ that satisfy Eq. (1.1). The goodness of the approximation can be measured in various ways including Frobenius norm, Kullback-Leibler divergence [65], and Bregman divergence [28]. This thesis is focused on the most common choice, which is Frobenius norm. The factors $\mathbf{W}$ and $\mathbf{H}$ are then found by solving the optimization problem:

$$\min_{\mathbf{W},\mathbf{H}} f(\mathbf{W},\mathbf{H}) = \frac{1}{2}\left\|\mathbf{A} - \mathbf{W}\mathbf{H}^T\right\|_F^2, \qquad (1.2)$$

$$\text{subject to } \mathbf{W}, \mathbf{H} \geq 0 \text{ element-wise.}$$

Due to nonnegativity constraints, Eq. (1.2) cannot be solved by traditional methods such as SVD. Eq. (1.2) is a non-convex optimization problem with respect to variables $\mathbf{W}$ and $\mathbf{H}$, and it is shown that solving NMF is NP-hard [100]. Therefore, a good algorithm is expected to compute a local minimum of Eq. (1.2).

In this thesis, an efficient algorithm for Eq. (1.2) is presented. A block coordinate descent algorithm for Eq. (1.2), known as the alternating nonnegative least squares framework, is recently shown theoretically sound and empirically efficient [56, 71]. A key component of this framework is an efficient algorithm to solve the nonnegativity-constrained least squares (NLS) problems. The NLS problem is generally written as

$$\min_{\mathbf{X} \geq 0} \|\mathbf{B}\mathbf{X} - \mathbf{C}\|_F^2 , \tag{1.3}$$

where $\mathbf{B} \in \mathbb{R}^{p \times q}$, $\mathbf{C} \in \mathbb{R}^{p \times r}$, and $\mathbf{X} \in \mathbb{R}^{q \times r}$. In Chapter 4, a fast algorithm for NMF is proposed using the block principal pivoting method [53] that efficiently solves the NLS subproblems in Eq. (1.3). The block principal pivoting method is similar to the classical active-set method due to Lawson and Hanson [63, 56], but it becomes more scalable than the classical one by exchanging multiple variables at a time. Observations on the typical structure of the NLS problems arising in the NMF computation are summarized, and an accelerated block principal pivoting method is designed exploiting the typical structure. Thorough experimental comparisons with competing numerical methods for NMF are conducted on synthetic and real-world data sets, and the proposed algorithm is shown to be a state-of-the-art method.

The material of Chapter 4 is partially based the following publication.

- Jingu Kim and Haesun Park, "Fast Nonnegative Matrix Factorization: An Active-set-like Method and Comparisons", *SIAM Journal on Scientific Computing*, To appear.

- Jingu Kim and Haesun Park, "Toward Faster Nonnegative Matrix Factorization: A New Algorithm and Comparisons," in *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining (ICDM)*, pp. 353-362, 2008.

1.2 Nonnegative Tensor Factorization

A tensor is a multi-dimensional array, of which a matrix is a two-dimensional special case. Whereas a matrix can only represent data lying in an outer product of two vector spaces, a tensor can flexibly represent data in an outer product of multiple vector spaces. The flexibility allows tensor-based methods to cope with complex data sets, which cannot be handled with matrix-based methods. The rank of a tensor can be defined analogously with the rank of a matrix, and various types of the low-rank approximations of tensors have been studied.

In this thesis, we consider nonnegative tensor factorization (NTF) in which the low-rank representation is constrained to be element-wise nonnegative. In particular, we address the nonnegative CANDECOMP/PARAFAC (NCP) decomposition [62]. There are other approximation models for higher order tensors as discussed in [62], but, in this thesis, whenever we mention nonnegative tensor factorization (NTF) for simplicity, we refer to NCP. Given an N-way tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{M_1 \times \cdots \times M_N}$ and an integer K, the problem of NCP is to find loading matrices $\mathbf{H}^{(1)} \in \mathbb{R}^{M_1 \times K}, \cdots, \mathbf{H}^{(N)} \in \mathbb{R}^{M_N \times K}$ by solving an optimization problem:

$$\min_{\mathbf{H}^{(1)},\cdots,\mathbf{H}^{(N)}} \frac{1}{2} \left\| \boldsymbol{\mathcal{A}} - \sum_{k=1}^{K} \mathbf{h}_k^{(1)} \circ \cdots \circ \mathbf{h}_k^{(N)} \right\|_F^2, \tag{1.4}$$
$$\text{subject to } \mathbf{H}^{(1)}, \cdots, \mathbf{H}^{(N)} \geq 0 \text{ element-wise,}$$

where $\mathbf{h}_k^{(n)}$ represents the k^{th} column vector of factor matrix $\mathbf{H}^{(n)}$, and $\circ$ represents an outer product. In Eq. (1.4), we want to approximate the given tensor $\boldsymbol{\mathcal{A}}$ by the sum of K rank-one tensors. Observe that Eq. (1.4) with $N = 2$ reduces to the NMF problem shown in Eq. (1.2). Nonnegativity-constraints imposed on the factor matrices enable benefits similar to that of NMF in image processing [93, 102] and text mining [5].

In Chapter 5, a fast algorithm for Eq. (1.4) is presented. A block coordinate descent algorithm for Eq. (1.4) again appears as a sequence of nonnegativity-constrained least squares (NLS) problems. Utilizing the accelerated block principal pivoting algorithm for NLS problems, Eq. (1.4) can be efficiently computed. Experiments with synthetic and real-world data sets show that the proposed method outperforms existing methods.

The material of Chapter 5 is based the following publication.

- Jingu Kim and Haesun Park, "Fast Nonnegative Tensor Factorization with an Active-set-like Method", *High-Performance Scientific Computing: Algorithms and Applications*, Springer, To appear.

1.3 L_1-regularized Linear Regression

In development of efficient algorithms for NMF and NTF, a fast method for the NLS problems play an important role. In particular, the block principal pivoting method is used in this thesis to construct fast NMF and NTF algorithms having state-of-the-art efficiency. Interestingly, it is discovered that a variant of the block principal pivoting method can be used to solve l_1-regularized linear regression, which is another very important least squares problem that received much attention recently.

L_1-regularized linear regression, also known as the Lasso [96], is an effective method that improves generalization and feature selection. By constraining the l_1-norm of the coefficient vector, this method simultaneously avoids over-fitting to training data and achieves sparsity in obtained coefficients. The sparsity has two important benefits; it improves the interpretation of a linear model by explicitly showing the relationship between the response and the features [96], and it also allows computationally efficient models because only a small number of coefficients remain nonzero. In signal processing, l_1-norm has been successfully used as a penalty term in a technique called 'basis pursuit denoising' [22], and a related topic on 'compressed sensing' [18, 30] has been a very active research area.

Researchers used l_1-regularization for many other learning problems including logistic regression [67], graphical model selection [6, 38], principal component analysis [106], and sparse coding [66, 76]. Although some researchers designed specialized algorithms for those problems, many utilized l_1-regularized linear regression as a subproblem to solve their intended sparse learning tasks. Hence, an efficient algorithm for l_1-regularized linear regression is important not only in its own right but also for those extended sparse learning problems.

Suppose that data are given as $(\mathbf{x}^{(n)}, y^{(n)})_{n=1}^{N}$ where $y^{(n)} \in \mathbb{R}$ is the response of $\mathbf{x}^{(n)} \in \mathbb{R}^P$. L_1-regularized linear regression can be written as

$$\min_{\boldsymbol{\beta} \in \mathbb{R}^P} \mathcal{L}(\boldsymbol{\beta}, \lambda) = \frac{1}{2} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda \|\boldsymbol{\beta}\|_1 , \tag{1.5}$$

where $\mathbf{X} = \left(\mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(N)}\right)^T$, $\mathbf{y} = \left(y^{(1)}, \cdots, y^{(N)}\right)^T$, and $\lambda \geq 0$. In Chapter 6, a new active-set-like method for Eq. (1.5) is presented. By showing that the optimality conditions of Eq. (1.5) appear in a form of the linear complementarity problem with bounds, we present an efficient block principal pivoting method, which overcomes some difficulties of existing methods such as least angle regression. Implementation results as well as extensions to the structure learning of Gaussian graphical models are demonstrated as well.

The material of Chapter 6 is based the following publication.

- Jingu Kim and Haesun Park, "Fast Active-set-type Algorithms for L_1-regularized Linear Regression," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS) 2010*, JMLR: W&CP 9:397-404, 2010.

1.4 Rank Deficiency

Active-set-like methods for solving the NLS problems, including the classical active-set method and the block principal method, is at the core of NMF and NTF algorithms studied in this thesis. Consider an NLS problem written as

$$\min_{\mathbf{x} \geq 0} \|\mathbf{B}\mathbf{x} - \mathbf{c}\|_F^2 , \tag{1.6}$$

where $\mathbf{B} \in \mathbb{R}^{M \times N}$, $\mathbf{c} \in \mathbb{R}^{M \times 1}$, and $\mathbf{x} \in \mathbb{R}^{N \times 1}$. There are trade-offs between active-set-like methods in handling Eq. (1.6) for the cases that $\mathbf{B}$ is of full column rank and that $\mathbf{B}$ is rank-deficient. The block principal pivoting method appears generally more efficient for the full-rank case, but it might break down when $\mathbf{B}$ is rank-deficient. It has been believed that the classical active-set method by Lawson and Hanson [63] handles the rank-deficient case; however, no supporting arguments were given to the behaviour of active-set method for rank-deficient NLS problems.

In Chapter 7, a theoretical discovery that the active-set method with a proper starting vector can actually solve the rank-deficient NLS problems is presented. An in-depth analysis of the active-set method is presented, providing additional insights on its properties.

The material of Chapter 7 is partially based the following publication.

- Barry Drake, Jingu Kim, Mahendra Mallick and Haesun Park, "Supervised Raman Spectra Estimation based on Nonnegative Rank Deficient Least Squares," in *Proceedings of the 13th International Conference on Information Fusion*, Edinburgh, UK, 2010.

1.5 Group Sparsity in Nonnegative Matrix Factorization

A recent challenge in data analysis for science and engineering is that data are often represented in a structured way. In particular, many data mining tasks have to deal with group-structured prior information, where features or data items are organized into groups. However, a principled approach to incorporating group information into NMF has been lacking in the literature. In Chapter 8, group-sparsity regularization methods for NMF is developed. Motivated by an observation that features or data items within a group are expected to share the same sparsity pattern in their latent factor representation, we propose mixed-norm regularization to promote group sparsity in the factor matrices of NMF.

Suppose the columns of $\mathbf{A} \in \mathbb{R}^{M\times N}$ are divided into B groups as $\mathbf{A} = \left(\mathbf{A}^{(1)}, \cdots, \mathbf{A}^{(B)}\right)$, where $\mathbf{A}^{(b)} \in \mathbb{R}^{M\times N_b}$ and $\sum_{b=1}^{B} N_b = N$. Accordingly, the coefficient matrix is divided into B groups as $\mathbf{H}^T = \left((\mathbf{H}^{(1)})^T, \cdots, (\mathbf{H}^{(B)})^T\right)$ where $\mathbf{H}^{(b)} \in \mathbb{R}^{N_b\times K}$. The proposed formulation is written as follows:

$$\min_{\mathbf{W}\geq 0,\mathbf{H}\geq 0} \frac{1}{2}\left\|\mathbf{A}-\mathbf{W}\mathbf{H}^T\right\|_F^2 + \alpha\left\|\mathbf{W}\right\|_F^2 + \beta\sum_{b=1}^{B}\left\|(\mathbf{H}^{(b)})^T\right\|_{1,q}. \quad (1.7)$$

$L_{1,q}$-norm, which is commonly called a mixed-norm, is defined in Chapter 8. The mixed-norm regularization has been shown to promote group-level sparsity [104, 74, 90]. Efficient convex optimization methods for solving Eq. (1.7) are presented along with computational comparisons between them. Application examples of the proposed method in factor recovery, semi-supervised clustering, and multilingual text analysis are demonstrated.

1.6 Structure of Thesis and Notations

The rest of this thesis is organized as follows.

- In Chapter 2, background materials on numerical linear algebra and convex optimization are introduced.
- In Chapter 3, algorithms for nonnegative matrix factorization are reviewed.
- In Chapter 4, a fast algorithm for nonnegative matrix factorization based on an accelerated block principal pivoting method is presented.
- In Chapter 5, a fast algorithm for nonnegative CANDECOMP/PARAFAC decomposition based on an accelerated block principal pivoting method is presented.
- In Chapter 6, fast block principal pivoting algorithms for l_1-regularized linear regression are presented.
- In Chapter 7, a theoretical discovery that the active-set method with a proper starting vector can solve the rank-deficient nonnegativity-constrained least squares problems is presented.

- In Chapter 8, a nonnegative matrix factorization method that promotes group sparsity using mixed-norm regularization method is presented.
- In Chapter 9, concluding remarks with the summary of contributions as well as suggestions on future directions are made.

Throughout the thesis, the following mathematical notations are used.

- (**Letters**)

  A lower-case or an upper-case letter, such as x or X, denotes a scalar. Boldface lower-case and upper-case letters, such as $\mathbf{x}$ and $\mathbf{X}$, denote a vector and a matrix, respectively. A boldface Euler script letter, such as $\boldsymbol{\mathcal{X}}$, denotes a tensor of order three or higher.

- (**Indices and sequence**)

  Indices typically grow from 1 to an upper-case letter, e.g., $n \in \{1, \cdots, N\}$. Elements of a sequence are denoted by superscripts within parentheses, e.g., $\mathbf{X}^{(1)}, \cdots, \mathbf{X}^{(N)}$, and the entire sequence is denoted by $\left\{\mathbf{X}^{(n)}\right\}$.

- (**Subvectors and elements**)

  For a matrix $\mathbf{X}$, $(\mathbf{X})_{\cdot i}$, $\mathbf{x}_{\cdot i}$, or $\mathbf{x}_i$ denotes its i^{th} column, $(\mathbf{X})_{i\cdot}$ or $\mathbf{x}_{i\cdot}$ denotes its i^{th} row, and x_{ij} denotes its $(i,j)^{th}$ element.

- (**Nonnegativity**)

  The set of nonnegative real numbers is denoted by $\mathbb{R}_+$, and $\mathbf{X} \geq 0$ indicates that the elements of $\mathbf{X}$ are nonnegative. The notation $[\mathbf{X}]_+$ is used to denote a matrix that is the same as $\mathbf{X}$ except that all its negative elements are set as zero. A nonnegative matrix or a nonnegative tensor refers to a matrix or a tensor with only nonnegative elements.

- (**Built-in matrices and vectors**)

  $\mathbf{1}_{M \times N}$ and $\mathbf{0}_{M \times N}$ denote matrices of size $M \times N$ in which every elements are one and zero, respectively. Similarly, $\mathbf{1}_M$ and $\mathbf{0}_M$ denote vectors of length M in which every elements are one and zero, respectively. $\mathbf{I}_M$ denotes an identity matrix of size $M \times M$.

- (**Cartesian product**)

  An operator '$\times$' denotes the Cartesian product.

CHAPTER II

PRELIMINARIES

Materials on numerical linear algebra and convex optimization that are relevant to this thesis are briefly summarized in this chapter.

2.1 Matrix Analysis

2.1.1 Singular value decomposition

Singular value decomposition (SVD) is a general and very strong property developed in matrix analysis. Detailed introduction to SVD as well as the proofs of the following theorems can be found in, e.g., Golub and Van Loan [42], Trefethen and Bau [97], and Horn and Johnson [47],

Theorem 2.1. *(Singular Value Decomposition) For any matrix* $\mathbf{A} \in \mathbb{R}^{M\times N}$*, there exist orthogonal matrices*

$$\mathbf{U} = \begin{bmatrix} \mathbf{u}_1 & \cdots & \mathbf{u}_M \end{bmatrix} \in \mathbb{R}^{M\times M} \text{ and } \mathbf{V} = \begin{bmatrix} \mathbf{v}_1 & \cdots & \mathbf{v}_N \end{bmatrix} \in \mathbb{R}^{N\times N}$$

such that

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T,$$

where $\mathbf{\Sigma} = diag(\sigma_1, \cdots, \sigma_P) \in \mathbb{R}^{M\times N}$ *with* $P = \min\{M, N\}$ *and*

$$\sigma_1 \geq \cdots \geq \sigma_P \geq 0.$$

Columns of $\mathbf{U}$ (that is, $\mathbf{u}_1, \cdots, \mathbf{u}_M$) are called the left singular vectors; columns of $\mathbf{V}$ (that is, $\mathbf{v}_1, \cdots, \mathbf{v}_N$) are called the right singular vectors; the diagonal values of $\mathbf{\Sigma}$ (that is, $\sigma_1, \cdots, \sigma_P$) are called the singular values. If the rank of $\mathbf{A}$ is R, the singular values satisfy

$$\sigma_1 \geq \cdots \geq \sigma_R \geq \sigma_{R+1} = \cdots = \sigma_P = 0,$$

and the SVD can be more compactly written as

$$\mathbf{A} = \tilde{\mathbf{U}}\tilde{\mathbf{\Sigma}}\tilde{\mathbf{V}}^T, \tag{2.1}$$

where

$$\begin{aligned}
\tilde{\mathbf{U}} &= \begin{bmatrix} \mathbf{u}_1 & \cdots & \mathbf{u}_R \end{bmatrix} \in \mathbb{R}^{M\times R}, \\
\tilde{\mathbf{V}} &= \begin{bmatrix} \mathbf{v}_1 & \cdots & \mathbf{v}_R \end{bmatrix} \in \mathbb{R}^{N\times R}, \\
\tilde{\mathbf{\Sigma}} &= \operatorname{diag}(\sigma_1, \cdots, \sigma_R) \in \mathbb{R}^{R\times R}.
\end{aligned}$$

One of the most important applications of SVD is its use for low-rank matrix approximations.

Theorem 2.2. *Suppose the SVD of a matrix* $\mathbf{A}$ *with rank* R *is given as Eq.* (2.1)*. For any integer* K *with* $0 \le K < R$*, define the rank-K SVD as*

$$\tilde{\mathbf{A}}_K = \sum_{k=1}^{K} \sigma_k \mathbf{u}_k \mathbf{v}_k^T. \tag{2.2}$$

Then,

$$\begin{aligned}
\left\| \mathbf{A} - \tilde{\mathbf{A}}_K \right\|_2 &= \inf_{\mathbf{B}\in\mathbb{R}^{M\times N}, rank(\mathbf{B})=K} \|\mathbf{A}-\mathbf{B}\|_2 = \sigma_{K+1}, \\
\left\| \mathbf{A} - \tilde{\mathbf{A}}_K \right\|_F &= \inf_{\mathbf{B}\in\mathbb{R}^{M\times N}, rank(\mathbf{B})=K} \|\mathbf{A}-\mathbf{B}\|_F = \sqrt{\sigma_{K+1}^2 + \cdots + \sigma_R^2}.
\end{aligned}$$

Theorem 2.2 states that the best rank-K approximation of $\mathbf{A}$ in terms of minimizing the l_2-norm or the Frobenius norm of the residual matrix is the rank-K SVD defined in Eq. (2.2).

2.1.2 Nonnegative matrices

Perron-Frobenius theorem [8, 47] is a well-known property of positive and nonnegative matrices. For our discussion, the following extended theorem shown in Chapter 2 of Berman and Plemmons [8] is convenient.

Theorem 2.3. *Consider a square matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$. If $\mathbf{A} \geq 0$, the eigenvalue of $\mathbf{A}$ with the largest magnitude is nonnegative, and there exists a nonnegative eigenvector corresponding to the largest eigenvalue.*

A direct consequence of Theorem 2.3 is the nonnegativity of the best rank-one approximation.

Corollary 2.4. *Consider a nonnegative matrix $\mathbf{A} \in \mathbb{R}^{M \times N}$ and the following minimization problem:*

$$\min_{\mathbf{u} \in \mathbb{R}^N, \mathbf{v} \in \mathbb{R}^N} \left\| \mathbf{A} - \mathbf{u}\mathbf{v}^T \right\|_F^2 . \tag{2.3}$$

There exists an optimal solution of Eq. (2.3) *satisfying $\mathbf{u} \geq 0$ and $\mathbf{v} \geq 0$.*

Another way to realizing Corollary 2.4 is using the SVD. Observing that for a nonnegative matrix $\mathbf{A} \geq 0$,

$$\left\| \mathbf{A} - \mathbf{u}\mathbf{v}^T \right\|_F^2 = \sum_{m=1}^{M} \sum_{n=1}^{N} \left(a_{mn} - u_m v_n \right)^2 \geq \sum_{m=1}^{M} \sum_{n=1}^{N} \left(a_{mn} - |u_m| \, |v_n| \right)^2 ,$$

for any vectors $\mathbf{u} \in \mathbb{R}^M$ and $\mathbf{v} \in \mathbb{R}^N$, element-wise absolute values can be taken from the first singular value and corresponding singular vectors to achieve the best rank-one approximation satisfying nonnegativities. There might be other optimal solutions of Eq. (2.3) involving negative numbers: See [39].

2.2 Optimization Theory

Two duality relationships, namely, Lagrangian duality and Fenchel duality, are used in this thesis to understand and efficiently handle difficult optimization problems. Main results of the duality relationships are briefly introduced in this section. Duality theory plays a central role in convex optimization and has strong implications across wide applications. Rockafellar [91], Boyd and Vandenberghe [14], Bertsekas et al. [11], and Borwein and Lewis [13] describe the material in detail.

Before we discuss duality, a few basic notions of convex analysis are needed.

Definition 2.5. (Affine hull and relative interior) For a subset $\mathcal{C}$ of $\mathbb{R}^N$, its affine hull and relative interior are defined as, respectively,

$$\operatorname{aff}\mathcal{C} = \left\{ \alpha_1\mathbf{x}_1 + \cdots + \alpha_K\mathbf{x}_K | \mathbf{x}_1, \cdots, \mathbf{x}_K \in \mathcal{C}, \sum_{k=1}^{K} \alpha_k = 1 \right\}, \text{ and}$$

$$\operatorname{ri}\mathcal{C} = \left\{ \mathbf{x} \in \mathcal{C} | \exists t > 0 \text{ such that } \left\{ \mathbf{y} \in \mathbb{R}^N | \mathbf{y} \in \operatorname{aff}\mathcal{C}, \|\mathbf{x} - \mathbf{y}\| < t \right\} \subseteq \mathcal{C} \right\}.$$

Convexity of a set, defined as follows, is the most elementary property in convex analysis.

Definition 2.6. (Convex set) A subset $\mathcal{C}$ of $\mathbb{R}^N$ is convex if, for $\forall \mathbf{x}, \mathbf{y} \in \mathcal{C}$ and $0 < \lambda < 1$, $(1 - \lambda)\mathbf{x} + \lambda\mathbf{y} \in \mathcal{C}$.

For easier discussion on convex functions, an extended definition of a function that is allowed to take $-\infty$ or ∞ is commonly used. Note that a function $f : \mathcal{X} \to \mathbb{R}$, where $\mathcal{X} \in \mathbb{R}^N$, can be straightforwardly extended to $\tilde{f} : \mathbb{R}^N \to \mathbb{R} \cup \{-\infty, \infty\}$ by a transformation

$$\tilde{f}(\mathbf{x}) = \begin{cases} f(\mathbf{x}), & \text{if } \mathbf{x} \in \mathcal{X}, \\ \infty, & \text{otherwise.} \end{cases}$$

For extended functions, two properties are defined as follows.

Definition 2.7. (Domain and epigraph) For a function $f : \mathbb{R}^N \to \mathbb{R} \cup \{-\infty, \infty\}$, its domain and epigraph is defined as

$$\begin{aligned} \operatorname{dom} f &= \left\{ \mathbf{x} \in \mathbb{R}^N : f(\mathbf{x}) < \infty \right\}, \\ \operatorname{epi} f &= \left\{ (\mathbf{x}, \mu) \in \mathbb{R}^N \times \mathbb{R} : f(\mathbf{x}) \le \mu \right\} \end{aligned}$$

The epigraph of a function is used to characterize the convexity of a function, as follows.

Definition 2.8. (Convex function) A function $f : \mathbb{R}^N \to \mathbb{R} \cup \{-\infty, \infty\}$ is said convex if its epigraph, epi f, is a convex subset of $\mathbb{R}^{N+1}$.

With these definitions, let us see Lagrangian duality first.

2.2.1 Lagrangian duality

Consider an optimization problem:

$$\begin{aligned} &\text{minimize } f(\mathbf{x}) \text{ with } \mathbf{x} \in \mathbb{R}^N \\ &\text{subject to } \mathbf{x} \in \mathcal{X}, \\ &\qquad\qquad g_i(\mathbf{x}) \le 0,\ \forall i \in \{1, \cdots, I\}, \\ &\qquad\qquad h_j(\mathbf{x}) = 0,\ \forall j \in \{1, \cdots, J\}. \end{aligned} \tag{2.4}$$

where $\mathcal{X}$ is a non-empty subset of $\mathbb{R}^N$, and f, all g_i's, and h_j's are functions from $\mathcal{X}$ to $\mathbb{R}$. Lagrangian function $\mathcal{L} : \mathcal{X} \times \mathbb{R}^I \times \mathbb{R}^J \to \mathbb{R}$ for Eq. (2.4) is defined with $\mathbf{x} \in \mathcal{X}$, $\boldsymbol{\lambda} \in \mathbb{R}^I$, and $\boldsymbol{\mu} \in \mathbb{R}^J$ as

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) + \sum_{i=1}^{I} \lambda_i g_i(\mathbf{x}) + \sum_{j=1}^{J} \mu_j h_j(\mathbf{x}). \tag{2.5}$$

Observe that the optimal value p^* of Eq. (2.4), which is called the *primal* problem, can be represented with the Lagrangian function as

$$p^* = \inf_{\mathbf{x} \in \mathcal{X}} \sup_{\boldsymbol{\lambda} \in \mathbb{R}^I, \boldsymbol{\lambda} \ge 0, \boldsymbol{\mu} \in \mathbb{R}^J} \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}). \tag{2.6}$$

The dual of this problem is obtained by exchanging the order of inf and sup. Defining the Lagrange dual function $\phi : \mathbb{R}^I \times \mathbb{R}^J \to [-\infty, \infty)$

$$\phi(\boldsymbol{\lambda}, \boldsymbol{\mu}) = \inf_{\mathbf{x} \in \mathcal{X}} \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}),$$

the Lagrange *dual* problem of Eq. (2.4) is then written as

$$\begin{aligned} &\text{maximize } \phi(\boldsymbol{\lambda}, \boldsymbol{\mu}) \text{ with } \boldsymbol{\lambda} \in \mathbb{R}^I, \boldsymbol{\mu} \in \mathbb{R}^J \\ &\text{subject to } \boldsymbol{\lambda} > 0. \end{aligned} \tag{2.7}$$

The optimal value d^* of Eq. (2.7) is expressed as

$$d^* = \sup_{\boldsymbol{\lambda} \in \mathbb{R}^I, \boldsymbol{\lambda} \ge 0, \boldsymbol{\mu} \in \mathbb{R}^J} \inf_{\mathbf{x} \in \mathcal{X}} \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}), \tag{2.8}$$

The following theorem states relationships between the optimal values p^* and d^* of, respectively, the primal and the dual problems.

Theorem 2.9. *(Lagrangian duality) Suppose that $\mathcal{X}$ is a nonempty subset of $\mathbb{R}^N$ and that f, all g_i's, and all h_i's are functions from $\mathcal{X}$ to $\mathbb{R}$. If p^* and d^* respectively represent the infimum and the supremum of Eq.* (2.6) *and Eq.* (2.8)*, then the following weal duality holds:*

$$d^* \leq p^*.$$

If $\mathcal{X}$ is a convex set, f, all g_i's, and all h_i's are convex functions on $\mathcal{X}$, and there exist $\bar{\mathbf{x}} \in ri\, \mathcal{X}$ such that $g_i(\bar{\mathbf{x}}) < 0$ for $\forall i \in \{1, \cdots, I\}$, then the following strong duality holds:

$$d^* = p^*.$$

Constraint qualifications for the strong duality in Theorem 2.9 is known as *Slater's conditions*. A common usage of the Lagrangian duality is that, when Slater's conditions are satisfied, we can transform the primal problem into the dual problem and apply optimization methods to the dual problem instead of solving the primal problem directly.

2.2.2 Fenchel duality

Fenchel duality deals with the conjugate of convex (or concave) functions. For the discussion of Fenchel duality, let us define proper functions first.

Definition 2.10. A convex function $f : \mathbb{R}^N \to \mathbb{R} \cup \{-\infty, \infty\}$ is called proper if $f(\mathbf{x}) > -\infty$ for $\forall \mathbf{x} \in \mathbb{R}^N$ and there exists $\tilde{\mathbf{x}} \in \mathbb{R}^N$ such that $f(\tilde{\mathbf{x}}) < \infty$. A function $g : \mathbb{R}^N \to \mathbb{R} \cup \{-\infty, \infty\}$ is called proper concave if $-g$ is proper convex.

Conjugacy of convex and concave functions is characterized as follows.

Definition 2.11. Let $f : \mathbb{R}^N \to \mathbb{R} \cup \{\infty\}$ and $g : \mathbb{R}^N \to \mathbb{R} \cup \{-\infty\}$ be proper convex and proper concave functions, respectively. The convex conjugate $f^* : \mathbb{R}^N \to \mathbb{R} \cup \{\infty\}$ of f is defined by

$$f^*(\mathbf{y}) = \sup_{\mathbf{x} \in \mathbb{R}^N} \{\langle \mathbf{y}, \mathbf{x} \rangle - f(\mathbf{x})\}, \tag{2.9}$$

and the concave conjugate of $g : \mathbb{R}^N \to \mathbb{R} \cup \{-\infty\}$ of g is defined by

$$g^*(\mathbf{y}) = \inf_{\mathbf{x} \in \mathbb{R}^N} \{\langle \mathbf{y}, \mathbf{x} \rangle - g(\mathbf{x})\}. \tag{2.10}$$

Theorem 2.12. *(Fenchel duality) Let f be a proper convex function on $\mathbb{R}^N$, and let g be a proper concave function on $\mathbb{R}^N$. Let $p^*, d^* \in [-\infty, \infty]$ be defined, respectively, by*

$$p^* = \inf_{\mathbf{x}} \{f(\mathbf{x}) - g(\mathbf{x})\}, \tag{2.11}$$

$$d^* = \sup_{\mathbf{y}} \{g^*(\mathbf{y}) - f^*(\mathbf{y})\}. \tag{2.12}$$

We have $d^* \leq p^*$. If $ri(dom\, f) \cap ri(dom\, g) \neq \emptyset$, then $d^* = p^*$, and the supremum in the dual problem in Eq. (2.12) *is attained at some $\mathbf{y}^* \in \mathbb{R}^N$.*

A direct proof of Fenchel duality is given in Rockafellar [91] and Borwein and Lewis [13], but it can also be derived from Lagrangian duality as discussed in Bertsekas et al. [11].

2.2.3 First-order optimality conditions

Among numerous useful properties that are derived from duality properties, in this thesis, the first-order optimality conditions are very important. The conditions are well-known as the Karush-Kuhn-Tucker (KKT) conditions. Recall the minimization problem in Eq. (2.4) and the Lagrangian function in Eq. (2.5).

Theorem 2.13. *(Karush-Kuhn-Tucker conditions) Suppose functions f, g_i's, and h_i's in Eq.* (2.4) *are differentiable. If $\mathbf{x}^*$ achieves the primal optimal value p^* in Eq.* (2.6) *and $(\boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$ achieves the dual optimal value d^* in Eq.* (2.8) *such that $d^* = p^*$,*

$\boldsymbol{\lambda}^* \in \mathbb{R}^I$ *and* $\boldsymbol{\mu}^* \in \mathbb{R}^J$ *are called Lagrange multipliers, and they satisfy:*

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \mathbf{x}}(\mathbf{x}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*) &= 0, \\
g_i(\mathbf{x}^*) &\leq 0, \ \textit{for} \ \forall i \in \{1, \cdots, I\}, \\
h_j(\mathbf{x}^*) &= 0, \ \textit{for} \ \forall j \in \{1, \cdots, J\}, \\
\lambda_i^* &\geq 0, \ \textit{for} \ \forall i \in \{1, \cdots, I\}, \\
\lambda_i^* g_i(\mathbf{x}^*) &= 0, \ \textit{for} \ \forall i \in \{1, \cdots, I\}.
\end{aligned}$$

That is, the KKT conditions are necessary conditions for $\mathbf{x}^*$ and $(\boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$ be the optimal solutions. A particularly interesting case is that $\mathcal{X}$ is a convex set, and f, all g_i's, and all h_i's are convex functions on $\mathcal{X}$. In this case the KKT conditions are sufficient for $\mathbf{x}^*$ and $(\boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$ be the optimal solutions. The KKT conditions play a crucial role in the development of efficient optimization algorithms for NMF, NTF, and Lasso in this thesis.

2.3 Block Coordinate Descent Method

The block coordinate descent (BCD) method is a divide-and-conquer strategy that can be generally applied to non-linear optimization problems. It divide variables into several disjoint subgroups and sequentially minimize the objective function with respect to the variables of each subgroup at a time. The BCD method is the backbone of algorithms developed in this thesis for NMF and NTF. We here summarize the BCD method and its convergence properties. More details can be found in Bertsekas [10].

Consider an optimization problem as follows:

$$\text{minimize} f(\mathbf{x}) \text{ subject to } \mathbf{x} \in \mathcal{X} \tag{2.13}$$

where $\mathcal{X}$ is a closed convex subset of $\mathbb{R}^N$. An important assumption to be exploited in the BCD method is that set $\mathcal{X}$ is represented by a Cartesian product:

$$\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_M, \tag{2.14}$$

where $\mathcal{X}_m$, $\forall m \in \{1, \cdots, M\}$, is assumed to be a closed convex subset of $\mathbb{R}^{N_m}$ satisfying $N = \sum_{m=1}^{M} N_m$. Accordingly, vector $\mathbf{x}$ is partitioned as $\mathbf{x} = (\mathbf{x}_1, \cdots, \mathbf{x}_M)$ so that $\mathbf{x}_m \in \mathcal{X}_m$ for $\forall m \in \{1, \cdots, M\}$. The BCD method solves for $\mathbf{x}_m$ fixing all other subvectors of $\mathbf{x}$ in a cyclic manner. That is, given the current iterate at the k^{th} step, $\mathbf{x}^{(k)} = (\mathbf{x}_1^{(k)}, \cdots, \mathbf{x}_M^{(k)})$, the algorithm generates the next iterate $\mathbf{x}^{(k+1)} = (\mathbf{x}_1^{(k+1)}, \cdots, \mathbf{x}_M^{(k+1)})$ block by block, according to the solution of a subproblem written as follows:

$$\mathbf{x}_m^{(k+1)} \leftarrow \arg\min_{\xi \in \mathcal{X}_m} f(\mathbf{x}_1^{(k+1)}, \cdots, \mathbf{x}_{m-1}^{(k+1)}, \xi, \mathbf{x}_{m+1}^{(k)}, \cdots, \mathbf{x}_M^{(k)}). \tag{2.15}$$

Also known as a *non-linear Gauss-Siedel* method [10], the BCD method updates one block each time, always using the most recently updated values of other blocks $\mathbf{x}_{\tilde{m}}, \tilde{m} \neq m$. This is important since it ensures that, after each update, the objective function value decreases. For a sequence $\{\mathbf{x}^{(k)}\}$ where each $\mathbf{x}^{(k)}$ is generated by the BCD method, the following property holds.

Theorem 2.14. *Suppose f is continuously differentiable in $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_M$, where $\mathcal{X}_m$, $m = 1, \cdots, M$, are closed convex sets. Furthermore, suppose that for all m and k, the minimum for*

$$\min_{\xi \in X_m} f(\mathbf{x}_1^{(k+1)}, \cdots, \mathbf{x}_{m-1}^{(k+1)}, \xi, \mathbf{x}_{m+1}^{(k)}, \cdots, \mathbf{x}_M^{(k)})$$

is uniquely attained. Let $\{\mathbf{x}^{(k)}\}$ be the sequence generated by the block coordinate descent method as in Eq. (2.15)*. Then, every limit point of $\{\mathbf{x}^{(k)}\}$ is a stationary point. The uniqueness of the minimum is not required when number of blocks M is two.*

The proof of Theorem 2.14 for an arbitrary number of blocks is shown in [10], and the last statement regarding the two block case is due to Grippo and Sciandrone [44]. When applying the BCD method to a constrained non-linear programming problem, it is critical to wisely choose a partition whose Cartesian product constitutes $\mathcal{X}$. An important criterion is whether the subproblems in Eq. (2.15) are efficiently solvable. In Chapter 3, difference choices of the block partition are discussed for the BCD method applied to NMF.

2.4 Dual Norm and Regularization Problem

The last section of this chapter is the application of Fenchel duality to specific cases. The results derived from this section will be used later in Chapter 8. Bach et al. [2] summarize related materials. Let us begin with the definition of dual norm.

Definition 2.15. (Dual norm) Let $\|\cdot\|$ be a norm defined on $\mathbb{R}^N$. The dual norm $\|\cdot\|_*$ of $\|\cdot\|$ is defined by

$$\|\mathbf{z}\|_* = \max_{\mathbf{x}\in\mathbb{R}^N} \mathbf{z}^T\mathbf{x} \text{ subject to } \|\mathbf{x}\| \leq 1.$$

The following theorem, stating a relationship between l_p-norm and its dual norm, is easy to verify [14].

Theorem 2.16. *If $p, q \in [1,\infty]$ satisfy $\frac{1}{p}+\frac{1}{q}=1$, l_p-norm and l_q-norm are dual to each other in $\mathbb{R}^N$.*

There is a useful relationship between a function involving a norm and an indicator function of its dual norm.

Theorem 2.17. *Consider a concave function $g(\mathbf{x}) = -\lambda\|\mathbf{x}\|$. Its concave conjugate is*

$$g^*(\mathbf{y}) = -I_{\left(\|\mathbf{y}\|_*\leq\lambda\right)} = \begin{cases} 0, & \text{if } \|\mathbf{y}\|_* \leq \lambda \\ -\infty, & \text{otherwise.} \end{cases}$$

Proof. First suppose $\|\mathbf{y}\|_* \leq \lambda$, and we will show that $g^*(\mathbf{y}) = 0$. Note that $\|\mathbf{y}\|_* \leq \lambda$ implies

$$-\lambda \leq \mathbf{y}^T\mathbf{x} \leq \lambda \text{ for } \forall\mathbf{x} \text{ satisfying } \|\mathbf{x}\| \leq 1. \tag{2.16}$$

For any nonzero $\mathbf{x} \in \mathbb{R}^N$,

$$\mathbf{y}^T\mathbf{x} - g(\mathbf{x}) = \mathbf{y}^T\mathbf{x} + \lambda \|\mathbf{x}\| = \|\mathbf{x}\| \left(\mathbf{y}^T \frac{\mathbf{x}}{\|\mathbf{x}\|} + \lambda \right) \geq 0,$$

where the last inequality is due to Eq. (2.16). Together with the fact that $\mathbf{y}^T\mathbf{x} - g(\mathbf{x}) = 0$ for $\mathbf{x} = 0$, we have that $g^*(\mathbf{y}) = \inf_{\mathbf{x} \in \mathbb{R}^N} \{\langle \mathbf{y}, \mathbf{x} \rangle - g(\mathbf{x})\} = 0$.

Next, let us suppose $\|\mathbf{y}\|_* > \lambda$, and we will show that $g^*(\mathbf{y}) = \infty$. Note that $\|\mathbf{y}\|_* > \lambda$ implies

$$\exists \tilde{\mathbf{x}} \text{ such that } \|\tilde{\mathbf{x}}\| \leq 1, \mathbf{y}^T\tilde{\mathbf{x}} > \lambda. \tag{2.17}$$

Then,

$$\begin{aligned} \inf_{\mathbf{x} \in \mathbb{R}^N} \{\langle \mathbf{y}, \mathbf{x} \rangle - g(\mathbf{x})\} &\leq \inf_{t \in \mathbb{R}} \{\langle \mathbf{y}, -t\tilde{\mathbf{x}} \rangle + \lambda \|-t\tilde{\mathbf{x}}\|\} \\ &= \inf_{t \in \mathbb{R}} t \{-\langle \mathbf{y}, \tilde{\mathbf{x}} \rangle + \lambda \|\tilde{\mathbf{x}}\|\} \\ &\leq \inf_{t \in \mathbb{R}} t \{-\langle \mathbf{y}, \tilde{\mathbf{x}} \rangle + \lambda\} \\ &= -\infty. \end{aligned}$$

where the last inequality is due to Eq. (2.17). □

The following corollary is a direct consequence of Fenchel duality (Theorem 2.12) and Theorem 2.17.

Corollary 2.18. *Let f be a proper convex function on $\mathbb{R}^N$. If f^* is the convex conjugate of f and $\|\cdot\|_*$ is the dual norm of $\|\cdot\|$,*

$$\max_{\mathbf{y} \in \mathbb{R}^N : \|\mathbf{y}\|_* \leq \lambda} -f^*(\mathbf{y}) \leq \min_{\mathbf{x} \in \mathbb{R}^N} f(\mathbf{x}) + \lambda \|\mathbf{w}\|.$$

If the domain of f has non-empty interior, than equality holds.

CHAPTER III

REVIEW OF NONNEGATIVE MATRIX FACTORIZATION ALGORITHMS

Let us recall the problem formulation of NMF. Suppose a nonnegative matrix $\mathbf{A} \in \mathbb{R}^{M \times N}$ is given. For a small integer $K < \min\{M, N\}$, we would like to solve the following optimization problem to find $\mathbf{W}$ and $\mathbf{H}$:

$$\min_{\mathbf{W},\mathbf{H}} f(\mathbf{W},\mathbf{H}) = \frac{1}{2} \left\| \mathbf{A} - \mathbf{W}\mathbf{H}^T \right\|_F^2, \tag{3.1}$$

$$\text{subject to } \mathbf{W}, \mathbf{H} \geq 0.$$

The goal in this chapter is to provide an overview of algorithms developed to solve Eq. (3.1). The review is organized based on the block coordinate descent (BCD) method presented in Section 2.3. Among numerous algorithms studied for NMF, the most popular is the multiplicative updating algorithm by Lee and Seung [65]. This algorithm has an advantage of being simple and easy to implement, and it has greatly contributed to the popularity of NMF. However, slow convergence of the multiplicative updating has been also pointed out [43, 71], and more efficient algorithms equipped with stronger theoretical convergence property have been introduced. They are based on either the alternating nonnegative least squares (ANLS) framework [56, 54, 71] or the hierarchical alternating least squares (HALS) method [25, 24].

In Section 3.1, we show that many promising NMF algorithms can be derived using one common framework of the BCD method. Other algorithms that are not based on the BCD framework are reviewed in Section 3.2 with relations to the algorithms within the BCD framework.

3.1 BCD Framework for NMF

Observe that the optimization variable of Eq. (3.1) is $(\mathbf{W}, \mathbf{H}) \in \mathbb{R}_+^{(M+N)\times K}$. The BCD method in Section 2.3 can be applied to Eq. (3.1) with difference choices of the block partition $\{\mathcal{X}_1, \cdots, \mathcal{X}_M\}$ that constitute $\mathbb{R}_+^{(M+N)\times K}$. In the following, we will discuss how different choices of partitions lead to NMF algorithms developed in the literature.

3.1.1 BCD with two matrix blocks – ANLS framework

In Eq. (3.1), the most natural partitioning of the variables is the two blocks representing $\mathbf{W}$ and $\mathbf{H}$ themselves. In this case, we take turns solving

$$\mathbf{W} \leftarrow \arg\min_{\mathbf{W}\geq 0} f(\mathbf{W}, \mathbf{H}) \quad \text{and} \quad \mathbf{H} \leftarrow \arg\min_{\mathbf{H}\geq 0} f(\mathbf{W}, \mathbf{H}).$$

These subproblems appear as

$$\min_{\mathbf{W}\geq 0} \|\mathbf{H}\mathbf{W}^T - \mathbf{A}^T\|_F^2 \quad \text{and} \quad \min_{\mathbf{H}\geq 0} \|\mathbf{W}\mathbf{H}^T - \mathbf{A}\|_F^2. \tag{3.2}$$

Since the subproblems in Eqs. (3.2) are the non-negativity constrained least squares (NLS) problems, the two-block BCD method has been called the alternating non-negative least squares (ANLS) framework [56, 71]. Even though the subproblems are convex optimization problems, they do not have a closed-form solution, and a numerical algorithm for the subproblem has to be provided. Several algorithms have been proposed to compute NMF based on the ANLS framework [56, 71, 54]. More discussion on those algorithms and an even faster algorithm is presented in Chapter 4.

Denote $\mathbf{W}$ and $\mathbf{H}$ after the i^{th} iteration by $\mathbf{W}^{(i)}$ and $\mathbf{H}^{(i)}$, respectively. According to Theorem 2.14, the convergence property of the ANLS framework can be written as follows.

Corollary 3.1. *If a minimum of each subproblem in Eq.* (3.2) *is attained in each step, every limit point of the sequence* $\{(\mathbf{W}^{(i)}, \mathbf{H}^{(i)})\}$ *generated by the ANLS framework is a stationary point.*

Note that a minimum is not required to be unique for the convergence result to hold because we have only two blocks. On the other hand, numerical methods to solve the NLS subproblems require some conditions for themselves to return a solution that attains a minimum, as discussed in Chapter 4.

Each NLS subproblem in Eq. (3.2) can be decomposed into independent NLS problems with a single right-hand side, where each problem concerns with a column of $\mathbf{W}^T$ or $\mathbf{H}^T$. This view corresponds to a BCD method with $M+N$ vector blocks, in which each block represents the row of $\mathbf{W}$ or $\mathbf{H}$. In literature, however, this view has not been emphasized because often it is more efficient to solve the NLS problem with multiple right-hand sides altogether.

3.1.2 BCD with $2K$ vector blocks–HALS/RRI algorithm

Let us now partition the unknowns into $2K$ blocks in which each block is a column in $\mathbf{W}$ or $\mathbf{H}$. In this case, it is easier to consider the objective function in the following form:

$$f(\mathbf{w}_1, \cdots, \mathbf{w}_K, \mathbf{h}_1, \cdots, \mathbf{h}_K) = \frac{1}{2}\|\mathbf{A} - \sum_{k=1}^{K} \mathbf{w}_k \mathbf{h}_k^T\|_F^2, \tag{3.3}$$

where $\mathbf{W} = (\mathbf{w}_1, \cdots \mathbf{w}_K) \in \mathbb{R}_+^{M\times K}$ and $\mathbf{H} = (\mathbf{h}_1, \cdots, \mathbf{h}_K) \in \mathbb{R}_+^{N\times K}$. The form in Eq. (3.3) represents that $\mathbf{A}$ is approximated by the sum of K rank-one matrices.

Following the BCD scheme, we can minimize f by iteratively solving

$$\mathbf{w}_k \leftarrow \arg\min_{\mathbf{w}_k \geq 0} f(\mathbf{w}_1, \cdots, \mathbf{w}_K, \mathbf{h}_1, \cdots, \mathbf{h}_K)$$

for $k = 1, \cdots, K$, and

$$\mathbf{h}_k \leftarrow \arg\min_{\mathbf{h}_k \geq 0} f(\mathbf{w}_1, \cdots, \mathbf{w}_K, \mathbf{h}_1, \cdots, \mathbf{h}_K)$$

for $k = 1, \cdots, K$. These subproblems appear as

$$\min_{\mathbf{w}\geq 0} \|\mathbf{w}\mathbf{h}_k^T - \mathbf{R}_k\|_F^2 \quad \text{and} \quad \min_{\mathbf{h}\geq 0} \|\mathbf{w}_k\mathbf{h}^T - \mathbf{R}_k\|_F^2, \tag{3.4}$$

where

$$\mathbf{R}_k = \mathbf{A} - \sum_{\tilde{k}=1,\tilde{k}\neq k}^{K} \mathbf{w}_{\tilde{k}} \mathbf{h}_{\tilde{k}}^T. \tag{3.5}$$

A promising aspect of this $2K$ block case is that each subproblem in Eq. (3.4) has a closed-form solution, as characterized in the following theorem.

Theorem 3.2. *Consider a minimization problem*

$$\min_{\mathbf{v}\geq 0} \|\mathbf{u}\mathbf{v}^T - \mathbf{B}\|_F^2 \tag{3.6}$$

where $\mathbf{B} \in \mathbb{R}^{M\times N}$ *and* $\mathbf{u} \in \mathbb{R}^M$ *are given. If* $\mathbf{u}$ *is a nonzero vector,* $\mathbf{v} = \frac{[\mathbf{B}^T\mathbf{u}]_+}{\mathbf{u}^T\mathbf{u}}$ *is the unique solution, where* $[\mathbf{b}_i^T\mathbf{u}]_+ = \max(\mathbf{b}_i^T\mathbf{u}, 0)$.

Proof. We show element by element. Regarding the i^{th} element of $\mathbf{v}$, v_i, the problem in Eq. (3.6) is narrowed down to

$$\min_{v_i\geq 0} \|\mathbf{u}v_i - \mathbf{b}_i\|_F^2,$$

where $\mathbf{b}_i$ is the i^{th} column of $\mathbf{B}$. Let $h(v_i) = \|\mathbf{u}v_i - \mathbf{b}_i\|_2^2 = \|\mathbf{u}\|_2^2 v_i^2 - 2v_i\mathbf{u}^T\mathbf{b}_i + \|\mathbf{b}_i\|_2^2$. Since $\frac{\partial h}{\partial v_i} = 2(v_i\|\mathbf{u}\|_2^2 - \mathbf{b}_i^T\mathbf{u})$, it is clear that the minimum value of $h(v_i)$ is obtained at $v_i = \frac{\mathbf{b}_i^T\mathbf{u}}{\mathbf{u}^T\mathbf{u}}$ if $\mathbf{b}_i^T\mathbf{u} \geq 0$. If $\mathbf{b}_i^T\mathbf{u} < 0$, the objective function value increases as v_i becomes larger above zero. Therefore, the minimum value is obtained at $v_i = 0$ if $\mathbf{b}_i^T\mathbf{u} < 0$. Combining these two cases, the solution can be expressed as $v_i = \frac{[\mathbf{b}_i^T\mathbf{u}]_+}{\mathbf{u}^T\mathbf{u}}$. □

Using Theorem 3.6, the solutions of Eq. (3.4) can be written as

$$\mathbf{w}_k \leftarrow \frac{[\mathbf{R}_k\mathbf{h}_k]_+}{\|\mathbf{h}_k\|_2^2} \quad \text{and} \quad \mathbf{h}_k \leftarrow \frac{[\mathbf{R}_k^T\mathbf{w}_k]_+}{\|\mathbf{w}_k\|_2^2}. \tag{3.7}$$

This $2K$-block BCD algorithm has been studied in the name of the hierarchical alternating least squares (HALS) method by Cichochi et al. [25, 24] and the rank-one residue iteration (RRI) independently by Ho [46]. In view of Theorem 2.14, the convergence property of the HALS/RRI algorithm can be written as follows.

Corollary 3.3. *If the columns of* $\mathbf{W}$ *and* $\mathbf{H}$ *remain nonzero throughout all the iterations and if the minimum of each problem in Eq.* (3.4) *is uniquely attained, every limit point of the sequence* $\{(\mathbf{W}^{(i)}, \mathbf{H}^{(i)})\}$ *generated by the HALS/RRI algorithm is a stationary point.*

In practice, zero columns could occur fairly easily in $\mathbf{W}$ or $\mathbf{H}$ during the HALS/RRI algorithm: In [40, 25], a small positive number is used for the maximum operator in Eq. (3.7). That is, $\max(\cdot, \epsilon)$ is used instead of $\max(\cdot, 0)$ with $\epsilon = 10^{-16}$.

For an efficient implementation, it is not necessary to explicitly compute $\mathbf{R}_k$. Replacing $\mathbf{R}_k$ in Eq. (3.7) with the expression in Eq. (3.5), the solutions can be rewritten as

$$\mathbf{w}_k \leftarrow \left[\mathbf{w}_k + \frac{(\mathbf{AH})_{\cdot k} - (\mathbf{WH}^T\mathbf{H})_{\cdot k}}{(\mathbf{H}^T\mathbf{H})_{kk}}\right]_+, \tag{3.8a}$$

$$\mathbf{h}_k \leftarrow \left[\mathbf{h}_k + \frac{(\mathbf{A}^T\mathbf{W})_{\cdot k} - (\mathbf{HW}^T\mathbf{W})_{\cdot k}}{(\mathbf{W}^T\mathbf{W})_{kk}}\right]_+. \tag{3.8b}$$

The choice of update formulae is related with the choice of an update order. Two versions of an update order can be considered:

$$\mathbf{w}_1 \to \mathbf{h}_1 \to \cdots \to \mathbf{w}_K \to \mathbf{h}_K \tag{3.9}$$

and

$$\mathbf{w}_1 \to \cdots \to \mathbf{w}_K \to \mathbf{h}_1 \to \cdots \to \mathbf{h}_K. \tag{3.10}$$

When using Eq. (3.7), because $\mathbf{R}_k$ is explicitly computed, the order of updates should be Eq. (3.9) for efficiency. When using Eqs. (3.8), although either Eq. (3.9) or Eq. (3.10) can be used, Eq. (3.10) tends to be more efficient in certain environments such as MATLAB. The convergence property in Corollary 3.3 is invariant of the choice of these orders. To update all the elements in $\mathbf{W}$ and $\mathbf{H}$, Eq. (3.7) with the ordering of Eq. (3.9) require $8KMN + 3K(M + N)$ flops, whereas Eqs. (3.8) with either choice of ordering require $4KMN + (4K^2 + 6K)(M + N)$ flops. When

$K \ll \min(M, N)$, the latter is more efficient. Moreover, the memory requirement of Eqs. (3.8) is smaller because $\mathbf{R}_k$ need not be stored. For more details, see [24].

3.1.3 BCD with $K(M+N)$ scalar blocks

In one extreme, the unknowns can be partitioned into $K(M+N)$ blocks of scalars. In this case, every element of $\mathbf{W}$ and $\mathbf{H}$ is considered as a block in the context of Theorem 2.14. In this case, it helps to write the objective function as a quadratic function of a scalar w_{mk} or h_{nk} assuming all other elements in $\mathbf{W}$ and $\mathbf{H}$ are fixed:

$$f(w_{mk}) = \frac{1}{2}\|(\mathbf{R}_k)_{m\cdot} - w_{mk}\mathbf{h}_{\cdot k}^T\|_2^2 + \text{constant}, \tag{3.11a}$$

$$f(h_{nk}) = \frac{1}{2}\|(\mathbf{R}_k)_{\cdot n} - \mathbf{w}_{\cdot k}h_{nk}\|_2^2 + \text{constant}. \tag{3.11b}$$

According to the BCD framework, we iteratively update each element by

$$w_{mk} \leftarrow \arg\min_{w_{mk}\geq 0} f(w_{mk}) = \left[w_{mk} + \frac{(\mathbf{AH})_{mk} - (\mathbf{WH}^T\mathbf{H})_{mk}}{(\mathbf{H}^T\mathbf{H})_{kk}}\right]_+, \tag{3.12a}$$

$$h_{nk} \leftarrow \arg\min_{h_{nk}\geq 0} f(h_{nk}) = \left[h_{nk} + \frac{(\mathbf{A}^T\mathbf{W})_{nk} - (\mathbf{HW}^T\mathbf{W})_{nk}}{(\mathbf{W}^T\mathbf{W})_{kk}}\right]_+. \tag{3.12b}$$

The updates of w_{mk} and h_{nk} are independent of all other elements in the same column. Therefore, it is possible to update all the elements in the same column of $\mathbf{W}$ and $\mathbf{H}$ *simultaneously*. Once we organize the updates of Eqs. (3.12) column-wise, the result is the same as Eqs. (3.8). Hence, a particular arrangement of the BCD method with scalar blocks is equivalent to the BCD method with $2K$ vector blocks discussed in Section 3.1.2. Accordingly, the HALS/RRI method can be derived by the BCD method either with vector blocks or with scalar blocks. On the other hand, it is not possible to simultaneously solve the *rows* of $\mathbf{W}$ and $\mathbf{H}$ because their solutions depend on each other.

The convergence property of the scalar block case is similar to that of the vector block case.

Corollary 3.4. *If the columns of* $\mathbf{W}$ *and* $\mathbf{H}$ *remain nonzero throughout all the iterations and if the minimum of each problem in Eqs.* (3.12) *is uniquely attained,* *every limit point of the sequence* $\{(\mathbf{W}^{(i)}, \mathbf{H}^{(i)})\}$ *generated by the BCD method with* $K(M+N)$ *scalar blocks is a stationary point.*

The multiplicative updating algorithm [65], which is one of the most widely used for NMF, also focuses on each element in its derivation. It is however different in a sense that the solution updated for each element is not the optimal one for the sub-problems in Eq. (3.12). We discuss more about the multiplicative updating algorithm in Section 3.2.1

3.1.4 BCD for NMF with regularization

To incorporate extra constraints or prior information, various regularization terms can be added to the NMF formulation in Eq. (3.1). In general, we can consider a minimization problem as follows:

$$\min_{\mathbf{W},\mathbf{H}\geq 0} \frac{1}{2} \left\| \mathbf{A} - \mathbf{W}\mathbf{H}^T \right\|_F^2 + \phi(\mathbf{W}) + \psi(\mathbf{H}), \tag{3.13}$$

where $\phi(\cdot)$ and $\psi(\cdot)$ are regularization terms that often involve matrix or vector norms. Here we discuss the Frobenius-norm and the l_1-norm regularization and show how NMF regularized by those norms can be easily computed using the BCD method. In this subsection, scalars α or β represent parameters that control the strength of regularization.

The Frobenius-norm regularization [87, 56] corresponds to

$$\phi(\mathbf{W}) = \alpha \|\mathbf{W}\|_F^2 \quad \text{and} \quad \psi(\mathbf{H}) = \beta \left\| \mathbf{H} \right\|_F^2. \tag{3.14}$$

The Frobenius-norm regularization may be used to prevent the elements of $\mathbf{W}$ or $\mathbf{H}$ from growing too large in their absolute values. In addition, it can be adopted to stabilize the BCD methods. In the two matrix block case, since the uniqueness of the minimum is not required according to Corollary 3.1 [44], the full column rank condition is not necessary for the convergence of the BCD method. It is however needed for some algorithms that solve the NLS subproblems, as we discuss in Section 4.5. The Frobenius-norm regularization ensures that the NLS subproblem of the two matrix block case is of full column rank, as shown below. In the $2K$ vector block or the $K(M+N)$ scalar block case, the condition that $\mathbf{w}_k$ and $\mathbf{h}_k$ remain nonzero throughout all the iterations can be relaxed when the Frobenius-norm regularization is used.

Applying the BCD framework with two matrix blocks to Eq. (3.13) with the regularization term of Eq. (3.14), $\mathbf{W}$ can be updated as

$$\mathbf{W} \leftarrow \arg\min_{\mathbf{W}\geq 0} \left\| \begin{pmatrix} \mathbf{H} \\ \sqrt{2\alpha}\mathbf{I}_K \end{pmatrix} \mathbf{W}^T - \begin{pmatrix} \mathbf{A}^T \\ \mathbf{0}_{K\times M} \end{pmatrix} \right\|_F^2, \tag{3.15}$$

and $\mathbf{H}$ can be updated with a similar reformulation. Clearly, if α is nonzero, $\begin{pmatrix} \mathbf{H} \\ \sqrt{2\alpha}\mathbf{I}_K \end{pmatrix}$ in Eq. (3.15) is of full column rank. Applying the BCD framework with two vector blocks, a column of $\mathbf{W}$ is updated as

$$\mathbf{w}_k \leftarrow \left[\frac{(\mathbf{H}^T\mathbf{H})_{kk}}{(\mathbf{H}^T\mathbf{H})_{kk} + 2\alpha}\mathbf{w}_k + \frac{(\mathbf{A}\mathbf{H})_{\cdot k} - (\mathbf{W}\mathbf{H}^T\mathbf{H})_{\cdot k}}{(\mathbf{H}^T\mathbf{H})_{kk} + 2\alpha} \right]_+ . \tag{3.16}$$

Observe that if α is nonzero, the solution of Eq. (3.16) is uniquely defined without requiring $\mathbf{h}_k$ to be a nonzero vector.

The l_1-norm regularization has been adopted to promote sparsity in the factor matrices. Sparsity was shown to improve the 'part-based' interpretation [48] or to improve clustering ability of NMF [55, 58]. When sparsity is desired on matrix $\mathbf{H}$, the l_1-norm regularization can be set as

$$\phi(\mathbf{W}) = \alpha\|\mathbf{W}\|_F^2 \quad \text{and} \quad \psi(\mathbf{H}) = \beta \sum_{n=1}^{N} \|\mathbf{h}_{n\cdot}\|_1^2, \tag{3.17}$$

where $\mathbf{h}_{n\cdot}$ represents the n^{th} row of $\mathbf{H}$. The l_1-norm term of $\psi(\mathbf{H})$ in Eq. (3.17) promotes sparsity on $\mathbf{H}$ while the Frobenius norm term of $\phi(\mathbf{W})$ is needed to prevent

$\mathbf{W}$ from growing too large. Similarly, sparsity can be imposed on matrix $\mathbf{W}$ or on the both of $\mathbf{W}$ and $\mathbf{H}$.

Applying the BCD framework with two matrix blocks to Eq. (3.13) with the regularization term of Eq. (3.17), $\mathbf{W}$ can be updated as Eq. (3.15), and $\mathbf{H}$ can be updated as

$$\mathbf{H} \leftarrow \arg\min_{\mathbf{H}\geq 0} \left\| \begin{pmatrix} \mathbf{W} \\ \sqrt{2\beta}\mathbf{1}_{1\times K} \end{pmatrix} \mathbf{H}^T - \begin{pmatrix} \mathbf{A} \\ \mathbf{0}_{1\times N} \end{pmatrix} \right\|_F^2 . \tag{3.18}$$

Applying the BCD framework with two vector blocks, a column of $\mathbf{W}$ is updated as Eq. (3.16), and a column of $\mathbf{H}$ is updated as

$$\mathbf{h}_k \leftarrow \left[\mathbf{h}_k + \frac{(\mathbf{A}^T\mathbf{W})_{\cdot k} - \mathbf{H}((\mathbf{W}^T\mathbf{W})_{\cdot k} + 2\beta \mathbf{1}_K)}{(\mathbf{W}^T\mathbf{W})_{kk} + 2\beta} \right]_+ . \tag{3.19}$$

Note that the l_1-norm term in Eq. (3.17) is written as the sum of the *squares* of the l_1-norm of the columns of $\mathbf{H}$. Alternatively, we can impose the l_1-norm based regularization without squaring: That is,

$$\phi(\mathbf{W}) = \alpha \|\mathbf{W}\|_F^2 \quad \text{and} \quad \psi(\mathbf{H}) = \beta \sum_{n=1}^{N} \sum_{k=1}^{K} |\mathbf{h}_{nk}| . \tag{3.20}$$

Although both Eq. (3.17) and Eq. (3.20) promote sparsity, the squared form in Eq. (3.17) is easier to handle with the two matrix block case, as shown above. Applying the $2K$ vector block framework on Eq. (3.13) with the regularization term of Eq. (3.20), the update for a column of $\mathbf{h}$ is written as

$$\mathbf{h}_k \leftarrow \left[\mathbf{h}_k + \frac{(\mathbf{A}^T\mathbf{W})_{\cdot k} - (\mathbf{H}\mathbf{W}^T\mathbf{W})_{\cdot k} + \beta \mathbf{1}_K}{(\mathbf{W}^T\mathbf{W})_{kk}} \right]_+ .$$

For more information, see [24], Section 4.7.4 of [26], and Section 4.5 of [46]. When the BCD framework with two matrix blocks is used, a custom algorithm for l_1-regularized least squares problem has to be involved: See, e.g., [34].

3.2 Other Approaches

3.2.1 Multiplicative updating rules

The multiplicative updating rule [65] is by far the most popular algorithm for NMF. In this algorithm, each element is updated *by a multiplication* as

$$w_{mk} \leftarrow w_{mk} \frac{(\mathbf{AH})_{mk}}{(\mathbf{WH}^T\mathbf{H})_{mk}} \quad \text{and} \quad h_{nk} \leftarrow h_{nk} \frac{(\mathbf{A}^T\mathbf{W})_{nk}}{(\mathbf{HW}^T\mathbf{W})_{nk}}. \tag{3.21}$$

Since elements are updated in this multiplication form, the nonnegativity is always satisfied when $\mathbf{A}$ is nonnegative. This algorithm can be contrasted with the HALS algorithm as follows. Observing that

$$\frac{\partial f}{\partial w_{mk}} \propto (\mathbf{WH}^T\mathbf{H})_{mk} - (\mathbf{AH})_{mk} \text{ and}$$
$$\frac{\partial f}{\partial h_{nk}} \propto (\mathbf{HW}^T\mathbf{W})_{nk} - (\mathbf{A}^T\mathbf{W})_{nk},$$

an element-wise gradient descent update can be written as

$$w_{mk} \leftarrow w_{mk} + \lambda_{mk} \left((\mathbf{AH})_{mk} - (\mathbf{WH}^T\mathbf{H})_{mk} \right),$$
$$h_{nk} \leftarrow h_{nk} + \mu_{nk} \left((\mathbf{A}^T\mathbf{W})_{nk} - (\mathbf{HW}^T\mathbf{W})_{nk} \right),$$

where λ_{mk} and μ_{nk} represent step-lengths. The multiplicative updating rule is obtained by taking

$$\lambda_{mk} = \frac{w_{mk}}{(\mathbf{WHH}^T)_{mk}} \text{ and } \mu_{nk} = \frac{h_{nk}}{(\mathbf{W}^T\mathbf{WH})_{nk}}, \tag{3.22}$$

whereas the HALS algorithm interpreted as the BCD method with scalar blocks as in Eqs. (3.12) is obtained by taking

$$\lambda_{mk} = \frac{1}{(\mathbf{H}^T\mathbf{H})_{kk}} \text{ and } \mu_{nk} = \frac{1}{(\mathbf{W}^T\mathbf{W})_{kk}}. \tag{3.23}$$

The step-lengths chosen in the multiplicative updating rule is conservative enough so that the result is always nonnegative. On the other hand, the step-lengths chosen in the HALS algorithm could potentially lead to a nonnegative value, and therefore the projection $[\cdot]_+$ is involved. Whereas the HALS algorithm is built upon the BCD framework and therefore convergence property in Corollary 3.4 holds, the property does not hold for the multiplicative updating rule since the step-lengths in Eq. (3.22) does not achieve the optimal solution. In practice, the convergence of the HALS algorithm is much faster than that of the multiplicative updating.

Lee and Seung [65] showed that under the multiplicative updating rule, the objective function in Eq. (3.1) is non-increasing. However, it is unknown whether it attains a stationary point. Gonzalez and Zhang [43] demonstrated the difficulty, and the slow convergence of multiplicative updates has been reported in [56, 71] and in Chapter 4 of this thesis. As an effort to overcome this issue, Lin [73] proposed a modified update rule for which every limit point is stationary; however, after this modification, the update rule becomes *additive* instead of multiplicative.

3.2.2 Alternating least squares method

In the BCD framework with the two blocks as shown in Section 3.1.1, it is important to find a minimum of the nonnegativity-constrained least squares (NLS) subproblems in Eq. (3.2). In some early work on NMF, Berry et al. [9] has proposed to approximately solve these NLS subproblems hoping to accelerate the algorithm. In their alternating least squares (ALS) method, they solved the least squares problems ignoring the nonnegativity constraints, and then negative elements in the computed solution are set to zeros. However, this solution does not achieve a minimum of the subproblems *with* nonnegativity constraints. Therefore, although each iteration of the ALS method is efficient, the convergence property in Corollary 3.1 is not applicable to the ALS method. In fact, the ALS method does not necessarily decrease the objective function after each iteration as shown in Chapter 4.

It is interesting to note that the HALS method does not have this problem although the same projection is used. In the HALS method, a quadratic objective function is minimized with respect to each column or each element in $\mathbf{W}$ or $\mathbf{H}$, and the solution is set back to zero when negative. In this case, the projected solution is indeed the minimum of the subproblem with nonnegativity constraints.

3.2.3 Successive rank-one deflation

Some algorithms have been designed to compute NMF based on successive rank-one deflation. This approach is motivated from the fact that the singular value decomposition (SVD) can be computed through successive rank-one deflation; however, when considered for NMF, the rank-one deflation method has a few issues as we summarize below.

Let us first recapitulate the deflation approach for SVD. Consider a matrix $\mathbf{A} \in \mathbb{R}^{M \times N}$ of rank R and its SVD written as

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T = \sum_{r=1}^{R} \sigma_r \mathbf{u}_r \mathbf{v}_r^T, \tag{3.24}$$

where $\mathbf{U} = \begin{bmatrix} \mathbf{u}_1 & \cdots & \mathbf{u}_R \end{bmatrix} \in \mathbb{R}^{M \times R}$ and $\mathbf{V} = \begin{bmatrix} \mathbf{v}_1 & \cdots & \mathbf{v}_R \end{bmatrix} \in \mathbb{R}^{N \times R}$ are orthogonal matrices, and $\mathbf{\Sigma} = \operatorname{diag}(\sigma_1, \cdots, \sigma_R) \in \mathbb{R}^{R \times R}$ with $\sigma_1 \geq \cdots \geq \sigma_R \geq 0$. Recall from Section 2.1.1 that the rank-K SVD for $K < R$ is a truncation of Eq. (3.24) obtained by taking only the first K singular values and corresponding singular vectors:

$$\tilde{\mathbf{A}}_K = \tilde{\mathbf{U}}_K \tilde{\mathbf{\Sigma}}_K \tilde{\mathbf{V}}_K^T = \sum_{k=1}^{K} \sigma_k \mathbf{u}_k \mathbf{v}_k^T,$$

where $\tilde{\mathbf{U}}_K \in \mathbb{R}^{M \times K}$ and $\tilde{\mathbf{V}}_K \in \mathbb{R}^{N \times K}$ are sub-matrices of $\mathbf{U}$ and $\mathbf{V}$ obtained by taking the left K columns. As state in Theorem 2.2, the best rank-K approximation of $\mathbf{A}$ in terms of minimizing the l_2-norm or the Frobenius norm of the residual matrix is achieved by the rank-K SVD [97, 42]. The rank-K SVD, particularly when $K \ll R$, can be computed through successive rank-one reduction as follows. First, the best rank-one approximation, $\sigma_1 \mathbf{u}_1 \mathbf{v}_1^T$, is computed by an efficient algorithm such as the power iteration. Then, the residual matrix is obtained as $\mathbf{E}_1 = \mathbf{A} - \sigma_1 \mathbf{u}_1 \mathbf{v}_1^T =$

$\sum_{r=2}^{R} \sigma_r \mathbf{u}_r \mathbf{v}_r^T$, and the rank of $\mathbf{E}_1$ is $R-1$. Applying the power iteration to $\mathbf{E}_1$, its best rank-one approximation, $\sigma_2 \mathbf{u}_2 \mathbf{v}_2^T$, is obtained, and the residual matrix $\mathbf{E}_2$, whose rank is $R-2$, can be found in the same manner: $\mathbf{E}_2 = \mathbf{E}_1 - \sigma_2 \mathbf{u}_2 \mathbf{v}_2^T = \sum_{r=3}^{R} \sigma_r \mathbf{u}_r \mathbf{v}_r^T$. Repeating this process for K times, one can obtain the rank-K SVD.

When it comes to NMF, Corollary 2.4 provides a useful insight about the $K = 1$ case. That is, for a nonnegative matrix, there exists a pair of nonnegative vectors that achieve the best rank-one approximation. In other words, for a nonnegative matrix, imposing nonnegativity on low-rank factors does not result in worse approximation error if $K = 1$. This elegant property, however, is not very useful when $K \geq 2$. After the best rank-one approximation is deflated, the residual matrix $\mathbf{E}_1$ may contain negative elements; if $\mathbf{E}_1$ contains negative elements, Corollary 2.4 is not applicable any more.

In general, successive rank-one deflation is not an optimal approach for NMF computation. Let us take a look at a small example which demonstrates this problem. Consider matrix $\mathbf{A}$ given as

$$\mathbf{A} = \begin{pmatrix} 4 & 6 & 0 \\ 6 & 4 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

The best rank-one approximation of $\mathbf{A}$ is shown as $\mathbf{B}_1$ below. The residual matrix is $\tilde{\mathbf{E}}_1 = \mathbf{A} - \mathbf{B}_1$, which contains nonnegative elements.

$$\mathbf{B}_1 = \begin{pmatrix} 5 & 5 & 0 \\ 5 & 5 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \tilde{\mathbf{E}}_1 = \begin{pmatrix} -1 & 1 & 0 \\ 1 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

One of the best rank-one approximation of $\tilde{\mathbf{E}}_1$ with nonnegativity constraints is $\mathbf{B}_2$, and the residual matrix is $\tilde{\mathbf{E}}_2$:

$$\mathbf{B}_2 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \ \tilde{\mathbf{E}}_2 = \begin{pmatrix} -1 & 1 & 0 \\ 1 & -1 & 0 \\ 0 & 0 & 0 \end{pmatrix}.$$

The nonnegative rank-two approximation obtained by this rank-one deflation approach is

$$\mathbf{B}_1 + \mathbf{B}_2 = \begin{pmatrix} 5 & 5 & 0 \\ 5 & 5 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

However, the best nonnegative rank-two approximation of $\mathbf{A}$ is in fact $\tilde{\mathbf{A}}_2$ with residual matrix $\mathbf{E}_2$:

$$\tilde{\mathbf{A}}_2 = \begin{pmatrix} 4 & 6 & 0 \\ 6 & 4 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \ \mathbf{E}_2 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Therefore, a strategy that successively finds the best rank-one approximation with nonnegativity constraints and deflates in each step does not lead to an optimal solution of NMF.

Due to this difficulty, some variations of rank-one deflation has been investigated for NMF. Biggs et al. [12] proposed a rank-one reduction algorithm in which they look for a nonnegative *submatrix* that is close to a rank-one approximation. Once such a submatrix is identified, they compute the best rank-one approximation using the power method and ignore the residual. Gillis and Glineur [41] sought a nonnegative rank-one approximation under the constraint that the residual remains nonnegative. Due to this constraints, however, the problem of finding the nonnegative rank-one approximation is more complicated and computationally expensive than the power iteration. Optimization properties such as convergence to a stationary point has not been shown for these modified rank-one reduction methods.

It is worth noting the difference between the HALS algorithm, described as the $2K$ vector block case in Section 3.1.2, and the successive rank-one deflation. Both approaches have a similarity in that the best rank-one approximation of the residual matrix with nonnegativity constraints is computed in each step, filling in the k^{th} columns of $\mathbf{W}$ and $\mathbf{H}$ for $k = 1, \cdots, K$. Whereas the columns of $\mathbf{W}$ and $\mathbf{H}$ are computed only once in the successive rank-one deflation, the HALS algorithm repeatedly updates all the columns until a local minimum is achieved. This repetition is necessary in NMF unlike in SVD.

CHAPTER IV

A FAST ACTIVE-SET-LIKE METHOD FOR NONNEGATIVE MATRIX FACTORIZATION

In this chapter, we introduce a new and fast algorithm for NMF based on the ANLS framework described in Section 3.1.1. Previous NMF algorithms using the ANLS framework include the active-set method [63, 56], the projected gradient method [71], and the projected quasi-Newton method [54]. The names tell how each algorithm solves the nonnegativity-constrained least squares (NLS) subproblem. The projected gradient and the projected quasi-Newton methods apply techniques from unconstrained optimization with modifications for nonnegativity constraints. The active-set method searches for the optimal active and passive sets of variables by maintaining working sets as candidates. At each iteration, an unconstrained least squares problem is solved, and the working sets are updated based on the result. The block principal pivoting method [89, 53], which we call an *active-set-like* method due to its similarity with the active-set method, also follows this framework, but it overcomes a limitation of the active-set method. Unlike the active-set method, in which typically only one variable is exchanged between working sets, the block principal pivoting method allows the exchanges of multiple variables with a goal of finding the optimal active and passive sets faster. In this chapter, we adopt the block principal pivoting method in NMF computation. We introduce ideas that improve the block principal pivoting method in the context of NMF and then build a new algorithm for NMF.

The block principal pivoting method was earlier introduced by Judice and Pires [89, 53] to solve the NLS problems with a single right hand side. In the computation of NMF, it is important to efficiently solve the NLS problems with multiple right hand sides, and we introduce strategies to accelerate the block principal pivoting method for multiple right hand sides. These acceleration strategies are based on important observations on the typical structure of the NLS problems arising in the NMF computation.

Thorough experimental comparisons among several NMF algorithms, including the one proposed in this chapter, will follow the introduction of the new algorithm. The ANLS-based algorithms [71, 54, 56] and the HALS algorithm [24] appeared recently, and there has not been other papers with substantial comparisons such as the ones we offer in this chapter. The comparisons were performed on text, image, and synthetic data sets, and the results are demonstrated showing the relative computational efficiency of various NMF algorithms. The proposed new algorithm exhibits state-of-the-art performance allowing only HALS to be comparable. We also show a condition under which the proposed method outperforms the HALS algorithm.

The rest of this chapter is organized as follows. In Section 4.1, the ANLS framework for NMF and related background materials are introduced. In Section 4.2, our new algorithm for NMF is described. Implementation details and experimentation settings are shown in Section 4.3, and comparison results are demonstrated in Section 4.4. We conclude the chapter in Section 4.5 with discussion.

4.1 ANLS Framework for NMF

We begin by recapitulating the ANLS framework described in Section 3.1.1, as follows.

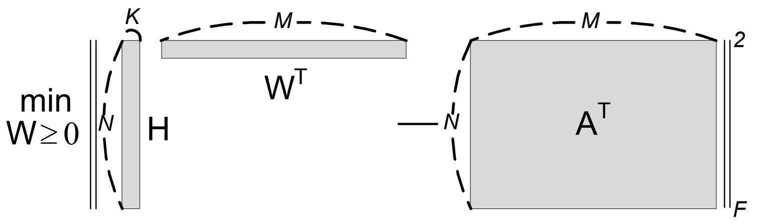
1. Initialize $\mathbf{H} \in \mathbb{R}^{K \times N}$ with nonnegative elements.

2. Repeat solving the following problems until a stopping criterion is satisfied:

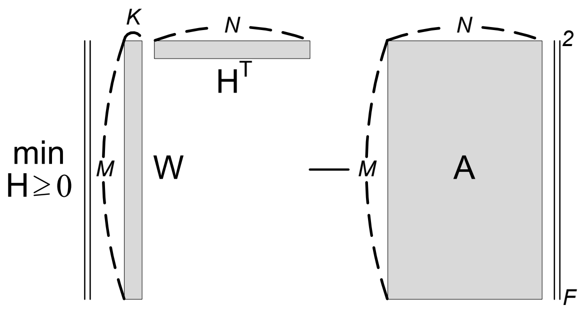
$$\min_{\mathbf{W} \geq 0} \left\| \mathbf{H}\mathbf{W}^T - \mathbf{A}^T \right\|_F^2 \tag{4.1a}$$

   where $\mathbf{H}$ is fixed, and

$$\min_{\mathbf{H} \geq 0} \left\| \mathbf{W}\mathbf{H}^T - \mathbf{A} \right\|_F^2 \tag{4.1b}$$



(a)



(b)

Figure 4.1: Typical structure of the NLS problems arising in NMF computation.

where $\mathbf{W}$ is fixed.

3. The columns of $\mathbf{W}$ are normalized to unit l_2-norm and the columns of $\mathbf{H}$ are scaled accordingly.

It is important to observe that the NLS problems in Eqs. (4.1) have special characteristics. NMF is a dimension reduction algorithm, which is applied to high-dimensional data. The original dimension is very large, e.g., several thousands or more, and the reduced dimension is small, e.g., on the order of tens. Therefore, the coefficient matrix $\mathbf{W} \in \mathbb{R}^{M \times K}$ in Eq. (4.1b) is typically very long and thin ($M \gg K$), and the coefficient matrix $\mathbf{H} \in \mathbb{R}^{N \times K}$ in Eq. (4.1a) is also long and thin ($N \gg K$) in general. At the same time, the matrices $\mathbf{W}^T$ and $\mathbf{H}^T$ of unknown variables in Eqs. (4.1a) and (4.1b), respectively, are flat and wide for the same reason. Figure 4.1 illustrates these characteristics. These observations are critical in designing an efficient algorithm for the subproblems in Eq. (4.1), and we will revisit this point in later sections.

In order to design an NMF algorithm based on the ANLS framework, one has to devise a specific method to solve the subproblems in Eqs. (4.1). A classic algorithm for the NLS problem is the active-set method by Lawson and Hanson [63]. The active-set method searches for the optimal active and passive sets by exchanging a variable between two working sets. Note that if we know the passive (i.e., strictly positive) variables of the solution in advance, then an NLS problem can be easily solved by a simple unconstrained least squares procedure on the passive variables. Although the Lawson and Hanson's algorithm has been a standard for the NLS problems,[1] it is extremely slow when used for NMF in a straightforward way. Faster versions of the algorithm were recently developed by Bro and De Jong [15] and Van Benthem and Keenan [99], and Kim and Park utilized them in their NMF algorithm [56].

A major limitation of the active-set method is that the variables of working sets are exchanged satisfying the nonnegativity of the solution vector while making sure that the objective function decreases after each iteration. As a result, typically only one variable is exchanged between working sets per iteration, slowing down the algorithm when the number of unknowns is large. Methods based on iterative optimization schemes such as the projected gradient method due to Lin [71] and the projected quasi-Newton method due to Kim et al. [54] are free of the above limitation. These algorithms are modified from techniques in unconstrained optimization by providing specialized rules to choose step-length and projecting the solution to the feasible nonnegative orthant at every iteration.

The block principal pivoting method overcomes the limitation of the active-set method in a different fashion. We now describe this method in detail.

[1]Lawson and Hanson's algorithm is adopted as a MATLAB function *lsqnonneg*.

4.2 Block Principal Pivoting Method

In this section, we present the block principal pivoting method for the NLS problems and the NMF algorithm based on the method. We will first review the block principal pivoting method that was developed for the NLS problems with a single right-hand side [53] and then introduce methods that improve upon this to handle multiple right-hand sides efficiently.

4.2.1 NLS with a single right-hand side vector

For the moment, let us focus on the NLS problem with a single right-hand side vector formulated as

$$\min_{\mathbf{x}\geq 0} \|\mathbf{B}\mathbf{x}-\mathbf{c}\|_2^2\,, \tag{4.2}$$

where $\mathbf{B}\in\mathbb{R}^{p\times q}$, $\mathbf{c}\in\mathbb{R}^{p\times 1}$, $\mathbf{x}\in\mathbb{R}^{q\times 1}$, and $p\geq q$. The subproblems in Eqs. (4.1) can be decomposed into several independent instances of Eq. (4.2) with respect to each right-hand side vector. Thus, an algorithm for Eq. (4.2) is a basic building block for an algorithm for Eqs. (4.1).

The Karush-Kuhn-Tucker optimality condition for Eq. (4.2) is written as follows:

$$\mathbf{y} = \mathbf{B}^T\mathbf{B}\mathbf{x}-\mathbf{B}^T\mathbf{c} \tag{4.3a}$$

$$\mathbf{y} \geq 0 \tag{4.3b}$$

$$\mathbf{x} \geq 0 \tag{4.3c}$$

$$x_i y_i = 0,\ i=1,\cdots,q. \tag{4.3d}$$

We assume that matrix $\mathbf{B}$ has full column rank. In this case, matrix $\mathbf{B}^T\mathbf{B}$ is positive definite, and the problem in Eq. (4.2) is strictly convex. Then, a solution $\mathbf{x}$ that satisfies the conditions in Eqs. (4.3) is the optimal solution of Eq. (4.2). Problems in the form of Eqs. (4.3) are known as linear complementarity problems (LCP).

We divide the index set $\{1,\cdots,q\}$ into two groups $\mathcal{F}$ and $\mathcal{G}$ such that $\mathcal{F}\cup\mathcal{G}=\{1,\cdots,q\}$ and $\mathcal{F}\cap\mathcal{G}=\emptyset$. Let $\mathbf{x}_{\mathcal{F}}$, $\mathbf{x}_{\mathcal{G}}$, $\mathbf{y}_{\mathcal{F}}$, and $\mathbf{y}_{\mathcal{G}}$ denote the subsets of variables with corresponding indices, and let $\mathbf{B}_\mathcal{F}$ and $\mathbf{B}_\mathcal{G}$ denote the submatrices of $\mathbf{B}$ with corresponding column indices. Initially, we assign

$$\mathbf{x}_\mathcal{G} = 0 \text{ and } \mathbf{y}_\mathcal{F} = 0.$$

That is, all the elements of $\mathbf{x}_\mathcal{G}$ and $\mathbf{y}_\mathcal{F}$ are set as zero. Then, $\mathbf{x}$ and $\mathbf{y}$ always satisfy Eq. (4.3d) for any values of $\mathbf{x}_\mathcal{F}$ and $\mathbf{y}_\mathcal{G}$. Now, we compute $\mathbf{x}_\mathcal{F}$ and $\mathbf{y}_\mathcal{G}$ using Eq. (4.3a) and check whether the computed values of $\mathbf{x}_\mathcal{F}$ and $\mathbf{y}_\mathcal{G}$ satisfy Eqs. (4.3b) and (4.3c). The computation of $\mathbf{x}_\mathcal{F}$ and $\mathbf{y}_\mathcal{G}$ is done as follows:

$$\mathbf{x}_\mathcal{F} = \arg\min_{\mathbf{x}_\mathcal{F}} \|\mathbf{B}_\mathcal{F}\mathbf{x}_\mathcal{F} - \mathbf{c}\|_2^2, \tag{4.4a}$$

$$\mathbf{y}_\mathcal{G} = \mathbf{B}_\mathcal{G}^T(\mathbf{B}_\mathcal{F}\mathbf{x}_\mathcal{F} - \mathbf{c}). \tag{4.4b}$$

One can first solve for $\mathbf{x}_\mathcal{F}$ in Eq. (4.4a) and substitute the result into Eq. (4.4b). We call the pair $(\mathbf{x}_\mathcal{F}, \mathbf{y}_\mathcal{G})$ a complementary basic solution if it is obtained by Eqs. (4.4).

If a complementary basic solution $(\mathbf{x}_\mathcal{F}, \mathbf{y}_\mathcal{G})$ satisfies $\mathbf{x}_\mathcal{F} \geq 0$ and $\mathbf{y}_\mathcal{G} \geq 0$, then it is called *feasible*. In this case, current $\mathbf{x}$ is the optimal solution of Eq. (4.2), and the algorithm terminates. Otherwise, a complementary basic solution $(\mathbf{x}_\mathcal{F}, \mathbf{y}_\mathcal{G})$ is *infeasible*, and we need to update $\mathcal{F}$ and $\mathcal{G}$ by exchanging variables for which Eq. (4.3b) or Eq. (4.3c) does not hold. Formally, we define the following index set

$$\mathcal{V} = \{i \in \mathcal{F} : x_i < 0\} \cup \{i \in \mathcal{G} : y_i < 0\}, \tag{4.5a}$$

and then a variable x_i with $i \in \mathcal{V}$ is called an *infeasible variable*. Now, choose a non-empty subset $\hat{\mathcal{V}} \subseteq \mathcal{V}$. Then, $\mathcal{F}$ and $\mathcal{G}$ are updated by the following rules:

$$\mathcal{F} = (\mathcal{F} - \hat{\mathcal{V}}) \cup (\hat{\mathcal{V}} \cap \mathcal{G}), \tag{4.6a}$$

$$\mathcal{G} = (\mathcal{G} - \hat{\mathcal{V}}) \cup (\hat{\mathcal{V}} \cap \mathcal{F}). \tag{4.6b}$$

The size $|\hat{\mathcal{V}}|$ represents how many variables are exchanged per iteration. If $|\hat{\mathcal{V}}| > 1$, then the algorithm is called a *block* principal pivoting algorithm; if $|\hat{\mathcal{V}}| = 1$, then the

Algorithm 4.1 Block principal pivoting method for the NLS problem with a single right-hand side vector

```
Input: $\mathbf{B} \in \mathbb{R}^{p \times q}$ and $\mathbf{c} \in \mathbb{R}^{p}$
Output: $\mathbf{x}(\in \mathbb{R}^{q \times 1}) = \arg\min_{\mathbf{x} \geq 0} \|\mathbf{B}\mathbf{x} - \mathbf{c}\|_2^2$
 1: Initialize $\mathcal{F} = \emptyset$, $\mathcal{G} = \{1, \cdots, q\}$, $\mathbf{x} = 0$, $\mathbf{y} = -\mathbf{B}^T\mathbf{c}$, $\alpha = 3$, and $\beta = q + 1$
 2: Compute $\mathbf{x}_{\mathcal{F}}$ and $\mathbf{y}_{\mathcal{G}}$ by Eqs. (4.4).
 3: while $(\mathbf{x}_{\mathcal{F}}, \mathbf{y}_{\mathcal{G}})$ is infeasible do
 4:    Compute $\mathcal{V}$ by Eqs. (4.5).
 5:    If $|\mathcal{V}| < \beta$, set $\beta = |\mathcal{V}|$, $\alpha = 3$, and $\hat{\mathcal{V}} = \mathcal{V}$.
 6:    If $|\mathcal{V}| \geq \beta$ and $\alpha \geq 1$, set $\alpha = \alpha - 1$ and $\hat{\mathcal{V}} = \mathcal{V}$.
 7:    If $|\mathcal{V}| \geq \beta$ and $\alpha = 0$, set $\hat{\mathcal{V}}$ by Eq. (4.7).
 8:    Update $\mathcal{F}$ and $\mathcal{G}$ by Eqs. (4.6).
 9:    Update $\mathbf{x}_{\mathcal{F}}$ and $\mathbf{y}_{\mathcal{G}}$ by Eqs. (4.4).
10: end while
```

algorithm is called a *single* principal pivoting algorithm. The active-set method can be understood as an instance of single principal pivoting algorithms. The algorithm repeats this procedure until the number of infeasible variables (i.e., $|\hat{\mathcal{V}}|$) becomes zero.

In order to speed up the search procedure, one usually uses $\hat{\mathcal{V}} = \mathcal{V}$, which we call the *full exchange rule*. The full exchange rule means that we exchange all variables of $\mathcal{F}$ and $\mathcal{G}$ that do not satisfy Eqs. (4.3), and the rule accelerates computation by reducing the number of iterations required until termination. However, contrary to the active-set method in which the variable to exchange is carefully selected to reduce the objective function, the full exchange rule may lead to a cycle and fail to find an optimal solution although it occurs rarely. To ensure finite termination, we need to employ a backup rule, which uses the following exchange set for Eqs. (4.6):

$$\hat{\mathcal{V}} = \{i : i = \max\{i \in \mathcal{V}\}\}. \tag{4.7}$$

The backup rule, where only the infeasible variable with the largest index is exchanged, is a single principal pivoting rule. This simple exchange rule guarantees a finite termination: Assuming that matrix $\mathbf{B}$ has full column rank, the exchange rule in Eq. (4.7) returns the solution of Eq. (4.3) in a finite number of iterations [53].

Combining the full exchange rule and the backup rule, the block principal pivoting method for the NLS problem with a single right-hand side is obtained as summarized in Algorithm 4.1. Because the backup rule is much slower than the full exchange rule, it is used only if the full exchange rule does not work well. Finite termination of Algorithm 4.1 is achieved by controlling the number of infeasible variables. In Algorithm 4.1, variable α is used as a buffer on the number of the full exchange rules that may be tried. If the full exchange rule increases the number of infeasible variables, then α is reduced by one. Once α becomes zero, the backup rule is used until it makes the number of infeasible variables smaller than the lowest value achieved so far, which is stored in β. This has to occur in a finite number of steps because the backup rule has a finite termination property. As soon as the backup rule achieves a new lowest number of infeasible variables, we return to the full exchange rule. We used three as the default value of α, which means that we can try the full exchange rule up to three times until it reduces the number of infeasible variables. Since the number of infeasible variables is systematically reduced, the algorithm terminates in a finite number of steps.

The block principal pivoting method was shown very efficient for the NLS problems [89, 19]. In principle, the computational cost of the block principal pivoting method will depend on how often the full exchange rule fails so that the backup rule has to be activated. In our extensive tests in this chapter, the backup rule appearance was not observed, suggesting that the full exchange rule works well in NMF. Since the full exchange rule allows the exchanges of multiple variables between $\mathcal{F}$ and $\mathcal{G}$, the block principal pivoting method becomes much faster than the active-set method, as we report in Section 4.4.

One might relate the two sets, $\mathcal{F}$ and $\mathcal{G}$, of the block principal pivoting method to the passive and active sets in the active-set method. However, they are not necessarily identical. In the active-set method, the solution is sought while satisfying the condition $\mathbf{x} \geq 0$, so a variable x_i in which i is in the passive set is required to satisfy

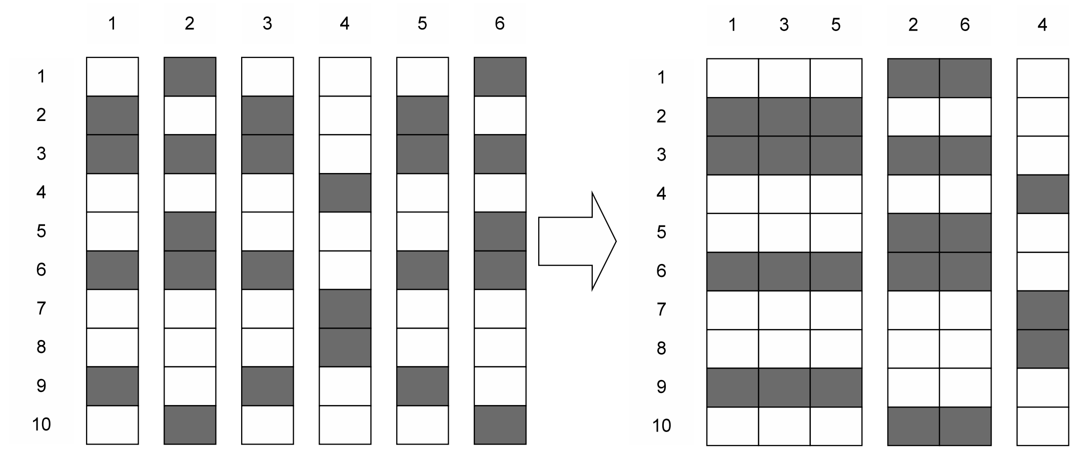


Figure 4.2: An example of the grouping of right-hand side vectors when $q = 10$ and $r = 6$. Dark cells indicate variables with indices in $\mathcal{F}$, which need to be computed by Eq. (4.9). By grouping the columns that have a common $\mathcal{F}$ set, i.e., columns $\{1, 3, 5\}$,$\{2, 6\}$ and $\{4\}$, we can avoid redundant computation for Cholesky factorization in solving Eq. (4.9).

$x_i \geq 0$ in every iteration. In the block principal pivoting method, on the other hand, a variable x_i with $i \in \mathcal{F}$ can be of any sign except for the final iteration. Therefore, the block principal pivoting method does not require an initial solution with $\mathbf{x} \geq 0$ while the active-set method does.

4.2.2 NLS with multiple right-hand side vectors

Now suppose we need to solve the following NLS problem:

$$\min_{\mathbf{X} \geq 0} \|\mathbf{BX} - \mathbf{C}\|_F^2 , \tag{4.8}$$

where $\mathbf{B} \in \mathbb{R}^{p \times q}$, $\mathbf{C} \in \mathbb{R}^{p \times r}$, and $\mathbf{X} \in \mathbb{R}^{q \times r}$. It is possible to simply run Algorithm 4.1 for each right-hand side vector $\mathbf{c}_1, \cdots, \mathbf{c}_r$, where $\mathbf{C} = (\mathbf{c}_1, \cdots, \mathbf{c}_r)$, since the columns of $\mathbf{X}$ do not depend on each other. However, this approach is not computationally efficient, and we will explain how we obtain an efficient algorithm for the multiple right-hand side case using the ideas from [15] and [99] in the context of the block principal pivoting method.

In Algorithm 4.1, the major computational burden is from the need to compute

$\mathbf{x}_{\mathcal{F}}$ and $\mathbf{y}_{\mathcal{G}}$ as shown in Eqs. (4.4). We can solve Eq. (4.4a) by a normal equation

$$\mathbf{B}_{\mathcal{F}}^T\mathbf{B}_{\mathcal{F}}\mathbf{x}_{\mathcal{F}} = \mathbf{B}_{\mathcal{F}}^T\mathbf{c}, \tag{4.9}$$

and Eq. (4.4b) can be rewritten as

$$\mathbf{y}_{\mathcal{G}} = \mathbf{B}_{\mathcal{G}}^T\mathbf{B}_{\mathcal{F}}\mathbf{x}_{\mathcal{F}} - \mathbf{B}_{\mathcal{G}}^T\mathbf{c}. \tag{4.10}$$

We need $\mathbf{B}_{\mathcal{F}}^T\mathbf{B}_{\mathcal{F}}$, $\mathbf{B}_{\mathcal{F}}^T\mathbf{c}$, $\mathbf{B}_{\mathcal{G}}^T\mathbf{B}_{\mathcal{F}}$, and $\mathbf{B}_{\mathcal{G}}^T\mathbf{c}$ for solving Eqs. (4.9) and (4.10), and these matrices and vectors vary throughout iterations because $\mathcal{F}$ and $\mathcal{G}$ are updated in each iteration.

The improvements are closely related with the observations mentioned in Section 4.1. First, note that for the NLS problems arising from NMF, matrix $\mathbf{B}$ is typically very long and thin, and computing $\mathbf{B}_{\mathcal{F}}^T\mathbf{B}_{\mathcal{F}}$, $\mathbf{B}_{\mathcal{F}}^T\mathbf{c}$, $\mathbf{B}_{\mathcal{G}}^T\mathbf{B}_{\mathcal{F}}$, and $\mathbf{B}_{\mathcal{G}}^T\mathbf{c}$ is computationally expensive. However, we can compute $\mathbf{B}^T\mathbf{B}$ and $\mathbf{B}^T\mathbf{C}$ in the beginning and reuse them in later iterations. As $\mathbf{B}_{\mathcal{F}}^T\mathbf{B}_{\mathcal{F}}$, $\mathbf{B}_{\mathcal{F}}^T\mathbf{c}$, $\mathbf{B}_{\mathcal{G}}^T\mathbf{B}_{\mathcal{F}}$, and $\mathbf{B}_{\mathcal{G}}^T\mathbf{c}$ can be retrieved as submatrices of $\mathbf{B}^T\mathbf{B}$ and $\mathbf{B}^T\mathbf{C}$ for any $\mathcal{F}$ and $\mathcal{G}$, we can avoid expensive matrix-matrix multiplications. Since the column size of $\mathbf{B}$ is small for the NLS problems arising from NMF, storage needed for $\mathbf{B}^T\mathbf{B}$ and $\mathbf{B}^T\mathbf{C}$ will also be small. This idea is also applicable to the single right-hand side case, but its impact is more dramatic in the multiple right-hand side case.

Another improvement comes from the fact that matrix $\mathbf{X}$ is typically flat and wide in the NLS problems for NMF. Suppose we simultaneously run Algorithm 4.1 for many right-hand side vectors. In each iteration, we have index sets $\mathcal{F}_j$ and $\mathcal{G}_j$ for each column $j \in \{1, \cdots, r\}$, and we must compute $\mathbf{x}_{\mathcal{F}_j}$ and $\mathbf{y}_{\mathcal{G}_j}$ using Eqs. (4.9) and (4.10). The idea is to find groups of columns that share the same index sets $\mathcal{F}_j$ and $\mathcal{G}_j$ and solve Eq. (4.9) for the columns in the same group. By doing so, we avoid repeated computation for Cholesky factorization in solving Eq. (4.9). Figure 4.2 illustrates this grouping idea. Note that if $\mathbf{X}$ is flat and wide, which is the case for the NLS problems in NMF, more columns are likely to share their index sets $\mathcal{F}_j$ and $\mathcal{G}_j$,

Algorithm 4.2 Block principal pivoting method for the NLS with multiple right-hand side vectors. $\mathbf{X}_{\mathcal{F}_j}$ and $\mathbf{Y}_{\mathcal{G}_j}$ represents the subsets of j-th column of $\mathbf{X}$ and $\mathbf{Y}$ indexed by $\mathcal{F}_j$ and $\mathcal{G}_j$, respectively.

```
Input: $\mathbf{B} \in \mathbb{R}^{p\times q}$,$\mathbf{C} \in \mathbb{R}^{p\times r}$
Output: $\mathbf{X}(\in \mathbb{R}^{q\times r}) = \arg\min_{\mathbf{x}\geq 0} \|\mathbf{BX}-\mathbf{C}\|_F^2$
 1: Compute $\mathbf{B}^T\mathbf{B}$ and $\mathbf{B}^T\mathbf{C}$.
 2: Initialize $\mathcal{F}_j = \emptyset$ and $\mathcal{G}_j = \{1,\cdots,q\}$ for all $j \in \{1,\cdots,r\}$. Set $\mathbf{X} = 0$, $\mathbf{Y} = -\mathbf{B}^T\mathbf{C}$, $\boldsymbol{\alpha}(\in \mathbb{R}^r) = 3$, and $\boldsymbol{\beta}(\in \mathbb{R}^r) = q+1$.
 3: Compute $\mathbf{X}_{\mathcal{F}_j}$ and $\mathbf{Y}_{\mathcal{G}_j}$ for all $j \in \{1,\cdots,r\}$ by Eqs. (4.4) using column grouping.
 4: while any $(\mathbf{X}_{\mathcal{F}_j}, \mathbf{Y}_{\mathcal{G}_j})$ is infeasible do
 5:     Find the indices of columns in which the solution is infeasible: $I = \{j : (\mathbf{X}_{\mathcal{F}_j}, \mathbf{Y}_{\mathcal{G}_j}) \text{ is infeasible}\}$.
 6:     Compute $\mathcal{V}_j$ for all $j \in I$ by Eqs. (4.5).
 7:     For all $j \in I$ with $|\mathcal{V}_j| < \beta_j$, set $\beta_j = |\mathcal{V}_j|$, $\alpha_j = 3$ and $\hat{\mathcal{V}}_j = \mathcal{V}_j$.
 8:     For all $j \in I$ with $|\mathcal{V}_j| \geq \beta_j$ and $\alpha_j \geq 1$, set $\alpha_j = \alpha_j - 1$ and $\hat{\mathcal{V}}_j = \mathcal{V}_j$.
 9:     For all $j \in I$ with $|\mathcal{V}_j| \geq \beta_j$ and $\alpha_j = 0$, set $\hat{\mathcal{V}}_j$ by Eq. (4.7).
10:     Update $\mathcal{F}_j$ and $\mathcal{G}_j$ for all $j \in I$ by Eqs. (4.6).
11:     Update $\mathbf{X}_{\mathcal{F}_j}$ and $\mathbf{Y}_{\mathcal{G}_j}$ for all $j \in I$ by Eqs. (4.4) using column grouping.
12: end while
```

allowing us to obtain bigger speed-up. We summarize the improved block principal pivoting method for multiple right-hand sides in Algorithm 4.2.

4.2.3 NMF based on ANLS with block principal pivoting

The block principal pivoting method combined with the improvements is quite efficient for the NLS problems with multiple right-hand sides. To compute NMF, we use Algorithm 4.2 to solve the subproblems in Eqs. (4.1). As explained in Section 3.1.4, the block principal pivoting method can also be applied to the Frobenius norm or l_1-norm regularized NMF, because the subproblems in those regularized formulations appear as the NLS problems with multiple right-hand sides. Implementation issues such as a stopping criterion are discussed in Section 4.3.

4.3 Implementation and Data Sets

We describe the details of our implementation and data sets used for comparisons.

4.3.1 NMF algorithms compared

We compared the following algorithms for NMF. Due to space limitations, we do not present the details of other algorithms but only refer to the papers in which they are presented.

1. (ANLS-BPP) ANLS with the block principal pivoting method proposed in this chapter
2. (ANLS-AS) ANLS with Kim and Park's active-set method [56]
3. (ANLS-PGRAD) ANLS with Lin's projected gradient method [71]
4. (ANLS-PQN) ANLS with Kim et al.'s projected quasi-Newton method [54]
5. (HALS) Cichocki and Phan's hierarchical alternating least squares algorithm [25, 24]
6. (MU) Lee and Seung's multiplicative updating algorithm [65]
7. (ALS) Berry et al.'s alternating least squares algorithm [9]

We implemented our ANLS-BPP method in MATLAB. For ANLS-AS, we implemented two different versions. The first one is using the grouping idea for multiple right-hand sides as described in [99, 56], and we refer to this implementation as ANLS-AS-GROUP. Alternatively, the ANLS-AS method can be implemented by solving the NLS problems with a single right-hand side separately using the updating of the Cholesky factor. We refer to this case as ANLS-AS-UPDATE. For the ANLS-PGRAD method, we used the MATLAB code written by its author, and for ANLS-PQN, we refined the code written by its authors since it was not carefully optimized for high-dimensional data, in which NMF is typically used. Our refined version of ANLS-PQN is much faster than the original one by the authors, and thus we used the refined version in our experiments. In [24], two versions of the HALS algorithm are introduced, and we used a faster version that updates all columns of $\mathbf{W}$ ($\mathbf{H}$) before proceeding to update $\mathbf{H}$ ($\mathbf{W}$). We implemented MU and ALS, which are straightforward to implement.

In all the algorithms, once we obtain $\mathbf{H}$ and $\mathbf{W}$ as the result of the i^{th} iteration, we used them as initial values of the $(i+1)^{th}$ iteration. As iteration progress, the solutions $\mathbf{H}$ and $\mathbf{W}$ do not change much, and therefore starting from the result of the previous iteration is a good warm start for the next iteration. In particular, for the block principal pivoting method, warm starting means that only the partitioning of the indices into sets $\mathcal{F}$ and $\mathcal{G}$ from the result of the previous iteration is used instead of specific numerical values.

4.3.2 Stopping criterion

When an iterative algorithm is executed in practice, one needs a criterion to stop iterations. In the case of NMF, a stopping criterion can be designed to check whether a local minimum of the objective function has been reached. In practice, one usually checks whether a point is stationary, which can be checked by the following criterion suggested by Lin [71].

According to the KKT condition, $(\mathbf{W}, \mathbf{H})$ is a stationary point of Eq. (3.1) if and only if

$$\mathbf{W} \geq 0 \quad , \quad \mathbf{H} \geq 0, \tag{4.11a}$$

$$\nabla f_{\mathbf{W}} = \partial f(\mathbf{W}, \mathbf{H})/\partial \mathbf{W} \geq 0 \quad , \quad \nabla f_{\mathbf{H}} = \partial f(\mathbf{W}, \mathbf{H})/\partial \mathbf{H} \geq 0, \tag{4.11b}$$

$$\mathbf{W}. * \nabla f_{\mathbf{W}} = 0 \quad , \quad \mathbf{H}. * \nabla f_{\mathbf{H}} = 0, \tag{4.11c}$$

where $.*$ represents element-wise multiplications. Defining the projected gradient $\nabla^p f_{\mathbf{W}}$ as

$$(\nabla^p f_{\mathbf{W}})_{ij} \equiv \begin{cases} (\nabla f_{\mathbf{W}})_{ij} & \text{if } (\nabla f_{\mathbf{W}})_{ij} < 0 \text{ or } \mathbf{W}_{ij} > 0 \\ 0 & \text{otherwise} \end{cases}$$

and $\nabla^p f_{\mathbf{H}}$ similarly, the conditions in Eqs. (4.11) can be rephrased as

$$\nabla^p f_{\mathbf{W}} = 0 \text{ and } \nabla^p f_{\mathbf{H}} = 0.$$

We use the norm of the projected gradients defined as

$$\Delta = \sqrt{\|\nabla^p f_{\mathbf{W}}\|_F^2 + \|\nabla^p f_{\mathbf{H}}\|_F^2}. \tag{4.12}$$

Using this definition, the stopping criterion is

$$\frac{\Delta}{\Delta_0} \leq \epsilon, \tag{4.13}$$

where Δ_0 is the value of Δ using the initial values of $\mathbf{W}$ and $\mathbf{H}$, and ϵ is a tolerance value to choose. This criterion has been useful in determining when to stop iterations, but we found some issues with it in our experiments. We explain them in Section. 4.4.2.

4.3.3 Data sets

We have used several real-world data sets for our comparison, and the information of four data sets is shown in Table 4.1. Among them, two text data sets are in sparse format. The Topic Detection and Tracking 2 (TDT2) text corpus[2] contains news articles from various sources such as NYT, CNN, and VOA in 1998. The corpus is manually labeled across 100 different topics, and it has been widely used for text mining research. From the corpus, we randomly selected 40 topics in which the number of articles in each topic is greater than 10. By counting the term frequency of each term in each document, we obtained a matrix of size $19,009 \times 3,087$. The 20 Newsgroups data set[3] is a collection of newsgroup documents in 20 different topics. We used a term-document matrix of size $26,214 \times 11,314$.[4]

[2]`http://projects.ldc.upenn.edu/TDT2/`

[3]`http://people.csail.mit.edu/jrennie/20Newsgroups/`

[4]`http://www.zjucadcg.cn/dengcai/Data/TextData.html`

Two image data sets in Table 4.1 are in dense format. The facial image database by AT&T Laboratories Cambridge[5] contains 400 facial images of 40 different people with 10 images per person. Each facial image has 92×112 pixels in 8-bit gray level. The resulting matrix was of size $10,304 \times 400$. The CMU PIE database[6] is collection of human faces under different poses, illumination conditions, and expressions. A matrix of size $4,096 \times 11,554$, from a resized version in 64×64 pixels,[7] was used for our executions.

In addition to data sets in Table 4.1, we also employed synthetically created data sets and massive-scale data sets. The details of these data sets are described in Section 4.4.3 and Section 4.4.4, respectively.

4.4 Comparison Results

All experiments were executed in MATLAB on a Linux machine with a 2.66GHz Intel Quad-core processor and 6GB memory except that the experiments on massive data sets in Section 4.4.4 were performed on machines with higher performance as explained there. The multi-threading option of MATLAB was disabled. In all the executions, all the algorithms were provided with the same initial values.

4.4.1 The active-set and the block principal pivoting methods

We first report observations about the ANLS-AS and the ANLS-BPP methods. As mentioned before, we implemented two versions of ANLS-AS: ANLS-AS-GROUP is based on a column grouping strategy, as used in Kim and Park [56], and ANLS-AS-UPDATE solves each right-hand side vector separately using the updating of the Cholesky factor. We compared the performance of the two versions, and we are also interested in how they perform compared to ANLS-BPP.

[5]`http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html`
[6]`http://www.ri.cmu.edu/projects/project_418.html`
[7]`http://www.zjucadcg.cn/dengcai/Data/FaceData.html`

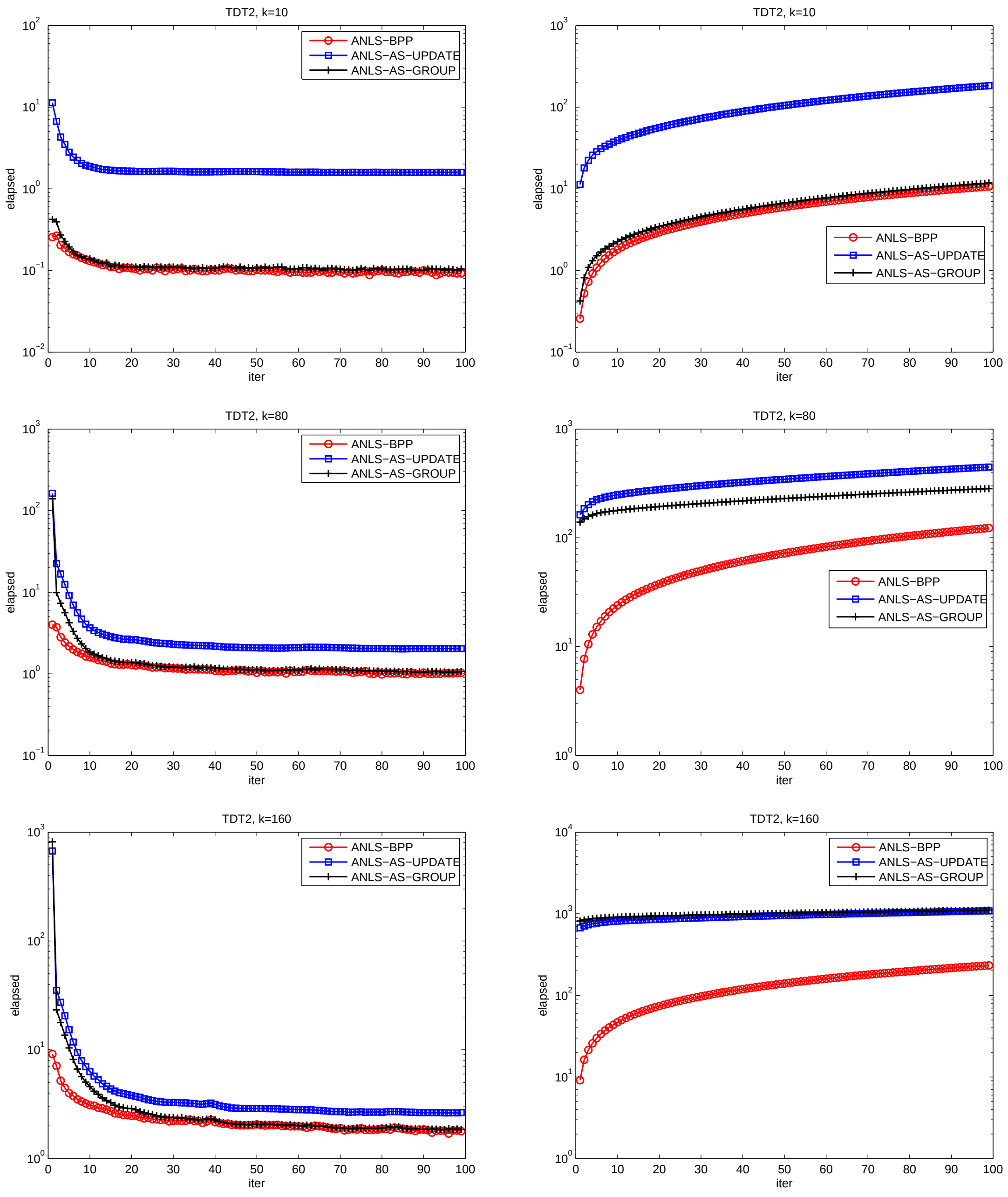


Figure 4.3: Comparison of the active-set (ANLS-AS) and the block principal pivoting (ANLS-BPP) methods on the TDT2 text data set. The left column shows the execution time of each iteration, and the right column shows cumulative execution time. Average of 5 different random initializations are shown. Top row: $K = 10$, middle row: $K = 80$, bottom row: $K = 160$.

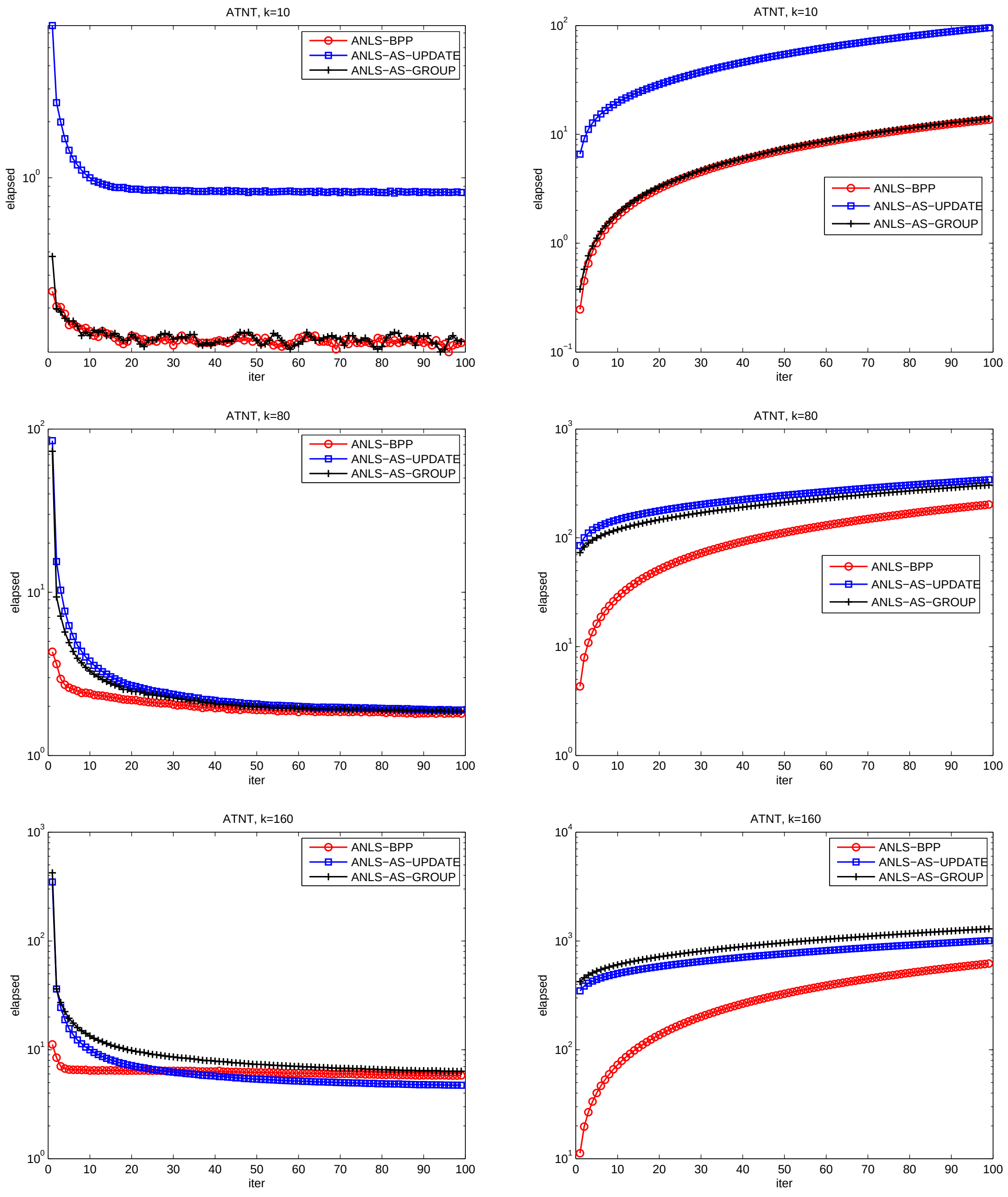


Figure 4.4: Comparison of the active-set (ANLS-AS) and the block principal pivoting (ANLS-BPP) methods on the ATNT image data set. The left column shows the execution time of each iteration, and the right column shows cumulative execution time. Average of 5 different random initializations are shown. Top row: $K = 10$, middle row: $K = 80$, bottom row: $K = 160$.

Table 4.1: Data sets, their sizes, the sparsity of solutions, and the grouping effects of the ANLS-AS-GROUP and the ANLS-BPP methods. Sparsities are calculated as the proportions of zero elements in each factor. See text for the description of grouping effects.

Data set	Size and Format	K	Sparsity (%)		Grouping effect (%)			
					BPP		AS-GROUP	
			W	H	W	H	W	H
TDT2	19,009×3,087 Sparse (99.58% sparsity)	10	53.1	46.4	94.3	76.2	94.4	76.2
		20	70.6	60.2	54.1	12.4	53.6	13.6
		80	85.4	77.7	40.7	3.1	28.6	3.2
		160	89.4	84	39.4	2.8	21.3	2.7
20 Newsgroups	26,214×11,314 Sparse (99.66% sparsity)	10	58.3	44.6	97.1	95.8	97	95.7
		20	67.5	53.3	48.0	19.4	53.3	48.5
		80	80.5	73.1	9.6	0.9	7.6	1.2
		160	86.8	78.8	7.4	0.7	4.6	0.8
ATNT	10,304×400 Dense	10	14.1	9.6	97.6	87.1	97.6	87.1
		20	20.8	18.8	66.4	20.3	66.4	21.8
		80	33.9	41.1	0.6	0.7	1.6	1.7
		160	32.7	61.3	0.5	0.5	0.6	0.7
PIE 64	4,096×11,554 Dense	10	27.4	14.5	91.7	96.6	91.8	96.8
		20	38.8	17.3	44.2	68.9	46.6	71.1
		80	58.7	21.3	0.8	1.5	1.4	3.3
		160	63.4	28.1	0.5	0.5	0.5	1.3

In Table 4.1, we present the results from the execution of ANLS-AS-GROUP and ANLS-BPP. Both methods were executed with 5 different random initializations for 100 iterations, and average results are shown in the table. Sparsities are calculated as the proportions of zero elements in each factor after 100 iterations. The grouping effect (GE) is calculated as

$$\text{GE} = \frac{\text{\# of Cholesky factorizations that are omitted thanks to column grouping}}{\text{\# of systems of equations needed to be solved}}.$$

Both the numerator and the denominator are summed over the 100 iterations. When GE is close to 100%, it means that significant savings are achieved; when GE is close to zero, there are only little savings. In Table 4.1, it can be seen that significant savings from grouping were observed when $K = 10$ and $K = 20$ whereas only limited savings were observed when $K = 80$ and $K = 160$.

Now let us see Figures 4.3 and 4.4 where we present the average execution time of each iteration and their accumulation. Unlike the ANLS-PGRAD and the ANLS-PQN methods, both the ANLS-AS and the ANLS-BPP methods solve the NLS subproblems in each iteration exactly. Hence, when the same initial values are provided, solutions after each iteration from these methods are the same up to numerical rounding errors. It is therefore enough to observe the amount of time spent in each iteration for the purpose of comparing these methods. In Figure 4.3, which shows results on the TDT2 text data set, it can be seen that ANLS-AS-UPDATE remains slower than ANLS-AS-GROUP for various K values. On the other hand, in Figure 4.4, which shows results on the ATNT image data set, the trend is a little different: While ANLS-AS-UPDATE performed slower than ANLS-AS-GROUP for small K values, ANLS-AS-UPDATE became faster than ANLS-AS-GROUP for large K values. Partial explanations of this difference are as follows.

For small K values, ANLS-AS-GROUP and ANLS-BPP achieved significant savings by grouping as can be seen from Table 4.1. Consequently, in the $K = 10$ cases of Figures 4.3 and 4.4, ANLS-AS-GROUP was significantly faster than ANLS-AS-UPDATE. For large K values, in contrast, it is generally expected that using the updating of the Cholesky factor is beneficial. For the ATNT image data set, this was the case as can be seen from the fact that ANLS-AS-UPDATE outperformed ANLS-AS-GROUP for $K = 160$. For the TDT2 text data set, however, nontrivial savings from grouping was observed even when $K = 160$. Hence, ANLS-AS-GROUP remained faster than ANLS-AS-UPDATE for all K values.

It is important to note that ANLS-BPP is either as fast as ANLS-AS-GROUP or is significantly faster than both the ANLS-AS methods. For the $K = 160$ case on the ATNT data set, the iteration cost of ANLS-AS-UPDATE becomes smaller than that of ANLS-BPP after many iterations; however, the cumulative cost of ANLS-BPP is still much smaller than that of ANLS-AS-UPDATE. This observation suggests that a hybrid method can be potentially investigated, where we employ ANLS-BPP but replace ANLS-BPP with ANLS-AS-UPDATE in later iterations only for large K values. On the other hand, since the ANLS-AS and the ANLS-BPP methods typically converge within 20 to 30 iterations, the benefit of the hybrid method is not expected to be significant in practice.

Results on the 20 Newsgroups and the PIE 64 data sets showed similar trends, and we do not present them here.

4.4.2 Comparison with other algorithms

We now show comparison results of ANLS-BPP with all other methods that are listed in Section 4.3.1. The computational cost of each iteration and the objective function reduced after each iteration in those algorithms are generally different, so a fair way to compare them would be observing "how well an algorithm minimizes the objective function in how much computation time." Figures 4.5 and 4.6 show the average execution results of 5 different random initializations. We have recorded the relative objective value $\left(\left\|\mathbf{A} - \mathbf{W}\mathbf{H}^T\right\|_F / \left\|\mathbf{A}\right\|_F\right)$ at the end of each iteration, and the time spent to compute the objective value is excluded from the execution time. One execution result with relative objective values measured at discrete time points gives us a piecewise-linear function, and we averaged piecewise-linear functions for different random initializations to plot in the figures. Because the first several iterations of ANLS-AS took too much time, the results of ANLS-AS are not shown.

The results on the TDT2 and the 20 Newsgroups data sets are shown in Figure 4.5. When $K = 10$, most algorithms except ANLS-PQN tended to quickly converge. When $K = 80$ or $K = 160$, it becomes clear that ANLS-BPP and HALS are the best performing algorithms among the ones we tested. The ANLS-PGRAD, the ANLS-PQN, and the MU algorithms showed a trend of convergence although the convergence was slow, but the ALS algorithm showed difficulty in convergence. Among ANLS-BPP and HALS, while HALS generally appeared to converge faster in early iterations,

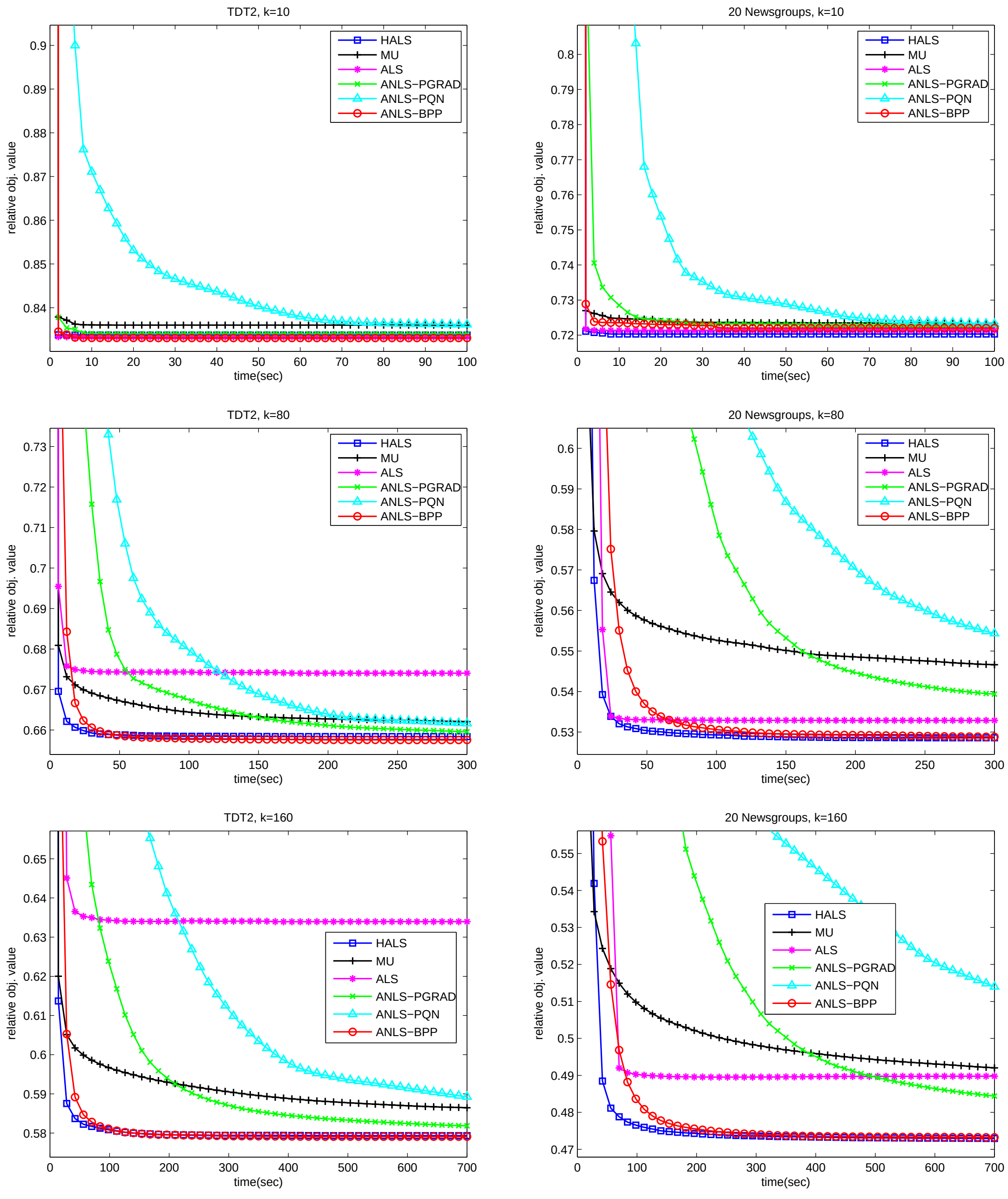


Figure 4.5: Relative objective value ($\|\mathbf{A} - \mathbf{W}\mathbf{H}^T\|_F / \|\mathbf{A}\|_F$) vs. execution time on the TDT2 and the 20 Newsgroups text data sets. Average results of 5 different random initializations are shown. Left column: TDT2, right column: 20 Newsgroups, top row: $K = 10$, middle row: $K = 80$, bottom row: $K = 160$. See text for more details.

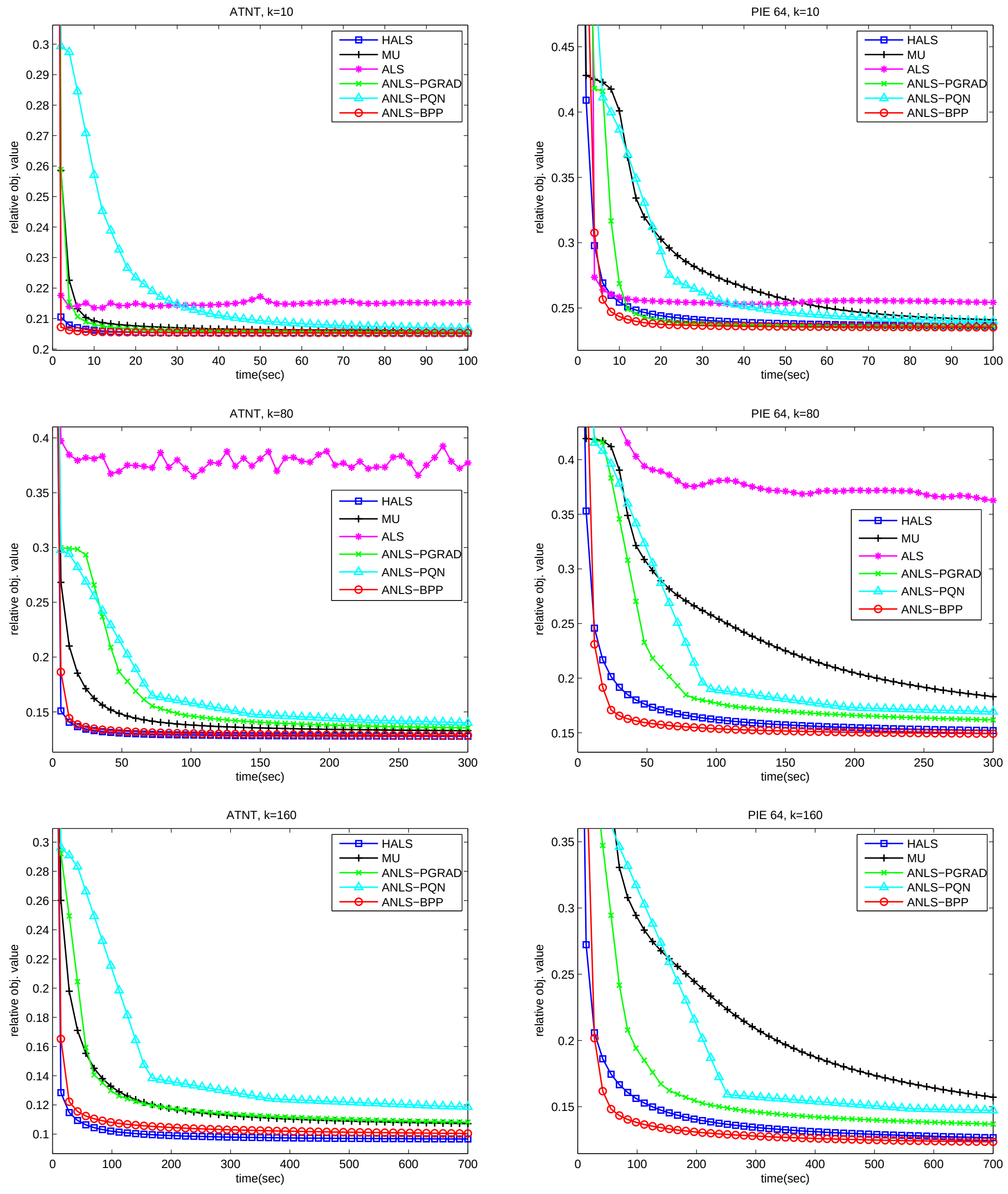


Figure 4.6: Relative objective value ($\|\mathbf{A} - \mathbf{W}\mathbf{H}^T\|_F / \|\mathbf{A}\|_F$) vs. execution time on the ATNT and the PIE 64 image data sets. Average results of 5 different random initializations are shown. Left column: ATNT, right column: PIE 64, top row: $K = 10$, middle row: $K = 80$, bottom row: $K = 160$. See text for more details.

ANLS-BPP was the only algorithm with comparable performance.

In Figure 4.6, execution results on the ATNT and the PIE 64 data sets are presented. Similarly to previous data sets, ANLS-PGRAD, ANLS-PQN, and MU showed slow convergence, and ALS was unpredictable as the relative objective value fluctuated up and down without converging to a small value. In the $K = 160$ cases, the results of ALS are not shown because it was not able to reduce the objective value in the range of the plot. Again, it can be seen that the ANLS-BPP and the HALS algorithms are the best performing ones. Among the two, HALS showed faster convergence in the ATNT data set whereas ANLS-BPP outperformed HALS in the PIE 64 data set.

In Table 4.2, relative norms of the projected gradient defined in Eq. (4.13) after executions for specified durations are shown. It can be seen that ANLS-BPP appeared very successful in minimizing this criterion. A caveat here is that a smaller value of $\frac{\Delta}{\Delta_0}$ does not necessarily imply a smaller objective value or vice versa, as can be seen from the fact that HALS sometimes produced high values of $\frac{\Delta}{\Delta_0}$ although it often showed one of the best performance in terms of the objective value. A partial explanation about these results is given as follows. Note that the diagonal scaling of $\mathbf{W}$ and $\mathbf{H}$ does not affect the quality of approximation: For a diagonal matrix $\mathbf{D} \in \mathbb{R}^{K \times K}$ with positive diagonal elements, $\mathbf{W}\mathbf{H}^T = \mathbf{W}\mathbf{D}^{-1}\mathbf{D}\mathbf{H}^T$. However, the norm of the projected gradients in Eq. (4.12) is affected by a diagonal scaling: It is easy to check that $\left(\frac{\partial f}{\partial(\mathbf{W}\mathbf{D}^{-1})}, \frac{\partial f}{\partial(\mathbf{D}\mathbf{H})}\right) = \left(\left(\frac{\partial f}{\partial \mathbf{W}}\right)\mathbf{D}, \left(\frac{\partial f}{\partial \mathbf{H}}\right)\mathbf{D}^{-1}\right)$. Hence, two solutions that are only different up to a diagonal scaling have the same objective function value, but they can be measured differently in terms of the norm of the projected gradients. In particular, the solution from the HALS algorithm is typically unbalanced having large elements in $\mathbf{W}$ and small elements in $\mathbf{H}$. This can be a reason for the relatively poor evaluation of the HALS algorithm in Table 4.2. Ho [46] considered including a normalization step before computing Eq. (4.12) to avoid this issue.

Table 4.2: The relative norm of the projected gradient (i.e., Δ/Δ_0) after executions of specified amounts of time.

Data set	K	time	ANLS			HALS	MU	ALS
			BPP	PGRAD	PQN			
TDT2	10	100	2.64e-17	1.93e-05	3.82e-04	0.0015	0.0057	3.51e-04
	80	300	1.35e-07	4.33e-05	8.29e-05	8.04e-05	0.0013	1.61e-04
	160	700	2.40e-06	1.61e-05	9.03e-05	3.04e-05	7.47e-04	9.65e-05
20 Newsgroups	10	100	2.45e-08	8.71e-05	0.0020	0.0051	0.0058	0.0025
	80	300	4.05e-05	1.09e-04	0.0013	1.54e-04	0.0012	2.34e-04
	160	700	1.14e-05	8.40e-05	8.52e-04	5.28e-05	6.48e-04	1.33e-04
ATNT	10	100	4.12e-05	1.98e-04	0.0022	3.4601	0.0553	31.6
	80	300	2.56e-04	0.0018	0.0065	0.865	0.0136	57.6
	160	700	4.18e-04	0.0015	0.0059	0.533	0.0115	71.6
PIE 64	10	100	6.43e-04	7.42e-04	0.0065	19.3	0.437	114
	80	300	7.46e-04	0.0034	0.0045	4.32	0.0584	140
	160	700	0.0010	0.0043	0.0050	3.24	0.0369	180

Among the ANLS-based algorithms that have been actively studied recently, results in Figures 4.5 and 4.6 demonstrate that ANLS-BPP is clearly the best. Among all the NMF algorithms, the HALS algorithm showed very promising behaviour as it outperformed ANLS-BPP in some cases. In the following subsection, we further investigated the two algorithms using synthetic data sets.

4.4.3 ANLS-BPP and HALS on synthetic data sets

In order to further understand the behaviour of the ANLS-BPP and the HALS algorithms, we performed experiments using synthetic data sets. Using $M = 10,000$, $N = 2,000$, and $K = 160$, we created factor matrices $\mathbf{W} \in \mathbb{R}^{M\times K}$ and $\mathbf{H} \in \mathbb{R}^{N\times K}$ having 50%, 90%, and 95% sparsities. We then multiplied the factors to obtain $\mathbf{A} = \mathbf{W}\mathbf{H}^T$ upon which the ANLS-BPP and the HALS algorithms were executed.

Figure 4.7 shows the results. Our expectation was that the sparser the factors are, the better ANLS-BPP would perform compared to HALS: If the factors are sparse, the ANLS-BPP method only needs to solve for a small number of nonzero elements in $\mathbf{W}$ and $\mathbf{H}$ matrices whereas the HALS method still needs to update all the elements of $\mathbf{W}$ and $\mathbf{H}$ in each iteration. In the top row of Figure 4.7, when the sparsity of the

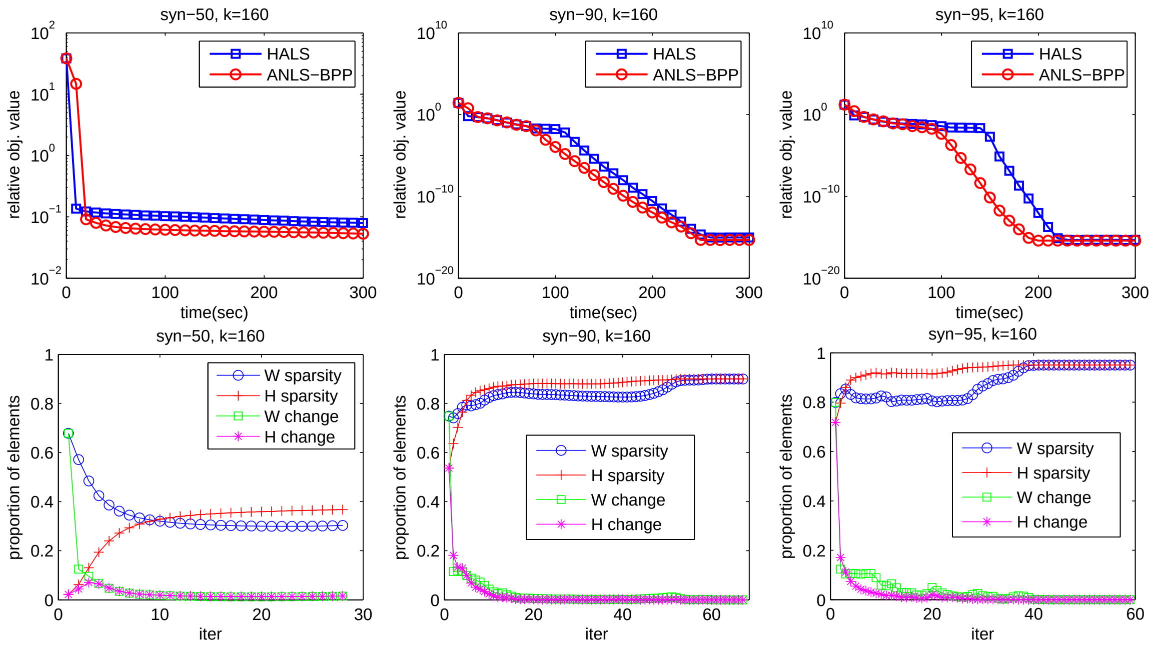


Figure 4.7: Execution results on synthetic data sets with factors of different sparsities. The top row shows relative objective values ($\|\mathbf{A} - \mathbf{W}\mathbf{H}^T\|_F / \|\mathbf{A}\|_F$) with respect to execution time, and the bottom row shows sparsity patterns of **W** and **H** that are obtained from ANLS-BPP. 'W sparsity' and 'H sparsity' show the proportions of zero elements of **W** and **H**, respectively, and 'W change' and 'H change' show the proportions of elements that switched between zero and nonzero in **W** and **H**, respectively. Sparsity in original factors used to create data sets are 50% in the left figures, 90% in the middle figures, and 95% in the right figures. Average results of 5 different random initializations are shown.

original factors increases, ANLS-BPP method showed noticeably faster convergence than HALS. Relevant information is shown in the bottom row: The sparsity pattern of **W** and **H** obtained by ANLS-BPP quickly became close to that of the original factors that were used to create data sets, and the pattern changed only little as iteration progressed. When **W** and **H** are sparse, the cost of each iteration of ANLS-BPP decreases since only the nonzero elements needs to be updated.

4.4.4 Massive scale factorization

In this subsection, we discuss issues in the factorization of massive-scale data as well as experimental results on huge matrices from real-world applications. A critical concern arising in massive-scale factorization is the amount of memory required to hold the

Table 4.3: Memory requirements of NMF algorithms. The data matrix may be represented as a sparse matrix, occupying memory only for nonzero elements.

Method	Common	Additional
ANLS-BPP	$\mathbf{A}$: $\mathbb{R}^{M\times K}$(maybe sparse) $\mathbf{W}$: $\mathbb{R}^{M\times K}$ $\mathbf{H}$: $\mathbb{R}^{N\times K}$ $\mathbf{AH}$ or $\mathbf{A}^T\mathbf{W}$: $\mathbb{R}^{\max\{M,N\}\times K}$ $\mathbf{H}^T\mathbf{H}$ or $\mathbf{W}^T\mathbf{W}$: $\mathbb{R}^{K\times K}$	$\{0,1\}^{\max\{M,N\}\times K}$+$\mathbb{R}^{K\times K}$
ANLS-AS-GROUP		
ANLS-AS-UPDATE		$\mathbb{R}^K$+$\mathbb{R}^{K\times K}$
ANLS-PGRAD		$\mathbb{R}^{2\times\max\{M,N\}\times K}$
ANLS-PQN		$\mathbb{R}^K$+$\mathbb{R}^{K\times K}$
HALS		N/A
MU		N/A
ALS		$\mathbb{R}^{K\times K}$

data matrix $\mathbf{A}$, factor matrices $\mathbf{W}$ and $\mathbf{H}$, and other information necessary to execute an algorithm. In Table 4.3, we present the memory requirement of the algorithms compared in this chapter. For all the algorithms, the following common information is needed in memory. Input matrix $\mathbf{A}$ may be represented in a sparse format because it is used only through $\mathbf{AH}$ and $\mathbf{A}^T\mathbf{W}$, for which a multiplication operator suffices. Typically, K is quite small in NMF applications, but, for a large matrices, factor matrices $\mathbf{W}$ and $\mathbf{H}$ could take a large amount of memory. In addition, matrix $\mathbf{AH}$ or $\mathbf{A}^T\mathbf{W}$ take as much as the factor matrices: Observe that either $\mathbf{AH}$ or $\mathbf{A}^T\mathbf{W}$ is needed to update $\mathbf{W}$ and $\mathbf{H}$, respectively; therefore, space for $\max\{M,N\}\times K$ real numbers is needed. Similarly, to store $\mathbf{H}^T\mathbf{H}$ or $\mathbf{W}^T\mathbf{W}$, space for $K\times K$ real numbers is needed. If the stopping criterion in Section 4.3.2 is adopted, additional memory is needed to compute Δ in Eq. (4.12).

In addition to the common information, some algorithms spend additional memory in their operation. In ANLS-based algorithms, observe that the subproblems in Eq. (4.1) involve variables of size $M\times K$ and $N\times K$, so the NLS subproblem with at most $\max\{M,N\}\times K$ variables needs to be handled. For the ANLS-BPP method equipped with the grouping acceleration for multiple right hand sides, the information of workings sets, $\mathcal{F}$ and $\mathcal{G}$, as binary numbers should kept track of, and space to store the Cholesky factor is needed when solving normal equations as in Eq. (4.9). Variable

$\mathbf{Y}$ in Algorithm 4.2 does not take additional space due to the complementarity condition in Eq. (4.3d). For the ANLS-AS-GROUP method, which involves the grouping acceleration, the same additional space with ANLS-BPP is needed; for the ANLS-AS-UPDATE method, only small space is enough because the NLS problem is solved for each right hand side separately. The ANLS-PGRAD method spend additional space for storing a gradient matrix and selecting step length. The ANLS-PQN method separately solves the NLS problem for each right hand side, and therefore the required amount of space is small. Method that are not based on the ANLS-framework require a less amount of memory than the ANLS-based methods. No additional memory is needed in the HALS and MU methods, and the ALS method only need space to store the Cholesky factor.

Having the memory characteristics noted, let us see the performance of the methods on the Netflix's movie ratings[8] and RCV1 text corpus[9] data sets, shown in Figure 4.8. The Netflix's movie ratings matrix is of size $17,770 \times 480,189$ with about 100 million nonzero entries, and the term-document matrix from the RCV1 text corpus is of size $34,803 \times 804,427$ with about 60 million nonzero entries. Experiments on Netflix and RCV1 data sets were executed on Linux machines with 16GB and 48GB memory, respectively, with the multi-threading option of MATLAB disabled. The results of ANLS-AS and ALS are not shown because of too expensive iterations and failure to converge, respectively. Similarly to previous results, ANLS-BPP and HALS showed the best performance, significantly outperforming the rest. Note that due to the massive size of the data matrix, the amount of execution time is much larger than those in Figure 4.5 and 4.6. In the $K = 80$ and the $K = 160$ cases, MU, ANLS-PGRAD, and ANLS-PQN were slower than ANLS-BPP and HALS with

[8]`http://www.netflixprize.com/`

[9]`http://trec.nist.gov/data/reuters/reuters.html`

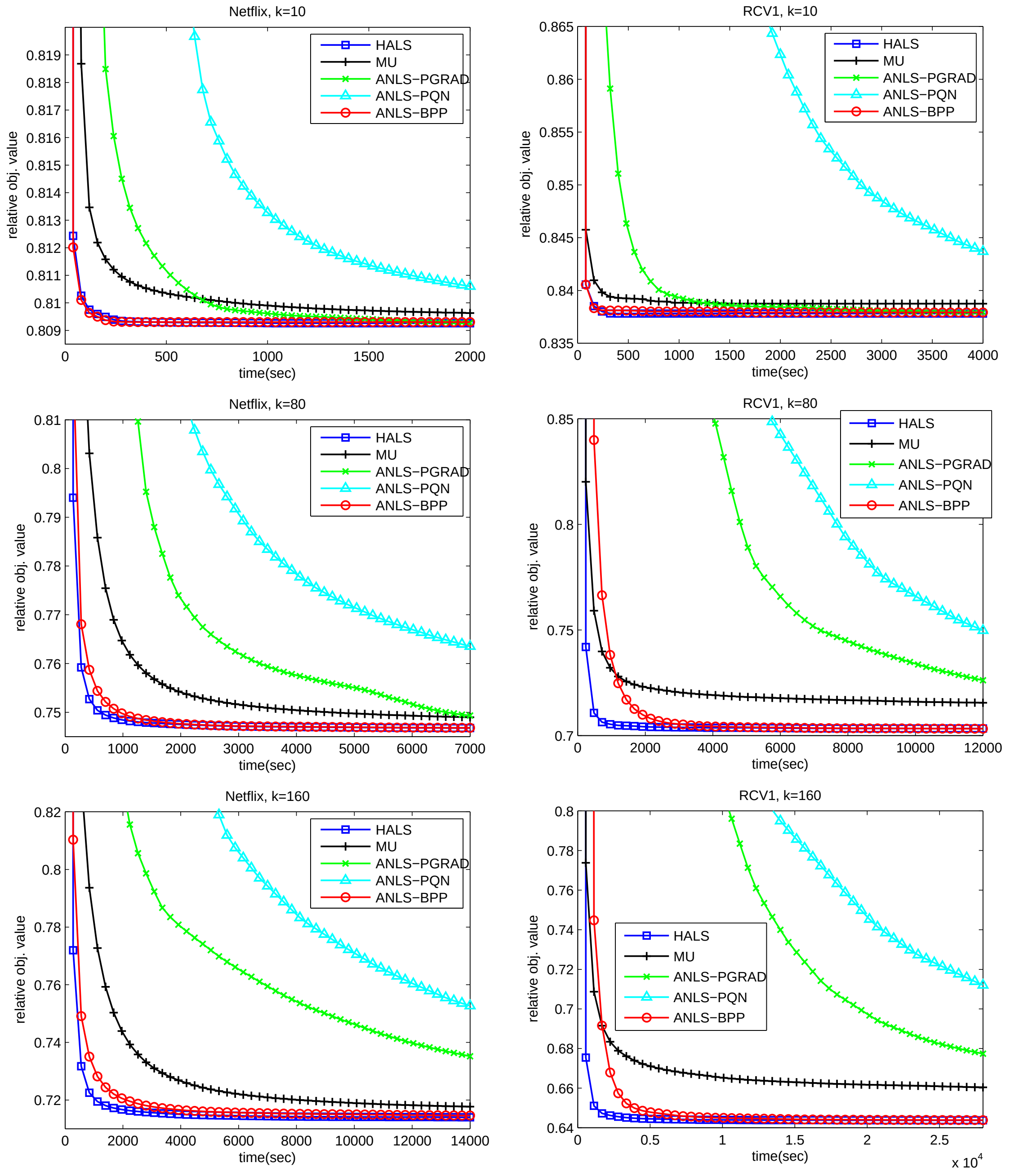


Figure 4.8: Relative objective value ($\left\|\mathbf{A}-\mathbf{W}\mathbf{H}^T\right\|_F / \left\|\mathbf{A}\right\|_F$) vs. execution time on the Netflix and the RCV1 data sets. Average results of 5 different random initializations are shown. Left column: Netflix, right column: RCV1, top row: $K = 10$, middle row: $K = 80$, bottom row: $K = 160$. See text for more details.

several hours of difference in computation time. It is common to repeat NMF computation with several initial values to find the best factor matrices, and then the difference in computation time becomes even more significant.

4.5 Discussion

In this chapter, a new algorithm for computing NMF based on the ANLS framework is proposed. The new algorithm is built upon the block principal pivoting method for the NLS problem. The method allows exchanges of multiple variables between working sets with a goal to reach the final partitioning into the active and passive sets quickly. We improved the method to handle the multiple right-hand side case of the NLS problems efficiently. The newly constructed algorithm inherits the convergence theory of the ANLS framework, and it can easily be extended to other constrained NMF formulations such as sparse or regularized NMF. Thorough experimental comparisons with recently developed NMF algorithms were performed using both real-world and synthetic data sets. The proposed algorithm demonstrated state-of-the-art performance allowing only the hierarchical alternating least squares (HALS) algorithm to be comparable. In a test using synthetic data sets, the proposed algorithm clearly outperformed the HALS algorithm when the low-rank factors are sparse.

Although we explored various values for K (from 10 to 160), it has to be understood that all these values are much smaller than the original dimension. This trend is generally expected for a dimension reduction method, as mentioned in Section 4.1. We emphasize that the long-and-thin structure of the coefficient matrix and the flat-and-wide structure of the matrix with unknowns are key features that enables us to use speed-up techniques explained in Section 4.2.2 and thus obtain the successful experimental results of our new algorithm.

Different limitations of different algorithms can be noted. A limitation of a NMF algorithm based on the ANLS framework with the active-set or the block principal pivoting method is that it may break down if matrix $\mathbf{B}$ in Eq. (4.2) does not have full column rank. The regularization method mentioned in Section 3.1.4 can be adopted to remedy this problem making these algorithms generally applicable for the computation of NMF. Even without regularization, the algorithms behave well in practice as shown in our experiments. An algorithm based on the ANLS framework with the projected quasi-Newton method [54] also requires the full column rank condition. On the other hand, the HALS algorithm breaks down when either a column of $\mathbf{W}$ or a row of $\mathbf{H}$ becomes a zero vector in the process of iterations. As this problem indeed happens quite often, typically a small number $\epsilon \approx 1e^{-16}$ is used in places that are supposed to be zero [26, 40]. Due to this modification, the factors obtained by the HALS algorithm are not sparse unless explicitly thresholded afterwards.

CHAPTER V

A FAST ACTIVE-SET-LIKE METHODS FOR NONNEGATIVE TENSOR FACTORIZATION

Tensors are mathematical objects for representing multidimensional arrays. Tensors include vectors and matrices as first-order and second-order special cases, respectively, and more generally, tensors of N^{th}-order can represent an outer product of N vector spaces. Recently, decompositions and low-rank approximations of tensors have been actively studied and applied in numerous areas including signal processing, image processing, data mining, and neuroscience. Several different decomposition models, their algorithms, and applications are summarized in recent reviews by Kolda and Bader [62] and Acar and Yener [1].

We discuss tensors with nonnegative elements and their low-rank approximations. In particular, we are interested in computing a CANDECOMP/PARAFAC decomposition [20, 45] with nonnegativity constraints on factors. In higher-order tensors with nonnegative elements, tensor factorizations with nonnegativity constraints on factors have been developed in several papers [15, 83, 102, 21]. Interestingly, some methods for finding nonnegative factors of higher-order tensors were introduced even before nonnegative matrix factorization [21]. Recent work dealt with properties such as degeneracy [70] and applications such as sound source separation [35], text mining [4], and computer vision [93]. Although we focus on CANDECOMP/PARAFAC decomposition, a few other decomposition models of higher order tensors have been studied [62, 1].

Suppose a tensor of order three, $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{M_1 \times M_2 \times M_3}$, is given. We introduce main concepts using this third-order tensor for the sake of simplicity and will deal with a tensor with a general order later. A canonical decomposition (CANDECOMP) [20], or equivalently a parallel factorization (PARAFAC) [45], of $\boldsymbol{\mathcal{A}}$ can be written as

$$\boldsymbol{\mathcal{A}} = \sum_{k=1}^{K} \mathbf{a}_k \circ \mathbf{b}_k \circ \mathbf{c}_k, \tag{5.1}$$

where $\mathbf{a}_k \in \mathbb{R}^{M_1}$, $\mathbf{b}_k \in \mathbb{R}^{M_2}$, $\mathbf{c}_k \in \mathbb{R}^{M_3}$, and "$\circ$" represents an outer product of vectors. Following Kolda and Bader [62], we will call the decomposition in Eq. (5.1) the CP (CANDECOMP/PARAFAC) decomposition. A tensor $\mathbf{a} \circ \mathbf{b} \circ \mathbf{c}$ is called a *rank-one* tensor: In the CP decomposition, tensor $\boldsymbol{\mathcal{A}}$ is represented as a sum of K rank-one tensors. A smallest integer K for which Eq. (5.1) holds with some vectors $\mathbf{a}_k$, $\mathbf{b}_k$, and $\mathbf{c}_k$ for $k \in \{1, \cdots K\}$ is called the *rank* of tensor $\boldsymbol{\mathcal{A}}$. The CP decomposition can be more compactly represented with *factor matrices* (or *loading matrices)*, $\mathbf{A} = [\mathbf{a}_1 \cdots \mathbf{a}_K]$, $\mathbf{B} = [\mathbf{b}_1 \cdots \mathbf{b}_K]$, and $\mathbf{C} = [\mathbf{c}_1 \cdots \mathbf{c}_K]$, as follows:

$$\boldsymbol{\mathcal{A}} = [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!],$$

where $[\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!] = \sum_{k=1}^{K} \mathbf{a}_k \circ \mathbf{b}_k \circ \mathbf{c}_k$. With a tensor $\boldsymbol{\mathcal{A}}$ of rank R, given an integer $K \leq R$, the computational problem of the CP decomposition is finding factor matrices $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$ that best approximates $\boldsymbol{\mathcal{A}}$.

Now, for a tensor $\boldsymbol{\mathcal{A}}$ with only nonnegative elements, we are interested in recovering factor matrices $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$ that also contain only nonnegative elements. Using the Frobenius norm as the criterion of approximation, the factor matrices can be found by solving an optimization problem:

$$\min_{\mathbf{A},\mathbf{B},\mathbf{C}} \frac{1}{2} \left\| \boldsymbol{\mathcal{A}} - [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!] \right\|_F^2 \text{ s.t. } \mathbf{A}, \mathbf{B}, \mathbf{C} \geq 0. \tag{5.2}$$

Inequalities $\mathbf{A}, \mathbf{B}, \mathbf{C} \geq 0$ denote that all the elements of $\mathbf{A}, \mathbf{B}$, and $\mathbf{C}$ are nonnegative. The factorization problem in Eq. (5.2) is known as nonnegative CP (NCP). The computation of NCP is demanding not only because many variables are involved in optimization but also because nonnegativity constraints are imposed on the factors.

A number of algorithms have been developed for NCP [15, 57, 36, 102], and we will review them in Section 5.1.

In this chapter, we present a new and efficient algorithm for computing NCP. Similarly to NMF, we apply the block coordinate descent (BCD) method of Section 2.3 and observe that the subproblems appear in the form of nonnegativity-constrained least squares (NLS) problems. The NLS subproblems can in turn be solved by the block principal pivoting method presented in Chapter 4.

The remaining of this chapter is organized as follows. In Section 5.1, related work is reviewed. In Section 5.2, the BCD methods for NCP are described, and the application of the block principal pivoting method is explained in Section 5.3. In Section 5.4, we describe how the proposed method can be used to solve regularized and sparse formulations. In Section 5.5, experimentation settings and results are shown. We conclude this chapter in Section 5.6.

5.1 Related Work

Several computational methods for NCP have been proposed based on the BCD method with matrix blocks, which we describe in Section 5.2.1 and call the alternating nonnegative least squares (ANLS) framework. A classical method for solving the NLS problem is the active-set method of Lawson and Hanson [63]; however, applying Lawson and Hanson's method directly to NCP is extremely slow. Bro and De Jong [15] suggested an improved active-set method to solve the NLS problems, and Ven Benthem and Keenan [99] further accelerated the active-set method, which was later utilized in NMF [56] and NCP [57]. In Friedlander and Hatz [36], the NCP subproblems are solved by a two-metric projected gradient descent method.

Numerous other algorithms that are not based on the ANLS framework were suggested. Paatero discussed a Gauss-Newton method [83] and a conjugate gradient method [84], but nonnegativity constraints were not rigorously handled in those work.

Extending the multiplicative updating rule of Lee and Seung [65], Welling and Weber [102] proposed a multiplicative updating method for NCP. Earlier in [21], Carroll et al. proposed a simple procedure that focuses on a rank-one approximation conditioned that other variables are fixed. Recently, Cichocki et al. proposed a similar algorithm, called hierarchical alternating least squares (HALS), which updates each column of factor matrices at a time [25].

5.2 BCD Framework for NCP

Let us describe the BCD method applied to the NCP problem. Toward that end, we consider a general N^{th}-order tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{M_1 \times \cdots \times M_N}$ and its nonnegative CANDECOMP/PARAFAC (CP) decomposition. For an integer K, we are interested in finding nonnegative factor matrices $\mathbf{H}^{(1)}, \cdots, \mathbf{H}^{(N)}$ where $\mathbf{H}^{(n)} \in \mathbb{R}^{M_n \times K}$ for $n = 1, \cdots, N$ such that

$$\boldsymbol{\mathcal{A}} \approx [\![\mathbf{H}^{(1)}, \cdots, \mathbf{H}^{(N)}]\!], \tag{5.3}$$

where

$$[\![\mathbf{H}^{(1)}, \cdots, \mathbf{H}^{(N)}]\!] = \sum_{k=1}^{K} \mathbf{h}_k^{(1)} \circ \cdots \circ \mathbf{h}_k^{(N)}. \tag{5.4}$$

Note that the CP decomposition reduces to matrix decomposition if $N = 2$. When K is small, which is typical for low-rank approximation methods, we find factor matrices that best approximate a tensor with the CP model. A corresponding optimization problem can be written as

$$\min_{\mathbf{H}^{(1)}, \cdots, \mathbf{H}^{(N)}} f(\mathbf{H}^{(1)}, \cdots, \mathbf{H}^{(N)}) = \frac{1}{2} \left\| \boldsymbol{\mathcal{A}} - [\![\mathbf{H}^{(1)}, \cdots, \mathbf{H}^{(N)}]\!] \right\|_F^2 \tag{5.5}$$

$$\text{s.t. } \mathbf{H}^{(n)} \geq 0 \text{ for } n = 1, \cdots N.$$

To present how the BCD method is applied to the NCP problem in Eq. (5.5), we need definitions of some operations of tensors. See Kolda and Bader [62] for more details.

Mode-n matricization The mode-n matricization of a tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{M_1 \times \cdots \times M_N}$, denoted by $\mathbf{A}^{(n)}$, is a matrix obtained by linearizing all indices except n. More formally, $\mathbf{A}^{(n)}$ is a matrix of size $M_n \times \prod_{i=1,i\neq n}^{N} M_i$, and the $(m_1, \cdots, m_N)^{th}$ element of $\boldsymbol{\mathcal{A}}$ is mapped to the $(m_n, I)^{th}$ element of $\mathbf{A}^{(n)}$ where

$$I = 1 + \sum_{i=1}^{N} (m_i - 1) I_i \text{ and } I_i = \prod_{j=1, j\neq n}^{i-1} M_j.$$

Mode-n product The mode-n product of a tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{M_1 \times \cdots \times M_N}$ and a matrix $\mathbf{U} \in \mathbb{R}^{J \times M_n}$ denoted by $\boldsymbol{\mathcal{A}} \times_n \mathbf{U}$, is a tensor obtained by multiplying all mode-n fibers of $\boldsymbol{\mathcal{A}}$ with the columns of $\mathbf{U}$. The result is a tensor of size $M_1 \times \cdots \times M_{n-1} \times J \times M_{n+1} \times \cdots \times M_N$ having elements as

$$(\boldsymbol{\mathcal{A}} \times_n \mathbf{U})_{m_1 \cdots m_{n-1} j m_{n+1} \cdots m_N} = \sum_{m_n=1}^{M_n} x_{m_1 \cdots m_N} u_{j m_n}.$$

In particular, the mode-n product of $\boldsymbol{\mathcal{A}}$ and a vector $\mathbf{u}$ is a tensor of size $M_1 \times \cdots \times M_{n-1} \times M_{n+1} \times \cdots \times M_N$.

Khatri-Rao product The Khatri-Rao product of two matrices $\mathbf{A} \in \mathbb{R}^{J_1 \times L}$ and $\mathbf{B} \in \mathbb{R}^{J_2 \times L}$, denoted by $\mathbf{A} \odot \mathbf{B} \in \mathbb{R}^{(J_1 J_2) \times L}$, is defined as

$$\mathbf{A} \odot \mathbf{B} = \begin{bmatrix} a_{11}\mathbf{b}_1 & a_{12}\mathbf{b}_2 & \cdots & a_{1L}\mathbf{b}_L \\ a_{21}\mathbf{b}_1 & a_{22}\mathbf{b}_2 & \cdots & a_{2L}\mathbf{b}_L \\ \vdots & \vdots & \ddots & \vdots \\ a_{J_1 1}\mathbf{b}_1 & a_{J_1 2}\mathbf{b}_2 & \cdots & a_{J_1 L}\mathbf{b}_L \end{bmatrix}.$$

5.2.1 BCD with N matrix blocks

A simple case of the BCD method is that each of the factor matrices $\mathbf{H}^{(1)}, \cdots, \mathbf{H}^{(N)}$ are considered as a block. Using above notations, the approximation model in Eq. (5.3) can be written as, for any $n \in \{1, \cdots, N\}$,

$$\mathbf{A}^{(n)} \approx \mathbf{H}^{(n)} \left(\mathbf{B}^{(n)}\right)^T, \tag{5.6}$$

where

$$\mathbf{B}^{(n)} = \mathbf{H}^{(N)} \odot \cdots \odot \mathbf{H}^{(n+1)} \odot \mathbf{H}^{(n-1)} \odot \cdots \odot \mathbf{H}^{(1)} \in \mathbb{R}^{\left(\prod_{i=1,i\neq n}^{N} M_i\right) \times K}. \tag{5.7}$$

Eq. (5.6) simplifies the treatment of this N matrix block case. Using the BCD method, $\mathbf{H}^{(2)}, \cdots, \mathbf{H}^{(N)}$ are initialized with arbitrary nonnegative elements. Then, for $n = 1, \cdots N$, the following subproblem is solved iteratively:

$$\mathbf{H}^{(n)} \leftarrow \arg\min_{\mathbf{H} \geq 0} \left\| \mathbf{B}^{(n)} \mathbf{H}^T - \left(\mathbf{A}^{(n)}\right)^T \right\|_F^2 . \tag{5.8}$$

Observe that Eq. (5.8) is the nonnegativity-constrained least squares (NLS) problem. Because the subproblems are in the NLS problems, similarly to the matrix factorization case in Chapter 4, we call this matrix-block BCD method as the alternating nonnegative least squares (ANLS) framework.

The convergence property of a BCD method in Theorem 2.14 states that if each subproblem in the form of Eq. (5.8) has a unique solution, then every limit point produced by the ANLS framework is a stationary point. In particular, if matrices $\mathbf{B}^{(n)}$ are of full column rank, each subproblem has a unique solution.

For higher order tensors, the row size of $\mathbf{B}^{(n)}$ and $\left(\mathbf{A}^{(n)}\right)^T$, that is, $\prod_{i=1, i\neq n}^{N} M_i$, can be quite large. However, often they need not be explicitly constructed, and it is enough to have $\left(\mathbf{B}^{(n)}\right)^T \left(\mathbf{A}^{(n)}\right)^T$ and $\left(\mathbf{B}^{(n)}\right)^T \mathbf{B}^{(n)}$. $\left(\mathbf{B}^{(n)}\right)^T \left(\mathbf{A}^{(n)}\right)^T$ can be computed by successive mode-n product of $\boldsymbol{\mathcal{A}}$ with $\mathbf{H}^{(1)},\cdots,\mathbf{H}^{(n-1)},\mathbf{H}^{(n+1)},\cdots,\mathbf{H}^{(N)}$. $\left(\mathbf{B}^{(n)}\right)^T \mathbf{B}^{(n)}$ can be efficiently computed as

$$\left(\mathbf{B}^{(n)}\right)^T \mathbf{B}^{(n)} = \bigotimes_{i=1, i\neq n}^{N} \left(\mathbf{H}^{(i)}\right)^T \mathbf{H}^{(i)},$$

where $\bigotimes$ represents element-wise multiplication.

5.2.2 BCD with KN vector blocks

Another case of the BCD method applied to the NCP problem is that each column of matrices $\mathbf{H}^{(1)}, \cdots, \mathbf{H}^{(N)}$ is used as a block. The subproblem pertaining only one column can be written as

$$\mathbf{h}_k^{(n)} \leftarrow \arg\min_{\mathbf{h} \geq 0} \left\| [\![\mathbf{h}_k^{(1)}, \cdots, \mathbf{h}_k^{(n-1)}, \mathbf{h}, \mathbf{h}_k^{(n+1)}, \cdots, \mathbf{h}_k^{(N)}]\!] - \boldsymbol{\mathcal{R}}_k \right\|_F^2 . \tag{5.9}$$

where

$$\boldsymbol{\mathcal{R}}_k = \boldsymbol{\mathcal{A}} - \sum_{\tilde{k}=1, \tilde{k}\neq k}^{K} \mathbf{h}_{\tilde{k}}^{(1)} \circ \cdots \circ \mathbf{h}_{\tilde{k}}^{(N)}.$$

Using matrix notation, the problem in Eq. (5.9) can be rewritten as

$$\mathbf{h}_k^{(n)} \leftarrow \arg\min_{\mathbf{h}\geq 0} \left\| \mathbf{b}_k^{(n)} \mathbf{h}^T - \left(\mathbf{R}_k^{(n)}\right)^T \right\|_F^2, \tag{5.10}$$

where $\mathbf{R}_k^{(n)}$ is the mode-n matricization of $\boldsymbol{\mathcal{R}}_k$ and

$$\mathbf{b}_k^{(n)} = \mathbf{h}_k^{(N)} \odot \cdots \odot \mathbf{h}_k^{(n+1)} \odot \mathbf{h}_k^{(n-1)} \odot \cdots \odot \mathbf{h}_k^{(1)} \in \mathbb{R}^{\left(\prod_{i=1, i\neq n}^{N} M_i\right)\times 1}.$$

If $\mathbf{b}_k^{(n)}$ is a nonzero vector, based on Theorem 3.2, the unique solution of Eq. (5.10) is,

$$\mathbf{h}_k^{(n)} \leftarrow \left[\frac{\mathbf{R}_k^{(n)} \mathbf{b}_k^{(n)}}{\left(\mathbf{b}_k^{(n)}\right)^T \mathbf{b}_k^{(n)}} \right]_+ .$$

Cichocki et al. [24] proposed this vector-block BCD method with the name of the hierarchical alternating least squares (HALS) method, and we present a comparison with this method in Section 5.5. In view of Theorem 2.14, for the sequence of solutions generated by the HALS method, every limit point is a stationary point. A condition for this property is that $\mathbf{b}_k^{(n)}$ is nonzero for all $\forall k$ and $\forall n$.

5.3 Block Principal Pivoting Method

The proposed algorithm for computing NCP is based on the ANLS framework in Section 5.2.1. Observe the following characteristics of the NLS subproblems in Eq. (5.8). Due to flattening by the Khatri-Rao product, matrix $\mathbf{B}^{(n)}$ in Eq. (5.8) is typically long and thin. Also, as NCP is often used for low-rank approximation, matrix $\left(\mathbf{H}^{(n)}\right)^T$ in Eq. (5.8) is typically flat and wide. Recall that the NLS subproblems of NMF in Chapter 4 have the same characteristics and that the block principal pivoting algorithm with accelerations for multiple right hand sides solves the subproblems very efficiently. We therefore propose to solve Eq. (5.8) with the block principal pivoting algorithm presented in Section 4.2 with accelerations for multiple right hand sides.

5.4 Regularized and Sparse NCP

We note that the BCD methods described in Section 5.2 can be easily extended to formulations with regularization. We focus on the ANLS framework described in Section 5.2.1, but the application of regularization is also straightforward to the HALS method.

In a general form, a regularized formulation appears as

$$\min_{\mathbf{H}^{(1)},\cdots,\mathbf{H}^{(N)}} \frac{1}{2}\left\|\boldsymbol{\mathcal{A}} - [\![\mathbf{H}^{(1)},\cdots,\mathbf{H}^{(N)}]\!]\right\|_F^2 + \sum_{n=1}^{N} \lambda_n \phi_n(\mathbf{H}^{(n)}), \tag{5.11}$$

$$\text{s.t. } \mathbf{H}^{(n)} \geq 0 \text{ for } n = 1, \cdots N,$$

where $\phi_n(\mathbf{H}^{(n)})$ represents a regularization term and $\lambda_n \geq 0$ is a parameter to be chosen. A commonly used regularization term is the Frobenius norm:

$$\phi_n(\mathbf{H}^{(n)}) = \left\|\mathbf{H}^{(n)}\right\|_F^2.$$

In this case, the subproblem for finding $\mathbf{H}^{(n)}$ is modified as

$$\min_{\mathbf{H}^{(n)}} \left\| \begin{pmatrix} \mathbf{B}^{(n)} \\ \sqrt{2\lambda_n}\mathbf{I}_{K\times K} \end{pmatrix} \times \left(\mathbf{H}^{(n)}\right)^T - \left(\mathbf{A}^{(n)}\right)^T \right\|_F^2 \tag{5.12}$$

$$\text{s.t. } \mathbf{H}^{(n)} \geq 0,$$

where $\mathbf{I}_{K\times K}$ is a $K \times K$ identity matrix. Observe that matrix $\begin{pmatrix} \mathbf{B}^{(n)} \\ \sqrt{2\lambda_n}\mathbf{I}_{K\times K} \end{pmatrix}$ is always of full column rank if $\lambda_n > 0$; hence, when $\mathbf{B}^{(n)}$ is not necessarily of full column rank, the Frobenius norm regularization can be adopted to ensure that the NLS subproblem is formed with only a matrix of full column rank, satisfying the requirement of the convergence property of the BCD method, mentioned in Section 5.2.1. In addition, the block principal pivoting method assumes that the matrix $\mathbf{B}$ in Eq. (4.8) is of full column rank, and the Frobenius norm regularization automatically satisfies this condition.

If it is desired to promote sparsity on factor matrix $\mathbf{H}^{(n)}$, l_1-norm regularization can be used:

$$\phi_n(\mathbf{H}^{(n)}) = \sum_{j=1}^{M_n} \left\| \mathbf{h}_{j\cdot}^{(n)} \right\|_1^2 ,$$

where $\mathbf{h}_{j\cdot}^{(n)}$ represents the j^{th} row of $\mathbf{H}^{(n)}$. See [55, 58] for applications of this l_1-norm regularization in microarray data analysis and clustering. In this case, the subproblem for finding $\mathbf{H}^{(n)}$ is modified as

$$\min_{\mathbf{H}^{(n)}} \left\| \begin{pmatrix} \mathbf{B}^{(n)} \\ \sqrt{2\lambda_n}\mathbf{1}_{1\times K} \end{pmatrix} \times \left(\mathbf{H}^{(n)}\right)^T - \left(\mathbf{A}^{(n)}\right)^T \right\|_F^2 \tag{5.13}$$
$$\text{s.t. } \mathbf{H}^{(n)} \geq 0,$$

where $\mathbf{1}_{1\times K}$ is a row vector of ones. Regularization term $\phi_n(\cdot)$ can be separately chosen for each factor $\mathbf{H}^{(n)}$, and if necessary, both of the Frobenius norm and the l_1-norm may be used.

5.5 Implementation and Results

In this section, we describe the details of our implementation, data sets used, and comparison results. All experiments were executed in MATLAB on a Linux machine with a 2.66GHz Intel Quad-core processor and 6GB memory. The multi-threading option of MATLAB was disabled. In all the executions, all the algorithms were provided with the same initial values.

5.5.1 Algorithms and data sets

The following algorithms for NCP were included in our comparison.

1. (ANLS-BPP) ANLS with the block principal pivoting method proposed in this chapter

2. (ANLS-AS) ANLS with H. Kim and Park's active-set method [57]

3. (HALS) Cichocki and Phan's hierarchical alternating least squares algorithm [25, 24]

4. (MU) Welling and Weber's multiplicative updating algorithm [102]

We implemented all algorithms in MATLAB. Besides above methods, we also have tested Friedlander and Hatz's two-metric projected gradient method [36] using their MATLAB code[1]; however, not only it was much slower than methods listed above, but it also required so much memory that we could not execute all comparison cases. We hence do not include the results of Friedlander and Hatz's method here. In all the algorithms, once we obtain factors $\left\{\mathbf{H}^{(1)}, \cdots, \mathbf{H}^{(N)}\right\}$, they are used as initial values of the next iteration.

We have used three data sets for comparisons. The first data set include dense tensors using synthetically generated factors. For each of $K = 10, 20, 60$, and 120, we constructed $\mathbf{H}^{(1)}$, $\mathbf{H}^{(2)}$, and $\mathbf{H}^{(3)}$ of size $300 \times K$ using random numbers from the uniform distribution over $[0, 1]$. Then, we randomly selected 50 percent of elements in $\mathbf{H}^{(1)}$, $\mathbf{H}^{(2)}$, and $\mathbf{H}^{(3)}$ to make them zero. Finally, a three way tensor of size $300 \times 300 \times 300$ is constructed by $[\![\mathbf{H}^{(1)}, \mathbf{H}^{(2)}, \mathbf{H}^{(3)}]\!]$. Different tensors were created for different K values.

The second data set is a dense tensor obtained from Extended Yale Face Database B.[2] We used aligned and cropped images of size 168×192. From total 2424 images, we obtained a three-way tensor of size $168 \times 192 \times 2424$.

The third data set is a sparse tensor from NIPS conference papers.[3] This data set contains NIPS papers volume 0 to 12, and a tensor is constructed as a four-way tensor representing author$\times$documents$\times$term$\times$year. By counting the occurrence of each entry, a sparse tensor of size $2037 \times 1740 \times 13649 \times 13$ was created.

[1]http://www.cs.ubc.ca/~mpf/2008-computing-nntf.html
[2]http://vision.ucsd.edu/~leekc/ExtYaleDatabase/ExtYaleB.html
[3]http://www.cs.nyu.edu/~roweis/data.html

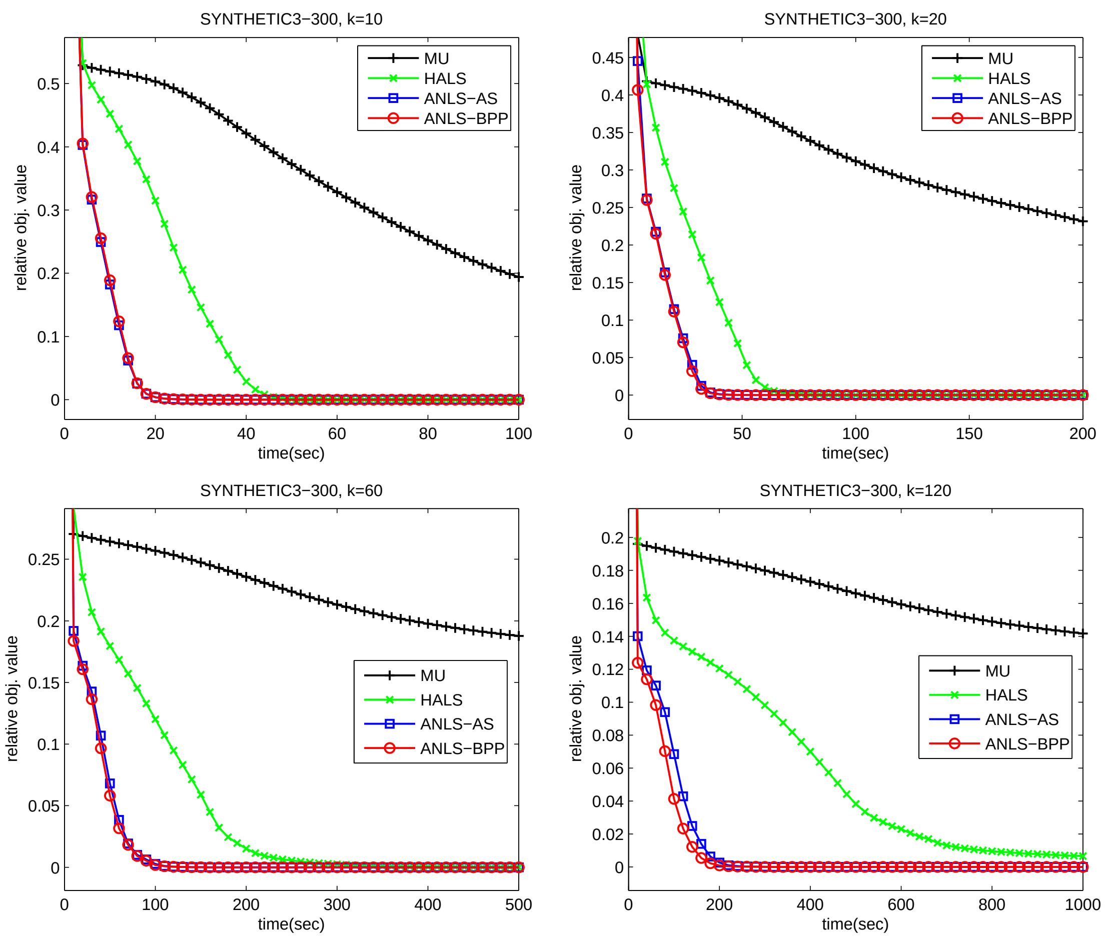


Figure 5.1: Relative objective value ($\left\|\mathcal{A} - [\![\mathbf{H}^{(1)}, \cdots, \mathbf{H}^{(N)}]\!]\right\|_F / \left\|\mathcal{A}\right\|_F$) vs. execution time on the synthetic tensors. Average results of 5 different random initializations are shown. Top row: $K = 10$ and $K = 20$, bottom row: $K = 60$ and $k = 120$.

5.5.2 Experimental results

To observe the performance of several algorithms, at the end of each iteration we have recorded the relative objective value, $\left\|\mathcal{A} - [\![\mathbf{H}^{(1)}, \cdots, \mathbf{H}^{(N)}]\!]\right\|_F / \left\|\mathcal{A}\right\|_F$. Time spent to compute the objective value is excluded from the execution time. One execution result involves relative objective values measured at discrete time points and appears as a piecewise-linear function. We averaged piecewise-linear functions from different random initializations to plot figures.

Results on the synthetic data set are shown in Figure 5.1. This data set was synthetically created, and the value of global optimum is zero. From Figure 5.1, it can be seen that ANLS-AS and ANLS-BPP performed the best among the algorithms we tested. The HALS method showed convergence within the time window we have observed, but the MU method was too slow to show convergence. ANLS-AS and ANLS-BPP showed almost the same performance although ANLS-BPP was slightly faster when $K = 120$. The difference between these two methods are better shown in next results.

Results on YaleB and NIPS data sets are shown in Figure 5.2. Similarly to the results in Figure 5.1, ANLS-AS and ANLS-BPP showed the best performance. In Figure 5.2, it can be clearly observed that ANLS-BPP outperforms ANLS-AS for $K = 60$ and $K = 120$ cases. Such a difference demonstrates a difficulty of the active-set method: Since typically only one variable is exchanged between working sets, the active-set method is slow for a problem with a large number of variables. On the other hand, the block principal pivoting method quickly solves large problems by allowing exchanges of multiple variables between $\mathcal{F}$ and $\mathcal{G}$. The convergence of HALS and MU was slower than ANLS-AS and ANLS-BPP. Although the convergence of HALS was faster than MU in the YaleB data set, the initial convergence of MU was faster than HALS in the NIPS data set.

Lastly, we present more detailed information regarding the executions of ANLS-AS and ANLS-BPP in Figure 5.3. In Figure 5.1 and Figure 5.2, we have observed that ANLS-BPP clearly outperforms ANLS-AS for large K's. Because both of the methods solve each NLS subproblem exactly, solutions after each iteration from the two methods are the same up to numerical rounding errors. Hence, it suffices to compare the amount of time spent at each iteration. In Figure 5.3, we showed average execution time of each iteration of the two methods. It can be seen that the time required for ANLS-BPP is significantly shorter than the time required for ANLS-AS in early iterations, and their time requirements became gradually closer to each other. The types of NLS problems in which ANLS-BPP accelerates ANLS-AS is that there

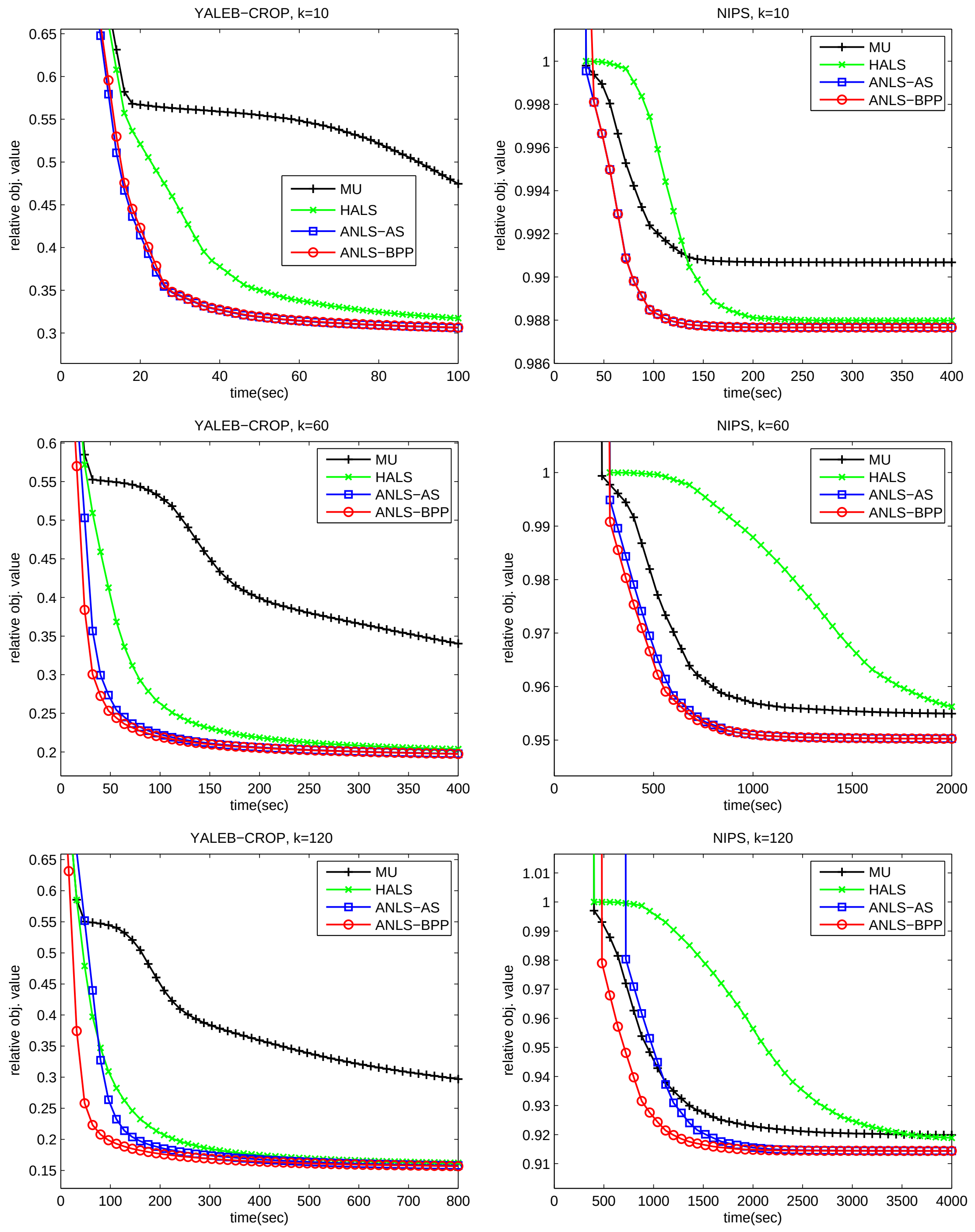


Figure 5.2: Relative objective value $(\|\boldsymbol{\mathcal{A}} - [\![\mathbf{H}^{(1)}, \cdots, \mathbf{H}^{(N)}]\!]\|_F / \|\boldsymbol{\mathcal{A}}\|_F)$ vs. execution time on the YaleB and NIPS data sets. Average results of 5 different random initializations are shown. Left: NIPS data set, right: YaleB data set, top row: $K = 10$, middle row: $K = 60$, and bottom row: $K = 120$.

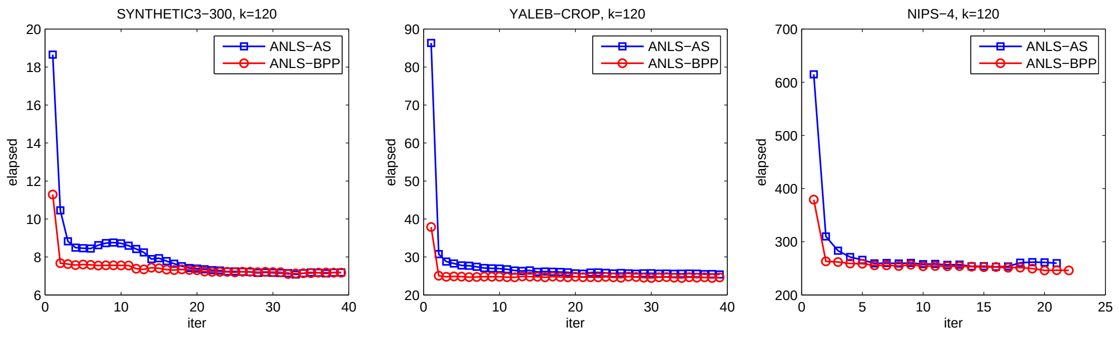


Figure 5.3: Execution time of each iteration of the active-set (ANLS-AS) and the block principal pivoting method (ANLS-BPP) for $K = 120$ cases of each data set. Average results of 5 different random initializations are shown. Left: synthetic data set, center: YaleB data set, right: NIPS data set.

is a much difference in the zero and nonzero pattern between the initial value and the final solution of the NLS problem. As iteration goes on, factors $\{\mathbf{H}^{(1)}, \cdots, \mathbf{H}^{(N)}\}$ do not change much from one iteration to the next; hence there are little difference between the computational costs of the two methods.

5.6 Discussion

We have introduced an efficient algorithm for nonnegative CP (NCP). The new method is based on the block principal pivoting method for the nonnegativity-constrained least squares (NLS) problems. The block principal pivoting method accelerates the classical active-set method by allowing exchanges of multiple variables per iteration. The NLS problems from NCP computation share the same long-and-thin and flat-and-wide structures with the NMF case in Chapter 4, and the acceleration of the block principal pivoting method for the multiple right-hand sides is effective to NCP as well. Computational comparisons showed the state-of-the-art performance of the proposed method for NCP.

A drawback of an NCP algorithm based on the active-set or the block principal pivoting method is that the methods assume that the Khatri-Rao product in Eq. (5.7) is of full column rank for all $n \in \{1, \cdots, N\}$ throughout iterations. To alleviate this concern, as noted in Section 5.4, Frobenius norm-based regularization can be used to avoid rank-deficient cases. In practice, the algorithms performed well in our experiments without the regularization.

CHAPTER VI

FAST ACTIVE-SET-LIKE METHODS FOR L_1-REGULARIZED LINEAR REGRESSION

L_1-regularized linear regression, also known as the Lasso [96], has been a highly successful method for various applications. By constraining the l_1-norm of the coefficient vector, this method simultaneously avoids over-fitting to training data and achieves sparsity in obtained coefficients. The sparsity has two important benefits; it improves interpretation by explicitly showing the relationship between the response and the features [96], and it also allows computationally efficient models because only a small number of coefficients remain nonzero. Researchers used l_1-regularization for many other learning problems including logistic regression [67], graphical model selection [6, 38], principal component analysis [106], and sparse coding [66, 76]. An efficient algorithm for l_1-regularized linear regression is important not only in its own right but also for those extended sparse learning problems.

Since the objective function of l_1-regularized linear regression is not differentiable, development of an efficient algorithm is not trivial. Among several approaches, one of the most influential has been the *least angle regression* (LARS) by Efron et al. [32]. In LARS, features are sequentially selected so that they remain equiangular, exploiting the fact that coefficient paths are piecewise linear with respect to the regularization parameter. Despite its ability to discover the full regularization paths, however, LARS is designed to select one feature at a time, and it can become very slow when applied to large problems. Lee et al. [66] proposed the *feature-sign search* algorithm as a part of their study on sparse coding. This algorithm is a relaxed form of LARS so that a particular solution can be efficiently found by not following the exact coefficient path. The algorithm follows the structure of active-set methods [82] and shares the same difficulty of LARS for large problems. Several iterative methods have been introduced to overcome the scalability issue. Recent developments in signal processing literature include an interior point method [59] and a gradient projection method [34]. Such methods have a particular advantage for their intended purpose, which is signal reconstruction, because they can handle very large problems. Another promising algorithm using a coordinate descent method [37] was recently introduced.

In this chapter, we present new active-set-like methods for l_1-regularized linear regression. By an active-set-like method, we mean an algorithm that directly searches for the sets of active and passive variables and computes the solution only up to numerical rounding errors. By showing a relationship between l_1-regularized linear regression and the linear complementarity problem with bounds (BLCP), we present an efficient method, called *block principal pivoting*, which overcomes the difficulty of the LARS and feature-sign search methods. Although the block principal pivoting method was proposed for linear complementarity problems [53], its substantial benefit to l_1-regularized linear regression has not been studied previously and is an important contribution of this chapter. We further propose an improvement of this method, discuss its characteristics, and also show a connection to the structure learning of Gaussian graphical models. Experimental comparisons on both synthetic and real data sets show the proposed method significantly outperforms several existing ones.

6.1 Active-set-like Methods

6.1.1 Formulations and optimality conditions

Suppose data are given as $(\mathbf{x}^{(n)}, y^{(n)})_{n=1}^{N}$ where $y^{(n)} \in \mathbb{R}$ is the response of $\mathbf{x}^{(n)} \in \mathbb{R}^P$. We assume that $y^{(n)}$'s are centered so that $\sum_{n=1}^{N} y^{(n)} = 0$. The coefficients $\boldsymbol{\beta} \in \mathbb{R}^P$ are to be found by solving a minimization problem,

$$\min_{\boldsymbol{\beta}\in\mathbb{R}^P} \mathcal{L}(\boldsymbol{\beta}, \lambda) = \frac{1}{2} \left\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\right\|_2^2 + \lambda \left\|\boldsymbol{\beta}\right\|_1 , \tag{6.1}$$

where $\mathbf{X} = \left(\mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(N)}\right)^T$, $\mathbf{y} = \left(y^{(1)}, \cdots, y^{(N)}\right)^T$, and $\lambda \geq 0$ is a parameter. Whereas we are concerned in Eq. (6.1) in this chapter, an alternative formulation in which l_1-norm is used for constraints is possible [96]. The Lagrangian dual of Eq. (6.1) can be written as

$$\min_{\mathbf{r} \in \mathbb{R}^N} \mathcal{L}(\mathbf{r}, \lambda) = \frac{1}{2}\mathbf{r}^T\mathbf{r} - y^T\mathbf{r} \text{ s.t. } \left\|\mathbf{X}^T\mathbf{r}\right\|_\infty \leq \lambda, \tag{6.2}$$

using the derivation given in [59]. Eq. (6.2) is easier to handle because the objective function is smooth and constraints are simple.

Our discussion for the derivation of new methods starts from writing down the optimality conditions for the dual problem in Eq. (6.2). Let $\mathbf{r}^*$ be the solution of Eq. (6.2), and let $\boldsymbol{\beta}_+^*, \boldsymbol{\beta}_-^* \in \mathbb{R}^P$ be corresponding Lagrange multipliers for the two inequality constraints, $\mathbf{X}^T\mathbf{r} \leq \lambda$ and $\mathbf{X}^T\mathbf{r} \geq -\lambda$, respectively. Defining $\boldsymbol{\beta}^* = \boldsymbol{\beta}_+^* - \boldsymbol{\beta}_-^*$, the Karush-Kuhn-Tucker (KKT) conditions for Eq. (6.2) can be written as

$$\mathbf{d}^* = \mathbf{X}^T\mathbf{r}^* \quad = \quad \mathbf{X}^T\mathbf{y} - \mathbf{X}^T\mathbf{X}\boldsymbol{\beta}^*, \tag{6.3a}$$

$$-\lambda \leq \ \mathbf{d}^* \ \leq \lambda, \tag{6.3b}$$

$$-\lambda < d_p^* < \lambda \ \Rightarrow \ \beta_p^* = 0 \text{ for } \forall p \in \{1, \cdots, P\}, \tag{6.3c}$$

$$d_p^* = \lambda \ \Rightarrow \ \beta_p^* \geq 0 \text{ for } \forall p \in \{1, \cdots, P\}, \tag{6.3d}$$

$$d_p^* = -\lambda \ \Rightarrow \ \beta_p^* \leq 0 \text{ for } \forall p \in \{1, \cdots, P\}. \tag{6.3e}$$

Note that because the problem in Eq. (6.2) is convex, a solution satisfying Eqs. (6.3) is optimal for Eq. (6.2).

6.1.2 Active-set methods and limitation

The key idea of active-set-like methods for Eqs. (6.1) and (6.2) is the following. Let $\mathcal{E}_+^*, \mathcal{E}_-^* \subseteq \{1, \cdots, P\}$ denote the active constraints of the solution r^* in Eq. (6.2); that is, $\mathcal{E}_+^* = \left\{p | \left(\mathbf{X}^T\mathbf{r}^*\right)_p = \lambda\right\}$ and $\mathcal{E}_-^* = \left\{p | \left(\mathbf{X}^T\mathbf{r}^*\right)_p = -\lambda\right\}$. If we know $\mathcal{E}_+^*$ and $\mathcal{E}_-^*$ in advance, then the solution $\boldsymbol{\beta}^*$ can be easily computed by using Eq. (6.3c) and solving a normal equation for Eq. (6.3a). The goal of active-set methods is to find the sets $\left(\mathcal{E}_+^*, \mathcal{E}_-^*\right)$ in some systematic way.

A standard example of active-set-like methods is the *active-set* method [82], which is a well known scheme generally applicable to quadratic programming problems. The active-set method begins with an initial feasible solution, for which a zero vector is usually used. The method maintains *working sets* $(\mathcal{E}_+, \mathcal{E}_-)$ as candidates for $\left(\mathcal{E}_+^*, \mathcal{E}_-^*\right)$ and iteratively exchanges a variable among $(\mathcal{E}_+, \mathcal{E}_-, \{1, \cdots, P\} - (\mathcal{E}_+ \cup \mathcal{E}_-))$ in such a way that the value of the objective function monotonically decreases. Due to the monotonic-decreasing property, the active-set method is guaranteed to finish in a finite number of steps.

In the case of l_1-regularized linear regression, the feature-sign algorithm in [66] exactly follows the structure of the active-set method. In fact, applying the standard active-set method to the dual problem in Eq. (6.2), the feature-sign algorithm can be directly derived although the authors did not use this derivation. LARS is a modified active-set algorithm in which the modification is elegantly designed so that the full coefficient paths can be computed.

Despite the finite termination property, the active-set methods have a major limitation: Because typically only one variable is exchanged among the working sets per iteration, they can become very slow for large problems. The number of iterations severely depends on the number of nonzero elements of the optimal solution. For large problems, an algorithm in which the number of iterations does not depend upon the problem size is needed. We now describe such an algorithm.

6.2 BLCP and Block Principal Pivoting Methods

The optimality conditions in Eqs. (6.3) are indeed equivalent to the linear complementarity problem with bounds (BLCP) [53]. BLCP is a generalized class of problems from linear complementarity problems (LCP) [80], and LCP and BLCP frequently arise in quadratic programming. Among several approaches for BLCP, an efficient algorithm proposed in [53] implements an intuitive way of speeding up the process of finding $\left(\mathcal{E}_+^*, \mathcal{E}_-^*\right)$. In this section, we summarize the algorithm and also propose its improvement.

6.2.1 Block principal pivoting

Suppose the index set $\{1, \cdots, P\}$ is partitioned into three disjoint subsets $(\mathcal{H}, \mathcal{F}_+, \mathcal{F}_-)$; i.e., $\mathcal{F} = \mathcal{F}_+ \cup \mathcal{F}_- = \{1, \cdots, P\} - \mathcal{H}$ and $\mathcal{F}_+ \cap \mathcal{F}_- = \emptyset$. The sets $(\mathcal{H}, \mathcal{F}_+, \mathcal{F}_-)$ will be our working sets. We eventually want to make $(\mathcal{F}_+, \mathcal{F}_-)$ identical to $\left(\mathcal{E}_+^*, \mathcal{E}_-^*\right)$ up to exchanges of degenerate variables.[1] Let $\boldsymbol{\beta}_{\mathcal{H}}$ and $\mathbf{d}_{\mathcal{H}}$ denote the subsets of vectors $\boldsymbol{\beta}$ and $\mathbf{d}$ corresponding to index set $\mathcal{H}$. Likewise, let $\mathbf{X}_{\mathcal{H}}$ denote a submatrix of $\mathbf{X}$ that consists only of the column vectors whose indices belong to $\mathcal{H}$. The subsets and submatrices for $\mathcal{F}_+, \mathcal{F}_-$, and $\mathcal{F}$ are similarly defined.

We start with an initial setting for $(\mathcal{H}, \mathcal{F}_+, \mathcal{F}_-)$, where $\mathcal{H} = \{1, \cdots, P\}, \mathcal{F}_+ = \mathcal{F}_- = \emptyset$ is usually used. For the subsets, we assume

$$\boldsymbol{\beta}_{\mathcal{H}} = 0,\ \mathbf{d}_{\mathcal{F}_+} = \lambda,\ \mathbf{d}_{\mathcal{F}_-} = -\lambda, \tag{6.4}$$

and compute the remaining elements of $\boldsymbol{\beta}$ and $\mathbf{d}$ using Eqs. (6.3) as follows:

$$\mathbf{d}_{\mathcal{F}} = \left(\mathbf{X}^T\mathbf{y}\right)_{\mathcal{F}} - \mathbf{X}_{\mathcal{F}}^T\mathbf{X}_{\mathcal{F}}\boldsymbol{\beta}_{\mathcal{F}}, \tag{6.5a}$$

$$\mathbf{d}_{\mathcal{H}} = \left(\mathbf{X}^T\mathbf{y}\right)_{\mathcal{H}} - \mathbf{X}_{\mathcal{H}}^T\mathbf{X}_{\mathcal{F}}\boldsymbol{\beta}_{\mathcal{F}}. \tag{6.5b}$$

Since $\mathbf{d}_{\mathcal{F}}$ is fixed by the assumptions in Eq. (6.4), one can first solve for $\boldsymbol{\beta}_{\mathcal{F}}$ in Eq. (6.5a) and substitute the result into Eq. (6.5b) to obtain $\mathbf{d}_{\mathcal{H}}$.

Then, we check if the obtained values are optimal using the following conditions:

$$-\lambda \le \mathbf{d}_{\mathcal{H}} \le \lambda,\ \boldsymbol{\beta}_{\mathcal{F}_+} \ge 0,\ \boldsymbol{\beta}_{\mathcal{F}_-} \le 0. \tag{6.6}$$

If $\boldsymbol{\beta}$ and $\mathbf{d}$ satisfy Eqs. (6.4)-(6.6), then they satisfy the optimality conditions in Eqs. (6.3). If $\boldsymbol{\beta}$ and $\mathbf{d}$ satisfy all these conditions, we call the pair $(\boldsymbol{\beta}, \mathbf{d})$ *feasible*;

[1] We say β_p is degenerate if $\beta_p = 0$ and $(d_p = \lambda$ or $-\lambda)$ are satisfied at the same time.

otherwise, it is called *infeasible.* If a feasible pair $(\boldsymbol{\beta}, \mathbf{d})$ is found, then it means that the current working sets $(\mathcal{F}_+, \mathcal{F}_-)$ are the same with $\left(\mathcal{E}_+^*, \mathcal{E}_-^*\right)$, and therefore the algorithm terminates with the solution $\boldsymbol{\beta}$. Otherwise, we change the sets $(\mathcal{H}, \mathcal{F}_+, \mathcal{F}_-)$ and try the process again.

The issue is how we update the sets $(\mathcal{H}, \mathcal{F}_+, \mathcal{F}_-)$. To this end, we define *infeasible variables* to be the ones that violate at least one condition in Eq. (6.6), which consist of the following four cases:

$$\mathcal{V} = \cup_{k=1}^{4} \mathcal{J}_k, \tag{6.7a}$$

$$\mathcal{J}_1 = \{p \in \mathcal{H} : d_p > \lambda\}, \tag{6.7b}$$

$$\mathcal{J}_2 = \{p \in \mathcal{H} : d_p < -\lambda\}, \tag{6.7c}$$

$$\mathcal{J}_3 = \{p \in \mathcal{F}_+ : \beta_p < 0\}, \tag{6.7d}$$

$$\mathcal{J}_4 = \{p \in \mathcal{F}_- : \beta_p > 0\}. \tag{6.7e}$$

The set $\mathcal{V}$ contains all infeasible variables. Now choose a subset $\hat{\mathcal{V}} \subseteq \mathcal{V}$ and let

$$\hat{\mathcal{J}}_1 = \hat{\mathcal{V}} \cap \mathcal{J}_1, \hat{\mathcal{J}}_2 = \hat{\mathcal{V}} \cap \mathcal{J}_2, \hat{\mathcal{J}}_3 = \hat{\mathcal{V}} \cap \mathcal{J}_3, \hat{\mathcal{J}}_4 = \hat{\mathcal{V}} \cap \mathcal{J}_4.$$

Then, the update rule is given as

$$\mathcal{H} \leftarrow (\mathcal{H} - (\hat{\mathcal{J}}_1 \cup \hat{\mathcal{J}}_2)) \cup (\hat{\mathcal{J}}_3 \cup \hat{\mathcal{J}}_4), \tag{6.8a}$$

$$\mathcal{F}_+ \leftarrow (\mathcal{F}_+ - \hat{\mathcal{J}}_3) \cup \hat{\mathcal{J}}_1, \tag{6.8b}$$

$$\mathcal{F}_- \leftarrow (\mathcal{F}_- - \hat{\mathcal{J}}_4) \cup \hat{\mathcal{J}}_2. \tag{6.8c}$$

The size $|\hat{\mathcal{V}}|$ represents how many variables are exchanged per iteration, and the choice of $\hat{\mathcal{V}}$ characterizes the algorithm. If $|\hat{\mathcal{V}}| > 1$, then the algorithm is called a *block* principal pivoting algorithm. If $|\hat{\mathcal{V}}| = 1$, then the algorithm is called a *single* principal pivoting algorithm. The active-set method can be understood as an instance of single principal pivoting algorithms. After updating by Eqs. (6.8), the algorithm repeats the entire procedure until the number of infeasible variables (i.e., $|\mathcal{V}|$) becomes zero.

In order to speed up the procedure, $\hat{\mathcal{V}} = \mathcal{V}$ is used, which we call the *full exchange rule*. This exchange rule considerably improves the search procedure by reducing number of iterations. Although this property is desirable, however, using only the full exchange rule does not guarantee finite termination, and we need more refinement.

6.2.2 Finite termination

In active-set methods, the variable to exchange is carefully selected to reduce the objective function. However, the full exchange rule does not have this property and may lead to a cycle although it occurs rarely. To fix this problem, a backup exchange rule is used to guarantee termination in a finite number of steps.

The backup rule is to exchange the infeasible variable with the largest index:

$$\hat{\mathcal{V}} = \{i : i = \max\{j : j \in \mathcal{V}\}\}. \tag{6.9}$$

In this case, the set $\hat{\mathcal{V}}$ contains only one variable, so it is a single principal pivoting rule. This simple rule guarantees a finite termination: Assuming that matrix $X^T X$ has full rank, the single principal pivoting rule in Eq. (6.9) returns the solution for Eqs. (6.3) in a finite number of steps [52].

Combining the full exchange rule and the backup rule, the *block principal pivoting algorithm* is summarized in Algorithm 6.1. Because the backup rule is slower than the full exchange rule, it is used only if the full exchange rule does not work well. Variable t is used to control the number of infeasible variables, and variable k is used as a buffer on the number of the full exchange rules that may be tried. When the full exchange rule fails to decrease the number of infeasible variables within K_{max} iterations, the backup rule is used until it reduces the number of infeasible variables under the lowest value achieved so far, which is stored in t. This has to occur in a finite number of steps because the backup rule has a finite termination property. As soon as the backup rule achieves a new lowest number of infeasible variables, then we return to the full exchange rule. Since the number of infeasible variables is

Algorithm 6.1 Block principal pivoting algorithm for the l_1-regularized liner regression

Input: $\mathbf{X} \in \mathbb{R}^{N \times P}$, $\mathbf{y} \in \mathbb{R}^N$, $\lambda \in \mathbb{R}$
Output: $\boldsymbol{\beta}$

```
1: Initialize $\mathcal{H} = \{1, \cdots, P\}$, $\mathcal{F}_+ = \mathcal{F}_- = \emptyset$, $\boldsymbol{\beta} = 0$, $\mathbf{d} = -\mathbf{X}^T\mathbf{y}$, $k = K_{max}$, $t = P+1$
2: Set Eq. (6.4) and compute $(\boldsymbol{\beta}_{\mathcal{F}},\mathbf{d}_{\mathcal{H}})$ by Eqs. (6.5).
3: while $(\boldsymbol{\beta}, \mathbf{d})$ is infeasible do
4:     Find $\mathcal{V}$ by Eqs. (6.7).
5:     If $|\mathcal{V}| < t$, set $t = |\mathcal{V}|$, $k = K_{max}$, and $\hat{\mathcal{V}} = \mathcal{V}$.
       If $|\mathcal{V}| \geq t$ and $k \geq 1$, set $k = k - 1$, and $\hat{\mathcal{V}} = \mathcal{V}$.
       If $|\mathcal{V}| \geq t$ and $k = 0$, set $\hat{\mathcal{V}}$ by Eq. (6.9).
6:     Update $(\mathcal{H}, \mathcal{F}_+, \mathcal{F}_-)$ by Eqs. (6.8).
7:     Set Eq. (6.4) and compute $(\boldsymbol{\beta}_{\mathcal{F}},d_{\mathcal{H}})$ by Eqs. (6.5).
8: end while
```

systematically reduced, the algorithm terminates in a finite number of steps. The choice of K_{max} has trade-offs, and $K_{max} = 3$ was shown to work well in practice [53].

6.2.3 Reduced block exchange

The full exchange rule is the most greedy version of exchange rules, and it can slow down if features are highly correlated. In this case, the full exchange rule tends to exchange too many variables unnecessarily and spend more iterations until termination. For large problems, this behaviour can be more problematic because unnecessarily increasing the size of $\mathcal{F}$ means that we have to solve large linear equations in Eq. (6.5a) even though the final number of nonzero values, $\left|\mathcal{E}_+^* \cup \mathcal{E}_-^*\right|$, can be small.

To address this difficulty, we designed *reduced block exchange rule* by constraining the maximum increase of $|\mathcal{F}|$ as follows.

1. Allow full exchanges for reducing $|\mathcal{F}|$, i.e., $\hat{\mathcal{J}}_3 = \mathcal{J}_3$ and $\hat{\mathcal{J}}_4 = \mathcal{J}_4$.
2. Limit the increase of $|\mathcal{F}|$ by enforcing that $|\hat{\mathcal{J}}_1 \cup \hat{\mathcal{J}}_2| \leq \alpha p$ where $0 < \alpha < 1$ is a parameter.

The sets $\hat{\mathcal{J}}_1$ and $\hat{\mathcal{J}}_2$ can be naturally determined by sorting $\mathcal{J}_1$ and $\mathcal{J}_2$ based on the absolute values of the violation, i.e., $d_i - \lambda$ for $\mathcal{J}_1$ and $-\lambda - d_i$ for $\mathcal{J}_2$. In our

Table 6.1: Characteristics of active-set-like methods: LARS, feature-sign (FS), and block principal pivoting (BP)

	LARS	FS	BP
variable search	equiangular	active-set	block
regularization path	✓		
monotonic decrease	✓	✓	
finiteness	✓	✓	✓
scalable			✓

experiments, $\alpha = 0.2$ generally produced good results. We modify Algorithm 6.1 by using this exchange rule in the first two cases of Step 5 and call the modified one *reduced block principal pivoting* method.

6.2.4 Summary of characteristics

A summary of active-set-like methods is shown in Table 6.1. LARS computes the entire regularization path by spending more time to compute the equiangular vector. The feature-sign algorithm maintains the monotonic-decreasing property but does not follow the regularization path. The block principal pivoting method maintains the finite termination property and becomes scalable to large problems. While LARS has benefits for computing the full regularization path, for obtaining a particular solution, block principal pivoting methods have advantage. This is the case if the parameter is estimated by prior knowledge or by theoretical analysis, or if l_1-regularized linear regression is used as a subroutine for other sparse learning tasks.

Block principal pivoting methods require that the feature matrix X has full column rank. However, since the other two methods are typically implemented using normal equations, they similarly require that features in active set have full column rank in every iteration. Furthermore, additional l_2-norm regularization such as elastic net [105] can be adopted to alleviate this concern. In return, block principal pivoting methods enable significant speed-up.

In some sparse learning tasks in which l_1-regularized linear regression is used as a subroutine, the full rank assumption is always satisfied. We describe an example in the following section.

6.3 Gaussian Structure Learning

For graphical models such as Markov random fields (MRFs), sparse structures often need to be learned from data. In the Gaussian case, learning structure is equivalent to identifying zero and nonzero elements of the inverse covariance matrix. We briefly describe how the proposed method can be used to accelerate the structure learning of Gaussian graphical models within the framework of [6].

Suppose $(\mathbf{x}^{(n)})_{n=1}^{N}$, $\mathbf{x}^{(n)} \in \mathbb{R}^P$ are observed from multivariate Gaussian distribution, $N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. Our goal is to estimate the inverse covariance matrix $\boldsymbol{\Sigma}^{-1}$ such that a small number of elements in $\boldsymbol{\Sigma}^{-1}$ are nonzero. Following [6], a penalized maximum likelihood formulation can be written as

$$\max_{\mathbf{Z}} \log\det\mathbf{Z} - \operatorname{trace}(\mathbf{Z}\hat{\boldsymbol{\Sigma}}) - \eta \sum_{i,j=1}^{P} |Z_{ij}|\,, \tag{6.10}$$

where $\hat{\boldsymbol{\Sigma}} = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}^{(n)} - \hat{\boldsymbol{\mu}})(\mathbf{x}^{(n)} - \hat{\boldsymbol{\mu}})^T$, $\hat{\boldsymbol{\mu}} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}^{(n)}$, and $\eta > 0$ is a parameter to control the strength of penalty. The dual form for Eq. (6.10) is written as

$$\begin{aligned} &\min_{\mathbf{W}} - \log\det(\hat{\boldsymbol{\Sigma}} + \mathbf{W}) - \mathbf{d} \qquad (6.11)\\ &\text{s.t. } \hat{\boldsymbol{\Sigma}} + \mathbf{W} \succ 0, \ \ -\eta \le W_{ij} \le \eta, \ \forall i,j \in \{1, \cdots, P\} \end{aligned}$$

where $\mathbf{A} \succ 0$ means that $\mathbf{A}$ is positive definite.

Eq. (6.11) can be efficiently solved by block coordinate descent approach. Let $\mathbf{V} = \hat{\boldsymbol{\Sigma}} + \mathbf{W}$, initialize $\mathbf{V}$ with $\hat{\boldsymbol{\Sigma}} + \eta\mathbf{I}$, and assume that $\hat{\boldsymbol{\Sigma}}$ and $\mathbf{V}$ are partitioned as

$$\mathbf{V} = \begin{pmatrix} \mathbf{V}_{11} & \mathbf{v}_{12} \\ \mathbf{v}_{12}^T & v_{22} \end{pmatrix}, \ \hat{\boldsymbol{\Sigma}} = \begin{pmatrix} \mathbf{S}_{11} & \mathbf{s}_{12} \\ \mathbf{s}_{12}^T & s_{22} \end{pmatrix},$$

where $\mathbf{v}_{12}, \mathbf{s}_{12} \in \mathbb{R}^{P-1}$ and $v_{22}, s_{22} \in \mathbb{R}$. Then, we update $\mathbf{v}_{12}$ (and $\mathbf{v}_{12}^T$, of course) while fixing all other elements. Using Schur complement, the update problem can be simplified as

$$\mathbf{v}_{12} = \arg\min_{\mathbf{v}} \mathbf{v}^T \mathbf{W}_{11}^{-1} \mathbf{v} \text{ s.t. } \|\mathbf{v} - \mathbf{s}_{12}\|_\infty \leq \eta. \tag{6.12}$$

One can solve Eq. (6.12) for each column (and corresponding row) of V by permutations. [6] solved Eq. (6.12) using a smooth optimization technique, and [38] used a coordinate descent method for the dual of Eq. (6.12).

We efficiently solve Eq. (6.12) by exploiting its relation to Eq. (6.2). As we did for Eq. (6.2), we can write the KKT optimality conditions for Eq. (6.12) as

$$\begin{aligned}
\mathbf{k} &= \mathbf{s}_{12} - \mathbf{W}_{11}\mathbf{l}, && (6.13a)\\
-\eta \leq \ \mathbf{k} &\leq \eta, && (6.13b)\\
-\eta < k_p < \eta &\Rightarrow l_p = 0 \text{ for } p \in \{1, \cdots, P\}, && (6.13c)\\
k_p = \eta &\Rightarrow l_p \geq 0 \text{ for } p \in \{1, \cdots, P\}, && (6.13d)\\
k_p = -\eta &\Rightarrow l_p \leq 0 \text{ for } p \in \{1, \cdots, P\}. && (6.13e)
\end{aligned}$$

where $\mathbf{v} = \mathbf{s}_{12} - \mathbf{k}$. Hence, the proposed method presented in Section 6.2 can be directly used for Eqs. (6.13). As shown in [6], $\mathbf{W}_{11}$ remains positive definite throughout iterations, satisfying the full rank assumption.

6.4 Experimental Validation

We implemented the proposed methods in MATLAB and compared with several existing ones. We first compared active-set-like methods under various conditions, and then we compared with iterative methods and tested on real data sets. We also provide results for Gaussian structure learning task. All experiments were executed on a 2.66GHz Intel Quad Core processor with Linux OS, with multi-threading option disabled. All results are the average of 10 executions.

Table 6.2: Execution results of the LARS, feature-sign (FS), block principal pivoting (BP), and reduced block principal pivoting (BPR, with $\alpha = 0.2$) methods on data sets with sparse random features (see text). The third column shows the number of nonzero elements in obtained $\boldsymbol{\beta}$ for the corresponding value of λ.

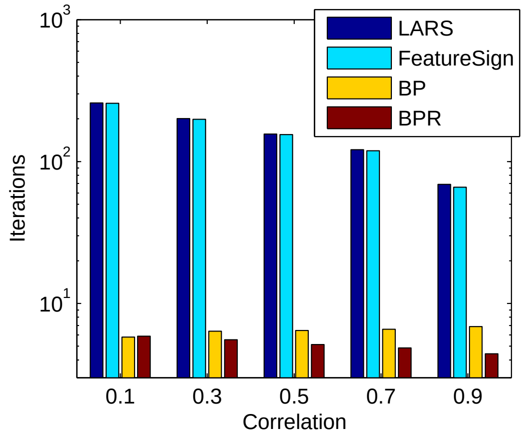
			# of iterations				Time (seconds)			
Problem size	λ	Nonzeros	LARS	FS	BP	BPR	LARS	FS	BP	BPR
2500 × 1000	16	70	71	71	**2**	**2**	1.07	0.367	**0.341**	**0.34**
	9.71	263	264	264	**4**	**4**	4.23	1.53	**0.357**	**0.352**
	5.89	485	486	486	**4**	**5**	12.3	6.49	**0.402**	**0.378**
	3.58	654	655	655	**4**	**8**	23.5	13.0	**0.439**	**0.459**
	2.17	769	772	770	**5**	**8**	41.0	21.2	**0.516**	**0.493**
5000 × 2000	25.9	83	84	84	**2**	**2**	5.46	**2.32**	**2.37**	**2.37**
	15.7	436	437	437	**4**	**4**	24.3	7.79	**2.41**	**2.41**
	9.56	884	887	885	**4**	**5**	91.1	45.0	**2.62**	**2.57**
	5.80	1226	1233	1228	**4**	**8**	222	118	**2.94**	**2.90**
	3.52	1493	1500	1495	**4**	**7**	462	226	**3.31**	**3.05**
10000 × 5000	35.3	269	270	270	**3**	**3**	79.7	29.6	**27.2**	**27.2**
	21.4	1134	1135	1135	**4**	**5**	394	145	**27.8**	**27.7**
	13.0	2128	2129	2129	**5**	**6**	2112	872	**31.2**	**29.6**
	7.89	3027	3030	3028	**5**	**7**	7632	2955	**36.4**	**33.1**
	4.79	3675	3696	3676	**5**	**9**	15969	5954	**42.4**	**40.8**

6.4.1 Active-set-like methods

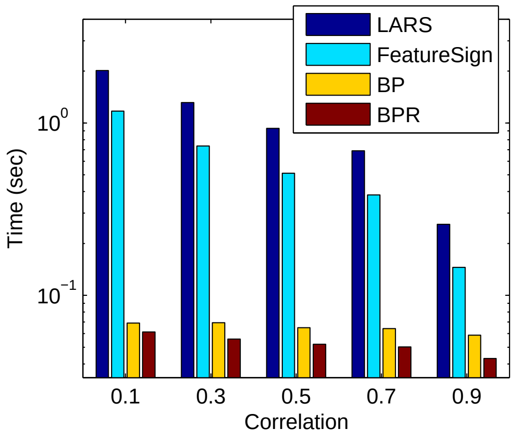
To see the behaviour of active-set-like methods under various conditions, we tested them with synthetic data sets generated by a linear model: $\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$. We tried two types of feature matrix $\mathbf{X}$, and how we generated them is described below. Each elements of $\boldsymbol{\beta}$ was sampled from uniform distribution on $[-1, 1]$. Then, independent Gaussian noise $\boldsymbol{\epsilon}$ was generated and scaled so that the average magnitude of elements in $\boldsymbol{\epsilon}$ is five percent of the average magnitude of elements in $\mathbf{X}\boldsymbol{\beta}$. MATLAB implementations of the feature-sign [66] and the LARS algorithms [95] were used.[2]

We first tested active-set-like methods with sparse random features. We sampled each element of $\mathbf{X}$ from uniform distribution on $[0, 1]$ and randomly selected 70 percent of the elements to make them zero. As shown in Table 6.2, considerable improvements

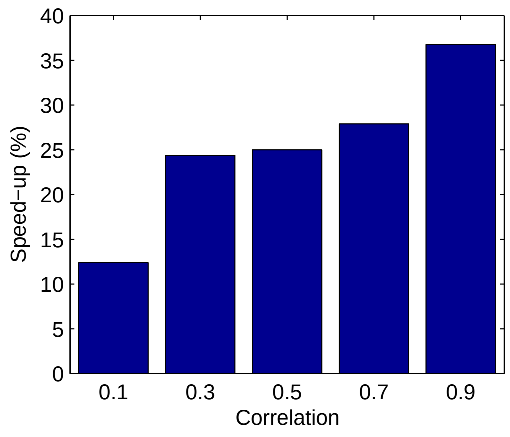
[2]The LARS algorithm was modified to stop at the solution for given parameter λ.



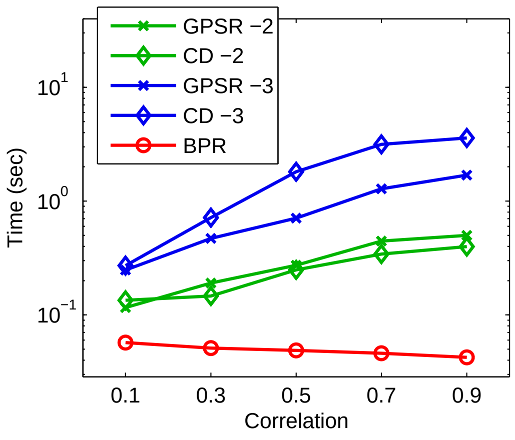
(a)



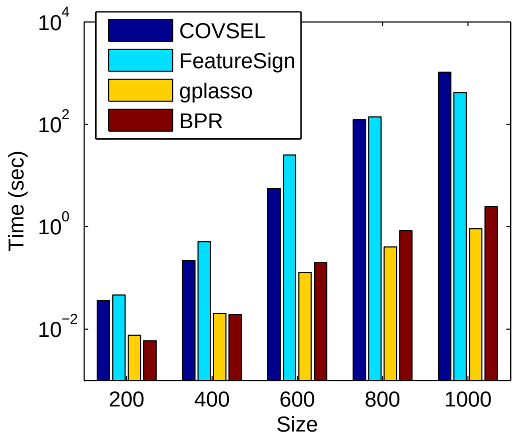
(b)



(c)



(d)



(e)

Figure 6.1: (a,b): Iteration counts and execution time of the LARS, FS, BP, and BPR methods on data sets with correlated features (see text) (c): Speed-up by BPR upon BP (d): Execution time of BPR and other iterative optimization schemes (e): Execution time for Gaussian structure learning for various sizes.

were achieved by the two new methods: the block principal pivoting (BP) and reduced block principal pivoting (BPR). The numbers of iterations required by these methods are small for various problem sizes and values of parameter λ. Accordingly, the execution time of these methods depend less severely on such variations than the LARS and feature-sign methods. One can see that the BP and BPR algorithms are up to 100 (or more) times faster than LARS.

Next, we observed behaviour with correlated features. For various values of ρ, we created $\mathbf{X}$ by generating 1000 samples of multivariate Gaussian distribution with 500 variables where the correlation coefficient of each pair of variables is ρ. We generated ten instances of such data for each ρ, and for each instance, all algorithms were executed for several λ's obtained as follows. We computed the smallest λ that allows $\boldsymbol{\beta} = 0$ as a solution, which is $\lambda_{max} = \left\|\mathbf{X}^T\mathbf{y}\right\|_\infty$, and then we divided the interval $(\lambda_{max}, \lambda_{max} \times 0.01)$ by six in log-scale to get the five values in the middle. Average results over ten data sets and the five parameter values are shown in Figure 6.1-(a,b,c). Initially, we thought that BP can much slow down with highly correlated features; however, both BP and BPR appeared faster than other methods even for the highly correlated case. As correlation increase, a smaller number of features tend to remain nonzero because all the features become more similar. This generally resulted in smaller number of iterations and execution time for the LARS, feature-sign, and BPR methods in high correlation.

BPR showed a clear advantage against BP especially for the highly correlated case. Figure 6.1-(a,b) show that BPR reduced both the number of iterations and the execution time of BP, and Figure 6.1-(c) illustrates the speed-up of BPR against BP. For high correlation, the reduced block exchange rule effectively controls the increase of $|\mathcal{F}|$ and becomes more efficient than the full exchange rule. Hence, we focused on BPR in the following experiments.

Table 6.3: Execution results on data sets from UCI data repository. The results for gradient projection (GPSR) and coordinate descent (CD) methods are given for two tolerance (τ) values. AS stands for active-set-like methods, which include LARS, feature sign (FS), and reduced block principal pivoting (BPR) methods. The number of selected features represent the number of nonzero coefficients.

			Time (sec)							# of selected features				
			AS			10^{-2}		10^{-3}		AS	10^{-2}		10^{-3}	
	size	λ	LARS	FS	BPR	GPSR	CD	GPSR	CD		GPSR	CD	GPSR	CD
Arrhy-	452	0.86	0.046	0.041	**0.014**	0.028	0.015	0.043	0.023	52	54	53	53	52
thmia	$\times$279	0.57	0.073	0.063	**0.022**	0.045	0.026	0.072	0.025	78	83	80	80	80
		0.16	0.224	0.144	**0.023**	0.046	0.033	0.163	0.042	140	174	155	143	140
	6238	0.74	2.61	0.75	**0.41**	4.80	0.693	36.4	0.954	116	137	117	116	116
Isolet	$\times$617	0.11	9.37	3.94	**0.431**	8.93	2.09	63.7	3.57	301	362	308	305	301
		0.017	24.5	13.3	**0.728**	4.49	3.76	74.2	8.73	515	604	532	529	520
Internet	3279	0.53	9.02	2.72	**1.52**	5.32	1.82	17.9	1.96	243	256	189	243	196
Ad	$\times$1558	0.28	17.0	6.14	**1.57**	3.79	1.86	19.3	2.54	408	448	330	424	357
		0.081	76.6	38.5	**1.85**	2.86	3.31	19.0	5.46	862	1075	755	882	771
	6000	0.64	160	44.8	**30.4**	75.6	36.3	249	39.0	465	534	464	466	465
Gisette	$\times$5000	0.34	448	153	**31.2**	73.5	48.5	264	58.1	1070	1330	1121	1083	1057
		0.18	1495	606	**35.0**	58.2	74.7	304	100.0	1829	2702	1964	1877	1825

6.4.2 Comparison with iterative methods

To see relative performance against iterative optimization methods, we compared BPR with two recently proposed methods: the gradient projection method by [34] (GPSR) and the coordinate descent method by [37] (CD).[3]

To compare these algorithms under the same condition, we stopped both algorithms in the following way. Suppose the solution in the k^{th} iteration is $\boldsymbol{\beta}^{(k)}$, let $\mathbf{d}^{(k)} = \mathbf{X}^T\mathbf{y} - \mathbf{X}^T\boldsymbol{X}\boldsymbol{\beta}^{(k)}$, and define $\mathbf{g}^{(k)}$ by

$$g_i^{(k)} = \begin{cases} -d_i^{(k)} + \lambda & \text{if } (\beta_i^{(k)} > 0) \text{ or } (\beta_i^{(k)} = 0 \text{ and } d_i > \lambda) \\ -d_i^{(k)} - \lambda & \text{if } (\beta_i^{(k)} < 0) \text{ or } (\beta_i^{(k)} = 0 \text{ and } d_i < -\lambda) \\ 0 & \text{if } \beta_i^{(k)} = 0 \text{ and } -\lambda \leq d_i \leq \lambda \end{cases}$$

[3]The code of GPSR was obtained from the authors. For CD, although the authors provide an R package, we reimplemented in MATLAB/C to compare in the same environment and stopping criterion. Our implementation of CD was carefully optimized by using C (i.e., MEX files) for iterative operations, which can be slow in MATLAB script. Note that other implementations used in our comparisons, including the implementations of BP and BPR, are purely in MATLAB only.

where $\left(\mathbf{g}^{(k)}\right)^T = \left(g_1^{(k)}, \cdots, g_P^{(k)}\right) \in \mathbb{R}^P$. Then, with $\Delta^k = (\left\|\mathbf{g}^k\right\|_2 / \#$ of nonzeros in $\mathbf{g}^k)$, an algorithm was stopped if $\Delta^k \leq \tau \Delta^0$ where Δ^0 is the value using initial values and τ is a chosen tolerance. This criterion can be obtained by using the subdifferential of l_1-norm along with the criterion in [72]. We observed that at most $\tau = 10^{-2}$ or 10^{-3} is recommended because values larger than these produced very inaccurate solutions in our repeated trials (See also Section 6.4.3).

Execution results of these algorithms on the correlated data set mentioned in the previous subsection are presented in Figure 6.1-(d).[4] The results indicate that BPR is highly competitive against these state-of-the-art iterative methods, even for a loose tolerance. Interestingly, BPR and the two iterative methods showed reverse trends with respect to the correlation coefficients: the two iterative methods became slower for highly correlated data unlike BPR.

It is worth mentioning that all the active-set-like methods, including LARS, feature-sign, and BPR, return an exact solution. In contrast, iterative methods do so only if very small tolerance is applied. The BPR method enjoys the property that it returns an exact solution without loosing scalability.

6.4.3 UCI data sets

In Table 6.3, the execution results on data sets from UCI repository[5] are shown. Linear regression was considered for each data set with elastic net penalty with small l_2-norm regularization (parameter $10^{-4} \times \lambda$ when λ is given for l_1-norm). Being consistently faster than the LARS and feature-sign methods, BPR was competitive against iterative methods. For $\tau = 10^{-2}$, solutions from iterative methods were inaccurate as can be seen from the fact that the number of selected features are different from that of exact solutions obtained by active-set-like methods. When

[4]We report the best results from the two methods: the Barzilai-Borwein version for GPSR and the covariance updating with active-set arrangement for CD.

[5]http://archive.ics.uci.edu/ml/

Table 6.4: Execution time (sec) of Gaussian structure learning on Rosetta Compendium gene expression data set (6316 features and 300 samples)

COVSEL	feature-sign	BP	gplasso
53.55	225.24	4.82	4.07

$\tau = 10^{-3}$ was applied for higher accuracy, the computation time increased. While providing exact solutions, BPR was very fast among all the methods tested.

6.4.4 Gaussian structure learning

Figure 6.1-(e) and Table 6.4 show execution results for the structure learning of Gaussian graphical models. Parameter η was selected using the formulation in [6], and time was measured until the duality gap, described in the same paper, of 10^{-1} is achieved. For the results shown in Figure 6.1-(e), with various dimensions P, $\frac{1}{3}P$ samples of multivariate Gaussian distribution with sparse inverse covariance matrix were generated and used. BPR appeared significantly faster than the COVSEL [6] and feature-sign method, where the feature-sign method was used for Eq. (6.12). BPR showed comparable performance with graphical Lasso (gplasso) [38]. The same trend can be observed from Table 6.4, which shows execution times on the gene expression data set used in [6].

6.5 Discussion

We introduced new active-set-like algorithms for l_1-regularized linear regression and demonstrated their computational benefits through experimental comparisons. The proposed method achieved significant speed-up maintaining the accuracy of solutions.

The block principal pivoting algorithm described in this chapter can be compared with the block principal pivoting method in Chapter 4 as follows. The optimality conditions for the NLS problems appear as a linear complementarity problem (LCP) as in Eq. (4.3), which is simpler than BLCP in Eq. (6.3). A notable difference is that in the NLS problem, variables are exchanged between two groups (zero or positive) due to nonnegativity constraints, whereas in l_1-regularized linear regression, variables can take any sign, and thus they are exchanged among three groups (negative, zero, or positive). Due to the difference, the exchange rules of the proposed method are more complicated, and the finite-termination proof becomes more difficult [52]. One might observe that l_1-regularized linear regression can be reformulated as the NLS problems [96, 34], and the algorithm in Chapter 4 might be used. However, this approach is inefficient because the reformulation doubles the variable size. Furthermore, after reformulation, the matrix $\mathbf{B}$ in Eq. (4.3) becomes always rank-deficient, making the application of the algorithm in the algorithm in Chapter 4 infeasible.

CHAPTER VII

NONNEGATIVE RANK-DEFICIENT LEAST SQUARES

In the efficient algorithms for NMF and NCP proposed in Chapters 4 and 5, a fast algorithm for the the nonnegativity-constrained least squares (NLS) problems plays a key role. The block principal pivoting method, discussed in those chapters, has similarities with the classical active-set method [63] but overcomes its limitation by allowing exchanges of multiple variables between working sets per iteration. By reducing the number of iterations required until termination, the block principal pivoting method significantly accelerates the active-set method when the number of unknowns are large. There are, however, other trade-offs between the two methods. Although the block principal pivoting method is typically faster than the active-set method, it requires that the matrix involved in the NLS problem is of full column rank. On the other hand, in this chapter, we show that the active-set method can deal with the rank-deficient NLS problems. We analyze the active-set method in detail and show that the columns corresponding to passive variables remain linearly independent if the method is appropriately initialized. Analysis provided in this chapter enables a deeper understanding of the behaviour of the active-set method for the NLS problems.

Consider a NLS problem as follows.

$$\min_{\mathbf{x}\geq 0} J(\mathbf{x}) = \frac{1}{2}\left\|\mathbf{B}\mathbf{x}-\mathbf{c}\right\|_2^2 \tag{7.1}$$

where $\mathbf{B}\in\mathbb{R}^{M\times N}$, $\mathbf{c}\in\mathbb{R}^{M}$. In Section 7.1, we will first introduce the active-set method for solving Eq. (7.1) due to Lawson and Hanson [63] and provide additional explanations for its property. In Section 7.2, we provide a proof of a statement regarding the behaviour of the active set method in handling the rank-deficient case.

Algorithm 7.1 The active-set method [63] for the NLS problems

```
Input: $\mathbf{B} \in \mathbb{R}^{M\times N}, \mathbf{c} \in \mathbb{R}^M$
Output: $\arg\min_{\mathbf{x}\geq 0} \|\mathbf{B}\mathbf{x} - \mathbf{c}\|_2^2$
1:  $\mathbf{x} \leftarrow \mathbf{0}$, $\mathcal{E} \leftarrow \{1, 2, \cdots, N\}$, $\mathcal{S} \leftarrow \emptyset$, $\mathbf{w} \leftarrow \mathbf{B}^T(\mathbf{c} - \mathbf{B}\mathbf{x})$.
2:  while $\mathcal{E} \neq \emptyset$ and $\exists j \in \mathcal{E}$ such that $w_j > 0$ do
3:      Let $t \in \mathcal{E}$ be such that $w_t = \max\{w_j : j \in \mathcal{E}\}$
4:      $\mathcal{E} \leftarrow \mathcal{E} - \{t\}$ and $\mathcal{S} \leftarrow \mathcal{S} \cup \{t\}$.
5:      Let $\mathbf{B}_\mathcal{S} \in \mathbb{R}^{M\times|\mathcal{S}|}$ be the submatrix of $\mathbf{B}$ containing only the columns indexed
        by $\mathcal{S}$. Consider a vector $\mathbf{z} \in \mathbb{R}^N$, whose elements are found by

            $$\mathbf{z}_\mathcal{S} \leftarrow \arg\min_{\mathbf{y}\in\mathbb{R}^{|\mathcal{S}|}} \|\mathbf{B}_\mathcal{S}\mathbf{y} - \mathbf{c}\|_2^2$$

        and $\mathbf{z}_\mathcal{E} \leftarrow \mathbf{0}$, where $\mathbf{z}_\mathcal{S}$ and $\mathbf{z}_\mathcal{E}$ represent the subvectors of $\mathbf{z}$ indexed by $\mathcal{S}$ and
        $\mathcal{E}$, respectively.
6:      while $z_j \leq 0$ for any $j \in \mathcal{E}$ do
7:          Let $q \in \mathcal{S}$ be such that $\frac{x_q}{x_q - z_q} = \min\left\{\frac{x_j}{x_j - z_j} : z_j \leq 0, j \in \mathcal{S}\right\}$, and let $\alpha \leftarrow \frac{x_q}{x_q - z_q}$.
8:          $\mathbf{x} \leftarrow \mathbf{x} + \alpha(\mathbf{z} - \mathbf{x})$.
9:          Let $\mathcal{H} = \{j : j \in \mathcal{S}$ and $x_j = 0\}$ and update $\mathcal{E} \leftarrow \mathcal{E} \cup \mathcal{H}$ and $\mathcal{S} \leftarrow \mathcal{S} - \mathcal{H}$.
10:         $\mathbf{z}_\mathcal{S} \leftarrow \arg\min_{\mathbf{y}\in\mathbb{R}^{|\mathcal{S}|}} \|\mathbf{B}_\mathcal{S}\mathbf{y} - \mathbf{c}\|$ and $\mathbf{z}_\mathcal{E} \leftarrow \mathbf{0}$.
11:     end while
12:     $\mathbf{x} \leftarrow \mathbf{z}$.
13:     $\mathbf{w} \leftarrow \mathbf{B}^T(\mathbf{c} - \mathbf{B}\mathbf{x})$.
14: end while
```

7.1 *Active-set method for NLS problems*

The active-set method for Eq. (7.1) is presented in Algorithm 7.1. This algorithm is a standard method for the NLS problems, and its implementation is included in MATLAB as the function *lsqnonneg*. A key idea underlying Algorithm 7.1 is that, if the zero and nonzero variables of the optimal solution of Eq. (7.1) is known in advance, the optimal solution can be easily computed. Let $\mathbf{x}^*$ be a minimizer of Eq. (7.1), then the index set

$$\mathcal{E}^* = \{j | x_j^* = 0, j = 1, 2, \cdots, N\} \tag{7.2}$$

is called the *active set* since the nonnegativity constraints are actively satisfied in those indices. Similarly, the index set

$$\mathcal{S}^* = \{j | x_j^* \neq 0, j = 1, 2, \cdots, N\} \tag{7.3}$$

is called the *passive set*. In Algorithm 7.1, $(\mathcal{E}, \mathcal{S})$ is maintained as candidates for $(\mathcal{E}^*, \mathcal{S}^*)$, and indices are exchanged between $\mathcal{E}$ and $\mathcal{S}$ until $(\mathcal{E}^*, \mathcal{S}^*)$ is identified. Typically, a zero vector is used as an initial value, i.e., $\mathcal{S} = \emptyset$.

Algorithm 7.1 is composed of two nested loops: the inner loop consisting of from Step 7 to Step 10 and the outer loop consisting of from Step 3 to Step 13. In the inner loop, the solution of an unconstrained least squares problem with respect to the current passive set $\mathcal{S}$ is computed in Step 10. If the solution is nonnegative, the inner loop is terminated at Step 6; otherwise, a step length is chosen so that at least one passive variable becomes active (Step 7), and the loop is repeated. In the outer loop, a check is made to determine if the current solution obtained from the inner loop is the optimal solution (Step 2). If it is not the optimal solution, then one index is chosen from the active set and moved to the passive set (Steps 3 and 4).

Assuming that $\mathbf{B}$ is of full column rank, the correctness of Algorithm 7.1 is shown in [63]. In order to formally describe this property, let us see the following lemma whose proof is given in Chapter 23 of [63].

Lemma 7.1. *Let $\mathbf{G} \in \mathbb{R}^{M \times N}$ be a matrix of rank N and let $\mathbf{h} \in \mathbb{R}^N$ satisfying*

$$\mathbf{G}^T \mathbf{h} = \begin{pmatrix} \mathbf{0}_{N-1} \\ \lambda \end{pmatrix}$$

with $\lambda > 0$. If $\mathbf{y}^* = \arg\min_{\mathbf{y} \in \mathbb{R}^\mathbf{N}} \|\mathbf{G}\mathbf{y} - \mathbf{h}\|_2^2$, then $y_N^* > 0$.

Lemma 7.1 is essential in proving that Algorithm 7.1 is well defined, as presented in the following theorem.

Theorem 7.2. *In Algorithm 7.1, for t chosen at Step 3, vector $\mathbf{z}$ obtained in Step 5 satisfies $z_t > 0$.*

Proof. First observe that $\mathbf{x}$ in Step 3 satisfy the relationship

$$\mathbf{x}_{\mathcal{S}} = \arg \min_{\mathbf{y} \in \mathbb{R}^{|\mathcal{S}|}} \|\mathbf{B}_{\mathcal{S}}\mathbf{y} - \mathbf{c}\|_2^2 . \tag{7.4}$$

Eq. (7.4) is trivial at the first iteration, in which $\mathcal{S} = \emptyset$, and at the second or later iteration, this is guaranteed by Step 5 and Step 10. Eq. (7.4) translates into

$$\frac{\partial \|\mathbf{B}_{\mathcal{S}}\mathbf{y} - \mathbf{c}\|_2^2}{\partial \mathbf{y}} (\mathbf{y} = \mathbf{x}_{\mathcal{S}}) = \mathbf{B}_{\mathcal{S}}^T \mathbf{B}_{\mathcal{S}} \mathbf{x}_{\mathcal{S}} - \mathbf{B}_{\mathcal{S}}^T \mathbf{c} = \mathbf{0}.$$

Using $\mathbf{x}_{\mathcal{E}} = \mathbf{0}$, we have $\mathbf{B}\mathbf{x} = \mathbf{B}_{\mathcal{S}}\mathbf{x}_{\mathcal{S}}$ and therefore

$$\mathbf{B}_{\mathcal{S}}^T \mathbf{B}\mathbf{x} - \mathbf{B}_{\mathcal{S}}^T \mathbf{c} = \mathbf{0}.$$

Now, for the index t chosen at Step 3, we know that

$$w_t = \left(\mathbf{B}^T(\mathbf{c} - \mathbf{B}\mathbf{x})\right)_t > 0$$

due to the condition of the while-loop. Let $\tilde{\mathbf{G}} = \begin{pmatrix} \mathbf{B}_{\mathcal{S}} & \mathbf{b}_t \end{pmatrix}$ and $\tilde{\mathbf{h}} = \mathbf{c} - \mathbf{B}\mathbf{x}$. Then

$$\tilde{\mathbf{G}}^T \tilde{\mathbf{h}} = \begin{pmatrix} \mathbf{B}_{\mathcal{S}}^T \\ \mathbf{b}_t^T \end{pmatrix} (\mathbf{c} - \mathbf{B}\mathbf{x}) = \begin{pmatrix} \mathbf{0} \\ w_t \end{pmatrix},$$

with $w_t > 0$.

Now, observe that

$$\begin{aligned} \tilde{\mathbf{G}}\mathbf{v} - \mathbf{c} &= \tilde{\mathbf{G}}\mathbf{v} - \left(\tilde{\mathbf{h}} + \mathbf{B}\mathbf{x}\right) \\ &= \tilde{\mathbf{G}}\mathbf{v} - \left(\tilde{\mathbf{h}} + \tilde{\mathbf{G}} \begin{pmatrix} \mathbf{x}_{\mathcal{S}} \\ 0 \end{pmatrix}\right) \\ &= \tilde{\mathbf{G}} \left(\mathbf{v} - \begin{pmatrix} \mathbf{x}_{\mathcal{S}} \\ 0 \end{pmatrix}\right) - \tilde{\mathbf{h}}. \end{aligned}$$

Therefore, two solutions

$$\mathbf{y}^* = \arg \min_{\mathbf{y} \in \mathbb{R}^{|\mathcal{S}|+1}} \left\|\tilde{\mathbf{G}}\mathbf{y} - \tilde{\mathbf{h}}\right\|_2^2$$

and

$$\mathbf{v}^* = \arg\min_{\mathbf{v}\in\mathbb{R}^{|\mathcal{S}|+1}} \left\|\tilde{\mathbf{G}}\mathbf{v} - \mathbf{c}\right\|_2^2$$

has the relationship

$$\mathbf{y}^* = \mathbf{v}^* - \begin{pmatrix} \mathbf{x}_{\mathcal{S}} \\ 0 \end{pmatrix},$$

since $\tilde{\mathbf{G}}$ is of the full column rank.

Applying Lemma 7.1, $y^*_{|\mathcal{S}|+1} > 0$, and therefore $v^*_{|\mathcal{S}|+1} > 0$. Finally, observe that $v^*_{|\mathcal{S}|+1}$ is assigned to z_t in Step 5. □

To understand the implication of Theorem 7.2, let us assume that the statement is not true, i.e., $z_t \leq 0$. In this case, α in Step 7 is zero, and therefore the current solution candidate $\mathbf{x}$ is not updated in Step 8 making further updates impossible. Therefore, Theorem 7.2 is essential in showing that α is positive and all the steps in Algorith 7.1 are well defined.

Now, the following argument shows that Algorithm 7.1 terminates in a finite number of iterations. For the inner loop, observe that at least one index is removed from $\mathcal{S}$ at each iteration. Hence, the inner loop terminates in at most $|\mathcal{S}|$ steps. The finiteness of the outer loop can be shown by considering the value of the cost function $J(\mathbf{x})$. Because the value of $J(\mathbf{x})$ is strictly reduced after each iteration, set $\mathcal{S}$ at Step 3 is different from all the previous instances of itself. Since only a finite number of cases are possible for set $\mathcal{S}$, the outer loop terminates in a finite number of iterations. In practice, the number of iterations of the outer loop is usually the same or slightly bigger than the size of the passive set, $|\mathcal{S}^*|$.

7.2 Rank deficient NLS

When $\mathbf{B}$ in Eq. (7.1) is rank deficient, it turns out that Algorithm 7.1 is applicable without modification. We prove this by asserting and proving a theorem regarding the new variable chosen at Step (3) in presence of the overall rank deficiency of $\mathbf{B}$.

A key issue is whether $\mathbf{B}_{\mathcal{S}}$ ever becomes rank deficient during the execution of Algorithm 7.1. If this happens, a problem arises because the solutions in Steps 5 and 10 are not uniquely determined, and then Theorem 7.2 does not hold. If Theorem 7.2 does not hold, it is difficult to show the finite termination property. In the following, however, we show that the columns in $\mathbf{B}_{\mathcal{S}}$ indeed remain linearly independent throughout all iterations.

Theorem 7.3. *In Algorithm 7.1, the column corresponding to the index t chosen in Step 3 is linearly independent of the columns indexed by the current $\mathcal{S}$.*

Proof. When $\mathcal{S} = \emptyset$, there is nothing to prove. Suppose that $\mathcal{S}$ is nonempty and that $k \notin \mathcal{S}$. Denote the k^{th} column of $\mathbf{B}$ by $\mathbf{b}_k$. We will show that if $\mathbf{b}_k$ is linearly dependent on the columns in $\mathbf{B}_{\mathcal{S}}$, then k is not selected in Step 3.

Note that, at the end of the previous iteration of the inner loop, $\mathbf{x}$ is feasible ($\mathbf{x} \geq 0$) and is the optimal solution with respect to the current passive set $\mathcal{S}$. We therefore have

$$\frac{\partial \|\mathbf{B}_{\mathcal{S}}\mathbf{x}_{\mathcal{S}} - \mathbf{c}\|_2^2}{\partial \mathbf{x}_{\mathcal{S}}} = \mathbf{B}_{\mathcal{S}}^T\mathbf{B}_{\mathcal{S}}\mathbf{x}_{\mathcal{S}} - \mathbf{B}_{\mathcal{S}}^T\mathbf{c} = \mathbf{0}. \tag{7.5}$$

Now, if $\mathbf{b}_k$ is linearly dependent on the columns in $\mathbf{B}_{\mathcal{S}}$, then $\mathbf{b}_k = \mathbf{B}_{\mathcal{S}}\mathbf{u}$ with some vector $\mathbf{u}$. Then, the k^{th} element of $\mathbf{w}$ in Step 13 is

$$\begin{aligned} w_k &= (\mathbf{b}_k)^T(\mathbf{B}^T\mathbf{x} - \mathbf{c}) \\ &= (\mathbf{B}_{\mathcal{S}}\mathbf{u})^T(\mathbf{B}_{\mathcal{S}}^T\mathbf{x}_{\mathcal{S}} - \mathbf{c}) \\ &= \mathbf{u}^T\mathbf{B}_{\mathcal{S}}^T(\mathbf{B}_{\mathcal{S}}^T\mathbf{x}_{\mathcal{S}} - \mathbf{c}) = 0, \end{aligned}$$

using Eq. (7.5). Therefore, k is not selected in Step 3. $\square$

Theorem 7.3 shows that $\mathbf{B}_{\mathcal{S}}$ does not become rank deficient after Step 3. Because the inner loop only reduces the passive set $\mathcal{S}$, $\mathbf{B}_{\mathcal{S}}$ does not become rank deficient during the inner loop. Hence, Theorem 7.3 is enough to show that $\mathbf{B}_{\mathcal{S}}$ remains full column rank throughout the iterations. The remaining argument of the finite termination property of Algorithm 7.1 for the rank-deficient case is the same as that of the full rank case described above.

Our proof provides deep understanding regarding the initialization of the active-set method. Although $\mathbf{x}$ is initialized with a zero vector in Algorithm 7.1, it is possible to use prior information and initialize $\mathbf{x}$ by a nonzero vector. When $\mathbf{B}$ is rank deficient, however, care must be taken if $\mathbf{x}$ is initialized with a nonzero vector. If $\mathbf{x}$ is set to be a zero vector initially, then $\mathcal{S}$ is initially empty and the column rank of $\mathbf{B}_{\mathcal{S}}$ remains full as we have shown above. If $\mathbf{x}$ is initialized with a nonzero vector for which the corresponding $\mathbf{B}_{\mathcal{S}}$ is rank deficient, then the steps of Algorithm 7.1 might not be well defined. Therefore, unless we have other information that an initial value of $\mathbf{x}$ can be set to nonzero and the corresponding $\mathbf{B}_{\mathcal{S}}$ has full column rank, Algorithm 7.1 needs to be started from $\mathbf{x} = \mathbf{0}$, i.e., with $\mathcal{S} = \emptyset$. For this case, the algorithm will correctly find a solution even when the matrix is rank deficient without ever running into rank-deficient subproblems.

CHAPTER VIII

GROUP SPARSITY IN NONNEGATIVE MATRIX FACTORIZATION

In this chapter, we propose an extension of NMF that incorporates group structure as prior information. Many matrix-represented data sets are inherently structured as groups. For example, a set of documents that are labeled with the same topic forms a group. In drug discovery, various treatment methods are typically applied to a large number of subjects, and the subjects that received the same treatment are naturally viewed as a group. In addition to these groups of data items, a set of features forms a group as well. In computer vision, different types of features such as pixel values, gradient features, and 3D pose features can be viewed as groups. Similarly in bioinformatics, features from microarray and metabolic profiling become different groups.

The motivation of our work is that there are similarities among data items or features belonging to the same group in that their low-rank representations share the same sparsity pattern. However, such similarities have not been previously utilized in NMF. In order to exploit the shared sparsity pattern, we propose to incorporate mixed-norm regularization in NMF. We adopt $l_{1,q}$-norm regularization, and the definition of $l_{1,q}$-norm is shown in Section 8.2. Regularization by l_1-norm is well-known to promote a sparse representation [96]. When l_1-norm is extended to groups of parameters, $l_{1,q}$-norm has been shown to induce a sparse representation at the level of groups [104]. By employing $l_{1,q}$-norm regularization, the latent factors obtained by NMF can be improved with an additional property of shared sparsity.

The adoption of mixed-norm regularization introduces a new challenge to the optimization algorithm for NMF. Since the mixed-norm term is not a smooth function, conventional methods such as the steepest gradient descent cannot be applied. To address the difficulty, we present two algorithms based on recent developments in convex optimization. Both algorithms are developed using the block coordinate descent (BCD) method [10]. The first approach is a matrix-block BCD method, in which the subproblems are handled with an efficient first order optimization scheme [98]. The second approach is a vector-block BCD method, which often converges faster than the former. A strength of the two algorithms we propose is that they generally handle $l_{1,q}$-norm regularization for common cases: $q = \infty$ and $q = 2$. We also provide computational comparisons of the two methods.

We show the effectiveness of mixed-norm regularization for factor recovery using a synthetic data set. In addition, we demonstrate application examples in semi-supervised clustering and multilingual text mining. Our application examples are novel in that the use of group-sparsity regularization for these applications has not been shown before. In the applications, the benefits of nonnegativity constraints and group-sparsity regularization are successfully combined demonstrating that the mixed-norm regularized NMF can be effectively used for real-world data mining applications where nonnegative representations are used.

The rest of this chapter is organized as follows. We begin with discussion on related work in Section 8.1. We then introduce the concept of group sparsity and lead to a problem formulation of NMF with mixed-norm regularization in Section 8.2. We describe optimization algorithms in Section 8.3. We provide the demonstration of recovery example, application examples, and computational comparisons in Section 8.4. We finalize the chapter with discussion in Section 8.5.

8.1 Related Work

Incorporating group information using mixed-norm regularization has been applied to numerous learning problems. Earlier, regularization for sparse representation was popularized with the l_1-norm penalized linear regression called Lasso [96]. L_1-norm penalization is known to promote a sparse solution and improve generalization. Techniques for promoting group-level sparsity using $l_{1,2}$-norm regularization have been investigated by Yuan and Lin and others [104, 61, 79] under the name of group Lasso. Approaches that adopt $l_{1,\infty}$-norm regularization have been subsequently proposed by Liu et al. and others [74, 90, 23] for multi-task learning problems. Regularization methods for more sophisticated structure have also been proposed recently [60, 75].

In matrix factorization, Bengio et al. [7] and Jenatton et al. [50] considered $l_{1,2}$-norm regularization in sparse coding and principal component analysis, respectively. Jenatton et al. [49] further considered hierarchical regularization with tree structure. Jia et al. [51] recently applied $l_{1,\infty}$-norm regularization to sparse coding with a focus on a computer vision application. Masaeli et al. [78] used the idea of $l_{1,\infty}$-norm regularization for feature selection in PCA. The group structure studied in our work is close to those of [7, 49, 51] since they also considered group sparsity shared across data items or features. On the other hand, the hierarchical regularization in [49] is different from ours because their regularization was imposed on parameters *within* each data item. Most importantly, we focus on *nonnegative* factorization in algorithm development as well as in applications, whereas [7, 49, 51] focused on sparse coding or PCA.

In the NMF literature, efforts to incorporate group structure have been fairly limited. Badea [3] presented a simultaneous factorization of two gene expression data sets by extending NMF with an offset vector, as in the affine NMF [26]. Li et al. [69] and Singh and Gordon [94] demonstrated how simultaneous factorization of multiple matrices can be used for knowledge transfer. Jenatton et al. [49] mentioned NMF as a special case in their work on sparse coding, but they only dealt with a particular example without further developments. In addition, the hierarchy structure considered in [49] is different from ours as explained in the previous paragraph. To our knowledge, algorithms and applications of applying group-sparsity regularization to NMF have not been fully investigated before our work in this chapter.

Efficient optimization methods presented in this chapter are built upon recent developments in convex optimization and NMF algorithms. The block coordinate descent (BCD) method forms the basis of our algorithms. In our first algorithm, which is a matrix-block BCD method, we adopt an efficient convex optimization method in [98]. The motivation of our second algorithm is from the hierarchical alternating least squares (HALS) method [24] for standard NMF. Convex optimization theory, in particular the Fenchel duality described in Chapter 2, plays an important role in both of the proposed algorithms.

8.2 Problem Statement

Let us begin our main discussion with a matrix $\mathbf{A} \in \mathbb{R}_+^{M\times N}$. Without loss of generality, we assume that the rows of $\mathbf{A}$ represent features and the columns of $\mathbf{A}$ represent data items. In standard NMF, we are interested in discovering two low-rank factor matrices $\mathbf{W} \in \mathbb{R}_+^{M\times K}$ and $\mathbf{H} \in \mathbb{R}_+^{N\times K}$ such that $\mathbf{A} \approx \mathbf{W}\mathbf{H}^T$. This is typically achieved by minimizing an objective function defined as

$$f(\mathbf{W},\mathbf{H}) = \frac{1}{2}\left\|\mathbf{A} - \mathbf{W}\mathbf{H}^T\right\|_F^2. \tag{8.1}$$

with constraints $\mathbf{W} \geq 0$ and $\mathbf{H} \geq 0$. In this section, we show how we can take group structure into account by adding a mixed-norm regularization term into Eq. (8.1).

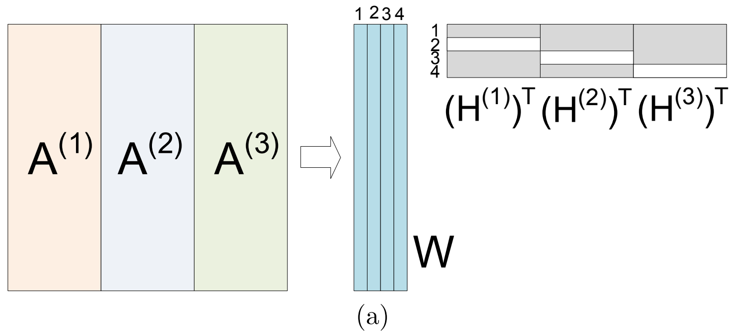
8.2.1 Group structure and group sparsity

Let us first describe the group structure using motivating examples. Diagrams in Figure 8.1 show group structure considered in our work. In Figure 8.1-(a), the columns (that is, data items) are divided into three groups. This group structure is prevalent in clustered data, where data items belonging to each cluster define groups. In text mining, a group can represent documents having the same topic assignment. Another example can be seen from bioinformatics. For the purpose of drug discovery, one typically applies various treatment options to different groups of subjects. In this case, it is important to analyze the difference at the level of groups and discover fundamental understanding of the treatments. The subjects to which the same treatment is applied can be naturally viewed as a group.

On the other hand, groups can be formed from the rows (that is, features) as shown in Figure 8.1-(b). This structure can be seen from multi-view learning problems. In computer vision, as Jia et al. [51] discussed, a few different feature types such as pixel values, gradient-based features, and 3D pose features can be simultaneously used to build a recognition system. In text mining, a parallel multilingual corpus can be seen as a multi-view data set, where the term-document frequency features in each language form a group.

Our motivation is that the feature or data instances that belong to a group are expected to share the same sparsity pattern in low-rank factors. In Figure 8.1-(a), the gray and white rows in $(\mathbf{H}^{(1)})^T$, $(\mathbf{H}^{(2)})^T$, and $(\mathbf{H}^{(3)})^T$ represent nonzero and zero values. For example, columns in $(\mathbf{H}^{(1)}))^T$ share the same sparsity pattern that their second components are all zero. Such group sparsity improves the interpretation of the factorization model: For the reconstruction of data items in $\mathbf{A}^{(1)}$, only the first, the third, and the fourth latent components are used whereas the second component is irrelevant. Similar explanation holds for $\mathbf{A}^{(2)}$ and $\mathbf{A}^{(3)}$ as well. That is, the association of latent components to data items can be understood at the level of groups instead of each data item.

In Figure 8.1-(b), group sparsity is shown for latent component matrices $\mathbf{W}^{(1)}$, $\mathbf{W}^{(2)}$, and $\mathbf{W}^{(3)}$. A common interpretation of multi-view matrix factorization is that


1 2 3 4
1
2
3
4
$(H^{(1)})^T$ $(H^{(2)})^T$ $(H^{(3)})^T$
$A^{(1)}$ $A^{(2)}$ $A^{(3)}$
W
(a)

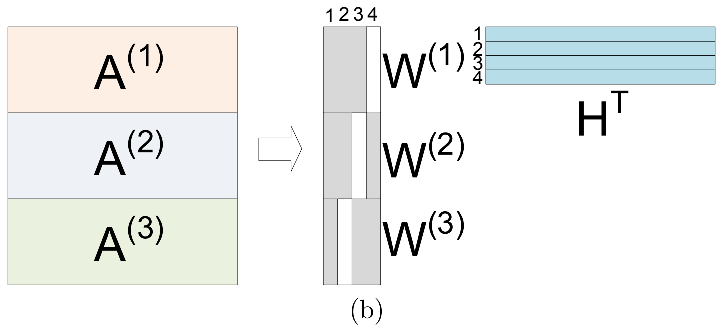


Figure 8.1: (a) Matrix with column groups and its factorization with group sparsity (b) Matrix with row groups and its factorization with group sparsity. The bright rows of $(\mathbf{H}^{(i)})^T$ in (a) and the bright columns of $\mathbf{W}^{(i)}$ in (b) ($i = 1, 2, 3$) represent zero subvectors.

the i^{th} columns of $\mathbf{W}^{(1)}$, $\mathbf{W}^{(2)}$, and $\mathbf{W}^{(3)}$ are associated with each other in a sense that they play the same role in explaining data. With group sparsity, missing associations can be discovered. In Figure 8.1-(b), the second components of $\mathbf{W}^{(1)}$ and $\mathbf{W}^{(2)}$ are associated with each other, but there is no corresponding component in the third view since the second column in $\mathbf{W}^{(3)}$ appeared as zero.

Examples and interpretations provided here are certainly not exhaustive, and we believe the group structure can be found in many other data mining problems. With these motivations in mind, now we proceed to discuss how group sparsity can be promoted by employing mixed-norm regularization.

8.2.2 Formulation with mixed-norm regularization

We discuss using the case of Figure 8.1-(a), where the columns are divided into groups. By considering the factorization $\mathbf{A}^T$, however, all the formulations can be applied to a case with row groups.

Suppose the columns of $\mathbf{A} \in \mathbb{R}^{M\times N}$ are divided into B groups as $\mathbf{A} = \left(\mathbf{A}^{(1)}, \cdots, \mathbf{A}^{(B)}\right)$, where $\mathbf{A}^{(b)} \in \mathbb{R}^{M\times N_b}$ and $\sum_{b=1}^{B} N_b = N$. Accordingly, the coefficient matrix is divided into B groups as $\mathbf{H} = \begin{pmatrix} \mathbf{H}^{(1)} \\ \vdots \\ \mathbf{H}^{(B)} \end{pmatrix}$ where $\mathbf{H}^{(b)} \in \mathbb{R}^{N_b\times K}$. The objective function in Eq. (8.1) is now written as a sum:

$$f(\mathbf{W},\mathbf{H}) = \frac{1}{2}\sum_{b=1}^{B} \left\|\mathbf{A}^{(b)} - \mathbf{W}(\mathbf{H}^{(b)})^T\right\|_F^2 .$$

To promote group sparsity, we add a mixed-norm regularization term for coefficient matrices $\left\{(\mathbf{H}^{(b)})^T\right\}$ using $l_{1,q}$-norm and consider an optimization problem

$$\min_{\mathbf{W}\geq 0,\mathbf{H}\geq 0} f(\mathbf{W},\mathbf{H}) + \alpha \left\|\mathbf{W}\right\|_F^2 + \beta \sum_{b=1}^{B} \left\|(\mathbf{H}^{(b)})^T\right\|_{1,q} . \tag{8.2}$$

We show the definition of $\left\|\cdot\right\|_{1,q}$ below, and the Frobenius norm regularization on $\mathbf{W}$ is used to prevent $\mathbf{W}$ from growing arbitrarily large. Parameters α and β control the strength of each regularization term.

Now let us discuss the role of $l_{1,q}$-norm regularization. For a matrix $\mathbf{Y} \in \mathbb{R}^{a\times b}$, its $l_{1,q}$-norm is defined by

$$\left\|\mathbf{Y}\right\|_{1,q} = \sum_{j=1}^{a} \left\|\mathbf{y}_{j\cdot}\right\|_q = \left\|\mathbf{y}_{1\cdot}\right\|_q + \cdots + \left\|\mathbf{y}_{a\cdot}\right\|_q .$$

That is, the $l_{1,q}$-norm is the sum of l_q-(vector) norms of its rows. Penalization with $l_{1,q}$-norm promotes as many number of zero rows as possible to appear in $\mathbf{Y}$. In Eq. (8.2), the penalty term on $(\mathbf{H}^{(b)})^T$ promotes that coefficient matrices $(\mathbf{H}^{(b)})^T$ contain as many zero rows as possible, and the zero-rows correspond to group sparsity described in Section 8.2.1. Any scalar q, $1 < q \leq \infty$, can be potentially used, but for the development of algorithms, we focus on the two cases of $q = 2$ and $q = \infty$, which are also common in related literature discussed in Section 8.1. In the following, we describe efficient optimization strategies for solving Eq. (8.2).

Algorithm 8.1 Matrix-block BCD method for Eq. (8.2)

```
Input: $\mathbf{A} = \left(\mathbf{A}^{(1)}, \cdots, \mathbf{A}^{(B)}\right) \in \mathbb{R}^{M \times N}$, $\alpha, \beta \in \mathbb{R}_+$
Output: $\mathbf{W} \in \mathbb{R}_+^{M \times K}$, $\mathbf{H}^T = \left((\mathbf{H}^{(1)})^T, \cdots, (\mathbf{H}^{(B)})^T\right) \in \mathbb{R}_+^{K \times N}$
1: Initialize $\mathbf{W}$ and $\mathbf{H}^{(1)}, \cdots, \mathbf{H}^{(B)}$, e.g., with random entries.
2: repeat
3:     Update $\mathbf{W}$ as
           $\mathbf{W} \leftarrow \arg\min_{\mathbf{W} \geq 0} \frac{1}{2} \left\| \mathbf{A} - \mathbf{W}\mathbf{H}^T \right\|_F^2 + \alpha \left\| \mathbf{W} \right\|_F^2 .$   (8.3)
4:     For each $b = 1, \cdots, B$, update $\mathbf{H}^{(b)}$ as
           $\mathbf{H}^{(b)} \leftarrow \arg\min_{\mathbf{H}^{(b)} \geq 0} \frac{1}{2} \left\| \mathbf{A}^{(b)} - \mathbf{W}(\mathbf{H}^{(b)})^T \right\|_F^2 + \beta \left\| (\mathbf{H}^{(b)})^T \right\|_{1,q} .$   (8.4)
5: until convergence
```

8.3 Optimization Algorithms

With mixed-norm regularization, the minimization problem in Eq. (8.2) becomes even more difficult than the standard NMF problem. We here propose two strategies based on the block coordinate descent (BCD) method shown in Chapter 2. The first method is a BCD method with matrix blocks; that is, a matrix variable is minimized at each step fixing all other entries. The second method is a BCD method with vector blocks; that is, a vector variable is minimized at each step fixing all other entries. In both algorithms, the $l_{1,q}$-norm term is handled by using Fenchel duality.

8.3.1 Matrix-block BCD method

The matrix-block BCD method minimizes Eq. (8.2) with one submatrix at a time fixing all other variables. Overall procedure is summarized in Algorithm 8.1.

The subproblem for $\mathbf{W}$ in Eq. (8.3) is easy to solve as it can be transformed to

$$\mathbf{W} \leftarrow \arg\min_{\mathbf{W} \geq 0} \frac{1}{2} \left\| \begin{pmatrix} \mathbf{H} \\ \sqrt{2\alpha}\mathbf{I}_K \end{pmatrix} \mathbf{W}^T - \begin{pmatrix} \mathbf{A}^T \\ \mathbf{0}_{K \times M} \end{pmatrix} \right\|_F^2 , \tag{8.5}$$

which is the nonnegativity-constrained least squares (NLS) problem. An efficient algorithm for the NLS problem, as discussed in Chapter 4, can be used to solve

Algorithm 8.2 A convex optimization method for Eq. (8.7)

```
Input: $\mathbf{B} \in \mathbb{R}^{p\times r}$, $\mathbf{C} \in \mathbb{R}^{p\times t}$, $\beta \in \mathbb{R}_+$
Output: $\mathbf{X} \in \mathbb{R}_+^{r\times t}$
1: Choose $\mathbf{X}^{(0)}$,$\tilde{\mathbf{X}}^{(0)}$ and let $\tau^{(0)} = 1$.
2: for $k = 0, 1, 2, \cdots$ , until convergence do
3:     $\mathbf{Y}^{(k)} \leftarrow \tau^{(k)}\mathbf{X}^{(k)} + (1 - \tau^{(k)})\tilde{\mathbf{X}}^{(k)}$
4:     Update
           $$\mathbf{X}^{(k+1)} \leftarrow \arg\min_{\mathbf{X}\geq 0} \left\|\mathbf{X} - \mathbf{U}^{(k)}\right\|_F^2 + \frac{2\beta}{\tau^{(k)}L} \|\mathbf{X}\|_{1,q}\,, \qquad (8.6)$$
       where $\mathbf{U}^{(k)} = \mathbf{X}^{(k)} - \frac{1}{\tau^{(k)}L}\left(\mathbf{B}^T\mathbf{B}\mathbf{Y}^{(k)} - \mathbf{B}^T\mathbf{C}\right)$ and $L = \sigma_{max}\left(\mathbf{B}^T\mathbf{B}\right)$.
5:     $\tilde{\mathbf{X}}^{(k+1)} \leftarrow \tau^{(k)}\mathbf{X}^{(k+1)} + (1 - \tau^{(k)})\tilde{\mathbf{X}}^{(k)}$
6:     Find $\tau^{(k+1)} > 0$ such that
           $$\left(\tau^{(k+1)}\right)^{-2} - \left(\tau^{(k+1)}\right)^{-1} = \left(\tau^{(k)}\right)^{-2}.$$
7: end for
8: Return $\tilde{\mathbf{X}}^{(k)}$.
```

Eq. (8.5). Solving the subproblem for $\mathbf{H}^{(b)}$ in Eq. (8.4) is a more involved task, and an algorithm for this problem is discussed in the following.

The problem for $\mathbf{H}^{(b)}$ can be generally written as follows. Given two matrices $\mathbf{B} \in \mathbb{R}_+^{p\times r}$ and $\mathbf{C} \in \mathbb{R}_+^{p\times t}$, we would like to solve

$$\min_{\mathbf{X}\geq 0} \frac{1}{2} \|\mathbf{B}\mathbf{X} - \mathbf{C}\|_F^2 + \beta \|\mathbf{X}\|_{1,q}\,. \tag{8.7}$$

Observe that the objective function of Eq. (8.7) is composed of two terms: $g(\mathbf{X}) = \frac{1}{2}\|\mathbf{B}\mathbf{X} - \mathbf{C}\|_F^2$ and $h(\mathbf{X}) = \beta\|\mathbf{X}\|_{1,q}$. The both of $g(\mathbf{X})$ and $h(\mathbf{X})$ are convex functions, the first term $g(\mathbf{X})$ is differentiable, and $\nabla g(\mathbf{X})$ is Lipschitz continuous. Hence, an efficient convex optimization method can be adopted.

Algorithm 8.2 shows a first-order convex optimization method [98] derived for Eq. (8.7). It is a variant of the Nesterov's method [81], which has been widely used due to its theoretical strength and empirical success. An important requirement in Algorithm 8.2 is the ability to efficiently solve the subproblem in Eq. (8.6). Observe that the problem can be separated with respect to each row of $\mathbf{X}$. Focusing on the

i^{th} row of $\mathbf{X}$, it suffices to solve a problem in the following form:

$$\min_{\mathbf{x}\geq 0} \frac{1}{2}\left\|\mathbf{x}-\mathbf{v}\right\|^2 + \eta \left\|\mathbf{x}\right\|_q \tag{8.8}$$

where $\mathbf{x}$, $\mathbf{v}$, and η replace $\mathbf{x}_{i\cdot}$, $\left(\mathbf{U}^{(k)}\right)_{i\cdot}$, and $\frac{\beta}{\tau^{(k)}L}$, respectively.

It is important to observe that the problem in Eq. (8.8) can be handled without the nonnegativity constraints. The following proposition summarizes this observation. Jenatton et al. [49] briefly mentioned the statement but did not provide the proof. Suppose $[\cdot]_+$ denotes the element-wise projection operator to nonnegative numbers.

Proposition 8.1. *Consider the minimization problem in Eq.* (8.8) *and a modified problem as follows:*

$$\min_{\mathbf{x}} \frac{1}{2}\left\|\mathbf{x}-[\mathbf{v}]_+\right\|^2 + \eta \left\|\mathbf{x}\right\|_q . \tag{8.9}$$

If $\mathbf{x}^*$ *is the minimizer of the problem in Eq.* (8.9)*, then* $\mathbf{x}^*$ *is element-wise nonnegative, and it also attains the minimum of the problem in Eq.* (8.8)*.*

Proof. The nonnegativity of $\mathbf{x}^*$ can be seen by the fact that any negative element can be set as zero decreasing the objective function of Eq. (8.9). The remaining relationship can be seen by considering an intermediate problem

$$\min_{\mathbf{x}\geq 0} \frac{1}{2}\left\|\mathbf{x}-[\mathbf{v}]_+\right\|^2 + \eta \left\|\mathbf{x}\right\|_q . \tag{8.10}$$

Comparing Eq. (8.9) and Eq. (8.10), since the minimizer $\mathbf{x}^*$ of the unconstrained problem in Eq. (8.9) satisfies nonnegativity, it is certainly a minimizer of the constrained problem in Eq. (8.10). Now, let the minimizer of Eq. (8.8) be $\tilde{\mathbf{x}}^*$, and consider the set of indices $\mathcal{N} = \{i : v_i \leq 0\}$. Then, it is easy to check $\left(\tilde{\mathbf{x}}^*\right)_i = \left(\mathbf{x}^*\right)_i = 0$ for all $i \in \mathcal{N}$. Moreover, ignoring the variables corresponding to $\mathcal{N}$, the problems in Eq. (8.8) and Eq. (8.10) are equivalent. Therefore, $\mathbf{x}^*$ is the minimizer of Eq. (8.8) $\square$

Proposition 8.1 transforms Eq. (8.8) into Eq. (8.9), where the nonnegativity constraints are dropped. This transformation is important since Eq. (8.9) can now be solved with Fenchel duality as follows. According to Fenchel duality and Corollary 2.18, the following problem is dual to Eq. (8.9):

$$\min_{\mathbf{s}} \frac{1}{2} \left\| \mathbf{s} - [\mathbf{v}]_+ \right\|_2^2 \text{ subject to } \|\mathbf{s}\|_{q^*} \leq \eta, \tag{8.11}$$

where $\|\cdot\|_{q^*}$ is the dual norm of $\|\cdot\|_q$. Eq. (8.11) is a projection problem to a l_{q^*}-norm ball of size η. Based on Theorem 2.16, the dual norm of $\|\cdot\|_2$ is itself, and the dual norm of $\|\cdot\|_\infty$ is $\|\cdot\|_1$. Therefore, the problem in Eq. (8.11) with $q = 2$ is written as

$$\min_{\mathbf{s}} \frac{1}{2} \left\| \mathbf{s} - [\mathbf{v}]_+ \right\|_2^2 \text{ subject to } \|\mathbf{s}\|_2 \leq \eta, \tag{8.12}$$

which can be solved simply by normalization. With $q = \infty$, Eq. (8.11) is written as

$$\min_{\mathbf{s}} \frac{1}{2} \left\| \mathbf{s} - [\mathbf{v}]_+ \right\|_2^2 \text{ subject to } \|\mathbf{s}\|_1 \leq \eta, \tag{8.13}$$

which can be solved by an algorithm described in [86, 31]. Once Eq. (8.11) is solved with a minimizer $\mathbf{s}^*$, the optimal solution for Eq. (8.8) is recovered by $\mathbf{x}^* = [\mathbf{v}]_+ - \mathbf{s}^*$.

8.3.2 Vector-block BCD Method

The matrix-block BCD algorithm has been shown to be quite successful for NMF and its variations. However, observations in Chapter 4 indicate that the vector-block method [24] is also very efficient, often outperforming the matrix-block method. Accordingly, for the group regularized problem in Eq. (8.2), we develop a vector-block method as follows.

In vector-block BCD method, optimal solutions to subproblems with only respect to each column of $\mathbf{W}, \mathbf{H}^{(1)}, \cdots, \mathbf{H}^{(b)}$ are sought. Overall procedure is shown in Algorithm 8.3. In Algorithm 8.3, the solution of the first subproblem in Eq. (8.14) is simply written as

$$\mathbf{w}_k \leftarrow \left[\frac{\mathbf{R}_k \mathbf{h}_k}{2\alpha + \|\mathbf{h}_k\|^2} \right]_+ .$$

The second subproblem in Eq. (8.15) is rewritten as

$$\min_{\mathbf{h} \geq 0} \frac{\|\mathbf{w}_k\|^2}{2} \left\| \mathbf{h} - \frac{\left(\mathbf{R}_k^{(b)}\right)^T \mathbf{w}_k}{\|\mathbf{w}_k\|_2^2} \right\|^2 + \beta \|\mathbf{h}\|_q , \tag{8.16}$$

Algorithm 8.3 Vector-block BCD method for Eq. (8.2)

```
Input: $\mathbf{A} = \left(\mathbf{A}^{(1)}, \cdots, \mathbf{A}^{(B)}\right) \in \mathbb{R}^{M\times N}$ ,$\alpha, \beta \in \mathbb{R}_+$
Output: $\mathbf{W} \in \mathbb{R}_+^{M\times K}$, $\mathbf{H}^T = \left((\mathbf{H}^{(1)})^T, \cdots, (\mathbf{H}^{(B)})^T\right) \in \mathbb{R}_+^{K\times N}$
1: Initialize $\mathbf{W}$ and $\mathbf{H}^{(1)}, \cdots, \mathbf{H}^{(B)}$, e.g., with random entries.
2: repeat
3:     For each $k = 1, \cdots, K$, update $\mathbf{w}_k (\in \mathbb{R}^{M\times 1})$ as
           $\mathbf{w}_k \leftarrow \arg\min_{\mathbf{w}\geq 0} \frac{1}{2} \left\|\mathbf{R}_k - \mathbf{w}\mathbf{h}_k^T\right\|_F^2 + \alpha \left\|\mathbf{w}\right\|^2,$    (8.14)
       where $\mathbf{R}_k = \mathbf{A} - \sum_{\tilde{k}=1, \tilde{k}\neq k}^{K} \mathbf{w}_{\tilde{k}} \mathbf{h}_{\tilde{k}}^T$.
4:     For each $b = 1, \cdots, B$ and then for each $k = 1, \cdots, K$, update $\mathbf{h}_k^{(b)} (\in \mathbb{R}^{N_b\times 1})$ as
           $\mathbf{h}_k^{(b)} \leftarrow \arg\min_{\mathbf{h}\geq 0} \frac{1}{2} \left\|\mathbf{R}_k^{(b)} - \mathbf{w}_k\mathbf{h}^T\right\|_F^2 + \beta \left\|\mathbf{h}\right\|_q,$    (8.15)
       where $\mathbf{R}_k^{(b)} = \mathbf{A}^{(b)} - \sum_{\tilde{k}=1, \tilde{k}\neq k}^{K} \mathbf{w}_{\tilde{k}} (\mathbf{h}_{\tilde{k}}^{(b)})^T$.
5: until convergence
```

where $\mathbf{h} \in \mathbb{R}^{N_b\times 1}$. Comparing Eq. (8.16) and Eq. (8.8), observe that the two are equivalent by replacing $\mathbf{v}$ and η of Eq. (8.8) with $\frac{\left(\mathbf{R}_k^{(b)}\right)^T \mathbf{w}_k}{\|\mathbf{w}_k\|_2^2}$ and $\frac{\beta}{\|\mathbf{w}_k\|^2}$ of Eq. (8.16). Therefore, Eq. (8.16) in turn can be solved based on Proposition 8.1 and the dual problem in Eq. (8.11).

Remark.

It is worth emphasizing the difference of the matrix-block and the vector-block BCD methods. Although the both eventually rely on Proposition 8.1 and the dual form in Eq. (8.11), the goal of matrix-block subproblems is finding optimal solutions of Eq. (8.3) and Eq. (8.4), where a matrix is a variable to optimize. In contrast, the vector-block method seeks optimal solutions of Eq. (8.14) and Eq. (8.15), where a vector is a variable to optimize. The matrix-block method is a BCD method with $B + 1$ blocks whereas the vector-block method is a BCD method with $K(B + 1)$ blocks. They share the same convergence property that every limit point is stationary based on Theorem 2.14, but their actual efficiency may be different as we show in Section 8.4.4.

8.4 Implementation Results

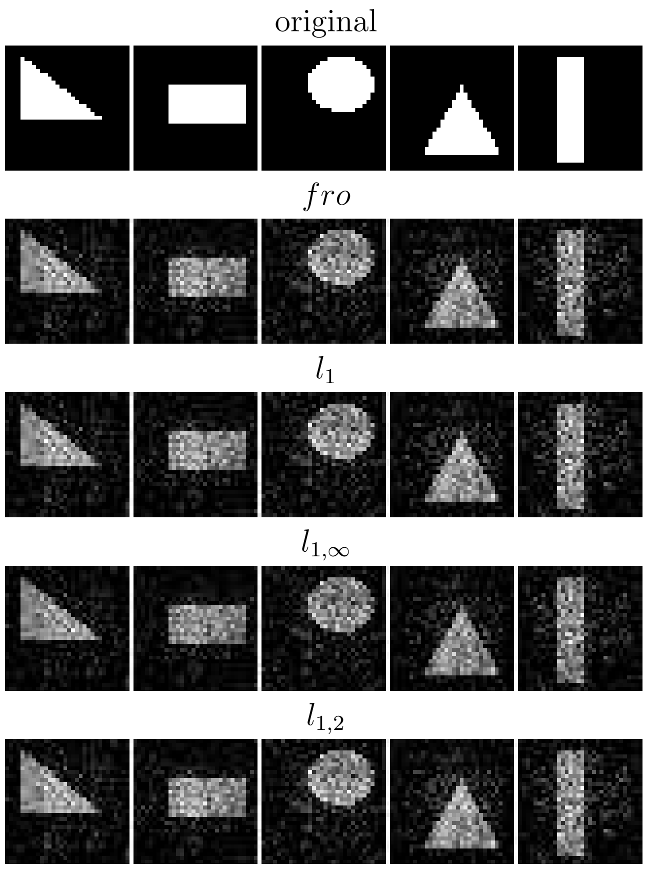
Our implementation section is composed of four folds. We first demonstrate the effectiveness of group-sparsity regularization with a synthetically generated example. We then show an application of the column grouping (Figure 8.1-(a)) in semi-supervised clustering and an application of the row grouping (Figure 8.1-(b)) in multilingual text analysis. Finally, we present the computational comparison of the matrix-block and the vector-block methods.

8.4.1 Factor recovery

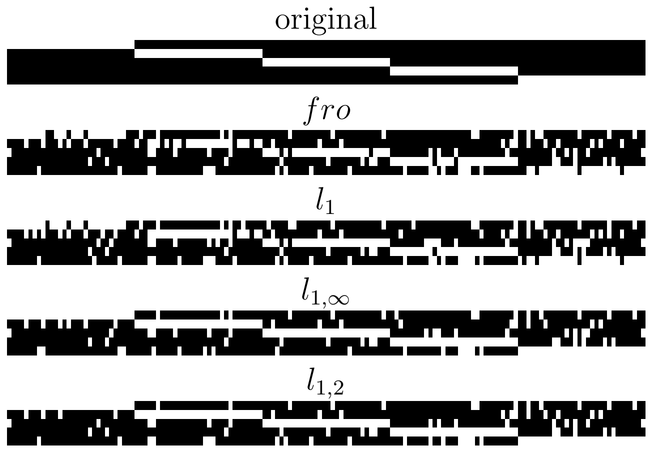
Our first demonstration is the comparison of several regularization methods using a synthetically created data set. Figure 8.2 shows the original data and recovery results. The five original images in the top of Figure 8.2-(a) are of 32×32 pixels, and each of them are vectorized to construct a $1,024 \times 5$ latent component matrix $\mathbf{W}$. Five coefficient matrices $(\mathbf{H}^{(1)})^T, \cdots, (\mathbf{H}^{(5)})^T$ of size 5×30 each are constructed by setting the i^{th} row of $(\mathbf{H}^{(i)})^T$ as zero for $i = 1, 2, 3, 4, 5$ and then filling all other entries by taking random numbers from the uniform distribution on $[0, 1]$. The top image of Figure 8.2-(b) shows the zero and nonzero pattern of $\mathbf{H}^T = \left((\mathbf{H}^{(1)})^T, \cdots, (\mathbf{H}^{(5)})^T\right) \in 5 \times 120$, where dark entries represent nonzeros and bright entries represent zeros. The zero rows of each block are clearly shown as bright rows.

We multiplied $\mathbf{W}$ with $\mathbf{H}^T$ to generate a matrix with five blocks and added Gaussian noise so that the signal-to-noise ratio is 0.3. Under this high noise condition, we tested the ability of various regularization methods in terms of recovering the group structure of the original matrices. Strong noise is common in applications such as video surveillance or Electroencephalography (EEG) analysis in neuroscience. Two alternative regularization methods are considered as competitors:

$$\min_{\mathbf{W}\geq 0, \mathbf{H}\geq 0} f(\mathbf{W}, \mathbf{H}) + \alpha \|\mathbf{W}\|_F^2 + \beta \|\mathbf{H}\|_F^2, \tag{8.17}$$



(a)



(b)

Figure 8.2: (a) Original latent factor images and recovered factor images with various regularization methods (b) Original coefficient matrix and recovered coefficient matrices with various regularization methods. In each of (a) and (b), first row: original factors, second row: recovered by Eq. (8.17), third row: recovered by Eq. (8.18), fourth row: recovered by Eq. (8.2) with $q = \infty$, fifth row; recovered by Eq. (8.2) with $q = 2$. See text for more details.

and

$$\min_{\mathbf{W}\geq 0,\mathbf{H}\geq 0} f(\mathbf{W},\mathbf{H}) + \alpha \left\|\mathbf{W}\right\|_F^2 + \beta \sum_{n=1}^{N} \left\|\mathbf{h}_{n\cdot}\right\|_1^2 . \tag{8.18}$$

Eq. (8.17) and Eq. (8.18) impose the Frobenius norm and l_1-norm regularization on $\mathbf{H}$, respectively, and neither of them take the group structure into account. Reformulations for solving Eq. (8.17) and Eq. (8.18) are described in Section 3.1.4. For the group-sparsity regularization method, we considered Eq. (8.2) with $q = \infty$ and $q = 2$. For all cases, parameters α and β need to be provided as input, and we determined them by cross validation: We iterated all possible combinations of $\alpha, \beta \in \{10^3, 10^2, \cdots, 10^{-4}\}$ and chose a pair for which the reconstruction error is the minimum for another data matrix constructed in the same way. For each case of α and β pair, ten random initializations are tried, and the best is chosen.

In Figure 8.2-(a), it can be seen that the recovered latent components from the four different regularization methods are visually similar to each other. However, in the coefficient matrix shown in Figure 8.2-(b), the drawback of conventional regularization methods stands out. In the coefficient matrices recovered by the Frobenius norm or the l_1-norm regularization, the group structure was lost, and nonzero (dark) elements appeared in the rows of zero (bright) values that present in the original matrix. In contrast, in the coefficient matrices recovered by the $l_{1,\infty}$-norm or the $l_{1,2}$-norm regularization, the group structure was preserved because the zero (bright) rows remained the same as the original matrix.

The failure to recover the group structure leads to a misinterpretation about the role of latent factors. In original matrices, the first group is constructed only with the 2^{nd}, 3^{rd}, 4^{th}, and 5^{th} latent components, and the second group is constructed with only 1^{st}, 3^{rd}, 4^{th}, and 5^{th} latent components, and so on. However, the coefficient matrices recovered by Frobenius norm or the l_1-norm regularization suggest that all the five factors participate in all the groups, which is an incorrect understanding.

8.4.2 Semi-supervised clustering

Our next demonstration is an application example of the group-sparsity regularization with the column groups as shown in Figure 8.1-(a). One of successful applications of NMF is document clustering, and here we show that the group-sparsity regularization can be used for incorporating side-information for clustering.

When NMF is used for clustering (see [103, 58]), the maximum element from each column of $\mathbf{H}$ is chosen to determine clustering assignments. That is, for a group of documents belonging to the same cluster, their representations in matrix $\mathbf{H}$ are similar to each other in a sense that the indices of elements having the maximum value at each column are the same. In particular, if a group of columns in $\mathbf{H}$ share the same sparsity pattern, it is likely that their clustering assignments are the same. Motivated by this observation, we propose to impose group-sparsity regularization for the documents that are supervised to be in the same cluster (i.e., *'must-link'* constraints). In this way, the documents will be promoted to have the same clustering assignments, and latent factor matrix will be accordingly adjusted. As a result, the accuracy of clustering assignments for the unsupervised part can be improved.

We tested this task with two text data sets as follows. The Topic Detection and Tracking corpus 2 (TDT2)[1] is a collection of English news articles from various sources such as NYT, CNN, and VOA in 1998. The 20 Newsgroups data set[2] is a collection of newsgroup documents in 20 different topics. From term-document matrices constructed from these data sets [17][3], we randomly selected $K = 5$ (and $K = 10$) topics that have at least 60 documents and extracted random samples of 60 documents from them. Then, 10 documents from each topic were used as a supervised set, and the rest 50 were an unsupervised (i.e., test) set. A matrix

[1]`http://projects.ldc.upenn.edu/TDT2/`
[2]`http://people.csail.mit.edu/jrennie/20Newsgroups/`
[3]`http://www.zjucadcg.cn/dengcai/Data/TextData.html`

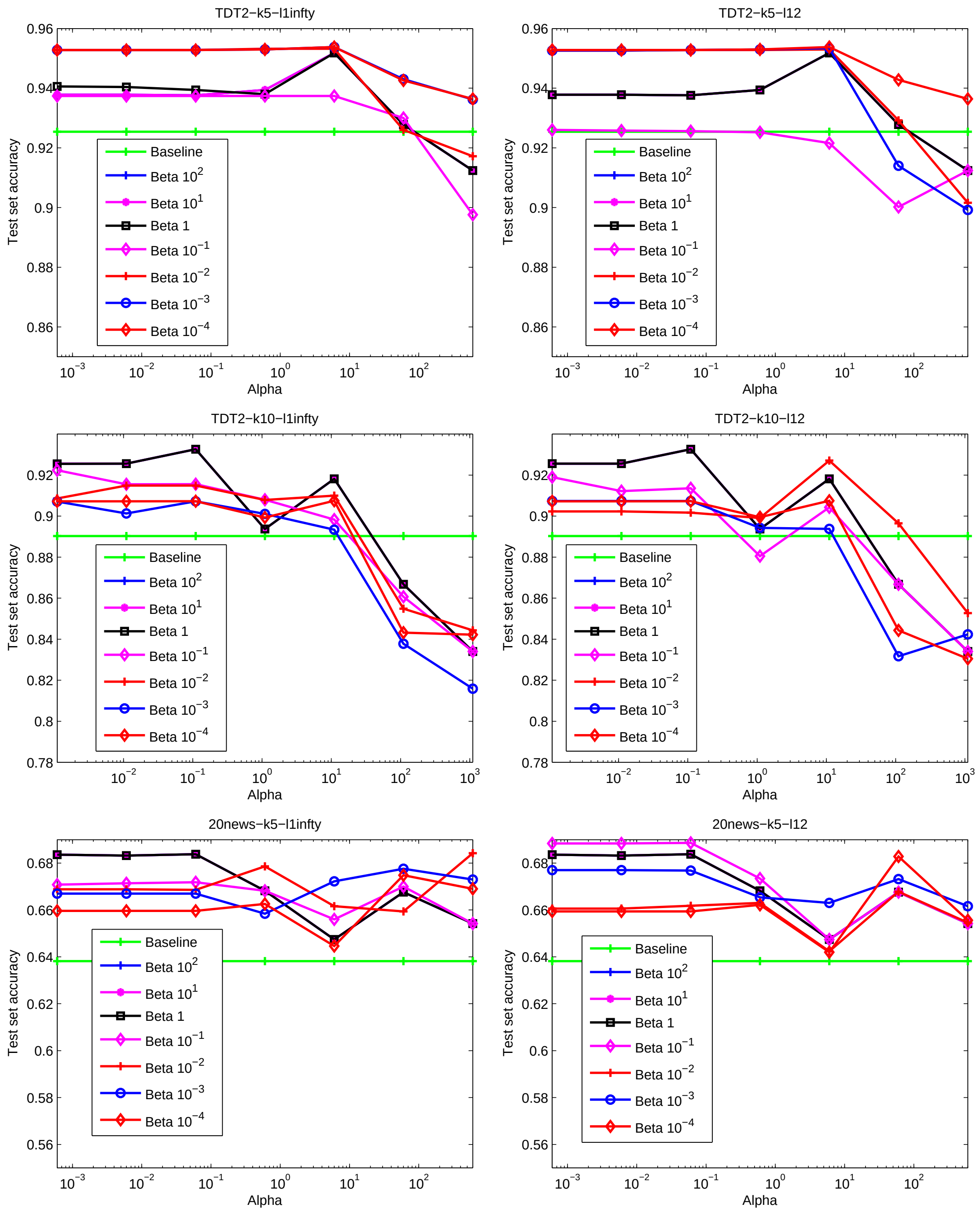


Figure 8.3: Accuracy of semi-supervised clustering with group-sparsity regularization. The x-axis shows the values of α, and the y-axis shows clustering accuracy. Baseline represents the result of no regularization ($\alpha = \beta = 0$). Top: TDT2 data set with $K = 5$, middle: TDT2 data set with $K = 10$, bottom: 20 newsgroups data set with $K = 5$, left: $q = \infty$, right: $q = 2$.

factorization problem is constructed with $(K+1)$ groups: For the first K groups each having 10 supervised documents, group-sparsity regularization is applied, whereas the last group having total $50 \times K$ unsupervised documents was with no regularization. As a result, for a matrix $\mathbf{A} = \left(\mathbf{A}^{(1)}, \cdots, \mathbf{A}^{(K)}, \mathbf{A}^{(K+1)}\right)$ where $\mathbf{A}^{(K+1)}$ represents the unsupervised part, we used the following formulation

$$\min_{\mathbf{W} \geq 0, \mathbf{H} \geq 0} f(\mathbf{W}, \mathbf{H}) + \alpha \|\mathbf{W}\|_F^2 + \beta \sum_{b=1}^{K} \left\|(\mathbf{H}^{(b)})^T\right\|_{1,q}, \tag{8.19}$$

where $\mathbf{H}^T = \left((\mathbf{H}^{(1)})^T, \cdots, (\mathbf{H}^{(K)})^T, (\mathbf{H}^{(K+1)})^T\right)$. Note that no regularization is imposed for the last group $\mathbf{H}^{(K+1)}$. The goal is to solve Eq. (8.19) and accurately assign clustering labels using $\mathbf{H}^{(K+1)}$. We selected the most frequent 10,000 terms to reduce the matrix size. We repeated with 10 different random samples and evaluated the average clustering accuracy with the Hugarian method.[4]

The execution results are shown in Figure 8.3. In semi-supervised clustering, choosing a good parameter setting is difficult because a standard method such as cross validation is not straightforward to apply. Therefore, instead of showing results for a specific choice of α and β, we present how the performance of suggested approach depends on α and β. The results shown in the figures show a reasonable trend. The group-sparsity regularization does boost the clustering performance, but too strong regularization such as $\alpha > 10$ is often harmful. It can be seen that a wide selection of the parameter values, $\alpha \in [10^{-4}, 10]$ and $\beta \in [10^{-4}, 10^2]$, can be used to improve the clustering accuracy.

Note that the goal of our demonstration is not to argue that the group-sparsity regularization is the best semi-supervised clustering approach. Such an investigation requires the in-depth consideration of other semi-supervised clustering methods, and it is beyond the scope of this chapter. In fact, the group-sparsity regularization can

[4]`http://en.wikipedia.org/wiki/Hungarian_method`

be potentially combined with other matrix-factorization based semi-supervised clustering methods [101], and the combination will be an interesting future work. In addition, the group-sparsity regularization takes into account only '*must-link*' constraints, and combining with another approach for handling '*cannot-link*' constraints is also a promising avenue for further study.

8.4.3 Multilingual text analysis

Now, we turn to an application of the group-sparsity regularization with the row groups (as opposed to the column groups in the previous subsection). We consider the task of analyzing multilingual text corpus, which is becoming more important under the trend of rapidly increasing amount of web text information. Demand for a multilingual system is particularly high in a nation or a community, such as EU, where multiple official languages are used. An effective approach in multilingual modeling is to make use of parallel (i.e., translated) corpus to discover aligned latent topics. Aligned latent topics can then be used for topic visualization, cross-lingual retrieval, or classification. In this subsection, we show how group-sparsity regularization can be used to improve the interpretation of aligned latent topics.

We have used the DGT Multilingual Translation Memory (DGT-TM)[5] in our analysis. This corpus contains the body of EU law, which is partially translated into 22 languages. We have analyzed documents in English, French, German, and Dutch, which will be denoted by EN, FR, DE, and NL. Applying stop-words and stemmer for each language, we selected the most frequent 10,000 terms in each language to construct term-document matrices. Matrix factorization problem was set up as in Figure 8.1-(b). As we deal with four languages, the source matrix $\mathbf{A}$ consists of four row blocks: $\mathbf{A}^T = \left((\mathbf{A}^{(1)})^T, \cdots, (\mathbf{A}^{(4)})^T\right)$. Columns of these matrices contain the term-document representation of the same document in four different languages.

[5] `http://langtech.jrc.it/DGT-TM.html`

Not all documents are translated into all languages, so a script to extract pairwise corpora was used. The source matrix $\mathbf{A}$ is not fully observed in this case, and missing parts were treated with zero weights. Once a low-rank factorization is obtained, the columns of $\mathbf{W}^{(1)}, \cdots, \mathbf{W}^{(4)}$ with the same column index are interpreted as aligned latent topics that convey the same meaning but in different natural languages.

The expected benefit of group-sparsity regularization is removing noisy alignments of the latent factors. That is, if a certain topic component appear in documents only in a subset of languages, we would like to detect a zero column in the latent factor for the language where the topic is missing. To test this task, we used a partial corpora from DGT-TM as follows. We collected pairwise translation corpora for EN-FR, EN-DE, and EN-NL (of sizes 1,273, 1,295, and 632, respectively), and appended single language documents in EN, FR, DE, and NL (of sizes 1,300, 930, 610, and 699, respectively). Using $q = \infty$, $K = 500$, $\alpha = 10$, and $\beta = 30$, the algorithm described in Section 8.3 was applied to $\mathbf{A}^T$. The columns in $\mathbf{W}$ are sorted in a decreasing amount of explained variance, and keywords in each topic are listed in a decreasing order of the weights given to each term. The results are summarized in Table 8.1.

Out of $K = 500$ columns, six of them resulted empty, making the 494th topic be the last one in Table 8.1. Two aspects of the results can be noted as a summary. First, the keywords in each language of the same topic appeared quite well aligned in general. Second, zero columns indeed were detected in some of the discovered topics. For example, the 231th topic, which is regarding vehicles and trailers, appeared only in English and Dutch documents. Similarly, the 452th topic, which is regarding ships, appeared only in English and German documents. When we tried without group-sparsity regularization, however, all the columns of $\mathbf{W}$ appeared as nonzero.

Table 8.1: Summary of topics analyzed by group-sparsity regularized NMF.

Id		Keywords
2	EN	member, state, institut, benefit, person, legisl, resid, employ, regul, compet, insur, pension
	FR	procdur, march, de, passat, membr, adjud, recour, d'un, consider, une, aux, concili
	DE	akt, gemeinschaft, rechtsakt, bestimm, europa, leitlini, organ, abfass, dies, sollt, erklar, artikel
	NL	regel, bevoegd, artikel, grondgebied, stat, organ, lid-stat, tijdvak, wettelijk, uitker, werknemer, krachten
14	EN	test, substanc, de, use, en, toxic, prepar, soil, concentr, effect, may, method
	DE	artikel, nr, verordn, flach, eg, absatz, mitgliedstaat, flachenzahl, gemass, erzeug, anhang, wirtschaftsjahr
	NL	word, and, effect, stoff, test, preparat, teststof, stof, la, per, om, kunn
231	EN	brake, shall, vehicl, test, system, point, trailer, control, line, annex, requir, type
	NL	de, moet, voertuig, punt, bijlag, aanhangwag, op, remm, wordt, niet, dor, mag
302	EN	statist, will, develop, polici, european, communiti, programm, inform, data, need, work, requir
	FR	statist, europen, une, polit, programm, un, dvelopp, don, aux, communautair, l'union, mis
	DE	statist, europa, dat, entwickl, programm, information, erford, bereich, neu, dies, gemeinschaft, arbeit
	NL	vor, statistisch, statistiek, europes, word, ontwikkel, over, zull, om, gebied, communautair, programma
392	EN	shall, requir, provid, class, system, space, door, deck, fire, bulkhead, ship, regul
	DE	so, schiff, raum, muss, klass, tur, absatz, vorhand, deck, stell, maschinenraum, regel
452	EN	must, machineri, design, use, oper, safeti, manufactur, risk, requir, construct, direct, person
	NL	moet, machin, zijn, de, dor, om, fabrikant, niet, lidstat, overeenstemm, eis, elk
488	EN	clinic, case, detect, antibodi, isol, compat, diseas, demonstr, specimen, fever, pictur, specif
	FR	dtect, cliniqu, une, cas, mis, vident, malad, isol, part, chantillon
	DE	nachweis, klinisch, prob, isolier, bild, vereinbar, fall, spezif, fieb, krankheit, akut, ohn
	NL	klinisch, geval, dor, ziekt, aanton, isolatie, beeld, detectie, monster, bevestigd, niet, teg
494	EN	european, council, schengen, union, treati, visa, decis, articl, provis, nation, protocol, common
	FR	europen, conseil, l'union, vis, dcis, prsent, trait, schengen, commun, tat, communaut, protocol
	DE	europa, rat, union, beschluss, vertrag, gemeinsam, ubereinkomm, artikel, dies, schengen-besitzstand
	NL	de, europes, rad, besluit, overeenkomst, verdrag, protocol, bepal, betreff, lidstat, unie, gemeenschap

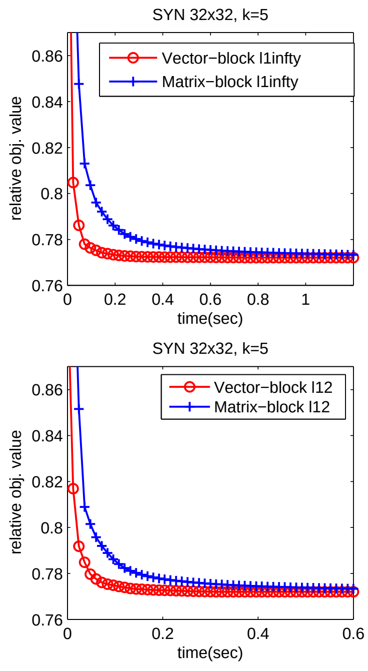
8.4.4 Timing comparison

Our last experiments are the comparison of computational efficiency of the two algorithms for solving group-sparsity regularized NMF. Using the data sets used for previous three demonstrations, we executed the two methods and compared time-vs-objective value graphs. Several initialization values are typically tried in NMF, and the execution of one initial value appears as a piece-wise linear decreasing function. We averaged the graphs for 10 initializations and used to generate the plots shown in Figure 8.4.

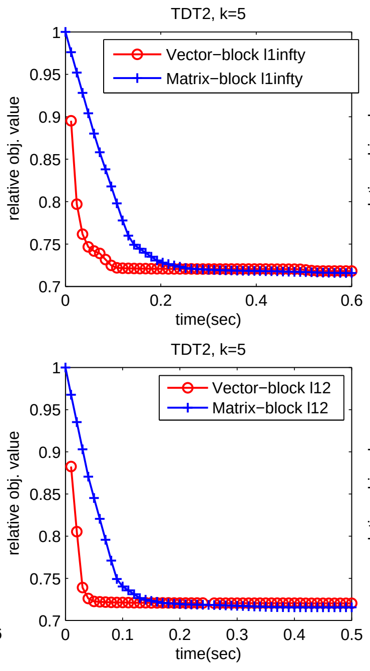
From the figure, it can be seen that the vector-block BCD method converges to a minimum faster than the matrix-block BCD method does. The trend is consistent in both dense (synthetic data set) and sparse (text data sets) matrices. In a non-convex optimization problem such as NMF, each execution may converge to a different local minimum, but the converged minima found by the two methods were in general close to each other.

8.5 Discussion

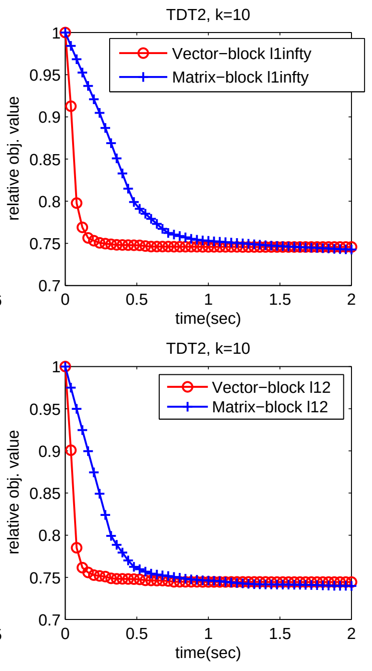
In this chapter, we proposed mixed-norm regularization methods for promoting group sparsity in NMF. Regularization by $l_{1,q}$-norm successfully promotes that the sparsity pattern is shared among data items or features within a group. Efficient convex optimization methods based on the block coordinate descent (BCD) method are presented, and the comparisons of them are also provided. Effectiveness of group-sparsity regularization is demonstrated with application examples for factor recovery, semi-supervised clustering, and multilingual analysis.



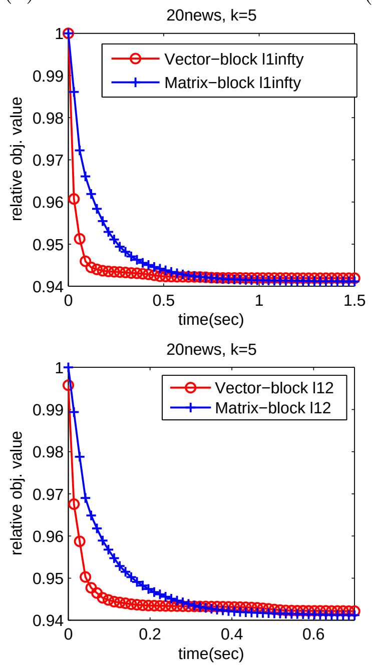
(a)



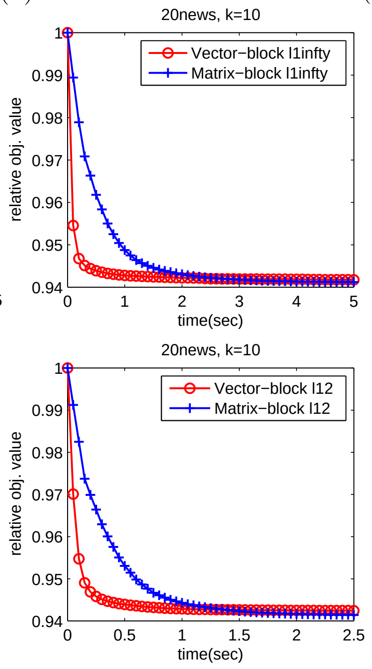
(b)



(c)



(d)



(e)

Figure 8.4: Computational comparisons of the matrix-block and the vector-block BCD methods. The x-axis shows execution time, and the y-axis shows the value of the objective function in Eq. (8.2.1) divided by its evaluation with initial random inputs. All graphs show average results from 10 random initializations. (a) synthetic data set used in Section 8.4.1 (b) TDT2 data set with $K = 5$ (c) TDT2 data set with $K = 10$ (d) 20 newsgroup data set with $K = 5$ (e) 20 newsgroup data set with $K = 10$. In each of (a)-(e), top: $q = \infty$, bottom: $q = 2$.

CHAPTER IX

CONCLUSIONS AND DISCUSSION

9.1 Summary of Contributions

Having the details of this thesis presented in previous chapters, now we summarize the contributions from a wider view as follows.

1. The block principal pivoting method with the grouping acceleration for the nonnegativity-constrained least squares (NLS) problem constitutes a state-of-the-art algorithm for nonnegative matrix factorization (NMF) and nonnegative CANDECOMP/PARAFAC (NCP) decomposition. The success of this method is closely related with a critical observation that matrices involved in the NLS subproblems that arise in the NMF and NCP computation have long-and-thin and flat-and-wide structures (See Chapters 4 and 5). There have been previous studies on NMF and NCP based on other algorithms for solving the NLS subproblems; however, these important characteristics of the NLS subproblems have not been recognized or used to design fast algorithms. Our contribution exemplifies a general lesson in algorithm design for real-world problems: There are numerous algorithmic strategies for the NLS problems in general, but no single algorithm performs the best in all circumstances. By understanding the characteristics of the NMF and NCP problem, we were able to design the best algorithm for the NLS subproblems in the context of NMF and NCP computation.

2. The trade-offs between active-set-like methods for the NLS problems, that include the block principal pivoting method and the active-set method, are explained empirically and theoretically. The two methods are called active-set-like

methods in this thesis because both of them work under the following framework: (1) the active and passive workings sets are kept track of, (2) an unconstrained least squares problem is solved in each iteration, and (3) the workings sets are updated based on the solution. Whereas the active-set method exchanges typically only one variable between workings sets, the block principal pivoting method exchanges multiple variables per iteration, thereby requiring smaller number iterations until termination. In Chapters 4 and 5, this computational advantage is utilized to develop fast algorithms for NMF and NCP: In particular, Section 4.4.1 provides detailed analysis on the behaviour of the two methods. Despite the computational advantage, the block principal pivoting method requires a stronger condition that the matrix involved in the NLS problem is of full column rank. The full column rank condition is not required in the active-set method, as we proved in Chapter 7. This theoretical discovery fills the gap in the trade-offs of the two methods: Previously, it has not been formally shown that the active set method is capable of handing rank-deficient problems. Overall, this thesis elucidates the trade-offs of active-set-like methods through empirical demonstrations and a theoretical proof.

3. The trade-offs of active-set-like methods are further demonstrated in l_1-regularized linear regression. The l_1-regularized linear regression, also known as the Lasso, has received much attention in the recent decade due to its ability to promote sparsity. Existing active-set methods for Lasso include the least angle regression (LARS) and the feature-sign search algorithm, and these algorithms are similar to the active-set method for the NLS problem in that typically only one variable is exchanged among working sets, thereby showing limitations for large problems. In Chapter 7, the block principal pivoting method for the l_1-regularized linear regression is developed. Similarly to the NLS problems, the block principal pivoting method considerably accelerates existing active-set methods in

large problems. These results strengthen understanding on the trade-offs of active-set-like methods: In addition to the NLS problems, the block principal pivoting strategy accelerates the active-set method in l_1-regularized squares problems.

4. An effective and efficient method for incorporating group-structure prior information in NMF is proposed. In many applications to which NMF was shown successful, group structure prior information is commonly found: Features or data items are divided into groups defined by a certain kind of similarity. Motivated by the fact that the features or data items belonging to the same group are likely to share the same sparsity pattern in their low-rank factor representation, we proposed to adopt mixed-norm regularization to promote group sparsity in NMF. In Chapter 8, mixed-norm based formulation, efficient algorithms based on convex optimization methods, and application examples are presented. This contribution deepens the discussion on NMF in this thesis: Whereas Chapters 4 and 5 addressed computational challenges in standard NMF, Chapter 8 addressed challenges in incorporating prior information with a goal to improve upon standard NMF.

9.2 Future Directions

There are a number of interesting directions of further investigation on the problems or techniques discussed in thesis. We summarize those directions with comments based on experiences and observations from the work of this thesis.

1. A deeper investigation on various block coordinate descent (BCD) methods for NMF and NCP is worth attention. In Chapters 4 and 5, the block principal pivoting method was shown to perform the best among several methods based on matrix-block BCD framework, which is called the alternating nonnegative least squares (ANLS). On the other hand, the vector-block BCD method, called the

hierarchical alternating least squares (HALS) algorithm, also perform very efficiently, often outperforming the ANLS method with the block principal pivoting (ANLS-BPP) for NMF. In Section 4.4.3, we presented interesting trade-offs between ANLS-BPP and HALS depending on the sparsity of the latent factors. On the other hand, for higher order tensors, our experimental results in Chapter 5 indicate that HALS is clearly slower than ANLS-BPP for NCP. Research on these two successful methods, potentially with deeper understanding on the BCD methods in NMF and NCP computation, will be a valuable direction.

2. Further refinements on the block principal pivoting strategy is an interesting direction. The performance of block principal pivoting strategy would depend on how often the full exchange rule fails so that the backup rule has to be activated. In our extensive tests on NMF, NCP, and Lasso, the backup rule appearance was not observed, showing that the full exchange rule in practice work well for various problems. Still, further analysis on conditions under which the backup rule could appear and how much it affects the performance of overall algorithm will help understanding on this method. In addition, the current simple backup rules in Eq. (4.7) and Eq. (6.9) might not be the best ones, and the design of alternative backup rules will be useful.

3. Challenging questions regarding the group-sparsity regularization and its extensions remain. In the $l_{1,q}$-norm regularization method studied in Chapter 8, an interesting open question is to understand its effects depending on the choice of q. In this thesis, for computational convenience, only the $q = 2$ and the $q = \infty$ cases are addressed, but other choices that define l_q-norm for vectors can be generally considered. In fact, this issue on $l_{1,q}$-norm regularization goes beyond NMF literature and concerns wider audience who investigate group sparsity in supervised multi-task learning. In NMF, improving upon the group-sparsity

regularization studied in Chapter 8, more sophisticated techniques can be considered to enrich NMF and NCP methodologies. For example, algorithms and applications incorporating prior information with overlapping and hierarchical groups can be considered [49, 75].

REFERENCES

[1] Acar, E. and Yener, B., "Unsupervised multiway data analysis: A literature survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, pp. 6–20, Jan 2009.

[2] Bach, F., Jenatton, R., Mairal, J., and Obozinski, G., "Convex optimization with sparsity-inducing norms," in *Optimization for Machine Learning* (Sra, S., Nowozin, S., and Wright., S. J., eds.), MIT Press, 2011.

[3] Badea, L., "Extracting gene expression profiles common to colon and pancreatic adenocarcinoma using simultaneous nonnegative matrix factorization," in *Proceedings of the Pacific Symposium on Biocomputing 2008*, p. 267, 2008.

[4] Bader, B. W., Berry, M. W., and Browne, M., "Discussion tracking in Enron email using PARAFAC," in *Survey of Text Mining II: Clustering, Classification, and Retrieval*, pp. 147–163, Springer, 2008.

[5] Bader, B. W., Puretskiy, A. A., and Berry, M. W., "Scenario discovery using nonnegative tensor factorization," in *Proceedings of the Thirteenth Iberoamerican Congress on Pattern Recognition, CIARP 2008, Havana, Cuba*, pp. 791–805, 2008.

[6] Banerjee, O., El Ghaoui, L., and d'Aspremont, A., "Model selection through sparse maximum likelihood estimation for multivariate gaussian or binary data," *The Journal of Machine Learning Research*, vol. 9, pp. 485–516, 2008.

[7] Bengio, S., Pereira, F., Singer, Y., and Strelow, D., "Group sparse coding," in *Advances in Neural Information Processing Systems 22*, pp. 82–89, 2009.

[8] Berman, A. and Plemmons, R. J., *Nonnegative matrices in the mathematical sciences*, vol. 9. Society for Industrial and Applied Mathematics, 1994.

[9] Berry, M., Browne, M., Langville, A., Pauca, V., and Plemmons, R., "Algorithms and applications for approximate nonnegative matrix factorization," *Computational Statistics and Data Analysis*, vol. 52, no. 1, pp. 155–173, 2007.

[10] Bertsekas, D. P., *Nonlinear programming.* Athena Scientific, 1999.

[11] Bertsekas, D. P., Nedid, A., and Ozdaglar, A. E., *Convex Analysis and Optimization.* Belmont, Massachusetts: Athena Scientific, 2003.

[12] Biggs, M., Ghodsi, A., and Vavasis, S., "Nonnegative matrix factorization via rank-one downdate," in *Proceedings of the 25th international conference on Machine learning*, (New York, NY, USA), pp. 64–71, ACM, 2008.

[13] Borwein, J. M. and Lewis, A. S., *Convex analysis and nonlinear optimization: Theory and Examples.* Springer-Verlag, 2006.

[14] Boyd, S. and Vandenberghe, L., *Convex Optimization.* Cambridge University Press, 2004.

[15] Bro, R. and De Jong, S., "A fast non-negativity-constrained least squares algorithm," *Journal of Chemometrics*, vol. 11, pp. 393–401, 1997.

[16] Brunet, J., Tamayo, P., Golub, T., and Mesirov, J., "Metagenes and molecular pattern discovery using matrix factorization," *Proceedings of the National Academy of Sciences*, vol. 101, no. 12, pp. 4164–4169, 2004.

[17] Cai, D., He, X., and Han, J., "Document clustering using locality preserving indexing," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, pp. 1624–1637, December 2005.

[18] Candés, E. J., "Compressive sampling," in *Proceedings of the International Congress of Mathematicians*, 2006.

[19] Cantarella, J. and Piatek, M., "tsnnls: A solver for large sparse least squares problem with non-negative variables," *ArXiv Computer Science e-prints*, 2004.

[20] Carroll, J. D. and Chang, J.-J., "Analysis of individual differences in multidimensional scaling via an n-way generalization of "eckart-young" decomposition," *Psychometrika*, vol. 35, no. 3, pp. 283–319, 1970.

[21] Carroll, J. D., Soete, G. D., and Pruzansky, S., "Fitting of the latent class model via iteratively reweighted least squares CANDECOMP with non-negativity constraints," in *Multiway data analysis*, pp. 463–472, Amsterdam, The Netherlands: North-Holland Publishing Co., 1989.

[22] Chen, S. S., Donoho, D. L., and Saunders, M. A., "Atomic decomposition by basis pursuit," *SIAM Review*, vol. 43, no. 1, pp. 129–159, 2001.

[23] Chen, X., Pan, W., Kwok, J. T., and Carbonell, J. G., "Accelerated gradient method for multi-task sparse learning problem," in *Proceedings of the 2009 Ninth IEEE International Conference on Data Mining*, pp. 746–751, 2009.

[24] Cichocki, A. and Phan, A.-H., "Fast local algorithms for large scale non-negative matrix and tensor factorizations," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E92-A, no. 3, pp. 708–721, 2009.

[25] Cichocki, A., Zdunek, R., and Amari, S.-I., "Hierarchical ALS algorithms for nonnegative matrix and 3d tensor factorization," in *Lecture Notes in Computer Science*, vol. 4666, pp. 169–176, Springer, 2007.

[26] Cichocki, A., Zdunek, R., Phan, A. H., and Amari, S.-I., *Nonnegative matrix and tensor factorizations: applications to exploratory multi-way data analysis and blind source separation.* Wiley, 2009.

[27] Devarajan, K., "Nonnegative matrix factorization: An analytical and interpretive tool in computational biology," *PLoS Computational Biology*, vol. 4, no. 7, p. e1000029, 2008.

[28] Dhillon, I. and Sra, S., "Generalized nonnegative matrix approximations with bregman divergences," in *Advances in Neural Information Processing Systems 18*, pp. 283–290, MIT Press, 2006.

[29] Ding, C., He, X., and Simon, H. D., "On the equivalence of nonnegative matrix factorization and spectral clustering," in *Proceedings of the 2005 SIAM International Conference on Data Mining*, 2005.

[30] Donoho, D. L., "Compressed sensing," *IEEE Transactions on Information Theory*, vol. 52, no. 4, pp. 1289–1306, 2006.

[31] Duchi, J., Shalev-Shwartz, S., Singer, Y., and Chandra, T., "Efficient projections onto the l1-ball for learning in high dimensions," in *Proceedings of the 25th International Conference on Machine Learning*, pp. 272–279, 2008.

[32] Efron, B., Hastie, T., Johnstone, I., and Tibshirani, R., "Least angle regression," *Annals of Statistics*, vol. 32, no. 2, pp. 407–499, 2004.

[33] Févotte, C., Bertin, N., and Durrieu, J., "Nonnegative matrix factorization with the Itakura-Saito divergence: With application to music analysis," *Neural Computation*, vol. 21, no. 3, pp. 793–830, 2009.

[34] Figueiredo, M. A. T., Nowak, R. D., and Wright, S. J., "Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems," *IEEE Journal of Selected Topics in Signal Processing*, vol. 1, no. 4, pp. 586–597, 2007.

[35] FitzGerald, D., Cranitch, M., and Coyle, E., "Non-negative tensor factorisation for sound source separation," in *Proceedings of the Irish Signals and Systems Conference*, 2005.

[36] Friedlander, M. P. and Hatz, K., "Computing nonnegative tensor factorizations," *Computational Optimization and Applications*, vol. 23, pp. 631–647, March 2008.

[37] Friedman, J., Hastie, T., and Tibshirani, R., "Regularized paths for generalized linear models via coordinate descent." 2008.

[38] Friedman, J., Hastie, T., and Tibshirani, R., "Sparse inverse covariance estimation with the graphical lasso," *Biostatistics*, vol. 9, no. 3, p. 432, 2008.

[39] Gillis, N., *Nonnegative Matrix Factorization Complexity, Algorithms and Applications.* PhD thesis, Universit catholique de Louvain, 2011.

[40] Gillis, N. and Glineur, F., "Nonnegative factorization and the maximum edge biclique problem." CORE Discussion Paper 2008/64, Universite catholique de Louvain, 2008.

[41] Gillis, N. and Glineur, F., "Using underapproximations for sparse nonnegative matrix factorization," *Pattern Recognition*, vol. 43, no. 4, pp. 1676–1687, 2010.

[42] Golub, G. and Van Loan, C., *Matrix computations.* Johns Hopkins University Press Baltimore, MD, USA, 1996.

[43] Gonzalez, E. F. and Zhang, Y., "Accelerating the Lee-Seung algorithm for non-negative matrix factorization," tech. rep., Tech Report, Department of Computational and Applied Mathematics, Rice University, 2005.

[44] Grippo, L. and Sciandrone, M., "On the convergence of the block nonlinear gauss-seidel method under convex constraints," *Operations Research Letters*, vol. 26, no. 3, pp. 127–136, 2000.

[45] Harshman, R. A., "Foundations of the parafac procedure: Models and conditions for an "explanatory" multi-modal factor analysis," in *UCLA Working Papers in Phonetics*, vol. 16, pp. 1–84, 1970.

[46] Ho, N.-D., *Nonnegative Matrix Factorization Algorithms and Applications.* PhD thesis, Univ. Catholique de Louvain, 2008.

[47] Horn, R. A. and Johnson, C. R., *Matrix analysis.* Cambridge University Press, 1990.

[48] Hoyer, P. O., "Non-negative matrix factorization with sparseness constraints," *The Journal of Machine Learning Research*, vol. 5, pp. 1457–1469, 2004.

[49] Jenatton, R., Mairal, J., Obozinski, G., and Bach, F., "Proximal methods for sparse hierarchical dictionary learning," in *Proceedings of the 27th Annual International Conference on Machine Learning*, 2010.

[50] Jenatton, R., Obozinski, G., and Bach, F., "Structured sparse principal component analysis," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS), JMLR: W&CP*, vol. 9, pp. 366–373, 2010.

[51] Jia, Y., Salzmann, M., and Darrell, T., "Factorized latent spaces with structured sparsity," in *Advances in Neural Information Processing Systems 23*, 2010.

[52] Júdice, J. J. and Pires, F. M., "Basic-set algorithm for a generalized linear complementarity problem," *Journal of Optimization Theory and Applications*, vol. 74, no. 3, pp. 391–411, 1992.

[53] Júdice, J. J. and Pires, F. M., "A block principal pivoting algorithm for large-scale strictly monotone linear complementarity problems," *Computers and Operations Research*, vol. 21, no. 5, pp. 587–596, 1994.

[54] Kim, D., Sra, S., and Dhillon, I. S., "Fast Newton-type methods for the least squares nonnegative matrix approximation problem," in *Proceedings of the 2007 SIAM International Conference on Data Mining*, 2007.

[55] Kim, H. and Park, H., "Sparse non-negative matrix factorizations via alternating non-negativity-constrained least squares for microarray data analysis," *Bioinformatics*, vol. 23, no. 12, pp. 1495–1502, 2007.

[56] Kim, H. and Park, H., "Nonnegative matrix factorization based on alternating nonnegativity constrained least squares and active set method," *SIAM Journal on Matrix Analysis and Applications*, vol. 30, no. 2, pp. 713–730, 2008.

[57] Kim, H., Park, H., and Eldén, L., "Non-negative tensor factorization based on alternating large-scale non-negativity-constrained least squares," in *Proceedings of IEEE 7th International Conference on Bioinformatics and Bioengineering (BIBE07)*, vol. 2, pp. 1147–1151, 2007.

[58] Kim, J. and Park, H., "Sparse nonnegative matrix factorization for clustering," tech. rep., Georgia Institute of Technology Technical Report GT-CSE-08-01, 2008.

[59] Kim, S.-J., Koh, K., Lustig, M., Boyd, S., and Gorinevsky, D., "An interior-point method for large-scale l1-regularized least squares," *IEEE Journal of Selected Topics in Signal Processing*, vol. 4, no. 1, pp. 606–617, 2007.

[60] Kim, S. and Xing, E. P., "Tree-guided group Lasso for multi-task regression with structured sparsity," in *Proceedings of the 27th International Conference on Machine Learning*, 2010.

[61] Kim, Y., Kim, J., and Kim, Y., "Blockwise sparse regression," *Statistica Sinica*, vol. 16, no. 2, pp. 375–390, 2006.

[62] Kolda, T. G. and Bader, B. W., "Tensor decompositions and applications," *SIAM review*, vol. 51, no. 3, pp. 455–500, 2009.

[63] Lawson, C. L. and Hanson, R. J., *Solving Least Squares Problems*. Prentice Hall, 1974.

[64] Lee, D. D. and Seung, H. S., "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, pp. 788–791, 1999.

[65] Lee, D. D. and Seung, H. S., "Algorithms for non-negative matrix factorization," in *Advances in Neural Information Processing Systems 13*, pp. 556–562, MIT Press, 2001.

[66] Lee, H., Battle, A., Raina, R., and Ng, A. Y., "Efficient sparse coding algorithms," in *Advances in Neural Information Processing Systems 19*, MIT Press, 2007.

[67] Lee, S.-I., Lee, H., Abbeel, P., and Ng, A. Y., "Efficient l1 regularized logistic regression," in *Proceedings of the Twenty-first National Conference on Artificial Intelligence (AAAI-06)*, pp. 1–9, 2006.

[68] Li, S. Z., Hou, X., Zhang, H., and Cheng, Q., "Learning spatially localized, parts-based representation," in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, 2001.

[69] Li, T., Sindhwani, V., Ding, C., and Zhang, Y., "Bridging domains with words: Opinion analysis with matrix tri-factorizations," in *Proceedings of the 2010 SIAM International Conference on Data Mining*, 2010.

[70] Lim, L.-H. and Comon, P., "Nonnegative approximations of nonnegative tensors," *Journal of Chemometrics*, vol. 23, no. 7-8, pp. 432–441, 2009.

[71] Lin, C.-J., "Projected gradient methods for nonnegative matrix factorization," *Neural Computation*, vol. 19, no. 10, pp. 2756–2779, 2007.

[72] Lin, C.-J. and Moré, J. J., "Newton's method for large bound-constrained optimization problems," *SIAM Journal on Optimization*, vol. 9, no. 4, pp. 1100–1127, 1999.

[73] Lin, C., "On the convergence of multiplicative update algorithms for nonnegative matrix factorization," *IEEE Transactions on Neural Networks*, vol. 18, no. 6, pp. 1589–1596, 2007.

[74] Liu, H., Palatucci, M., and Zhang, J., "Blockwise coordinate descent procedures for the multi-task lasso, with applications to neural semantic basis discovery," in *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 649–656, 2009.

[75] Liu, J. and Ye, J., "Moreau-yosida regularization for grouped tree structure learning," in *Advances in Neural Information Processing Systems 23*, pp. 1459–1467, 2010.

[76] Mairal, J., Bach, F., Ponce, J., and Sapiro, G., "Online dictionary learning for sparse coding," in *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 689–696, 2009.

[77] Manning, C. D., Raghavan, P., and Schutze, H., *Introduction to Information Retrieval.* Cambridge University Press, 2008.

[78] Masaeli, M., Yan, Y., Cui, Y., Fung, G., and Dy, J. G., "Convex principal feature selection," in *Proceedings of the 2010 SIAM International Conference on Data Mining*, pp. 619–628, 2010.

[79] Meier, L., Van De Geer, S., and Bühlmann, P., "The group lasso for logistic regression," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 70, no. 1, pp. 53–71, 2008.

[80] Murty, K., *Linear complementarity, linear and nonlinear programming.* Heldermann Verlag, 1988.

[81] Nesterov, Y., *Introductory lectures on convex optimization: A basic course.* Kluwer Academic Publishers, 2004.

[82] Nocedal, J. and Wright, S. J., *Numerical Optimization.* Springer, 1999.

[83] Paatero, P., "A weighted non-negative least squares algorithm for three-way PARAFAC factor analysis," *Chemometrics and Intelligent Laboratory Systems*, vol. 38, no. 2, pp. 223–242, 1997.

[84] Paatero, P., "The multilinear engine: A table-driven, least squares program for solving multilinear problems, including the n-way parallel factor analysis model," *Journal of Computational and Graphical Statistics*, vol. 8, pp. 854–888, Dec. 1999.

[85] Paatero, P. and Tapper, U., "Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values," *Environmetrics*, vol. 5, no. 1, pp. 111–126, 1994.

[86] Pardalos, P. M. and Kovoor, N., "An algorithm for a singly constrained class of quadratic programs subject to upper and lower bounds," *Mathematical Programming*, vol. 46, no. 1-3, pp. 321–328, 1990.

[87] Pauca, V. P., Piper, J., and Plemmons, R. J., "Nonnegative matrix factorization for spectral data analysis," *Linear Algebra and Its Applications*, vol. 416, no. 1, pp. 29–47, 2006.

[88] Pauca, V. P., Shahnaz, F., Berry, M. W., and Plemmons, R. J., "Text mining using non-negative matrix factorizations," in *Proceedings of the 2004 SIAM International Conference on Data Mining*, 2004.

[89] Portugal, L. F., Judice, J. J., and Vicente, L. N., "A comparison of block pivoting and interior-point algorithms for linear least squares problems with nonnegative variables," *Mathematics of Computation*, vol. 63, no. 208, pp. 625–643, 1994.

[90] Quattoni, A., Carreras, X., Collins, M., and Darrell, T., "An efficient projection for l1,infty regularization," in *Proceedings of the 26th International Conference on Machine Learning*, pp. 857–864, 2009.

[91] Rockafellar, R. T., *Convex analysis.* Princeton University Press, 1972.

[92] Shahnaz, F., Berry, M., Pauca, V., and Plemmons, R., "Document clustering using nonnegative matrix factorization," *Information Processing and Management*, vol. 42, no. 2, pp. 373–386, 2006.

[93] Shashua, A. and Hazan, T., "Non-negative tensor factorization with applications to statistics and computer vision," in *Proceedings of the 22nd international conference on Machine learning*, (New York, NY, USA), pp. 792–799, ACM, 2005.

[94] Singh, A. P. and Gordon, G. J., "Relational learning via collective matrix factorization," in *Proceeding of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 650–658, 2008.

[95] Sjöstrand, K., "Matlab implementation of lasso, lars, the elastic net, and spca," 2005. Version 2.0.

[96] Tibshirani, R., "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996.

[97] Trefethen, L. N. and Bau, D., *Numerical linear algebra.* Society for Industrial and Applied Mathematics, 1997.

[98] Tseng, P., "On accelerated proximal gradient methods for convex-concave optimization," *submitted to SIAM Journal on Optimization*, 2008.

[99] Van Benthem, M. H. and Keenan, M. R., "Fast algorithm for the solution of large-scale non-negativity-constrained least squares problems," *Journal of Chemometrics*, vol. 18, pp. 441–450, 2004.

[100] Vavasis, S. A., "On the complexity of nonnegative matrix factorization," *SIAM Journal on Optimization*, vol. 20, no. 3, pp. 1364–1377, 2009.

[101] Wang, F., Li, T., and Zhang, C., "Semi-supervised clustering via matrix factorization," in *Proceedings of the 2008 SIAM International Conference on Data Mining*, 2008.

[102] Welling, M. and Weber, M., "Positive tensor factorization," *Pattern Recogn. Lett.*, vol. 22, no. 12, pp. 1255–1261, 2001.

[103] Xu, W., Liu, X., and Gong, Y., "Document clustering based on nonnegative matrix factorization," in *Proceedings of the 26th annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*, pp. 267–273, 2003.

[104] Yuan, M. and Lin, Y., "Model selection and estimation in regression with grouped variables," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 68, no. 1, pp. 49–67, 2006.

[105] Zou, H. and Hastie, T., "Regularization and variable selection via the elastic net," *Journal of the Royal Statistical Society, Series B*, vol. 67, no. 2, pp. 301–320, 2005.

[106] Zou, H., Hastie, T., and Tibshirani, R., "Sparse principal component analysis," *Journal of Computational and Graphical Statistics*, vol. 15, no. 2, pp. 265–286, 2006.