Constrained Matrix and Tensor Factorization: Theory, Algorithms, and Applications

A DISSERTATION
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Kejun Huang

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
Doctor of Philosophy

Nikos Sidiropoulos

August, 2016

Acknowledgments

First and foremost, I would like to thank my advisor, Professor Nikos Sidiropoulos. I cannot imagine what a person I would become had I not been working with him for the past five years. I feel so lucky to be his first Ph.D. student since he moved back to Minneapolis. His way of thinking problems, solving problems, and presenting works has influenced me in almost every way. I cherish all the time and effort he spent on me in making me a qualified researcher.

I would also like to express my sincere thank to Dr. Xiao Fu. Together we have worked out some of my proudest works. Interestingly, we are starting to realize how important these contributions are to the field of signal processing and machine learning, much more so than when we first worked them out. I am fortunate to have him as a more senior collaborator and a good friend.

I am very grateful to Professor Georgios Giannakis, Professor George Karypis, Professor Zhi-Quan Luo, and Professor Yousef Saad for being the committee members. Especially, I want to thank Professor Giannakis for recommending me to Professor Sidiropoulos, and for being so kind to me during my time at the Digital Technology Center. Also, many thanks to Professor Zhi-Quan Luo, with whom I have taken the most number of courses, and they all played important roles in my research afterwards.

I have had the privilege to work with many collaborators, on a wide varieties of research topics, ranging from natural language processing to phase retrieval. My sincere thanks to Professor Rasmus Bro, Professor Yonina Eldar, Professor Christos Faloutsos, Dr. Matt Gardner, Professor Mingyi Hong, Professor Athanasios Liavas, Professor Wing-Kin Ma, Professor Tom Mitchell, Dr. Evangelos Papalexakis, Dr. Ananthram Swami, and many more. Special thanks to the collaboration with the group of Professor Faloutsos and Professor Mitchell from the Carnegie Mellon University, and the group

of Professor Eldar from the Technion – Isreal Institute of Technology.

I have had a great time with my group-mates Balasubramanian Gopalakrishnan, Charilaos Kanatsoulis, Nikos Kargas, Aritra Konar, Omar Mehanna, Artem Mosesov, Cheng Qian, Niranjay Ravindran, John Tranter, Efthimios Tsakonas, Bo Yang, and Ahmed Zamzam. Many thanks to the friends I made in Minneapolis: Jie Kang, Hai Xu, Kejian Wu, Yu Zhang, Shaden Smith, and many many more.

Finally, I would like to express my deepest love to my family. My parents, Keping Huang and Jian Wang, have given me unlimited support in my pursuit of the doctoral degree. I hope to make them proud by becoming a doctor now, and I will keep on working hard in the future. And, most importantly, to my beloved wife, Shu Xiao, who has been so supportive and considerate. Life is wonderful with you.

Dedication

To Shu Xiao, Keping Huang, and Jian Wang

Abstract

This dissertation studies constrained matrix and tensor factorization problems which appear in the context of estimating factor analysis models used in signal processing and machine learning. Factor analysis dates back to the celebrated principal component analysis (PCA), which provides the optimal low rank approximation to matrix data. PCA is a key dimensionality reduction technique, however it cannot be used for estimating the underlying generative matrix factors, which are generally non-orthogonal. On the other hand, it has been observed that imposing simple constraints on the latent factors, for example non-negativity (which in a way opposes orthogonality) can make these factors essentially unique – although theoretical understanding of this phenomenon was very limited prior to this dissertation. Moving from two-way to higher-way *tensor* data, the so-called parallel factor analysis (PARAFAC) model can provide essentially unique factors under mild conditions, but it is also widely accepted that if correct priors are imposed on the latent factors, the estimation performance can be greatly enhanced.

One of the main contributions of this dissertation is the development of a simple *sufficiently scattered* condition that turns out to underpin a great variety of factor analysis models – including but not limited to *non-negative matrix factorization*(NMF), and other models relying on volume minimization, which are nowadays pervasive in signal processing, machine learning, and geoscience and remote sensing – where it helps resolve a popular conjecture in multispectral imaging known as *Craig's belief*, that sources can be blindly identified via "minimum volume transformation" under certain conditions. In these and other applications, the *sufficiently scattered* condition provides the best identifiability results known to this date, and it has significant potential for further applications.

Besides theoretical results on the essential uniqueness for factor analysis models under non-negativity constraints, a general algorithmic framework is proposed, which seamlessly and effortlessly incorporates many common types of constraints imposed onto the latent factors. The new framework is a hybrid between alternating optimization (AO) and the alternating direction method of multipliers (ADMM): each matrix factor is updated in turn, using ADMM, hence the name AO-ADMM. This combination

can naturally accommodate a great variety of constraints on the factor matrices, and almost all possible loss measures for the fitting. Computation caching and warm start strategies are used to ensure that each update is evaluated efficiently, while the outer AO framework exploits recent developments in block coordinate descent (BCD)-type methods which help ensure that every limit point is a stationary point, as well as faster and more robust convergence in practice. Extensive simulations and experiments with real data are used to showcase the effectiveness and broad applicability of the proposed framework.

In addition to establishing essential uniqueness and providing effective computational algorithms for various matrix and tensor factorization models, we further study how well these models estimate the correct latent factors in noise. Towards this end, pertinent Cramér-Rao bounds (CRB) for constrained matrix and tensor factorization are derived. This turns out being a nontrivial task, mainly due to the presence of constraints, trivial ambiguities, and computational challenges as well. In particular, the Fisher Information Matrix (FIM) is always singular for the models considered (even without constraints) – but modern CRB analysis shows that taking the pseudo-inverse still provides a valid (albeit potentially loose) bound. For big data analytics, however, the challenge is how to efficiently compute this pseudo-inverse. Towards this end we succeeded in identifying the null space of the FIM for several special cases, leading to very efficient algorithms for computing the CRB. Equipped with these results, we test-drive the performance of various algorithms and benchmark them against the CRB. Interestingly, our algorithms can approach this (optimistic) CRB under certain conditions.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Factor analysis plays an important role in latent parameter estimation and blind source separation in signal processing, dimensionality reduction and clustering in machine learning, and numerous other applications in diverse disciplines, such as chemistry and psychology. What lies underneath is the belief, and often the case the fact, that our observation can be explained by a simplified model with much fewer latent components. For matrix data, the classical *principal component analysis* has paved the way for optimal dimensionality reduction by imposing, without loss of generality, orthogonality constraints.

In this dissertation we also consider *tensors*, which are data arrays indexed by three or more indices—a generalization of matrices, which are indexed by two indices. Although it looks like a simple generalization, there are some surprising differences when we move beyond two indices. Within the factor analysis framework, the biggest difference lies in the fact that, while low rank approximation of a matrix can be solved in polynomial time, that of a tensor is in general NP-hard; on the other hand, latent factors of a matrix is non-unique, while those of a tensor can be uniquely identified (up to trivial ambiguities) under very mild conditions.

The focus of this dissertation is to study what happens if we impose non-paramatric constraints onto the latent factors, either of a matrix or of a tensor. Starting from the seminal work of Lee and Seung [1], people have observed that this simple constraint on the latent factors usually helps resolving the non-uniqueness issue inherent in the matrix factorization models [2]. We will roam around different possibilities of priors

imposed onto the latent factors, and see how they enhance identifiability that does not appear in their unconstrained counterparts. Pertinent algorithmic issues, performance analysis tools, and selected applications will also be discussed in detail.

Notation. A scalar is denoted by an italic letter x, a vector is denoted by a bold lowercase letter $\boldsymbol{x}$, a matrix bold uppercase letter $\boldsymbol{X}$; the i-th column of $\boldsymbol{X}$ is denoted by $\boldsymbol{x}_i$ with a subscript, and the j-th row of $\boldsymbol{X}$, transposed to be a column vector, is denoted by $\boldsymbol{x}[j]$ with a brace.

We denote the (approximate) factorization of a matrix $\boldsymbol{Y} \approx \boldsymbol{W}\boldsymbol{H}^T$, where $\boldsymbol{Y}$ is $m \times n$, $\boldsymbol{W}$ is $m \times k$, and $\boldsymbol{H}$ is $n \times k$, with $k \leq m, n$, and in most cases much smaller. Note that adding constraints on $\boldsymbol{W}$ and $\boldsymbol{H}$ may turn the solution from easy to find (via SVD) but non-unique, to NP-hard to find but essentially unique. It has been shown that simple constraints like non-negativity and sparsity can make the factors identifiable, but at the same time, computing the optimal solution becomes NP-hard—see [3] and references therein.

An N-way array of dimension $n_1 \times n_2 \times ... \times n_N$, with $N \geq 3$, is denoted with an underscore, e.g., $\underline{\boldsymbol{Y}}$. In what follows, we focus on the so-called canonical polyadic decomposition (CPD), also known as parallel factor analysis (PARAFAC) model or canonical decomposition (CANDECOMP), which is essentially unique under mild conditions [4], but constraints certainly help enhance estimation performance, and even identifiability. The factorization is denoted as $\underline{\boldsymbol{Y}} \approx [\boldsymbol{H}_d]_{d=1}^N$, which is a concise way of representing the model

$$\underline{\boldsymbol{Y}}(i_1, ..., i_N) \approx \sum_{j=1}^{k} \prod_{d=1}^{N} \boldsymbol{H}_d(i_d, j), \quad \forall i_1, ..., i_N.$$

Each matrix $\boldsymbol{H}_d$ is of size $n_d \times k$, corresponding to the factor of the d-th mode.

Roadmap. The rest of this chapter is dedicated to basics of multi-linear algebra. We start by introducing the matrix Kronecker product, Khatri-Rao product, and Hadamard product, together with a special class of permutation matrices called the commutation matrices. Then we introduce the notion of tensor matricization and tensor vectorization, and their relationship with the aforementioned matrix operators. With the increasing interest in tensor data processing, there exist many tutorials on this topic, for example,

[5–7]. Here we briefly review some basic multi-linear operations that will be useful for the purposes of this paper, and refer the readers to those tutorials and the references therein for a more comprehensive introduction.

Chapter 2 studies the uniqueness of non-negative matrix factorization (NMF), which is for a long time conjectured to be unique under mild conditions. We propose the so-called sufficiently scattered condition that confirms this conjecture. Although checking if the sufficiently scattered condition is satisfied is NP-hard, we provide useful insights and heuristics indicating that this condition is indeed easy to satisfy in practice. Main contents written in this chapter can also be found in [3, 8].

In Chapter 3, we relax the non-negativity constraint in one of the matrix factors in the NMF model, but instead put a "minimum-volume" criterion on the unconstrained factor. This model also dates back a long time—in the 90's, Craig, in the context of remote sensing, conjectured that this "minimum-volume" criterion is able to identify sources blindly under mild conditions. Interestingly, so far the best identifiability condition is again sufficiently scattered, and the proof is given for a number of different "volume-minimization" formulation. This chapter is concluded with an interesting application to the neuro-functional connectivity discovery. Main contents written in this chapter can also be found in [9–12].

Chapter 4 is dedicated to a flexible and efficient algorithmic framework for constrained matrix and tensor factorization. The new framework is a hybrid between alternating optimization (AO) and the alternating direction method of multipliers (ADMM): each matrix factor is updated in turn, using ADMM, hence the name AO-ADMM. This combination can naturally accommodate a great variety of constraints on the factor matrices, and almost all possible loss measures for the fitting. Computation caching and warm start strategies are used to ensure that each update is evaluated efficiently, while the outer AO framework exploits recent developments in block coordinate descent (BCD)-type methods which help ensure that every limit point is a stationary point, as well as faster and more robust convergence in practice. Three special cases are studied in detail: non-negative matrix/tensor factorization, constrained matrix/tensor completion, and dictionary learning. Extensive simulations and experiments with real data are used to showcase the effectiveness and broad applicability of the proposed framework. Main contents written in this chapter can also be found in [13, 14].

In Chapter 5, we derive the pertinent Cramér-Rao bound for matrix and tensor factorization models. Since the focus of this dissertation is constrained factor analysis, we begin with a concise tutorial on recent developments on the subject of Cramér-Rao bound, with a focus on how constraints and singularity should be handled. Then we derive the Cramér-Rao bound for both matrix and CP factorization models, discuss in detail the rank deficiency and computational issues that we encounter in the derivations. We present some of the well-known NMF algorithms benchmarked with the derived CRB to showcase their effectiveness in estimating the true latent factors. Main contents written in this chapter can also be found in [15, 16].

Finally, we study an interesting variant of NMF by restraining symmetry to the two latent factors in Chapter 6. The former topics are revisited on this specific subject, namely uniqueness, algorithm, and CRB, followed by numerical simulations and an application to a co-authorship clustering task, showing significant performance improvement comparing to the state-of-the-arts. Main contents written in this chapter can also be found in [3, 15, 17].

1.1 Kronecker, Khatri-Rao, and Hadamard Product

The **Kronecker product** [18] of a $m \times n$ matrix $\boldsymbol{S}$ and $p \times q$ matrix $\boldsymbol{T}$, denoted as $\boldsymbol{S} \otimes \boldsymbol{T}$, is the $mp \times nq$ block matrix

$$\boldsymbol{S} \otimes \boldsymbol{T} = \begin{bmatrix} s_{11}\boldsymbol{T} & \cdots & s_{1n}\boldsymbol{T} \\ \vdots & \ddots & \vdots \\ s_{m1}\boldsymbol{T} & \cdots & s_{mn}\boldsymbol{T} \end{bmatrix}.$$

There are some very interesting properties associated with the Kronecker product, and we list some of them here that are going to be useful in the sequel.

$$\begin{aligned} (\boldsymbol{S} \otimes \boldsymbol{T}) \otimes \boldsymbol{R} &= \boldsymbol{S} \otimes (\boldsymbol{T} \otimes \boldsymbol{R}), \\ (\boldsymbol{S} \otimes \boldsymbol{T})(\boldsymbol{X} \otimes \boldsymbol{Y}) &= \boldsymbol{S}\boldsymbol{X} \otimes \boldsymbol{T}\boldsymbol{Y}, \\ (\boldsymbol{S} \otimes \boldsymbol{T})^{\dagger} &= \boldsymbol{S}^{\dagger} \otimes \boldsymbol{T}^{\dagger}, \\ (\boldsymbol{S} \otimes \boldsymbol{T})^{T} &= \boldsymbol{S}^{T} \otimes \boldsymbol{T}^{T}, \\ \mathsf{vec}\left(\boldsymbol{S}\boldsymbol{X}\boldsymbol{T}^{T}\right) &= (\boldsymbol{T} \otimes \boldsymbol{S})\mathsf{vec}\left(\boldsymbol{X}\right). \end{aligned}$$

The **Khatri-Rao product** of matrices $\boldsymbol{S}$ and $\boldsymbol{T}$ having the same number of columns, denoted as $\boldsymbol{S} \odot \boldsymbol{T}$, is defined as the column-wise Kronecker product of $\boldsymbol{S}$ and $\boldsymbol{T}$. More explicitly, if $\boldsymbol{S}$ and $\boldsymbol{T}$ both have k columns, then

$$\boldsymbol{S} \odot \boldsymbol{T} = \begin{bmatrix} \boldsymbol{s}_1 \otimes \boldsymbol{t}_1 & \cdots & \boldsymbol{s}_k \otimes \boldsymbol{t}_k \end{bmatrix}.$$

A few properties of the Khatri-Rao product appear similarly with the Kronecker product, although not all of them are inherited. Namely,

$$\begin{aligned} (\boldsymbol{S} \odot \boldsymbol{T}) \odot \boldsymbol{R} &= \boldsymbol{S} \odot (\boldsymbol{T} \odot \boldsymbol{R}), \\ (\boldsymbol{S} \otimes \boldsymbol{T})(\boldsymbol{X} \odot \boldsymbol{Y}) &= \boldsymbol{S}\boldsymbol{X} \odot \boldsymbol{T}\boldsymbol{Y}, \\ (\boldsymbol{S} \odot \boldsymbol{T})(\boldsymbol{X} \odot \boldsymbol{Y}) &= \boldsymbol{S}\boldsymbol{X} \circledast \boldsymbol{T}\boldsymbol{Y}, \\ \mathsf{vec}\left(\boldsymbol{S}\mathrm{Diag}\left\{\boldsymbol{x}\right\}\boldsymbol{T}^T\right) &= (\boldsymbol{T} \odot \boldsymbol{S})\boldsymbol{x}, \end{aligned}$$

where $\circledast$ denotes the element-wise (Hadamard) product of two matrices with the same size.

Both Kronecker product and Khatri-Rao product are associative, as we have shown, but neither of them are commutative. There are, however, operations that make them appear commutative. A commutation matrix [19] associated with $m \times n$ matrices, denoted as $\boldsymbol{C}_{m,n}$, is a permutation matrix of size $mn \times mn$, with the corresponding permutation defined as follows. Starting with the integer sequence $1, 2, ..., mn$, we first reshape them into a $m \times n$ matrix

$$\boldsymbol{\Pi} = \begin{bmatrix} 1 & m+1 & \cdots & (n-1)m+1 \\ \vdots & \vdots & \ddots & \vdots \\ m & 2m & \cdots & mn \end{bmatrix}.$$

Then the corresponding permutation can be obtained by $\mathsf{vec}\left(\boldsymbol{\Pi}^T\right)$. The direct consequence is that for any $m \times n$ matrix $\boldsymbol{X}$, we have that

$$\boldsymbol{C}_{m,n}\mathsf{vec}\left(\boldsymbol{X}\right) = \mathsf{vec}\left(\boldsymbol{X}^T\right).$$

However, the more interesting properties of the commutation matrices are how they interact with the Kronecker and Khatri-Rao products:

$$\begin{aligned} \boldsymbol{C}_{p,m}(\boldsymbol{S} \otimes \boldsymbol{T}) &= (\boldsymbol{T} \otimes \boldsymbol{S})\boldsymbol{C}_{q,n}, \\ \boldsymbol{C}_{p,m}(\boldsymbol{S} \odot \boldsymbol{T}) &= \boldsymbol{T} \odot \boldsymbol{S}. \end{aligned}$$

Moreover, we also have that

$$\boldsymbol{C}_{n,m} = \boldsymbol{C}_{m,n}^T = \boldsymbol{C}_{m,n}^{-1},$$
$$\boldsymbol{C}_{mp,n}\boldsymbol{C}_{mn,p} = \boldsymbol{C}_{m,np}.$$

1.2 Tensor Unfolding: Matricization and Vectorization

The **mode-d matricization**, also known as mode-d matrix unfolding, of $\underline{\boldsymbol{Y}}$, denoted as $\boldsymbol{Y}_{(d)}$, is a matrix of size $\prod_{j=1,j\neq d}^{N} n_j \times n_d$. Each row of $\boldsymbol{Y}_{(d)}$ is a vector obtained by fixing all the indices of $\underline{\boldsymbol{Y}}$ except the d-th one, and the matrix is formed by stacking these row vectors by traversing the rest of the indices from 1 to N. As an example, for $N = 3$, the three matricizations are

$$\boldsymbol{Y}_{(1)} = \begin{bmatrix} \underline{\boldsymbol{Y}}(:,1,:)^T \\ \vdots \\ \underline{\boldsymbol{Y}}(:,n_2,:)^T \end{bmatrix}, \boldsymbol{Y}_{(2)} = \begin{bmatrix} \underline{\boldsymbol{Y}}(1,:,:)^T \\ \vdots \\ \underline{\boldsymbol{Y}}(n_1,:,:)^T \end{bmatrix}, \boldsymbol{Y}_{(3)} = \begin{bmatrix} \underline{\boldsymbol{Y}}(1,:,:) \\ \vdots \\ \underline{\boldsymbol{Y}}(n_1,:,:) \end{bmatrix}.$$

where $\underline{\boldsymbol{Y}}(i,:,:)$, $\underline{\boldsymbol{Y}}(:,i,:)$, $\underline{\boldsymbol{Y}}(:,:,i)$ are the i-th matrix slabs of the three-way tensor $\underline{\boldsymbol{Y}}$, of size $n_2 \times n_3$, $n_1 \times n_3$, $n_1 \times n_2$, respectively. Notice that, though essentially in the same spirit, this definition of mode-d matricization may be different from other expressions that have appeared in the literature, but we adopt this one for ease of our use.

The Khatri-Rao product is associative (although not commutative). We therefore generalize the operator $\odot$ to accept more than two arguments in the following way

$$\bigodot_{\substack{j=1 \\ j\neq d}}^{N} \boldsymbol{H}_i = \boldsymbol{H}_N \odot \cdots \odot \boldsymbol{H}_{d+1} \odot \boldsymbol{H}_{d-1} \odot \cdots \odot \boldsymbol{H}_1.$$

With the help of this notation, if $\underline{\boldsymbol{Y}}$ admits an exact PARAFAC model $\underline{\boldsymbol{Y}} = [\![\boldsymbol{H}_d]\!]_{d=1}^N$, then it can be expressed in matricized form as

$$\boldsymbol{Y}_{(d)} = \left(\bigodot_{\substack{j=1 \\ j\neq d}}^{N} \boldsymbol{H}_j \right) \boldsymbol{H}_d^T.$$

We similarly generalize the Hadamard product to accept more than two argument, and it is easy to see that

$$\left(\bigodot_{\substack{j=1 \\ j\neq d}}^{N} \boldsymbol{H}_j \right)^T \left(\bigodot_{\substack{j=1 \\ j\neq d}}^{N} \boldsymbol{H}_j \right) = \bigcircledast_{\substack{j=1 \\ j\neq d}}^{N} \boldsymbol{H}_j^T \boldsymbol{H}_j.$$

The **vectorization** of $\underline{\boldsymbol{Y}}$, denoted as $\mathsf{vec}\left(\underline{\boldsymbol{Y}}\right)$, is defined similarly as the vectorization of a matrix: the vector is formed by stacking all the elements of $\underline{\boldsymbol{Y}}$ by traversing the indices from 1 to N. If $\underline{\boldsymbol{Y}}$ admits an exact CP model $\underline{\boldsymbol{Y}} = [\![\boldsymbol{H}_d]\!]_{d=1}^{N}$, then it can be expressed in vectorized form as

$$\mathsf{vec}\left(\underline{\boldsymbol{Y}}\right) = (\boldsymbol{H}_N \odot \cdots \odot \boldsymbol{H}_1)\mathbf{1}.$$

Unlike tensor matricizations, the convention is that there is only one version of tensor vectorization. We can also see that

$$\mathsf{vec}\left(\underline{\boldsymbol{Y}}\right) = \mathsf{vec}\left(\boldsymbol{Y}_{(N)}^T\right).$$

As for the other tensor matricizations, they are related to $\mathsf{vec}\left(\underline{\boldsymbol{Y}}\right)$ through the commutation matrices

$$\mathsf{vec}\left(\underline{\boldsymbol{Y}}\right) = \boldsymbol{C}_{n_{d-1}\ldots n_1, n_N \ldots n_d}\mathsf{vec}\left(\boldsymbol{Y}_{(d)}^T\right).$$

Chapter 2

Uniqueness of Non-negative Matrix Factorization

Non-negative matrix factorization (NMF) is the problem of (approximately) factoring $\boldsymbol{Y} = \boldsymbol{W}\boldsymbol{H}^T$, where $\boldsymbol{Y}$ is $m \times n$, $\boldsymbol{W}$ is $m \times k$, with $k < \min(m, n)$, $\boldsymbol{H}$ is $n \times k$, $\boldsymbol{W} \geq 0$, $\boldsymbol{H} \geq 0$ (inequalities interpreted element-wise). The smallest possible K for which such decomposition is possible is the *non-negative rank* of $\boldsymbol{Y}$. Due to the non-negativity constraints, the non-negative rank can be higher than the usual matrix rank over the real field.

NMF has been studied for more than 30 years [20–22], originally known as *non-negative rank factorization* or *positive matrix factorization*. Non-negative matrices have many interesting properties and a long history in science and engineering [23]. Lee and Seung [1] popularized NMF when they discovered that it tends to decompose images of visual objects in meaningful parts - i.e., NMF "is able to learn the parts of objects". NMF quickly found numerous other applications in diverse disciplines - see [24] and references therein. Unfortunately, it was recently shown that (asymmetric) NMF is NP-hard [25]; yet sustained interest in NMF has produced many good algorithms.

NMF has been such a success story across disciplines because non-negativity is a valid constraint in so many applications, *and* NMF often provides meaningful / interpretable results, and sometimes even 'correct' results—that is, it yields the true latent factors $\boldsymbol{W}$, $\boldsymbol{H}$. As an example, matrix $\boldsymbol{Y}$ could represent an $m \times n$ keyword by document incidence

matrix, wherein entry $y_{i,j}$ is the number of occurences of keyword i in document j. The k-dimensional reduction $\boldsymbol{Y} = \boldsymbol{W}\boldsymbol{H}^T$ can be interpreted as follows. The columns of $\boldsymbol{W}$ represent k document prototypes as consisting of keywords in specific proportions. The rows of $\boldsymbol{H}$ represent each document as a weighted combination of these prototypes. In a clustering scenario, the columns of $\boldsymbol{W}$ are the cluster centroids, and the rows of $\boldsymbol{H}$ are the (soft) cluster membership indicators. A specific application of clustering via NMF is [26], where NMF was applied to an article by journal matrix, resulting in a soft clustering of articles by topic. See also Xu *et al.* [27] for an application of NMF to document clustering. Another interpretation of $\boldsymbol{Y} = \boldsymbol{W}\boldsymbol{H}^T$ is that the i-th row of $\boldsymbol{W}$ is a reduced $1 \times k$ representation of the i-th 'keyword' or 'concept' in latent k-dimensional space, and likewise the j-th rows of $\boldsymbol{H}$ is the reduced $1 \times k$ representation of the j-th document *in the same latent space*; $\boldsymbol{Y}_{i,j}$ is the inner product of these two latent representations, measuring 'relevance'.

Uniqueness of NMF is tantamount to the question of whether or not these true latent factors are *the only* interpretation of the data, or alternative ones exist. Unfortunately, NMF is in general non-unique. One can inquire about existence and uniqueness of NMF of $\boldsymbol{Y}$ for a given k without any other side information; or about uniqueness of a *particular factorization* $\boldsymbol{Y} = \boldsymbol{W}\boldsymbol{H}^T$, i.e., given a particular pair of factors $\boldsymbol{W}$, $\boldsymbol{H}$ as side information. Additional constraints can be added to help make the factorization unique, e.g. sparsity [28] or minimum determinant [29]. Thomas [30] and Chen [21] first gave different geometric interpretations of NMF, and stated the uniqueness problem in a geometric way. Donoho *et al.* [2] and Laurberg *et al.* [31] later provided uniqueness conditions by exploiting the aforementioned geometric viewpoint. The sufficient conditions they provided, however, require one of the two matrix factors to contain a scaled identity matrix—a particularly strict condition. Moussaoui *et al.* [32] and Laurberg *et al.* [31] also provided necessary conditions. We will review these conditions in Section 2.1, and discuss their relationship with the uniqueness conditions we provide herein.

2.1 Uniqueness of NMF: Precursors

Recall that we are interested in the factorization $\boldsymbol{Y} = \boldsymbol{W}\boldsymbol{H}^T$ where the $m \times k$ matrix $\boldsymbol{W}$ and the $n \times k$ matrix $\boldsymbol{H}$ have non-negative elements. For uniqueness analysis, we

shall assume that $k = \operatorname{rank}\{\boldsymbol{Y}\}$, and thus both $\boldsymbol{W}$ and $\boldsymbol{H}$ are full rank. (Notice that if $\boldsymbol{W}$ and $\boldsymbol{H}$ are drawn from a jointly continuous distribution and $\boldsymbol{Y}$ is in fact constructed by taking their product, then $\operatorname{rank}\{\boldsymbol{Y}\} = k$ almost surely.) Therefore, if $\boldsymbol{Y} = \boldsymbol{W}\boldsymbol{H}^T = \tilde{\boldsymbol{W}}\tilde{\boldsymbol{H}}^T$, then there exists a $k \times k$ full rank matrix $\boldsymbol{A}$ such that $\tilde{\boldsymbol{W}} = \boldsymbol{W}\boldsymbol{A}^{-T}$ and $\tilde{\boldsymbol{H}} = \boldsymbol{H}\boldsymbol{A}$. A trivial choice of $\boldsymbol{A}$ would be a positively scaled permutation matrix; such ambiguity is unavoidable without side information.

Definition 2.1 (uniqueness of NMF)**.** *The NMF of $\boldsymbol{Y} = \boldsymbol{W}\boldsymbol{H}^T$ is said to be (essentially) unique if $\boldsymbol{Y} = \tilde{\boldsymbol{W}}\tilde{\boldsymbol{H}}^T$ implies $\tilde{\boldsymbol{W}} = \boldsymbol{W}\boldsymbol{P}\boldsymbol{D}$ and $\tilde{\boldsymbol{H}} = \boldsymbol{H}\boldsymbol{P}\boldsymbol{D}^{-1}$, where $\boldsymbol{D}$ is a diagonal matrix with its diagonal entries positive, and $\boldsymbol{P}$ is a permutation matrix.*

From this definition, it is straight-forward to derive a necessary condition for the uniqueness of NMF.

Theorem 2.1 (necessary condition)**.** *Let* $\operatorname{supp}\{\boldsymbol{x}\}$ *denote the index set of the non-zero entries (support) of a vector $\boldsymbol{x}$, i.e.,* $\operatorname{supp}\{\boldsymbol{x}\} = \{i \mid x_i \neq 0\}$*. If the NMF $\boldsymbol{Y} = \boldsymbol{W}\boldsymbol{H}^T$ is unique, then there do not exist $\mu, \nu \in \{1, \cdots, k\}, \mu \neq \nu$, such that* $\operatorname{supp}\{\boldsymbol{w}_\mu\} \subseteq \operatorname{supp}\{\boldsymbol{w}_\nu\}$*, or* $\operatorname{supp}\{\boldsymbol{h}_\mu\} \subseteq \operatorname{supp}\{\boldsymbol{h}_\nu\}$*.*

Proof. Suppose $\operatorname{supp}\{\boldsymbol{h}_\mu\} \subseteq \operatorname{supp}\{\boldsymbol{h}_\nu\}$, then there exist a positive scalar α such that

$$\boldsymbol{h}_\nu - \alpha\boldsymbol{h}_\mu \geq 0.$$

Define a $k \times k$ upper triangular matrix $\boldsymbol{A}$ as

$$\boldsymbol{A} = \boldsymbol{I} - \alpha\boldsymbol{e}_\mu\boldsymbol{e}_\nu^T,$$

then

$$\boldsymbol{A}^{-1} = \boldsymbol{I} + \alpha\boldsymbol{e}_\mu\boldsymbol{e}_\nu^T.$$

Let $\tilde{\boldsymbol{W}} = \boldsymbol{W}\boldsymbol{A}^{-T}$ and $\tilde{\boldsymbol{H}} = \boldsymbol{H}\boldsymbol{A}$. Since $\boldsymbol{A}^{-1} \geq 0$, we have that $\tilde{\boldsymbol{W}} \geq 0$. On the other hand, $\tilde{\boldsymbol{H}}$ satisfies that

$$\tilde{\boldsymbol{h}}_j = \begin{cases} \boldsymbol{h}_\nu - \alpha\boldsymbol{h}_\mu & , j = \nu \\ \boldsymbol{h}_j & , \text{otherwise.} \end{cases}$$

Therefore $\tilde{\boldsymbol{H}} \geq 0$, which means $\boldsymbol{Y} = \tilde{\boldsymbol{W}}\tilde{\boldsymbol{H}}^T = \boldsymbol{W}\boldsymbol{A}^{-T}\boldsymbol{A}\boldsymbol{H}^T$, but $\boldsymbol{A}$ is not a scaled permutation matrix.

A similar argument can be applied when $\operatorname{supp}\{\boldsymbol{w}_\mu\} \subseteq \operatorname{supp}\{\boldsymbol{w}_\nu\}$. □

A very interesting implication from Theorem 2.1 is as follows, which relates the uniqueness of NMF with sparsity of the latent factors.

Corollary 2.1. *If the NMF $\boldsymbol{Y} = \boldsymbol{W}\boldsymbol{H}^T$ is unique, then each column of $\boldsymbol{W}$ and $\boldsymbol{H}$ contains at least one element equal to 0.*

Proof. If the ν-th column of $\boldsymbol{H}$ does not have 0 element, then clearly $\text{supp}\{\boldsymbol{h}_\nu\} = \{1, \cdots, n\}$, and $\text{supp}\{\boldsymbol{h}_\mu\} \subseteq \text{supp}\{\boldsymbol{h}_\nu\}, \forall\, \mu \neq \nu$, which violates the condition given in Theorem 2.1; and likewise for the columns of $\boldsymbol{W}$. □

In terms of sufficient conditions, which are in practice more of interest, the existing results are far from satisfactory. Here we list two most noticeable results prior to our work, followed by discussions.

Theorem 2.2 (Donoho and Stodden [2])**.** *The NMF $\boldsymbol{Y} = \boldsymbol{W}\boldsymbol{H}^T$ is unique if the following conditions are satisfied.*

- Separability: *For each $\kappa = 1, \cdots, k$ there exists $i \in \{1, \cdots, n\}$ such that*

$$\begin{aligned} \boldsymbol{H}_{i,\kappa} &\neq 0 \\ \boldsymbol{H}_{i,l} &= 0, \forall\ l \neq \kappa \end{aligned}$$

- Generative Model: *The set $\{1, 2, \cdots, k\}$ is partitioned into A groups $\mathcal{P}_1, \cdots, \mathcal{P}_A$, each containing exactly P elements (therefore $k = AP$). For every $i = 1, \cdots, m$ and $a = 1, \cdots, A$, there exists an element $w_{i,\kappa}$ such that*

$$\begin{aligned} \boldsymbol{W}_{i,\kappa} &\neq 0, \kappa \in \mathcal{P}_a, \\ \boldsymbol{W}_{i,l} &= 0, \forall l \in \mathcal{P}_a, l \neq \kappa \end{aligned}$$

- Complete Factorial Sampling: *For any $\kappa_1 \in \mathcal{P}_1, \kappa_2 \in \mathcal{P}_2, \cdots, \kappa_A \in \mathcal{P}_A$, there exists $i \in \{1, \cdots, n\}$ such that*

$$\boldsymbol{W}_{i,\kappa_1} \neq 0, \boldsymbol{W}_{i,\kappa_2} \neq 0, \cdots, \boldsymbol{W}_{i,\kappa_A} \neq 0$$

Theorem 2.3 (Laurberg *et al.* [33])**.** *The NMF $\boldsymbol{Y} = \boldsymbol{W}\boldsymbol{H}^T$ is unique if the following assumptions are satisfied.*

- Sufficiently Spread: *For each* $\kappa = 1, \cdots, k$ *there is a* $i \in \{1, \cdots, n\}$ *such that*

$$\begin{aligned} \boldsymbol{H}_{i,\kappa} &\neq 0 \\ \boldsymbol{H}_{i,l} &= 0, \forall\ l \neq \kappa \end{aligned}$$

- Strongly Boundary Close: *the matrix* $\boldsymbol{W}$ *satisfies the following conditions*

  1. *For each* $\kappa = 1, \cdots, k$ *there is an* $i \in \{1, \cdots, m\}$ *such that*

$$\begin{aligned} \boldsymbol{W}_{i,\kappa} &= 0 \\ \boldsymbol{W}_{i,j} &\neq 0, \forall\, j \neq i \end{aligned}$$

  2. *There exists a permutation matrix* $\boldsymbol{P}$ *such that for all* $i < k$ *there exists a set* $\{\kappa_1, \cdots, \kappa_{k-i}\}$ *fulfilling:* $(\boldsymbol{WP})_{i,\kappa_j} = 0$ *for all* $j \leq k - i$*; and the matrix*

$$\begin{bmatrix} (\boldsymbol{WP})_{i+1,\kappa_1} & \cdots & (\boldsymbol{WP})_{i+1,\kappa_{k-i}} \\ \vdots & \ddots & \vdots \\ (\boldsymbol{WP})_{k,\kappa_1} & \cdots & (\boldsymbol{WP})_{k,\kappa_{k-i}} \end{bmatrix}$$

  is invertible.

As we can see, the existing results are 1) very complicated 2) asymmetric, which is non-intuitive mathematically, and 3) imposes a very strict condition on $\boldsymbol{H}$, which is termed *separability* by Donoho and Stodden, and *sufficiently spread* by Laurberg *et al.*, but in fact the same condition. This condition asks that for each $\kappa = 1, ..., k$, there exists a row of $\boldsymbol{H}$ such that only the κ-th entry is non-zero. As a result, every column of $\boldsymbol{W}$ actually appears in the columns of $\boldsymbol{Y}$. This is an over-simplified model, and very hard to satisfy in practice. However, empirically people have seen that NMF admits a unique solution under much more relaxed conditions.

2.2 Geometric Interpretation

Before we analyze the properties of NMF, we briefly review some prerequisites from convex analysis; see [34, 35] for further background.

Definition 2.2 (polyhedral cone)**.** *A polyhedral cone* $\mathcal{K}$ *is a set that is both a polyhedron and a cone.*

There are two ways to describe a polyhedral cone. The first is by taking the intersection of a number of half-spaces, which takes the form

$$\mathcal{K} = \{\boldsymbol{x} \mid \boldsymbol{A}^T\boldsymbol{x} \geq 0\}$$

where each column of $\boldsymbol{A}$ defines a half-space which contains the origin at the boundary. Notice that the right hand side is 0 in order to make this set a cone. Assuming there are no redundant constraints, the i-th constraint satisfied as equality is called a *facet* of $\mathcal{K}$.

The other way to describe a polyhedral cone is by taking the conic hull of a number of vectors, i.e.,

$$\mathcal{K} = \{\boldsymbol{x} = \boldsymbol{B}\boldsymbol{\lambda} \mid \boldsymbol{\lambda} \geq 0\} = \text{cone}\,\{\boldsymbol{B}\}$$

where the columns of $\boldsymbol{B}$ are the vectors we are taking. If a column of $\boldsymbol{B}$ cannot be represented by the conic combination (non-negative linear combination) of the rest of the columns, then it is called an *extreme ray* of $\mathcal{K}$.

Given a polyhedral cone represented by the intersection of half-spaces, an extreme ray of the cone would be a vector satisfying all the inequality constraints, and furthermore, at least $n-1$ linearly independent constraints are satisfied as equalities. Similarly, given a polyhedral cone represented by the conic hull of vectors, a facet of it is a hyperplane defined by at least $n-1$ linearly independent extreme rays (and the origin), and the rest of the extreme rays must be on one side of that hyperplane.

Definition 2.3 (simplicial cone)**.** *A simplicial cone is a polyhedral cone such that all of its extreme rays are linearly independent.*

If $\mathcal{K} = \{\boldsymbol{x} = \boldsymbol{B}\boldsymbol{\lambda} \mid \boldsymbol{\lambda} \geq \boldsymbol{0}\} = \text{cone}\,\{\boldsymbol{B}\}$ is a simplicial cone, then for every element $\boldsymbol{x} \in \mathcal{K}$, there is a *unique* corresponding $\boldsymbol{\lambda}$ that indicates how to conically combine the extreme rays to generate $\boldsymbol{x}$. For general polyhedral cones, this combination is in most cases not unique.

Moreover, it is easy to change the representation of a simplicial cone $\mathcal{K}$ from halfspace-intersection to conic hull. For example, if $\mathcal{K} = \text{cone}\,\{\boldsymbol{B}\}$ where $\boldsymbol{B}$ is invertible, then $\mathcal{K}$ is a simplicial cone and $\mathcal{K} = \{\boldsymbol{x} \mid \boldsymbol{B}^{-1}\boldsymbol{x} \geq 0\}$. However, for general polyhedral cones this switching between representations is a hard problem [36].

Definition 2.4 (dual cone)**.** *The dual cone of a set $\mathcal{K}$, denoted by $\mathcal{K}^*$, is defined as* $\mathcal{K}^* = \{\boldsymbol{y} \mid \boldsymbol{x}^T\boldsymbol{y} \geq 0, \forall\, \boldsymbol{x} \in \mathcal{K}\}$.

Some important properties of dual cones are as follows (cf. Laurberg *et al.* [31]):

Property 2.1. $\operatorname{cone}\{\boldsymbol{A}\}^* = \{\boldsymbol{x} \mid \boldsymbol{A}^T\boldsymbol{x} \geq 0\}$.

For one column of the matrix $\boldsymbol{A}$, in the primal cone it defines an extreme ray, while in the dual cone it defines a facet. Therefore, there is a one-to-one correspondence between the extreme rays of the primal polyhedral cone and facets of the dual polyhedral cone, and vice versa.

Property 2.2. *If $\boldsymbol{A}$ is invertible, then* $\operatorname{cone}\{\boldsymbol{A}\}^* = \operatorname{cone}\left\{\boldsymbol{A}^{-T}\right\}$.

From this property, it is easy to see that if $\boldsymbol{A}$ is unitary, i.e., $\boldsymbol{A}^{-1} = \boldsymbol{A}^T$, then $\operatorname{cone}\{\boldsymbol{A}\}^* = \operatorname{cone}\left\{\boldsymbol{A}^{-T}\right\} = \operatorname{cone}\{\boldsymbol{A}\}$. In other words, the conic hull of a unitary matrix is *self dual*.

Property 2.3. *If $\mathcal{A}$ and $\mathcal{B}$ are convex cones, and $\mathcal{A} \subseteq \mathcal{B}$, then $\mathcal{B}^* \subseteq \mathcal{A}^*$.*

Here is an example of a cone and its dual cone, which will be useful later. Donoho and Stodden also studied the following cones in [2].

Example 2.1. *Define the second-order cone in $\mathbb{R}^k$*

$$\mathcal{C} = \{\boldsymbol{x} | \boldsymbol{x}^T\mathbf{1} \geq \sqrt{k-1}\|\boldsymbol{x}\|_2\} \tag{2.1}$$

Its dual cone is another second-order cone

$$\mathcal{C}^* = \{\boldsymbol{x} | \boldsymbol{x}^T\mathbf{1} \geq \|\boldsymbol{x}\|_2\} \tag{2.2}$$

The reason we are interested in $\mathcal{C}$ and its dual cone is because they have a very special relationship with the non-negative orthant $\mathbb{R}^k_+$: $\mathcal{C} \subseteq \mathbb{R}^k_+ \subseteq \mathcal{C}^*$. Fig. 2.1 gives a graphical view of $\mathcal{C}$, $\mathcal{C}^*$ and $\mathbb{R}^n_+$ in $\mathbb{R}^3$. In fact, $\mathbb{R}^k_+$ and its rotated versions are the only simplicial cones that satisfy this, as stated in the following lemma.

Lemma 2.1. *If* $\operatorname{cone}\{\boldsymbol{A}\}$ *satisfies that* $\mathcal{C} \subseteq \operatorname{cone}\{\boldsymbol{A}\} \subseteq \mathcal{C}^*$*, and the columns of $\boldsymbol{A}$ are scaled to have unit ℓ_2 norm, then*

1. $\boldsymbol{A}^T\mathbf{1} = \mathbf{1}$

2. $\boldsymbol{A}^T\boldsymbol{A} = \boldsymbol{I}$

Proof. Assume $\text{cone}\{\boldsymbol{A}\}$ satisfies that

$$\mathcal{C} \subseteq \text{cone}\{\boldsymbol{A}\} \subseteq \mathcal{C}^*$$

According to Property 2.2 and 2.3 of dual cone, this is equivalent to

$$\text{cone}\{\boldsymbol{A}\} \subseteq \mathcal{C}^*, \text{cone}\left\{\boldsymbol{A}^{-T}\right\} \subseteq \mathcal{C}^*$$

$\text{cone}\{\boldsymbol{A}\} \subseteq \mathcal{C}^*$ means that every column of $\boldsymbol{A}$ is in $\mathcal{C}^*$, therefore

$$\mathbf{1}^T\boldsymbol{A}\boldsymbol{e}_i \geq \|\boldsymbol{A}\boldsymbol{e}_i\|_2, i = 1, \cdots, k \tag{2.3}$$

Let $\boldsymbol{B} = \boldsymbol{A}^{-T}$, i.e. $\boldsymbol{A}^T\boldsymbol{B} = \boldsymbol{I}$. Similarly, for $\text{cone}\{\boldsymbol{B}\} \subseteq \mathcal{C}^*$, we have

$$\mathbf{1}^T\boldsymbol{B}\boldsymbol{e}_i \geq \|\boldsymbol{B}\boldsymbol{e}_i\|_2, i = 1, \cdots, k \tag{2.4}$$

Both (2.3) and (2.4) involve only non-negative numbers, so we can take their product and sum over all i's to get

$$\mathbf{1}^T\boldsymbol{A}\boldsymbol{B}^T\mathbf{1} \geq \sum_{i=1}^{k} \|\boldsymbol{A}\boldsymbol{e}_i\|_2\|\boldsymbol{B}\boldsymbol{e}_i\|_2 \tag{2.5}$$

The left hand side of (2.5) equals n, since $\mathbf{AB^T} = \mathbf{I}$. Using the Cauchy-Schwarz inequality, the right hand side of (2.5) is

$$\sum_{i=1}^{k} \|\boldsymbol{A}\boldsymbol{e}_i\|_2\|\boldsymbol{B}\boldsymbol{e}_i\|_2 \geq \sum_{i=1}^{k} (\boldsymbol{B}\boldsymbol{e}_i)^T\boldsymbol{A}\boldsymbol{e}_i \tag{2.6}$$

The right hand side equals to n too, again thanks to $\boldsymbol{A}^T\boldsymbol{B} = \boldsymbol{I}$. Therefore, all the inequalities (2.3)-(2.6) are equalities. Notice that (2.6) is satisfied as an equality if and only if $\boldsymbol{B}\boldsymbol{e}_i$ is a positively scaled version of $\boldsymbol{A}\boldsymbol{e}_i$, for all $i = 1, \cdots, k$. Since we assume $\|\boldsymbol{A}\boldsymbol{e}_i\|_2 = 1$ for all i's, then $\boldsymbol{A} = \boldsymbol{B}$, i.e., $\boldsymbol{A} = \boldsymbol{A}^{-T}$. Therefore, the columns of $\boldsymbol{A}$ are orthogonal to each other. Furthermore, if (2.5) is satisfied as equality, it implies that (2.3) and (2.4) are also equalities. In other words, the extreme rays of $\boldsymbol{A}$ lie on the boundary of $\mathcal{C}^*$, i.e. $\boldsymbol{A}^T\mathbf{1} = \mathbf{1}$. □

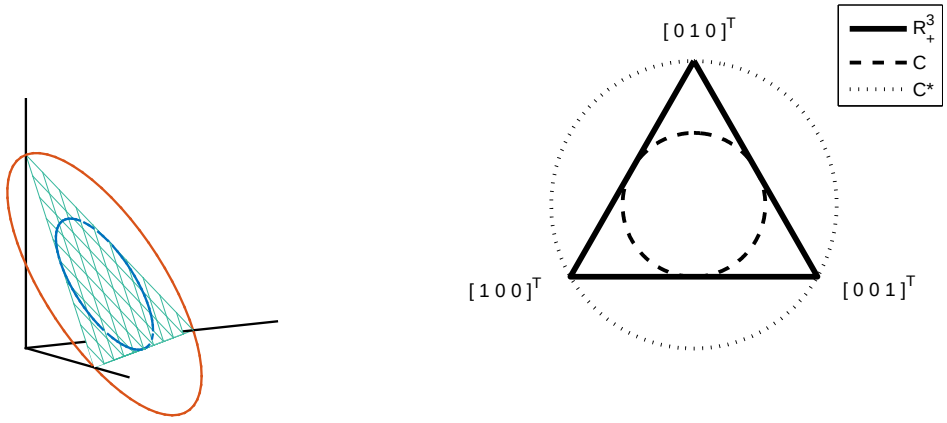


Figure 2.1: A 2-D slice view of the relationship between $\mathbb{R}^3_+, \mathcal{C}, \mathcal{C}^*$ in $\mathbb{R}^3$, looking at the plane $\mathbf{1}^T \boldsymbol{x} = 1$.

Equipped with the convex analysis bases, we can revisit the uniqueness of NMF, and give it a geometric interpretation.

Lemma 2.2. *If* $\operatorname{rank}\{\boldsymbol{Y}\} = k$*, the NMF* $\boldsymbol{Y} = \boldsymbol{W}\boldsymbol{H}^T$ *is unique if and only if the non-negative orthant is the only simplicial cone* $\mathcal{A}$ *with* k *extreme rays that satisfies* $\operatorname{cone}\left\{\boldsymbol{H}^T\right\} \subseteq \mathcal{A} \subseteq \operatorname{cone}\left\{\boldsymbol{W}^T\right\}^*$.

Proof. Since $\operatorname{rank}\{\boldsymbol{Y}\} = k$, $\boldsymbol{Y} = \boldsymbol{W}\boldsymbol{H}^T = \tilde{\boldsymbol{W}}\tilde{\boldsymbol{H}}^T$ implies that there exists a non-singular matrix $\boldsymbol{A}$ such that $\tilde{\boldsymbol{W}} = \boldsymbol{W}\boldsymbol{A}, \tilde{\boldsymbol{H}} = \boldsymbol{H}\boldsymbol{A}^{-T}$. For NMF we further require that $\tilde{\boldsymbol{W}}$ and $\tilde{\boldsymbol{H}}$ to be non-negative, therefore,

$$\tilde{\boldsymbol{W}} = \boldsymbol{W}\boldsymbol{A} \geq 0, \ \tilde{\boldsymbol{H}} = \boldsymbol{H}\boldsymbol{A}^{-T} \geq 0,$$

which, geometrically, means that

$$\operatorname{cone}\{\boldsymbol{A}\} \subseteq \operatorname{cone}\left\{\boldsymbol{W}^T\right\}^*, \ \operatorname{cone}\left\{\boldsymbol{H}^T\right\} \subseteq \operatorname{cone}\left\{\boldsymbol{A}^{-T}\right\}^*.$$

Furthermore, from Property 2.2 we have that $\operatorname{cone}\{\boldsymbol{A}\} = \operatorname{cone}\left\{\boldsymbol{A}^{-T}\right\}^*$, leading to

$$\operatorname{cone}\left\{\boldsymbol{H}^T\right\} \subseteq \operatorname{cone}\{\boldsymbol{A}\} \subseteq \operatorname{cone}\left\{\boldsymbol{W}^T\right\}^*.$$

From Definition 2.1, NMF is unique is the matrix $\boldsymbol{A}$ can only be chosen to be a scaled permutation matrix, meaning $\operatorname{cone}\{\boldsymbol{A}\}$ can only be the non-negative orthant. □

2.3 Uniqueness of NMF: Main Results

We are now ready to present our new conditions on the uniqueness of NMF. We need the second-order cone $\mathcal{C}$ and its dual cone defined in (2.1) and (2.2) from Example 2.1 to help derive our sufficient condition.

2.3.1 Sufficient Condition

Definition 2.5 (sufficiently scattered)**.** *A non-negative matrix $\boldsymbol{X}$ is sufficiently scattered if*

1. $\text{cone}\left\{\boldsymbol{X}^T\right\} \supseteq \mathcal{C}$,
2. $\text{cone}\left\{\boldsymbol{X}^T\right\}^* \cap \text{bd}\left\{\mathcal{C}\right\}^* = \{\lambda \boldsymbol{e}_\kappa \mid \lambda \geq 0, \kappa = 1, ..., k.\}$,

where $\mathcal{C}$ and $\mathcal{C}^*$ are

$$\mathcal{C} = \{\boldsymbol{x} | \boldsymbol{x}^T \boldsymbol{1} \geq \sqrt{k-1}\|\boldsymbol{x}\|_2\}, \quad \mathcal{C}^* = \{\boldsymbol{x} | \boldsymbol{x}^T \boldsymbol{1} \geq \|\boldsymbol{x}\|_2\}.$$

The interpretation of the second part of Definition 2.5 is the following. First of all, according to Property 2.3 of the dual cones, $\text{cone}\left\{\boldsymbol{X}^T\right\}^* \subseteq \mathcal{C}^*$ is equivalent to $\text{cone}\left\{\boldsymbol{X}^T\right\} \supseteq \mathcal{C}$, which means all the extreme rays of $\text{cone}\left\{\boldsymbol{X}^T\right\}^*$ are contained in $\mathcal{C}^*$. Per the proof of Theorem 2.4, $\boldsymbol{e}_\kappa$'s are the extreme rays of $\text{cone}\left\{\boldsymbol{X}^T\right\}^*$, and they lie on the boundary of $\mathcal{C}^*$ too. This statement requires that all the other extreme rays of $\text{cone}\left\{\boldsymbol{X}^T\right\}^*$ lie in the interior of $\mathcal{C}^*$.

Theorem 2.4 (sufficient condition)**.** *If $\boldsymbol{W}$ and $\boldsymbol{H}$ are both sufficiently scattered, then the NMF $\boldsymbol{Y} = \boldsymbol{W}\boldsymbol{H}^T$ is essentially unique.*

Proof. We first prove that if $\text{cone}\left\{\boldsymbol{X}^T\right\} \supseteq \mathcal{C}$, then $\forall \kappa = 1, \ldots, k$, $\boldsymbol{e}_\kappa$ is an extreme ray of $\text{cone}\left\{\boldsymbol{X}^T\right\}^*$. This is because 1) $\boldsymbol{e}_\kappa$ is an extreme ray of $\mathcal{C}^*$ (the constraint (2.2) is satisfied as equality at $\boldsymbol{e}_\kappa$, hence $\boldsymbol{e}_\kappa$'s are on the boundary of $\mathcal{C}^*$, and every ray that lies on the boundary of a second order cone is an extreme ray of this cone); and 2) $\boldsymbol{e}_\kappa$ is contained in $\text{cone}\left\{\boldsymbol{X}^T\right\}^*$ (obviously $\boldsymbol{X}\boldsymbol{e}_\kappa \geq 0$). The $\boldsymbol{e}_\kappa$'s being *extreme* rays of $\mathcal{C}^*$ means that there do not exist two other elements in $\mathcal{C}^*$ that can conically generate $\boldsymbol{e}_\kappa$, and since $\text{cone}\left\{\boldsymbol{X}^T\right\}^* \subseteq \mathcal{C}^*$, there certainly do not exist two other elements in

cone $\left\{\boldsymbol{X}^T\right\}^*$ that can conically generate $\boldsymbol{e}_\kappa$; therefore, the $\boldsymbol{e}_\kappa$'s are the extreme rays of cone $\left\{\boldsymbol{X}^T\right\}^*$.

According to Lemma 2.2, we need to show that under this condition, if a simplicial cone $\mathcal{A}$ satisfies that cone $\left\{\boldsymbol{H}^T\right\} \subseteq \mathcal{A} \subseteq$ cone $\left\{\boldsymbol{W}^T\right\}^*$, then $\mathcal{A} = \mathbb{R}_+^k$. Since cone $\left\{\boldsymbol{H}^T\right\} \supseteq \mathcal{C}$ and cone $\left\{\boldsymbol{W}^T\right\} \supseteq \mathcal{C}$, obviously $\mathcal{C} \subseteq \mathcal{A} \subseteq \mathcal{C}^*$. According to Lemma 2.1, we know that $\mathcal{A}$ can only be a rotated version of $\mathbb{R}_+^k$, and that all of its extreme rays lie on the boundary of $\mathcal{C}^*$. However, since none of the extreme rays of cone $\left\{\boldsymbol{W}^T\right\}^*$ except $\boldsymbol{e}_\kappa$'s lie on the boundary of $\mathcal{C}^*$, $\mathcal{A}$ can only be the non-negative orthant $\mathbb{R}_+^k$ itself. Therefore under this condition the asymmetric NMF $\boldsymbol{Y} = \boldsymbol{W}\boldsymbol{H}^T$ is unique. □

Our result given in Theorem 2.4 is first of all a clean statement, and is also symmetric with respect to both $\boldsymbol{W}$ and $\boldsymbol{H}$. To gain some insights to this condition, we revisit one toy example first appeared by Laurberg *et al.* [31], but the uniqueness cannot be identified using their analysis. Using our analysis, however, the uniqueness can be exactly demonstrated.

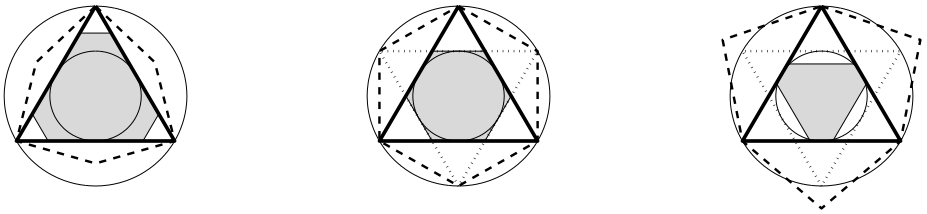
Example 2.2. *[31] Consider the NMF* $\boldsymbol{Y} = \boldsymbol{W}\boldsymbol{H}^T$ *where*

$$\boldsymbol{W} = \boldsymbol{H} = \begin{bmatrix} \omega & 1 & 1 & \omega & 0 & 0 \\ 1 & \omega & 0 & 0 & \omega & 1 \\ 0 & 0 & \omega & 1 & 1 & \omega \end{bmatrix}^T$$

For $0 \leq \omega \leq 1$*, the NMF is unique if and only if* $\omega < 0.5$*. This is because if* $\omega < 0.5$*, both* $\boldsymbol{W}$ *and* $\boldsymbol{H}$ *are sufficiently scattered. If* $\omega \geq 0.5$*, define*

$$\boldsymbol{A} = \frac{1}{3}\begin{bmatrix} -1 & 2 & 2 \\ 2 & -1 & 2 \\ 2 & 2 & -1 \end{bmatrix}.$$

One can check that $\mathbf{A}$ *is unitary and* $\boldsymbol{W}\boldsymbol{A} = \boldsymbol{H}\boldsymbol{A} \geq 0$ *in this case, hence this NMF is not unique. Fig. 2.2 illustrates the relationship between* cone $\left\{\boldsymbol{H}^T\right\}$, cone $\left\{\boldsymbol{W}^T\right\}$ *and* $\mathcal{C}$ *for selected values of* ω*. Notice that when* $\omega = 0.5$*, as is shown in Fig. 2.2b, although* cone $\left\{\boldsymbol{W}^T\right\} \supseteq \mathcal{C}$ *is satisfied, there are other extreme rays of* cone $\left\{\boldsymbol{W}^T\right\}^*$ *that lie on the boundary of* $\mathcal{C}^*$ *and are orthogonal to each other. Thus, it is still possible in this case to form a suitable simplicial cone which satisfies the requirement given in Lemma 2.2.*



(a) $\omega = 0.25$ (b) $\omega = 0.5$ (c) $\omega = 0.75$

Figure 2.2: A graphical view of Example 2.2 plotted in the same manner as Figure 2.1. The triangle drawn with solid line is $\mathbb{R}^3_+$; the inner and outer circles are $\mathcal{C}$ and $\mathcal{C}^*$ respectively; the shaded area is cone $\left\{\boldsymbol{H}^T\right\}$, and the polygon drawn with the dashed line is cone $\left\{\boldsymbol{W}^T\right\}^*$. When $\omega = 0.25$, the sufficiently scattered condition is satisfied, so the NMF is unique; when $\omega = 0.75$, cone $\left\{\boldsymbol{W}^T\right\} \not\supseteq \mathcal{C}$, so the NMF is not unique; when $\omega = 0.5$, although cone $\left\{\boldsymbol{W}^T\right\} \supseteq \mathcal{C}$, all the extreme rays of cone $\left\{\boldsymbol{W}^T\right\}^*$ lie on the boundary of $\mathcal{C}^*$, therefore it is still not "scattered enough". The dotted triangle in Figure 2.2b and 2.2c shows another self-dual simplicial cone $\mathcal{A}$ satisfying cone $\left\{\boldsymbol{H}^T\right\} \subseteq \mathcal{A} \subseteq$ cone $\left\{\boldsymbol{W}^T\right\}^*$.

Example 2.2 illustrates the usage of Theorem 2.4 to check the uniqueness of NMF in low-dimensional cases. Unfortunately, as the dimension increases, it becomes very hard to check the sufficient condition, since checking whether cone $\left\{\boldsymbol{W}^T\right\} \supseteq \mathcal{C}$ is true is NP-complete. To see the NP-completeness of this problem, we can first intersect both cone $\left\{\boldsymbol{W}^T\right\}$ and $\mathcal{C}$ with the hyperplane $\mathbf{1}^T\boldsymbol{x} = 1$. Then it becomes checking whether a ball is a subset of a polytope described as the convex hull of points. Freund and Orlin [37] considered several set containment problems and proved that the aforementioned one is NP-complete. In the next subsection consider the practical value of sufficiently scattered.

2.3.2 Uniqueness in Practice

From a computational complexity point of view, we started from a criterion (Lemma 2.2) that is NP-hard to check, and reached a sufficient condition that is NP-complete to check, which does not seem like much progress. However, Theorem 2.4 treats $\boldsymbol{W}$

and $\boldsymbol{H}$ in a balanced fashion, unlike other sufficient conditions which are strict on one matrix factor, lenient on the other. Furthermore, Theorem 2.4 gives implications about how the latent factors look like if they indeed satisfy its sufficient conditions, as shown next.

Corollary 2.2. *If the condition given in Theorem 2.4 is satisfied for the NMF* $\boldsymbol{Y} = \boldsymbol{W}\boldsymbol{H}^T$*, then for all* $\kappa = 1, 2, \cdots, k$*,* $\boldsymbol{e}_\kappa$ *is an extreme ray of* $\text{cone}\left\{\boldsymbol{W}^T\right\}^*$ *and* $\text{cone}\left\{\boldsymbol{H}^T\right\}^*$*. Thus, each column of* $\boldsymbol{W}$ *and* $\boldsymbol{H}$ *contains at least* $k-1$ *zero entries.*

Proof. We have argued in the proof of Theorem 2.4 that if the sufficient condition is satisfied, the $\boldsymbol{e}_\kappa$'s are extreme rays of $\text{cone}\left\{\boldsymbol{W}^T\right\}^*$ and $\text{cone}\left\{\boldsymbol{H}^T\right\}^*$. An extreme ray is the intersection of at least $k-1$ independent facets of a polyhedral cone with dimension K. Therefore for $\text{cone}\left\{\boldsymbol{W}^T\right\}^*$, we have

$$\boldsymbol{W}\boldsymbol{e}_\kappa \geq 0$$

and that at least $k-1$ of them are satisfied as equalities. Therefore, each column of $\boldsymbol{W}$ and $\boldsymbol{H}$ contains at least $k-1$ zero entries. □

Corollary 2.2 builds a link between uniqueness and latent sparsity of NMF. It is observed in practice that if the true latent factors are sparse, NMF usually tends to recover the correct solution, up to scaling and permutation. However, counter-examples do exist, e.g. Example 2.2 when $0.5 \leq \omega < 1$. Now we understand that the reason sparse latent factors usually lead to unique NMF is because if the latent factors are sparse, it is more *likely* that the sufficient condition given in Theorem 2.4 is satisfied.

Example 2.3. *In this example we randomly generate a* 200×30 *non-negative matrix* $\boldsymbol{W}$ *and a* 30×250 *non-negative matrix* $\boldsymbol{H}$*, with a certain proportion of randomly selected entries set to zero, and the non-zero entries drawn from an i.i.d. exponential distribution. The columns of* $\boldsymbol{W}$ *are scaled to sum up to 1*

$$\sum_{i=1}^{m} w_{i,1} = \sum_{i=1}^{m} w_{i,2} = \cdots = \sum_{i=1}^{m} w_{i,k} = 1,$$

and the columns of $\boldsymbol{H}$ *are ordered such that*

$$\sum_{j=1}^{n} h_{j,1} > \sum_{j=1}^{n} h_{j,2} > \cdots > \sum_{j=1}^{n} h_{j,k}.$$

Table 2.1: Maximum reconstruction error for asymmetric NMF

density	max $\|\hat{\boldsymbol{W}} - \boldsymbol{W}\|_F$	max $\|\hat{\boldsymbol{H}} - \boldsymbol{H}\|_F$
0.5	0.0070×10^{-7}	0.0083×10^{-5}
0.6	0.0030×10^{-7}	0.0067×10^{-5}
0.7	0.0184×10^{-7}	0.0302×10^{-5}
0.8	0.1154×10^{-7}	0.1991×10^{-5}

Then we form the low rank non-negative matrix $\boldsymbol{Y} = \boldsymbol{W}\boldsymbol{H}^T$*, and apply NMF on* $\boldsymbol{Y}$ *to get an estimate* $\hat{\boldsymbol{W}}$ *and* $\hat{\boldsymbol{H}}$*. The columns of* $\hat{\boldsymbol{W}}$ *are scaled to sum up to 1, with the scaling absorbed into the rows of* $\hat{\boldsymbol{H}}$*, in order to avoid the scaling ambiguity; and the rows of* $\hat{\boldsymbol{H}}$ *are then re-ordered, with the same re-ordering applied to the columns of* $\hat{\boldsymbol{W}}$*, to fix the permutation ambiguity. For density varying from 0.5 to 0.8, in which case the matrices* $\boldsymbol{W}$ *and* $\boldsymbol{H}$ *we randomly generated satisfy that* $\boldsymbol{e}_\kappa$*'s are extreme rays of* cone $\left\{\boldsymbol{W}^T\right\}^*$ *and* cone $\left\{\boldsymbol{H}^T\right\}^*$ *with high probability, this procedure is repeated 100 times, and the maximum reconstruction errors for* $\hat{\boldsymbol{W}}$ *and* $\hat{\boldsymbol{H}}$ *are given in the following table*

Notice that the zero elements are randomly located, therefore neither Donoho's separability assumption [2] nor Laurberg's sufficiently spread assumption [31] are satisfied. However, as can be observed, in all cases the asymmetric NMF is able to reconstruct the true latent factors. The algorithm used here was AO-ADMM proposed in [14].

The results given in the above examples are reassuring to a certain degree, in light of the fact that the true sufficient condition is NP-complete to check. They also showcase how strict the previously suggested sufficient conditions [2, 31] are, since in those examples the columns of $\boldsymbol{H}$ are highly unlikely to contain all scaled versions of $\boldsymbol{e}_\kappa$'s.

On a different path, we can attempt to check if a matrix $\boldsymbol{H}$ is sufficiently scattered via solving the following non-convex quadratic programming optimally.

$$\begin{aligned} \underset{\boldsymbol{x}}{\text{maximize}} \quad & \|\boldsymbol{x}\|_2^2, \\ \text{subject to} \quad & \boldsymbol{H}\boldsymbol{x} \geq 0, \quad \boldsymbol{x}^T\boldsymbol{1} = 1. \end{aligned} \tag{2.7}$$

Then cone $\left\{\boldsymbol{H}^T\right\}^* \subseteq \mathcal{C}^*$ is true if and only if the optimal value of (2.7) is strictly larger than 1. A simple observation is that since $\boldsymbol{H}$ is non-negative, the standard vectors $\boldsymbol{e}_\kappa$

are clearly feasible, and that they lead to the cost equal to 1. In fact, if a feasible point makes the cost equal to 1, then it lies on $\mathbf{bd}\mathcal{C}^*$. Therefore, $\boldsymbol{H}$ is sufficiently scatteredif and only if the optimal value of (2.7) is 1, and all the optimal solutions are the set of standard vectors.

Since that particular set containment problem is NP-hard to check, clearly (2.7) is also NP-hard. A heuristic is to iteratively linearize the objective function and solve a linear program, i.e.,

$$\begin{aligned} \underset{\boldsymbol{x}}{\text{maximize}} \quad & \boldsymbol{x}_r^T \boldsymbol{x}, \\ \text{subject to} \quad & \boldsymbol{H}\boldsymbol{x} \geq 0, \quad \boldsymbol{x}^T \mathbf{1} = 1. \end{aligned} \tag{2.8}$$

where $\boldsymbol{x}_r$ is obtained from the solution of the previous iteration. Since the first-order Taylor expansion of a convex function is always a lower-bound of that function, we can see that by iteratively solving (2.8), we are actually iteratively maximizing a lower-bound of (2.7), which falls into the majorization-optimization method category [38].

Using Corollary 2.2 as a rule of thumb for the sparsity requirement for the matrix, which is not very strict in terms of sparsity, we can generate random sparse non-negative matrices and try to check whether they satisfy sufficiently scatteredby approximately solving (2.7). Although the method we propose to solve (2.7) is not guaranteed to be optimal, we can try different initializations to ensure that most of the local optima are found.

As a simple example, we randomly generate matrices with $n = 300$, and with various number of rows k and/or ratio of non-zeros s. Similar to the synthetic data generated in Example 2.3, whether the entries are zeros follow an i.i.d. Bernoulli distribution, and the non-zeros are drawn from an i.i.d. exponential distribution. For each case 100 random matrices are generated and set as input to the optimization problem (2.7), and then approximately solved by successively solving (2.8) with 100 random initial points. The percentage of the matrices that result in a solution with norm larger than 1 is given in Table 2.2. As we can see, sparse, non-negative tall matrices satisfies sufficiently scattered with very high probability.

Table 2.2: The percentage of the matrices with $n = 300$ columns that result in a solution with norm larger than 1.

	$k = 20$	$k = 30$	$k = 50$
$s = 0.3$	0%	0%	0%
$s = 0.5$	0%	0%	1%
$s = 0.7$	0%	1%	3%

Chapter 3

Volume Minimization-Based Matrix Factorization

In the previous chapter we studied the non-negative matrix factorization (NMF) $\boldsymbol{Y} = \boldsymbol{W}\boldsymbol{H}^T$ where both $\boldsymbol{W}$ and $\boldsymbol{H}$ are constrained to be element-wise non-negative, and showed that if both $\boldsymbol{W}$ and $\boldsymbol{H}$ satisfy the sufficiently scattered condition, the factorization is essentially unique, thus the true latent factors can be revealed via NMF. Even though such condition is computationally hard to check, we illustrated that it is satisfied with very high probability as long as the latent factors are sparse, which is a very relaxed condition to hold in a lot of applications. In certain other applications, however, such latent structures cannot be assumed to be practical, even though we still want to identify the latent factors using matrix factorization methods. For example, a widely considered application, hyperspectral unmixing, deals with dense image data, and domain knowledge reveals that we can only assume one of the latent factor is non-negative and sparse, while the other factor is known to be dense (and can sometimes contain negative values).

In this chapter, we study a variant of NMF $\boldsymbol{Y} = \boldsymbol{W}\boldsymbol{H}^T$ that constrains *only* $\boldsymbol{H} \geq 0$. Without additional specifications, such a "semi-non-negative" matrix factorization is clearly non-unique, since we can use any non-singular non-negative matrix $\boldsymbol{A}$ and let $\tilde{\boldsymbol{H}} = \boldsymbol{H}\boldsymbol{A}$, $\tilde{\boldsymbol{W}} = \boldsymbol{W}\boldsymbol{A}^{-T}$, so that $\boldsymbol{Y} = \tilde{\boldsymbol{W}}\tilde{\boldsymbol{H}}^T$, and $\tilde{\boldsymbol{H}}$ is non-negative. However, with proper scaling on $\boldsymbol{H}$, we will see that using the so-called "volume minimization"

(VolMin) criterion, one can still guarantee (essential) uniqueness of the solution, as long as $\boldsymbol{H}$ satisfies the previously discussed sufficiently scattered condition. Notice that now we only require $\boldsymbol{H}$ to be sufficiently scattered, while no structure is assumed for $\boldsymbol{W}$, although in practice one may impose additional constraints reflecting prior information on $\boldsymbol{W}$, where applicable (such as non-negativity) to improve estimation performance.

3.1 Uniqueness: Existing Results

In the previous chapter we provided a geometric interpretation of NMF, focusing on the latent domain of the two factors. Perhaps a more straightforward way to give a geometric interpretation of matrix factorization models with $\boldsymbol{H} \geq 0$ is directly looking at the column space of $\boldsymbol{Y}$—each column $\boldsymbol{y}_j = \boldsymbol{W}\boldsymbol{h}[j] = \sum_{l=1}^{k} h_{jl}\boldsymbol{w}_l$ is a non-negative combination of columns of $\boldsymbol{W}$, therefore geometrically $\boldsymbol{Y} = \boldsymbol{W}\boldsymbol{H}^T$ means we want to find a simplicial cone $\mathcal{W}$ such that $\text{cone}\{\boldsymbol{Y}\} \subseteq \mathcal{W}$, and the extreme rays of $\mathcal{W}$ are the columns of $\boldsymbol{W}$. Depending on the particular application, we may or may not require $\mathcal{W} \in \mathbb{R}_+^m$.

Now suppose that, in addition, the columns of $\boldsymbol{Y}$ also lie on a common affine space, for example $\{\boldsymbol{x}|\boldsymbol{x}^T\mathbf{1} = 1\}$, then we can also make $\boldsymbol{W}^T\mathbf{1} = \mathbf{1}$, and $\boldsymbol{H}^T\mathbf{1} = \mathbf{1}$, which means columns of $\boldsymbol{Y}$ are *convex* combinations of columns of $\boldsymbol{W}$, transforming the problem into finding an *enclosing simplex* that contains all the column vectors of $\boldsymbol{Y}$. Notice that finding the vertices of a polytope can be formulated as a set of convex feasibility problems, and if all the vertices happen to be linearly independent, then we can simply use them as the columns of $\boldsymbol{W}$, and then solve for $\boldsymbol{H}$ very easily. The condition under which this conceptually simple procedure succeeds is, not surprisingly, exactly the *separability* condition Donoho *et al.* first proposed for NMF [2]. Indeed, what the procedure implies is that all the columns of $\boldsymbol{W}$ can be sought from the columns of $\boldsymbol{Y}$. Depending on the application, this *separability* condition may also be termed "pure-pixel" in hyperspectral unmixing [39, 40], "anchor word" in document topic modeling [41], and "local dominance" in speech separation [9], to name just a few.

Many algorithms have been proposed to tackle this index-picking problem, and they can be in general divided into two categories—linear programming based methods and greedy pursuit based methods. Realizing that determining if a point is a vertex of the

Algorithm 3.1: Successive projection algorithm (SPA)

Input: $\boldsymbol{Y}$, k

$\boldsymbol{\Sigma} = \text{Diag}\left\{\boldsymbol{Y}^T\mathbf{1}\right\}$;
$\boldsymbol{X} = \boldsymbol{Y}\boldsymbol{\Sigma}^{-1}$(normalization);
$\Lambda = \emptyset$;
for $\kappa = 1, ..., k$ **do**
  $j_\kappa \leftarrow \arg\min_{j\in\{1,...,n\}} \|\boldsymbol{X}(:,j)\|_2$;
  $\Lambda \leftarrow [\Lambda, j_\kappa]$;
  $\boldsymbol{\Theta} \leftarrow \arg\min_{\boldsymbol{\Theta}\geq 0} \|\boldsymbol{X} - \boldsymbol{X}(:,\Lambda)\boldsymbol{\Theta}\|_F^2$;
  $\boldsymbol{X} \leftarrow \boldsymbol{X} - \boldsymbol{X}(:,\Lambda)\boldsymbol{\Theta}$;
end
$\boldsymbol{W} = \boldsymbol{Y}(:,\Lambda)$;
$\boldsymbol{H} = \arg\min_{\boldsymbol{H}\geq 0} \|\boldsymbol{Y} - \boldsymbol{Y}(:,\Lambda)\boldsymbol{H}\|_F^2$;

convex hull of a set of points is equivalent to a linear feasibility problem, LP-based methods [42, 43] take advantage of this hidden convexity rendered by the separability assumption, while at the same time try to avoid going through the whole dataset one by one. Greedy methods, on the other hand, replace the step of solving a full-blown linear programming problem with a simple projection so that the approach is much more efficient on big data sets. Almost all greedy pursuit based methods come from the so-called *successive projection algorithm* (SPA) [44] that is presented in Algorithm 3.1. SPA-like algorithms first define a normalized matrix $\boldsymbol{X} = \boldsymbol{Y}\boldsymbol{\Sigma}^{-1}$ so that each column of the normalized data $\boldsymbol{X}$ sum to one. Note that $\boldsymbol{X} = \boldsymbol{A}\boldsymbol{\Theta}$ where $\boldsymbol{a}_\kappa = \boldsymbol{w}_\kappa / \boldsymbol{w}_\kappa^T\mathbf{1}$ and

$$\boldsymbol{\Theta}(\kappa, j) = \frac{\boldsymbol{H}(j,\kappa)\boldsymbol{w}_\kappa^T\mathbf{1}}{\boldsymbol{h}[j]^T\mathbf{1}\boldsymbol{w}_\kappa^T\mathbf{1}}.$$

Consequently, we have $\boldsymbol{\Theta}^T\mathbf{1} = \mathbf{1}$ if $\boldsymbol{H} \geq 0$, meaning the columns of $\boldsymbol{X}$ all lie on the simplex spanned by the columns of $\boldsymbol{A}$. Also, the columns of $\boldsymbol{\Theta}$ all lie in the unit simplex. After normalization, SPA sequentially identifies the vertices of the data simplex, in conjunction with a deflation procedure.

Both LP-based methods and SPA-based methods rely strongly on the separability assumption, and will fail miserably if it is violated. This in practice usually means when the number of latent factors is relatively small. As the number of latent components

becomes larger, separability becomes harder to satisfy, thus the chances that SPA fails becomes higher.

As early as 1994, Craig conjectured that even in the absence of "pure-pixels", the sources can still be uniquely identified, as long as the mixtures are "sufficiently scattered" in the convex hull of all the sources [45], via searching for the smallest simplex that encloses all the data points. To quantify "smallest", the notion of volume is used. In $\mathbb{R}^d$, a simplex with solid interior can be defined by its $d+1$ vertices $\boldsymbol{x}_0, ..., \boldsymbol{x}_d$, and its volume is defined as

$$\begin{aligned}\text{vol}\left(\boldsymbol{x}_0, ..., \boldsymbol{x}_d\right) &= \frac{1}{d!}\left|\det\begin{bmatrix}\boldsymbol{x}_0 & ... & \boldsymbol{x}_d \\ 1 & ... & 1\end{bmatrix}\right| \\ &= \frac{1}{d!}|\det[\ (\boldsymbol{x}_1 - \boldsymbol{x}_0)\ ...\ (\boldsymbol{x}_d - \boldsymbol{x}_0)\ ]|.\end{aligned}$$

Coming back to our model, assume that the columns of $\boldsymbol{Y}$ are scaled to lie on a common affine space, which can be defined by the affine hull of the columns of $\boldsymbol{W}$. Suppose $\boldsymbol{W}$ is square, then we can use the volume of the simplex defined by the columns of $\boldsymbol{W}$ and the origin as the criterion for "small" in Craig's approach, i.e., $|\det \boldsymbol{W}|$. Notice that

$$|\det \boldsymbol{W}| = \sqrt{\det \boldsymbol{W}^T\boldsymbol{W}},$$

$\det \boldsymbol{W}^T\boldsymbol{W}$ is a monotonic mapping of the volume $|\det \boldsymbol{W}|$, which also generalizes to the case when $\boldsymbol{W}$ is tall. We therefore formalize (one form of) the volume minimization-based matrix factorization as follows

$$\begin{aligned}\underset{\boldsymbol{W},\boldsymbol{H}}{\text{minimize}} \quad & \det \boldsymbol{W}^T\boldsymbol{W} \\ \text{subject to} \quad & \boldsymbol{Y} = \boldsymbol{W}\boldsymbol{H}^T, \boldsymbol{H}\mathbf{1} = \mathbf{1}, \boldsymbol{H} \geq 0.\end{aligned} \tag{3.1}$$

Craig's criterion turns out to be a useful one in searching for the correct sources blindly in the absence of separability. Part of the reason is that it, as a sanity check, is able to recover the true latent factors if separability is indeed satisfied. However, for decades this has been the best theoretical result people can show, despite that empirically it is known to be much more powerful than mere separability assumption. For completeness, we provide a simple proof here, following the lines of [46].

Theorem 3.1. *Denote an optimal solution of (3.1) as $\tilde{\boldsymbol{W}}, \tilde{\boldsymbol{H}}$. If the true latent factor $\boldsymbol{H}$ is separable and each of its rows sums to one, then there exists a permutation matrix*

$\boldsymbol{P}$ such that $\tilde{\boldsymbol{H}}\boldsymbol{P} = \boldsymbol{H}$. In other words, $\boldsymbol{W}$ and $\boldsymbol{H}$ can be identified up to permutation via solving (3.1).

Proof. By contradiction. Suppose $\tilde{\boldsymbol{W}}$ and $\tilde{\boldsymbol{H}}$ is an optimal solution of (3.1), but $\tilde{\boldsymbol{H}}$ is not a column permutation of $\boldsymbol{H}$. Since $\boldsymbol{W}$ and $\boldsymbol{H}$ are clearly feasible for (3.1), this means that $\det \tilde{\boldsymbol{W}}^T\tilde{\boldsymbol{W}} \leq \det \boldsymbol{W}^T\boldsymbol{W}$.

Since the columns of $\boldsymbol{H}$ and $\tilde{\boldsymbol{H}}$ span the same subspace, there is a $k \times k$ matrix $\boldsymbol{A}$ such that

$$\tilde{\boldsymbol{H}} = \boldsymbol{H}\boldsymbol{A},$$

and since $\boldsymbol{Y} = \boldsymbol{W}\boldsymbol{H}^T = \tilde{\boldsymbol{W}}\tilde{\boldsymbol{H}}^T$, this means

$$\tilde{\boldsymbol{W}} = \boldsymbol{W}\boldsymbol{A}^{-T}.$$

This means

$$\det \tilde{\boldsymbol{W}}^T\tilde{\boldsymbol{W}} = |\det \boldsymbol{A}|^{-2} \det \boldsymbol{W}^T\boldsymbol{W} \leq \det \boldsymbol{W}^T\boldsymbol{W},$$

i.e.,

$$|\det \boldsymbol{A}| \geq 1. \tag{3.2}$$

Because $\boldsymbol{H}$ is separable, this means rows of $\boldsymbol{A}$ appear in the rows of $\tilde{\boldsymbol{H}}$, and since $\tilde{\boldsymbol{H}}$ is feasible for (3.1), this means

$$\boldsymbol{A} \geq 0, \quad \boldsymbol{A}\mathbf{1} = \mathbf{1}.$$

Then we have the following chain

$$\begin{aligned}
|\det \boldsymbol{A}| &\leq \prod_{i=1}^{k} \|\boldsymbol{a}_i\| && \text{(3.3a)}\\
&\leq \prod_{i=1}^{k} \|\boldsymbol{a}_i\|_1 && \text{(3.3b)}\\
&\leq \prod_{i=1}^{k} \mathbf{1}^T\boldsymbol{a}_i && \text{(3.3c)}\\
&\leq \left(\frac{\sum_{i=1}^{k} \mathbf{1}^T\boldsymbol{a}_i}{k}\right)^k && \text{(3.3d)}\\
&= \left(\frac{\mathbf{1}^T\boldsymbol{A}\mathbf{1}}{k}\right)^k && \text{(3.3e)}\\
&= 1, && \text{(3.3f)}
\end{aligned}$$

where (3.3a) is Hadamard's inequality, (3.3b) comes from elementary properties of vector norms, (3.3c) is because $\boldsymbol{A} \geq 0$, and (3.3d) is by the arithmetic-geometric mean inequality.

Combining (3.2) and (3.3), we conclude that $|\det \boldsymbol{A}| = 1$. Then, all the inequalities in (3.3) hold as equality, and specifically (3.3b) means that the cardinalities of all the rows of $\boldsymbol{A}$ are equal to one. Together with the fact that each row of $\boldsymbol{A}$ sum up to one, this means $\boldsymbol{A}$ can only be a permutation matrix. In other words, $\tilde{\boldsymbol{W}}$ is a column permutation of $\boldsymbol{H}$. □

As a sneak peek to the forthcoming results, we will see how a much more relaxed condition can lead to the same identifiability result—the key observation is that the proof can be carried out in almost exactly the same steps, except that (3.3b) is removed. As it turns out, the following inequality (3.3c) looks very related to the definition of the second-order-cone $\mathcal{C}^*$ defined in our proposed sufficiently scattered assumption.

3.2 Uniqueness: Main Results

According to the previously discussed convex geometry, by dropping the separability condition on $\boldsymbol{H}$, one cannot identify the columns of $\boldsymbol{W}$ by merely searching in the columns of $\boldsymbol{Y}$. We therefore resort to Craig's criterion to try to identify the true sources, assuming the mixtures are "sufficiently scattered" in the convex hull of all the sources [45], via searching for the smallest simplex that encloses all the data points. As it turns out, the conjecture is correct, and this was first proven in our work *et al.* [9], and later by Lin *et al.* [47]. The precise condition that eluded Craig is the sufficiently scattered condition in definition 2.5, reproduced below for convenience.

Definition (sufficiently scattered)**.** *A non-negative matrix $\boldsymbol{X}$ is sufficiently scattered if*

1. $\operatorname{cone}\left\{\boldsymbol{X}^T\right\} \supseteq \mathcal{C}$,
2. $\operatorname{cone}\left\{\boldsymbol{X}^T\right\}^* \cap \operatorname{bd}\left\{\mathcal{C}^*\right\} = \{\lambda \boldsymbol{e}_\kappa \mid \lambda \geq 0, \kappa = 1, ..., k.\}$,

where $\mathcal{C}$ and $\mathcal{C}^*$ are

$$\mathcal{C} = \{\boldsymbol{x} | \boldsymbol{x}^T \mathbf{1} \geq \sqrt{k-1}\|\boldsymbol{x}\|_2\}, \quad \mathcal{C}^* = \{\boldsymbol{x} | \boldsymbol{x}^T \mathbf{1} \geq \|\boldsymbol{x}\|_2\}.$$

Theorem 3.2. *Denote an optimal solution of (3.1) as $\tilde{\boldsymbol{W}}, \tilde{\boldsymbol{H}}$. If the true latent factor $\boldsymbol{H}$ is sufficiently scattered (cf. Definition 2.5) and each of its rows sums to one, then there exists a permutation matrix $\boldsymbol{P}$ such that $\tilde{\boldsymbol{H}}\boldsymbol{P} = \boldsymbol{H}$. In other words, $\boldsymbol{W}$ and $\boldsymbol{H}$ can be identified up to permutation via solving (3.1).*

Theorem 3.2 can be proven easily using the following lemma, which is proven in the appendix.

Lemma 3.1. *Suppose the matrix $\boldsymbol{H}$ satisfies that $\boldsymbol{H} \geq 0$, $\boldsymbol{H}\mathbf{1} = \mathbf{1}$, and is sufficiently scattered. If a $k \times k$ matrix $\boldsymbol{A}$ satisfies that*

$$\boldsymbol{H}\boldsymbol{A} \geq 0, \boldsymbol{H}\boldsymbol{A}\mathbf{1} = \mathbf{1},$$

then we have that $|\det \boldsymbol{A}| \leq 1$. Furthermore, equality holds if and only if $\boldsymbol{A}$ is a permutation matrix.

Proof of Theorem 3.2. By contradiction. Suppose $\tilde{\boldsymbol{W}}$ and $\tilde{\boldsymbol{H}}$ is an optimal solution of (3.1), but $\tilde{\boldsymbol{H}}$ is not a column permutation of $\boldsymbol{H}$, the true latent factor. Since $\boldsymbol{W}$ and $\boldsymbol{H}$ are clearly feasible for (3.1), this means that $\det \tilde{\boldsymbol{W}}^T\tilde{\boldsymbol{W}} \leq \det \boldsymbol{W}^T\boldsymbol{W}$.

Since the columns of $\boldsymbol{H}$ and $\tilde{\boldsymbol{H}}$ span the same subspace, there is a $k \times k$ matrix $\boldsymbol{A}$ such that

$$\tilde{\boldsymbol{H}} = \boldsymbol{H}\boldsymbol{A},$$

and since $\boldsymbol{Y} = \boldsymbol{W}\boldsymbol{H}^T = \tilde{\boldsymbol{W}}\tilde{\boldsymbol{H}}^T$, this means

$$\tilde{\boldsymbol{W}} = \boldsymbol{W}\boldsymbol{A}^{-T}.$$

Since $\boldsymbol{H}$ is sufficiently scattered, according to Lemma 3.1, and our first assumption that $\boldsymbol{A}$ is not a permutation matrix, we have

$$|\det \boldsymbol{A}| < 1.$$

However, the optimal objective of (3.1) obtained by $\tilde{\boldsymbol{W}}, \tilde{\boldsymbol{H}}$ is

$$\begin{aligned}
\det \tilde{\boldsymbol{W}}^T\tilde{\boldsymbol{W}} &= \det \boldsymbol{A}^{-1}\boldsymbol{W}^T\boldsymbol{W}\boldsymbol{A}^{-T} \\
&= \det \boldsymbol{A}^{-1} \det \boldsymbol{W}^T\boldsymbol{W} \det \boldsymbol{A}^{-T} \\
&= |\det \boldsymbol{A}|^{-2} \det \boldsymbol{W}^T\boldsymbol{W} \\
&> \det \boldsymbol{W}^T\boldsymbol{W},
\end{aligned}$$

which contradicts our first assumption that $\tilde{\boldsymbol{W}}, \tilde{\boldsymbol{H}}$ is an optimal solution for (3.1). Therefore, $\tilde{\boldsymbol{H}}$ must be a column permutation of $\boldsymbol{H}$. □

In the special case when $\boldsymbol{W}$ is square, an alternative formulation to (3.12) is to re-parameterize $\boldsymbol{X} = \boldsymbol{W}^{-1}$, and then eliminate variable $\boldsymbol{H}$, yielding

$$\begin{aligned} \underset{\boldsymbol{X}}{\text{maximize}} \quad & |\det \boldsymbol{X}| \\ \text{subject to} \quad & \boldsymbol{X}\boldsymbol{Y} \geq 0, \boldsymbol{Y}^T \boldsymbol{X}^T \mathbf{1} = \mathbf{1}. \end{aligned} \tag{3.4}$$

A similar result holds for this case.

Proposition 3.1. *Denote an optimal solution of (3.4) as $\tilde{\boldsymbol{X}}$. If the true latent factor $\boldsymbol{H}$ is sufficiently scattered (cf. Definition 2.5) and each of its rows sums to one, then there exists a permutation matrix $\boldsymbol{P}$ such that $\boldsymbol{Y}^T \tilde{\boldsymbol{X}}^T \boldsymbol{P} = \boldsymbol{H}$.*

Proof. The proof is by contradiction, and very similar to that of Theorem 3.2. Let $\tilde{\boldsymbol{X}}$ be an optimal solution of (3.4), but $\boldsymbol{Y}^T \tilde{\boldsymbol{X}}^T$ is not a column permutation of $\boldsymbol{H}$, since $\boldsymbol{X} = \boldsymbol{W}^{-1}$ is clearly feasible for (3.4), this means that $|\det \tilde{\boldsymbol{X}}| \geq |\det \boldsymbol{W}^{-1}|$.

Since the columns of $\boldsymbol{Y}^T \tilde{\boldsymbol{X}}^T$ and $\boldsymbol{H}$ span the same subspace, there is a $k \times k$ matrix $\boldsymbol{A}$ such that

$$\boldsymbol{Y}^T \tilde{\boldsymbol{X}}^T = \boldsymbol{H}\boldsymbol{A},$$

or equivalently

$$\tilde{\boldsymbol{X}} = \boldsymbol{A}^T \boldsymbol{W}^{-1}.$$

Since $\boldsymbol{H}$ is sufficiently scattered, according to Lemma 3.1, and our first assumption that $\boldsymbol{A}$ is not a permutation matrix, we have

$$|\det \boldsymbol{A}| < 1.$$

However, the optimal objective of (3.4) obtained by $\tilde{\boldsymbol{X}}$ is

$$\begin{aligned} |\det \tilde{\boldsymbol{X}}| &= |\det \boldsymbol{A}^T \boldsymbol{W}^{-1}| \\ &= |\det \boldsymbol{A}||\det \boldsymbol{W}^{-1}| \\ &< |\det \boldsymbol{W}^{-1}|, \end{aligned}$$

which contradicts our first assumption that $\tilde{\boldsymbol{X}}$ is an optimal solution for (3.4). Therefore, $\boldsymbol{Y}^T \tilde{\boldsymbol{X}}^T$ must be a column permutation of $\boldsymbol{H}$. □

The aforementioned analysis is based on convex geometry, which is insightful and intuitive – the downside though is that to apply convex geometry we need to first rescale the columns of $\boldsymbol{Y}$ to lie on a common affine space. This rescaling step not only seems redundant in terms of analysis, but it may also color the noise for real data, making the estimation problem harder to solve. In the sequel, we will show that it is indeed not necessary to rescale the data before applying VolMin matrix factorization techniques, and the nice identifiability results still hold in this case, with an even slightly simpler proof. The only thing we miss then is the geometric interpretation of the problem, which is conceptually nice but not needed for practical applications.

Recall that for a matrix factorization $\boldsymbol{Y} = \boldsymbol{W}\boldsymbol{H}^T$, we can always put a diagonal matrix and its inverse in-between $\boldsymbol{Y} = \boldsymbol{W}\boldsymbol{D}^{-1}\boldsymbol{D}\boldsymbol{H}^T$. Therefore we can, without loss of generality, assume that the *columns* of $\boldsymbol{H}$ sum up to one. In contrast, assuming the rows of $\boldsymbol{H}$ sum to one implicitly assumes that the columns of $\boldsymbol{Y}$ lie on a common affine space, which then requires some pre-processing step to make it happen. The pre-process-free formulation can then be written as

$$\begin{aligned} \underset{\boldsymbol{W},\boldsymbol{H}}{\text{minimize}} \quad & \det \boldsymbol{W}^T\boldsymbol{W} \\ \text{subject to} \quad & \boldsymbol{Y} = \boldsymbol{W}\boldsymbol{H}^T, \boldsymbol{H}^T\mathbf{1} = \mathbf{1}, \boldsymbol{H} \geq 0, \end{aligned} \tag{3.5}$$

and we have a similar result.

Theorem 3.3. *Denote an optimal solution of (3.5) as $\tilde{\boldsymbol{W}}, \tilde{\boldsymbol{H}}$. If the true latent factor $\boldsymbol{H}$ is sufficiently scattered (cf. Definition 2.5) and each of its columns sums to one, then there exists a permutation matrix $\boldsymbol{P}$ such that $\tilde{\boldsymbol{H}}\boldsymbol{P} = \boldsymbol{H}$. In other words, $\boldsymbol{W}$ and $\boldsymbol{H}$ can be identified up to permutation via solving (3.5).*

The proof is almost exactly the same as the one for Theorem 3.2 – the only difference is that instead of using Lemma 3.1, we need to use Lemma 3.2, which is given below. The proof of Lemma 3.2 can be found in the appendix. We omit the rest of the proof of Theorem 3.3 for brevity.

Lemma 3.2. *Suppose the matrix $\boldsymbol{H}$ satisfies that $\boldsymbol{H} \geq 0$, $\boldsymbol{H}^T\mathbf{1} = \mathbf{1}$, and is sufficiently scattered. If a $k \times k$ matrix $\boldsymbol{A}$ satisfies that*

$$\boldsymbol{H}\boldsymbol{A} \geq 0, \boldsymbol{A}^T\boldsymbol{H}^T\mathbf{1} = \mathbf{1},$$

then we have that $|\det \boldsymbol{A}| \leq 1$. *Furthermore, equality holds if and only if* $\boldsymbol{A}$ *is a permutation matrix.*

When $\boldsymbol{W}$ is square, again we can define $\boldsymbol{X} = \boldsymbol{W}^{-1}$ and work with the alternative formulation

$$\begin{aligned} \underset{\boldsymbol{X}}{\text{maximize}} \quad & |\det \boldsymbol{X}| \\ \text{subject to} \quad & \boldsymbol{X}\boldsymbol{Y} \geq 0, \boldsymbol{X}\boldsymbol{Y}\mathbf{1} = \mathbf{1}. \end{aligned} \tag{3.6}$$

Proposition 3.2. *Denote an optimal solution of (3.6) as* $\tilde{\boldsymbol{X}}$. *If the true latent factor* $\boldsymbol{H}$ *is sufficiently scattered(cf. Definition 2.5) and each of its columns sums to one, then there exists a permutation matrix* $\boldsymbol{P}$ *such that* $\boldsymbol{Y}^T\tilde{\boldsymbol{X}}^T\boldsymbol{P} = \boldsymbol{H}$.

The proof of Proposition 3.2 follows the sames steps of that of Proposition 3.1, except using Lemma 3.2 instead of Lemma 3.1, thus omitted here.

3.3 Application: Principled Neuro-Functional Connectivity Discovery

How can we reverse-engineer the brain connectivity, given the input stimulus, and the corresponding brain-activity measurements, for several experiments? We show how to solve the problem in a principled way, modeling the brain as a linear dynamical system (LDS), and solving the resulting "system identification" problem after imposing sparsity and non-negativity constraints on the appropriate matrices. These are reasonable assumptions in some applications, including magnetoencephalography (MEG).

In computational neuroscience, one of the major research challenges is estimating the *functional connectivity* of the brain, i.e. a relation between neurons (or groups of neurons) which encodes co-activation of the neurons involved. Functional connectivity is often determined via cross-correlation or mutual information statistics [48, 49], albeit these approaches do not explicitly model neuronal state dynamics. An introductory overview of techniques for estimating brain connectivity can be found in [50].

Here, we want to estimate the functional connectivity of the brain, under the following experimental regime: a human subject is presented with a stimulus (in particular concrete nouns of the English language) as well as a task (such as answering a simple question regarding the shown noun, like *is it alive?*, *can you buy it?* and so on), and

their brain activity is measured [1] over a course of a few seconds. Recently, [51] proposed the "GeBM" model, a simple model for the brain, that successfully captures the temporal dynamics of the brain. The "GeBM" model is as follows:

$$\begin{aligned} \boldsymbol{x}(t+1) &= \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t), \\ \boldsymbol{y}(t) &= \boldsymbol{C}\boldsymbol{x}(t), \end{aligned} \tag{3.7}$$

where $\boldsymbol{u}(t)$ is the stimulus signal, $\boldsymbol{y}(t)$ is the observed brain activity measured via MEG, $\boldsymbol{x}(t)$ is the latent brain activity, $\boldsymbol{A}$ is the functional connectivity matrix, $\boldsymbol{B}$ is the stimulus matrix, and $\boldsymbol{C}$ models the measurement that maps the internal state of the brain into MEG sensor values. In the original paper, the "GeBM" model is solved using "system identification" from control theory, and an ad-hoc, greedy method, to sparsify the connectivity matrix $\boldsymbol{A}$.

In this work [12], we formalize the problem, and provide a principled and theoretically sound treatment of sparse system identification under an additional non-negativity condition on $\boldsymbol{C}$, with application to brain functional connectivity estimation. Our main contributions are the following: (a) a *rigorous proof* of identifiability for the constrained problem we propose; (b) an effective, two-stage *algorithm*; and (c) *validation* of both, using semi-synthetic and real (MEG) data.

Figure 3.1 shows an illustration of our results on real MEG data. The left shows the recovered graph, and the right part shows the corresponding adjacency matrix. Notice that there are several 'white' (= empty) cells, exactly because our algorithm enforces sparsity. See the experiments section for more details.

State-space and, more generally, dynamic latent variable (e.g., Markov) models have a long history across science, and neural data analysis in particular [52, 53]. Even the simplest dynamic models, though, can only be identified up to inherent indeterminacies which generally hide the underlying connectivity pattern. For the linear state-space model in (3.7), for example, this comes in the form of a simplicity transformation that alters the structure of $\boldsymbol{A}$. Such indeterminacy is inherent to the model in (3.7), and is also borne out of classical subspace-based system identification methods [54, 55], which

[1] Measurements may be taken either via Magnetoencephalography (MEG) or functional magnetic resonance imaging (fMRI), although the former offers finer temporal granularity and is preferred when fine grained temporal dynamics are considered.

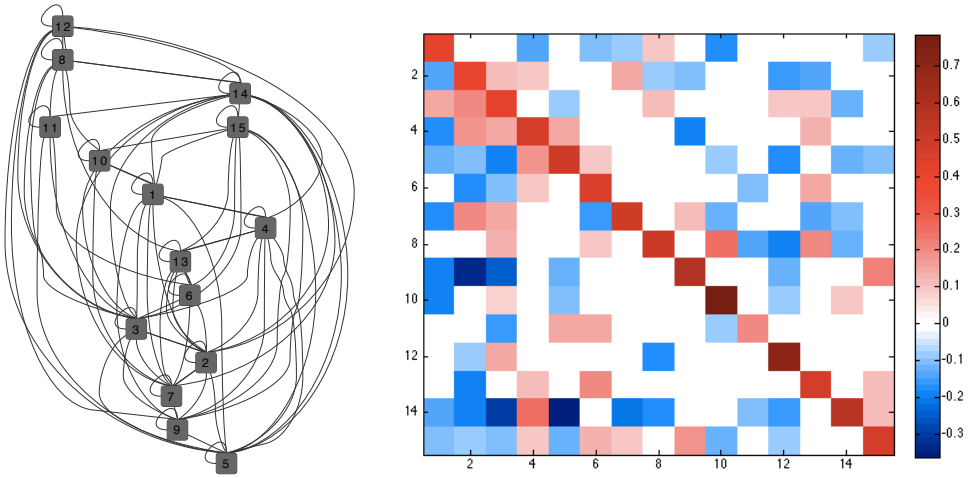


Figure 3.1: The neural connectivity (left), and its corresponding adjacency matrix (right), obtained from real MEG data.

can only provide $\hat{\boldsymbol{A}}$, $\hat{\boldsymbol{B}}$, $\hat{\mathbf{C}}$ satisfying

$$\begin{aligned}\hat{\boldsymbol{A}} &= \boldsymbol{M}\boldsymbol{A}\boldsymbol{M}^{-1},\\ \hat{\boldsymbol{B}} &= \boldsymbol{M}\boldsymbol{B},\\ \hat{\boldsymbol{C}} &= \boldsymbol{C}\boldsymbol{M}^{-1},\end{aligned}$$

for some unknown non-singular matrix $\boldsymbol{M}$, due to rotational freedom. The mapping from $(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}) \rightarrow (\hat{\boldsymbol{A}}, \hat{\boldsymbol{B}}, \hat{\boldsymbol{C}})$ is known as a *similarity transformation*. A key message of this paper is that sparsity and non-negativity of $\boldsymbol{C}$ can overcome this limitation and render not only $\boldsymbol{C}$, but also $\boldsymbol{A}$ and $\boldsymbol{B}$ identifiable without rotational ambiguity (except the unavoidable permutation and scaling).

For MEG sensors, the assumption that $\boldsymbol{C}$ is non-negative and sparse can be motivated as follows. The brain activity recorded by MEG is limited to very low frequencies, typically ≤ 30 Hz [56]. Since the corresponding wavelength is so much larger than the size of a skull, the spatial phase variation of the magnetic wave from one sensor to the next is insignificant, i.e., the MEG sensors are approximately *in phase*, hence $\boldsymbol{C}$ can be assumed non-negative. Furthermore, since field intensity decays very fast as the distance between the source and the sensor increases, a MEG sensor only measures

(diffuse) sources that are close to it, leading to a sparse $\boldsymbol{C}$. Similar arguments can be made for electroencephalography (EEG) [57].

Notation We denote the true system matrices as $\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{C}$. The matrices $\hat{\boldsymbol{A}}$, $\hat{\boldsymbol{B}}$, and $\hat{\boldsymbol{C}}$ represent the results obtained from the subspace method, i.e., they are unconstrained. If we take the constraints into consideration, the estimates are denoted $\tilde{\boldsymbol{A}}$, $\tilde{\boldsymbol{B}}$, and $\tilde{\boldsymbol{C}}$. Generally speaking, when we study identifiability, we compare $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$ with $\tilde{\boldsymbol{A}}$, $\tilde{\boldsymbol{B}}$, $\tilde{\boldsymbol{C}}$, and when we design algorithms, we use $\hat{\boldsymbol{A}}$, $\hat{\boldsymbol{B}}$, $\hat{\boldsymbol{C}}$, and the constraints to obtain $\tilde{\boldsymbol{A}}$, $\tilde{\boldsymbol{B}}$, and $\tilde{\boldsymbol{C}}$.

Since linear dynamical systems can generally only be identified up to a similarity transformation, special transformations can be used to bring the system matrices into certain convenient forms. In the controls field, $\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{C}$ are often put into *canonical forms*, such as the controller forms or the observer forms [58, Sec. 3.4], in which case the structure of $\boldsymbol{A}$ is confined to a specific pattern. Since our main purpose here is to *discover* the underlying structure of the true $\boldsymbol{A}$, *imposing* structure through transformation to some canonical form is generally inappropriate.

One interesting difference between our model and the classical controls literature is that the dimension of the output is larger than the number of states[2] . In other words, the matrix $\boldsymbol{C}$ is tall. Moreover, because of the nature of MEG sensors, we can also assume that $\boldsymbol{C}$ is sparse and takes only non-negative values. In this section, we will first propose a condition under which a non-negative $\boldsymbol{C}$ is identifiable, and then study the connection between sparsity, non-negativity, and the proposed condition.

Throughout this paper we will assume that $\boldsymbol{C}^T\mathbf{1} = \mathbf{1}$, because otherwise we can scale the columns of $\boldsymbol{C}$ to satisfy that, which will result in a similarity transformation with a diagonal matrix, thus does not change the structure of the system.

Since we are given $\hat{\boldsymbol{C}}$, which we know is a transformed version of a non-negative matrix $\boldsymbol{C}$, and its columns are scaled to sum up to 1, we should be able to find a matrix $\boldsymbol{M}$ that satisfies

$$\hat{\boldsymbol{C}}\boldsymbol{M} \geq 0, \quad \boldsymbol{M}^T\hat{\boldsymbol{C}}^T\mathbf{1} = \boldsymbol{M}^T\mathbf{1} = \mathbf{1}.$$

In fact, there are clearly infinitely many $\boldsymbol{M}$ that satisfy that. Therefore, we can set up a criterion and try to find the one with the maximum $|\det \boldsymbol{M}|$ (one can relate this idea

[2] As we discovered from rank analysis of experimental MEG data.

to SVM, in which case there are infinitely many linear separators and we seek for the one with the maximum "margin"), i.e.,

$$\begin{aligned} \underset{\boldsymbol{M}}{\text{maximize}} \quad & |\det \boldsymbol{M}|, \\ \text{subject to} \quad & \hat{\boldsymbol{C}}\boldsymbol{M} \geq 0, \quad \boldsymbol{M}^T\mathbf{1} = \mathbf{1}, \end{aligned} \tag{3.8}$$

Geometrically, (3.8) tries to find the simplicial cone that contains all the row vectors of $\hat{\boldsymbol{C}}$, and with minimum "volume" [40]. Next, we will propose a condition under which the true $\boldsymbol{C}$ can be recovered by solving (3.8) and then set the estimate as $\hat{\boldsymbol{C}}\boldsymbol{M}$.

As we can see, formulation (3.8) is exactly as we have discussed in (3.6), which, according to Theorem 3.2, is able to identify the correct $\boldsymbol{C}$ up to column permutation, as long as $\boldsymbol{C}$ is non-negative and reasonably sparse.

3.3.1 Proposed Method

Theoretically, if $\boldsymbol{C}$ is sparse, non-negative and tall, in which case sufficiently scatteredis satisfied with high probability, it is enough to only work on $\hat{\boldsymbol{C}}$ to figure out the true similarity transformation, thus successfully identifying the true $\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{C}$, up to permutation of the states. We will first introduce an algorithm that approximately solves (3.8), under a noiseless scenario. In practice, when the measurements are noisy, we found that the aforementioned formulation is very sensitive to noise. We therefore use a modified robust formulation, followed by a least-squares refinement procedure to make the resulting $\tilde{\boldsymbol{A}}$, $\tilde{\boldsymbol{B}}$, and $\tilde{\boldsymbol{C}}$ sparse (and non-negative, if/as appropriate).

The absolute value of the determinant of a non-symmetric matrix is proportional to the volume of a simplex defined by the columns of that matrix (and the origin) [40]. The objective function of (3.8) is non-convex, therefore (3.8) is presumably hard to solve. Two approaches that can be used to handle this type of non-convexity are successive linearization, and block coordinate descent—see [40] and the references therein. In our experiments we found that the block coordinate descent method works better in the noiseless case, therefore this method is briefly explained next.

If we fix all columns of $\boldsymbol{M}$ but one, the objective is linear over that column,

$$\det \boldsymbol{M} = \sum_{i=1}^{n} (-1)^{i+j} \boldsymbol{m}_j(i) \det \boldsymbol{M}_{i,j},$$

where $\boldsymbol{m}_j(i)$ means the i-th entry of the j-th column of $\boldsymbol{M}$, and $\boldsymbol{M}_{i,j}$ is obtained by deleting the i-th row and j-th column of $\boldsymbol{M}$. Thus, the update of one column of $\boldsymbol{M}$ becomes

$$\begin{aligned} \underset{\boldsymbol{m}_j}{\text{maximize}} \quad & \left| \sum_{i=1}^{n} (-1)^{i+j} \boldsymbol{m}_j(i) \det \boldsymbol{M}_{i,j} \right|, \\ \text{subject to} \quad & \hat{\boldsymbol{C}} \boldsymbol{m}_j \geq 0, \quad \boldsymbol{m}_j^T \mathbf{1} = \mathbf{1}. \end{aligned} \tag{3.9}$$

Now (3.9) is still non-convex, but we can get rid of the absolute value and solve two linear programming problems instead (maximizing and minimizing the linear objective function), then set $\boldsymbol{m}_j$ as the one that gives the larger absolute value.

If $\hat{\boldsymbol{C}}$ is not exactly a transformed version of $\boldsymbol{C}$, but includes some noise due to subspace estimation errors, one potential problem with formulation (3.8) is that it may not even be feasible. As a trade-off between the maximum determinant criterion and non-negativity of $\boldsymbol{C}$, we can use instead

$$\begin{aligned} \underset{\boldsymbol{M}}{\text{maximize}} \quad & \log |\det \boldsymbol{M}| - \lambda \sum_{i,j} [\hat{\boldsymbol{C}} \boldsymbol{M}]_-, \\ \text{subject to} \quad & \boldsymbol{M}^T \mathbf{1} = \mathbf{1}, \end{aligned} \tag{3.10}$$

where $[\cdot]_-$ sums the negative elements of its argument. We put a log to the determinant term because we found that otherwise, in order to make the second term large, the algorithm tends to make $\boldsymbol{M}$ singular, even if the regularization parameter λ is very small. By taking the log of the determinant term, it will decrease the objective function sharply when $\boldsymbol{M}$ is close to singular, while increasing it slowly when it is not. Algorithmically, we can still update $\boldsymbol{M}$ column by column, since the sub-problem can still be cast as two convex optimization problems.

Sparse refinement While our ultimate goal is to estimate $\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{C}$, our identifiability results show that a tall non-negative and sparse of $\boldsymbol{C}$ can be enough to guarantee identifiability, in the noiseless case. In practice, when the noiseless scenario is not realistic, what we observe is that while the robust formulation (3.10) is able to recover $\boldsymbol{C}$ with minor errors, the resulting $\tilde{\boldsymbol{A}}$ and $\tilde{\boldsymbol{B}}$ are not close to $\boldsymbol{A}$ and $\boldsymbol{B}$, even when the perturbation in the data is quite small. We therefore propose to refine the result from (3.10) by solving the problem given in (3.11), which also takes into account possible

sparsity in $\boldsymbol{A}$ and $\boldsymbol{B}$. In (3.11), the operation $\|\cdot\|_0$ returns the cardinality of the argument. Notice that we have introduced an auxiliary variable $\boldsymbol{M}_{\text{inv}}$ and a penalty term $\lambda\|\boldsymbol{M}_{\text{inv}}\boldsymbol{M}-\mathbf{I}\|_F^2$ to make it close to $\boldsymbol{M}^{-1}$, instead of working directly with both $\boldsymbol{M}$ and $\boldsymbol{M}^{-1}$, which is presumably hard.

$$\begin{aligned}\underset{\substack{\tilde{\boldsymbol{A}},\tilde{\boldsymbol{B}},\tilde{\boldsymbol{C}},\\ \boldsymbol{M},\boldsymbol{M}_{\text{inv}}}}{\text{minimize}} \quad & \|\tilde{\boldsymbol{A}}-\boldsymbol{M}_{\text{inv}}\hat{\boldsymbol{A}}\boldsymbol{M}\|_F^2+\|\tilde{\boldsymbol{B}}-\boldsymbol{M}_{\text{inv}}\hat{\boldsymbol{B}}\|_F^2+\|\tilde{\boldsymbol{C}}-\hat{\boldsymbol{C}}\boldsymbol{M}\|_F^2,+\lambda\|\boldsymbol{M}_{\text{inv}}\boldsymbol{M}-\mathbf{I}\|_F^2 \\ \text{subject to} \quad & \|\tilde{\boldsymbol{A}}\|_0\le s_A,\quad \|\tilde{\boldsymbol{B}}\|_0\le s_B,\quad \|\tilde{\boldsymbol{C}}\|_0\le s_C,\quad \tilde{\boldsymbol{C}}\ge 0,\end{aligned} \tag{3.11}$$

This formulation is still non-convex; but it is amenable to block coordinate descent—the updates for $\boldsymbol{M}$ and $\boldsymbol{M}_{\text{inv}}$ are classical linear least-squares, whereas the updates for $\tilde{\boldsymbol{A}}$, $\tilde{\boldsymbol{B}}$, and $\tilde{\boldsymbol{C}}$ are simple projections. Although the cardinality constraints are not convex, the corresponding projections are very easy—we only need to keep the entries with largest (absolute) values and zero out the others. Note that there is no reason to use an l_1 norm surrogate of the cardinality constraints, as hard projection is in fact easier here, and the $\boldsymbol{M}_{\text{inv}}\boldsymbol{M}$ term makes the problem non-convex, regardless. Due to non-convexity, the block coordinate descent algorithm only guarantees that every limit point is a stationary point. This is why initializing it with the result of (3.10) is crucial.

Summary of the proposed method The method proposed for principled neurofunctional connectivity discovery (NFCD) is summarized in Alg. 3.2, which consists of three steps: i) A system identification step to get $\hat{\boldsymbol{A}}$, $\hat{\boldsymbol{B}}$, and $\hat{\boldsymbol{C}}$ from the input-output data $\boldsymbol{U}$ and $\boldsymbol{Y}$, and a given system order n; ii) a determinant maximization step as discussed in §3.3.1; and iii) a sparse refinement step as discussed in §3.3.1.

In the system identification step, an interesting observation is that our model has more outputs than states, unlike typical LDS models in automatic control where the number of outputs is smaller than the number of states. Having more outputs than states makes system identification easier. As described in the first step of Alg. 3.2, we only need to take the 'thin' SVD of the output samples, and then solve a linear least-squares problem. In line 3, $\boldsymbol{T}$ is a random $n\times n$ matrix—we found by simulations that this improves the conditioning. In line 4, $\hat{\boldsymbol{X}}_0$ is the first $N-1$ columns of $\hat{\boldsymbol{X}}$, and $\hat{\boldsymbol{X}}_1$ is the last $N-1$ columns of $\hat{\boldsymbol{X}}$, where N is the number of samples, i.e., the number of columns of $\hat{\boldsymbol{X}}$.

In the 3rd step of Alg.3.2, we used the (hard) thresholding operator $\mathcal{T}_t(\cdot)$ parameterized by t, and its non-negative version $\mathcal{T}_t^+(\cdot)$, which are defined as:

$$\mathcal{T}_t(z) = \begin{cases} z \text{ , if } |z| \geq t \\ 0 \text{ , else} \end{cases}, \quad \mathcal{T}_t^+(z) = \begin{cases} z \text{ , if } z \geq t \\ 0 \text{ , else} \end{cases}.$$

A brief discussion on the complexity of NFCD is useful at this point. For the MEG data that we consider in this paper, m (the input dimension) and p (the number of MEG sensors) are no more than a few hundreds, and, as shown in [51], for n ranging from 10 to 30 the LDS model is able to capture most of the brain dynamics. Therefore, steps 2 and 3 of NFCD are relatively small scaled. The only number that can possibly go large is N, the number of samples collected by the MEG sensors, which only comes into play in step 1. Notice that for both SVD and least-squares, complexity grows linearly in the large dimension (times the small dimension squared). Thus, even if N is very large, NFCD is able to scale well.

3.3.2 Experiments

We next present some numerical results to corroborate our theoretical claims and illustrate the robustness of our methods. The convex optimization sub-problems are solved by using CVX, a package for specifying and solving convex programs [**?**].

Synthetic data We start by experimenting with synthetically generated data, where we know $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$, and we can check whether our proposed method is able to recover them from input-output data. We begin by assuming that the system is noiseless, and simply use (3.8) without refinement. The true $\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{C}$ are generated randomly, and whether an entry is zero or not is determined by drawing from an i.i.d. Bernoulli distribution. The non-zeros entries of $\boldsymbol{A}$ and $\boldsymbol{B}$ are drawn from an i.i.d. Gaussian distribution, whereas the non-zeros of $\boldsymbol{C}$ are drawn from an i.i.d. exponential distribution, to ensure non-negativity of $\boldsymbol{C}$. Then $\boldsymbol{A}$ is scaled down by its spectral radius to ensure stability of the system, and the columns of $\boldsymbol{C}$ are scaled to sum up to 1. The inputs $\mathbf{u}(t), t = 1, \ldots, N$, as well as the initial state $\mathbf{x}(0)$, are generated from an i.i.d. Gaussian distribution. Then the inputs are sent into the system in (3.7) to obtain the outputs $\mathbf{y}(t), t = 1, \ldots, N$.

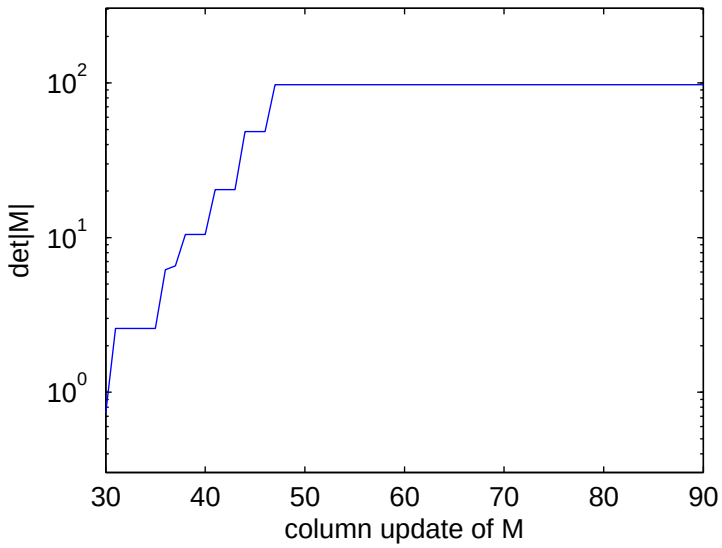


Figure 3.2: Column update of $\boldsymbol{M}$ for approximately solving (3.8).

As one particular example, with $n = 30$, $m = 50$, $p = 300$, and approximately 50% of the entries of $\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{C}$ being zero, 10^4 input-output pairs are used to do subspace system identification, and then the estimated $\hat{\boldsymbol{C}}$ is fed to (3.8). The convergence of the proposed block coordinate descent method is shown in Figure 3.2. Notice that the horizontal axis starts at 30 because $\boldsymbol{M}$ becomes feasible and non-singular only after the first round of column updates, therefore it is meaningless to show the objective before 30.

As shown in Figure 3.2, the algorithm converges very fast. In fact, considering that the first round of column updates tries to find a feasible $\boldsymbol{M}$, it converges even before the second round of column updates finishes. Let $\boldsymbol{M}_\star$ be the result obtained from solving (3.8); in this noiseless case we simply set

$$\begin{aligned}\tilde{\boldsymbol{A}} &= \boldsymbol{M}_\star^{-1}\hat{\boldsymbol{A}}\boldsymbol{M}_\star,\\ \tilde{\boldsymbol{B}} &= \boldsymbol{M}_\star^{-1}\hat{\boldsymbol{B}},\\ \tilde{\boldsymbol{C}} &= \hat{\boldsymbol{C}}\boldsymbol{M}_\star.\end{aligned}$$

Before we compare $\tilde{\boldsymbol{A}}$, $\tilde{\boldsymbol{B}}$, and $\tilde{\boldsymbol{C}}$ with the ground truth, we need to be aware that there

is still a permutation ambiguity, i.e., the similarity transformation can be a permutation matrix, which does not affect the true structure of the system, but only relabels the states. We resolve this by first matching the columns of $\tilde{\boldsymbol{C}}$ with $\boldsymbol{C}$, i.e.,

$$\min_{\mathbf{P}\in\Pi} \|\boldsymbol{C} - \tilde{\boldsymbol{C}}\mathbf{P}\|_F^2,$$

where Π indicates the set of permutation matrices. This problem can be cast as a linear assignment problem, which be solved optimally by the Hungarian method [59] [3] . After obtaining the best permutation $\mathbf{P}$, the rows and/or columns of $\tilde{\boldsymbol{A}}$, $\tilde{\boldsymbol{B}}$, and $\tilde{\boldsymbol{C}}$ are permuted accordingly.

In Table 3.1, we provide the normalized estimation error of the system matrices for various settings, where s indicates the ratio of nonzero entries. Sometimes the algorithm fails to generate a non-singular matrix, in which case a different initialization is used and the algorithm is run again. For each setting, 10 Monte-Carlo trials are performed, and we only show the largest error. In all cases, $m = 50$, and $p = 300$. As we can see, this simulation justifies the claim in Proposition 3.2 that sparse, non-negative and tall $\boldsymbol{C}$ yield an identifiable system.

Semi-synthetic data Next we try noisy data. Instead of synthetically generating the whole system, the $\boldsymbol{A}$ matrix we use here comes from real data – the neural connectivity of C. elegans[4] . Specifically, we take the connectivity of the first $10 \sim 20$ C. elegans neurons as the matrix $\boldsymbol{A}$ (those neurons are relatively more densely connected), again scaled down by its spectral radius to ensure stability. Then we synthetically generate $\boldsymbol{B}$ and $\boldsymbol{C}$, similar to the previous experiment. The inputs and the initial state of the system are generated as before, but now we introduce state and measurement noise, i.e.,

$$\begin{aligned}\mathbf{x}(t+1) &= \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) + \mathbf{v}(t),\\ \mathbf{y}(t) &= \mathbf{C}\mathbf{x}(t) + \mathbf{w}(t),\end{aligned}$$

where $\mathbf{v}(t)$ and $\mathbf{w}(t)$ are white Gaussian, with standard deviation $\sigma = 10^{-3}$. The matrices $\boldsymbol{B}$ and $\boldsymbol{C}$ are generated with $m = 50$, $p = 300$, and approximately 50% zeros. Then Algorithm 3.2 is applied to the input-output data. In step 2, we set $\lambda = 0.5$,

[3] A MATLAB inplementation of the Hungarian method is used and available at `http://www.mathworks.com/matlabcentral/fileexchange/11609-hungarian-algorithm`

[4] available at `http://www.wormatlas.org/neuronalwiring.html`.

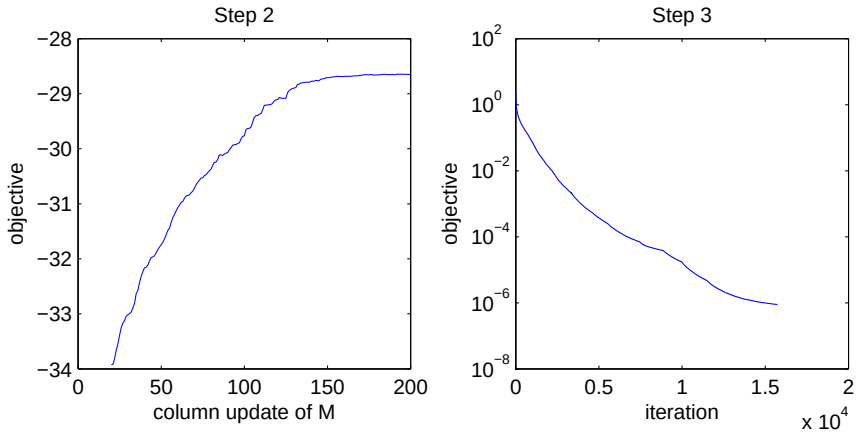


Figure 3.3: Convergence of Algorithm 3.2, step 2 (left) and step 3 (right).

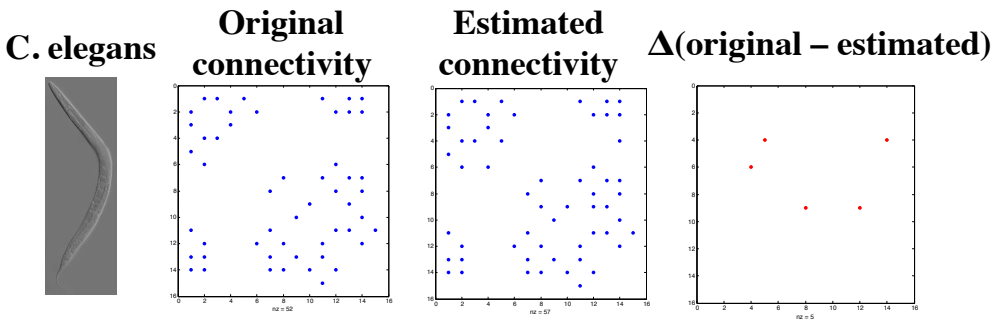


Figure 3.4: Recovery of C. elegans neural connectivity.

and in step 3, we set $\lambda = 100$. The cardinality constraints in step 3 are set to be approximately 10% more than the true density. For $n = 20$, the convergence of these two steps is shown in Figure 3.3.

Finally, the resulting $\tilde{\boldsymbol{A}}$, $\tilde{\boldsymbol{B}}$, and $\tilde{\boldsymbol{C}}$ are compared with the true system $\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{C}$, after column matching using the Hungarian method, and the normalized errors for different values of n are shown in Table 3.2. Notice that the recovery is almost perfect when $n = 10$. For $n = 15$, the estimated connectivity matrix is compared with the true connectivity in Figure 3.4, in which case we managed to recover all the true connectivity, with only a few redundant ones. In fact, if we set the sparsity constraint in (3.11) to be the exact one, the connectivity is recovered perfectly.

Real data Next we apply our proposed method to a set of real input-output data. The experiment was conducted by asking a yes/no question about a particular word to a human subject, and then his/her brain activities are measured by the 306 MEG sensors. Approximately 20 questions were asked for 60 words, and then 340 MEG measurements were collected for each particular question/word. We sampled 10 samples for each experiment, and also added 2 dimensions to indicate the time the subject responded to the question, and the answer given. This forms the output matrix $\boldsymbol{Y}$, with $p = 308$ and $N \approx 12000$. The input dimension $m = 40$, which is a subset of the 218-questions description of those 20 words conducted by Amazon Mechanical Turks. For more details on the dataset, see [51, 60].

As mentioned earlier, the LDS (Linear Dynamical System) modeling of the brain provides good input-output predictions. However, the ultimate goal is not simply to predict outputs, but also to study the functional connectivity of the brain. As we have argued at the beginning, for MEG sensors the measurement matrix $\boldsymbol{C}$ is non-negative and sparse, therefore satisfies sufficiently scatteredwith high probability. Using the identifiability results and the algorithm developed in this paper, we can analyze this dataset and see if we obtain interpretable results.

We tried this real input-output data using Algorithm 3.2 to fit a 15-state LDS, with the same λ values as in the previous simulation, assuming 50% sparsity of $\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{C}$. The regularization parameters in the optimization problem of step 2 and 3 are set equal to the previous simulation for the C. elegans data. The resulting $\tilde{\boldsymbol{A}}$ is represented as a graph to show the functional connectivity, in Figure 3.1 in the introduction. As expected, the (hidden) functional connectivity matrix obtained from the MEG experiments is quite sparse, and diagonally dominant. In lieu of ground truth data, we gain confidence in our model because of the fact that under our assumptions, our algorithms are able to recover a sparse functional connectivity matrix which successfully (and in a stable and robust manner) models MEG brain activity in the least squares sense. We omit the corresponding figures due to space restrictions, however, in our experiments we observed robust reconstruction of the MEG recorded brain activity using the obtained model.

3.3.3 Conclusion

Our goal is to solve the linear dynamical system (LDS) model of the brain by tackling the subtle, identifiability issue as well as the sparsity and non-negativity constraints, in a principled, effective way. Our contributions are the following:

- *Proof* that our proposed conditions resolve the identifiability issue.
- *Algorithm*: our two-stage algorithm is carefully designed. We give a robust problem reformulation when the data is noisy; and we propose a refinement step to sparsify the connectivity matrix.
- *Validation*, using real and synthetic data. For the semi-synthetic data, we used a subset of the neuro-connectivity of the C. elegans as the system to simulate a set of noisy input-output data, and managed to recover the true neuro-connectivity with high accuracy. On real data measured by MEG, our method produced interpretable results.

Appendix

Proof of Lemma 3.1

Proof. First of all, the tall matrix $\boldsymbol{H}$ has full column rank, so $\boldsymbol{H}\mathbf{1} = \mathbf{1}$ and $\boldsymbol{H}\boldsymbol{A}\mathbf{1} = \mathbf{1}$ implies that $\boldsymbol{A}\mathbf{1} = \mathbf{1}$.

Geometrically, $\boldsymbol{H}\boldsymbol{A} \geq 0$ means that $\boldsymbol{a}_i \in \operatorname{cone}\left\{\boldsymbol{H}^T\right\}^*$, where $\boldsymbol{a}_i$ is the i-column of $\boldsymbol{A}$, for all $i = 1, ..., k$. Because $\boldsymbol{H}$ is sufficiently scattered, we have that $\operatorname{cone}\left\{\boldsymbol{H}^T\right\} \subseteq \mathcal{C}^*$, which further implies that $\boldsymbol{a}_i \in \mathcal{C}^*$, which means

$$\|\boldsymbol{a}_i\| \leq \mathbf{1}^T \boldsymbol{a}_i.$$

Then we have the following chain

$$\begin{aligned}
|\det \boldsymbol{A}| &\leq \prod_{i=1}^{k} \|\boldsymbol{a}_i\| && \text{(3.12a)}\\
&\leq \prod_{i=1}^{k} \mathbf{1}^T \boldsymbol{a}_i && \text{(3.12b)}\\
&\leq \left(\frac{\sum_{i=1}^{k} \mathbf{1}^T \boldsymbol{a}_i}{k} \right)^k && \text{(3.12c)}\\
&= \left(\frac{\mathbf{1}^T \boldsymbol{A}\mathbf{1}}{k} \right)^k && \text{(3.12d)}\\
&= 1, && \text{(3.12e)}
\end{aligned}$$

where (3.12a) is Hadamard's inequality, (3.12b) is because $\boldsymbol{a}_i \in \mathcal{C}^*$, and (3.12c) is by the arithmetic-geometric mean inequality.

Now, suppose equality is attained, i.e., $|\det \boldsymbol{A}| = 1$, then all the inequalities in (3.12) hold as equality, and specifically (3.12b) means that the columns of $\boldsymbol{A}$ lie on the boundary of $\mathcal{C}^*$. Recall that $\boldsymbol{a}_i \in \operatorname{cone}\left\{\boldsymbol{H}^T\right\}^*$, and $\boldsymbol{H}$ being sufficiently scattered, according to the second requirement in Definition 2.5, shows that

$$\operatorname{cone}\left\{\boldsymbol{H}^T\right\}^* \cap \operatorname{bd}\left\{\mathcal{C}^*\right\} = \{\lambda \boldsymbol{e}_\kappa \mid \lambda \geq 0, \kappa = 1, ..., k\},$$

therefore $\boldsymbol{a}_i$'s can only be the $\boldsymbol{e}_\kappa$'s. In other words, $\boldsymbol{A}$ can only be a permutation matrix. □

Proof of Lemma 3.2

Proof. First of all, since we assume that $\boldsymbol{H}^T\mathbf{1} = \mathbf{1}$, we have that $\boldsymbol{A}^T\boldsymbol{H}^T\mathbf{1} = \boldsymbol{A}^T\mathbf{1} = \mathbf{1}$.

Geometrically, $\boldsymbol{H}\boldsymbol{A} \geq 0$ means that $\boldsymbol{a}_i \in \operatorname{cone}\left\{\boldsymbol{H}^T\right\}^*$, where $\boldsymbol{a}_i$ is the i-column of $\boldsymbol{A}$, for all $i = 1, ..., k$. Because $\boldsymbol{H}$ is sufficiently scattered, we have that $\operatorname{cone}\left\{\boldsymbol{H}^T\right\} \subseteq \mathcal{C}^*$, which further implies that $\boldsymbol{a}_i \in \mathcal{C}^*$, which means

$$\|\boldsymbol{a}_i\| \leq \mathbf{1}^T\boldsymbol{a}_i.$$

Then we have the following chain

$$|\det \boldsymbol{A}| \leq \prod_{i=1}^{k} \|\boldsymbol{a}_i\| \tag{3.13a}$$

$$\leq \prod_{i=1}^{k} \mathbf{1}^T\boldsymbol{a}_i \tag{3.13b}$$

$$= 1, \tag{3.13c}$$

where (3.13a) is Hadamard's inequality, (3.13b) is because $\boldsymbol{a}_i \in \mathcal{C}^*$.

Now, suppose equality is attained, i.e., $|\det \boldsymbol{A}| = 1$, then all the inequalities in (3.12) hold as equality, and specifically (3.12b) means that the columns of $\boldsymbol{A}$ lie on the boundary of $\mathcal{C}^*$. Recall that $\boldsymbol{a}_i \in \operatorname{cone}\left\{\boldsymbol{H}^T\right\}^*$, and $\boldsymbol{H}$ being sufficiently scattered, according to the second requirement in Definition 2.5, shows that

$$\operatorname{cone}\left\{\boldsymbol{H}^T\right\}^* \cap \operatorname{bd}\left\{\mathcal{C}^*\right\} = \{\lambda\boldsymbol{e}_\kappa \mid \lambda \geq 0, \kappa = 1, ..., k\},$$

therefore $\boldsymbol{a}_i$'s can only be the $\boldsymbol{e}_\kappa$'s. In other words, $\boldsymbol{A}$ can only be a permutation matrix. □

As we can see, the proof of Lemma 3.2 is almost the same as that of Lemma 3.1, and actually simpler since we saved one step of applying arithmetic-geometric mean inequality to conclude that $|\det \boldsymbol{A}| \leq 1$.

Algorithm 3.2: Algorithm for NFCD

Input: $\boldsymbol{Y}$, $\boldsymbol{U}$, n, s_A, s_B, s_C

Output: $\tilde{\boldsymbol{A}}$, $\tilde{\boldsymbol{B}}$, $\tilde{\boldsymbol{C}}$

begin Step 1: simple subspace system identification

  $\boldsymbol{Y} \approx \boldsymbol{U}_n \boldsymbol{\Sigma}_n \boldsymbol{V}_n^T$;

  $\hat{\boldsymbol{C}} \leftarrow \boldsymbol{U}_n \boldsymbol{T}^{-1}, \quad \hat{\boldsymbol{X}} \leftarrow \boldsymbol{T} \boldsymbol{\Sigma}_n \boldsymbol{V}_n^T$;

  $\begin{bmatrix} \hat{\boldsymbol{A}} & \hat{\boldsymbol{B}} \end{bmatrix} = \hat{\boldsymbol{X}}_1 \begin{bmatrix} \hat{\boldsymbol{X}}_0 \\ \boldsymbol{U} \end{bmatrix}^{\dagger}$;

  scale the columns of $\hat{\boldsymbol{C}}$ to sum up to 1, and then counter-scale $\hat{\boldsymbol{A}}$ and $\hat{\boldsymbol{B}}$;

end

begin Step 2: solve Problem 3.10

  initialize $\boldsymbol{M}$ as a random matrix ;

  repeat

    for $j = 1, \ldots, n$ **do**

      Solve (3.10) with respect to $\boldsymbol{m}_j$;

    end

  until *convergence*;

end

begin Step 3: solve Problem 3.11

  initialize $\boldsymbol{M}$ as the result from the previous step, and $\boldsymbol{M}_{\text{inv}}$ as its inverse ;

  repeat

    $t \leftarrow$ the s_A-th largest value in $|\boldsymbol{M}_{\text{inv}} \hat{\boldsymbol{A}} \boldsymbol{M}|$;

    $\tilde{\boldsymbol{A}} \leftarrow \mathcal{T}_t(\boldsymbol{M}_{\text{inv}} \hat{\boldsymbol{A}} \boldsymbol{M})$;

    $t \leftarrow$ the s_B-th largest value in $|\boldsymbol{M}_{\text{inv}} \hat{\boldsymbol{B}}|$;

    $\tilde{\boldsymbol{B}} \leftarrow \mathcal{T}_t(\boldsymbol{M}_{\text{inv}} \hat{\boldsymbol{B}})$;

    $t \leftarrow \max(0,$the s_C-th largest value in $\hat{\boldsymbol{C}} \boldsymbol{M})$;

    $\tilde{\boldsymbol{C}} \leftarrow \mathcal{T}_t^{+}(\hat{\boldsymbol{C}} \boldsymbol{M})$ $\boldsymbol{M}_{\text{inv}} \leftarrow \begin{bmatrix} \tilde{\boldsymbol{A}} & \tilde{\boldsymbol{B}} & \lambda \mathbf{I} \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{A}} \boldsymbol{M} & \hat{\boldsymbol{B}} & \lambda \boldsymbol{M} \end{bmatrix}^{\dagger}$;

    $\boldsymbol{M} \leftarrow \begin{bmatrix} \boldsymbol{M}_{\text{inv}} \hat{\boldsymbol{A}} \\ \hat{\boldsymbol{C}} \\ \lambda \boldsymbol{M}_{\text{inv}} \end{bmatrix}^{\dagger} \begin{bmatrix} \tilde{\boldsymbol{A}} \\ \tilde{\boldsymbol{C}} \\ \lambda \mathbf{I} \end{bmatrix}$;

  until *convergence*;

end

Table 3.1: System matrices recovery in the noiseless case.

	$n = 30$ $s = 0.5$	$n = 30$ $s = 0.3$	$n = 15$ $s = 0.5$
$\frac{\|\boldsymbol{A}-\hat{\boldsymbol{A}}\|_F}{\|\boldsymbol{A}\|_F}$	7.33e-07	5.85e-06	1.03e-05
$\frac{\|\boldsymbol{B}-\hat{\boldsymbol{B}}\|_F}{\|\boldsymbol{B}\|_F}$	7.11e-07	5.49e-06	1.50e-05
$\frac{\|\boldsymbol{C}-\hat{\boldsymbol{C}}\|_F}{\|\boldsymbol{C}\|_F}$	5.93e-07	5.14e-06	1.30e-05

Table 3.2: System matrices recovery of the C. elegans system with noisy data.

	$n = 10$	$n = 15$	$n = 20$
$\frac{\|\boldsymbol{A}-\hat{\boldsymbol{A}}\|_F}{\|\boldsymbol{A}\|_F}$	3.71e-04	0.0650	0.0299
$\frac{\|\boldsymbol{B}-\hat{\boldsymbol{B}}\|_F}{\|\boldsymbol{B}\|_F}$	4.77e-04	0.0455	0.0153
$\frac{\|\boldsymbol{C}-\hat{\boldsymbol{C}}\|_F}{\|\boldsymbol{C}\|_F}$	3.73e-04	0.0375	0.0135

Chapter 4

A Flexible and Efficient Algorithmic Framework

Constrained matrix and tensor factorization techniques are widely used for latent parameter estimation and blind source separation in signal processing, dimensionality reduction and clustering in machine learning, and numerous other applications in diverse disciplines, such as chemistry and psychology. Least-squares low-rank factorization of matrices and tensors without additional constraints is relatively well-studied, as in the matrix case the basis of any solution is simply the principal components of the singular value decomposition (SVD) [61], also known as principal component analysis (PCA), and in the tensor case alternating least squares (ALS) and other algorithms usually yield satisfactory results [62]. ALS is also used for matrix factorization, especially when the size is so large that performing the exact PCA is too expensive.

Whereas unconstrained matrix and tensor factorization algorithms are relatively mature, their *constrained* counterparts leave much to be desired as of this writing, and a unified framework that can easily and naturally incorporate multiple constraints on the latent factors is sorely missing. Existing algorithms are usually only able to handle one or at most few specialized constraints, and/or the algorithm needs to be redesigned carefully if new constraints are added. Commonly adopted constraints imposed on the latent factors include non-negativity [1], sparsity (usually via sparsity-inducing ℓ_1 regularization [63]), and simplex constraints [64], to name just a few.

On top of the need to incorporate constraints on the latent factors, many established and emerging signal processing applications entail cost (*loss*) functions that differ from classical least-squares. Important examples include matrix completion [65] where missing values are ignored by the loss function, and robust PCA [66] where the ℓ_1 loss is used. In the matrix case without constraints on the latent factors, these can be formulated as convex problems via nuclear norm regularizations and solved in polynomial-time [67]. With explicit constraints imposed on the latent factors, and/or for tensor data, however, non-convex (multi-linear) formulations are unavoidable, and a unified algorithmic framework that can handle a variety of constraints and loss functions would be very welcome.

In this chapter, we describe a general algorithmic framework that seamlessly and relatively effortlessly incorporates many common types of constraints and loss functions, building upon and bridging together the alternating optimization (AO) framework and the alternating direction method of multipliers (ADMM), hence the name AO-ADMM.

While combining these frameworks may seem conceptually straightforward at first sight, what is significant is that AO-ADMM outperforms all prior algorithms for constrained matrix and tensor factorization under nonparametric constraints on the latent factors. One example is non-negative matrix factorization, where the prior art includes decades of research. This is the biggest but not the only advantage of AO-ADMM. Carefully developing various aspects of this combination, we show that

- AO-ADMM converges to a stationary point of the original NP-hard problem;
- Using computation caching, warm-start, and good parameter settings, its per-iteration complexity is similar to that of ALS;
- AO-ADMM can incorporate a wide-range of constraints and regularization penalties on the latent factors at essentially the same complexity;
- It can also accommodate a wide variety of cost / loss functions, with only moderate increase in complexity relative to the classical least-squares loss; and
- The core computations are exactly the same as ALS for unconstrained factorization, with some additional element-wise operations to handle constraints, making

it easy to incorporate smart implementations of ALS, including sparse, parallel, and high-performance computing enhancements.

4.1 Alternating Optimization Framework: Preliminaries

We start by formulating the factorization problem as an optimization problem in the most general form

$$\underset{\boldsymbol{H}_1,\ldots,\boldsymbol{H}_N}{\text{minimize}} \quad l\left(\underline{\boldsymbol{Y}} - [\boldsymbol{H}_d]_{d=1}^N\right) + \sum_{d=1}^N r_d(\boldsymbol{H}_d), \tag{4.1}$$

with a slight abuse of notation by assuming N can also take the value of 2. In (4.1), $l(\cdot)$ can be any loss measure, most likely separable down to the entries of the argument, and $r_d(\boldsymbol{H}_d)$ is the generalized regularization on $\boldsymbol{H}_d$, which may take the value of $+\infty$ so that any hard constraints can also be incorporated. For example, if we require that the elements of $\boldsymbol{H}_d$ are nonnegative, denoted as $\boldsymbol{H}_d \geq 0$, then

$$r_d(\boldsymbol{H}_d) = \begin{cases} 0, & \boldsymbol{H}_d \geq 0, \\ +\infty, & \text{otherwise.} \end{cases}$$

Because of the multi-linear term $[\boldsymbol{H}_d]_{d=1}^N$, the regularized fitting problem is non-convex, and in many cases NP-hard [25,68]. A common way to handle this is to use the alternating optimization (AO) technique, i.e., update each factor $\boldsymbol{H}_d$ in a cyclic fashion. The popular ALS algorithm is a special case of this when $l(\cdot)$ is the least-squares loss, and there is no regularization. In this section, we will first revisit the ALS algorithm, with the focus on the per-iteration complexity analysis. Then, we will briefly discuss the convergence of the AO framework, especially some recent advances on the convergence of the traditional block coordinate descent (BCD) algorithm.

4.1.1 Alternating Least-Squares Revisited

Consider the unconstrained matrix factorization problem

$$\underset{\boldsymbol{W},\boldsymbol{H}}{\text{minimize}} \quad \frac{1}{2}\|\boldsymbol{Y} - \boldsymbol{W}\boldsymbol{H}^T\|_F^2, \tag{4.2}$$

and momentarily ignore the fact that the optimal solution of (4.2) is given by the SVD. The problem (4.2) is non-convex in $\boldsymbol{W}$ and $\boldsymbol{H}$ jointly, but is convex if we fix one and

treat only the other as variable. Supposing $\boldsymbol{W}$ is fixed, the sub-problem for $\boldsymbol{H}$ becomes the classical linear least squares and, if $\boldsymbol{W}$ has full column rank, the unique solution is given by

$$\boldsymbol{H}^T = (\boldsymbol{W}^T\boldsymbol{W})^{-1}\boldsymbol{W}^T\boldsymbol{Y}. \tag{4.3}$$

In practice, the matrix inverse $(\boldsymbol{W}^T\boldsymbol{W})^{-1}$ is almost never explicitly calculated. Instead, the Cholesky decomposition of the Gram matrix $\boldsymbol{W}^T\boldsymbol{W}$ is computed, and for each column of $\boldsymbol{W}^T\boldsymbol{Y}$, a forward and a backward substitution are performed to get the corresponding column of $\boldsymbol{H}^T$. Since $\boldsymbol{W}$ is $m \times k$ and $\boldsymbol{Y}$ is $m \times n$, forming $\boldsymbol{W}^T\boldsymbol{W}$ and $\boldsymbol{W}^T\boldsymbol{Y}$ takes $\mathcal{O}(mk^2)$ and $\mathcal{O}(mnk)$ flops, respectively, computing the Cholesky decomposition requires $\mathcal{O}(k^3)$ flops, and finally the back substitution step takes $\mathcal{O}(nk^2)$ flops, similar to a matrix multiplication. If $m, n > k$, then the overall complexity is $\mathcal{O}(mnk)$.

An important implication is the following. Clearly, if $n = 1$, the cost of solving a least-squares problem is $\mathcal{O}(mk^2)$. However, as n grows, the complexity does not simply grow proportionally with n, but also amortizes a factor of k and goes to $\mathcal{O}(mnk)$. The reason is that, although it seems we are now trying to solve n least-squares problems, they all share the same matrix $\boldsymbol{W}$, thus the Cholesky decomposition of $\boldsymbol{W}^T\boldsymbol{W}$ can be reused throughout. This is a very nice property of the unconstrained least squares problems, which can be exploited to improve the computational efficiency of the ALS algorithm.

One may recall that another well-adopted method for least-squares is to compute the QR decomposition of $\boldsymbol{W}$ as $\boldsymbol{W} = \boldsymbol{Q}\boldsymbol{R}$, so that $\boldsymbol{H}^T = \boldsymbol{R}^{-1}\boldsymbol{Q}^T\boldsymbol{Y}$. This can be shown to give the same computational complexity as the Cholesky version, and is actually more stable numerically. However, if $\boldsymbol{W}$ has some special structure, it is easier to exploit this structure if we use Cholesky decomposition. Therefore, in this paper we only consider solving least-squares problems using the Cholesky decomposition.

One important structure that we encounter is in the tensor case. For the ALS algorithm for CP factorization, the update of $\boldsymbol{H}_d$ is the solution of the following least squares problem

$$\underset{\boldsymbol{H}_d}{\text{minimize}} \quad \frac{1}{2}\left\| \boldsymbol{Y}_{(d)} - \left(\bigodot_{\substack{j=1 \\ j\neq d}}^{N} \boldsymbol{H}_j \right) \boldsymbol{H}_d^T \right\|_F^2,$$

and the solution is given by

$$\boldsymbol{H}_d^T = \left(\overset{N}{\underset{\substack{j=1 \\ j\neq d}}{\circledast}} \boldsymbol{H}_j^T \boldsymbol{H}_j \right)^{-1} \left(\overset{N}{\underset{\substack{j=1 \\ j\neq d}}{\odot}} \boldsymbol{H}_j \right)^T \boldsymbol{Y}_{(d)}.$$

As we can see, the Gram matrix is computed efficiently by exploiting the structure, and its Cholesky decomposition can be reused. The most expensive operation is actually the computation of $(\odot_{j\neq d} \boldsymbol{H}_j)^T \boldsymbol{Y}_{(d)}$, but very efficient algorithms for this (that work without explicitly forming the Khatri-Rao product and the d-mode matricization) are available [69–75]. If we were to adopt the QR decomposition approach, however, none of these methods could be applied.

In summary, least squares is a very mature technique with many favorable properties that render the ALS algorithm very efficient. On the other hand, most of the algorithms that deal with problems with constraints on the factors or different loss measures do not inherit these good properties. The goal of this paper is to propose an AO-based algorithmic framework, which can easily handle many types of constraints on the latent factors and many loss functions, with per-iteration complexity essentially the same as the complexity of an ALS step.

4.1.2 The Convergence of AO

Consider the following (usually non-convex) optimization problem with variables separated into N blocks, each with its own constraint set

$$\begin{aligned} \underset{\boldsymbol{x}_1,...,\boldsymbol{x}_N}{\text{minimize}} \quad & f(\boldsymbol{x}_1, ..., \boldsymbol{x}_N) \\ \text{subject to} \quad & \boldsymbol{x}_d \in \mathcal{X}_d, \quad \forall d = 1, ..., N. \end{aligned} \tag{4.4}$$

A classical AO method called block coordinate descent (BCD) cyclically updates $\boldsymbol{x}_d$ via solving

$$\begin{aligned} \underset{\xi}{\text{minimize}} \quad & f(\boldsymbol{x}_1^{r+1}, ..., \boldsymbol{x}_{d-1}^{r+1}, \xi, \boldsymbol{x}_{d+1}^{r}, ..., \boldsymbol{x}_N^{r}) \\ \text{subject to} \quad & \xi \in \mathcal{X}_d, \end{aligned} \tag{4.5}$$

at the $(r+1)$-th iteration [76, § 2.7]. Obviously, this will decrease the objective function monotonically. If some additional assumptions are satisfied, then we can have stronger convergence claims [76, Proposition 2.7.1]. Simply put, if the sub-problem (4.5) is convex

and has a *unique* solution, then every limit point is a stationary point; furthermore, if $\mathcal{X}_1, ..., \mathcal{X}_N$ are all compact, which implies that the sequence generated by BCD is bounded, then BCD is guaranteed to converge to a stationary point, even if (4.4) is non-convex [77].

In many cases (4.5) is convex, but the uniqueness of the solution is very hard to guarantee. A special case that does not require uniqueness, first noticed by Grippo and Sciandrone [78], is when $N = 2$. On hindsight, this can be explained by the fact that for $N = 2$, BCD coincides with the so-called maximum block improvement (MBI) algorithm [79], which converges under very mild conditions. However, instead of updating the blocks cyclically, MBI only updates the one block that decreases the objective the most, thus the per-iteration complexity is $(N-1)$ times higher than BCD; therefore MBI is not commonly used in practice when N is large.

Another way to ensure convergence, proposed by Razaviyayn *et al.* [80], is as follows. Instead of updating $\boldsymbol{x}_d$ as the solution of (4.5), the update is obtained by solving a majorized version of (4.5), called the block successive upper-bound minimization (BSUM). The convergence of BSUM is essentially the same, but now we can deliberately design the majorizing function to ensure that the solution is unique. One simple way to do this is to put a proximal regularization term

$$\begin{aligned} \underset{\boldsymbol{\xi}}{\text{minimize}} \quad & f(\boldsymbol{x}_1^{r+1}, ..., \boldsymbol{x}_{d-1}^{r+1}, \boldsymbol{\xi}, \boldsymbol{x}_{d+1}^{r}, ..., \boldsymbol{x}_N^{r}) + \frac{\mu_d^{r+1}}{2} \|\boldsymbol{\xi} - \boldsymbol{x}_d^r\|^2 \\ \text{subject to} \quad & \boldsymbol{\xi} \in \mathcal{X}_d, \end{aligned} \tag{4.6}$$

for some $\mu_d^{r+1} > 0$ at every iteration for each block, where $\boldsymbol{x}_d^r$ is the update of $\boldsymbol{x}_d$ from the previous iteration. If (4.5) is convex, then (4.6) is strongly convex, which gives a unique minimizer. Thus, the algorithm is guaranteed to converge to a stationary point, as long as the sequence generated by the algorithm is bounded. In the contex of ALS, this type of update strategy is independently shown in [81] to converge to a stationary point. Similar results are also proved in [82], where the authors used a different majorization for constrained matrix/tensor factorization; we shall compare with them in numerical experiments.

4.2 Solving the Sub-problems Using ADMM

The AO algorithm framework is usually adopted when each of the sub-problems can be solved efficiently. This is indeed the case for the ALS algorithm, since each update is in closed-form. For the general factorization problem (4.1), we denote the sub-problem as

$$\underset{\boldsymbol{H}}{\text{minimize}} \quad l(\boldsymbol{Y} - \boldsymbol{W}\boldsymbol{H}^T) + r(\boldsymbol{H}). \tag{4.7}$$

For the matrix case, this is simply the sub-problem for the right factor, and one can easily figure out the update of the left factor by transposing everything; for the tensor case, this becomes the update of $\boldsymbol{H}_d$ by setting $\boldsymbol{Y} = \boldsymbol{Y}_{(d)}$ and $\boldsymbol{W} = \odot_{j \neq d} \boldsymbol{H}_j$. This is for ease of notation, as these matricizations and Khatri-Rao products need not be actually computed explicitly. Also notice that this is the sub-problem for the BCD algorithm, and for better convergence we may want to add a proximal regularization term to (4.7), which is very easy to handle, thus omitted here.

We propose to use the alternating direction method of multipliers (ADMM) to solve (4.7). ADMM, if used in the right way, inherits a lot of the good properties that appeared in each update of the ALS method. Furthermore, the AO framework naturally provides good initializations for ADMM, which further accelerates its convergence for the subproblem. As a preview, the implicit message here is that *closed-form solution is not necessary for computational efficiency*, as we will explain later. After a brief introduction of ADMM, we first apply it to (4.7) which has least-squares loss, and then generalize it to universal loss measures.

4.2.1 Alternating Direction Method of Multipliers

ADMM solves convex optimization problems that can be written in the form

$$\begin{aligned} \underset{\boldsymbol{x},\boldsymbol{z}}{\text{minimize}} \quad & f(\boldsymbol{x}) + g(\boldsymbol{z}) \\ \text{subject to} \quad & \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{z} = \boldsymbol{c}, \end{aligned}$$

by iterating the following updates

$$\begin{aligned}
\boldsymbol{x} &\leftarrow \arg\min_{\boldsymbol{x}} f(\boldsymbol{x}) + (\rho/2)\|\boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{z} - \boldsymbol{c} + \boldsymbol{u}\|_2^2, \\
\boldsymbol{z} &\leftarrow \arg\min_{\boldsymbol{z}} g(\boldsymbol{z}) + (\rho/2)\|\boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{z} - \boldsymbol{c} + \boldsymbol{u}\|_2^2, \\
\boldsymbol{u} &\leftarrow \boldsymbol{u} + (\boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{z} - \boldsymbol{c}),
\end{aligned}$$

where $\boldsymbol{u}$ is a scaled version of the dual variables corresponding to the equality constraint $\boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{z} = \boldsymbol{c}$, and ρ is specified by the user.

A comprehensive review of the ADMM algorithm can be found in [83] and the references therein. The beauty of ADMM is that it converges under mild conditions (in the convex case), while artful splitting of the variables into the two blocks $\boldsymbol{x}$ and $\boldsymbol{z}$ can yield very efficient updates, and/or distributed implementation. Furthermore, if f is strongly convex and Lipschitz continuous, then linear convergence of ADMM can be achieved; cf. guidelines on the optimal step-size ρ in [84, §9.3], and [85] for an analysis of ADMM applied to quadratic programming. It is also shown empirically that ADMM works very well for non-convex problems as well, e.g. [86, 87], although its convergence is much harder to analyze.

4.2.2 Least-Squares Loss

We start by considering $l(\cdot)$ in (4.7) to be the least-squares loss $(1/2)\|\cdot\|_F^2$. The problem is reformulated by introducing a $k \times n$ auxiliary variable $\tilde{\boldsymbol{H}}$

$$\begin{aligned}
\underset{\boldsymbol{H},\tilde{\boldsymbol{H}}}{\text{minimize}} \quad & \frac{1}{2}\|\boldsymbol{Y} - \boldsymbol{W}\tilde{\boldsymbol{H}}\|_F^2 + r(\boldsymbol{H}) \\
\text{subject to} \quad & \boldsymbol{H} = \tilde{\boldsymbol{H}}^T.
\end{aligned} \tag{4.8}$$

It is easy to adopt the ADMM algorithm and derive the following iterates:

$$\begin{aligned}
\tilde{\boldsymbol{H}} &\leftarrow (\boldsymbol{W}^T\boldsymbol{W} + \rho\boldsymbol{I})^{-1}(\boldsymbol{W}^T\boldsymbol{Y} + \rho(\boldsymbol{H} + \boldsymbol{U})^T), \\
\boldsymbol{H} &\leftarrow \arg\min_{\boldsymbol{H}} r(\boldsymbol{H}) + \frac{\rho}{2}\|\boldsymbol{H} - \tilde{\boldsymbol{H}}^T + \boldsymbol{U}\|_F^2, \\
\boldsymbol{U} &\leftarrow \boldsymbol{U} + \boldsymbol{H} - \tilde{\boldsymbol{H}}^T.
\end{aligned} \tag{4.9}$$

One important observation is that, throughout the iterations the same matrix $\boldsymbol{W}^T\boldsymbol{Y}$ and matrix inverse $(\boldsymbol{W}^T\boldsymbol{W} + \rho\boldsymbol{I})^{-1}$ are used. Therefore, to save computations, we can

cache $\boldsymbol{W}^T\boldsymbol{Y}$ and the Cholesky decomposition of $\boldsymbol{W}^T\boldsymbol{W} + \rho\boldsymbol{I} = \boldsymbol{L}\boldsymbol{L}^T$. Then the update of $\tilde{\boldsymbol{H}}$ is dominated by one forward substitution and one backward substitution, resulting in a complexity of $\mathcal{O}(k^2 n)$.

The update of $\boldsymbol{H}$ is the so-called *proximity operator* of the function $(1/\rho)r(\cdot)$ around point $(\tilde{\boldsymbol{H}}^T - \boldsymbol{U})$, and in particular if $r(\cdot)$ is the indicator function of a convex set, then the update of $\boldsymbol{H}$ becomes a projection operator, a special case of the proximity operator. For a lot of regularizations/constraints, especially those that are often used in matrix/tensor factorization problems, the update of $\boldsymbol{H}$ boils down to element-wise updates, i.e., costing $\mathcal{O}(kn)$ flops. Here we list some of the most commonly used constraints/regularizations in the matrix factorization problem, and refer the reader to [88, §6]. For simplicity of notation, let us define $\bar{\boldsymbol{H}} = \tilde{\boldsymbol{H}}^T - \boldsymbol{U}$.

- Non-negativity. In this case $r(\cdot)$ is the indicator function of $\mathbb{R}_+$, and the update is simply zeroing out the negative values of $\bar{\boldsymbol{H}}$. In fact, any element-wise bound constraints can be handled similarly, since element-wise projection is trivial.

- Lasso regularization. For $r(\boldsymbol{H}) = \lambda\|\boldsymbol{H}\|_1$, the sparsity inducing regularization, the update is the well-known *soft-thresholding* operator: $h_{ij} = [1 - (\lambda/\rho)|\bar{h}_{ij}|^{-1}]_+\bar{h}_{ij}$. The element-wise thresholding can also be converted to block-wise thresholding if one wants to impose structured sparsity, leading to the group Lasso regularization.

- Simplex constraint. In some probabilistic model analysis we need to constrain the columns or rows to be element-wise non-negative and sum up to one. As described in [89], this projection can be done with a randomized algorithm with linear-time complexity on average.

- Smoothness regularization. We can encourage the columns of $\boldsymbol{H}$ to be smooth by adding the regularization $r(\boldsymbol{H}) = (\lambda/2)\|\boldsymbol{T}\boldsymbol{H}\|_F^2$ where $\boldsymbol{T}$ is obtained from an $n \times n$ tri-diagonal matrix with 2 on the diagonal and -1 on the super- and sub-diagonal by removing its first and last row. Its proximity operator is given by $\boldsymbol{H} = \rho(\lambda\boldsymbol{T}^T\boldsymbol{T} + \rho\boldsymbol{I})^{-1}\bar{\boldsymbol{H}}$. Although it involves a large matrix inversion, notice that it has a fixed bandwidth of 2, thus can be efficiently calculated in $\mathcal{O}(kn)$ time [90, §4.3].

- It is also possible to define projections onto non-convex constraints, for example

cardinality constraints can be handled by hard thresholding (as opposed to soft thresholding for lasso regularization). However, ADMM is not guaranteed to converge to the conditionally optimal solution in this case, therefore it is not discussed in this paper. If it cannot be avoided, one can still use AO-ADMM attempting to obtain good results, and the performance is not bad compared to the alternatives.

We found empirically that by setting $\rho = \|\boldsymbol{W}\|_F^2/k$, the ADMM iterates for the regularized least-squares problem (4.8) converge very fast. This choice of ρ can be seen as an approximation to the optimal ρ given in [84], but much cheaper to obtain. With a good initialization, naturally provided by the AO framework, the update of $\boldsymbol{H}$ usually does not take more than 5 or 10 ADMM iterations, and very soon reduces down to only 1 iteration. The proposed algorithm for the sub-problem (4.8) is summarized in Alg. 4.1. As we can see, the pre-calculation step takes $\mathcal{O}(k^2m + k^3)$ flops to form the Cholesky decomposition, and $\mathcal{O}(mnk)$ flops to form $\boldsymbol{F}$. Notice that these are actually the only computations in Alg. 4.1 that involve $\boldsymbol{W}$ and $\boldsymbol{Y}$, which implies that in the tensor case, all the tricks to compute $\boldsymbol{W}^T\boldsymbol{W}$ and $\boldsymbol{W}^T\boldsymbol{Y}$ can be applied here, and then we do not need to worry about them anymore. The computational load of each ADMM iteration is dominated by the $\tilde{\boldsymbol{H}}$-update, with complexity $\mathcal{O}(k^2n)$.

It is interesting to compare Alg. 4.1 with an update of the ALS algorithm, whose complexity is essentially the same as the pre-calculation step plus one iteration. For a small number of ADMM iterations, the complexity of Alg. 4.1 is of the same order as an ALS step.

For declaring termination, we adopted the general termination criterion described in [83, §3.3.1]. After some calibration, we define the relative primal residual

$$r = \|\boldsymbol{H} - \tilde{\boldsymbol{H}}^T\|_F^2/\|\boldsymbol{H}\|_F^2, \tag{4.10}$$

and the relative dual residual

$$s = \|\boldsymbol{H} - \boldsymbol{H}_0\|_F^2/\|\boldsymbol{U}\|_F^2, \tag{4.11}$$

where $\boldsymbol{H}_0$ is $\boldsymbol{H}$ from the previous ADMM iteration, and terminate Alg. 4.1 if both of them are smaller than some threshold.

Furthermore, if the BSUM framework is adopted, we need to solve a proximal regularized version of (4.8), and that term can easily be absorbed into the update of $\tilde{\boldsymbol{H}}$.

Algorithm 4.1: Solve (4.8) using ADMM

Input: $\boldsymbol{Y}$, $\boldsymbol{W}$, $\boldsymbol{H}$, $\boldsymbol{U}$, k

Initialize $\boldsymbol{H}$ and $\boldsymbol{U}$;

$\boldsymbol{G} = \boldsymbol{W}^T\boldsymbol{W}$;

$\rho = \text{trace}\{\boldsymbol{G}\}/k$;

Calculate $\boldsymbol{L}$ from the Cholesky decomposition of $\boldsymbol{G} + \rho\boldsymbol{I} = \boldsymbol{L}\boldsymbol{L}^T$;

$\boldsymbol{F} = \boldsymbol{W}^T\boldsymbol{Y}$;

repeat

  $\tilde{\boldsymbol{H}} \leftarrow (\boldsymbol{L}^T)^{-1}\boldsymbol{L}^{-1}(\boldsymbol{F} + \rho(\boldsymbol{H} + \boldsymbol{U})^T)$ using forward/backward substitution ;

  $\boldsymbol{H} \leftarrow \arg\min_{\boldsymbol{H}} r(\boldsymbol{H}) + \frac{\rho}{2}\|\boldsymbol{H} - \tilde{\boldsymbol{H}}^T + \boldsymbol{U}\|_F^2$;

  $\boldsymbol{U} \leftarrow \boldsymbol{U} + \boldsymbol{H} - \tilde{\boldsymbol{H}}^T$;

until $r < \varepsilon$ and $s < \varepsilon$; r and s defined in (4.10) and (4.11)

return $\boldsymbol{H}$ and $\boldsymbol{U}$.

4.2.3 General Loss

Now let us derive an ADMM algorithm to solve the more general problem (4.7). For this case, we reformulate the problem by introducing two auxiliary variables $\tilde{\boldsymbol{H}}$ and $\tilde{\boldsymbol{Y}}$

$$\begin{aligned} &\underset{\boldsymbol{H},\tilde{\boldsymbol{H}},\tilde{\boldsymbol{Y}}}{\text{minimize}} \quad l(\boldsymbol{Y} - \tilde{\boldsymbol{Y}}) + r(\boldsymbol{H}) \\ &\text{subject to} \quad \boldsymbol{H} = \tilde{\boldsymbol{H}}^T, \quad \tilde{\boldsymbol{Y}} = \boldsymbol{W}\tilde{\boldsymbol{H}}. \end{aligned} \tag{4.12}$$

To apply ADMM, let $\tilde{\boldsymbol{H}}$ be the first block, and $(\tilde{\boldsymbol{Y}}, \boldsymbol{H})$ be the second block, and notice that in the second block update $\tilde{\boldsymbol{Y}}$ and $\boldsymbol{H}$ can in fact be updated independently. This yields the following iterates:

$$\begin{aligned} &\tilde{\boldsymbol{H}} \leftarrow (\boldsymbol{W}^T\boldsymbol{W} + \rho\boldsymbol{I})^{-1}(\boldsymbol{W}^T(\tilde{\boldsymbol{Y}} + \boldsymbol{V}) + \rho(\boldsymbol{H} + \boldsymbol{U})^T) \\ &\begin{cases} \boldsymbol{H} \leftarrow \arg\min_{\boldsymbol{H}} \; r(\boldsymbol{H}) + \dfrac{\rho}{2}\|\boldsymbol{H} - \tilde{\boldsymbol{H}}^T + \boldsymbol{U}\|_F^2, \\ \tilde{\boldsymbol{Y}} \leftarrow \arg\min_{\tilde{\boldsymbol{Y}}} \; l(\boldsymbol{Y} - \tilde{\boldsymbol{Y}}) + \dfrac{1}{2}\|\tilde{\boldsymbol{Y}} - \boldsymbol{W}\tilde{\boldsymbol{H}} + \boldsymbol{V}\|_F^2, \end{cases} \\ &\begin{cases} \boldsymbol{U} \leftarrow \boldsymbol{U} + \boldsymbol{H} - \tilde{\boldsymbol{H}}^T, \\ \boldsymbol{V} \leftarrow \boldsymbol{V} + \tilde{\boldsymbol{Y}} - \boldsymbol{W}\tilde{\boldsymbol{H}}. \end{cases} \end{aligned} \tag{4.13}$$

where $\boldsymbol{U}$ is the scaled dual variable corresponding to the constraint $\boldsymbol{H} = \tilde{\boldsymbol{H}}^T$, and $\boldsymbol{V}$ is the scaled dual variable corresponding to the equality constraint $\tilde{\boldsymbol{Y}} = \boldsymbol{W}\tilde{\boldsymbol{H}}$. Notice that we set the penalty parameter ρ corresponding to the second constraint to be 1, since it works very well in practice, and also leads to very intuitive results for some loss functions. This can also be interpreted as first pre-conditioning this constraint to be $\frac{1}{\sqrt{\rho}}\tilde{\boldsymbol{Y}} = \frac{1}{\sqrt{\rho}}\boldsymbol{W}\tilde{\boldsymbol{H}}$, and then a common ρ is used. Again we set $\rho = \|\boldsymbol{W}\|_F^2/k$.

As we can see, the update of $\tilde{\boldsymbol{H}}$ is simply a linear least squares problem, and all the previous discussion about caching the Cholesky decomposition applies. It is also easy to absorb an additional proximal regularization term into the update of $\tilde{\boldsymbol{H}}$, if the BSUM framework is adopted. The update of $\tilde{\boldsymbol{Y}}$ is (similar to the update of $\boldsymbol{H}$) a proximity operator, and since almost all loss functions we use are element-wise, the update of $\tilde{\boldsymbol{Y}}$ is also very easy. The updates for some of the most commonly used non-least-squares loss functions are listed below. For simplicity, we define $\bar{\boldsymbol{Y}} = \boldsymbol{W}\tilde{\boldsymbol{H}} - \boldsymbol{V}$, similar to the previous sub-section.

- Missing values. In the case that only a subset of the entries in $\boldsymbol{Y}$ are available, a common way to handle this is to simply fit the low-rank model only to the available entries. Let $\mathcal{A}$ denote the index set of the available values in $\boldsymbol{Y}$, then the loss function becomes $l(\boldsymbol{Y} - \tilde{\boldsymbol{Y}}) = \frac{1}{2}\sum_{(i,j)\in\mathcal{A}}(y_{ij} - \tilde{y}_{ij})^2$. Thus, the update of $\tilde{\boldsymbol{Y}}$ in (4.13) becomes

$$\tilde{y}_{ij} = \begin{cases} \frac{1}{2}(y_{ij} + \bar{y}_{ij}), & (i,j) \in \mathcal{A}, \\ \bar{y}_{ij}, & \text{otherwise.} \end{cases}$$

- Robust fitting. In the case that data entries are not uniformly corrupted by noise but only sparingly corrupted by outliers, or when the noise is dense but heavy-tailed (e.g., Laplacian-distributed), we can use the ℓ_1 norm as the loss function for robust (resp. maximum-likelihood) fitting, i.e., $l(\boldsymbol{Y} - \tilde{\boldsymbol{Y}}) = \|\boldsymbol{Y} - \tilde{\boldsymbol{Y}}\|_1$. This is similar to the ℓ_1 regularization, and the element-wise update is

$$\tilde{y}_{ij} = \begin{cases} y_{ij}, & |\bar{y}_{ij} - y_{ij}| \le 1, \\ \bar{y}_{ij} - 1, & \bar{y}_{ij} - y_{ij} > 1, \\ \bar{y}_{ij} + 1, & \bar{y}_{ij} - y_{ij} < -1. \end{cases}$$

- Huber fitting. Another way to deal with possible outliers in $\boldsymbol{Y}$ is to use the Huber function to measure the loss $l(\boldsymbol{Y} - \tilde{\boldsymbol{Y}}) = \sum_{i,j} \phi_\lambda(y_{ij} - \tilde{y}_{ij})$ where

$$\phi_\lambda(z) = \begin{cases} \frac{1}{2}z^2, & |z| \leq \lambda, \\ \lambda|z| - \frac{1}{2}\lambda^2, & \text{otherwise.} \end{cases}$$

  The element-wise closed-form update is

$$\tilde{y}_{ij} = \begin{cases} \frac{1}{2}(\bar{y}_{ij} + y_{ij}), & |\bar{y}_{ij} - y_{ij}| \leq 2\lambda, \\ \bar{y}_{ij} - \lambda, & \bar{y}_{ij} - y_{ij} > 2\lambda, \\ \bar{y}_{ij} + \lambda, & \bar{y}_{ij} - y_{ij} < -2\lambda. \end{cases}$$

- Kullback-Leibler divergence. A commonly adopted loss function for non-negative integer data is the Kullback-Leibler (K-L) divergence defined as

$$D(\boldsymbol{Y}||\tilde{\boldsymbol{Y}}) = \sum_{i,j} \left(y_{ij} \log \frac{y_{ij}}{\tilde{y}_{ij}} - y_{ij} + \tilde{y}_{ij} \right)$$

  for which the proximity operator is

$$\tilde{\boldsymbol{Y}} = \frac{1}{2}\left((\bar{\boldsymbol{Y}} - 1) + \sqrt{(\bar{\boldsymbol{Y}} - 1)^2 + 4\boldsymbol{Y}} \right),$$

  where all the operations are taken element-wise [91]. Furthermore, the K-L divergence is a special case of certain families of divergence functions, such as α-divergence and β-divergence [92], whose corresponding updates are also very easy to derive (boil down to the proximity operator of a scalar function).

An interesting observation is that if the loss function is in fact the least-squares loss, the matrix $(\tilde{\boldsymbol{Y}} + \boldsymbol{V})$ that $\tilde{\boldsymbol{H}}$ is trying to fit in (4.13) is the data matrix $\boldsymbol{Y}$ *per se*. Therefore, the update rule (4.13) boils down to the update rule (4.9) in the least-squares loss case, with some redundant updates of $\tilde{\boldsymbol{Y}}$ and $\boldsymbol{V}$. The detailed ADMM algorithm for (4.12) is summarized in Alg. 4.2. We use the same termination criterion as in Alg. 4.1.

Everything seems to be in place to seamlessly move from the least-squares loss to arbitrary loss. Nevertheless, closer scrutiny reveals that some compromises must be made to take this leap. One relatively minor downside is that with a general loss function we may lose the linear convergence rate of ADMM – albeit with the good initialization naturally provided by the AO framework and our particular choice of ρ,

Algorithm 4.2: Solve (4.12) using ADMM

Input: $\boldsymbol{Y}$, $\boldsymbol{W}$, $\boldsymbol{H}$, $\boldsymbol{U}$, $\tilde{\boldsymbol{Y}}$, $\boldsymbol{V}$, k

Initialize $\boldsymbol{H}$, $\boldsymbol{U}$, $\tilde{\boldsymbol{Y}}$, and $\boldsymbol{V}$;

$\boldsymbol{G} = \boldsymbol{W}^T\boldsymbol{W}$;

$\rho = \text{trace}\left\{\boldsymbol{G}\right\}/k$;

Calculate $\boldsymbol{L}$ from the Cholesky decomposition of $\boldsymbol{G} + \rho\boldsymbol{I} = \boldsymbol{L}\boldsymbol{L}^T$;

repeat

  $\tilde{\boldsymbol{H}} \leftarrow (\boldsymbol{L}^T)^{-1}\boldsymbol{L}^{-1}(\boldsymbol{W}^T(\tilde{\boldsymbol{Y}} + \boldsymbol{V}) + \rho(\boldsymbol{H} + \boldsymbol{U})^T)$ using forward/backward substitution ;

  $\boldsymbol{H} \leftarrow \arg\min_{\boldsymbol{H}} r(\boldsymbol{H}) + \frac{\rho}{2}\|\boldsymbol{H} - \tilde{\boldsymbol{H}}^T + \boldsymbol{U}\|_F^2$;

  $\tilde{\boldsymbol{Y}} \leftarrow \arg\min_{\tilde{\boldsymbol{Y}}}\ l(\boldsymbol{Y} - \tilde{\boldsymbol{Y}}) + \frac{1}{2}\|\tilde{\boldsymbol{Y}} - \boldsymbol{W}\tilde{\boldsymbol{H}} + \boldsymbol{V}\|_F^2$;

  $\boldsymbol{U} \leftarrow \boldsymbol{U} + \boldsymbol{H} - \tilde{\boldsymbol{H}}^T$;

  $\boldsymbol{V} \leftarrow \boldsymbol{V} + \tilde{\boldsymbol{Y}} - \boldsymbol{W}\tilde{\boldsymbol{H}}$;

until $r < \varepsilon$ and $s < \varepsilon$; r and s defined in (4.10) and (4.11)

return $\boldsymbol{H}$, $\boldsymbol{U}$, $\tilde{\boldsymbol{Y}}$, and $\boldsymbol{V}$.

it still converges very fast in practice. The biggest drawback is that, by introducing the auxiliary variable $\tilde{\boldsymbol{Y}}$ and its dual variable $\boldsymbol{V}$, the big matrix product $\boldsymbol{W}^T(\tilde{\boldsymbol{Y}} + \boldsymbol{V})$ must be re-computed in each ADMM iteration, whereas in the previous case one only needs to compute $\boldsymbol{W}^T\boldsymbol{Y}$ once. This is the price we must pay; but it can be moderated by controlling the maximum number of ADMM iterations.

Scalability considerations. As big data analytics become increasingly common, it is important to keep scalability issues in mind as we develop new analysis methodologies and algorithms. Big data $\underline{\boldsymbol{Y}}$ is usually stored as a sparse array, i.e., a list of (`index,value`) pairs, with the unlisted entries regarded as zeros or missing. With the introduction of $\tilde{\boldsymbol{Y}}$ and $\boldsymbol{V}$, both of size($\underline{\boldsymbol{Y}}$), one hopes to be able to avoid dense operations. Fortunately, for some commonly used loss functions, this is possible. Notice that by defining $\bar{\boldsymbol{Y}} = \boldsymbol{W}\tilde{\boldsymbol{H}} - \boldsymbol{V}$, the $\boldsymbol{V}$-update essentially becomes

$$\boldsymbol{V} \leftarrow \tilde{\boldsymbol{Y}} - \bar{\boldsymbol{Y}},$$

which means a significant portion of entries in $\boldsymbol{V}$ are constants—0 if the entries are regarded as missing, ± 1 or $\pm\lambda$ in the robust fitting or Huber fitting case if the entries are regarded as "corrupted"—thus they can be efficiently stored as a sparse array. As for $\tilde{\boldsymbol{Y}}$, one can simply generate it "on-the-fly" using the closed-form we provided earlier (notice that $\bar{\boldsymbol{Y}}$ has the memory-efficient "low-rank plus sparse" structure). The only occasion that $\tilde{\boldsymbol{Y}}$ is needed is when computing $\boldsymbol{W}^T\tilde{\boldsymbol{Y}}$.

4.3 Summary of the Proposed Algorithm

We propose to use Alg. 4.1 or 4.2 as the core sub-routine for alternating optimization. The proposed "universal" multi-linear factorization algorithm is summarized as Alg. 4.3. A few remarks on implementing Alg. 4.3 are in order.

Since each factor $\boldsymbol{H}_d$ is updated in a cyclic fashion, one expects that after a certain number of cycles $\boldsymbol{H}_d$ (and its dual variable $\boldsymbol{U}_d$) obtained in the previous iteration will not be very far away from the update for the current iteration. In this sense, the outer AO framework naturally provides a good initial point to the inner ADMM iteration. With this warm-start strategy, the optimality gap for the sub-problem is then bounded by the per-step improvement of the AO algorithm, which is small. This mode of operation is crucial for insuring the efficiency of Alg. 4.3. Our experiments suggest that soon after an initial transient stage, the sub-problems can be solved in just one ADMM iteration (with reasonable precision).

Similar ideas can be used for $\tilde{\boldsymbol{Y}}$ and $\boldsymbol{V}$ in the matrix case if we want to deal with non-least-squares loss, and actually only one copy of them is needed in the updates of both factors. A few different options are available in the tensor case. If memory is not an issue in terms of the size of $\underline{\boldsymbol{Y}}$, a convenient approach that is commonly adopted in ALS implementations is to store all N matricizations $\boldsymbol{Y}_{(1)}, ..., \boldsymbol{Y}_{(N)}$, so they are readily available without need for repetitive data re-shuffling during run-time. If this practice is adopted, then it makes sense to also have N copies of $\underline{\tilde{\boldsymbol{Y}}}$ and $\underline{\boldsymbol{V}}$, in order to save computation. Depending on the size and nature of the data and how it is stored, it may be completely unrealistic to keep multiple copies of the data and the auxiliary variables, at which point our earlier discussion on scalable implementation of Alg. 4.2 for big but sparse data can be instrumental.

Algorithm 4.3: AO-ADMM for (4.1)

Initialize $\boldsymbol{H}_1$, ..., $\boldsymbol{H}_N$;
Initialize $\boldsymbol{U}_1$, ..., $\boldsymbol{U}_N$ to be all zero matrices;
if *least-squares loss* **then**
  repeat
    for $d = 1, ..., N$ **do**
      $\boldsymbol{Y} = \boldsymbol{Y}_{(d)}$ and $\boldsymbol{W} = \odot_{j \neq d} \boldsymbol{H}_j$; `// not necessarily formed explicitly`
      update $\boldsymbol{H}_d$, $\boldsymbol{U}_d$ using Alg. 4.1 initialized with the previous $\boldsymbol{H}_d$, $\boldsymbol{U}_d$;
    end
    update μ if necessary ; `// refer to (4.14)`
  until *some termination criterion is reached*;
else
  Initialize $\underline{\tilde{\boldsymbol{Y}}} \leftarrow \underline{\boldsymbol{Y}}$, $\underline{\boldsymbol{V}} \leftarrow \underline{\boldsymbol{0}}$;
  repeat
    for $d = 1, ..., N$ **do**
      $\boldsymbol{Y} = \boldsymbol{Y}_{(d)}$ and $\boldsymbol{W} = \odot_{j \neq d} \boldsymbol{H}_j$; `// not necessarily formed explicitly`
      update $\boldsymbol{H}_d$, $\boldsymbol{U}_d$, $\tilde{\boldsymbol{Y}}_{(d)}$, $\boldsymbol{V}_{(d)}$ using Alg. 4.2 initialized with the previous $\boldsymbol{H}_d$, $\boldsymbol{U}_d$, $\tilde{\boldsymbol{Y}}_{(d)}$, $\boldsymbol{V}_{(d)}$;
    end
    update μ if necessary ; `// refer to (4.14)`
  until *some termination criterion is reached*;
end

Sometimes an additional proximal regularization is added to the sub-problems. The benefit is two-fold: it helps the convergence of the AO outer-loop when $N \geq 3$; while for the ADMM inner-loop it improves the conditioning of the sub-problem, which may accelerate the convergence of ADMM, especially in the general loss function case when we do not have strong convexity. The convergence of AO-ADMM is summarized in Proposition 4.1.

Proposition 4.1. *If the sequence generated by AO-ADMM in Alg. 4.3 is bounded, then for*

1. $N = 2$,

2. $N > 2, \mu > 0$,

AO-ADMM converges to a stationary point of (4.1).

Proof. The first case with $\mu = 0$ is covered in [79, Theorem 3.1], and the cases when $\mu > 0$ are covered in [80, Theorem 2]. □

Note that for $N = 2$, using $\mu = 0$ yields faster convergence than $\mu > 0$. For $N > 2$, i.e., for tensor data, we can update μ as follows

$$\mu \leftarrow 10^{-7} + 0.01 \frac{\|\underline{\boldsymbol{Y}} - [\boldsymbol{H}_d]_{d=1}^N\|}{\|\underline{\boldsymbol{Y}}\|}, \tag{4.14}$$

which was proposed in [80] for unconstrained tensor factorization, and works very well in our context as well.

The convergence result in Proposition 4.1 has an additional assumption that the sequence generated by the algorithm is bounded. For unconstrained CP, diverging components may be encountered during AO iterations [93, 94], but adding Frobenius norm regularization for each matrix factor (with a small weight) ensures that the iterates remain bounded.

As we can see, the ADMM is an appealing sub-routine for alternating optimization, leading to a simple plug-and-play generalization of the workhorse ALS algorithm. Theoretically, they share the same per-iteration complexity if the number of inner ADMM iterations is small, which is true in practice, after an initial transient. Efficient implementation of the overall algorithm should include data-structure-specific algorithms for $\boldsymbol{W}^T\boldsymbol{Y}$ or $(\odot_{j\neq d}\boldsymbol{H}_j)^T\boldsymbol{Y}_{(d)}$, which dominate the per-iteration complexity, and may include parallel/distributed computation along the lines of [95].

Finally, if a non-least-squares loss is to be used, we suggest that the least-squares loss is first employed to get preliminary estimates (using Alg. 4.3 calling Alg. 4.1) which can then be fed as initialization to run Alg. 4.3 calling Alg. 4.2. The main disadvantage of Alg. 4.2 compared to Alg. 4.1 is that the big matrix (or tensor) multiplication $\boldsymbol{W}^T(\tilde{\boldsymbol{Y}} + \boldsymbol{V})$ needs to be calculated in each ADMM iteration. Therefore, this strategy can save a significant amount of computations at the initial stage.

4.4 Case Studies and Numerical Results

In this section we will study some well-known constrained matrix/tensor factorization problems, derive the corresponding update for $\boldsymbol{H}$ in Alg. 4.1 or $\boldsymbol{H}$ and $\tilde{\boldsymbol{Y}}$ in Alg. 4.2, and compare it to some of the state-of-the-art algorithms for that problem. In all examples we denote our proposed algorithm as **AO-ADMM**. All experiments are performed in MATLAB 2015a on a Linux server with 32 Xeon 2.00GHz cores and 128GB memory.

4.4.1 Non-negative Matrix and CP Factorization

Perhaps the most common constraint imposed on the latent factors is non-negativity – which is often supported by physical considerations (e.g., when the latent factors represent chemical concentrations, or power spectral densities) or other prior information, or simply because non-negativity sometimes yields interpretable factors [1]. Due to the popularity and wide range of applications of NMF, numerous algorithms have been proposed for fitting the NMF model, and most of them can be easily generalized to the tensor case. After a brief review of the existing algorithms for NMF, we compare our proposed algorithm to some of the best algorithms reported in the literature to showcase the efficiency of AO-ADMM.

Let us start by considering NMF with least-squares loss, which is the prevailing loss function in practice. By adopting the alternating optimization framework, the sub-problem that emerges for each matrix factor is non-negative (linear) least-squares (NNLS). Some of the traditional methods for NNLS are reviewed in [96] (interestingly, not including ADMM), and most of them have been applied to NMF or NCP, e.g., the active-set (AS) method [97, 98] and block-principle-pivoting (BPP) [99, 100]. Recall that in the context of the overall multi-linear factorization problem we actually need to solve a large number of (non-negative) least-squares problems sharing the same mixing matrix $\boldsymbol{W}$, and in the unconstrained case this means we only need to calculate the Cholesky factorization of $\boldsymbol{W}^T\boldsymbol{W}$ once. Unfortunately, this good property that enables high efficiency implementation of ALS is not preserved by either AS or BPP. Sophisticated methods that group similar rows to reduce the number of inversions have been proposed [101], although as k grows larger this does not seem appealing in the

worst case. Some other methods, like the multiplicative-update (MU) [102] or hierarchical alternating least squares (HALS) [92], ensure that the per-iteration complexity is dominated by calculating $\boldsymbol{W}^T\boldsymbol{W}$ and $\boldsymbol{W}^T\boldsymbol{Y}$, although more outer-loops are needed for convergence. These are actually one step majorization-minimization or block coordinate descent applied to the NNLS problem. An accelerated version of MU and HALS is proposed in [103], which essentially does a few more inner-loops after computing the most expensive $\boldsymbol{W}^T\boldsymbol{Y}$.

ADMM, on the other hand, may not be the fastest algorithm for a single NNLS problem, yet its overhead can be amortized when there are many NNLS problem instances sharing the same mixing matrix, especially if good initialization is readily available. This is in contrast to an earlier attempt to adopt ADMM to NMF [104], which did not use Cholesky caching, warm start, and a good choice of ρ to speed up the algorithm. Furthermore, ADMM can seamlessly incorporate different regularizations as well as non-least-squares loss.

We should emphasize that AO forms the backbone of our proposed algorithm – ADMM is only applied to the sub-problems. There are also algorithms that directly apply an ADMM approach to the whole problem [86,91,95]. The per-iteration complexity of those algorithms is also the same as the unconstrained alternating least-squares. However, due to the non-convexity of the whole problem, the loss is not guaranteed to decrease monotonically, unlike alternating optimization. Moreover, both ADMM and AO guarantee that every limit point is a stationary point, but in practice AO almost always converges (as long as the updates stay bounded), which is not the case for ADMM applied to the whole problem.

In another recent line of work [82], a similar idea of using an improved AO framework to ensure convergence is used. When [82] is specialized to non-negative matrix/tensor factorization, each update becomes a simple proximal-gradient step with an extrapolation. The resulting algorithm is also guaranteed to converge (likewise assuming that the iterates remain bounded), but it turns out to be slower than our algorithm, as we will show in our experiments. Some interesting work on non-negative CP can also been found in [105] and the references therein.

To apply our proposed algorithm to NMF or NCP with least-squares loss, Alg. 4.1

is used to solve the sub-problems, with line 8 customized as

$$\boldsymbol{H} \leftarrow \left[\tilde{\boldsymbol{H}}^T - \boldsymbol{U}\right]_+,$$

i.e., zeroing out the negative values of $(\tilde{\boldsymbol{H}}^T - \boldsymbol{U})$. The tolerance for the ADMM inner-loop is set to 0.01.

Non-negative matrix factorization

We compare AO-ADMM with the following algorithms:

AO-BPP. AO using block principle pivoting [99][1] ;

accHALS. Accelerated HALS [103][2] ;

APG. Alternating proximal gradient [82][3] ;

ADMM. ADMM applied to the whole problem [86] [4] .

AO-BPP and HALS are reported in [99] to outperform other methods, accHALS is proposed in [103] to improve HALS, APG is reported in [82] to outperform AO-BPP, and we include ADMM applied to the whole problem to compare the convergence behavior of AO and ADMM for this non-convex factorization problem.

The aforementioned NMF algorithms are tested on two datasets. One is a dense image data set, the Extended Yale Face Database B[5] , of size 32256×1932, where each column is a vectorized 168×192 image of a face, and the dataset is a collection of face images of 29 subjects under various poses and illumination conditions. The other one is the Topic Detection and Tracking 2 (TDT2) text corpus[6] , of size 10212×36771, which is a sparse document-term matrix where each entry counts the frequency of a term in one document.

The convergence of the relative error $\|\boldsymbol{Y} - \boldsymbol{W}\boldsymbol{H}^T\|_F / \|\boldsymbol{Y}\|_F$ versus time in seconds for the Extended Yale B dataset is shown in Fig. 4.1, with $k = 100$ on the left and

[1] http://www.cc.gatech.edu/~hpark/nmfsoftware.php
[2] https://sites.google.com/site/nicolasgillis/code
[3] http://www.math.ucla.edu/~wotaoyin/papers/bcu/matlab.html
[4] http://mcnf.blogs.rice.edu/
[5] http://vision.ucsd.edu/~leekc/ExtYaleDatabase/ExtYaleB.html
[6] http://www.cad.zju.edu.cn/home/dengcai/Data/TextData.html

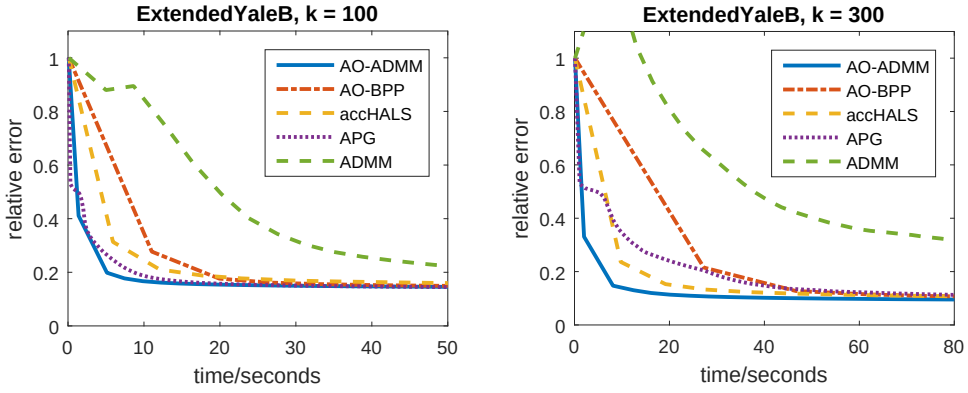


Figure 4.1: Convergence of some NMF algorithms on the Extended Yale B dataset.

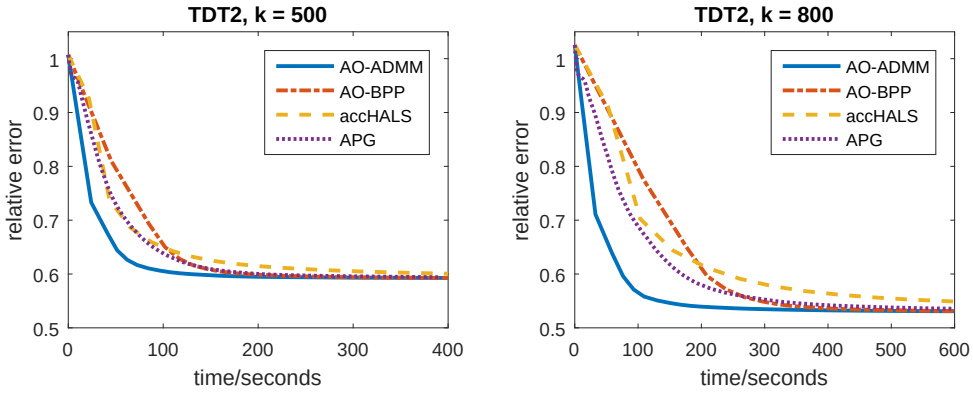


Figure 4.2: Convergence of some NMF algorithms on the TDT2 dataset.

$k = 300$ on the right; and for the TDT2 dataset in Fig. 4.2, with $k = 500$ on the left and $k = 800$ on the right. The ADMM algorithm [86] is not included for TDT2 because the code provided online is geared towards imputation of matrices with missing values – it does not treat a sparse input matrix as the full data, unless we fill-in all zeros.

We also tested these algorithms on synthetic data. For $m = n = 2000$ and $k = 100$, the true $\boldsymbol{W}$ and $\boldsymbol{H}$ are generated by drawing their elements from an i.i.d. exponential distribution with mean 1, and then 50% of the elements are randomly set to 0. The data matrix $\boldsymbol{Y}$ is then set to be $\boldsymbol{Y} = \boldsymbol{W}\boldsymbol{H}^T + \boldsymbol{N}$, where the elements of $\boldsymbol{N}$ are drawn from an i.i.d. Gaussian distribution with variance 0.01. The averaged results of 100 Monte-Carlo trials are shown in Table 4.1. As we can see, AO-based methods are able to attain

Table 4.1: Averaged performance of NMF algorithms on synthetic data.

Algorithm	$\|\boldsymbol{Y} - \boldsymbol{W}\boldsymbol{H}^T\|_F$	run time	iterations
AO-ADMM	193.1026	21.7s	86.9
AO-BPP	193.1516	40.9s	52.2
accHALS	193.1389	26.8s	187.0
APG	193.1431	25.3s	240.2
ADMM	193.6808	31.9s	125.2

smaller fitting errors than directly applying ADMM to this non-convex problem, while AO-ADMM provides the most efficient per-iteration complexity.

Non-negative CP factorization

Similar algorithms are compared in the NCP case:

AO-BPP. AO using block principle pivoting [100][1];

HALS. Hierarchical alternating least-squares [92][1];

APG. Alternating proximal gradient [82][2];

ADMM. ADMM applied to the whole problem [95];

SDF. Structured data fusion provided by `tensorlab` [106], using "all-at-once" updates based on quasi-Newton or Gauss-Newton method [107, 108].

For our proposed AO-ADMM algorithm, a diminishing proximal regularization term in the form (4.6) is added to each sub-problem to enhance the overall convergence, with the regularization parameter μ updated as (4.14).

Two real datasets are being tested: one is a dense CT image dataset[7] of size $260 \times 190 \times 150$, which is a collection of 150 CT images of a female's ankle, each with size 260×190; the other one is a sparse social network dataset – Facebook Wall Posts[8] , of size $46952 \times 46951 \times 1592$, that collects the number of wall posts from one Facebook

[7] `http://www.nlm.nih.gov/research/visible/`
[8] `http://konect.uni-koblenz.de/networks/facebook-wosn-wall`

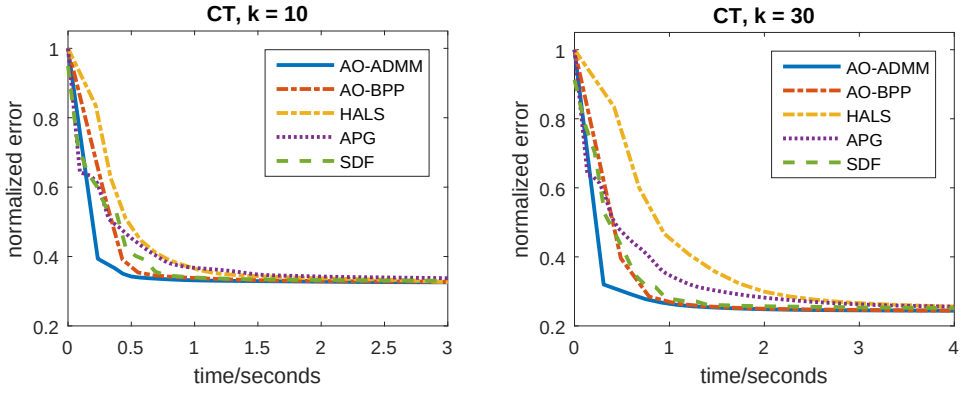


Figure 4.3: Convergence of some NCP algorithms on the CT dataset.

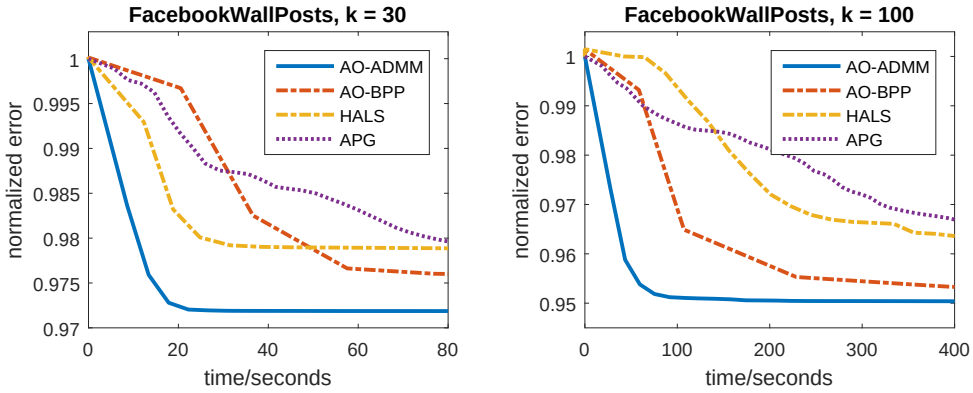


Figure 4.4: Convergence of some NCP algorithms on the Facebook Wall Posts dataset.

user to another over a period of 1592 days. The sparse tensor is stored in the `sptensor` format supported by the `tensor_toolbox` [109], and all the aforementioned algorithms use this toolbox to handle sparse tensor data, except SDF, which only accepts the sparse tensor structure defined by `tensorlab`. However, due to the algorithms being used by SDF, the memory requirement exceeded the limit for the latter case, thus it is omitted for the Facebook wall posts dataset.

Similar to the matrix case, the normalized root mean squared error versus time in seconds for the CT dataset is shown in Fig. 4.3, with $k = 10$ on the left and $k = 30$ on the right, and that for the Facebook Wall Posts data is shown in Fig. 4.4, with $k = 30$ on the left and $k = 100$ on the right. As we can see, AO-ADMM again converges the fastest,

Table 4.2: Averaged performance of NCP algorithms on synthetic data

Algorithm	$\|\underline{\boldsymbol{Y}} - [\boldsymbol{H}_1, \boldsymbol{H}_2, \boldsymbol{H}_3]\|$	run time	iterations
AO-ADMM	1117.597	145.2s	25.1
AO-BPP	1117.728	679.0s	22.6
HALS	1117.655	1838.7s	137.7
APG	1117.649	1077.4s	156.3
ADMM	1156.799	435.9s	77.2
SDF	1118.427	375.8s	N/A

not only because of the efficient per-iteration update from Alg. 4.1, but also thanks to the additional proximal regularization to help the algorithm avoid swamps, which are not uncommon in alternating optimization-based algorithms for tensor decomposition.

Monte-Carlo simulations were also conducted using synthetic data for 3-way nonnegative tensors with $n_1 = n_2 = n_3 = 500$ and $k = 100$, with the latent factors generated in the same manner as for the previous NMF synthetic data, and the tensor data generated as the low-rank model synthesized from those factors plus i.i.d. Gaussian noise with variance 0.01. The averaged result over 100 trials is given in Table 4.2. As we can see, AO-ADMM again outperforms all other algorithms in all cases considered.

4.4.2 Constrained Matrix and Tensor Completion

As discussed before, real-world data are often stored as a sparse array, i.e., in the form of (`index,value`) pairs. Depending on the application, the unlisted entries in the array can be treated as zeros, or as not (yet) observed but possibly nonzero. A well-known example of the latter case is the *Netflix prize problem*, which involves an array of movie ratings indexed by customer and movie. The data is extremely sparse, but the fact that a customer did not rate a movie does not mean that the customer's rating of that movie would be zero—and the goal is actually to predict those unseen ratings to provide good movie recommendations.

For matrix data with no constraints on the latent factors, convex relaxation techniques that involve the matrix nuclear norm have been proposed with provable matrix

reconstruction bounds [65]. Some attempts have been made to generalize the matrix nuclear norm to tensor data [110, 111], but that boils down to the Tucker model rather than the CP model that we consider here. A key difference is that Tucker modeling can only hope to impute (recover missing values) in the data, whereas CP can uniquely recover the latent factors – the important 'dimensions' of consumer preference in this context. Another key difference is that the aforementioned convex relaxation techniques cannot incorporate constraints on the latent factors, which can improve the estimation performance. Taking the Netflix problem as an example, *user-bias* and *movie-bias* terms are often successfully employed in recommender systems; these can be easily subsumed in the factorization formulation by constraining, say, the first column of $\boldsymbol{W}$ and the second column of $\boldsymbol{H}$ to be equal to the all-one vector. Moreover, interpreting each column of $\boldsymbol{W}$ ($\boldsymbol{H}$) as the appeal of a certain movie genre to the different users (movie ratings for a given type of user, respectively), it is natural to constrain the entries of $\boldsymbol{W}$ and $\boldsymbol{H}$ to be non-negative.

When matrix/tensor completion is formulated as a constrained factorization problem using a loss function as in Sec. 4.2.3, there are traditionally two ways to handle it. One is directly using alternating optimization, although due to the random positions of the missing values, the least-squares problem for each row of $\boldsymbol{H}$ will involve a different subset of the rows of $\boldsymbol{W}$, thus making the update inefficient even in the unconstrained case. A more widely used way is an instance of expectation-maximization (EM): one starts by filling the missing values with zeros, and then iteratively fits a (constrained) low-rank model and imputes the originally missing values with predictions from the interim low-rank model. More recently, an ADMM approach that uses an auxiliary variable for the full data was proposed [86], although if we look carefully at that auxiliary variable, it is exactly equal to the filled-in data given by the EM method.

In fact, the auxiliary variable $\tilde{\boldsymbol{Y}}$ that we introduce is similar to that of [86], thus also related to the way that EM imputes the missing values—one can treat our method as imputing the missing values per ADMM inner-loop, the method in [86] as imputing per iteration, and EM as imputing after several iterations. However, our proposed AO-ADMM is able to give better results than EM, despite the similarities. As an illustrative example, consider the Amino acids fluorescence data[9] , which is a $5 \times 201 \times 61$ tensor

[9] `http://www.models.kvl.dk/Amino_Acid_fluo`

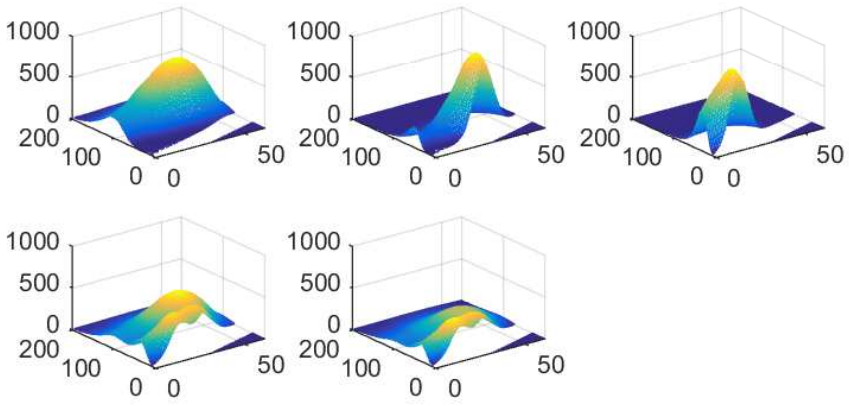


Figure 4.5: Illustration of the missing values in the Amino acids fluorescence data.

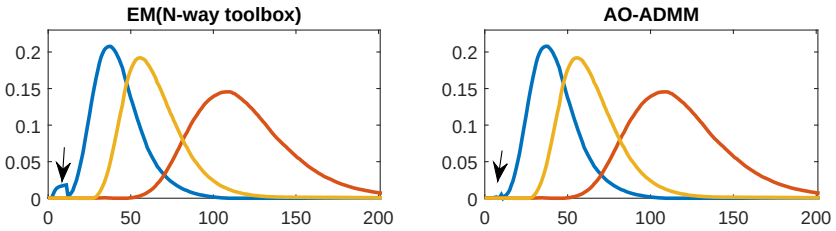


Figure 4.6: The emission loadings ($\boldsymbol{H}_2$) produced by the N-way toolbox on the left, which uses EM, and by AO-ADMM on the right.

known to be generated by a rank-3 NCP model [112]. However, some of the entries are known to be badly contaminated, and are thus deleted, as shown in Fig. 4.5. Imposing non-negativity on the latent factors, the emission loadings $\boldsymbol{H}_2$ of the three chemical components provided by the EM method using the N-way toolbox [113] and AO-ADMM are shown in Fig. 4.6. While both results are satisfactory, AO-ADMM is able to suppress the artifacts caused by the systematically missing values in the original data, as indicated by the arrows in Fig. 4.6.

We now evaluate our proposed AO-ADMM on a movie rating dataset called MovieLens[10] , which consists of 100,000 movie ratings from 943 users on 1682 movies. MovieLens includes 5 sets of 80%-20% splits of the ratings for training and testing, and for

[10] `http://grouplens.org/datasets/movielens/`

each split we fit a matrix factorization model based on the 80% training data, and evaluate the correctness of the model on the 20% testing data. The averaged performance on this 5-fold cross validation is shown in Fig. 4.7, where we used the mean absolute error (MAE) for comparison with the classical collaborative filtering result [114] (which attains a MAE of 0.73). On the left of Fig. 4.7, we used the traditional least-squares criterion to fit the available ratings, whereas on the right we used the Kullback-Leibler divergence for fitting, since it is a meaningful statistical model for integer data. For each fitting criterion, we compared the performance by imposing Tikhonov regularization $(\lambda/2)\|\cdot\|_F^2$ with $\lambda = 0.1$, or non-negativity, or non-negativity with biases (i.e., in addition constraining the first column of $\boldsymbol{W}$ and second column of $\boldsymbol{H}$ to be all ones). Some observations are as follows:

- Low-rank indeed seems to be a good model for this movie rating data, and the right rank seems to be 4 or 5, higher rank leads to over-fitting, as evident from Fig. 4.7;
- Imposing non-negativity reduces the over-fitting at higher ranks, whereas the fitting criterion does not seem to be playing a very important role in terms of performance;
- By adding biases, the best case prediction MAE at rank 4 is less than 0.69, an approximately 6% improvement over the best result reported in [114].

Notice that our aim here is to showcase how AO-ADMM can be used to explore possible extentions to the matrix completion problem formulation, rather than come up with the best recommender system method, which would require significant exploration in its own right. We believe with the versability of AO-ADMM, researchers can easily test various models for matrix/tensor completion, and quickly narrow down the one that works the best for their specific application.

4.4.3 Dictionary Learning

Many natural signals can be represented as an (approximately) sparse linear combination of some (possibly over-complete) basis, for example the Fourier basis for speech signals and the wavelet basis for images. If the basis (or *dictionary* when over-complete)

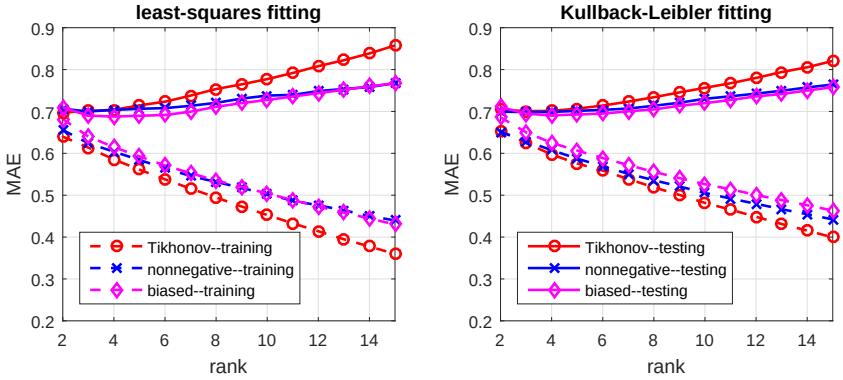


Figure 4.7: Training and testing mean absolute error (MAE) versus model rank of the MovieLens data, averaged over a 5-fold cross validation, comparing least-squares fitting (on the left) and Kullback-Leibler fitting (on the right), with Tikhonov regularization, non-negativity constraint, or non-negativity with biases on the latent factors.

is known, one can directly do data compression via greedy algorithms or convex relaxations to obtain the sparse representation [115], or even design the sensing procedure to reduce the samples required for signal recovery [116]. If the dictionary is not known, then one can resort to the so called *dictionary learning* (DL) to try to learn a sparse representation [117], if one exists. The well-known benchmark algorithm for DL is called k-SVD [118], which is a geometry-based algorithm, and can be viewed as a generalization of the clustering algorithms k-means and k-planes. However, as noted in the original paper, k-SVD does not scale well as the size of the dictionary increases. Thus k-SVD is often used to construct a dictionary of small image patches of size 8×8, with a few hundreds of atoms.

DL can also be formulated as a matrix factorization problem

$$\begin{aligned} \underset{\boldsymbol{D},\boldsymbol{S}}{\text{minimize}} \quad & \frac{1}{2}\|\boldsymbol{Y} - \boldsymbol{D}\boldsymbol{S}\|_F^2 + r(\boldsymbol{S}) \\ \text{subject to} \quad & \boldsymbol{D} \in \mathcal{D}, \end{aligned} \tag{4.15}$$

where $r(\cdot)$ is a sparsity inducing regularization, e.g., the cardinality, the ℓ_1 norm, or the log penalty; conceptually there is no need for a constraint on $\boldsymbol{D}$, however, due to the scaling ambiguity inherent in the matrix factorization problem, we need to impose some norm constraint on the scaling of $\boldsymbol{D}$ to make the problem better defined. For example,

we can bound the norm of each atom in the dictionary, $||\boldsymbol{d}_i|| \leq 1, \forall i = 1, ..., k$, where $\boldsymbol{d}_i$ is the i-th column of $\boldsymbol{D}$, and we adopt this constraint here.

Although bounding the norm of the columns of $\boldsymbol{D}$ works well, it also complicates the update of $\boldsymbol{D}$—without this constraint, each row of $\boldsymbol{D}$ is the solution of an independent least-squares problem sharing the same mixing matrix, while the constraint couples the columns of $\boldsymbol{D}$, making the problem non-separable. Existing algorithms either solve it approximately [119] or by sub-optimal methods like cyclic column updates [120]. On the other hand, this is not a problem at all for our proposed ADMM sub-routine Alg. 4.1: the row separability of the cost function and the column separability of the constraints are handled separately by the two primal variable blocks, while our previously discussed Cholesky caching, warm starting, and good choice of ρ ensure that an exact dictionary update can be done very efficiently.

The update of $\boldsymbol{S}$, sometimes called the sparse coding step, is a relatively well-studied problem for which numerous algorithms have been proposed. We mainly focus on the ℓ_1 regularized formulation, in which case the sub-problem becomes the well-known LASSO, and in fact a large number of LASSOs sharing the same mixing matrix. Alg. 4.1 can be used by replacing the proximity step with the soft-thresholding operator. Furthermore, if an over-complete dictionary is trained, the least-squares step can also be accelerated by using the matrix inversion lemma:

$$(\boldsymbol{D}^T\boldsymbol{D} + \rho\boldsymbol{I})^{-1} = \rho^{-1}\boldsymbol{I} - \rho^{-1}\boldsymbol{D}^T(\rho\boldsymbol{I} + \boldsymbol{D}\boldsymbol{D}^T)^{-1}\boldsymbol{D}.$$

Thus, if $m \ll k$, one can cache the Cholesky of $\rho\boldsymbol{I} + \boldsymbol{D}\boldsymbol{D}^T = \boldsymbol{L}\boldsymbol{L}^T$ instead, and replace the least-squares step in Alg. 4.1 with

$$\tilde{\boldsymbol{S}} \leftarrow \rho^{-1}(\boldsymbol{B} - \boldsymbol{D}^T(\boldsymbol{L}^T)^{-1}\boldsymbol{L}^{-1}\boldsymbol{D}\boldsymbol{B}),$$

where $\boldsymbol{B} = \boldsymbol{D}^T\boldsymbol{Y} + \rho(\boldsymbol{S} + \boldsymbol{U})$. The use of ADMM for LASSO is also discussed in [121–123], and [83], and we generally followed the one described in [83, §6]. Again, one should notice that compared to a plain LASSO, our LASSO sub-problem in the AO framework comes with a good initialization, therefore only a very small number of ADMM-iterations are required for convergence.

It is interesting to observe that for the particular constraints and regularization used in DL, incorporating non-negativity maintains the simplicity of our proposed

algorithm—for both the norm bound constraint and ℓ_1 regularization, the proximity operator in Alg. 4.1 with non-negativity constraint simply requires zeroing out the negative values before doing the same operations. In some applications non-negativity can greatly help the identification of the dictionary [124].

As an illustrative example, we trained a dictionary from the MNIST handwritten digits dataset[11] , which is a collection of gray-scale images of handwritten digits of size 28×28, and for each digit we randomly sampled 1000 images, forming a matrix of size $784 \times 10,000$. Non-negativity constraints are imposed on both the dictionary and the sparse coefficients. For $k = 100$, and by setting the ℓ_1 penalty parameter $\lambda = 0.5$, the trained dictionary after 100 AO-ADMM (outer-)iterations is shown in Fig. 4.8. On average approximately 11 atoms are used to represent each image, and the whole model is able to describe approximately 60% of the energy of the original data, and the entire training time takes about 40 seconds. Most of the atoms in the dictionary remain readable, which shows the good interpretability afforded by the additional non-negativity constraint.

For comparison, we tried the same data set with the same parameter settings with the popular and well-developed DL package SPAMS[12] . For fair comparison, we used SPAMS in batch mode with batch size equal to the size of the training data, and run it for 100 iterations (same number of iterations as AO-ADMM). The quality of the SPAMS dictionary is almost the same as that of AO-ADMM, but it takes SPAMS about 3 minutes to run through these 100 iterations, versus 40 seconds for AO-ADMM. The performance does not change much if we remove the non-negativity constraint when using SPAMS, although the resulting dictionary then loses interpretability. Notice that SPAMS is fully developed in C++, whereas our implementation is simply written in MATLAB, which leaves considerable room for speed improvement using a lower-level language compiler.

[11] `http://www.cs.nyu.edu/~roweis/data.html`
[12] `http://spams-devel.gforge.inria.fr/index.html`

Figure 4.8: Trained dictionary from the MNIST handwritten digits dataset.

4.5 Conclusion

In this paper we proposed a novel AO-ADMM algorithmic framework for matrix and tensor factorization under a variety of constraints and loss functions. The main advantages of the proposed AO-ADMM framework are:

- **Efficiency.** By carefully adopting AO as the optimization backbone and ADMM for the individual sub-problems, a significant part of the required computations can be effectively cached, leading to a per-iteration complexity similar to the workhorse ALS algorithm for unconstrained factorization. Warm-start that is naturally provided by AO together with judicious regularization and choice of parameters further reduce the number of inner ADMM and outer AO iterations.

- **Flexibility.** Thanks to ADMM, which is a special case of the proximal algorithm, non-least-squares terms can be handled efficiently with element-wise complexity using the well-studied proximity operators. This includes almost all non-parametric constraints and regularization penalties commonly imposed on the factors, and even non-least-squares fitting criteria.

- **Convergence.** AO guarantees monotone decrease of the loss function, which is a nice property for the NP-hard factorization problems considered. Moreover, recent advances on generalizations of the traditional BCD algorithms further guarantee convergence to a stationary point.

Case studies on non-negative matrix/tensor factorization, constrained matrix/tensor completion, and dictionary learning, with extensive numerical experiments using real data, corroborate our main claims. We believe that AO-ADMM can serve as a plug-and-play framework that allows easy exploration of different types of constraints and loss functions, as well as different types of matrix and tensor (co-)factorization models.

Chapter 5

Performance Analysis via the Cramér-Rao Bound

We have discussed the uniqueness of several matrix factorization models, in complement to the already well-known and celebrated uniqueness of the CP decomposition for tensors, and an optimization algorithm that works well for constrained matrix/CP factorization in terms of minimizing the fitting error. On the other hand, these NP-hard models were so successfully applied in practice thanks to their ability to identify the true latent factors. Therefore, apart from making sure that the algorithm is good at finding latent factors that fits the data well, but also is close to the true latent factors even if the data have noises.

The Cramér-Rao bound (CRB) [125, Ch. 3] is the most widely used estimation benchmark in signal processing. In many cases it is relatively easy to compute, and it is asymptotically achievable by maximum likelihood (ML) estimators in high signal to noise ratio (SNR) scenarios [125, pp. 164]. In other cases, there may be technical difficulties in deriving (or complexity issues in computing) the pertinent CRB; but due to the central role of this bound in signal processing research, work on developing CRB tools continues [126–129], thereby enlarging the set of problems for which the CRB can be used in practice.

After a brief review of the Cramér-Rao bound and some of its modern developments, we will derive the Cramér-Rao for both matrix and CP factorizations. The Fisher

information matrices (FIM) for the matrix and CP factorization models are irrespective to the constraints imposed onto the latent factors, even though some of the constraints (like non-negativity) are crucial in terms of identifiability. We also discuss efficient ways to (peudo-)invert the FIM to avoid the massive requirement of computation and memory when the problem size is moderately large, since FIM can easily become huge as it is a symmetric matrix with number of rows equal to the number of parameters we want to estimate. Finally, we put some NMF algorithms to the test and evaluate how these algorithms work for this NP-hard problem in terms of estimating the true latent factors.

5.1 The Cramér-Rao Bound

Suppose a set of measurements $\boldsymbol{y}$ is drawn from a probability density function $p(\boldsymbol{y};\boldsymbol{\theta})$ parameterized by $\boldsymbol{\theta}$, and our goal is to estimate $\boldsymbol{\theta}$ given the realizations of $\boldsymbol{y}$. If the regularity condition

$$\mathsf{E}\left\{\nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{y};\boldsymbol{\theta})\right\} = 0$$

is satisfied, we can define the Fisher information matrix (FIM) as

$$\boldsymbol{\Phi} = -\mathsf{E}\left\{\nabla^2_{\boldsymbol{\theta}} \log p(\boldsymbol{y};\boldsymbol{\theta})\right\},$$

which can be shown to be equal to [125]

$$\boldsymbol{\Phi} = \mathsf{E}\left\{\nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{y};\boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{y};\boldsymbol{\theta})^T\right\};$$

then for any unbiased estimator $\hat{\boldsymbol{\theta}}$, i.e., $\mathsf{E}\left\{\hat{\boldsymbol{\theta}}\right\} = \boldsymbol{\theta}$, we have

$$\mathsf{cov}\{\hat{\boldsymbol{\theta}}\} = \mathsf{E}\left\{(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta})(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta})^T\right\} \succeq \boldsymbol{\Phi}^{-1},$$

or we can simply take

$$\mathsf{E}\left\{\|\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}\|^2\right\} \geq \mathrm{trace}\left\{\boldsymbol{\Phi}^{-1}\right\}.$$

A simple way to prove the CRB is as follows. Let us look at the following covariance

$$\mathsf{E}\left\{ \begin{bmatrix} \hat{\boldsymbol{\theta}} - \boldsymbol{\theta} \\ \nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{y};\boldsymbol{\theta}) \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{\theta}} - \boldsymbol{\theta} \\ \nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{y};\boldsymbol{\theta}) \end{bmatrix}^T \right\} = \begin{bmatrix} \mathsf{cov}\{\hat{\boldsymbol{\theta}}\} & \boldsymbol{G} \\ \boldsymbol{G}^T & \boldsymbol{\Phi} \end{bmatrix} \succeq 0 \tag{5.1}$$

where $\boldsymbol{G} = \mathsf{E}\left\{(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta})\nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{y};\boldsymbol{\theta})^T\right\}$. According to Schur complement [35, Appendix A.5.5], if $\boldsymbol{\Phi} \succ 0$, then (5.1) holds iff

$$\mathsf{cov}\{\hat{\boldsymbol{\theta}}\} - \boldsymbol{G}\boldsymbol{\Phi}^{-1}\boldsymbol{G}^T \succeq 0. \tag{5.2}$$

Looking at

$$\begin{aligned}\mathsf{E}\left\{\nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{y};\boldsymbol{\theta})\right\} &= \int_{\mathcal{Y}} \left(\nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{y};\boldsymbol{\theta})\right) p(\boldsymbol{y};\boldsymbol{\theta}) d\boldsymbol{y} \\ &= \int_{\mathcal{Y}} \nabla_{\boldsymbol{\theta}} p(\boldsymbol{y};\boldsymbol{\theta}) d\boldsymbol{y},\end{aligned}$$

suppose the support of the random variable $\boldsymbol{y}$, denoted as $\mathcal{Y}$, is independent of $\boldsymbol{\theta}$, then we can reverse the order of derivative and integration, leading to

$$\mathsf{E}\left\{\nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{y};\boldsymbol{\theta})\right\} = \nabla_{\boldsymbol{\theta}} \int_{\mathcal{Y}} p(\boldsymbol{y};\boldsymbol{\theta}) d\boldsymbol{y} = 0,$$

which gives us the regularity condition. Then for the matrix $\boldsymbol{G}$ we have that

$$\begin{aligned}\boldsymbol{G} &= \mathsf{E}\left\{(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta})\nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{y};\boldsymbol{\theta})^T\right\} \\ &= \mathsf{E}\left\{\hat{\boldsymbol{\theta}}\nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{y};\boldsymbol{\theta})^T\right\} \\ &= \int_{\mathcal{Y}} \hat{\boldsymbol{\theta}} \left(\nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{y};\boldsymbol{\theta})\right)^T p(\boldsymbol{y};\boldsymbol{\theta}) d\boldsymbol{y} \\ &= \int_{\mathcal{Y}} \hat{\boldsymbol{\theta}} \nabla_{\boldsymbol{\theta}} p(\boldsymbol{y};\boldsymbol{\theta})^T d\boldsymbol{y} \\ &= \mathcal{D}_{\boldsymbol{\theta}} \mathsf{E}\left\{\hat{\boldsymbol{\theta}}\right\}^T = \boldsymbol{I},\end{aligned}$$

where we again used the fact that the order of integral and derivative can be reversed, and that $\hat{\boldsymbol{\theta}}$ is unbiased $\mathsf{E}\left\{\hat{\boldsymbol{\theta}}\right\} = \boldsymbol{\theta}$. Thus, we plug it back into (5.2) and obtain the Cramér-Rao bound

$$\mathsf{cov}\{\hat{\boldsymbol{\theta}}\} \succeq \boldsymbol{\Phi}^{-1}.$$

If the FIM $\boldsymbol{\Phi}$ is singular, we can use the generalized Schur complement result [35, §A.5.5] to conclude that

$$\begin{bmatrix} \mathsf{cov}\{\hat{\boldsymbol{\theta}}\} & \boldsymbol{H} \\ \boldsymbol{H}^T & \boldsymbol{\Phi} \end{bmatrix} \succeq 0$$

if and only if

$$\boldsymbol{\Phi} \succeq 0, \quad (\boldsymbol{I} - \boldsymbol{\Phi}\boldsymbol{\Phi}^{\dagger})\boldsymbol{G} = 0, \quad \mathsf{cov}\{\hat{\boldsymbol{\theta}}\} - \boldsymbol{G}\boldsymbol{\Phi}^{\dagger}\boldsymbol{G}^T \succeq 0.$$

This means that:

1. $\mathsf{E}\left\{\|\hat{\boldsymbol{\theta}}-\boldsymbol{\theta}\|^2\right\} \geq \operatorname{trace}\left\{\boldsymbol{\Phi}^{\dagger}\right\}$ is still a valid bound;
2. this looser bound is in theory not attainable, because $\boldsymbol{I}-\boldsymbol{\Phi}\boldsymbol{\Phi}^{\dagger} \neq 0$.

5.1.1 CRB and identifiability

It is natural to suspect that the singularity of FIM is caused by the fact that the model is not identifiable (meaning the solution is not unique in the noiseless case). However, identifiability in general neither implies nor is implied by a non-singular FIM. A famous example is given in [130]: consider the scalar signal model

$$y=\theta^2+\nu, \text{where } \nu \sim \mathcal{N}(0,\sigma^2),$$

the FIM with respect to θ is

$$\Phi=\frac{4}{\sigma^2}\theta^2;$$

interestingly, $\Phi=0$ if and only if $\theta=0$, the only identifiable point. Experience shows that the rank deficiency of FIM is usually related to trivial ambiguities of the problem, but not the critical ones. Take phase retrieval as an example, it has been shown that the FIM corresponding to this problem is always rank one deficient [131, 132], which, by identifying its null space, seems to be highly related to the global phase ambiguity inherent to this problem. For certain measurement systems, e.g., 1D Fourier measurement, the problem is not identifiable besides the trivial phase ambiguity, but the rank deficiency is still one [133, 134], which means the critical non-uniqueness issue is not revealed by the singularity of FIM. The practical implication for us is that, for trivial ambiguities, for example the permutation and scaling ambiguity in the matrix and CP factorization considered in this dissertation, we should do whatever we can to resolve these trivial ambiguities, and simply compare the MSE with the generalized CRB obtained from the pseudo-inverse of the singular FIM.

5.1.2 CRB under constraints

Suppose for estimating $\boldsymbol{\theta}$ we have the prior information that

$$\boldsymbol{f}(\boldsymbol{\theta}) \leq 0, \quad \boldsymbol{g}(\boldsymbol{\theta})=0.$$

Roughly speaking, the inequality constraints do not affect the CRB. For equality constraints, denote the Jacobian matrix of the vector function $\boldsymbol{g}(\boldsymbol{\theta})$ at point $\boldsymbol{\theta}$ as $\mathcal{D}_{\boldsymbol{\theta}}\boldsymbol{g}(\boldsymbol{\theta})$, i.e.,

$$[\ \mathcal{D}_{\boldsymbol{\theta}}\boldsymbol{g}(\boldsymbol{\theta})\ ]_{i,j} = \frac{\partial g_i(\boldsymbol{\theta})}{\theta_j};$$

we can find a matrix $\boldsymbol{Q}$ with ortho-normal columns that spans the null space of $\mathcal{D}_{\boldsymbol{\theta}}\boldsymbol{g}(\boldsymbol{\theta})$, i.e.,

$$\mathcal{D}_{\boldsymbol{\theta}}\boldsymbol{g}(\boldsymbol{\theta})^T\boldsymbol{Q} = 0, \quad \boldsymbol{Q}^T\boldsymbol{Q} = \boldsymbol{I}.$$

Then the constrained CRB is modified as follows

$$\mathsf{E}\left\{\|\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}\|^2\right\} \geq \operatorname{trace}\left\{\boldsymbol{Q}\left(\boldsymbol{Q}^T\boldsymbol{\Phi}\boldsymbol{Q}\right)^{\dagger}\boldsymbol{Q}^T\right\}.$$

5.1.3 CRB under Gaussian noise

Suppose the data model admits the form

$$\boldsymbol{y} = \boldsymbol{\varphi}(\boldsymbol{\theta}) + \boldsymbol{\nu}, \tag{5.3}$$

where $\boldsymbol{\nu}$ are i.i.d. Gaussian noise with variance σ^2, the most commonly used noise model in practice. In this case, it can be shown that the Fisher information matrix admits a very simple form [130], as presented in the following.

Proposition 5.1. *The Fisher information matrix for the data model (5.3) is given by*

$$\boldsymbol{\Phi} = \frac{1}{\sigma^2}\mathcal{D}_{\boldsymbol{\theta}}\boldsymbol{\varphi}(\boldsymbol{\theta})^T\mathcal{D}_{\boldsymbol{\theta}}\boldsymbol{\varphi}(\boldsymbol{\theta}).$$

Proof. For i.i.d. Gaussian noise $\boldsymbol{\nu}$, the log-likelihood is simply given by

$$\log p(\boldsymbol{y};\boldsymbol{\theta}) = -\frac{1}{2\sigma^2}\|\boldsymbol{y} - \boldsymbol{\varphi}(\boldsymbol{\theta})\|^2,$$

a non-linear least squares function. The Hessian matrix of it at point $\boldsymbol{\theta}$ has the form [76, §1.5]

$$-\nabla^2_{\boldsymbol{\theta}}\log p(\boldsymbol{y};\boldsymbol{\theta}) = \frac{1}{\sigma^2}\left(\mathcal{D}_{\boldsymbol{\theta}}\boldsymbol{\varphi}(\boldsymbol{\theta})^T\mathcal{D}_{\boldsymbol{\theta}}\boldsymbol{\varphi}(\boldsymbol{\theta}) + \sum_i (y_i - \varphi_i(\boldsymbol{\theta}))\nabla^2_{\boldsymbol{\theta}}\varphi_i(\boldsymbol{\theta})\right).$$

The FIM is taken as the expected value of $\nabla^2_{\boldsymbol{\theta}}\log p(\boldsymbol{y};\boldsymbol{\theta})$ over $\boldsymbol{y}$. Notice that in the above equation, $\boldsymbol{y}$ only appears in the second term; furthermore, according to our data

model, we have that $\mathsf{E}\{y_i\} = \varphi_i(\boldsymbol{\theta})$, which means the second term becomes zero after we take expectation. Hence, we have that

$$\boldsymbol{\Phi} = -\mathsf{E}\left\{\nabla^2_{\boldsymbol{\theta}} \log p(\boldsymbol{y};\boldsymbol{\theta})\right\} = \frac{1}{\sigma^2}\mathcal{D}_{\boldsymbol{\theta}}\boldsymbol{\varphi}(\boldsymbol{\theta})^T\mathcal{D}_{\boldsymbol{\theta}}\boldsymbol{\varphi}(\boldsymbol{\theta}).$$

□

5.1.4 CRB under non-Gaussian noise

In some cases we may observe that the noise is more heavy-tailed, thus we may wish to model noise different from the Gaussian distribution, for example the Laplacian or Cauchy. Luckily, it has been shown that the CRB for a family of non-Gaussian noise models are simply scaled versions of their Gaussian counter-parts [135,136]. Specifically, it is often the case that the FIM under Gaussian noise with zero mean and variance σ^2 takes the form

$$\boldsymbol{\Phi} = \frac{1}{\sigma^2}\boldsymbol{\Psi},$$

which could be derived via an easier way as we discussed before; then for the same model, if the additive noise ν is changed to Laplacian noise

$$p(\nu) = \frac{1}{2b}\exp\left(-\frac{|\nu|}{b}\right),$$

the modified CRB is

$$\boldsymbol{\Phi} = \frac{2}{b^2}\boldsymbol{\Psi};$$

for Cauchy noise with distribution

$$p(\nu) = \frac{1}{\pi\gamma}\left(\frac{\gamma^2}{\nu^2+\gamma^2}\right),$$

the CRB becomes

$$\boldsymbol{\Phi} = \frac{1}{2\gamma^2}\boldsymbol{\Psi}.$$

5.2 Cramér-Rao Bound for Matrix Factorization Models

Consider the $m \times n$ matrix generated as

$$\boldsymbol{Y} = \boldsymbol{W}\boldsymbol{H}^T + \boldsymbol{N},$$

where $\boldsymbol{W}$ is $m \times k$, $\boldsymbol{H}$ is $n \times k$, and the elements of $\boldsymbol{N}$ are drawn from an i.i.d. Gaussian distribution with zero-mean and variance σ^2. Then the log-likelihood of $\boldsymbol{Y}$ parameterized by $\boldsymbol{W}$ and $\boldsymbol{H}$ is

$$\log p(\boldsymbol{Y}; \boldsymbol{W}, \boldsymbol{H}) = -\frac{1}{2\sigma^2} \left\| \boldsymbol{Y} - \boldsymbol{W}\boldsymbol{H}^T \right\|_F^2 = -\frac{1}{2\sigma^2} \left\| \mathsf{vec}\left(\boldsymbol{Y}\right) - \boldsymbol{\varphi}(\boldsymbol{\theta}) \right\|^2,$$

where the unknown parameters we want to estimate, $\boldsymbol{W}$ and $\boldsymbol{H}$, are stacked into a single long vector of size $(m+n)k$ as follows

$$\boldsymbol{\theta} = \begin{bmatrix} \mathsf{vec}\left(\boldsymbol{W}\right)^T & \mathsf{vec}\left(\boldsymbol{H}\right)^T \end{bmatrix}^T,$$

and the non-linear function

$$\begin{aligned} \boldsymbol{\varphi}(\boldsymbol{\theta}) &= \mathsf{vec}\left(\boldsymbol{W}\boldsymbol{H}^T\right) \\ &= (\boldsymbol{H} \otimes \boldsymbol{I}_m)\mathsf{vec}\left(\boldsymbol{W}\right) \qquad (5.4) \\ &= \boldsymbol{C}_{n,m}\mathsf{vec}\left(\boldsymbol{H}\boldsymbol{W}^T\right) \\ &= \boldsymbol{C}_{n,m}(\boldsymbol{W} \otimes \boldsymbol{I}_n)\mathsf{vec}\left(\boldsymbol{H}\right), \qquad (5.5) \end{aligned}$$

where $\boldsymbol{C}_{m,n}$ represents the commutation matrix of size $mn \times mn$, cf. §1.1.

5.2.1 The Fisher Information Matrix

Invoking Proposition 5.1, the FIM for matrix factorization model under Gaussian noise has the form

$$\boldsymbol{\Phi} = \frac{1}{\sigma^2}\boldsymbol{\Psi} = \frac{1}{\sigma^2}\mathcal{D}_{\boldsymbol{\theta}}\boldsymbol{\varphi}(\boldsymbol{\theta})^T\mathcal{D}_{\boldsymbol{\theta}}\boldsymbol{\varphi}(\boldsymbol{\theta}).$$

The Jacobian matrix of $\boldsymbol{\varphi}(\boldsymbol{\theta})$ can be partitioned into two blocks

$$\mathcal{D}_{\boldsymbol{\theta}}\boldsymbol{\varphi}(\boldsymbol{\theta}) = \begin{bmatrix} \mathcal{D}_{\boldsymbol{W}}\boldsymbol{\varphi}(\boldsymbol{\theta}) & \mathcal{D}_{\boldsymbol{H}}\boldsymbol{\varphi}(\boldsymbol{\theta}) \end{bmatrix},$$

and according to (5.4) and (5.5), we have that

$$\begin{aligned} \mathcal{D}_{\boldsymbol{W}}\boldsymbol{\varphi}(\boldsymbol{\theta}) &= \boldsymbol{H} \otimes \boldsymbol{I}_m, \\ \mathcal{D}_{\boldsymbol{H}}\boldsymbol{\varphi}(\boldsymbol{\theta}) &= \boldsymbol{C}_{n,m}(\boldsymbol{W} \otimes \boldsymbol{I}_n). \end{aligned}$$

Using properties of the commutation matrices, we have that

$$\begin{aligned} \mathcal{D}_{\boldsymbol{W}}\boldsymbol{\varphi}(\boldsymbol{\theta})^T\mathcal{D}_{\boldsymbol{W}}\boldsymbol{\varphi}(\boldsymbol{\theta}) &= \boldsymbol{H}^T\boldsymbol{H} \otimes \boldsymbol{I}_m, \\ \mathcal{D}_{\boldsymbol{H}}\boldsymbol{\varphi}(\boldsymbol{\theta})^T\mathcal{D}_{\boldsymbol{H}}\boldsymbol{\varphi}(\boldsymbol{\theta}) &= \boldsymbol{W}^T\boldsymbol{W} \otimes \boldsymbol{I}_n, \end{aligned}$$

and

$$\begin{aligned}\mathcal{D}_{\boldsymbol{W}}\boldsymbol{\varphi}(\boldsymbol{\theta})^T\mathcal{D}_{\boldsymbol{H}}\boldsymbol{\varphi}(\boldsymbol{\theta}) &= (\boldsymbol{H}^T\otimes\boldsymbol{I}_m)\boldsymbol{C}_{n,m}(\boldsymbol{W}\otimes\boldsymbol{I}_n)\\ &= (\boldsymbol{H}^T\otimes\boldsymbol{I}_m)(\boldsymbol{I}_n\otimes\boldsymbol{W})\boldsymbol{C}_{n,k}\\ &= (\boldsymbol{H}^T\otimes\boldsymbol{W})\boldsymbol{C}_{n,k}\\ &= (\boldsymbol{I}_k\otimes\boldsymbol{W})(\boldsymbol{H}^T\otimes\boldsymbol{I}_k)\boldsymbol{C}_{n,k}\\ &= (\boldsymbol{I}_k\otimes\boldsymbol{W})\boldsymbol{C}_k(\boldsymbol{I}_k\otimes\boldsymbol{H}^T).\end{aligned}$$

Hence, we can then express the $(m+n)k\times(m+n)k$ Fisher information matrix $\boldsymbol{\Phi}=\sigma^{-2}\boldsymbol{\Psi}$ compactly as follows [15]

$$\begin{aligned}\boldsymbol{\Psi} &= \begin{bmatrix}\boldsymbol{H}^T\boldsymbol{H}\otimes\boldsymbol{I}_m & (\boldsymbol{I}_k\otimes\boldsymbol{W})\boldsymbol{C}_k(\boldsymbol{I}_k\otimes\boldsymbol{H})^T\\ (\boldsymbol{I}_k\otimes\boldsymbol{H})\boldsymbol{C}_k(\boldsymbol{I}_k\otimes\boldsymbol{W})^T & \boldsymbol{W}^T\boldsymbol{W}\otimes\boldsymbol{I}_n\end{bmatrix} \qquad (5.6)\\ &= \begin{bmatrix}\boldsymbol{H}^T\boldsymbol{H}\otimes\boldsymbol{I}_m & 0\\ 0 & \boldsymbol{W}^T\boldsymbol{W}\otimes\boldsymbol{I}_n\end{bmatrix} + \begin{bmatrix}\boldsymbol{I}_k\otimes\boldsymbol{W} & 0\\ 0 & \boldsymbol{I}_k\otimes\boldsymbol{H}\end{bmatrix}\begin{bmatrix}0 & \boldsymbol{C}_k\\ \boldsymbol{C}_k & 0\end{bmatrix}\begin{bmatrix}\boldsymbol{I}_k\otimes\boldsymbol{W} & 0\\ 0 & \boldsymbol{I}_k\otimes\boldsymbol{H}\end{bmatrix}^T.\end{aligned}$$

The FIM for the matrix factorization model (5.6) is rank deficient, as shown in the following proposition.

Proposition 5.2. *If $\boldsymbol{W}$ and $\boldsymbol{H}$ both have full column rank, then the rank of the $(m+n)k\times(m+n)k$ FIM $\boldsymbol{\Phi}$ is at most $(m+n)k-k^2$.*

Proof. This proposition is equivalent to the claim that the linear system $\boldsymbol{\Psi}\boldsymbol{z}=0$ has at least k^2 linearly independent nonzero solutions. Let

$$\boldsymbol{z} = [\ \boldsymbol{z}_1^T\ \boldsymbol{z}_2^T\ ]^T = [\ \mathsf{vec}\,(\boldsymbol{Z}_1)^T\ \mathsf{vec}\,(\boldsymbol{Z}_2)^T\ ]^T,$$

where $\boldsymbol{Z}_1$ is an $m\times k$ matrix, and $\boldsymbol{Z}_2$ is an $n\times k$ matrix. Then

$$\begin{aligned}\boldsymbol{\Psi}\boldsymbol{z} &= \begin{bmatrix}\boldsymbol{H}^T\boldsymbol{H}\otimes\boldsymbol{I}_m\mathsf{vec}\,(\boldsymbol{Z}_1)+(\boldsymbol{I}_k\otimes\boldsymbol{W})\boldsymbol{C}_k(\boldsymbol{I}_k\otimes\boldsymbol{H})^T\mathsf{vec}\,(\boldsymbol{Z}_2)\\ (\boldsymbol{I}_k\otimes\boldsymbol{H})\boldsymbol{C}_k(\boldsymbol{I}_k\otimes\boldsymbol{W})^T\mathsf{vec}\,(\boldsymbol{Z}_1)+\boldsymbol{W}^T\boldsymbol{W}\otimes\boldsymbol{I}_n\mathsf{vec}\,(\boldsymbol{Z}_1)\end{bmatrix}\\ &= \begin{bmatrix}\mathsf{vec}\left(\boldsymbol{Z}_1\boldsymbol{H}^T\boldsymbol{H}+\boldsymbol{W}\boldsymbol{Z}_2^T\boldsymbol{H}\right)\\ \mathsf{vec}\left(\boldsymbol{H}\boldsymbol{Z}_1^T\boldsymbol{W}+\boldsymbol{Z}_2\boldsymbol{W}^T\boldsymbol{W}\right)\end{bmatrix}.\end{aligned}$$

Now let $\boldsymbol{Z}_1=\boldsymbol{w}_\kappa\boldsymbol{e}_l^T$, $\boldsymbol{Z}_2=-\boldsymbol{h}_l\boldsymbol{e}_\kappa^T$, where $\kappa,l=1,2,...,k$, then $\boldsymbol{z}\neq 0$ and

$$\boldsymbol{\Psi}\boldsymbol{z} = \begin{bmatrix}\mathsf{vec}\left(\left(\boldsymbol{w}_\kappa\boldsymbol{h}_l^T-\boldsymbol{w}_\kappa\boldsymbol{h}_l^T\right)\boldsymbol{H}\right)\\ \mathsf{vec}\left(\left(\boldsymbol{h}_l\boldsymbol{w}_\kappa^T-\boldsymbol{h}_l\boldsymbol{w}_\kappa^T\right)\boldsymbol{W}\right)\end{bmatrix} = 0.$$

Thus, we have found k^2 solutions in that form, and indeed they are linearly independent, if $\boldsymbol{W}$ and $\boldsymbol{H}$ both have full column rank. □

5.2.2 Computing the Cramér-Rao Bound

For classical CRB, once we have derived the FIM, the CRB is simply given by the inverse of FIM. As we have argued in Proposition 5.2, the FIM for matrix factorization models is always rank deficient. Nevertheless, pseudo-inverse of the FIM can be used to compute a lowerbound, albeit not necessarily attainable in theory. In terms of identifiability, it is well-known that additional constraints are needed to insure uniqueness of the solution; however, as we have argued, simple constraints like non-negativity can provide identifiability under mild conditions, and in fact does not affect the CRB since it can be represented as inequality constraints. Therefore, we discuss how to efficiently compute the psuedo-inverse of the FIM without modifying it to accommodate any equality constraints.

Without exploiting any structure of the matrix, the usual way to calculate the pseudo-inverse is by using the singular value decomposition (SVD), which entails complexity approximately cubic in the matrix dimension. The FIM for matrix factorization is $(m+n)k \times (m+n)k$, and the complexity of brute-force pseudo-inversion via the SVD is problematic. At first glance, the FIM in (5.6) exhibits good structure: $\boldsymbol{\Phi}$ given in (5.6) is the summation of a non-singular matrix and a low rank term. However, we cannot directly apply the matrix inversion lemma (Woodbury's identity) or the blockwise inversion formula (cf. [137]), simply because they are both singular, as we have argued in Propositions 5.2.

There exist similar results for the pseudo-inverse, e.g., [138, 139], but the formulas are very complicated. In fact, $\boldsymbol{\Phi}$ also has Kronecker structure, and it is appealing to try using the following Property of the Kronecker pruduct [137]

$$(\boldsymbol{A} \otimes \boldsymbol{B})^\dagger = \boldsymbol{A}^\dagger \otimes \boldsymbol{B}^\dagger$$

to greatly reduce the computation complexity. However, the formulas given in [138,139] are so complicated that such structure would be destroyed. Therefore, in this section we seek specialized methods to compute the pseudo-inverse of $\boldsymbol{\Phi}$.

The basic idea of our method is based on the fact that we have not only identified the singularity but also bases for the null space of $\boldsymbol{\Phi}$. In this case, the basis of the null space helps us to calculate the pseudo-inverse by using the techniques for calculating the inverse, as described in the following lemma.

Lemma 5.1. *Let matrix $\boldsymbol{M}$ be symmetric and singular, and the matrix $\boldsymbol{L}$ satisfying* $\text{range}\,\{\boldsymbol{L}\} = \text{null}\,\{\boldsymbol{M}\}$*, then*

$$\boldsymbol{M}^\dagger = (\boldsymbol{M} + \boldsymbol{L}\boldsymbol{L}^T)^{-1} - (\boldsymbol{L}^\dagger)^T\boldsymbol{L}^\dagger. \tag{5.7}$$

Proof. Let the thin eigen-decomposition of $\boldsymbol{M}$ be

$$\boldsymbol{M} = \boldsymbol{U}_M\boldsymbol{\Lambda}_M\boldsymbol{U}_M^T,$$

then the left-hand-side of (5.7) is

$$\boldsymbol{M}^\dagger = \boldsymbol{U}_M\boldsymbol{\Lambda}_M^{-1}\boldsymbol{U}_M^T.$$

Let the thin SVD of $\boldsymbol{L}$ be

$$\boldsymbol{L} = \boldsymbol{U}_L\boldsymbol{\Sigma}_L\boldsymbol{V}_L^T.$$

Since $\text{range}\,\{\boldsymbol{L}\} = \text{null}\,\{\boldsymbol{M}\}$, we have that the matrix $[\ \boldsymbol{U}_M\ \boldsymbol{U}_L\ ]$ is a square unitary matrix, therefore

$$\boldsymbol{M} + \boldsymbol{L}\boldsymbol{L}^T = \begin{bmatrix}\boldsymbol{U}_M & \boldsymbol{U}_L\end{bmatrix}\begin{bmatrix}\boldsymbol{\Lambda}_M & 0\\ 0 & \boldsymbol{\Sigma}_L^2\end{bmatrix}\begin{bmatrix}\boldsymbol{U}_M & \boldsymbol{U}_L\end{bmatrix}^T$$

is invertible. Thus, the right-hand-side of (5.7) is

$$\begin{aligned}&\left(\boldsymbol{M} - \boldsymbol{L}\boldsymbol{L}^T\right)^{-1} - (\boldsymbol{L}^\dagger)^T\boldsymbol{L}^\dagger\\ =\ &\begin{bmatrix}\boldsymbol{U}_M & \boldsymbol{U}_L\end{bmatrix}\begin{bmatrix}\boldsymbol{\Lambda}_M & 0\\ 0 & \boldsymbol{\Sigma}_L^2\end{bmatrix}^{-1}\begin{bmatrix}\boldsymbol{U}_M & \boldsymbol{U}_L\end{bmatrix}^T - \boldsymbol{U}_L\boldsymbol{\Sigma}_L^{-2}\boldsymbol{U}_L^T\\ =\ &\boldsymbol{U}_M\boldsymbol{\Lambda}_M^{-1}\boldsymbol{U}_M^T,\end{aligned}$$

which is obviously equal to the left-hand-side of (5.7). □

In the sequel, we will also invoke the matrix inversion lemma

$$(\boldsymbol{A} + \boldsymbol{B}\boldsymbol{C}\boldsymbol{D})^{-1} = \boldsymbol{A}^{-1} - \boldsymbol{A}^{-1}\boldsymbol{B}(\boldsymbol{C}^{-1} + \boldsymbol{D}\boldsymbol{A}^{-1}\boldsymbol{B})^{-1}\boldsymbol{D}\boldsymbol{A}^{-1},$$

and the block-wise matrix inversion formula

$$\begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{D} & \boldsymbol{C} \end{bmatrix} = \begin{bmatrix} (\boldsymbol{A} - \boldsymbol{B}\boldsymbol{C}^{-1}\boldsymbol{D})^{-1} & \maltese \\ \maltese & (\boldsymbol{C} - \boldsymbol{D}\boldsymbol{A}^{-1}\boldsymbol{B})^{-1} \end{bmatrix}.$$

The expressions for the off-diagonal blocks are omitted ($\maltese$), since we do not need to compute them in our context.

Now we are ready to derive a computationally efficient way of calculating the pseudo-inverse of $\boldsymbol{\Psi}$. Recall that we have fully identified the null space of $\boldsymbol{\Psi}$ in Proposition 5.2, and a basis of its null space are of the form

$$\begin{aligned} \boldsymbol{z} &= \begin{bmatrix} \mathsf{vec}\left(\boldsymbol{w}_\kappa \boldsymbol{e}_l^T\right) \\ -\mathsf{vec}\left(\boldsymbol{h}_l \boldsymbol{e}_\kappa^T\right) \end{bmatrix} = \begin{bmatrix} \boldsymbol{I}_k \otimes \boldsymbol{W} & 0 \\ 0 & \boldsymbol{I}_k \otimes \boldsymbol{H} \end{bmatrix} \begin{bmatrix} \boldsymbol{e}_\kappa \otimes \boldsymbol{e}_l \\ -\boldsymbol{e}_l \otimes \boldsymbol{e}_\kappa \end{bmatrix} \\ &= \begin{bmatrix} \boldsymbol{I}_k \otimes \boldsymbol{W} & 0 \\ 0 & \boldsymbol{I}_k \otimes \boldsymbol{H} \end{bmatrix} \begin{bmatrix} \boldsymbol{I}_{k^2} & 0 \\ 0 & \boldsymbol{C}_k \end{bmatrix} \begin{bmatrix} \boldsymbol{e}_\kappa \otimes \boldsymbol{e}_l \\ -\boldsymbol{e}_\kappa \otimes \boldsymbol{e}_l \end{bmatrix}, \end{aligned}$$

Thus, we can stack all the vectors of this form and define the matrix $\boldsymbol{L}$ as

$$\begin{aligned} \boldsymbol{L} &= \begin{bmatrix} \boldsymbol{I}_k \otimes \boldsymbol{W} & 0 \\ 0 & \boldsymbol{I}_k \otimes \boldsymbol{H} \end{bmatrix} \begin{bmatrix} \boldsymbol{I}_{k^2} & 0 \\ 0 & \boldsymbol{C}_k \end{bmatrix} \begin{bmatrix} \boldsymbol{I}_{k^2} \\ -\boldsymbol{I}_{k^2} \end{bmatrix} \\ &= \begin{bmatrix} \boldsymbol{I}_k \otimes \boldsymbol{W} & 0 \\ 0 & \boldsymbol{I}_k \otimes \boldsymbol{H} \end{bmatrix} \begin{bmatrix} \boldsymbol{I}_{k^2} \\ -\boldsymbol{C}_k \end{bmatrix} = \begin{bmatrix} \boldsymbol{I}_k \otimes \boldsymbol{W} \\ -\left(\boldsymbol{I}_k \otimes \boldsymbol{H}\right)\boldsymbol{C}_k \end{bmatrix}, \end{aligned}$$

whose columns are linearly independent, thus we can write its pseudo-inverse explicitly as

$$\begin{aligned} \boldsymbol{L}^\dagger &= \left(\boldsymbol{L}^T\boldsymbol{L}\right)^{-1}\boldsymbol{L}^T \\ &= \left(\boldsymbol{I}_k \otimes \boldsymbol{W}^T\boldsymbol{W} + \boldsymbol{C}_k\left(\boldsymbol{I}_k \otimes \boldsymbol{H}^T\boldsymbol{H}\right)\boldsymbol{C}_k\right)^{-1} \begin{bmatrix} \boldsymbol{I}_k \otimes \boldsymbol{W} \\ -\left(\boldsymbol{I}_k \otimes \boldsymbol{H}\right)\boldsymbol{C}_k \end{bmatrix} \\ &= \left(\boldsymbol{I}_k \otimes \boldsymbol{W}^T\boldsymbol{W} + \boldsymbol{H}^T\boldsymbol{H} \otimes \boldsymbol{I}_k\right)^{-1} \begin{bmatrix} \boldsymbol{I}_k \otimes \boldsymbol{W} \\ -\left(\boldsymbol{I}_k \otimes \boldsymbol{H}\right)\boldsymbol{C}_k \end{bmatrix}^T, \end{aligned}$$

and we have that $\mathrm{range}\{\boldsymbol{L}\} = \mathrm{null}\{\boldsymbol{\Psi}\}$. Then we can "complete" the range of $\boldsymbol{\Psi}$ via

defining

$$
\begin{aligned}
&\boldsymbol{\Omega} = \boldsymbol{\Psi} + \boldsymbol{L}\boldsymbol{L}^T \\
&= \begin{bmatrix} \boldsymbol{H}^T\boldsymbol{H} \otimes \boldsymbol{I}_m & 0 \\ 0 & \boldsymbol{W}^T\boldsymbol{W} \otimes \boldsymbol{I}_n \end{bmatrix} + \begin{bmatrix} \boldsymbol{I}_k \otimes \boldsymbol{W} & 0 \\ 0 & \boldsymbol{I}_k \otimes \boldsymbol{H} \end{bmatrix} \begin{bmatrix} 0 & \boldsymbol{C}_k \\ \boldsymbol{C}_k & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{I}_k \otimes \boldsymbol{W} & 0 \\ 0 & \boldsymbol{I}_k \otimes \boldsymbol{H} \end{bmatrix}^T \\
&\quad + \begin{bmatrix} \boldsymbol{I}_k \otimes \boldsymbol{W} & 0 \\ 0 & \boldsymbol{I}_k \otimes \boldsymbol{H} \end{bmatrix} \begin{bmatrix} \boldsymbol{I}_{k^2} & -\boldsymbol{C}_k \\ -\boldsymbol{C}_k & \boldsymbol{I}_{k^2} \end{bmatrix} \begin{bmatrix} \boldsymbol{I}_k \otimes \boldsymbol{W} & 0 \\ 0 & \boldsymbol{I}_k \otimes \boldsymbol{H} \end{bmatrix}^T \\
&= \begin{bmatrix} \boldsymbol{H}^T\boldsymbol{H} \otimes \boldsymbol{I}_m + (\boldsymbol{I}_k \otimes \boldsymbol{W})(\boldsymbol{I}_k \otimes \boldsymbol{W})^T & 0 \\ 0 & \boldsymbol{W}^T\boldsymbol{W} \otimes \boldsymbol{I}_n + (\boldsymbol{I}_k \otimes \boldsymbol{H})(\boldsymbol{I}_k \otimes \boldsymbol{H})^T \end{bmatrix}, \\
&= \begin{bmatrix} \boldsymbol{\Omega}_{\boldsymbol{W}} & 0 \\ 0 & \boldsymbol{\Omega}_{\boldsymbol{H}} \end{bmatrix}
\end{aligned}
$$

which is, surprisingly, block diagonal, and each diagonal block can be inverted easily using matrix inversion lemma as follows

$$
\begin{aligned}
\boldsymbol{\Omega}^{-1} &= \begin{bmatrix} \boldsymbol{\Omega}_{\boldsymbol{W}}^{-1} & 0 \\ 0 & \boldsymbol{\Omega}_{\boldsymbol{H}}^{-1} \end{bmatrix}, \\
\boldsymbol{\Omega}_{\boldsymbol{W}}^{-1} &= \left(\boldsymbol{H}^T\boldsymbol{H}\right)^{-1} \otimes \boldsymbol{I}_m - \\
&\qquad \left(\left(\boldsymbol{H}^T\boldsymbol{H}\right)^{-1} \otimes \boldsymbol{W}\right) \left(\boldsymbol{I}_{k^2} + \left(\boldsymbol{H}^T\boldsymbol{H}\right)^{-1} \otimes \boldsymbol{W}^T\boldsymbol{W}\right)^{-1} \left(\left(\boldsymbol{H}^T\boldsymbol{H}\right)^{-1} \otimes \boldsymbol{W}^T\right), \\
\boldsymbol{\Omega}_{\boldsymbol{H}}^{-1} &= \left(\boldsymbol{W}^T\boldsymbol{W}\right)^{-1} \otimes \boldsymbol{I}_n - \\
&\qquad \left(\left(\boldsymbol{W}^T\boldsymbol{W}\right)^{-1} \otimes \boldsymbol{H}\right) \left(\boldsymbol{I}_{k^2} + \left(\boldsymbol{W}^T\boldsymbol{W}\right)^{-1} \otimes \boldsymbol{H}^T\boldsymbol{H}\right)^{-1} \left(\left(\boldsymbol{W}^T\boldsymbol{W}\right)^{-1} \otimes \boldsymbol{H}^T\right),
\end{aligned}
$$

and finally

$$
\boldsymbol{\Psi}^{\dagger} = \begin{bmatrix} \boldsymbol{\Omega}_{\boldsymbol{W}}^{-1} & 0 \\ 0 & \boldsymbol{\Omega}_{\boldsymbol{H}}^{-1} \end{bmatrix} - \begin{bmatrix} \boldsymbol{I}_k \otimes \boldsymbol{W} \\ -\left(\boldsymbol{I}_k \otimes \boldsymbol{H}\right)\boldsymbol{C}_k \end{bmatrix} \left(\boldsymbol{I}_k \otimes \boldsymbol{W}^T\boldsymbol{W} + \boldsymbol{H}^T\boldsymbol{H} \otimes \boldsymbol{I}_k\right)^{-2} \begin{bmatrix} \boldsymbol{I}_k \otimes \boldsymbol{W} \\ -\left(\boldsymbol{I}_k \otimes \boldsymbol{H}\right)\boldsymbol{C}_k \end{bmatrix}^T,
$$

thanks to Lemma 5.1.

In a lot of cases we are only interested in evaluating how small $\|\boldsymbol{W} - \hat{\boldsymbol{W}}\|_F^2$ and $\|\boldsymbol{H} - \hat{\boldsymbol{H}}\|_F^2$ can be, on average. We can then define

$$
\begin{aligned}
\beta_{\boldsymbol{W}} = &\operatorname{trace}\left\{\left(\boldsymbol{H}^T\boldsymbol{H}\right)^{-1} \otimes \boldsymbol{I}_m\right\} \\
&- \operatorname{trace}\left\{\left(\boldsymbol{I}_{k^2} + \left(\boldsymbol{H}^T\boldsymbol{H}\right)^{-1} \otimes \boldsymbol{W}^T\boldsymbol{W}\right)^{-1} \left(\left(\boldsymbol{H}^T\boldsymbol{H}\right)^{-2} \otimes \boldsymbol{W}^T\boldsymbol{W}\right)\right\} \\
&- \operatorname{trace}\left\{\left(\boldsymbol{I}_k \otimes \boldsymbol{W}^T\boldsymbol{W} + \boldsymbol{H}^T\boldsymbol{H} \otimes \boldsymbol{I}_k\right)^{-2} \left(\boldsymbol{I}_k \otimes \boldsymbol{W}^T\boldsymbol{W}\right)\right\},
\end{aligned}
$$

and

$$\begin{aligned}\beta_{\boldsymbol{H}} = &\operatorname{trace}\left\{\left(\boldsymbol{W}^T\boldsymbol{W}\right)^{-1}\otimes\boldsymbol{I}_n\right\}\\ &-\operatorname{trace}\left\{\left(\boldsymbol{I}_{k^2}+\left(\boldsymbol{W}^T\boldsymbol{W}\right)^{-1}\otimes\boldsymbol{H}^T\boldsymbol{H}\right)^{-1}\left(\left(\boldsymbol{W}^T\boldsymbol{W}\right)^{-2}\otimes\boldsymbol{H}^T\boldsymbol{H}\right)\right\}\\ &-\operatorname{trace}\left\{\left(\boldsymbol{I}_k\otimes\boldsymbol{W}^T\boldsymbol{W}+\boldsymbol{H}^T\boldsymbol{H}\otimes\boldsymbol{I}_k\right)^{-2}\left(\boldsymbol{H}^T\boldsymbol{H}\otimes\boldsymbol{I}_k\right)\right\},\end{aligned}$$

and they are the Cramér-Rao bound for the matrix factorization model, i.e.,

$$\begin{aligned}\mathsf{E}\left\{\|\boldsymbol{W}-\hat{\boldsymbol{W}}\|_F^2\right\} &\geq \sigma^2\beta_{\boldsymbol{W}},\\ \mathsf{E}\left\{\|\boldsymbol{H}-\hat{\boldsymbol{H}}\|_F^2\right\} &\geq \sigma^2\beta_{\boldsymbol{H}},\end{aligned}$$

for any unbiased estimators $\hat{\boldsymbol{W}}$ and $\hat{\boldsymbol{H}}$.

5.3 Cramér-Rao Bound for CP Factorization Models

The CRB for the CP factorization model exhibits a lot of similarities to the one for the matrix factorization model, but also a fair number of differences, thus it deserves to be derived from scratch and study its properties separately.

Consider the N-way tensor generated as

$$\underline{\boldsymbol{Y}} = [\![\boldsymbol{H}_d]\!]_{d=1}^N + \underline{\boldsymbol{N}},$$

where $\boldsymbol{H}_d$ is $n_d \times k$ and the elements of $\underline{\boldsymbol{N}}$ are drawn from an i.i.d. Gaussian distribution with zero mean and variance σ^2. Then the log-likelihood of $\underline{\boldsymbol{Y}}$ parameterized by $\boldsymbol{H}_1, ..., \boldsymbol{H}_N$ is

$$\begin{aligned}\log p(\underline{\boldsymbol{Y}}; \boldsymbol{H}_1, ..., \boldsymbol{H}_N) &= -\frac{1}{\sigma^2}\left\|\mathsf{vec}\left(\underline{\boldsymbol{Y}}\right) - (\boldsymbol{H}_N \odot ... \odot \boldsymbol{H}_1)\mathbf{1}\right\|^2\\ &= -\frac{1}{\sigma^2}\|\mathsf{vec}\left(\underline{\boldsymbol{Y}}\right) - \boldsymbol{\varphi}(\boldsymbol{\theta})\|^2,\end{aligned}$$

where the unknown parameters we want to estimate, $\boldsymbol{H}_1, ..., \boldsymbol{H}_N$, are stacked into one single long vector of size $(n_1 + ... + n_N)k$

$$\boldsymbol{\theta} = \left[\begin{array}{cccc}\mathsf{vec}\left(\boldsymbol{H}_1\right)^T & ... & \mathsf{vec}\left(\boldsymbol{H}_N\right)^T\end{array}\right]^T,$$

and the nonlinear function

$$\begin{aligned}\boldsymbol{\varphi}(\boldsymbol{\theta}) &= (\boldsymbol{H}_N \odot \cdots \odot \boldsymbol{H}_1)\mathbf{1} \\ &= \boldsymbol{C}_{n_{d-1}\ldots n_1, n_N \ldots n_d}(\boldsymbol{H}_{d-1} \odot \cdots \odot \boldsymbol{H}_1 \odot \boldsymbol{H}_N \odot \cdots \odot \boldsymbol{H}_d)\mathbf{1} \\ &= \boldsymbol{C}_{n_{d-1}\ldots n_1, n_N \ldots n_d}\mathsf{vec}\left(\boldsymbol{H}_d(\boldsymbol{H}_{d-1} \odot \cdots \odot \boldsymbol{H}_1 \odot \boldsymbol{H}_N \odot \cdots \odot \boldsymbol{H}_{d+1})^T\right) \\ &= \boldsymbol{C}_{n_{d-1}\ldots n_1, n_N \ldots n_d}\left((\boldsymbol{H}_{d-1} \odot \cdots \odot \boldsymbol{H}_1 \odot \boldsymbol{H}_N \odot \cdots \odot \boldsymbol{H}_{d+1}) \otimes \boldsymbol{I}_{n_d}\right)\mathsf{vec}\left(\boldsymbol{H}_d\right).\end{aligned} \tag{5.8}$$

5.3.1 The Fisher Information Matrix

Invoking Proposition 5.1, the FIM for the CP model under Gaussian noise has the form

$$\boldsymbol{\Phi} = \frac{1}{\sigma^2}\boldsymbol{\Psi} = \frac{1}{\sigma^2}\mathcal{D}_{\boldsymbol{\theta}}\boldsymbol{\varphi}(\boldsymbol{\theta})^T\mathcal{D}_{\boldsymbol{\theta}}\boldsymbol{\varphi}(\boldsymbol{\theta}).$$

The Jacobian matrix of $\boldsymbol{\varphi}(\boldsymbol{\theta})$ can be partitioned into N blocks

$$\mathcal{D}_{\boldsymbol{\theta}}\boldsymbol{\varphi}(\boldsymbol{\theta}) = \begin{bmatrix}\mathcal{D}_{\boldsymbol{H}_1}\boldsymbol{\varphi}(\boldsymbol{\theta}) & \cdots & \mathcal{D}_{\boldsymbol{H}_N}\boldsymbol{\varphi}(\boldsymbol{\theta})\end{bmatrix},$$

and (5.8) simply gives us

$$\mathcal{D}_{\boldsymbol{H}_d}\boldsymbol{\varphi}(\boldsymbol{\theta}) = \boldsymbol{C}_{n_{d-1}\ldots n_1, n_N \ldots n_d}\left((\boldsymbol{H}_{d-1} \odot \cdots \odot \boldsymbol{H}_1 \odot \boldsymbol{H}_N \odot \cdots \odot \boldsymbol{H}_{d+1}) \otimes \boldsymbol{I}_{n_d}\right)$$

Using the properties of the commutation matrices, we have that

$$\mathcal{D}_{\boldsymbol{H}_d}\boldsymbol{\varphi}(\boldsymbol{\theta})^T\mathcal{D}_{\boldsymbol{H}_d}\boldsymbol{\varphi}(\boldsymbol{\theta}) = (\boldsymbol{H}_{d-1}^T\boldsymbol{H}_{d-1} \circledast \cdots \circledast \boldsymbol{H}_1^T\boldsymbol{H}_1 \circledast \boldsymbol{H}_N^T\boldsymbol{H}_N \circledast \cdots \circledast \boldsymbol{H}_{d+1}^T\boldsymbol{H}_{d+1}) \otimes \boldsymbol{I}_{n_d}$$

and as for the off-diagonal blocks $\mathcal{D}_{\boldsymbol{H}_c}\boldsymbol{\varphi}(\boldsymbol{\theta})^T\mathcal{D}_{\boldsymbol{H}_d}\boldsymbol{\varphi}(\boldsymbol{\theta})$, consider multiplying this matrix with $\mathsf{vec}\left(\tilde{\boldsymbol{H}}_d\right)$ where $\tilde{\boldsymbol{H}}_d$ is a $n_d \times k$ matrix,

$$
\begin{aligned}
&\mathcal{D}_{\boldsymbol{H}_c}\boldsymbol{\varphi}(\boldsymbol{\theta})^T\mathcal{D}_{\boldsymbol{H}_d}\boldsymbol{\varphi}(\boldsymbol{\theta})\mathsf{vec}\left(\tilde{\boldsymbol{H}}_d\right)\\
=&\mathcal{D}_{\boldsymbol{H}_c}\boldsymbol{\varphi}(\boldsymbol{\theta})^T\boldsymbol{C}_{n_{d-1}\ldots n_1,n_N\ldots n_d}\mathsf{vec}\left(\tilde{\boldsymbol{H}}_d\left(\boldsymbol{H}_{d-1}\odot\cdots\odot\boldsymbol{H}_1\odot\boldsymbol{H}_N\odot\cdots\odot\boldsymbol{H}_{d+1}\right)^T\right)\\
=&\mathcal{D}_{\boldsymbol{H}_c}\boldsymbol{\varphi}(\boldsymbol{\theta})^T\boldsymbol{C}_{n_{d-1}\ldots n_1,n_N\ldots n_d}(\boldsymbol{H}_{d-1}\odot\cdots\odot\boldsymbol{H}_1\odot\boldsymbol{H}_N\odot\cdots\odot\boldsymbol{H}_{d+1}\odot\tilde{\boldsymbol{H}}_d)\mathbf{1}\\
=&\mathcal{D}_{\boldsymbol{H}_c}\boldsymbol{\varphi}(\boldsymbol{\theta})^T(\boldsymbol{H}_N\odot\cdots\odot\boldsymbol{H}_{d+1}\odot\tilde{\boldsymbol{H}}_d\odot\boldsymbol{H}_{d-1}\odot\cdots\odot\boldsymbol{H}_1)\mathbf{1}\\
=&\left((\boldsymbol{H}_{c-1}\odot\cdots\odot\boldsymbol{H}_1\odot\boldsymbol{H}_N\odot\cdots\odot\boldsymbol{H}_{c+1})^T\otimes\boldsymbol{I}_{n_c}\right)\boldsymbol{C}_{n_N\ldots n_c,n_{c-1}\ldots n_1}\\
&\qquad(\boldsymbol{H}_{d-1}\odot\cdots\odot\boldsymbol{H}_1\odot\boldsymbol{H}_N\odot\cdots\odot\boldsymbol{H}_{d+1}\odot\tilde{\boldsymbol{H}}_d)\mathbf{1}\\
=&\left(\left(\mathop{\circledast}_{\substack{j=1\\j\neq d,c}}^{N}\boldsymbol{H}_j^T\boldsymbol{H}_j\circledast\boldsymbol{H}_d^T\tilde{\boldsymbol{H}}_d\right)\odot\boldsymbol{H}_c\right)\mathbf{1}\\
=&\mathsf{vec}\left(\boldsymbol{H}_c\left(\mathop{\circledast}_{\substack{j=1\\j\neq d,c}}^{N}\boldsymbol{H}_j^T\boldsymbol{H}_j\circledast\boldsymbol{H}_d^T\tilde{\boldsymbol{H}}_d\right)^T\right)\\
=&(\boldsymbol{I}_k\otimes\boldsymbol{H}_c)\mathrm{diag}\left\{\mathop{\circledast}_{\substack{j=1\\j\neq d,c}}^{N}\boldsymbol{H}_j^T\boldsymbol{H}_j\right\}\mathsf{vec}\left(\boldsymbol{H}_d^T\tilde{\boldsymbol{H}}_d\right)\\
=&(\boldsymbol{I}_k\otimes\boldsymbol{H}_c)\mathrm{diag}\left\{\mathop{\circledast}_{\substack{j=1\\j\neq d,c}}^{N}\boldsymbol{H}_j^T\boldsymbol{H}_j\right\}\boldsymbol{C}_{k,k}(\boldsymbol{I}_k\otimes\boldsymbol{H}_d)\mathsf{vec}\left(\tilde{\boldsymbol{H}}_c\right).
\end{aligned}
$$

This holds for all possible $\tilde{\boldsymbol{H}}\in\mathbb{R}^{n_d\times k}$, implying

$$
\mathcal{D}_{\boldsymbol{H}_c}\boldsymbol{\varphi}(\boldsymbol{\theta})^T\mathcal{D}_{\boldsymbol{H}_d}\boldsymbol{\varphi}(\boldsymbol{\theta})=(\boldsymbol{I}_k\otimes\boldsymbol{H}_c)\mathrm{diag}\left\{\mathop{\circledast}_{\substack{j=1\\j\neq d,c}}^{N}\boldsymbol{H}_j^T\boldsymbol{H}_j\right\}\boldsymbol{C}_{k,k}(\boldsymbol{I}_k\otimes\boldsymbol{H}_d)
$$

We can then express the $(n_1+\ldots+n_N)k\times(n_1+\ldots+n_N)k$ Fisher information matrix $\boldsymbol{\Phi}=\sigma^{-2}\boldsymbol{\Psi}$ compactly as the following block form

$$
\boldsymbol{\Psi}=\begin{bmatrix}\boldsymbol{\Psi}_{1,1}&\cdots&\boldsymbol{\Psi}_{1,N}\\\vdots&\ddots&\vdots\\\boldsymbol{\Psi}_{N,1}&\cdots&\boldsymbol{\Psi}_{N,N}\end{bmatrix},\tag{5.9}
$$

where

$$\boldsymbol{\Psi}_{d,c} = \begin{cases} \boldsymbol{\Gamma}_{d,d} \otimes \boldsymbol{I}_{n_d}, & d = c, \\ (\boldsymbol{I}_k \otimes \boldsymbol{H}_d)\, \boldsymbol{C}_k \mathrm{diag}\left\{\mathsf{vec}\left(\boldsymbol{\Gamma}_{d,c}\right)\right\} (\boldsymbol{I}_k \otimes \boldsymbol{H}_c)^T, & d \neq c, \end{cases} \tag{5.10}$$

and

$$\boldsymbol{\Gamma}_{d,c} = \mathop{\circledast}_{\substack{j=1 \\ j \neq d,c}}^{N} \boldsymbol{H}_j^T \boldsymbol{H}_j. \tag{5.11}$$

Alternatively, we can also write it in the form of "block diagonal plus low rank" as follows

$$\boldsymbol{\Psi} = \boldsymbol{\Delta} + \boldsymbol{\Upsilon} \boldsymbol{K} \boldsymbol{\Upsilon}^T,$$

where $\boldsymbol{\Delta}$ and $\boldsymbol{\Upsilon}$ are both block diagonal

$$\boldsymbol{\Delta} = \begin{bmatrix} \boldsymbol{\Gamma}_{1,1} \otimes \boldsymbol{I}_{n_1} & 0 & \cdots & 0 \\ 0 & \boldsymbol{\Gamma}_{2,2} \otimes \boldsymbol{I}_{n_2} & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & \boldsymbol{\Gamma}_{N,N} \otimes \boldsymbol{I}_{n_N} \end{bmatrix},$$

$$\boldsymbol{\Upsilon} = \begin{bmatrix} \boldsymbol{I}_k \otimes \boldsymbol{H}_1 & 0 & \cdots & 0 \\ 0 & \boldsymbol{I}_k \otimes \boldsymbol{H}_2 & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & \boldsymbol{I}_k \otimes \boldsymbol{H}_N \end{bmatrix},$$

and the $Nk^2 \times Nk^2$ matrix $\boldsymbol{K}$ is partitioned into $N \times N$ blocks each of size $k^2 \times k^2$, and the d, c-th block equals to

$$\boldsymbol{K}_{d,c} = \begin{cases} 0, & d = c, \\ \boldsymbol{C}_k \mathrm{diag}\left\{\mathsf{vec}\left(\boldsymbol{\Gamma}_{d,c}\right)\right\}, & d \neq c. \end{cases}$$

Formulae for the Jacobian matrix and FIM have appeared in [140–144], but the derivation is not as clear and straight-forward as the one given here.

Remark. The FIM for the CP model indeed looks very similar to the FIM for the matrix factorization model. In fact, for $N = 2$, if we overload the definition of $\boldsymbol{\Gamma}_{d,c}$ to be $\boldsymbol{\Gamma}_{1,2} = \boldsymbol{\Gamma}_{2,1} = \mathbf{1}_{k \times k}$, then the FIM for CP defined in (5.9)-(5.11) for $N = 2$ becomes

exactly equal to the FIM for matrix factorization defined in (5.6). Similar to the matrix factorization case, the FIM is also rank deficient. However, the null space result for the matrix factorization case does not simply generalize to the CP case, as will be seen in the following proposition.

Proposition 5.3. *If $\boldsymbol{H}_1, ..., \boldsymbol{H}_N$ all have full column rank, then the rank of the $(n_1 + ... + n_N)k \times (n_1 + ... + n_N)k$ FIM $\boldsymbol{\Phi}$ is at most $(n_1 + ... + n_N)k - (N-1)k$.*

Proof. Again, it suffices to find $(N-1)k$ linearly independent solutions to the linear system $\boldsymbol{\Psi z} = 0$. Consider a vector $\boldsymbol{z}$ of the following form

$$\boldsymbol{z} = [\ \underbrace{0\ ...\ 0}_{(n_1+...+n_{c-1})k}\ \ \mathsf{vec}\left(\boldsymbol{H}_c\boldsymbol{D}\right)^T\ \ \underbrace{0\ ...\ 0}_{(n_{c+1}+...+n_N)k}\ \ ]^T,$$

where $\boldsymbol{D}$ is an arbitrary diagonal matrix, then

$$\boldsymbol{\Psi z} = \begin{bmatrix} \boldsymbol{\Psi}_{1,c}\mathsf{vec}\left(\boldsymbol{H}_c\boldsymbol{D}\right) \\ \boldsymbol{\Psi}_{2,c}\mathsf{vec}\left(\boldsymbol{H}_c\boldsymbol{D}\right) \\ \vdots \\ \boldsymbol{\Psi}_{N,c}\mathsf{vec}\left(\boldsymbol{H}_c\boldsymbol{D}\right) \end{bmatrix},$$

where

$$\boldsymbol{\Psi}_{c,c}\mathsf{vec}\left(\boldsymbol{H}_c\boldsymbol{D}\right) = \mathsf{vec}\left(\boldsymbol{H}_c\boldsymbol{D}\boldsymbol{\Gamma}_{c,c}\right),$$

and

$$\begin{aligned} \boldsymbol{\Psi}_{d,c}\mathsf{vec}\left(\boldsymbol{H}_c\boldsymbol{D}\right) &= \left(\boldsymbol{I}_k \otimes \boldsymbol{H}_d\right)\boldsymbol{C}_k\mathrm{diag}\left\{\mathsf{vec}\left(\boldsymbol{\Gamma}_{d,c}\right)\right\}\left(\boldsymbol{I}_k \otimes \boldsymbol{H}_c\right)^T\mathsf{vec}\left(\boldsymbol{H}_c\boldsymbol{D}\right) \\ &= \left(\boldsymbol{I}_k \otimes \boldsymbol{H}_d\right)\boldsymbol{C}_k\mathrm{diag}\left\{\mathsf{vec}\left(\boldsymbol{\Gamma}_{d,c}\right)\right\}\mathsf{vec}\left(\boldsymbol{H}_c^T\boldsymbol{H}_c\boldsymbol{D}\right) \\ &= \left(\boldsymbol{I}_k \otimes \boldsymbol{H}_d\right)\boldsymbol{C}_k\mathsf{vec}\left(\boldsymbol{\Gamma}_{d,d}\boldsymbol{D}\right) \\ &= \left(\boldsymbol{I}_k \otimes \boldsymbol{H}_d\right)\mathsf{vec}\left(\boldsymbol{D}\boldsymbol{\Gamma}_{d,d}\right) \\ &= \mathsf{vec}\left(\boldsymbol{H}_d\boldsymbol{D}\boldsymbol{\Gamma}_{d,d}\right), \end{aligned}$$

for $d \neq c$. Notice that at the third step, $\mathrm{diag}\left\{\mathsf{vec}\left(\boldsymbol{\Gamma}_{d,c}\right)\right\}\mathsf{vec}\left(\boldsymbol{H}_c^T\boldsymbol{H}_c\boldsymbol{D}\right) = \mathsf{vec}\left(\boldsymbol{\Gamma}_{d,d}\boldsymbol{D}\right)$ if and only if $\boldsymbol{D}$ is a diagonal matrix. As we can see, for $\boldsymbol{z}$ of this form, the result of $\boldsymbol{\Psi z}$ is independent of c.

Next, consider $\boldsymbol{z}$ to be the difference of two vectors of the aforementioned form, the non-zero block of one of them being the first block

$$\boldsymbol{z} = [\ \mathsf{vec}\,(\boldsymbol{H}_1\boldsymbol{D})^T \quad \underbrace{0\ ...\ 0}_{(n_2+...+n_{c-1})k} \quad -\mathsf{vec}\,(\boldsymbol{H}_c\boldsymbol{D})^T \quad \underbrace{0\ ...\ 0}_{(n_{c+1}+...+n_N)k} \quad]^T,$$

then

$$\boldsymbol{\Psi}\boldsymbol{z} = \begin{bmatrix} \boldsymbol{\Psi}_{1,1}\mathsf{vec}\,(\boldsymbol{H}_1\boldsymbol{D}) \\ \boldsymbol{\Psi}_{2,1}\mathsf{vec}\,(\boldsymbol{H}_1\boldsymbol{D}) \\ \vdots \\ \boldsymbol{\Psi}_{N,1}\mathsf{vec}\,(\boldsymbol{H}_1\boldsymbol{D}) \end{bmatrix} - \begin{bmatrix} \boldsymbol{\Psi}_{1,c}\mathsf{vec}\,(\boldsymbol{H}_c\boldsymbol{D}) \\ \boldsymbol{\Psi}_{2,c}\mathsf{vec}\,(\boldsymbol{H}_c\boldsymbol{D}) \\ \vdots \\ \boldsymbol{\Psi}_{N,c}\mathsf{vec}\,(\boldsymbol{H}_c\boldsymbol{D}) \end{bmatrix} = 0.$$

Fixing c, a $k \times k$ diagonal matrix $\boldsymbol{D}$ has k degrees of freedom, and c can be chosen from $2, ..., N$, so we have found in total $(N-1)k$ linearly independent solutions to the linear system $\boldsymbol{\Psi}\boldsymbol{z} = 0$.

Notice that we can also make the d-th block and c-block of $\boldsymbol{z}$ being $\mathsf{vec}\,(\boldsymbol{H}_d\boldsymbol{D})$ and $-\mathsf{vec}\,(\boldsymbol{H}_c\boldsymbol{D})$, but it is equal to the first and d-th, minus first and c-th, so this does not introduce additional dimension to the null space of $\boldsymbol{\Psi}$. □

Remark. In terms of the dimension of the null space of $\boldsymbol{\Phi}$, the FIM for the CP model does not generalize to the matrix case, since the rank deficiency is $(N-1)k$, whereas the rank deficiency of the FIM for the MF model is k^2, which is not equal to $(2-1)k$. In fact, let us pick a basis for the span of the diagonal matrices to be $\{\ \boldsymbol{e}_1\boldsymbol{e}_1^T, ..., \boldsymbol{e}_k\boldsymbol{e}_k^T\ \}$, then it becomes apparent that the null space is closely related to the inherent scaling ambiguity in the CP model (meaning if the l-th column of $\boldsymbol{H}_1$ and the l-th column of $\boldsymbol{H}_c$ move in the opposite direction, it does not affect the CRB), whereas in the two factor case, the $\boldsymbol{D}$ matrix is not restricted to be diagonal, which seems related to the fact that for matrix factorization $\boldsymbol{Y} = \boldsymbol{W}\boldsymbol{H}^T$, we can put a more general non-singular matrix in between $\boldsymbol{Y} = \boldsymbol{W}\boldsymbol{A}\boldsymbol{A}^{-1}\boldsymbol{H}^T$. Nevertheless, these are trivial ambiguities within these factor analysis models, and it is not obvious how, for example, simple non-negativity constraints can lead to essentially unique solutions using our proposed *sufficiently scattered* condition.

5.3.2 Computing the Cramér-Rao Bound

To compute the pseudo-inverse of the FIM we derived in (5.9)-(5.11) to obtain the CRB for the CP factorization model, we use the similar idea used in CRB for the MF case, which is by invoking Lemma 5.1, and the fact that we have identified the null space of $\boldsymbol{\Phi}$ in Proposition 5.3.

First, define the matrix $\boldsymbol{L}$ whose columns span the null space of $\boldsymbol{\Psi}$

$$\boldsymbol{L} = \begin{bmatrix} \mathsf{vec}\left(\boldsymbol{H}_1\boldsymbol{e}_1\boldsymbol{e}_1^T\right) & \cdots & \mathsf{vec}\left(\boldsymbol{H}_1\boldsymbol{e}_N\boldsymbol{e}_N^T\right) & \cdots & \mathsf{vec}\left(\boldsymbol{H}_1\boldsymbol{e}_1\boldsymbol{e}_1^T\right) & \cdots & \mathsf{vec}\left(\boldsymbol{H}_1\boldsymbol{e}_N\boldsymbol{e}_N^T\right) \\ -\mathsf{vec}\left(\boldsymbol{H}_2\boldsymbol{e}_1\boldsymbol{e}_1^T\right) & \cdots & -\mathsf{vec}\left(\boldsymbol{H}_2\boldsymbol{e}_N\boldsymbol{e}_N^T\right) & 0 & \cdots & & 0 \\ \vdots & & \vdots & \ddots & & \ddots & \vdots \\ 0 & \cdots & & 0 & -\mathsf{vec}\left(\boldsymbol{H}_N\boldsymbol{e}_1\boldsymbol{e}_1^T\right) & \cdots & -\mathsf{vec}\left(\boldsymbol{H}_N\boldsymbol{e}_N\boldsymbol{e}_N^T\right) \end{bmatrix}$$

$$= \begin{bmatrix} \boldsymbol{I}_k \otimes \boldsymbol{H}_1 & 0 & \cdots & 0 \\ 0 & \boldsymbol{I}_k \otimes \boldsymbol{H}_2 & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & \boldsymbol{I}_k \otimes \boldsymbol{H}_N \end{bmatrix} \begin{bmatrix} \boldsymbol{I}_k \odot \boldsymbol{I}_k & \cdots & \boldsymbol{I}_k \odot \boldsymbol{I}_k \\ -\boldsymbol{I}_k \odot \boldsymbol{I}_k & & 0 \\ & \ddots & \vdots \\ 0 & \cdots & -\boldsymbol{I}_k \odot \boldsymbol{I}_k \end{bmatrix} = \boldsymbol{\Upsilon}\boldsymbol{E},$$

where $\boldsymbol{E}$ is $Nk^2 \times (N-1)k$, partitioned into $N \times (N-1)$ blocks, with d, c-th block defined as

$$\boldsymbol{E}_{d,c} = \begin{cases} \boldsymbol{I}_k \odot \boldsymbol{I}_k, & d = 1, \\ -\boldsymbol{I}_k \odot \boldsymbol{I}_k, & d = c+1, \\ 0, & \text{otherwise.} \end{cases}$$

Since $\boldsymbol{L}$ has full column rank, its pseudo-inverse is

$$\begin{aligned} \boldsymbol{L}^\dagger &= \left(\boldsymbol{L}^T\boldsymbol{L}\right)^{-1}\boldsymbol{L}^T \\ &= \left(\boldsymbol{E}^T\boldsymbol{\Upsilon}^T\boldsymbol{\Upsilon}\boldsymbol{E}\right)^{-1}\boldsymbol{E}^T\boldsymbol{\Upsilon}^T, \end{aligned}$$

where the matrix we want to invert has the form

$$\boldsymbol{E}^T\boldsymbol{\Upsilon}^T\boldsymbol{\Upsilon}\boldsymbol{E} = \begin{bmatrix} \boldsymbol{I}_k \circledast (\boldsymbol{H}_1^T\boldsymbol{H}_1 + \boldsymbol{H}_2^T\boldsymbol{H}_2) & \boldsymbol{I}_k \circledast \boldsymbol{H}_1^T\boldsymbol{H}_1 & \cdots & \boldsymbol{I}_k \circledast \boldsymbol{H}_1^T\boldsymbol{H}_1 \\ \boldsymbol{I}_k \circledast \boldsymbol{H}_1^T\boldsymbol{H}_1 & \boldsymbol{I}_k \circledast (\boldsymbol{H}_1^T\boldsymbol{H}_1 + \boldsymbol{H}_3^T\boldsymbol{H}_3) & & \boldsymbol{I}_k \circledast \boldsymbol{H}_1^T\boldsymbol{H}_1 \\ \vdots & & \ddots & \vdots \\ \boldsymbol{I}_k \circledast \boldsymbol{H}_1^T\boldsymbol{H}_1 & \boldsymbol{I}_k \circledast \boldsymbol{H}_1^T\boldsymbol{H}_1 & \cdots & \boldsymbol{I}_k \circledast (\boldsymbol{H}_1^T\boldsymbol{H}_1 + \boldsymbol{H}_N^T\boldsymbol{H}_N) \end{bmatrix}$$

$$= \begin{bmatrix} \boldsymbol{I}_k \circledast \boldsymbol{H}_2^T\boldsymbol{H}_2 & 0 & \cdots & 0 \\ & \boldsymbol{I}_k \circledast \boldsymbol{H}_3^T\boldsymbol{H}_3 & & \vdots \\ \vdots & & \ddots & \\ 0 & \cdots & & \boldsymbol{I}_k \circledast \boldsymbol{H}_N^T\boldsymbol{H}_N \end{bmatrix}$$

$$+ \begin{bmatrix} \boldsymbol{I}_k \circledast \boldsymbol{H}_1^T\boldsymbol{H}_1 & \boldsymbol{I}_k \circledast \boldsymbol{H}_1^T\boldsymbol{H}_1 & \cdots & \boldsymbol{I}_k \circledast \boldsymbol{H}_1^T\boldsymbol{H}_1 \\ \boldsymbol{I}_k \circledast \boldsymbol{H}_1^T\boldsymbol{H}_1 & \boldsymbol{I}_k \circledast \boldsymbol{H}_1^T\boldsymbol{H}_1 & & \\ \vdots & & \ddots & \vdots \\ \boldsymbol{I}_k \circledast \boldsymbol{H}_1^T\boldsymbol{H}_1 & \cdots & & \boldsymbol{I}_k \circledast \boldsymbol{H}_1^T\boldsymbol{H}_1 \end{bmatrix},$$

which is "diagonal plus low rank", thus can be inverted efficiently.

Next, we define $\boldsymbol{\Omega}$ by completing the range space of $\boldsymbol{\Psi}$

$$\begin{aligned} \boldsymbol{\Omega} &= \boldsymbol{\Psi} + \boldsymbol{L}\boldsymbol{L}^T \\ &= \boldsymbol{\Delta} + \boldsymbol{\Upsilon}\boldsymbol{K}\boldsymbol{\Upsilon}^T + \boldsymbol{\Upsilon}\boldsymbol{E}\boldsymbol{E}^T\boldsymbol{\Upsilon}^T \\ &= \boldsymbol{\Delta} + \boldsymbol{\Upsilon}\left(\boldsymbol{K} + \boldsymbol{E}\boldsymbol{E}^T\right)\boldsymbol{\Upsilon}^T. \end{aligned}$$

In the CP factorization case, the "completed" matrix $\boldsymbol{\Omega}$ does not have the nice block diagonal structure as we did in the matrix case. Such a hope is arguably impossible, as we notice that each $k^2 \times k^2$ block not on the diagonal is full rank, and different, whereas the rank of $\boldsymbol{E}\boldsymbol{E}^T$ has rank $(N-1)k$, therefore we cannot construct a matrix $\boldsymbol{E}$ such that each of its off-diagonal block have rank k^2, unless $k < N$. Nevertheless, if $\boldsymbol{K} + \boldsymbol{E}\boldsymbol{E}^T$ is invertible, applying matrix inversion lemma on $\boldsymbol{\Omega}$ leads to

$$\boldsymbol{\Omega}^{-1} = \boldsymbol{\Delta}^{-1} - \boldsymbol{\Delta}^{-1}\boldsymbol{\Upsilon}\left(\left(\boldsymbol{K} + \boldsymbol{E}\boldsymbol{E}^T\right)^{-1} + \boldsymbol{\Upsilon}\boldsymbol{\Delta}^{-1}\boldsymbol{\Upsilon}\right)^{-1}\boldsymbol{\Upsilon}^T\boldsymbol{\Delta}^{-1}.$$

Notice that $\boldsymbol{\Delta}$ is block diagonal, and each of its diagonal blocks is a Kronecker product, thus computing $\boldsymbol{\Delta}^{-1}$ only requires inverting N number of $k \times k$ matrices. The most expensive step is to compute $\left(\boldsymbol{K} + \boldsymbol{E}\boldsymbol{E}^T\right)^{-1}$ and $\left(\left(\boldsymbol{K} + \boldsymbol{E}\boldsymbol{E}^T\right)^{-1} + \boldsymbol{\Upsilon}\boldsymbol{\Delta}^{-1}\boldsymbol{\Upsilon}\right)^{-1}$, both of size $Nk^2 \times Nk^2$. However, it is still a huge improvement, considering the size of $\boldsymbol{\Psi}$ is $(n_1 + ... + n_N)k \times (n_1 + ... + n_N)k$, if $Nk < n_1 + ... + n_N$. Otherwise, the CP factorization

is not considered "low-rank", thus directly (pseudo-)invert the FIM is not a bad idea, nonetheless.

Finally, by subtracting $(\boldsymbol{L}^\dagger)^T\boldsymbol{L}^\dagger$ from $\boldsymbol{\Omega}^{-1}$ we obtain

$$\begin{aligned}\boldsymbol{\Psi}^\dagger &= \boldsymbol{\Omega}^{-1} - (\boldsymbol{L}^\dagger)^T\boldsymbol{L}^\dagger \\ &= \boldsymbol{\Delta}^{-1} - \boldsymbol{\Delta}^{-1}\boldsymbol{\Upsilon}\left(\left(\boldsymbol{K}+\boldsymbol{E}\boldsymbol{E}^T\right)^{-1} + \boldsymbol{\Upsilon}\boldsymbol{\Delta}^{-1}\boldsymbol{\Upsilon}\right)^{-1}\boldsymbol{\Upsilon}^T\boldsymbol{\Delta}^{-1} \\ &\qquad - \boldsymbol{\Upsilon}\boldsymbol{E}\left(\boldsymbol{E}^T\boldsymbol{\Upsilon}^T\boldsymbol{\Upsilon}\boldsymbol{E}\right)^{-2}\boldsymbol{E}^T\boldsymbol{\Upsilon}^T,\end{aligned}$$

and the CRB is simply

$$\boldsymbol{\Phi}^\dagger = \sigma^2\boldsymbol{\Psi}^\dagger.$$

5.4 Putting NMF to the Test

In this section we illustrate how several NMF algorithms performs compared to our derived CRB for matrix factorization models. Notice that the data we synthetically generated were corrupted by additive i.i.d. Gaussian noise, so using Euclidian distance as the objective actually gives us the ML estimate. This is why algorithms that use other divergence functions as the objective were not considered here. The algorithms tested are

MU Multiplicative Update proposed by Lee and Seung [102]

ALS Alternating Least Squares proposed by Berry *et al.* [145]

PG Projected Gradient proposed by Lin [146][1]

HALS Fast Hierarchical Alternating Least Squares proposed by Cichocki and Phan [92, Algorithm 2]

BPP Alternating Nonnegative Least Squares using Block Principle Pivoting proposed by Kim and Park [99][2]

The entries of $\boldsymbol{W}$ and $\boldsymbol{H}$ were generated such that a certain proportion of them are randomly set to 0, and the rest are drawn from an i.i.d. exponential distribution. Then the columns of $\boldsymbol{W}$ are scaled to sum up to 1.

[1] Matlab code downloaded from `http://www.csie.ntu.edu.tw/~cjlin/nmf/index.html`
[2] Matlab code downloaded from `http://www.cc.gatech.edu/~hpark/nmfsoftware.php`

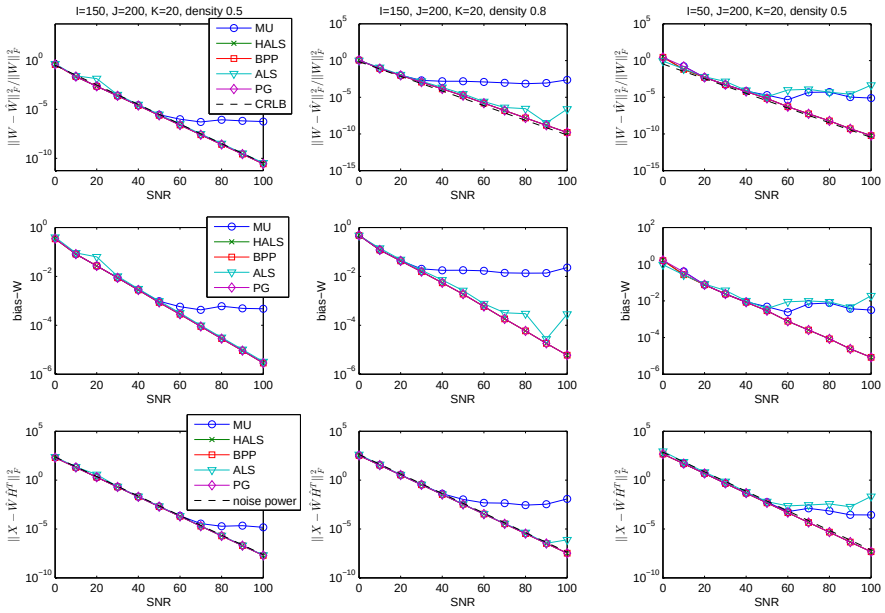


Figure 5.1: The first row shows the normalized squared error for $\boldsymbol{W}$ using various asymmetric NMF algorithms versus the CRB; similarly, the second row shows the (aggregate) bias for $\boldsymbol{W}$, and the third row shows the fitting error.

Three tests were conducted and illustrated in Figs. 5.1 and 5.2 for $\boldsymbol{W}$ and $\boldsymbol{H}$ respectively—low-rank and sparse latent factors on the left, low-rank but moderately dense in the middle, and an unbalanced case (n much larger than m) where the rank is not small compared to the smaller outer dimension, with density set relatively small to ensure identifiability. The top row of Figs. 5.1 and 5.2 shows the normalized squared error for each algorithm benchmarked by the CRLB, the second row shows the (aggregate) bias of $\boldsymbol{W}$, and similarly for $\boldsymbol{H}$, and the bottom row of Fig. 5.1 shows the fitting error for each algorithm.

As we can see from the second row of Figs. 5.1 and 5.2, the biases are generally small and approach zero with increasing SNR, indicating that we can use the CRB to approximately bound performance. In all three cases, HALS, BPP and PG were able to provide a good estimate with mean square error close to the CRLB, under all SNRs tested. On the other hand, MU and ALS are not guaranteed to work well even under

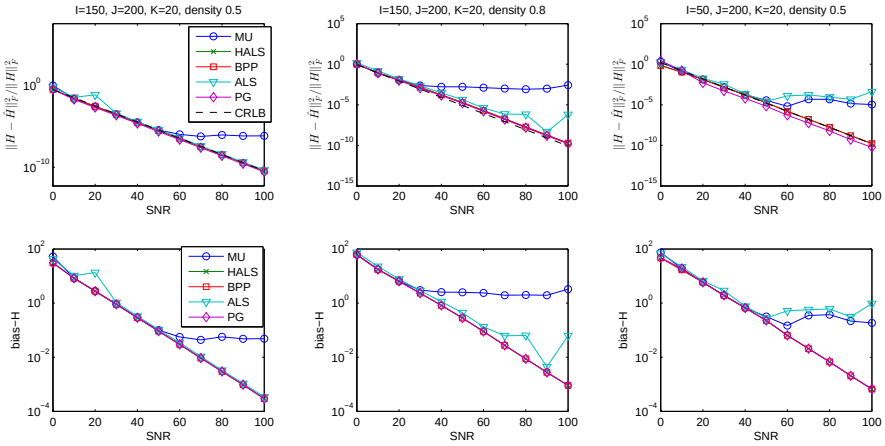


Figure 5.2: The first row shows the normalized squared error for $\boldsymbol{H}$ using various asymmetric NMF algorithms versus the CRB; similarly, the second row shows the (aggregate) bias for $\boldsymbol{H}$.

very high SNR. All methods separate the variables into blocks, and HALS, BPP and PG aim to find the conditionally optimal point before moving to the next block, whereas the updates of MU and ALS cannot guarantee this.

5.5 Concluding Remarks

NMF and other constrained matrix and tensor factorization models entail singular FIMs as well as constraints and ambiguities that must be dealt with in the computation of the pertinent CRB. We learned how to tackle those, and used the results to benchmark and develop insights on what can be expected from some of the best available algorithms. For symmetric NMF, the CRLB can be approached using the Procrustes rotation algorithm [3] in the high SNR regime, or α/β-SNMF in low SNR cases. For asymmetric NMF, the best-performing algorithms were able to give results with mean squared error close to the CRLB. In both cases, approaching the CRLB is possible when the signal rank is small and the latent factors are not dense; i.e., when there is a small number of latent components whose loadings contain sufficiently many zeros. This is quite remarkable given that the CRLB with a singular FIM is generally unattainable. There may be

room for improvement in cases involving moderate (SNR, rank, density).

Beyond NMF, the approach and techniques we learned can be used to facilitate analogous derivations for related factor analysis problems. For example, the FIMs provided here can be applied to more general bilinear matrix factorizations, e.g., using other types of constraints on $\boldsymbol{H}$. The FIM will remain the same, but the $\boldsymbol{U}$ matrix will be different. Also, we can exploit a basis of the nullspace of the FIM to reduce the complexity of computing its pseudo-inverse, and this idea is more broadly applicable to other bilinear matrix factorizations. The results can also be extended towards, e.g., non-negative tensor factorization.

Chapter 6

Symmetric Non-negative Matrix Factorization

In this chapter we study an interesting variant of non-negative matrix factorization, namely by constraining the two matrix factors to be the same, hence we call it symmetric NMF. The factorization is denoted as $\boldsymbol{Y} \approx \boldsymbol{H}\boldsymbol{H}^T$, where $\boldsymbol{Y}$ is a $n \times n$ symmetric and possibly positive semidefinite matrix, and we further constrain the $n \times k$ matrix $\boldsymbol{H} \geq 0$ element-wise. Notice that in the exact case $\boldsymbol{Y} = \boldsymbol{H}\boldsymbol{H}^T$, $\boldsymbol{H}$ in symmetric NMF is a square-root factor of the symmetric positive semi-definite matrix $\boldsymbol{Y}$ that is also element-wise non-negative. Thus symmetric NMF is an element-wise non-negative square-root factorization of positive semidefinite matrices. In the mathematical programming field, such matrices are called *completely positive matrices* [147]. It has recently been shown that checking whether a positive semidefinite matrix is completely positive is also NP-hard [148].

Symmetric NMF is relatively less studied in literature, but is it gaining more and more popular as people have found applications of it in soft-clustering [149] and topic modeling [41]. In this chapter we give a complete study of this problem, following a similar path we have already gone through in this dissertation—starting by studying the uniqueness condition for this model, the Cramér-Rao bound is then derived for performance analysis, then an efficient algorithm is described which represents the state-of-the-art for symmetric NMF, and conclude with numerical experiments to support our

claims.

6.1 Uniqueness of Symmetric NMF

Recall that in the asymmetric case, if $\boldsymbol{Y} = \boldsymbol{W}\boldsymbol{H}^T = \tilde{\boldsymbol{W}}\tilde{\boldsymbol{H}}^T$ and $\text{rank}\{\boldsymbol{Y}\} = k$, then there exists a non-singular matrix $\boldsymbol{A}$ such that $\tilde{\boldsymbol{W}} = \boldsymbol{W}\boldsymbol{A}^{-T}$ and $\tilde{\boldsymbol{H}} = \boldsymbol{H}\boldsymbol{A}$. In the symmetric case, by further constraining $\boldsymbol{W} = \boldsymbol{H}$, $\tilde{\boldsymbol{W}} = \tilde{\boldsymbol{H}}$, we have that $\boldsymbol{A}^{-T} = \boldsymbol{A}$, i.e., $\boldsymbol{A}$ is a $k \times k$ unitary matrix. A trivial choice of $\boldsymbol{A}$ would be a permutation matrix, which is an unavoidable ambiguity without side information. Thus, we define (essential) uniqueness of symmetric NMF as follows.

Definition 6.1 (uniqueness of symmetric NMF)**.** *The symmetric NMF of $\boldsymbol{Y} = \boldsymbol{H}\boldsymbol{H}^T$ is said to be (essentially) unique if $\boldsymbol{Y} = \tilde{\boldsymbol{H}}\tilde{\boldsymbol{H}}^T$ implies $\tilde{\boldsymbol{H}} = \boldsymbol{H}\boldsymbol{P}$, where $\boldsymbol{P}$ is a permutation matrix.*

Notice that in the symmetric case we do not have scaling ambiguity now. Using this definition, a necessary condition for the essential uniqueness of symmetric NMF can be derived, similar to the asymmetric case.

Theorem 6.1 (necessary condition)**.** *Let* $\text{supp}\{\boldsymbol{x}\}$ *denote the index set of the non-zero entries (support) of a vector $\boldsymbol{x}$, i.e.,* $\text{supp}\{\boldsymbol{x}\} = \{i \mid x_i \neq 0\}$*. If the symmetric NMF $\boldsymbol{Y} = \boldsymbol{H}\boldsymbol{H}^T$ is unique, then there do not exist $\mu, \nu \in \{1, \cdots, k\}, \mu \neq \nu$, such that* $\text{supp}\{\boldsymbol{h}_\mu\} \subseteq \text{supp}\{\boldsymbol{h}_\nu\}$.

Proof. Suppose $\text{supp}\{\boldsymbol{h}_\mu\} \subseteq \text{supp}\{\boldsymbol{h}_\nu\}$, then there exists a positive scalar α such that

$$\boldsymbol{h}_\nu - \alpha \boldsymbol{h}_\mu \geq 0$$

Let $\boldsymbol{A}$ be a Givens rotation matrix [150] defined as

$$\boldsymbol{A} = [\ \boldsymbol{e}_\mu\ \boldsymbol{e}_\nu\ ] \begin{bmatrix} c & -s \\ s & c \end{bmatrix} [\ \boldsymbol{e}_\mu\ \boldsymbol{e}_\nu\ ]^T + \sum_{\substack{l=1 \\ l \neq \mu,\nu}}^{k} \boldsymbol{e}_l \boldsymbol{e}_l^T,$$

where

$$c = \frac{1}{\sqrt{1+\alpha^2}}, s = \frac{\alpha}{\sqrt{1+\alpha^2}}$$

Since $\boldsymbol{A}$ is unitary but not a permutation matrix, and $\boldsymbol{H}\boldsymbol{A} \geq 0$, according to Definition 6.1, the symmetric NMF $\boldsymbol{Y} = \boldsymbol{H}\boldsymbol{H}^T$ is not unique. □

Using the geometric interpretation of NMF in § 2.2, we can give a similar geometric interpretation of symmetric NMF.

Lemma 6.1. *If* $\operatorname{rank}\{\boldsymbol{Y}\} = k$*, the symmetric NMF* $\boldsymbol{Y} = \boldsymbol{H}\boldsymbol{H}^T$ *is unique if and only if the non-negative orthant is the only self-dual simplicial cone* $\mathcal{A}$ *with* k *extreme rays that satisfies* $\operatorname{cone}\left\{\boldsymbol{H}^T\right\} \subseteq \mathcal{A} = \mathcal{A}^*$.

Proof. By Definition 6.1, if the symmetric NMF $\boldsymbol{Y} = \boldsymbol{H}\boldsymbol{H}^T$ is essentially unique, then for any unitary matrix $\boldsymbol{A}$, $\boldsymbol{H}\boldsymbol{A} \geq 0$ implies that $\boldsymbol{A}$ is a permutation matrix. Now $\boldsymbol{H}\boldsymbol{A} \geq 0$ implies $\operatorname{cone}\left\{\boldsymbol{H}^T\right\} \subseteq \operatorname{cone}\{\boldsymbol{A}\}^* = \operatorname{cone}\{\boldsymbol{A}\}$, and $\boldsymbol{A}$ being a permutation matrix means $\operatorname{cone}\{\boldsymbol{A}\} = \mathbb{R}^k_+$. Thus, this is simply a geometric way to describe Definition 6.1. □

For a sufficient condition, the similar *sufficiently scattered* condition can be used, which we repeat here.

Definition (sufficiently scattered)**.** *A non-negative matrix* $\boldsymbol{X}$ *is sufficiently scattered if*

1. $\operatorname{cone}\left\{\boldsymbol{X}^T\right\} \supseteq \mathcal{C}$,
2. $\operatorname{cone}\left\{\boldsymbol{X}^T\right\}^* \cap \operatorname{bd}\{\mathcal{C}^*\} = \{\lambda \boldsymbol{e}_\kappa \mid \lambda \geq 0, \kappa = 1, ..., k.\}$,

where $\mathcal{C}$ *and* $\mathcal{C}^*$ *are*

$$\mathcal{C} = \{\boldsymbol{x} | \boldsymbol{x}^T\mathbf{1} \geq \sqrt{k-1}\|\boldsymbol{x}\|_2\}, \quad \mathcal{C}^* = \{\boldsymbol{x} | \boldsymbol{x}^T\mathbf{1} \geq \|\boldsymbol{x}\|_2\}.$$

Theorem 6.2 (sufficient condition)**.** *The symmetric NMF* $\boldsymbol{Y} = \boldsymbol{H}\boldsymbol{H}^T$ *is essentially unique if* $\boldsymbol{H}$ *is sufficiently scattered.*

Proof. As we have proved in Theorem 2.4, if $\operatorname{cone}\left\{\boldsymbol{X}^T\right\} \supseteq \mathcal{C}$, then $\forall \kappa = 1, \ldots, k$, $\boldsymbol{e}_\kappa$ is an extreme ray of $\operatorname{cone}\left\{\boldsymbol{X}^T\right\}^*$. According to Lemma 6.1, if the symmetric NMF is unique, then we cannot rotate $\mathbb{R}^K_+$ to $\mathcal{A}$ such that $\operatorname{cone}\left\{\boldsymbol{H}^T\right\} \subseteq \mathcal{A}$. Since $\operatorname{cone}\left\{\boldsymbol{H}^T\right\} \supseteq \mathcal{C}$, and $\mathcal{A}$ is self-dual, then we have $\mathcal{A} \subseteq \operatorname{cone}\left\{\boldsymbol{H}^T\right\}^* \subseteq \mathcal{C}^*$. As shown in Lemma 2.1, any rotated version of $\mathbb{R}^K_+$ that is a subset of $\mathcal{C}^*$ satisfies that its extreme rays lie on the boundary of $\mathcal{C}^*$. However, none of the extreme rays of $\operatorname{cone}\left\{\boldsymbol{H}^T\right\}$ except $\mathbf{e}_k$'s lie on the boundary of $\mathcal{C}^*$, therefore $\mathcal{A}$ can only be the non-negative orthant $\mathbb{R}^K_+$ itself. As a result, under this condition the symmetric NMF $\boldsymbol{Y} = \boldsymbol{H}\boldsymbol{H}^T$ is unique. □

Table 6.1: Maximum reconstruction error for symmetric NMF

density	max $\|\hat{\boldsymbol{H}} - \boldsymbol{H}\|_F$
0.5	2.57×10^{-13}
0.6	4.20×10^{-13}
0.7	6.42×10^{-13}
0.8	3.29×10^{-12}

Following our discussion on the sufficiently scatteredcondition in § 2.3, even though this condition is NP hart to check in the worst case, in practice it is usually satisfied as long as the columns of $\boldsymbol{H}$ contain at least $k-1$ zeros. To illustrate this, we randomly generate a 200×30 non-negative matrix $\boldsymbol{H}$, with a certain proportion of randomly selected entries set to zero, and the non-zero entries drawn from an i.i.d. exponential distribution. The columns of $\boldsymbol{H}$ are ordered so that

$$\sum_{j=1}^{n} h_{j,1} > \sum_{j=1}^{n} h_{j,2} > \cdots > \sum_{j=1}^{n} h_{j,k}.$$

Then we form the low rank complete positive matrix $\boldsymbol{Y} = \boldsymbol{H}\boldsymbol{H}^T$, and apply a symmetric NMF algorithm to it to get an estimate $\hat{\boldsymbol{H}}$. The columns of $\hat{\boldsymbol{H}}$ are then ordered analogously, to fix the permutation ambiguity. For density (the proportion of non-zero entries in $\boldsymbol{H}$) varying from 0.5 to 0.8, in which case the matrix $\boldsymbol{H}$ we randomly generated satisfies that $\mathbf{e}_k$'s are extreme rays of cone $\left\{\boldsymbol{H}^T\right\}^*$ with high probability, this procedure is repeated 100 times, and the maximum reconstruction error $\|\hat{\boldsymbol{H}} - \boldsymbol{H}\|_F$ is given in Table 2.1. This indicates that over the 400 Monte-Carlo tests we tried, symmetric NMF successfully recovered the latent factors in each and every case. The algorithm we used for symmetric NMF is the Procrustes-based algorithm proposed in [3].

If it is not realistic to assume the latent factors are sufficiently scattered, we propose the following VolMin symmetric NMF that works for general symmetric non-negative matrices, while still ensuring uniqueness of the latent factors. Consider the following "Tucker" matrix factorization

$$\boldsymbol{Y} = \boldsymbol{H}\boldsymbol{G}\boldsymbol{H}^T,$$

where $\boldsymbol{H} \geq 0$, we can seek for a core matrix $\boldsymbol{G}$ that has the smallest "volume"

$$\begin{aligned} \underset{\boldsymbol{H},\boldsymbol{G}}{\text{minimize}} \quad & |\det \boldsymbol{G}| \\ \text{subject to} \quad & \boldsymbol{Y} = \boldsymbol{H}\boldsymbol{G}\boldsymbol{H}^T, \boldsymbol{H} \geq 0, \boldsymbol{H}^T\mathbf{1} = \mathbf{1}. \end{aligned} \tag{6.1}$$

Notice that constraining the columns of $\boldsymbol{H}$ to sum up to one is without loss of generality, since any column scaling of $\boldsymbol{H}$ can be absorbed into the core matrix $\boldsymbol{G}$.

Theorem 6.3. *Denote an optimal solution of (6.1) to be $\tilde{\boldsymbol{H}}$, $\tilde{\boldsymbol{G}}$. If the true latent factor $\boldsymbol{H}$ is sufficiently scattered (cf. Definition 6.1) and column sum to one, then there exists a permutation matrix $\boldsymbol{P}$ such that $\tilde{\boldsymbol{H}}\boldsymbol{P} = \boldsymbol{H}$. In other words, $\boldsymbol{H}$ and $\boldsymbol{G}$ can be identified up to permutation via solving (6.1).*

Proof. By contradiction. Suppose $\tilde{\boldsymbol{G}}$ and $\tilde{\boldsymbol{H}}$ is an optimal solution of (6.1), but $\tilde{\boldsymbol{H}}$ is not a column permutation of $\boldsymbol{H}$, the true latent factor. Since $\boldsymbol{G}$ and $\boldsymbol{H}$ are clearly feasible for (3.1), this means that $|\det \tilde{\boldsymbol{G}}| \leq |\det \boldsymbol{G}|$.

Since the columns of $\boldsymbol{H}$ and $\tilde{\boldsymbol{H}}$ span the same subspace, there is a $k \times k$ matrix $\boldsymbol{A}$ such that

$$\tilde{\boldsymbol{H}} = \boldsymbol{H}\boldsymbol{A},$$

and since $\boldsymbol{Y} = \boldsymbol{H}\boldsymbol{G}\boldsymbol{H}^T = \tilde{\boldsymbol{H}}\tilde{\boldsymbol{G}}\tilde{\boldsymbol{H}}^T$, this means

$$\tilde{\boldsymbol{G}} = \boldsymbol{A}^{-1}\boldsymbol{G}\boldsymbol{A}^{-T}.$$

Since $\boldsymbol{H}$ is sufficiently scattered, according to Lemma 3.2, and our first assumption that $\boldsymbol{A}$ is not a permutation matrix, we have

$$|\det \boldsymbol{A}| < 1.$$

However, the optimal objective of (6.1) obtained by $\tilde{\boldsymbol{G}}, \tilde{\boldsymbol{H}}$ is

$$\begin{aligned} |\det \tilde{\boldsymbol{G}}| &= |\det \boldsymbol{A}^{-1}\boldsymbol{G}\boldsymbol{A}^{-T}| \\ &= |\det \boldsymbol{A}|^{-2} |\det \boldsymbol{G}| \\ &> |\det \boldsymbol{G}|, \end{aligned}$$

which contradicts to our first assumption that $\tilde{\boldsymbol{G}}, \tilde{\boldsymbol{H}}$ is an optimal solution for (6.1). Therefore, $\tilde{\boldsymbol{H}}$ must be a column permutation of $\boldsymbol{H}$. □

6.2 A Procrustes-based Algorithm

Suppose there exists an exact symmetric NMF of $\boldsymbol{Y}$ with $k = \operatorname{rank}\{\boldsymbol{Y}\}$ components. Then $\boldsymbol{Y}$ is symmetric positive semi-definite; consider its reduced eigen-decomposition

$$\boldsymbol{Y} = \boldsymbol{U}_k \boldsymbol{\Lambda}_k \boldsymbol{U}_k^T,$$

where $\boldsymbol{U}_k$ is $n \times k$ orthogonal and $\boldsymbol{\Lambda}_k$ is $k \times k$ diagonal. Define

$$\boldsymbol{B} = \boldsymbol{U}_k \boldsymbol{\Lambda}_k^{1/2}.$$

Since

$$\boldsymbol{Y} = \boldsymbol{B}\boldsymbol{B}^T = \boldsymbol{H}\boldsymbol{H}^T,$$

where both $\boldsymbol{B}$ and $\boldsymbol{H}$ are $n \times k$, there exists a unitary matrix $\boldsymbol{Q}$ such that

$$\boldsymbol{B}\boldsymbol{Q} = \boldsymbol{H}.$$

Therefore, after obtaining $\boldsymbol{B}$ via eigen-analysis, we can formulate the recovery of $\boldsymbol{H}$ as follows:

$$\begin{aligned} \underset{\boldsymbol{H},\boldsymbol{Q}}{\text{minimize}} \quad & \|\boldsymbol{H} - \boldsymbol{B}\boldsymbol{Q}\|_F^2 \\ \text{subject to} \quad & \boldsymbol{H} \geq 0, \boldsymbol{Q}^T\boldsymbol{Q} = \boldsymbol{Q}\boldsymbol{Q}^T = \boldsymbol{I}. \end{aligned} \tag{6.2}$$

The constraint $\boldsymbol{Q}^T\boldsymbol{Q} = \boldsymbol{Q}\boldsymbol{Q}^T = \boldsymbol{I}$ is not convex with respect to $\boldsymbol{Q}$, suggesting that (6.2) is a hard problem. We propose updating $\boldsymbol{H}$ and $\boldsymbol{Q}$ in an alternating fashion. The updating rule for $\boldsymbol{H}$ is extremely simple: since $\boldsymbol{H}$ is non-negative, we simply set

$$\boldsymbol{H} \leftarrow \max(0, \boldsymbol{B}\boldsymbol{Q}) \tag{6.3}$$

When updating $\boldsymbol{Q}$, the solution is given by the Procrustes projection [151], i.e.

$$\boldsymbol{Q} \leftarrow \boldsymbol{V}\boldsymbol{U}^T \tag{6.4}$$

where $\boldsymbol{U}$ and $\boldsymbol{V}$ are unitary matrices given by the singular value decomposition of $\boldsymbol{H}^T\boldsymbol{B}$

$$\boldsymbol{H}^T\boldsymbol{B} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T \tag{6.5}$$

This simple algorithm is summarized in Alg. 6.1. The Karush-Kuhn-Tucker conditions for problem (6.2), after some calibrations, are

$$\begin{cases} \boldsymbol{Q}^T\boldsymbol{Q} = \boldsymbol{Q}\boldsymbol{Q}^T = \boldsymbol{I}, \\ \boldsymbol{H} \geq 0, \\ \boldsymbol{H} - \boldsymbol{B}\boldsymbol{Q} \geq 0, \\ \boldsymbol{H} \circledast (\boldsymbol{H} - \boldsymbol{B}\boldsymbol{Q}) = 0. \end{cases} \tag{6.6}$$

For Alg. 6.1, the first three of (6.6) are satisfied throughout the iterations of Alg. 6.1, and the fourth one is satisfied only after the $\boldsymbol{H}$-update. We therefore measure the norm of $\boldsymbol{H} \circledast (\boldsymbol{H} - \boldsymbol{B}\boldsymbol{Q})$ after the $\boldsymbol{Q}$-update and use it as our termination criterion. This is preferable to checking successive differences of the cost in (6.2), because it avoids early termination during *swamps*—intervals during which the progress in terms of the cost function is slow.

Algorithm 6.1: Proposed algorithm for symmetric NMF

$\boldsymbol{Y} = \boldsymbol{U}_k\boldsymbol{\Lambda}_k\boldsymbol{U}_k^T$; `// thin eigen-decomposition`

$\boldsymbol{B} \leftarrow \boldsymbol{U}_k\boldsymbol{\Lambda}_k^{1/2}$, $\boldsymbol{Q} \leftarrow \boldsymbol{I}$;

repeat

  $\boldsymbol{H} \leftarrow \max(0, \boldsymbol{B}\boldsymbol{Q})$;

  $\boldsymbol{H}^T\boldsymbol{B} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$; `// full SVD`

  $\boldsymbol{Q} \leftarrow \boldsymbol{V}\boldsymbol{U}^T$;

until $\|\boldsymbol{H} \circledast (\boldsymbol{H} - \boldsymbol{B}\boldsymbol{Q})\|_F^2 <$ tolerance;

Proposition 6.1. *The value of the objective function in (6.2) is monotonically non-increasing during iterations of the algorithm given in Alg. 6.1, since each update step is conditionally optimal for $\boldsymbol{H}$ given $\boldsymbol{Q}$ or vice-versa. Furthermore, every limit point generated by the algorithm shown in Alg. 6.1 is a KKT point of problem (6.2).*

Proof. Since Alg. 6.1 is alternating over two blocks of variables, it falls into the framework of *maximum block improvement* (MBI) method [79]. Convergence of MBI requires that every constraint set is a compact set, which directly holds for the set of real unitary matrices (i.e., the orthogonal group) [152] with respect to $\boldsymbol{Q}$. The explicit constraint

set for $\boldsymbol{H}$ is the non-negative orthant, which is not compact because it is unbounded. However, because $\boldsymbol{Q}$ is unitary, we can put a redundant constraint $\|\boldsymbol{H}\|_F^2 \leq \|\boldsymbol{B}\|_F^2$, because

$$\begin{aligned}\|\boldsymbol{H}\|_F^2 &= \|\max(0, \boldsymbol{BQ})\|_F^2 \\ &\leq \|\boldsymbol{BQ}\|_F^2 \\ &= \|\boldsymbol{B}\|_F^2.\end{aligned}$$

Thus, the constraint set for $\boldsymbol{H}$ is essentially compact. According to [79, Theorem 3.1 and Corollary 3.2], every limit point of Alg. 6.1 is a KKT point of problem (6.2). □

In terms of per-iteration complexity, the matrix multiplications $\boldsymbol{BQ}$ and $\boldsymbol{H}^T\boldsymbol{B}$ both require $O(nk^2)$ flops, whereas the SVD performed on the relatively small-sized $k \times k$ matrix $\boldsymbol{H}^T\boldsymbol{B}$ requires $O(k^3)$ flops [150, pp. 254]. If we assume $n \gg k$, which is typically the case in practice, then the $O(nk^2)$ term dominates, which results in a $O(nk^2)$ per-iteration complexity. The computation of k dominant eigenvalues and eigenvectors in the very first step entails complexity $O(n^2k)$, but considering the fact that this is done only once, its $O(n^2k)$ complexity is amortized over many iterations of symmetric NMF.

Remark. Besides the per-iteration complexity we just discussed, another step that consists of the overall complexity of Algorithm 6.1 is computing the eigen-decomposition in line 1. Despite the fact that it has in general $O(n^3)$ complexity, there are many algorithms that can reduce the complexity down to $O(\mathsf{nnz}(\boldsymbol{Y})k)$ for computing k principal components of $\boldsymbol{Y}$, for example, the famous Lanczos algorithm [150]. Moreover, there are many tricks we can use to avoid increasing $\mathsf{nnz}(\boldsymbol{Y})$. For example, if what we are given is a data matrix $\boldsymbol{X}$, and the matrix $\boldsymbol{Y}$ we want to factor is actually constructed as $\boldsymbol{Y} = \boldsymbol{X}^T\boldsymbol{X}$, then to apply Algorithm 6.1, it is not necessary to explicitly calculate $\boldsymbol{Y}$ and introduce excessive amount of nonzeros into $\boldsymbol{Y}$, but rather directly perform SVD on the original sparse data $\boldsymbol{X}$ to obtain the left singular vectors and their corresponding singular values. Even if $\boldsymbol{X}$ is not explicitly sparse, there are ways that tries to avoid filling-in non-zeros and pertain a "sparse matrix-vector multiplication" complexity, e.g., [153].

In practice, we may encounter cases where $k \neq \operatorname{rank}\{\boldsymbol{Y}\}$. For $k < \operatorname{rank}\{\boldsymbol{Y}\}$, we are trying to find a good non-negative low rank approximation of $\boldsymbol{Y}$, and we can simply take

the first k dominant eigen-components, then apply the same updating rules afterwards. For $k > \operatorname{rank}\{\boldsymbol{Y}\}$, we need to modify the basic algorithm; the modified version can be found in the appendix.

6.3 Cramér-Rao Bound

Consider the $n \times n$ symmetric matrix generated as

$$\boldsymbol{Y} = \boldsymbol{H}\boldsymbol{H}^T + \boldsymbol{N},$$

where $\boldsymbol{H}$ is $n \times k$, and the elements of $\boldsymbol{N}$ are drawn from an i.i.d. Gaussian distribution with zero-mean and variance σ^2. Then the log-likelihood of $\boldsymbol{y}$ parameterized by $\boldsymbol{H}$ is

$$L(\boldsymbol{Y}; \boldsymbol{H}) = -\frac{1}{2\sigma^2} \sum_{i=1}^{n} \sum_{j=1}^{n} \left(\sum_{l=1}^{k} j_{il} h_{jl} - y_{ij} \right)^2.$$

In order to define the corresponding Fisher information matrix, we stack the unknown parameter $\boldsymbol{H}$ into a long vector $\boldsymbol{\theta} = \mathsf{vec}\,(\boldsymbol{H})$. Taking the partial derivative over one element $h_{\imath\kappa}$, we get

$$\frac{\partial L}{\partial h_{\imath\kappa}} = \frac{1}{\sigma^2} \sum_{i=1}^{n} \left(\sum_{l=1}^{k} h_{il} h_{\imath l} - y_{i\imath} \right) h_{i\kappa} + \frac{1}{\sigma^2} \sum_{j=1}^{n} \left(\sum_{l=1}^{k} h_{\imath l} h_{jl} - y_{\imath j} \right) h_{j\kappa}.$$

It is easy to check that the regularity condition $\mathsf{E}\left\{ \frac{\partial L}{\partial h_{\imath\kappa}} \right\} = 0$ is indeed satisfied for any $h_{\imath\kappa}$, since the support of the distribution is independent of the parameter $\boldsymbol{H}$ that we are trying to estimate [125, pp. 67]. The covariance between the scores are given by

$$\mathsf{E}\left\{ \frac{\partial L}{\partial h_{\imath\kappa}} \frac{\partial L}{\partial h_{jl}} \right\} = \frac{2}{\sigma^2} \Big(\delta_{\imath,\jmath} \sum_{i=1}^{n} h_{i\kappa} h_{il} + h_{\jmath\kappa} h_{\imath l} \Big),$$

where $\delta_{\imath,\jmath}$ is the Kronecker delta. We can then define the $nk \times nk$ Fisher information matrix $\boldsymbol{\Phi} = 2\sigma^{-2}\boldsymbol{\Psi}$ compactly as

$$\boldsymbol{\Psi} = \boldsymbol{H}^T\boldsymbol{H} \otimes \boldsymbol{I}_n + (\boldsymbol{I}_k \otimes \boldsymbol{H})\boldsymbol{C}_k(\boldsymbol{I}_k \otimes \boldsymbol{H})^T, \tag{6.7}$$

where $\boldsymbol{C}_k$ is the same $k^2 \times k^2$ commutation matrix we have been using in the FIM for (asymmetric) matrix and CP factorization models.

Once again, the FIM for symmetric matrix factorization is rank deficient, as shown in the following proposition.

Proposition 6.2. *If $\boldsymbol{H}$ has full column rank, then the rank of the $nk \times nk$ Fisher information matrix $\boldsymbol{\Phi}$ is at most $nk - k(k-1)/2$.*

Proof. This proposition is equivalent to the claim that the linear system $\boldsymbol{\Psi z} = 0$ has at least $k(k-1)/2$ linearly independent nonzero solutions. Let $\boldsymbol{z} = \mathsf{vec}\,(\boldsymbol{Z})$, where $\boldsymbol{Z}$ is a $n \times k$ matrix, then

$$\begin{aligned}
\boldsymbol{\Psi z} &= (\boldsymbol{H}^T\boldsymbol{H} \otimes \boldsymbol{I}_n)\mathsf{vec}\,(\boldsymbol{Z}) + (\boldsymbol{I}_k \otimes \boldsymbol{H})\boldsymbol{C}_k(\boldsymbol{I}_k \otimes \boldsymbol{H})^T\mathsf{vec}\,(\boldsymbol{Z}) \\
&= \mathsf{vec}\left(\boldsymbol{Z}\boldsymbol{H}^T\boldsymbol{Z}\right) + (\boldsymbol{I}_k \otimes \boldsymbol{H})\boldsymbol{C}_k\mathsf{vec}\left(\boldsymbol{H}^T\boldsymbol{Z}\right) \\
&= \mathsf{vec}\left(\boldsymbol{Z}\boldsymbol{H}^T\boldsymbol{H}\right) + (\boldsymbol{I}_k \otimes \boldsymbol{H})\mathsf{vec}\left(\boldsymbol{Z}^T\boldsymbol{H}\right) \\
&= \mathsf{vec}\left(\boldsymbol{Z}\boldsymbol{H}^T\boldsymbol{H}\right) + \mathsf{vec}\left(\boldsymbol{H}\boldsymbol{Z}\boldsymbol{H}^T\right).
\end{aligned}$$

Now let $\boldsymbol{Z} = \boldsymbol{h}_\kappa\boldsymbol{e}_l^T - \boldsymbol{h}_l\boldsymbol{e}_\kappa^T$, where $\kappa, l = 1, ..., k$, and $\kappa \neq l$, then $\boldsymbol{Z} \neq 0$ and

$$\boldsymbol{\Psi z} = \mathsf{vec}\left(\left(\boldsymbol{h}_\kappa\boldsymbol{h}_l^T - \boldsymbol{h}_l\boldsymbol{h}_\kappa^T + \boldsymbol{h}_l\boldsymbol{h}_\kappa^T - \boldsymbol{h}_\kappa\boldsymbol{h}_l^T\right)\boldsymbol{H}\right) = 0.$$

Thus, we have found $\binom{k}{2} = k(k-1)/2$ solutions in that form, and indeed these solutions are linearly independent, if the columns of $\boldsymbol{H}$ are all linearly independent to each other. □

Now that we have identified the null space of $\boldsymbol{\Phi}$, we can use that to calculate the pseudo-inverse of $\boldsymbol{\Phi}$ more efficiently. First, define the $nk \times k(k-1)/2$ matrix $\boldsymbol{L}$ whose columns span the null space of $\boldsymbol{\Phi}$

$$\boldsymbol{L} = (\boldsymbol{I}_k \otimes \boldsymbol{H})\boldsymbol{R},$$

where $\boldsymbol{R}$ is a $k^2 \times k(k-1)/2$ matrix defined as

$$\boldsymbol{R} = [\ \boldsymbol{r}_{1,2}\ \boldsymbol{r}_{1,3}\ ...\ \boldsymbol{r}_{1,k}\ \boldsymbol{r}_{2,3}\ ...\ \boldsymbol{r}_{k-1,k}\ ],$$

and

$$\boldsymbol{r}_{i,j} = \mathsf{vec}\left(\boldsymbol{e}_i\boldsymbol{e}_j^T - \boldsymbol{e}_j\boldsymbol{e}_i^T\right), \quad i = 1, ..., k-1, \quad j = i+1, ..., k.$$

The matrix $\boldsymbol{L}$ has full column rank, therefore its pseudo-inverse is given by

$$\begin{aligned}
\boldsymbol{L}^\dagger &= \left(\boldsymbol{L}^T\boldsymbol{L}\right)^{-1}\boldsymbol{L}^T \\
&= \left(\boldsymbol{R}^T(\boldsymbol{I}_k \otimes \boldsymbol{H}^T\boldsymbol{H})\boldsymbol{R}\right)^{-1}\boldsymbol{R}^T(\boldsymbol{I}_k \otimes \boldsymbol{H})^T.
\end{aligned}$$

Next, we define $\boldsymbol{\Omega}$ by completing the range space of $\boldsymbol{\Psi}$

$$\begin{aligned}\boldsymbol{\Omega} &= \boldsymbol{\Psi} + \boldsymbol{L}\boldsymbol{L}^T \\ &= \boldsymbol{H}^T\boldsymbol{H} \otimes \boldsymbol{I}_n + (\boldsymbol{I}_k \otimes \boldsymbol{H})(\boldsymbol{C}_k + \boldsymbol{R}\boldsymbol{R}^T)(\boldsymbol{I}_k \otimes \boldsymbol{H})^T,\end{aligned}$$

which is full rank, and its inverse can be obtained via matrix inversion lemma as

$$\begin{aligned}&\boldsymbol{\Omega}^{-1} = \left(\boldsymbol{H}^T\boldsymbol{H}\right)^{-1} \otimes \boldsymbol{I}_n - \\ &\left(\left(\boldsymbol{H}^T\boldsymbol{H}\right)^{-1} \otimes \boldsymbol{H}\right)\left(\left(\boldsymbol{C}_k + \boldsymbol{R}\boldsymbol{R}^T\right)^{-1} + \left(\boldsymbol{H}^T\boldsymbol{H}\right)^{-1} \otimes \boldsymbol{H}^T\boldsymbol{H}\right)^{-1}\left(\left(\boldsymbol{H}^T\boldsymbol{H}\right)^{-1} \otimes \boldsymbol{H}\right)^T.\end{aligned}$$

Finally, the CRB is obtained by $\boldsymbol{\Phi}^\dagger = (\sigma^2/2)\boldsymbol{\Psi}^\dagger$ where

$$\boldsymbol{\Psi}^\dagger = \boldsymbol{\Omega}^{-1} - (\boldsymbol{I}_k \otimes \boldsymbol{H})\boldsymbol{R}\left(\boldsymbol{R}^T(\boldsymbol{I}_k \otimes \boldsymbol{H}^T\boldsymbol{H})\boldsymbol{R}\right)^{-2}\boldsymbol{R}^T(\boldsymbol{I}_k \otimes \boldsymbol{H})^T.$$

Using this procedure, the biggest matrix that need to be inverted is of size $k^2 \times k^2$, comparing to the plain size of the FIM $nk \times nk$.

6.4 Simulations

Now we provide some numerical experiments regarding the symmetric NMF problem. We start by testing the optimization performance of Alg. 6.1 on synthetic data, then the estimation performance benchmarked by the CRB that we developed for the symmetric matrix factorization model, and finally apply symmetric NMF to a clustering task on real data.

6.4.1 Synthetic Data

The matrix $\boldsymbol{Y}$ is generated by taking $\boldsymbol{Y} = \boldsymbol{H}\boldsymbol{H}^T + \boldsymbol{N}$, where $\boldsymbol{H}$ is a non-negative matrix with certain amount of zeros and the non-zero entries drawn from an i.i.d. exponential distribution, and the elements of $\boldsymbol{N}$ first drawn from an i.i.d. Gaussian distribution, and then symmetrized by taking $\boldsymbol{N} \leftarrow \boldsymbol{N} + \boldsymbol{N}^T$. We take the size of $\boldsymbol{H}$ to be 1000×150. The tolerance we set to terminate Alg. 6.1 is 10^{-5}, and we let α-SNMF and β-SNMF to run the same amount of time to compare their performances.

The convergence of a single run of our proposed algorithm under various conditions is illustrated in Fig. 6.1 (in terms of time used) and Fig. 6.2 (in terms of number

of iterations used), comparing to α-SNMF and β-SNMF provided in [154] with $\alpha = \beta = 0.99$, since their experiments showed (and we verified) that this value gives faster convergence, and the low-rank approximation (LRA) version of these algorithms, on the same $\boldsymbol{Y}$. The cost employed in both α-SNMF and β-SNMF is $\|\boldsymbol{Y} - \boldsymbol{H}\boldsymbol{H}^T\|_F^2$, which is different from the one used in Alg. 6.1, but we compare all of them using $\|\boldsymbol{Y} - \boldsymbol{H}\boldsymbol{H}^T\|_F$ on the y-axis as common basis. Since our proposed algorithm uses eigen-decomposition of the data matrix as a pre-processing step, we include the time it takes to compute this eigen-decomposition in the timing reported on the x-axis in Fig. 6.1, for fair comparison.

In Fig. 6.1, we show the convergence when $\boldsymbol{Y}$ is noiseless ($\boldsymbol{N} = 0$) in the top row, and with small noise (the entries of $\boldsymbol{N}$ are first drawn from an i.i.d. Gaussian distribution with standard deviation $\sigma = 10^{-1}$ and then symmetrized) in the bottom row; and the densities (proportion of non-zero entries) of the true latent factor $\boldsymbol{H}$ are (from left to right) 0.5, 0.7, 0.9, 1. In the noiseless case, our proposed algorithm tends to converge to an exact decomposition, whereas none of the SNMF variants is able to give a good approximation within that amount of time, although at the beginning they reduce the cost function faster. When small noise is added, the proposed algorithm shows good robustness, and again out-performs the two SNMF algorithms after some point. Notice that given the noise power, the symmetrization strategy, and the size of the matrix, the value of $\|\boldsymbol{N}\|_F$ is approximately 150, and our proposed algorithm is able to reach that error bound. An interesting observation is that the rate of convergence is somehow related to the sparsity of the true latent factor – the smaller the density, the faster the algorithm converges. Furthermore, overall the convergence rate looks linear, but swamps are sometimes encountered, which is why our proposed termination criterion is preferable than checking successive differences of the cost function. Our algorithm clearly outperforms all of the SNMF variants in this case.

It is important to note that after the computation of the k dominant eigen-components in the first step of our algorithm, each iteration (update cycle) entails complexity $O(nk^2)$, whereas one iteration of either α-SNMF or β-SNMF entails complexity $O(n^2k)$ (note that $k \leq n$). Therefore our algorithm also has an edge in terms of scaling up for big data, provided we have a scalable way to compute dominant eigenvalues and eigenvectors. Reduced-complexity variants of α-SNMF and β-SNMF have been very recently proposed to employ low-rank approximation (LRA) preprocessing to reduce

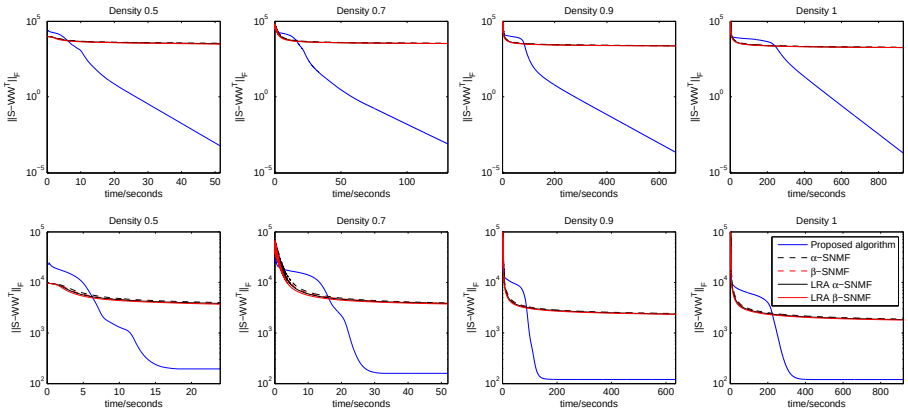


Figure 6.1: Convergence of the proposed algorithm vs. α-SNMF and β-SNMF [154] with $\alpha = \beta = 0.99$ and their modified versions employing low rank approximation (LRA): noiseless (top row) and noisy (bottom row). x-axis counts elapsed time.

the per-iteration complexity to $O(IK^2)$. Such a comparison of per-iteration counts is incomplete, however, as it does not take into account the number of iterations till convergence. Fig. 6.2 shows that the number of iterations is much smaller and the average convergence rate much faster for the proposed algorithm relative to the SNMF variants, in all cases considered. Note that in Fig. 6.2 the x-axis counts the number of iterations instead of total elapsed time as in Fig. 6.1.

6.4.2 Estimation performance

We compare three algorithms for symmetric NMF with the CRB derived before. These are α-SNMF and β-SNMF with $\alpha = \beta = 0.99$ [154], and Alg. 6.1. The true $\boldsymbol{H}$ is generated such that a certain proportion of its entries are randomly set to 0, and the rest are drawn from an i.i.d. exponential distribution. Using the generative model $\boldsymbol{Y} = \boldsymbol{H}\boldsymbol{H}^T + \boldsymbol{N}$, the resulting $\boldsymbol{Y}$ will not be symmetric, so we use $\frac{1}{2}(\boldsymbol{Y} + \boldsymbol{Y}^T)$, since all algorithms are designed specifically for symmetric non-negative matrices. The α-SNMF and β-SNMF algorithms proposed in [154] did not provide a termination criterion, so both α-SNMF and β-SNMF are left to run for a large number of iterations (10^4), to ensure the best possible results. The tolerance for the termination criterion in Alg. 6.1 is set to machine precision `eps`. We used a single draw of $\boldsymbol{H}$ for each (size,density)

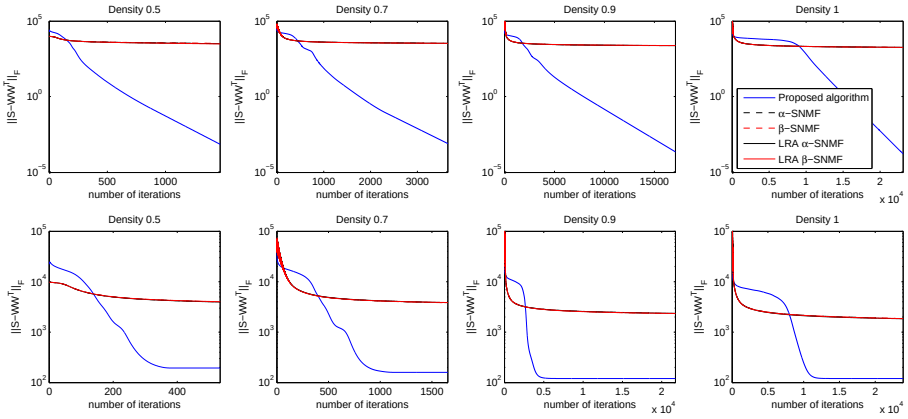


Figure 6.2: Convergence of the proposed algorithm vs. α-SNMF and β-SNMF [154] with $\alpha = \beta = 0.99$ and their modified versions employing low rank approximation (LRA): noiseless (top row) and noisy (bottom row). x-axis counts number of iterations.

combination reported. Under various SNRs, the normalized squared error $\frac{\|\hat{\boldsymbol{H}}-\boldsymbol{H}\|_F^2}{\|\boldsymbol{H}\|_F^2}$ is calculated and averaged over 100 Monte-Carlo tests, so that we can get a better approximation to the expected error $\mathsf{E}\left\{\frac{\|\hat{\boldsymbol{H}}-\boldsymbol{H}\|_F^2}{\|\boldsymbol{H}\|_F^2}\right\}$.

The results are plotted in Fig. 6.3, where the first row shows the normalized squared error benchmarked by the CRB, the second row shows the (aggregate) bias for each estimate, defined as

$$\text{bias} = \left\| \frac{1}{T} \sum_{t=1}^{T} (\boldsymbol{H} - \hat{\boldsymbol{H}}_t) \right\|_F , \tag{6.8}$$

where T is the number of trials, in this case 100; and the third row shows the model fitting error for each algorithm. The dashed lines in the third row show the total noise power; a good approximation should yield a fitting error close to the noise power. The plots in the left column show a case where the symmetric NMF problem is relatively "over-determined", since the inner dimension (30) is small compared to the outer dimension (200), and the latent factors are quite sparse (density 0.5). The two other columns show more difficult cases—low rank (30 vs. 200) but relatively dense latent factors for the middle column, not-so-low rank (50 vs. 100) but relatively sparse latent factors for the right column. Recall the discussion in § 2.3 on the rule of thumb for when identifiability can be expected—the middle and right plots illustrate cases where

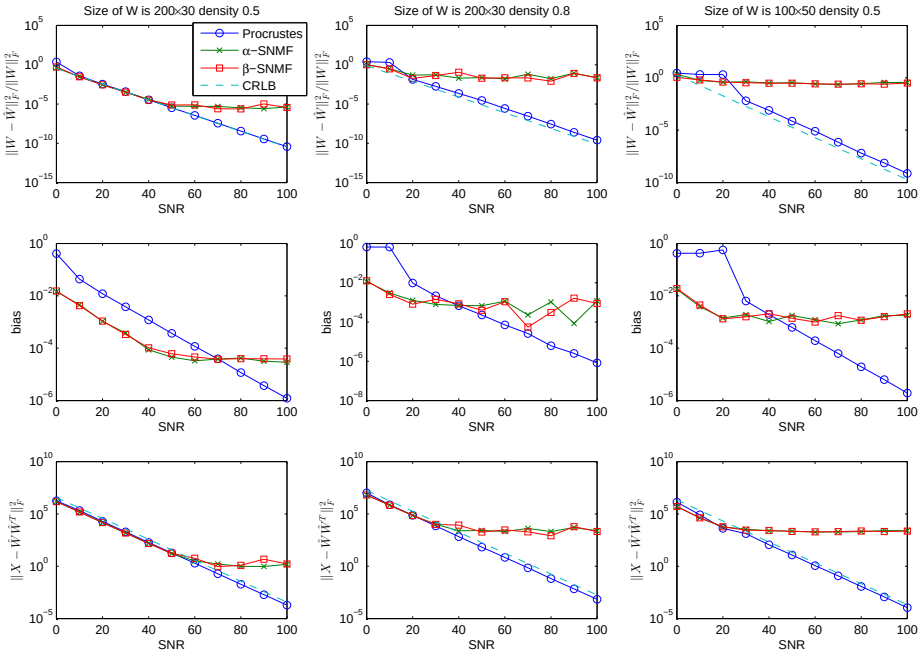


Figure 6.3: The first row shows the normalized squared error of three existing symmetric NMF algorithms versus the CRLB; similarly, the second row shows the (aggregate) bias, and the third row shows the fitting error.

this requirement is barely satisfied.

In all cases, the aggregate bias is small, and goes to zero as SNR increases, indicating that the estimates provided by these algorithms are asymptotically unbiased, and we can use the CRLB to approximately bound the performance. Generally speaking, α/β-SNMF slightly outperform Alg. 6.1 in the low SNR regime, but fail to reach the CRLB in the high SNR regime. Alg. 6.1 exhibits classic threshold behavior—for SNR higher than some threshold, the mean square error (MSE) stays close to the CRLB. The reason is that it employs eigen-analysis to estimate the column space of $\boldsymbol{H}$ as a first step, then applies Procrustes rotations in the estimated subspace. On the other hand, both SNMF variants are modifications of the multiplicative update algorithm using $\|\boldsymbol{Y} - \mathbf{W}\mathbf{W}^T\|_F^2$ (Gaussian log-likelihood) as the objective, so that it is not surprising that they perform better in the low-SNR regime. We can also see this from the second row of Fig. 6.3, as

the biases of α/β-SNMF are lower than that of the Procrustes method under low SNR.

6.4.3 Co-authorship clustering

We applied Alg. 6.1 to a real-life dataset containing co-authorship data from the U.S. Army Research Laboratory Collaborative Technology Alliance (ARL-CTA) on Communications and Networks (C&N), a large-scale research project that involved multiple academic and industry research groups, led by Telcordia. The ARL C&N CTA was an 8-year program, and produced numerous publications, involving over 500 individuals. A. Swami and N. Sidiropoulos were both involved as researchers and authors in this project, and A. Swami had significant oversight on much of the research – they know the 'social dynamics' and history of the consortium, and can interpret / sanity check the results of automated social network analysis of this dataset. The particular data analyzed here is a 518×518 symmetric non-negative matrix $\boldsymbol{Y}$, where $y_{i,j}$ is the number of papers co-authored by author-i and author-j ($y_{i,i}$ is the number of papers written by author-i). The task is to cluster the authors, based only on $\boldsymbol{Y}$. Ding *et al.* [149] have shown that k-means clustering can be approximated by symmetric NMF of the pair-wise similarity matrix $\boldsymbol{Y} = \boldsymbol{X}^T\boldsymbol{X} = \boldsymbol{H}\boldsymbol{H}^T$, where the columns of $\boldsymbol{X}$ represent the data points that we want to cluster, and the number of columns of $\boldsymbol{H}$, k, is the number of clusters. The cluster that $\boldsymbol{x}_i$ belongs to is determined by taking $\arg\max_k h_{i,k}$. In our case, we do not have access to $\boldsymbol{X}$, but we may interpret $\boldsymbol{Y}$ as the pair-wise similarity matrix $\boldsymbol{Y} = \boldsymbol{X}^T\boldsymbol{X}$, to be decomposed as $\boldsymbol{Y} = \boldsymbol{H}\boldsymbol{H}^T$, with $\boldsymbol{H} \geq 0$.

We run symmetric NMF on $\boldsymbol{Y}$ for $k = 3, 10$. The weight of cluster c is measured by $\|\boldsymbol{h}_c\|_2$, and the weight of author i in the cluster c is measured by $h_{i,c}$. Table 6.2 lists the top-10 contributors of the top-3 clusters, for $k = 3$ (top) and $k = 10$ (bottom). The results are very reasonable. The first cluster is Georgios Giannakis' group at the University of Minnesota, the participant who contributed most publications to the project.The second cluster is more interesting: it comprises Lang Tong's group at Cornell, but also close collaborators from ARL (Brian Sadler, Ananthram Swami) who co-authored many papers with Cornell researchers and alumni over the years. The third cluster is even more interesting, and would have been harder to decipher for someone without direct knowledge of the project. It consists of Telcordia researchers (Telcordia was the lead of the project), but it also contains researchers from the City University of New York

Table 6.2: Top-10 contributors of the top-3 clusters

$K = 3$	cluster 1	cluster 2	cluster 3
	Georgios B. Giannakis	Lang Tong	Mariusz A. Fecko
	Shengli Zhou	Ananthram Swami	Sunil Samtani
	Xiaoli Ma	Qing Zhao	M. Umit Uyar
	Pengfei Xia	Brian M. Sadler	Ibrahim Hokelek
	Xiaodong Cai	Yunxia Chen	Jianping Zou
	Tairan Wang	Min Dong	Jianliang Zheng
	Qingwen Liu	Youngchul Sung	Myung Jong Lee
	Xing Wang	Ting He	Tarek N. Saadawi
	Zhengdao Wang	P. Venkitasubramaniam	Ulas C. Kozat
	Alfonso Cano	Zhengyuan Xu	Phillip T. Conrad
$K = 10$	cluster 1	cluster 2	cluster 3
	Georgios B. Giannakis	Lang Tong	Mariusz A. Fecko
	Shengli Zhou	Ananthram Swami	Sunil Samtani
	Pengfei Xia	Brian M. Sadler	M. Umit Uyar
	Xiaodong Cai	Min Dong	Ibrahim Hokelek
	Qingwen Liu	Ting He	Jianping Zou
	Tairan Wang	Youngchul Sung	Jianliang Zheng
	Xing Wang	P. Venkitasubramaniam	Ulas C. Kozat
	Zhengdao Wang	Srihari Adireddy	Phillip T. Conrad
	Yingqun Yu	Gokhan Mergen	Ahmed Abdelal
	Alfonso Cano	Animashree Anandkumar	J. Sucec

(CUNY), and, to a lesser extent, the University of Delaware (UDEL), suggesting that geographic proximity may have a role. Interestingly, the network of collaborations between Telcordia, CUNY, and UDEL dates back to the FEDLAB project (which was in a sense the predecessor of the CTA), and continued through much of the CTA as well. Notice that the three clusters remain stable even when $k = 10 > 3$ is used, although NMF is not guaranteed to be nested (for even higher k, e.g., $k = 30$, this stability breaks down, as larger clusters are broken down into more tightly woven pieces).

Appendix

Modified Symmetric NMF Algorithm For Rank Less Than CP-Rank

The completely positive rank (cp-rank) of a completely positive matrix $\boldsymbol{Y}$ is the minimum k that allows exact symmetric NMF of $\boldsymbol{Y}$ [147]. It is well-known that the cp-rank need not be equal to the rank of $\boldsymbol{Y}$. If we indeed encounter a completely positive matrix with cp-rank strictly larger than its rank, this appendix shows how to modify Alg. 6.1 to seek an exact symmetric NMF, assuming we know the cp-rank.

Let the rank of a complete positive matrix $\boldsymbol{Y}$ be r, then we can perform the thin eigenvalue decomposition

$$\mathbf{S} = \boldsymbol{U}_r \boldsymbol{\Lambda}_r \boldsymbol{U}_r^T,$$

where $\boldsymbol{U}_r$ is $n \times r$ with orthonormal columns, and $\boldsymbol{\Lambda}_r$ is $r \times r$ diagonal. Assume the cp-rank is k, then there exists a symmetric NMF of $\boldsymbol{Y}$

$$\boldsymbol{Y} = \boldsymbol{H}\boldsymbol{H}^T,$$

where $\boldsymbol{H}$ is $n \times k$ with non-negative elements. The rank of $\boldsymbol{H}$ is r, otherwise the rank of $\boldsymbol{Y}$ would not be r. Therefore, we can take the thin SVD of $\boldsymbol{H}$

$$\boldsymbol{H} = \boldsymbol{U}_H \boldsymbol{\Sigma}_H \boldsymbol{V}_H^T,$$

where $\boldsymbol{U}_H$ is $n \times r$ with orthonormal columns, $\boldsymbol{\Sigma}_H$ is $r \times r$ diagonal, and $\boldsymbol{V}_H$ is $r \times k$ with orthonormal columns. Then

$$\boldsymbol{Y} = \boldsymbol{H}\boldsymbol{H}^T = \boldsymbol{U}_H \boldsymbol{\Sigma}_H^2 \boldsymbol{U}_H^T.$$

As we can see, the right-hand-side is also an eigenvalue decomposition. Since the eigenvalue decomposition is unique, this implies that $\boldsymbol{U}_r = \boldsymbol{U}_H$ and $\boldsymbol{\Lambda}_r^{1/2} = \boldsymbol{\Sigma}_H$. In other words, let

$$\boldsymbol{Y} = \boldsymbol{B}\boldsymbol{B}^T$$

where $\boldsymbol{B} = \boldsymbol{U}_r \boldsymbol{\Lambda}_r^{1/2}$, then there exists a $k \times r$ orthonormal matrix $\boldsymbol{Q}$ such that $\boldsymbol{B}\boldsymbol{Q}^T \geq 0$. Thus, finding $\boldsymbol{H}$ can be posed as the following optimization problem

$$\begin{aligned} \underset{\boldsymbol{H},\boldsymbol{Q}}{\text{minimize}} \quad & \|\boldsymbol{H} - \boldsymbol{B}\boldsymbol{Q}^T\|_F^2 \\ \text{subject to} \quad & \boldsymbol{H} \geq 0, \\ & \boldsymbol{Q}^T\boldsymbol{Q} = \boldsymbol{I}. \end{aligned} \tag{6.9}$$

Notice that compared to problem (6.2), we only have $\boldsymbol{Q}^T\boldsymbol{Q} = \boldsymbol{I}$, since $\mathbf{Q}$ is not square, and $\boldsymbol{Q}\boldsymbol{Q}^T$ is now a projection matrix.

Similar to Alg. 6.1, we propose to solve this problem by alternatingly updating $\boldsymbol{H}$ and $\boldsymbol{Q}$. For $\boldsymbol{H}$, the update rule is simply

$$\boldsymbol{H} \leftarrow \max(0, \boldsymbol{B}\boldsymbol{Q}^T) \tag{6.10}$$

For the case of $\boldsymbol{Q}$, the answer is not directly given by the Procrustes rotation. However, it is easy to show that the solution is similar to what the Procrustes rotation provided us in the unitary $\mathbf{Q}$ case. Since

$$\begin{aligned} & \|\boldsymbol{H} - \boldsymbol{B}\boldsymbol{Q}^T\|_F^2 \\ = \ & \text{trace}\left\{(\boldsymbol{H} - \boldsymbol{B}\boldsymbol{Q}^T)(\boldsymbol{H} - \boldsymbol{B}\boldsymbol{Q}^T)^T\right\} \\ = \ & \text{trace}\left\{\boldsymbol{H}\boldsymbol{H}^T\right\} + \text{trace}\left\{\boldsymbol{B}\boldsymbol{B}^T\right\} - 2\text{trace}\left\{\boldsymbol{H}^T\boldsymbol{B}\boldsymbol{Q}^T\right\}, \end{aligned}$$

minimizing $\|\boldsymbol{H} - \boldsymbol{B}\boldsymbol{Q}^T\|_F^2$ is equivalent to maximizing $\text{trace}\left\{\boldsymbol{H}^T\boldsymbol{B}\boldsymbol{Q}^T\right\}$.

Proposition 6.3. *A solution of the following optimization problem*

$$\begin{aligned} \underset{\boldsymbol{Q}}{\text{maximize}} \quad & \text{trace}\left\{\boldsymbol{H}^T\boldsymbol{B}\boldsymbol{Q}^T\right\} \\ \text{subject to} \quad & \boldsymbol{Q}^T\boldsymbol{Q} = \boldsymbol{I}, \end{aligned}$$

is $\boldsymbol{Q} = \boldsymbol{U}\boldsymbol{V}^T$, where $\boldsymbol{U}$ and $\boldsymbol{V}$ come from the singular value decomposition of $\boldsymbol{H}^T\boldsymbol{B}$.

$$\boldsymbol{H}^T\boldsymbol{B} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T.$$

Proof. Let the singular value decomposition of $\boldsymbol{H}^T\boldsymbol{B}$ be

$$\boldsymbol{H}^T\boldsymbol{B} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T,$$

where $\boldsymbol{U}$ is $k \times r$, $\boldsymbol{\Sigma}$ is $r \times r$ and $\boldsymbol{V}$ is $r \times r$. Then we have

$$\begin{aligned} & \text{trace}\left\{\boldsymbol{H}^T\boldsymbol{B}\boldsymbol{Q}^T\right\} \\ = \ & \text{trace}\left\{\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T\boldsymbol{Q}^T\right\} \\ = \ & \text{trace}\left\{\boldsymbol{\Sigma}\boldsymbol{V}^T\boldsymbol{Q}^T\boldsymbol{U}\right\} \\ = \ & \sum_{l=1}^{r} \sigma_l \boldsymbol{v}_l^T\boldsymbol{Q}^T\boldsymbol{u}_l \end{aligned}$$

where σ_l is the l-th diagonal entry of $\boldsymbol{\Sigma}$, and $\boldsymbol{v}_l$, $\boldsymbol{u}_l$ are the l-th column of $\boldsymbol{V}$, $\boldsymbol{U}$, respectively. Therefore,

$$\text{trace}\left\{\boldsymbol{H}^T\boldsymbol{B}\boldsymbol{Q}^T\right\} \leq \sum_{k=1}^{r}\sigma_k$$

because

$$\begin{aligned} & \boldsymbol{v}_l^T\boldsymbol{Q}^T\boldsymbol{u}_k \\ \leq\ & \|\boldsymbol{v}_l\|_2\|\boldsymbol{Q}^T\boldsymbol{u}_l\|_2 \\ \leq\ & \|\boldsymbol{v}_l\|_2\|\boldsymbol{u}_l\|_2 \\ =\ & 1 \end{aligned}$$

Furthermore, let $\boldsymbol{Q} = \boldsymbol{U}\boldsymbol{V}^T$, then

$$\begin{aligned} & \text{trace}\left\{\boldsymbol{H}^T\boldsymbol{B}\boldsymbol{Q}^T\right\} \\ =\ & \text{trace}\left\{\boldsymbol{\Sigma}\boldsymbol{V}^T\boldsymbol{Q}^T\boldsymbol{U}\right\} \\ =\ & \text{trace}\left\{\boldsymbol{\Sigma}\boldsymbol{V}^T\boldsymbol{V}\boldsymbol{U}^T\boldsymbol{U}\right\} \\ =\ & \text{trace}\left\{\boldsymbol{\Sigma}\right\} \end{aligned}$$

which attains the upper bound we just derived. Therefore, a solution for this optimization problem is $\boldsymbol{Q} = \boldsymbol{U}\boldsymbol{V}^T$. □

As an example, let

$$\boldsymbol{Y} = \begin{bmatrix} 2 & 1 & 1 & 0 \\ 1 & 2 & 0 & 1 \\ 1 & 0 & 2 & 1 \\ 0 & 1 & 1 & 2 \end{bmatrix}$$

Then $\text{rank}\left\{\boldsymbol{Y}\right\} = 3$ while $\text{cp-rank}\{\boldsymbol{Y}\} = 4$. If we apply the plain Alg. 6.1, in which case we set $k = 3$, the result is

$$\boldsymbol{H} = \begin{bmatrix} 0 & 1.3635 & 0.3630 \\ 0 & 0.3637 & 1.3633 \\ 1.1193 & 0.8516 & 0 \\ 1.1190 & 0 & 0.8520 \end{bmatrix}$$

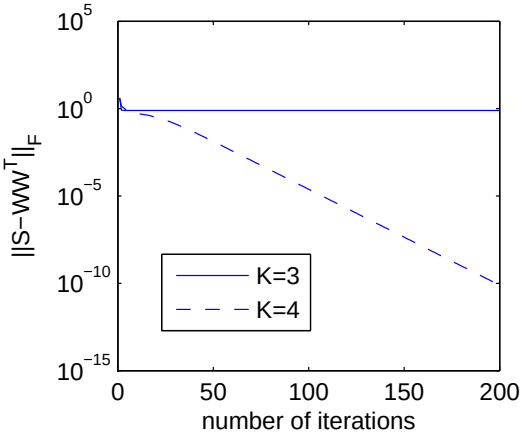


Figure 6.4: Convergence of the modified proposed algorithm vs. original Alg. 6.1 for a matrix whose rank is less than its cp-rank

and the factorization is not exact, whereas if we set $k = 4$ and apply the modified algorithm described in this appendix, the result is

$$\boldsymbol{H} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

and the factorization is exact. The convergence of each case is shown in Fig. 6.4.

References

[1] D. D. Lee and H. S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.

[2] D. Donoho and V. Stodden. When does non-negative matrix factorization give a correct decomposition into parts? In *Advances in Neural Information Processing Systems (NIPS)*, pages 1141–1148, 2003.

[3] K. Huang, N. D. Sidiropoulos, and A. Swami. Non-negative matrix factorization revisited: Uniqueness and algorithm for symmetric decomposition. *IEEE Transactions on Signal Processing*, 62(1):211–224, Jan 2014.

[4] N. D. Sidiropoulos and R. Bro. On the uniqueness of multilinear decomposition of N-way arrays. *Journal of chemometrics*, 14(3):229–239, 2000.

[5] A. Smilde, R. Bro, and P. Geladi. *Multi-way analysis: applications in the chemical sciences*. John Wiley & Sons, 2005.

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

[7] P. Comon. Tensors: a brief introduction. *IEEE Signal Processing Magazine*, 31(3):44–53, 2014.

[8] K. Huang, N. D. Sidiropoulos, and A. Swami. NMF revisited: New uniqueness results and algorithms. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2013.

[9] X. Fu, W.-K. Ma, K. Huang, and N. D. Sidiropoulos. Blind separation of quasi-stationary sources: exploiting convex geometry in covariance domain. *IEEE Transactions on Signal Processing*, 63(9):2306–2320, May 2015.

[10] X. Fu, W.-K. Ma, K. Huang, and N. D. Sidiropoulos. Robust volume minimization-based matrix factorization via alternating optimization. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2016.

[11] X. Fu, K. Huang, B. Yang, W.-K. Ma, and N. D. Sidiropoulos. Robust volume minimization-based matrix factorization for remote sensing and document clustering. *IEEE Transactions on Signal Processing*, 2016, to appear.

[12] K. Huang, N. D. Sidiropoulos, E. E. Papalexakis, C. Faloutsos, P. P. Talukdar, and T. Mitchell. Principled neuro-functional connectivity discovery. In *SIAM International Conference on Data Mining (SDM)*, 2015.

[13] K. Huang, N. D. Sidiropoulos, and A. P. Liavas. Efficient algorithms for 'universally' constrained matrix and tensor factorization. In *European Signal Processing Conference (EUSIPCO)*.

[14] K. Huang, N. D. Sidiropoulos, and A. P. Liavas. A flexible and efficient algorithmic framework for constrained matrix and tensor factorization. *IEEE Transactions on Signal Processing*, 64(19):5052 – 5065, Oct. 2016.

[15] K. Huang and N. D. Sidiropoulos. Putting nonnegative matrix factorization to the test: a tutorial derivation of pertinent Cramér-Rao bounds and performance benchmarking. *IEEE Signal Processing Magazine*, 31(3):76–86, May 2014.

[16] N. D. Sidiropoulos, L. De Lathauwer, X. Fu, K. Huang, E. E. Papalexakis, and C. Faloutsos. Tensor decomposition for signal processing and machine learning. *IEEE Transactions on Signal Processing*, 2016, submitted. arXiv preprint arXiv:1607.01668.

[17] K. Huang, X. Fu, and N. D. Sidiropoulos. Anchor-free correlated topic modeling: Identifiability and algorithm. In *Advances in Neural Information Processing Systems*, 2016, accepted.

[18] J. Brewer. Kronecker products and matrix calculus in system theory. *IEEE Transactions on circuits and systems*, 25(9):772–781, 1978.

[19] J. R. Magnus and H. Neudecker. The commutation matrix: some properties and applications. *The Annals of Statistics*, pages 381–394, 1979.

[20] S. L. Campbell and G. D. Poole. Computing nonnegative rank factorizations. *Linear Algebra and its Applications*, 35:175–182, 1981.

[21] J.-C. Chen. The nonnegative rank factorizations of nonnegative matrices. *Linear algebra and its applications*, 62:207–217, 1984.

[22] P. Paatero and U. Tapper. Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics*, 5(2):111–126, 1994.

[23] A. Berman and R. J. Plemmons. *Nonnegative matrices*. SIAM, 1979.

[24] A. Cichocki, R. Zdunek, A. H. Phan, and S. Amari. *Nonnegative Matrix and Tensor Factorizations: Applications to Exploratory Multi-way Data Analysis and Blind Source Separation*. John Wiley & Sons, 2009.

[25] S. A. Vavasis. On the complexity of nonnegative matrix factorization. *SIAM Journal on Optimization*, 20(3):1364–1377, 2009.

[26] F. A. Nielsen. Clustering of scientific citations in Wikipedia. In *WikiMania*, July 2008.

[27] W. Xu, X. Li, and Y. Gong. Document clustering based on non-negative matrix factorization. In *ACM International Conference on Research and Development in Informaion Retrieval (SIGIR)*, pages 267–273, August 2003.

[28] F. J. Theis, K. Stadlthanner, and T. Tanaka. First results on uniqueness of sparse non-negative matrix factorization. In *European Signal Processing Conference (EUSIPCO)*, Sept. 4-8, 2005, Antalya, Turkey.

[29] R. Schachtner, G. Pöppel, and E. W. Lang. Towards unique solutions of non-negative matrix factorization problems by a determinant criterion. *Digital Signal Processing*, 21(4):528–534, 2011.

[30] L. B. Thomas. Solution to problem 73-14, rank factorization of nonnegative matrices (by A. Berman and R.J. Plemmons). *SIAM Review*, 16(3):393–394, 1974.

[31] H. Laurberg, M. G. Christensen, M. D. Plumbley, L. K. Hansen, and S. H. Jensen. Theorems on positive data: On the uniqueness of NMF. *Computational Intelligence and Neuroscience*, 2008, doi:10.1155/2008/764206, 2008.

[32] S. Moussaoui, D. Brie, and J. Idier. Non-negative source separation: range of admissible solutions and conditions for the uniqueness of the solution. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 5, pages 289–292, March 18-23, 2005.

[33] H. Laurberg. *Non-negative Matrix Factorization: Theory and Methods*. PhD thesis, Aalgorg University, 2008.

[34] R. T. Rockafellar. *Convex analysis*, volume 28. Princeton University Press, 1996.

[35] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge University Press, 2004.

[36] L. Khachiyan, E. Boros, K. Borys, K. Elbassioni, and V. Gurvich. Generating all vertices of a polyhedron is hard. *Discrete & Computational Geometry*, 39(1):174–190, 2008.

[37] R. M. Freund and J. B. Orlin. On the complexity of four polyhedral set containment problems. *Mathematical programming*, 33(2):139–145, 1985.

[38] O. Mehanna, K. Huang, B. Gopalakrishnan, A. Konar, and N. D. Sidiropoulos. Feasible point pursuit and successive approximation of non-convex qcqps. *IEEE Signal Processing Letters*, 22(7):804–808, 2015.

[39] J. M. Bioucas-Dias, A. Plaza, N. Dobigeon, M. Parente, Q. Du, P. Gader, and J. Chanussot. Hyperspectral unmixing overview: Geometrical, statistical, and sparse regression-based approaches. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 5(2):354–379, 2012.

[40] W.-K. Ma, J. M. Bioucas-Dias, T.-H. Chan, N. Gillis, P. Gader, A. J. Plaza, A. Ambikapathi, and C.-Y. Chi. A signal processing perspective on hyperspectral unmixing: Insights from remote sensing. *IEEE Signal Processing Magazine*, 31(1):67–81, Jan 2014.

[41] S. Arora, R. Ge, and A. Moitra. Learning topic models–going beyond svd. In *IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1–10. IEEE, 2012.

[42] B. Recht, C. Re, J. Tropp, and V. Bittorf. Factoring nonnegative matrices with linear programs. In *Advances in Neural Information Processing Systems*, pages 1214–1222, 2012.

[43] X. Fu and W.-K. Ma. Robustness analysis of structured matrix factorization via self-dictionary mixed-norm optimization. *IEEE Signal Processing Letters*, 23(1):60–64, 2016.

[44] M. C. U. Araújo, T. C. B. Saldanha, R. K. H. Galvão, T. Yoneyama, H. C. Chame, and V. Visani. The successive projections algorithm for variable selection in spectroscopic multicomponent analysis. *Chemometrics and Intelligent Laboratory Systems*, 57(2):65–73, 2001.

[45] M. D. Craig. Minimum-volume transforms for remotely sensed data. *IEEE Transactions on Geoscience and Remote Sensing*, 32(3):542–552, 1994.

[46] F.-Y. Wang, C.-Y. Chi, T.-H. Chan, and Y. Wang. Nonnegative least-correlated component analysis for separation of dependent sources by volume maximization. *IEEE transactions on pattern analysis and machine intelligence*, 32(5):875–888, 2010.

[47] C.-H. Lin, W.-K. Ma, W.-C. Li, C.-Y. Chi, and A. Ambikapathi. Identifiability of the simplex volume minimization criterion for blind hyperspectral unmixing: The no-pure-pixel case. *IEEE Transactions on Geoscience and Remote Sensing*, 53(10):5530–5546, 2015.

[48] B. Chai, D. Walther, D. Beck, and F.-F. Li. Exploring functional connectivities of the human brain using multivariate information analysis. In *Advances in Neural Information Processing Systems 22*, pages 270–278. 2009.

[49] M. Mørup, K. Madsen, A. M. Dogonowski, H. Siebner, and L. K. Hansen. Infinite relational modeling of functional connectivity in resting state fMRI. In *Advances in Neural Information Processing Systems 23*, pages 1750–1758. 2010.

[50] V. Sakkalis. Review of advanced techniques for the estimation of brain connectivity measured with EEG/MEG. *Computers in Biology and Medicine*, 41(12):1110–1117, 2011. Special Issue on Techniques for Measuring Brain Connectivity.

[51] E. E. Papalexakis, A. Fyshe, N. D. Sidiropoulos, P. P. Talukdar, T. M. Mitchell, and C. Faloutsos. Good-enough brain model: Challenges, algorithms and discoveries in multi-subject experiments. *ACM SIGKDD*, 2014.

[52] S. Turaga, L. Buesing, A. M. Packer, H. Dalgleish, N. Pettit, M. Hausser, and J. Macke. Inferring neural population dynamics from multiple partial recordings of the same neural circuit. In *Advances in Neural Information Processing Systems*, pages 539–547, 2013.

[53] D. Pfau, E. A. Pnevmatikakis, and L. Paninski. Robust learning of low-dimensional dynamics from large neural ensembles. In *Advances in Neural Information Processing Systems 26*, pages 2391–2399. Curran Associates, Inc., 2013.

[54] L. Ljung. *System Identification: Theory for the User*. Prentice Hall, 2nd edition, 1999.

[55] Z. Liu and L. Vandenberghe. Interior-point method for nuclear norm approximation with application to system identification. *SIAM Journal on Matrix Analysis and Applications*, 31(3):1235–1256, 2009.

[56] S. Muthukumaraswamy. High-frequency brain activity and muscle artifacts in MEG/EEG: A review and recommendations. *Frontiers in Human Neuroscience*, 7(138), 2013.

[57] S. Roberts and R. Choudrey. Bayesian independent component analysis with prior constraints: An application in biosignal analysis. In *Deterministic and Statistical Methods in Machine Learning*, pages 159–179. Springer Berlin Heidelberg, 2005.

[58] P. J. Antsaklis and A. N. Michel. *Linear Systems*. Springer, 2006.

[59] H. W. Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

[60] G. Sudre, D. Pomerleau, M. Palatucci, L. Wehbe, A. Fyshe, R. Salmelin, and T. Mitchell. Tracking neural coding of perceptual and semantic features of concrete nouns. *NeuroImage*, 62(1):451–463, 2012.

[61] C. Eckart and G. Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.

[62] G. Tomasi and R. Bro. A comparison of algorithms for fitting the PARAFAC model. *Computational Statistics & Data Analysis*, 50(7):1700–1734, 2006.

[63] B. A. Olshausen and D. J. Field. Sparse coding with an overcomplete basis set: A strategy employed by V1? *Vision Research*, 37(23):3311–3325, 1997.

[64] T. Hofmann. Probabilistic latent semantic indexing. In *Proc. ACM SIGIR Conference*, pages 50–57, 1999.

[65] E. J. Candès and B. Recht. Exact matrix completion via convex optimization. *Foundations of Computational Mathematics*, 9(6):717–772, 2009.

[66] E. J. Candès, X. Li, Y. Ma, and J. Wright. Robust principal component analysis? *Journal of the ACM*, 58(3):11, 2011.

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

[68] C. J. Hillar and L.-H. Lim. Most tensor problems are NP-hard. *Journal of the ACM*, 60(6):45, 2013.

[69] B. W. Bader and T. G. Kolda. Efficient MATLAB computations with sparse and factored tensors. *SIAM Journal on Scientific Computing*, 30(1):205–231, 2007.

[70] U Kang, E. E. Papalexakis, A. Harpale, and C. Faloutsos. GigaTensor: scaling tensor analysis up by 100 times-algorithms and discoveries. In *Proc. ACM SIGKDD*, pages 316–324, 2012.

[71] N. Ravindran, N. D. Sidiropoulos, S. Smith, and G. Karypis. Memory-efficient parallel computation of tensor and matrix products for big tensor decomposition. In *Asilomar Conference on Signals, Systems, and Computers*, 2014.

[72] J. H. Choi and S. V. N. Vishwanathan. DFacTo: Distributed factorization of tensors. In *Advances in Neural Information Processing Systems*, pages 1296–1304, 2014.

[73] S. Smith, N. Ravindran, N. D. Sidiropoulos, and G. Karypis. SPLATT: Efficient and parallel sparse tensor-matrix multiplication. In *IEEE International Parallel & Distributed Processing Symposium*, 2015.

[74] S. Smith and G. Karypis. Tensor-matrix products with a compressed sparse tensor. In *Proceedings of the 5th Workshop on Irregular Applications: Architectures and Algorithms*, page 7. ACM, 2015.

[75] S. Smith and G. Karypis. A medium-grained algorithm for distributed sparse tensor factorization. In *IEEE International Parallel & Distributed Processing Symposium*, 2016.

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

[77] P. Tseng. Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of Optimization Theory and Applications*, 109(3):475–494, 2001.

[78] L. Grippo and M. Sciandrone. On the convergence of the block nonlinear Gauss-Seidel method under convex constraints. *Operations Research Letters*, 26(3):127–136, 2000.

[79] B. Chen, S. He, Z. Li, and S. Zhang. Maximum block improvement and polynomial optimization. *SIAM Journal on Optimization*, 22(1):87–107, 2012.

[80] M. Razaviyayn, M. Hong, and Z.-Q. Luo. A unified convergence analysis of block successive minimization methods for nonsmooth optimization. *SIAM Journal on Optimization*, 23(2):1126–1153, 2013.

[81] N. Li, S. Kindermann, and C. Navasca. Some convergence results on the regularized alternating least-squares method for tensor decomposition. *Linear Algebra and its Applications*, 438(2):796–812, 2013.

[82] Y. Xu and W. Yin. A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion. *SIAM Journal on Imaging Sciences*, 6(3):1758–1789, 2013.

[83] S. P. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122, 2011.

[84] E. Ryu and S. P. Boyd. Primer on monotone operator methods. Preprint, available at `http://web.stanford.edu/~eryu/papers/monotone_notes.pdf`.

[85] E. Ghadimi, A. Teixeira, I. Shames, and M. Johansson. Optimal parameter selection for the alternating direction method of multipliers (ADMM): quadratic problems. *IEEE Transactions on Automatic Control*, 60(3):644–658, March 2015.

[86] Y. Xu, W. Yin, Z. Wen, and Y. Zhang. An alternating direction algorithm for matrix completion with nonnegative factors. *Frontiers of Mathematics in China*, 7(2):365–384, 2012.

[87] K. Huang and N. D. Sidiropoulos. Consensus ADMM for general quadratically constrained quadratic programming. *IEEE Transactions on Signal Processing*, 2016, to appear.

[88] N. Parikh and S. P. Boyd. Proximal algorithms. *Foundations and Trends® in Optimization*, 1(3):123–231, 2014.

[89] J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra. Efficient projections onto the l_1-ball for learning in high dimensions. In *Proc. ACM ICML*, pages 272–279, 2008.

[90] G. H. Golub and C. F. Van Loan. *Matrix Computations*, volume 3. Johns Hopkins University Press, 1996.

[91] D. L. Sun and C. Fevotte. Alternating direction method of multipliers for non-negative matrix factorization with the beta-divergence. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 6201–6205, 2014.

[92] A. Cichocki and A.-H. Phan. Fast local algorithms for large scale nonnegative matrix and tensor factorizations. *IEICE Trans. on Fundamentals of Electronics, Communications and Computer Sciences*, 92(3):708–721, 2009.

[93] J. B. Kruskal, R. A. Harshman, and M. E. Lundy. How 3-MFA data can cause degenerate PARAFAC solutions, among other relationships. In *Multiway Data Analysis*, pages 115–122. 1989.

[94] A. Stegeman. Finding the limit of diverging components in three-way candecomp/parafac-a demonstration of its practical merits. *Comput. Stat. Data Anal.*, 75:203–216, July 2014.

[95] A. P. Liavas and N. D. Sidiropoulos. Parallel algorithms for constrained tensor factorization via the alternating direction method of multipliers. *IEEE Trans. on Signal Processing*, 63(20):5450–5463, 2015.

[96] D. Chen and R. J. Plemmons. Nonnegativity constraints in numerical analysis. In *Symposium on the Birth of Numerical Analysis*, pages 109–140, 2009.

[97] R. Bro and S. De Jong. A fast non-negativity-constrained least squares algorithm. *Journal of Chemometrics*, 11(5):393–401, 1997.

[98] H. Kim and H. Park. Nonnegative matrix factorization based on alternating nonnegativity constrained least squares and active set method. *SIAM Journal on Matrix Analysis and Applications*, 30(2):713–730, 2008.

[99] J. Kim and H. Park. Fast nonnegative matrix factorization: An active-set-like method and comparisons. *SIAM Journal on Scientific Computing*, 33(6):3261–3281, 2011.

[100] J. Kim and H. Park. Fast nonnegative tensor factorization with an active-set-like method. In *High-Performance Scientific Computing*, pages 311–326. Springer, 2012.

[101] M. H. Van Benthem and M. R. Keenan. Fast algorithm for the solution of large-scale non-negativity-constrained least squares problems. *Journal of Chemometrics*, 18(10):441–450, 2004.

[102] D. D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. In *Advances in Neural Information Processing Systems (NIPS)*, pages 556–562, 2001.

[103] N. Gillis and F. Glineur. Accelerated multiplicative updates and hierarchical als algorithms for nonnegative matrix factorization. *Neural Computation*, 24(4):1085–1105, 2012.

[104] X. Cai, Y. Chen, and D. Han. Nonnegative tensor factorizations using an alternating direction method. *Frontiers of Mathematics in China*, 8(1):3–18, 2013.

[105] J. E. Cohen, R. C. Farias, and P. Comon. Fast decomposition of large nonnegative tensors. *IEEE Signal Processing Letters*, 22(7):862–866, 2015.

[106] L. Sorber, M. Van Barel, and L. De Lathauwer. Tensorlab v2.0. `http://www.tensorlab.net/`, January 2014.

[107] L. Sorber, M. Van Barel, and L. De Lathauwer. Structured data fusion. *IEEE Journal of Selected Topics in Signal Processing*, 9(4):586–600, 2015.

[108] N. Vervliet, O. Debals, L. Sorber, and L. De Lathauwer. Breaking the curse of dimensionality using decompositions of incomplete tensors: Tensor-based scientific computing in big data analysis. *IEEE Signal Processing Magazine*, 31(5):71–79, 2014.

[109] B. W. Bader, T. G. Kolda, et al. Matlab tensor toolbox version 2.6. `http://www.sandia.gov/~tgkolda/TensorToolbox/`, February 2015.

[110] S. Gandy, B. Recht, and I. Yamada. Tensor completion and low-n-rank tensor recovery via convex optimization. *Inverse Problems*, 27(2):025010, 2011.

[111] J. Liu, P. Musialski, P. Wonka, and J. Ye. Tensor completion for estimating missing values in visual data. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 35(1):208–220, 2013.

[112] X. Fu, K. Huang, W.-K. Ma, N. D. Sidiropoulos, and R. Bro. Joint tensor factorization and outlying slab suppression with applications. *IEEE Transactions on Signal Processing*, 63(23):6315–6328, Dec. 2015.

[113] C. A Andersson and R. Bro. The N-way toolbox for matlab. *Chemometrics and Intelligent Laboratory Systems*, 52(1):1–4, 2000.

[114] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on World Wide Web*, pages 285–295, 2001.

[115] A. M. Bruckstein, D. L. Donoho, and M. Elad. From sparse solutions of systems of equations to sparse modeling of signals and images. *SIAM review*, 51(1):34–81, 2009.

[116] E. J. Candès and M. B. Wakin. An introduction to compressive sampling. *IEEE Signal Processing Magazine*, 25(2):21–30, 2008.

[117] I. Tosic and P. Frossard. Dictionary learning. *IEEE Signal Processing Magazine*, 28(2):27–38, 2011.

[118] M. Aharon, M. Elad, and A. Bruckstein. k-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Trans. on Signal Processing*, 54(11):4311–4322, 2006.

[119] M. Razaviyayn, H.-W. Tseng, and Z.-Q. Luo. Dictionary learning for sparse representation: Complexity and algorithms. In *Proc. IEEE ICASSP*, pages 5247–5251, 2014.

[120] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online learning for matrix factorization and sparse coding. *Journal of Machine Learning Research*, 11:19–60, 2010.

[121] M. V. Afonso, J. M. Bioucas-Dias, and M. A. T. Figueiredo. Fast image recovery using variable splitting and constrained optimization. *IEEE Trans. on Image Processing*, 19(9):2345–2356, 2010.

[122] J. Yang and Y. Zhang. Alternating direction algorithms for l_1-problems in compressive sensing. *SIAM Journal on Scientific Computing*, 33(1):250–278, 2011.

[123] E. Esser, Y. Lou, and J. Xin. A method for finding structured sparse solutions to nonnegative least squares problems with applications. *SIAM Journal on Imaging Sciences*, 6(4):2010–2046, 2013.

[124] P. O. Hoyer. Non-negative matrix factorization with sparseness constraints. *Journal of Machine Learning Research*, 5:1457–1469, 2004.

[125] S. M. Kay. *Fundamentals of Statistical Signal Processing, Volume I: Estimation Theory*. Prentice-Hall, 1993.

[126] J. D. Gorman and A. O. Hero. Lower bounds for parametric estimation with constraints. *IEEE Transactions on Information Theory*, 36(6):1285–1301, 1990.

[127] P. Stoica and B. C. Ng. On the Cramér-Rao bound under parametric constraints. *IEEE Signal Processing Letters*, 5(7):177–179, 1998.

[128] P. Stoica and T. L. Marzetta. Parameter estimation problems with singular information matrices. *IEEE Transactions on Signal Processing*, 49(1):87–90, 2001.

[129] Z. Ben-Haim and Y. C. Eldar. On the constrained Cramér-Rao bound with a singular Fisher information matrix. *IEEE Signal Processing Letters*, 16(6):453–456, 2009.

[130] S. Basu and Y. Bresler. The stability of nonlinear least squares problems and the Cramér-Rao bound. *IEEE Transactions on Signal Processing*, 48(12):3426–3436, 2000.

[131] C. Qian, N. D. Sidiropoulos, K. Huang, L. Huang, and H.-C. So. Least squares phase retrieval using feasible point pursuit. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2016.

[132] C. Qian, N. D. Sidiropoulos, K. Huang, L. Huang, and H.-C. So. Phase retrieval using feasible point pursuit: Algorithms and cramer-rao bound. *IEEE Transactions on Signal Processing*, 2016, to appear.

[133] K. Huang, Y. C. Eldar, and N. D. Sidiropoulos. On convexity and identifiability in 1-d fourier phase retrieval. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2016.

[134] K. Huang, Y. C. Eldar, and N. D. Sidiropoulos. Phase retrieval from 1d fourier measurements: Convexity, uniqueness, and algorithms. *IEEE Transactions on Signal Processing*, 2016, to appear.

[135] A. Swami. Cramér-Rao bounds for deterministic signals in additive and multiplicative noise. *Signal Processing*, 53(2):231–244, 1996.

[136] A. Swami and B. M. Sadler. On some detection and estimation problems in heavy-tailed noise. *Signal Processing*, 82(12):1829–1846, 2002.

[137] K. B. Petersen and M. S. Pedersen. *The matrix cookbook*. Technical University of Denmark, 2006.

[138] C. Hung and T. L. Markham. The Moore-Penrose inverse of a partitioned matrix $M = \begin{pmatrix} AD \\ BC \end{pmatrix}$. *Linear Algebra and its Applications*, 11(1):73–86, 1975.

[139] C. Hung and T. L. Markham. The Moore-Penrose inverse of a sum of matrices. *J. of the Australian Mathematical Society (Series A)*, 24(4):385–392, 1977.

[140] X. Liu and N. D. Sidiropoulos. Cramér-rao lower bounds for low-rank decomposition of multidimensional arrays. *IEEE Transactions on Signal Processing*, 49(9):2074–2086, 2001.

[141] S. A. Vorobyov, Y. Rong, N. D. Sidiropoulos, and A. B. Gershman. Robust iterative fitting of multilinear models. *IEEE Transactions on Signal Processing*, 53(8):2678–2689, 2005.

[142] G. Tomasi. *Practical and computational aspects in chemometric data analysis*. PhD thesis, 2006.

[143] A.-H. Phan, P. Tichavsky, and A. Cichocki. Low complexity damped gauss–newton algorithms for candecomp/parafac. *SIAM Journal on Matrix Analysis and Applications*, 34(1):126–147, 2013.

[144] P. Tichavsky, A.-H. Phan, and Z. Koldovsky. Cramér-rao-induced bounds for candecomp/parafac tensor decomposition. *IEEE Transactions on Signal Processing*, 61(8):1986–1997, 2013.

[145] M. W. Berry, M. Browne, A. N. Langville, V. P. Pauca, and R. J. Plemmons. Algorithms and applications for approximate nonnegative matrix factorization. *Computational Statistics & Data Analysis*, 52(1):155–173, 2007.

[146] C.-J. Lin. Projected gradient methods for nonnegative matrix factorization. *Neural Computation*, 19(10):2756–2779, 2007.

[147] A. Berman and N. Shaked-Monderer. *Completely positive matrices*. World Scientific, 2003.

[148] P. J. C. Dickinson and L. Gijben. On the computational complexity of membership problems for the completely positive cone and its dual. *Computational optimization and applications*, 57(2):403–415, 2014.

[149] C. Ding, X. He, and H.D. Simon. On the equivalence of nonnegative matrix factorization and spectral clustering. In *Proc. SIAM Data Mining Conf.*, pages 606–610, 2005.

[150] G. H. Golub and C. F. Van Loan. *Matrix computations*. Johns Hopkins University Press, 1996.

[151] P. H. Schönemann. A generalized solution of the orthogonal Procrustes problem. *Psychometrika*, 31(1):1–10, 1966.

[152] M. R. Sepanski. *Compact Lie Groups*, volume 235. Springer, 2007.

[153] M. Gardner, K. Huang, E. E. Papalexakis, X. Fu, P. P. Talukdar, C. Faloutsos, N. D. Sidiropoulos, and T. Mitchell. Translation invariant word embeddings. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2015.

[154] Z. He, S. Xie, R. Zdunek, G. Zhou, and A. Cichocki. Symmetric nonnegative matrix factorization: Algorithms and applications to probabilistic clustering. *IEEE Trans. on Neural Networks*, 22(12):2117–2131, 2011.