Unsupervised Learning of Latent Structure from Linear and Nonlinear Measurements

A DISSERTATION

SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL

OF THE UNIVERSITY OF MINNESOTA

BY

Bo Yang

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

Nicholas D. Sidiropoulos, Advisor

June, 2019

Acknowledgments

My sincerest gratitude goes to my advisor, Prof. Nikos Sidiropoulos. He helped me navigate through the mist at the frontier of human knowledge with his technical acumen. He taught me, directly and indirectly, many invaluable lessons on the various aspects of conducting research: formulating problems, solving problems, and presenting results. He clearly identified my weaknesses and suggested ways to overcome them. In a way, he practices the ancient wisdom of Confucius, "to teach in accordance with their aptitude", and I have benefited greatly.

I am deeply indebted to Dr. Xiao Fu. Dr. Fu was a postdoctoral researcher with our research group when I joined the group. Effectively, he served as my first-order advisor. My collaboration with him put me in a unique position to capitalize on his outstanding technical capability, as well as his very good taste in research. He taught me the importance of focusing on real research problems – problems that have a clear motivation and significance.

I am very grateful to Professor Georgios Giannakis, Professor George Karypis, Professor Mingyi Hong for serving on my thesis committee. They have influenced me greatly, through lectures, personal interactions, as well as their research products (papers, books, and software codes). Their commitment to excellence in research and teaching sets good examples for me to follow. I would also like to thank several other faculty members in University of Minnesota that have helped me vastly through their wonderful lectures: Professor Zhi-Quan Luo, Professor Yousef Saad, Professor Arindam Banerjee, and Professor Jarvis Haupt.

I have had an enjoyable and intellectually stimulating experience with my former and current lab-mates: Balasubramanian Gopalakrishnan, John Tranter, Cheng Qian, Kejun Huang, Aritra Konar, Mikael Sorensen, Charilaos Kanatsoulis, Ahmed Zamzam, Nikos Kargas, Faisal Almutairi, Mohamed Salah Ibrahim, and Magda Amiridi. I am also very thankful to my friends Yanning Shen, Gang Wang, Liang Zhang, Tianyi Chen, Zhimeng Yin, Shuai Wang, and many others.

Finally, I want to thank my family: my mother Guixian Li, my late father Jianbin Yang, my sister Ling Yang, my wife Xi Chen and my son Bi-Qiang Will Yang. This work could not have been done without their unconditional love and support.

Dedication

To my family.

Abstract

The past few decades have seen a rapid expansion of our digital world. While early dwellers of the Internet exchanged simple text messages via email, modern citizens of the digital world conduct a much richer set of activities online: entertainment, banking, booking for restaurants and hotels, just to name a few. In our digitally enriched lives, we not only enjoy great convenience and efficiency, but also leave behind massive amounts of data that offer ample opportunities for improving these digital services, and creating new ones. Meanwhile, technical advancements have facilitated the emergence of new sensors and networks, that can measure, exchange and log data about real world events. These technologies have been applied to many different scenarios, including environmental monitoring, advanced manufacturing, healthcare, and scientific research in physics, chemistry, bio-technology and social science, to name a few. Leveraging the abundant data, learning-based and data-driven methods have become a dominating paradigm across different areas, with data analytics driving many of the recent developments.

However, the massive amount of data also bring considerable challenges for analytics. Among them, the collected data are often high-dimensional, with the true knowledge and signal of interest hidden underneath. It is of great importance to reduce data dimension, and transform the data into the *right* space. In some cases, the data are generated from certain generative models that are *identifiable*, making it possible to reduce the data back to the original space. In addition, we are often interested in performing some analysis on the data after dimensionality reduction (DR), and it would be helpful to be mindful about these subsequent analysis steps when performing DR, as latent structures can serve as a valuable prior. Based on this reasoning, we develop two methods, one for the linear generative model case, and the other one for the nonlinear case. In a related setting, we study parameter estimation under unknown nonlinear distortion. In this case, the unknown nonlinearity in measurements poses a severe challenge. In practice, various mechanisms can introduce nonlinearity in the measured data. To combat this challenge, we put forth a nonlinear mixture model, which is well-grounded in real world applications. We show that this model is in fact identifiable up to some trivial indeterminancy. We develop an efficient algorithm to recover latent parameters of this model, and confirm the effectiveness of our theory and algorithm via numerical experiments.

Contents

Acknowledgments **i**

Dedication **iii**

Abstract **iv**

List of Tables **ix**

List of Figures **x**

1 Introduction **1**

1.1 Motivation 1

1.2 Thesis Outline and Contributions 2

1.3 Notational Conventions 4

2 Joint Factorization and Latent Clustering **5**

2.1 Introduction 5

2.2 Background 9

2.2.1 Clustering and latent clustering 9

2.2.2 Identifiable factorization models 11

2.3 Proposed approach 14

2.3.1 Problem formulation 14
2.3.2 Design considerations 15
2.4 Optimization algorithms 18
2.4.1 Joint NMF and K-means (JNKM) 18
2.4.2 Joint NTF and K-means (JTKM) 20
2.4.3 Joint VolMin and K-means (JVKM) 21
2.5 Extension: Joint factor analysis and subspace clustering 22
2.5.1 Formulation 22
2.5.2 Joint NMF and K-subspace (JNKS) algorithm 23
2.6 Convergence and complexity 25
2.6.1 Convergence properties 25
2.6.2 Complexity 26
2.7 Synthetic data study 27
2.7.1 Joint Matrix Factorization and Latent K-means 27
2.8 Real-data validation 36
2.8.1 Document clustering 36
2.8.2 Image clustering 38
2.8.3 Tensor social network data analysis 38
2.9 Chapter summary 41

3 Simultaneous Deep Learning and Clustering **42**
3.1 Introduction 42
3.2 Background and related works 46
3.3 Proposed formulation 48
3.4 Optimization procedure 50
3.4.1 Initialization via layer-wise pre-training 51
3.4.2 Alternating stochastic optimization 51

3.5 Experiments . . . 53
3.5.1 Synthetic-data demonstration . . . 54
3.5.2 Real-data validation . . . 55
3.6 Chapter summary . . . 63
3.7 Appendix . . . 63
3.7.1 Additional synthetic data experiments . . . 63
3.7.2 Additional real data experiments . . . 66

4 Learning Nonlinear Mixture Models **67**
4.1 Introduction . . . 67
4.1.1 Contributions . . . 68
4.1.2 Related work . . . 70
4.1.3 Organization . . . 71
4.2 Problem statement . . . 72
4.2.1 Review of linear mixture model . . . 72
4.2.2 Proposed signal model . . . 73
4.3 Identifiability analysis . . . 75
4.3.1 A technical lemma . . . 75
4.3.2 Nonlinear mixture model identification . . . 76
4.3.3 Existence of solutions . . . 79
4.3.4 Parameter identification after removing nonlinearity . . . 81
4.4 Learning algorithm . . . 82
4.5 Numerical experiments . . . 85
4.5.1 Synthetic data study . . . 85
4.5.2 Case study with a hyperspectral image . . . 89
4.6 Chapter summary . . . 92
4.7 Appendix . . . 93

4.7.1 Some definitions in convex geometry 93
4.7.2 Identifiability of LMM 93
4.7.3 Proof of Lemma 1 95
4.7.4 Proof of Theorem 1 96
4.7.5 Proof of Proposition 1 96

5 Summary and Future Directions **99**

References **102**

List of Tables

2.1 Comparison of cosine distance in data and latent domain. Data samples are taken from two clusters for each dataset. 7

2.2 Clustering and factorization accuracy for identifiable NMF vs. SNR_2, for $I = 50, J = 1000, F = 7, K = 10$, $\text{SNR}_1 = 15$ dB. 30

2.3 Clustering and factorization accuracy for identifiable NMF vs. SNR_1, for $I = 50, J = 1000, F = 7, K = 10$, $\text{SNR}_2 = 10$ dB. 30

2.4 Simulation comparison of the clustering methods, identifiable NMF model. $I = 50$, $F = 7$, $J = 100K$. 32

2.5 Clustering and factorization accuracy for identifiable VolMin vs. SNR_2, for $I = 50, J = 1000, F = 7, K = 10$. 34

2.6 MSE and clustering accuracy of JTKM vs. NTF for various $F = K$. 35

2.7 Simulation comparison of proposed JNKS with LS3C 36

2.8 The Legal cluster identified by the three methods, Bold entries are miss-clustered. 40

3.1 Evaluation on the RCV1-v2 dataset 58

3.2 Evaluation on the 20Newsgroup dataset. 59

3.3 Evaluation on the raw MNIST dataset. 61

3.4 Evaluation on pre-processed MNIST 62

3.5 Evaluation on the Pendigits dataset 66

List of Figures

2.1 The effect of distance distortion introduced by $\boldsymbol{W}$. Left: $\boldsymbol{H}$ on a 2-dimensional plane. Right: $\boldsymbol{X} = \boldsymbol{W}\boldsymbol{H}$ on a 2-dimensional plane. 6

2.2 Normalization in the latent domain helps bringing data points together, creating tight cluster structures. Figure generated by taking a plain NMF on two clusters of documents from Reuters-21578 dataset. Left: 2-dimensional representations of documents; Right: the normalized representations. 16

2.3 Illustration of how linear transformation obscures the latent cluster structure, and how identifiable models can recover this cluster structure. Top left: true latent factor $\boldsymbol{H}$; Top right: data domain $\boldsymbol{X} = \boldsymbol{W}\boldsymbol{H} + \boldsymbol{E}_1$, visualized using SVD (two principal components); Middle left: projected data found by RKM, $\boldsymbol{P}^\mathsf{T}\boldsymbol{X}$; Middle right: projected data found by FKM, $\boldsymbol{P}^\mathsf{T}\boldsymbol{X}$; Bottom left: $\boldsymbol{H}$ found by JNKM; Bottom right: $\boldsymbol{H}\boldsymbol{D}$ found by JNKM. In the top right subfigure, the clustering accuracy of running K-means directly on the data is shown; for other figures, the clustering accuracy given by corresponding method is shown. 31

2.4 AC and MSE versus parameter λ 33

2.5 Clustering accuracy with different number of clusters on Reuters-21578 dataset 37

2.6 Clustering accuracy with different number of clusters on MNIST dataset, each cluster has 200 samples. 39

2.7 Clustering accuracy with different number of samples, all the 10 digits (clusters) from MNIST dataset. 39

3.1 The learned 2-D reduced-dimension data by different methods. The observable data is in the 100-D space and is generated from 2-D data (cf. the first subfigure) through the nonlinear transformation in (3.9). The true cluster labels are indicated using different colors. 45

3.2 A problematic *joint* deep clustering structure. To avoid clutter, some links are omitted. 48

3.3 Proposed deep clustering network (DCN). 50

3.4 The sizes of 20 clusters in the experiment. 57

3.5 Clustering performance metrics v.s. training epochs. 59

3.6 Visualization using t-SNE. From top-left to bottom-right: Original data, DEC result, DCN initialization, DCN result 60

3.7 Visualization on the 4-clusters subset of RCV1-v2 60

3.8 Clustering performance on MNIST with different λ. 63

3.9 The generated latent representations $\{\boldsymbol{h}_i\}$ in the 2-D space and the recovered 2-D representations from $\boldsymbol{x}_i \in \mathbb{R}^{100}$, where $\boldsymbol{x}_i = (\sigma(\boldsymbol{W}\boldsymbol{h}_i))^2$. 64

3.10 The generated latent representations $\{\boldsymbol{h}_i\}$ in the 2-D space of the recovered 2-D representations from $\boldsymbol{x}_i \in \mathbb{R}^{100}$, where $\boldsymbol{x}_i = \tanh(\sigma(\boldsymbol{W}\boldsymbol{h}_i))$. 65

4.1 Learned functions ($\widehat{f}_i$'s, as parametrized by the neural network) and their composition with the ground truth nonlinear functions used for data generation. The four functions for data generation are $\phi_1(x) = x$, $\phi_2(x) = \sqrt{x}$, $\phi_3(x) = \sqrt[4]{x}$, $\phi_4(x) = \log(x+1)$. The ϕ_i's are kept secret in the learning stage. 85

4.2 Empirical CDF of MSE: the legend shows the learning method, and the nonlinear function used in data generation. For each nonlinear function, 100 trials are generated. A point $(-6, 0.99)$ on a curve means the corresponding learning method yields MSE $\leq 10^{-6}$ in 99% of the 100 trials. 86
4.3 Empirical CDF of MSE of estimating $\boldsymbol{S}$ with different number of data samples. 88
4.4 MSE results with different number of neurons. 89
4.5 MSE results with different rank 90
4.6 The Moffett Field hyperspectral image. 91
4.7 Visualizing columns of the estimated $\boldsymbol{S}$ corresponding to the water region. . . . 91
4.8 A geometric illustration of the sufficiently scattered condition (middle), a special case called separability (left), and a case that is not sufficiently scattered (right). 94

Chapter 1

Introduction

1.1 Motivation

Recent technology advancements have introduced and nurtured numerous large-scale digital services in many aspects of modern life, and the most well known examples include e-commerce, search engines, social networks, and e-services such as banking and bill pay. Human engagement in these services is producing massive amount of data. The ability to analyze these data is critical for building better digital products and services, and it also empowers scientific discovery [11, 122, 108]. The ever increasing capability of data collection provides novel views of many activities and phenomena that were previously hard to quantify. Yet, the resultant data tsunami also brings significant challenges for analytics. One is that, the collected data are often high-dimensional. The central task in these scenarios is to design methods that unveil the *underlying* causes, thus providing actionable insights based on the data. In order to tackle the high dimentionality problem, many dimensionality reduction (DR) methods have been proposed. The ultimate goal of DR is to shrink the dimension of data, so that they are easier for manipulation, analysis, communication, and storage.

In many cases, DR is employed as a pre-processing step on data, before the analysis step,

such as clustering. However, there is a critical issue with this approach. For real world data sets, where we have prior knowledge that clustering structure exists, using a general purpose DR technique can be suboptimal. Instead, a better method would involve incorporating this prior information when performing the DR step. Based on this intuition, we develop two methods for joint DR and latent clustering, to tackle the cases of linear and nonlinear transformation, respectively. In the linear case, we employ identifiable matrix factorization models to pin down the true latent space for clustering. In the nonlinear case, we employ a deep neural network to find a space that is "friendly" for clustering, so that very simple clustering algorithms perform well.

In a closely related setting, we study the problem of identifying the latent parameters of a nonlinear generative model. Many machine learning methods aim at learning a nonlinear representation of data. Prominent examples include kernel methods [106] and (deep) neural network-based methods [58]. A critical premise behind these methods is that, there exists a space, in which data representations are easier for downstream tasks, e.g., classification and regression. However, when indeed data are generated from a latent space via a nonlinear transformation, it is often unclear whether recovery of the latent parameters is possible. To study this problem, we propose a new nonlinear data model, which is well-motivated from real world applications. We show, surprisingly, that recovery of latent parameters in this highly nontrivial model is in fact possible, under realistic assumptions on the nonlinear data model. Furthermore, we design a novel learning method based on the new theory, and verify the effectiveness of the theory and algorithm via numerical experiments.

1.2 Thesis Outline and Contributions

In Chapter 2, we focus on developing a joint DR and clustering method, leveraging identifiable matrix and tensor factorization models. Our key observation is that when observing data from a

linear transformation, the latent clustering structure can be distorted. Methods based on principal component analysis (PCA) cannot rectify this issue, as the orthogonality structure is enforced in PCA. This orthogonality structure is unlikely to be compatible with the linear transformation in data generation. Building upon recent advances in identifiable matrix and tensor models, we propose a framework of joint factorization and latent clustering. Companion algorithms are devised based on the alternating-minimization algorithmic framework, and convergence issues are discussed. We also perform a comprehensive set of numerical experiments to evaluate the proposed approach. These results are reported in [129, 128].

In Chapter 3, we propose a clustering method exploiting deep neural networks. Many traditional clustering methods can be seen as first learning a nonlinear transformation of data, then performing a simple clustering algorithm, e.g., the K-means algorithm. Two well-known examples are spectral clustering [98, 120] and subspace clustering [118]. The hope is that the data will be mapped into a space that is suitable for clustering after the first step, so that a simple algorithm like K-means achieves good results. Unlike these methods that require much expertise and domain knowledge, e.g., in picking the right type of graph for spectral clustering, we design a data-driven approach where the nonlinear representations are learned from data, with the final goal of clustering in mind. As such, the learned representations are naturally suitable for clustering. Our method also resolves an issue in existing works [124, 132], where undesirable trivial solutions can emerge. The results in this chapter are reported in [130].

In Chapter 4, we present our results on learning the latent parameters of a nonlinear generative model. The model is motivated by several real world applications. We first present our new theoretical result, showing that it is possible to resolve *unknown* nonlinear distortion in an unsupervised fashion. Then we establish an optimization criterion to achieve this goal, and we show how it can be employed in practice – via an optimization formulation, building upon judiciously designed invertible neural networks. Finally, experimental results on synthetic data and real world data are provided, corroborating the effectiveness of the theory and algorithm.

These results are reported in [131].

Conclusions and discussion on future research directions are presented in Chapter 5.

1.3 Notational Conventions

Capital letters with underscore denote tensors, e.g., $\underline{\boldsymbol{X}}$; bold capital letters $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$ denote matrices; $\odot$ denotes the Khatri-Rao (column-wise Kronecker) product; $k_{\boldsymbol{A}}$ denotes the Kruskal rank of matrix $\boldsymbol{A}$, i.e., the maximal k such that *any* k columns of $\boldsymbol{A}$ are linearly independent; $\boldsymbol{A}^{\mathsf{T}}$ denotes the transpose of $\boldsymbol{A}$ and $\boldsymbol{A}^{\dagger}$ denotes the pseudo inverse of $\boldsymbol{A}$; $\boldsymbol{A}(i,:)$ and $\boldsymbol{A}(:,j)$ denote the ith row and the jth column of $\boldsymbol{A}$, respectively; $\underline{\boldsymbol{X}}(:,:,k)$ denotes the kth matrix slab of the three-way tensor $\underline{\boldsymbol{X}}$ taken perpendicular to the third mode; and likewise for slabs taken perpendicular to the second and first mode, $\underline{\boldsymbol{X}}(:,j,:)$, $\underline{\boldsymbol{X}}(i,:,:)$, respectively; $\|\mathbf{x}\|_0$ counts the non-zero elements in the vector $\mathbf{x}$; calligraphic letters denote sets, such as $\mathcal{J}$, $\mathcal{F}$. $\|\cdot\|_F$, $\|\cdot\|_2$, $\|\cdot\|_1$ denote the Frobenius norm, ℓ_2-norm and ℓ_1-norm, respectively. $\mathcal{R}(\boldsymbol{A})$ denotes the range space of matrix $\boldsymbol{A}$. Lowercase bold letters refer to vectors, e.g., $\boldsymbol{a}$, $\boldsymbol{b}$, and $\boldsymbol{c}$. These vectors are assumed to be column vectors, unless transposed with $(\cdot)^{\mathsf{T}}$. Additional notation will be introduced when needed.

Chapter 2

Joint Factorization and Latent Clustering

2.1 Introduction

Many signal processing and machine learning applications nowadays involve high-dimensional raw data that call for appropriate compaction before any further processing. Dimensionality reduction (DR) is often applied before clustering and classification, for example. Matrix and tensor factorization (or *factor analysis*) plays an important role in DR of matrix and tensor data, respectively. Traditional factorization models, such as singular value decomposition (SVD) and principal component analysis (PCA) have proven to be successful in analyzing high-dimensional data – e.g., PCA has been used for noise suppression, feature extraction, and subspace estimation in numerous applications. In recent years, alternative models such as non-negative matrix factorization (NMF) [84, 47] have drawn considerable interest (also as DR tools), because they tend to produce unique and interpretable reduced-dimension representations. In parallel, tensor factorization for multi-way data continues to gain popularity in the machine learning community, e.g., for social network mining and latent variable modeling [95, 2, 100, 36, 73].

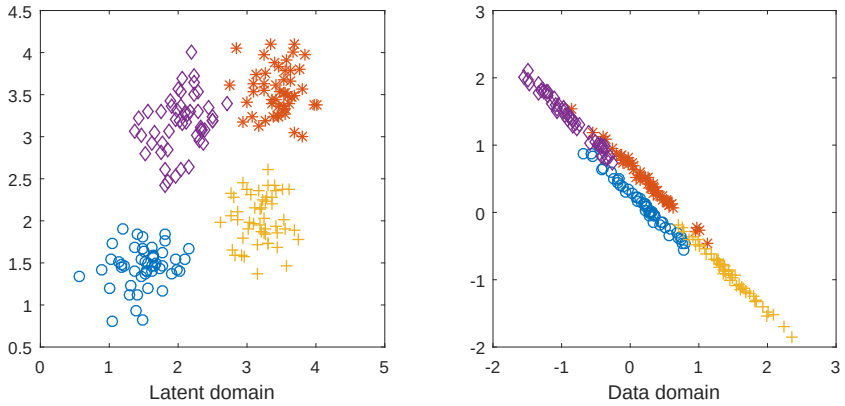


Figure 2.1: The effect of distance distortion introduced by $\boldsymbol{W}$. Left: $\boldsymbol{H}$ on a 2-dimensional plane. Right: $\boldsymbol{X} = \boldsymbol{W}\boldsymbol{H}$ on a 2-dimensional plane.

When performing DR or factor analysis, several critical questions frequently arise. First, what type of factor analysis should be considered for producing useful data representations for further processing, e.g., classification and clustering? Intuitively, if the data indeed coalesce in clusters in *some* low-dimensional representation, then DR should ideally map the input vectors to this particular representation – identifying the right subspace is not enough, for linear transformation can distort cluster structure (cf. Fig. 2.1). Therefore, if the data follow a factor analysis model that is unique (e.g., NMF is unique under certain conditions [39, 67]) *and* the data form clusters in the latent domain, then fitting that factor analysis model will reveal those clusters.

The second question is what kind of prior information can help get better latent representations of data? Using prior information is instrumental for fending against noise and modeling errors in practice, and thus is well-motivated. To this end, various constraints and regularization priors have been considered for matrix and tensor factorization, e.g., sparsity, smoothness, unimodality, total variation, and nonnegativity [101, 20, 136], to name a few.

In this chapter, we consider using a new type of prior information to assist factor analysis, namely, the latent cluster structure. This is motivated by the fact dimension-reduced data usually yield better performance in clustering tasks, which suggests that the cluster structure of the data

Dataset		Inter	Intra-1	Intra-2	Ratio
MNIST	Data	0.62	0.22	0.26	2.58
	Latent	0.88	0.30	0.34	2.75
Yale Face B	Data	0.38	0.18	0.17	2.17
	Latent	0.70	0.27	0.25	2.69

Table 2.1: Comparison of cosine distance in data and latent domain. Data samples are taken from two clusters for each dataset.

is more pronounced in *some* latent domain relative to the data domain. Some evidence can be seen in Tab. 2.1, where we compare the average data-domain and latent domain cosine distances[1] of data points from two clusters of image data from the Yale B [2] face image database and the MNIST handwritten digit image database[3], where the latent representations are produced via NMF. We see that the average latent domain distance between the data from two different clusters is significantly larger than the corresponding data domain distance in both cases. Moreover, we calculate the ratio between inter-cluster distance and average intra-cluster distance. This ratio can serve as an indicator of the quality of cluster structure: the higher this value, the better the cluster structure is. We observe that this ratio is higher in the latent domain than in the data domain for both datasets, as shown in Tab. 2.1. This observation motivates us to make use of such a property to enhance the performance of both factor analysis and clustering.

Using clustering to aid NMF-based DR was considered in [22, 23], where a distance graph of the data points was constructed and used as regularization for NMF – which essentially forces reduced-dimension representations to be far (close) in the latent domain, if the high-dimension vectors are far (resp. close) in the data domain. However, the data domain distance and the latent domain distance are not necessarily proportional to each other, as is evident from Fig. 2.1.

To see this clearly, consider a matrix factorization model $\boldsymbol{X} = \boldsymbol{W}\boldsymbol{H}$ where each column of $\boldsymbol{X}$ represents a data point, and the corresponding column of $\boldsymbol{H}$ is its latent representation.

[1]The cosine distance between two vectors $\mathbf{x}$ and $\mathbf{y}$ is defined as $\mathrm{d}(\mathbf{x}, \mathbf{y}) = 1 - \mathbf{x}^{\mathsf{T}}\mathbf{y}/(\|\mathbf{x}\|_2\|\mathbf{y}\|_2)$.

[2]Online available: http://web.mit.edu/emeyers/www/face_databases.html

[3]Online available: http://yann.lecun.com/exdb/mnist/

Consider the squared distance between the first two columns of $\boldsymbol{X}$, i.e., $||\boldsymbol{X}(:,1)-\boldsymbol{X}(:,2)||_2^2 = (\boldsymbol{H}(:,1)-\boldsymbol{H}(:,2))^\mathsf{T}\boldsymbol{W}^\mathsf{T}\boldsymbol{W}(\boldsymbol{H}(:,1)-\boldsymbol{H}(:,2))$, where : stands for all values of the respective argument. On the other hand, the distance of the latent representations of the first two columns of $\boldsymbol{X}$ is given by $\|\boldsymbol{H}(:,1)-\boldsymbol{H}(:,2)\|_2^2 = (\boldsymbol{H}(:,1)-\boldsymbol{H}(:,2))^\mathsf{T}(\boldsymbol{H}(:,1)-\boldsymbol{H}(:,2))$. Notice how the matrix $\boldsymbol{W}^\mathsf{T}\boldsymbol{W}$ weighs the latent domain distance to produce the data domain distance; also see Fig. 2.1 for illustration.

An earlier line of work [37, 117] that is not as well-known in the signal processing community considered latent domain clustering on the factor $\boldsymbol{H}$, while taking the other factor ($\boldsymbol{W}$) to be a semi-orthogonal matrix. However this cannot mitigate the weighting effect brought by $\boldsymbol{W}^\mathsf{T}\boldsymbol{W}$, since orthogonal projection cannot cancel the distorting effect brought in by the left factor $\boldsymbol{W}$.

Aiming to make full use of the latent cluster structure, we propose a novel *joint* factor analysis and latent clustering framework in this chapter. Our goal is to identify the *unique* latent representation of the data in some domain which is discriminative for clustering purposes, and also to use the latent cluster structure to help produce more accurate factorization results at the same time. We propose several pertinent problem formulations to exemplify the generality and flexibility of the proposed framework, and devise corresponding computational algorithms that build upon alternating optimization, i.e., alternating between factorization and clustering, until convergence. Identifiability of the latent factors plays an essential role in our approach, as it counteracts the distance distortion effect illustrated in Fig. 2.1. This is a key difference with relevant prior work such as [37, 117, 102].

We begin with K-means latent clustering using several identifiable factorization models of matrix and tensor data, namely, nonnegative matrix factorization, convex geometry (CG)-based matrix factorization model [44], and low-rank tensor factorization or *parallel factor analysis* (PARAFAC) [50]. Carefully designed optimization criteria are proposed, detailed algorithms are derived, and convergence properties are discussed. We next consider extension to joint factorization and subspace clustering, which is motivated by the popularity of subspace clustering

[118]. The proposed algorithms are carefully examined in judiciously designed simulations. Real experiments using document, handwritten digit, and three-way E-mail datasets are also used to showcase the effectiveness of the proposed approach.

2.2 Background

In this section, we briefly review the pertinent prior art in latent clustering and factor analysis.

2.2.1 Clustering and latent clustering

Let us begin with the classical K-means problem [92]: Given a data matrix $\boldsymbol{X} \in \mathbb{R}^{I \times J}$, we wish to group the columns of $\boldsymbol{X}$ into K clusters; i.e., we wish to assign the column index of $\boldsymbol{X}(:,j)$ to cluster $\mathcal{J}_k$, $k \in \{1,\ldots,K\}$, such that $\mathcal{J}_1 \cap \cdots \cap \mathcal{J}_K = \emptyset$, $\mathcal{J}_1 \cup \cdots \cup \mathcal{J}_K = \{1,\ldots,J\}$, and the sum of intra-cluster dispersions is minimized. The K-means problem can be written in optimization form as follows:

$$\begin{aligned} \min_{\boldsymbol{S} \in \mathbb{Z}^{K \times J}, \boldsymbol{M} \in \mathbb{R}^{I \times K}} \quad & \|\boldsymbol{X} - \boldsymbol{M}\boldsymbol{S}\|_F^2 \\ \text{s.t.} \quad & \|\boldsymbol{S}(:,j)\|_0 = 1,\ \boldsymbol{S}(i,j) \in \{0,1\}, \end{aligned} \tag{2.1}$$

where the matrix $\boldsymbol{S}$ is an assignment matrix, $\boldsymbol{S}(k,j) = 1$ means that $\boldsymbol{X}(:,j)$ belongs to the kth cluster, and $\boldsymbol{M}(:,k)$ denotes the centroid of the kth cluster. When I is very large and/or there are redundant features (e.g., highly correlated rows of $\boldsymbol{X}$), then it makes sense to perform DR either before or together with clustering. *Reduced K-means* (RKM) [37] is a notable joint DR and latent clustering method that is based on the following formulation:

$$\begin{aligned} \min_{\boldsymbol{S} \in \mathbb{Z}^{K \times J}, \boldsymbol{M} \in \mathbb{R}^{F \times K}, \boldsymbol{P} \in \mathbb{R}^{I \times F}} \quad & \|\boldsymbol{X} - \boldsymbol{P}\boldsymbol{M}\boldsymbol{S}\|_F^2 \\ \text{s.t.} \quad & \|\boldsymbol{S}(:,j)\|_0 = 1,\ \boldsymbol{S}(i,j) \in \{0,1\}, \\ & \boldsymbol{P}^\mathsf{T}\boldsymbol{P} = \boldsymbol{I}, \end{aligned} \tag{2.2}$$

where $\boldsymbol{P}$ is a tall semi-orthogonal matrix. Essentially, RKM aims at factoring $\boldsymbol{X}$ into $\boldsymbol{H} = \boldsymbol{M}\boldsymbol{S} \in \mathbb{R}^{I \times F}$ and $\boldsymbol{P}$, while enforcing a cluster structure on the columns of $\boldsymbol{H}$ – which is conceptually joint factorization and latent clustering. However, such a formulation loses generality since F (the rank of the model) cannot be smaller than K (the number of clusters); otherwise, the whole problem is ill-posed. Note that in practice, the number of clusters and the rank of $\boldsymbol{X}$ are not necessarily related; forcing a relationship between them (e.g., $F = K$) can be problematic from an application modeling perspective. In addition, the cluster structure is imposed as a straight jacket in latent space (no residual deviation, modeled by $\boldsymbol{R}$ in $\boldsymbol{P}(\boldsymbol{M}\boldsymbol{S}+\boldsymbol{R})$ is allowed in (2.2)), and this is too rigid in some applications.

Another notable formulation that is related to but distinct from RKM is *factorial K-means* (FKM) [117]:

$$\begin{aligned} \min_{\boldsymbol{S}\in\mathbb{Z}^{K\times J},\boldsymbol{M}\in\mathbb{R}^{F\times K},\boldsymbol{P}\in\mathbb{R}^{F\times I}} \quad & \|\boldsymbol{P}^{\mathsf{T}}\boldsymbol{X} - \boldsymbol{M}\boldsymbol{S}\|_F^2 \\ \text{s.t.} \quad & \|\boldsymbol{S}(:,j)\|_0 = 1,\ \boldsymbol{S}(i,j) \in \{0,1\}, \\ & \boldsymbol{P}^{\mathsf{T}}\boldsymbol{P} = \boldsymbol{I}. \end{aligned} \tag{2.3}$$

FKM seeks a 'good projection' of the data such that the projected data can be better clustered, and essentially performs clustering on $\boldsymbol{P}^{\mathsf{T}}\boldsymbol{W}\boldsymbol{H}$ if $\boldsymbol{X}$ admits a low-dimensional factorization as $\boldsymbol{X} = \boldsymbol{W}\boldsymbol{H}$. FKM does not force a coupling between K and F, and takes the latent modeling error into consideration. On the other hand, FKM ignores the part of $\boldsymbol{X}$ that is outside the chosen subspace, so it seeks *some* discriminative subspace where the projections cluster well, but ignores variation in the orthogonal complement of $\boldsymbol{P}$, since

$$\begin{aligned} \|\boldsymbol{X} - \boldsymbol{P}\boldsymbol{M}\boldsymbol{S}\|_F^2 &= \|[\boldsymbol{P}\ \boldsymbol{P}_\perp]^{\mathsf{T}}(\boldsymbol{X} - \boldsymbol{P}\boldsymbol{M}\boldsymbol{S})\|_F^2, \\ &= \|\boldsymbol{P}^{\mathsf{T}}\boldsymbol{X} - \boldsymbol{M}\boldsymbol{S}\|_F^2 + \left\|\boldsymbol{P}_\perp^{\mathsf{T}}\boldsymbol{X}\right\|_F^2, \end{aligned}$$

where $\boldsymbol{P}_\perp$ is a basis for the orthogonal complement of $\boldsymbol{P}$. So the cost of RKM equals the cost

function of FKM plus a penalty for the part of $\boldsymbol{X}$ in the orthogonal complement space. FKM was later adapted in [102], where a similar formulation was proposed to combine orthogonal factorization and sparse subspace clustering.

Seeking an orthogonal projection may not be helpful in terms of revealing the cluster structure of $\boldsymbol{H}$, since $\boldsymbol{P}^\mathsf{T}\boldsymbol{W}$ is still a general (oblique) linear transformation that acts on the columns of $\boldsymbol{H}$, potentially distorting cluster structure, even if $\boldsymbol{P}$ is a basis for $\boldsymbol{W}$.

2.2.2 Identifiable factorization models

Unlike RKM and FKM that seek an orthogonal factor or projection matrix $\boldsymbol{P}$, we propose to perform latent clustering using *identifiable* low-rank factorization models for matrices and tensors. The main difference in our approach is that we exploit identifiability of the latent factors to help unravel the hidden cluster structure, and in return improve factorization accuracy at the same time. This is sharply different from orthogonal projection models, such as RKM and FKM. Some important factorization models are succinctly reviewed next.

Nonnegative matrix factorization (NMF)

Low-rank matrix factorization models are in general non-unique, but nonnegativity can help in this regard [39], [67]. Loosely speaking, if $\boldsymbol{X} = \boldsymbol{W}\boldsymbol{H}$, where $\boldsymbol{W}$ and $\boldsymbol{H}^\mathsf{T}$ are (element-wise) nonnegative and have sufficiently sparse columns and sufficiently spread rows (over the nonnegative orthant), then any nonnegative $(\tilde{\boldsymbol{W}}, \tilde{\boldsymbol{H}})$ satisfying $\boldsymbol{X} = \tilde{\boldsymbol{W}}\tilde{\boldsymbol{H}}$ can be expressed as $\tilde{\boldsymbol{W}} = \boldsymbol{W}\boldsymbol{\Pi}\boldsymbol{D}$ and $\tilde{\boldsymbol{H}} = \boldsymbol{D}^{-1}\boldsymbol{\Pi}^\mathsf{T}\boldsymbol{H}$, where $\boldsymbol{D}$ is a full rank diagonal nonnegative matrix and $\boldsymbol{\Pi}$ is permutation matrix – i.e., $\boldsymbol{W}$ and $\boldsymbol{H}$ are *essentially* unique, or, identifiable up to a common column permutation and scaling-counterscaling. In practice, NMF is posed as a bilinear optimization problem,

$$\begin{aligned} &\min_{\boldsymbol{W},\boldsymbol{H}} \|\boldsymbol{X} - \boldsymbol{W}\boldsymbol{H}\|_F^2 \\ &\text{s.t. } \boldsymbol{W} \geq \boldsymbol{0},\ \boldsymbol{H} \geq \boldsymbol{0}. \end{aligned} \tag{2.4}$$

NMF is an NP-hard optimization problem. Nevertheless, many algorithms give satisfactory results in practice; see [65].

The plain NMF formulation (2.4) may yield arbitrary nonnegative scaling of the columns of $\boldsymbol{W}$ and the rows of $\boldsymbol{H}$ due tho the inherent scaling / counter-scaling ambiguity, which can distort distances. This can be avoided by adding a norm-balancing penalty

$$\begin{aligned} \min_{\boldsymbol{W},\boldsymbol{H}} \ & \|\boldsymbol{X} - \boldsymbol{W}\boldsymbol{H}\|_F^2 + \mu(\|\boldsymbol{W}\|_F^2 + \|\boldsymbol{H}\|_F^2) \\ \text{s.t. } & \boldsymbol{W} \geq \boldsymbol{0},\ \boldsymbol{H} \geq \boldsymbol{0}. \end{aligned} \tag{2.5}$$

It can be easily shown [101] that an optimum solution of (2.5) must satisfy $\|\boldsymbol{W}(:,f)\|_2 = \|\boldsymbol{H}(f,:)\|_2\,,\ \forall f$.

Volume minimization (VolMin)-based factorization

NMF has been widely used because it works well in many applications. If $\boldsymbol{W}$ or $\boldsymbol{H}$ is dense, however, then the NMF criterion in (2.4) cannot identify the true factors, which is a serious drawback for several applications. Recent research has shown that this challenge can be overcome using *Volume Minimization* (VolMin)-based structured matrix factorization methods [44, 43, 15, 25] In the VolMin model, the columns of $\boldsymbol{H}$ are assumed to live in the unit simplex, i.e., $\boldsymbol{1}^\mathsf{T}\boldsymbol{H}(:,j) = 1$ and $\boldsymbol{H}(:,j) \geq \boldsymbol{0}$ for all j, which is meaningful in applications like document clustering [43], hyperspectral imaging [91], and probability mixture models [60]. Under this structural constraint, $\boldsymbol{W}$ and $\boldsymbol{H}$ are sought via finding a minimum-volume simplex which encloses all the data columns $\boldsymbol{X}(:,j)$. In optimization form, the VolMin problem can be expressed as follows:

$$\begin{aligned} \min_{\boldsymbol{W}\in\mathbb{R}^{I\times F},\boldsymbol{H}\in\mathbb{R}^{F\times J}} \ & \text{vol}(\boldsymbol{W}) \\ \text{s.t.} \ & \boldsymbol{X} = \boldsymbol{W}\boldsymbol{H} \\ & \boldsymbol{H} \geq \boldsymbol{0},\ \boldsymbol{1}^\mathsf{T}\boldsymbol{H} = \boldsymbol{1}^\mathsf{T}, \end{aligned} \tag{2.6}$$

where $\mathrm{vol}(\cdot)$ measures the volume of the simplex spanned by the columns of $\boldsymbol{W}$ and is usually a function associated with the determinant of $\boldsymbol{W}$ or $\boldsymbol{W}^{\mathsf{T}}\boldsymbol{W}$ [44, 43, 15, 25]. Notably, it was proven in [44] that the optimal solution of (2.6) is $\boldsymbol{W}\boldsymbol{\Pi}$ and $\boldsymbol{\Pi}^{\mathsf{T}}\boldsymbol{H}$, where $\boldsymbol{\Pi}$ is again a permutation matrix, if the $\boldsymbol{H}(:,j)$'s are sufficient spread in the probability simplex and $\boldsymbol{W}$ is full column rank. Notice that $\boldsymbol{W}$ can be dense and even contain negative or complex elements, and still uniqueness can be guaranteed via VolMin.

Parallel factor analysis (PARAFAC)

For tensor data (i.e., data indexed by more than two indices) low-rank factorization is unique under mild conditions [77, 110]. Low-rank tensor decomposition is known as *parallel factor analysis* (PARAFAC) or canonical (polyadic) decomposition (CANDECOMP or CPD). For example, for a three-way tensor $\underline{\boldsymbol{X}}(i,j,l) = \sum_{f=1}^{F} \boldsymbol{A}(i,f)\boldsymbol{B}(j,f)\boldsymbol{C}(l,f)$, if

$$k_{\boldsymbol{A}} + k_{\boldsymbol{B}} + k_{\boldsymbol{C}} \geq 2F + 2, \tag{2.7}$$

where $k_{\boldsymbol{A}}$ is the Kruskal rank of $\boldsymbol{A}$, if $(\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{B}}, \tilde{\boldsymbol{C}})$ is such that

$$\underline{\boldsymbol{X}}(i,j,k) = \sum_{f=1}^{F} \tilde{\boldsymbol{A}}(i,f)\tilde{\boldsymbol{B}}(j,f)\tilde{\boldsymbol{C}}(k,f),$$

then $\boldsymbol{A} = \tilde{\boldsymbol{A}}\boldsymbol{\Pi}\boldsymbol{\Lambda}_1$, $\boldsymbol{B} = \tilde{\boldsymbol{B}}\boldsymbol{\Pi}\boldsymbol{\Lambda}_2$, $\boldsymbol{C} = \tilde{\boldsymbol{C}}\boldsymbol{\Pi}\boldsymbol{\Lambda}_3$, where $\boldsymbol{\Pi}$ is a permutation matrix and $\{\boldsymbol{\Lambda}_i\}_{i=1}^{3}$ are diagonal scaling matrices such that $\prod_{i=1}^{3} \boldsymbol{\Lambda}_i = \boldsymbol{I}$. Note that $\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{C}$ do not need to be full-column rank for ensuring identifiability.

Making use of the Khatri-Rao product, the tensor factorization problem can be written as

$$\min_{\boldsymbol{A},\boldsymbol{B},\boldsymbol{C}} \ \left\| \boldsymbol{X}_{(1)} - (\boldsymbol{C} \odot \boldsymbol{B})\boldsymbol{A}^{\mathsf{T}} \right\|_F^2, \tag{2.8}$$

where $\boldsymbol{X}_{(1)}$ is a matrix unfolding of the tensor $\underline{\boldsymbol{X}}$. There are three matrix unfoldings for this

three-way tensor that admit similar model expressions (because one can permute modes and $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$ accordingly)

$$\begin{aligned}\boldsymbol{X}_{(1)} &= [\text{vec}(\underline{\boldsymbol{X}}(1,:,:)), \cdots, \text{vec}(\underline{\boldsymbol{X}}(I,:,:))] \\ \boldsymbol{X}_{(2)} &= [\text{vec}(\underline{\boldsymbol{X}}(:,1,:)), \cdots, \text{vec}(\underline{\boldsymbol{X}}(:,J,:))] \\ \boldsymbol{X}_{(3)} &= [\text{vec}(\underline{\boldsymbol{X}}(:,:,1)), \cdots, \text{vec}(\underline{\boldsymbol{X}}(:,:,K))]\,.\end{aligned}$$

Like NMF, PARAFAC is NP-hard in general, but there exist many algorithms offering good performance and flexibility to incorporate constraints, e.g., [127], [66].

Our work brings these factor analysis models together with a variety of clustering tools ranging from K-means to K-subspace [114, 118] clustering to devise novel joint factorization and latent clustering formulations and companion algorithms that outperform the prior art – including two-step and joint approaches, such as RKM and FKM.

2.3 Proposed approach

2.3.1 Problem formulation

Suppose that $\boldsymbol{X} \approx \boldsymbol{W}\boldsymbol{H} \in \mathbb{R}^{I\times J}$, for some element-wise nonnegative $\boldsymbol{W} \in \mathbb{R}^{I\times F}$ and $\boldsymbol{H} \in \mathbb{R}^{F\times J}$, where the columns of $\boldsymbol{H}$ are clustered around K centroids. A natural problem formulation is then

$$\begin{aligned}\min_{\substack{\boldsymbol{W}\in\mathbb{R}^{I\times F},\boldsymbol{H}\in\mathbb{R}^{F\times J}\\ \boldsymbol{S}\in\mathbb{Z}^{K\times J},\boldsymbol{M}\in\mathbb{R}^{F\times K}}} \quad & \|\boldsymbol{X}-\boldsymbol{W}\boldsymbol{H}\|_F^2+\lambda\|\boldsymbol{H}-\boldsymbol{M}\boldsymbol{S}\|_F^2 \\ \text{s.t.} \quad & \boldsymbol{W}\geq\boldsymbol{0},\ \boldsymbol{H}\geq\boldsymbol{0}, \\ & \|\boldsymbol{S}(:,j)\|_0=1,\ \boldsymbol{S}(k,j)\in\{0,1\},\end{aligned} \tag{2.9}$$

where the second term is a K-means penalty that enforces the clustering prior on the columns of $\boldsymbol{H}$, and the tuning parameter $\lambda \geq 0$ balances data fidelity and the clustering prior. This formulation admits a *Maximum A Posteriori* (MAP) interpretation, if $\boldsymbol{X} = \boldsymbol{W}(\boldsymbol{M}\boldsymbol{S} + \boldsymbol{E}_2) + \boldsymbol{E}_1$, where the data-domain noise $\boldsymbol{E}_1$ and the latent-domain noise $\boldsymbol{E}_2$ are both drawn from an i.i.d. (independent and identically distributed) Gaussian distribution and independent of each other, with variance σ_1^2 and σ_2^2, respectively, and $\lambda = \frac{\sigma_1^2}{\sigma_2^2}$.

Assuming that $(\boldsymbol{W}, \boldsymbol{H})$ satisfy NMF identifiability conditions, and that $\boldsymbol{E}_1$ is negligible (i.e., NMF is exact), $\boldsymbol{H}$ will be exactly recovered and thus clustering will be successful as well. In practice of course the factorization model (DR) will be imperfect, and so the clustering penalty will help obtain a more accurate factorization, and in turn better clustering. Note that this approach decouples K and F, because it uses a clustering penalty instead of the hard constraint $\boldsymbol{H} = \boldsymbol{M}\boldsymbol{S}$ that RKM uses, which results in rank-deficiency when $F < K$.

Formulation (2.9) may seem intuitive and well-motivated from a MAP point of view, but there are some caveats. These are discussed next.

2.3.2 Design considerations

The first problem is scaling. In (2.9), the regularization on $\boldsymbol{H}$ implicitly favors a small-norm $\boldsymbol{H}$, since if $\boldsymbol{H}$ is small, then simply taking $\boldsymbol{M} = \boldsymbol{0}$ works. On the other hand, the first term is invariant with respect to scaling of $\boldsymbol{H}$, so long as this is compensated in $\boldsymbol{W}$. To prevent this, we introduce norm regularization for $\boldsymbol{W}$, resulting in

$$\begin{aligned} \min_{\boldsymbol{W},\boldsymbol{H},\boldsymbol{S},\boldsymbol{M}} \quad & \|\boldsymbol{X} - \boldsymbol{W}\boldsymbol{H}\|_F^2 + \lambda\|\boldsymbol{H} - \boldsymbol{M}\boldsymbol{S}\|_F^2 + \eta\|\boldsymbol{W}\|_F^2 \\ \text{s.t.} \quad & \boldsymbol{W} \geq \boldsymbol{0},\ \boldsymbol{H} \geq \boldsymbol{0}, \\ & \|\boldsymbol{S}(:,j)\|_0 = 1,\ \boldsymbol{S}(k,j) \in \{0,1\}. \end{aligned} \tag{2.10}$$

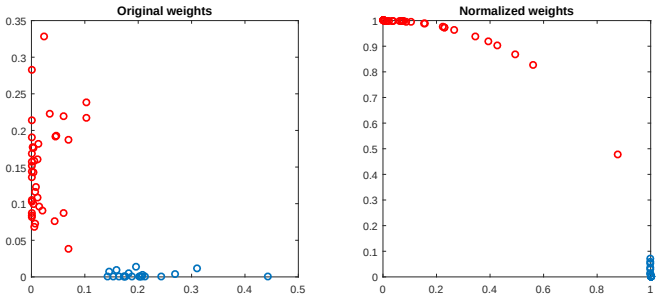


Figure 2.2: Normalization in the latent domain helps bringing data points together, creating tight cluster structures. Figure generated by taking a plain NMF on two clusters of documents from Reuters-21578 dataset. Left: 2-dimensional representations of documents; Right: the normalized representations.

Note that $\|\boldsymbol{W}\|_1$ can be used instead of $\|\boldsymbol{W}\|_F^2$ to encourage sparsity, if desired [4]

Another consideration is more subtle. In many applications, such as document clustering, it has been observed [111, 62] that the correlation coefficient or cosine similarity are more appropriate for clustering than Euclidean distance. We have observed that this is also true for latent clustering, which speaks for the need for normalization in the latent domain. Corroborating evidence is provided in Fig. 2.2, which shows the latent representations of two document clusters from the Reuters-25718 dataset. These representations were extracted by plain NMF using $F = 2$. In Fig. 2.2, the latent representations on the left are difficult to cluster, especially those close to the origin, but after projection onto the unit ℓ_2-norm ball (equivalent to using cosine similarity to cluster the points on the left) the cluster structure becomes more evident on the right.

If K-means is applied in the data domain, the cosine similarity metric can be incorporated easily: by normalizing the data columns using their ℓ_2-norms in advance, K-means is effectively using cosine dissimilarity as the distance measure. In our context, however, naive adoption of the cosine similarity for the clustering part can complicate things, since $\boldsymbol{H}$ changes in every

[4] In the (contrived) case where $\boldsymbol{H}$ is *exactly* equal to $\boldsymbol{MS}$, there is freedom to scale up $\boldsymbol{H}$ arbitrarily; but in practice we use formulation (2.11) instead of (2.10), and (2.11) is not subject to this issue.

iteration. To accommodate this, we reformulate the problem as follows.

$$\begin{aligned}\min_{\substack{\boldsymbol{W},\boldsymbol{H}\\ \boldsymbol{S},\boldsymbol{M},\{d_j\}_{j=1}^J}} \quad & \|\boldsymbol{X}-\boldsymbol{W}\boldsymbol{H}\boldsymbol{D}\|_F^2+\lambda\|\boldsymbol{H}-\boldsymbol{M}\boldsymbol{S}\|_F^2+\eta\|\boldsymbol{W}\|_F^2\\ \text{s.t. } & \mathbf{W}\geq\mathbf{0},\ \boldsymbol{H}\geq\mathbf{0},\ \|\boldsymbol{H}(:,j)\|_2=1,\ \forall j,\\ & \boldsymbol{D}=\mathrm{Diag}(d_1,\ldots,d_J),\\ & \|\boldsymbol{S}(:,j)\|_0=1,\ \boldsymbol{S}(k,j)\in\{0,1\}.\end{aligned} \tag{2.11}$$

Introducing the diagonal matrix $\boldsymbol{D}$ is crucial: It allows us to fix the columns of $\boldsymbol{H}$ onto the unit ℓ_2-norm ball without loss of generality of the factorization model.

The formulation in (2.11) can be generalized to tensor factorization models. Consider a three-way tensor $\underline{\boldsymbol{X}}\in\mathbb{R}^{I\times J\times L}$ with loading factors $\boldsymbol{A}\in\mathbb{R}^{I\times F}$, $\boldsymbol{B}\in\mathbb{R}^{J\times F}$, $\boldsymbol{C}\in\mathbb{R}^{L\times F}$. Assuming that *rows* of $\boldsymbol{A}$ can be clustered into K groups, the *joint tensor factorization and $\boldsymbol{A}$-mode latent clustering problem* can be formulated as

$$\begin{aligned}\min_{\substack{\boldsymbol{A},\boldsymbol{B},\boldsymbol{C}\\ \boldsymbol{S},\boldsymbol{M},\{d_\ell\}_{\ell=1}^L}} \quad & \left\|\boldsymbol{X}_{(1)}-(\boldsymbol{C}\odot\boldsymbol{B})(\boldsymbol{D}\boldsymbol{A})^{\mathsf{T}}\right\|_F^2+\lambda\|\boldsymbol{A}-\boldsymbol{S}\boldsymbol{M}\|_F^2\\ & \qquad +\eta\|\boldsymbol{B}\|_F^2+\eta\|\boldsymbol{C}\|_F^2\\ \text{s.t. } & \boldsymbol{A},\boldsymbol{B},\boldsymbol{C}\geq\mathbf{0},\ \|\boldsymbol{A}(\ell,:)\|_2=1,\ \forall\ell,\\ & \boldsymbol{D}=\mathrm{Diag}(d_1,\ldots,d_I),\\ & \|\boldsymbol{S}(i,:)\|_0=1,\ \boldsymbol{S}(i,k)\in\{0,1\},\ \forall i,k,\end{aligned} \tag{2.12}$$

where the regularization terms $\|\boldsymbol{B}\|_F^2$ and $\|\boldsymbol{C}\|_F^2$ are there to control scaling. If one wishes to perform latent clustering in more modes, then norm regularization can be replaced by K-means regularization for $\boldsymbol{B}$ and / or $\boldsymbol{C}$ modes as well. It is still important to have norm regularization for those modes that do not have K-means regularization.

An interesting point worth mentioning is that if one adopts VolMin as factorization criterion,

then introducing $\boldsymbol{D}$ is not necessary, since VolMin already confines $\boldsymbol{H}(:,j)$ on the unit-(ℓ_1-)norm ball. We also do not need to regularize with the norm of $\boldsymbol{W}$, since in this case the scaling of $\boldsymbol{W}$ cannot be arbitrary. This yields

$$\begin{aligned} \min_{\substack{\boldsymbol{W},\boldsymbol{H}\\ \boldsymbol{S},\boldsymbol{M}}} \ & \|\boldsymbol{X}-\boldsymbol{W}\boldsymbol{H}\|_F^2+\beta\cdot\mathrm{vol}(\boldsymbol{W})+\lambda\|\boldsymbol{H}-\boldsymbol{M}\boldsymbol{S}\|_F^2 \\ \text{s.t. } & \boldsymbol{H}\geq\boldsymbol{0},\ \boldsymbol{1}^\mathsf{T}\boldsymbol{H}=\boldsymbol{1}^\mathsf{T}, \\ & \|\boldsymbol{S}(:,j)\|_0=1,\ \boldsymbol{S}(k,j)\in\{0,1\}. \end{aligned} \tag{2.13}$$

2.4 Optimization algorithms

In this section, we provide algorithms for dealing with the various problems formulated in the previous section. The basic idea is alternating optimization – breaking the variables down to blocks and solving the partial problems one by one. Updating strategies and convergence properties are also discussed.

2.4.1 Joint NMF and K-means (JNKM)

We first consider (2.11) and (2.12). For ease of exposition, we use (2.11) as a working example. Generalization to Problem (2.12) is straightforward. Our basic strategy is to alternate between updating $\boldsymbol{W}$, $\boldsymbol{H}$, $\boldsymbol{S}$, $\boldsymbol{M}$, and $\{d_i\}_{i=1}^I$ one at a time, while fixing the others. For the subproblems w.r.t. $\boldsymbol{S}$ and $\boldsymbol{M}$, we propose to use the corresponding (alternating) steps of classical K-means [51]. The minimization w.r.t. $\boldsymbol{H}$ needs more effort, due to the unit norm and nonegativity constraints. Here, we propose to employ a variable-splitting strategy. Specifically, we consider

the following optimization surrogate:

$$\begin{aligned}\min_{\boldsymbol{W},\boldsymbol{H},\boldsymbol{Z},\boldsymbol{S},\boldsymbol{M},\{d_i\}_{i=1}^{I}} \quad & \|\boldsymbol{X}-\boldsymbol{W}\boldsymbol{H}\boldsymbol{D}\|_F^2+\lambda\|\boldsymbol{H}-\boldsymbol{M}\boldsymbol{S}\|_F^2 \\ & +\eta\|\boldsymbol{W}\|_F^2+\mu\|\boldsymbol{H}-\boldsymbol{Z}\|_F^2 \\ \text{s.t.} \quad & \boldsymbol{W}\geq\boldsymbol{0},\ \boldsymbol{H}\geq\boldsymbol{0},\ \|\boldsymbol{Z}(:,j)\|_2=1,\ \forall j, \\ & \boldsymbol{D}=\mathrm{Diag}(d_1,\ldots,d_J), \\ & \|\boldsymbol{S}(:,i)\|_0=1,\ \boldsymbol{S}(k,j)\in\{0,1\},\ \forall k,j,\end{aligned} \tag{2.14}$$

where $\mu \geq 0$ and $\boldsymbol{Z}$ is a slack variable. Note that $\boldsymbol{Z}$ is introduced to 'split' the effort of dealing with $\boldsymbol{H} \geq \boldsymbol{0}$ and $\|\boldsymbol{H}(:,j)\|_2 = 1$ in two different subproblems. Notice that when $\mu = +\infty$, (2.14) is equivalent to (2.11); in practice, a large μ can be employed to enforce $\boldsymbol{H} \approx \boldsymbol{Z}$.

Problem (2.14) can be handled as follows. First, $\boldsymbol{H}$ can be updated by solving

$$\begin{aligned}\boldsymbol{H} \leftarrow \arg\min_{\boldsymbol{H}\geq\boldsymbol{0}} \quad & \|\boldsymbol{X}-\boldsymbol{W}\boldsymbol{H}\boldsymbol{D}\|_F^2+\lambda\|\boldsymbol{H}-\boldsymbol{M}\boldsymbol{S}\|_F^2 \\ & +\mu\|\boldsymbol{H}-\boldsymbol{Z}\|_F^2,\end{aligned} \tag{2.15}$$

which can be easily converted to a nonnegative least squares (NLS) problem, and efficiently solved to optimality by many existing methods. Here, we employ an alternating direction method of multipliers (ADMM)-based [18] algorithm to solve Problem (2.15). The update of $\boldsymbol{W}$, i.e.,

$$\boldsymbol{W} \leftarrow \arg\min_{\boldsymbol{W}\geq\boldsymbol{0}} \ \|\boldsymbol{X}-\boldsymbol{W}\boldsymbol{H}\boldsymbol{D}\|_F^2+\eta\|\boldsymbol{W}\|_F^2, \tag{2.16}$$

is also an NLS problem. The subproblem w.r.t. d_j admits closed-form solution,

$$d_j \leftarrow \boldsymbol{b}_j^{\mathsf{T}}\boldsymbol{X}(:,j)/(\boldsymbol{b}_j^{\mathsf{T}}\boldsymbol{b}_j), \tag{2.17}$$

where $\boldsymbol{b}_j = \boldsymbol{W}\boldsymbol{H}(:,j)$. The update of $\boldsymbol{Z}(:,j)$ is also closed-form,

$$\boldsymbol{Z}(:,j) \leftarrow \frac{\boldsymbol{H}(:,j)}{\|\boldsymbol{H}(:,j)\|_2}. \tag{2.18}$$

The update of $\boldsymbol{M}$ comes from the K-means algorithm. Let $\mathcal{J}_k = \{j \mid \boldsymbol{S}(k,j) = 1\}$. Then

$$\boldsymbol{M}(:,k) \leftarrow \frac{\sum_{j\in\mathcal{J}_k} \boldsymbol{H}(:,j)}{|\mathcal{J}_k|}. \tag{2.19}$$

The update of $\boldsymbol{S}$ also comes from the K-means algorithm

$$\boldsymbol{S}(\ell,j) \leftarrow \begin{cases} 1, & \ell = \arg\min_k \ \|\boldsymbol{H}(:,j) - \boldsymbol{M}(:,k)\|_2 \\ 0, & \text{otherwise.} \end{cases} \tag{2.20}$$

The overall algorithm alternates between updates (2.15)-(2.20).

2.4.2 Joint NTF and K-means (JTKM)

As in the JNKM case, we employ variable splitting and introduce a $\boldsymbol{Z}$ variable to (2.12)

$$\begin{aligned} \min_{\substack{\boldsymbol{A},\boldsymbol{B},\boldsymbol{C} \\ \boldsymbol{S},\boldsymbol{M},\{d_\ell\}_{\ell=1}^L}} \ & \left\|\boldsymbol{X}_{(1)} - (\boldsymbol{C}\odot\boldsymbol{B})(\boldsymbol{D}\boldsymbol{A})^\mathsf{T}\right\|_F^2 + \lambda\|\boldsymbol{A} - \boldsymbol{S}\boldsymbol{M}\|_F^2 \\ & + \eta(\|\boldsymbol{B}\|_F^2 + \|\boldsymbol{C}\|_F^2) + \mu\|\boldsymbol{A} - \boldsymbol{Z}\|_F^2 \\ \text{s.t. } & \boldsymbol{A},\boldsymbol{B},\boldsymbol{C} \geq \boldsymbol{0},\ \|\boldsymbol{Z}(\ell,:)\|_2 = 1,\ \forall \ell, \\ & \boldsymbol{D} = \mathrm{Diag}(d_1,\ldots,d_I), \\ & \|\boldsymbol{S}(i,:)\|_0 = 1,\ \boldsymbol{S}(i,k) \in \{0,1\},\ \forall i,k, \end{aligned} \tag{2.21}$$

The algorithm for dealing with Problem (2.21) is similar to that of the NMF case. By treating $\boldsymbol{X}_{(1)}$ as $\boldsymbol{X}$, $(\boldsymbol{B}\odot\boldsymbol{C})$ as $\boldsymbol{W}$ and $\boldsymbol{A}^\mathsf{T}$ as $\boldsymbol{H}$, the updates of $\boldsymbol{A}$, $\boldsymbol{D}$, $\boldsymbol{Z}$, $\boldsymbol{S}$ and $\boldsymbol{M}$ are the same as those in the previous section. To update $\boldsymbol{B}$ and $\boldsymbol{C}$, we make use of the other two matrix

unfoldings

$$\boldsymbol{B} \leftarrow \arg\min_{\boldsymbol{B}\geq\boldsymbol{0}} \left\| \boldsymbol{X}_{(2)} - (\boldsymbol{C} \odot \boldsymbol{D}\boldsymbol{A})\boldsymbol{B}^{\mathsf{T}} \right\|_F^2 + \eta \left\| \boldsymbol{B} \right\|_F^2, \tag{2.22a}$$

$$\boldsymbol{C} \leftarrow \arg\min_{\boldsymbol{C}\geq\boldsymbol{0}} \left\| \boldsymbol{X}_{(3)} - (\boldsymbol{B} \odot \boldsymbol{D}\boldsymbol{A})\boldsymbol{C}^{\mathsf{T}} \right\|_F^2 + \eta \left\| \boldsymbol{C} \right\|_F^2. \tag{2.22b}$$

These are nonnegative linear least squares problems, and thus can be efficiently solved.

2.4.3 Joint VolMin and K-means (JVKM)

For VolMin-based factorization, one major difficulty is dealing with the volume measure $\mathrm{vol}(\boldsymbol{W})$, which is usually defined as $\mathrm{vol}(\boldsymbol{W}) = \det(\boldsymbol{W}^{\mathsf{T}}\boldsymbol{W})$ [15, 44, 43]. If clustering is the ultimate objective, however, we can employ more crude volume measures for the sake of computational simplicity. With this in mind, we propose to employ the following approximation of simplex volume [13]: $\mathrm{vol}(\boldsymbol{W}) \approx \sum_{f=1}^{F}\sum_{\ell>f}^{F}\|\boldsymbol{W}(:,f) - \boldsymbol{W}(:,\ell)\|_2^2 = \mathrm{Tr}(\boldsymbol{W}\mathbf{G}\boldsymbol{W}^{\mathsf{T}})$, where $\mathbf{G} = F\boldsymbol{I} - \mathbf{1}\mathbf{1}^{\mathsf{T}}$. The regularizer $\mathrm{Tr}(\boldsymbol{W}\mathbf{G}\boldsymbol{W}^{\mathsf{T}})$ measures the volume of the simplex that is spanned by the columns of $\boldsymbol{W}$ by simply measuring the distances between the vertices. This approximation is coarse, but reasonable. Hence, Problem (2.13) can be tackled using a four-block BCD, i.e.,

$$\boldsymbol{W} \leftarrow \arg\min_{\boldsymbol{W}} \quad \|\boldsymbol{X} - \boldsymbol{W}\boldsymbol{H}\|_F^2 + \beta\mathrm{Tr}(\boldsymbol{W}\mathbf{G}\boldsymbol{W}^{\mathsf{T}}), \tag{2.23a}$$

$$\boldsymbol{H} \leftarrow \arg\min_{\mathbf{1}^{\mathsf{T}}\boldsymbol{H}=\mathbf{1}^{\mathsf{T}},\boldsymbol{H}\geq\boldsymbol{0}} \quad \|\boldsymbol{X} - \boldsymbol{W}\boldsymbol{H}\|_F^2 + \lambda\|\boldsymbol{H} - \boldsymbol{M}\boldsymbol{S}\|_F^2 \tag{2.23b}$$

Note that Problem (2.23a) is a convex quadratic problem that has closed-form solution, and Problem (2.23b) is a simplex-constrained least squares problem that can be solved efficiently via many solvers. The updates for $\boldsymbol{M}$ and $\boldsymbol{S}$ are still given by (2.19) and (2.20).

2.5 Extension: Joint factor analysis and subspace clustering

In addition to considering Problems (2.11)-(2.13), we also consider their subspace clustering counterparts, i.e., with K-means penalties replaced by subspace clustering ones. Subspace clustering deals with data that come from a union of subspaces [52]. Specifically, consider $\boldsymbol{X}(:,j) \in \mathcal{R}(\boldsymbol{W}(:,\mathcal{F}_k))$, where $\mathcal{F}_k$ denotes an index set of a subset of $\boldsymbol{W}$'s columns, $\mathcal{F}_1 \cap \cdots \cap \mathcal{F}_K = \emptyset$ and $\mathcal{F}_1 \cup \cdots \cup \mathcal{F}_K = \{1,\ldots,K\}$. Also assume that $\mathcal{R}(\boldsymbol{W}(:,\mathcal{F}_1)) \cap \cdots \cap \mathcal{R}(\boldsymbol{W}(:,\mathcal{F}_K)) = \{\boldsymbol{0}\}$, which is the independent subspace model [118]. Then, it is evident that

$$\boldsymbol{H}(f,\mathcal{J}_k) = \boldsymbol{0},\ f \notin \mathcal{F}_k, \quad \boldsymbol{H}(\mathcal{F}_k,j) = \boldsymbol{0},\ j \notin \mathcal{J}_k,$$

where $\mathcal{J}_k$ denotes the set of indices of columns of $\boldsymbol{X}$ in cluster k, i.e. $\boldsymbol{X}(:,\mathcal{J}_k) \in \mathcal{R}(\boldsymbol{W}(:,\mathcal{F}_k))$.

Under this data structure, consider a simple example: if $I = J = F = 4$, $K = 2$, $\mathcal{F}_1 = \{1,2\}$ and $\mathcal{F}_2 = \{3,4\}$, then $\boldsymbol{H}$ is a block diagonal matrix where the nonzero diagonal blocks are 2×2. From this illustrative example, it is easy to see that the columns of $\boldsymbol{H}$ also come from different subspaces. The difference is that these subspaces that are spanned by $\boldsymbol{H}(:,\mathcal{J}_k)$ and $\boldsymbol{H}(:,\mathcal{J}_\ell)$ are not only independent, but also orthogonal to each other – which are much easier to distinguish. This suggests that performing subspace clustering on $\boldsymbol{H}$ is more appealing.

In [102], Patel *et al.* applied joint dimensionality reduction and subspace clustering on $\boldsymbol{P}^\mathsf{T}\boldsymbol{X}$ using a semi-orthogonal $\boldsymbol{P}$, which is basically the same idea as FKM, but using subspace clustering instead of K-means as in FKM. However, as we discussed before, when $\boldsymbol{W}$ and $\boldsymbol{H}$ are identifiable, taking advantage of the identifiability of the factors can further enhance the performance and should therefore be preferred in this case.

2.5.1 Formulation

Many subspace clustering formulations such as those in [89], [41] can be integrated into our framework, but we limit ourselves to simple K-subspace clustering, and assume that the number

of subspaces K and subspace dimensions $\{r_i\}_{i=1}^K$ are known, for simplicity. Taking joint NMF and K-subspace clustering as an example, we can express the problem as

$$\begin{aligned}\min_{\substack{\boldsymbol{W},\boldsymbol{H}\\ \boldsymbol{S},\{\boldsymbol{\theta}_j,\boldsymbol{\mu}_k,\boldsymbol{U}_k\}}} \quad & \|\boldsymbol{X}-\boldsymbol{W}\boldsymbol{H}\|_F^2+\eta\|\boldsymbol{W}\|_F^2 \\ & +\lambda\sum_{j=1}^{J}\sum_{k=1}^{K}\boldsymbol{S}(k,j)\,\|\boldsymbol{H}(:,j)-\boldsymbol{\mu}_k-\boldsymbol{U}_k\boldsymbol{\theta}_j\|_F^2 \\ \text{s.t. } & \mathbf{W}\geq\mathbf{0},\ \boldsymbol{H}\geq\mathbf{0}, \\ & \boldsymbol{U}_k^{\mathsf{T}}\boldsymbol{U}_k=\boldsymbol{I},\ \forall\, k, \\ & \|\boldsymbol{S}(:,j)\|_0=1,\ \boldsymbol{S}(k,j)\in\{0,1\},\end{aligned} \tag{2.24}$$

where $\boldsymbol{S}$ is again a cluster assignment variable, $\boldsymbol{U}_k$ denotes an orthogonal basis of the columns of $\boldsymbol{H}$ which lie in the kth cluster, $\boldsymbol{\mu}_k$ is a mean vector of the kth cluster, and $\boldsymbol{\theta}_j \in \mathbb{R}^{r_k\times 1}$ is the coordinates of the jth vector in the subspace. As in the VolMin case, subspace clustering is also insensitive to the scaling of $\boldsymbol{H}(:,j)$, since the metric for measuring distance to a cluster centroid is the distance to a subspace. Therefore, the constraints that were added for normalizing $\boldsymbol{H}(:,j)$ can be removed.

2.5.2 Joint NMF and K-subspace (JNKS) algorithm

Similar to JNKM, the updates of $\boldsymbol{W}, \boldsymbol{H}$ in (2.24) are both NLS problems, and can be easily handled. To update the subspace and the coefficients, we need to solve a K-subspace clustering problem [118]

$$\begin{aligned}\min_{\{\boldsymbol{U}_k\},\{\boldsymbol{\mu}_k\},\{\boldsymbol{\theta}_j\}} \ & \sum_{j=1}^{J}\sum_{k=1}^{K}\boldsymbol{S}(k,j)\,\|\boldsymbol{H}(:,j)-\boldsymbol{\mu}_k-\boldsymbol{U}_k\boldsymbol{\theta}_j\|_F^2 \\ \text{s.t. } & \boldsymbol{U}_k^{\mathsf{T}}\boldsymbol{U}_k=\boldsymbol{I},\ \forall\, k.\end{aligned} \tag{2.25}$$

Let $\boldsymbol{H}(:,\mathcal{F}_k)$ denote the data points in cluster k, and $\boldsymbol{\Theta}_k := \{\boldsymbol{\theta}_j | j \in \mathcal{F}_k\}$. We can equivalently write (2.25) as

$$\begin{aligned} \min_{\{\boldsymbol{U}_k\},\{\boldsymbol{\mu}_k\},\{\boldsymbol{\Theta}_k\}} & \sum_{k=1}^{K} \left\| \boldsymbol{H}(:,\mathcal{F}_k) - \boldsymbol{\mu}_k \mathbf{1}^\mathsf{T} - \boldsymbol{U}_k \boldsymbol{\Theta}_k \right\|_F^2 \\ & \text{s.t. } \boldsymbol{U}_k^\mathsf{T} \boldsymbol{U}_k = \boldsymbol{I},\ \forall\, k. \end{aligned} \tag{2.26}$$

It can be readily seen that the update of each subspace is independent of the others. For cluster k, we first remove its center, and then take a SVD,

$$\boldsymbol{\mu}_k \leftarrow \frac{\boldsymbol{H}(:,\mathcal{F}_k)\mathbf{1}}{\|\boldsymbol{S}(k,:)\|_0}, \tag{2.27a}$$

$$[\widehat{\boldsymbol{U}}, \widehat{\boldsymbol{\Sigma}}, \widehat{\boldsymbol{V}}^\mathsf{T}] \leftarrow \text{svd}(\boldsymbol{H}(:,\mathcal{F}_k) - \boldsymbol{\mu}_k \mathbf{1}^\mathsf{T}), \tag{2.27b}$$

$$\boldsymbol{U}_k \leftarrow \widehat{\boldsymbol{U}}(:, 1:r_k), \tag{2.27c}$$

$$\boldsymbol{\Theta}_k \leftarrow \widehat{\boldsymbol{\Sigma}}(1:r_k, 1:r_k)\widehat{\boldsymbol{V}}(:, 1:r_k)^\mathsf{T}. \tag{2.27d}$$

To update the subspace assignment $\boldsymbol{S}$, we solve

$$\begin{aligned} \min_{\boldsymbol{S}} & \sum_{j=1}^{J} \sum_{k=1}^{K} \boldsymbol{S}(k,j) \left\| \boldsymbol{H}(:,j) - \boldsymbol{\mu}_k - \boldsymbol{U}_k \boldsymbol{\theta}_j \right\|_F^2 \\ & \text{s.t. } \|\boldsymbol{S}(:,j)\|_0 = 1,\ \boldsymbol{S}(k,j) \in \{0,1\}. \end{aligned} \tag{2.28}$$

With $\text{dist}(j,k) := \left\| (\boldsymbol{I} - \boldsymbol{U}_k \boldsymbol{U}_k^\mathsf{T})(\boldsymbol{H}(:,j) - \boldsymbol{\mu}_k) \right\|_2$, the update of $\boldsymbol{S}$ is given by [118]

$$\boldsymbol{S}(\ell, j) \leftarrow \begin{cases} 1 & \ell = \text{argmin}_k\ \text{dist}(j,k) \\ 0 & \text{otherwise.} \end{cases} \tag{2.29}$$

Remark 1. *Subspace clustering is considered suitable for image processing, where occlusions or outliers are often spotted and special care needs to be taken for dealing with the occlusions [123]. Since occlusions are sparsely distributed over the data matrix (e.g., image), one way to*

handle this situation is to change the cost function of (2.24) *as follows:*

$$\begin{aligned}\min_{\substack{\boldsymbol{W},\boldsymbol{H}\\ \boldsymbol{S},\{\boldsymbol{\theta}_j,\boldsymbol{\mu}_k,\boldsymbol{U}_k\}}} \quad & \|\boldsymbol{X}-\boldsymbol{W}\boldsymbol{H}\|_1+\eta\|\boldsymbol{W}\|_F^2 \\ & +\lambda\sum_{j=1}^{J}\sum_{k=1}^{K}\boldsymbol{S}(k,j)\,\|\boldsymbol{H}(:,j)-\boldsymbol{\mu}_k-\boldsymbol{U}_k\boldsymbol{\theta}_j\|_2^2\,.\end{aligned} \tag{2.30}$$

As ℓ_1-norm based fitting is known to be robust to sparse outliers, the above is expected to exhibit better occlusion-robustness than using the Frobenius norm. The price to pay is that ℓ_1 norm is non-smooth and may require more effort for optimization.

2.6 Convergence and complexity

2.6.1 Convergence properties

The proposed algorithms, i.e., JNKM, JVKM, JTKM, and their subspace clustering counterparts, share some similar traits. Specifically, all algorithms solve the conditional updates (block minimization problems) optimally. From this it follows that

Property 1. *JNKM, JTKM, and JVKM, yield a non-increasing cost sequence in terms of their respective cost functions in* (2.14)*,* (2.21)*, and* (2.13)*, respectively. The same property is true for their subspace clustering counterparts.*

Monotonicity is important in practice – it ensures that an algorithm makes progress in every iteration towards the corresponding design objective. In addition, it leads to convergence of the cost sequence when the cost function is bounded from below (as in our case), and such convergence can be used for setting up stopping criteria in practice.

In terms of convergence of the iterates (the sequence of 'interim' solutions), when using a cyclical block updating strategy all algorithms fall under the Gauss-Seidel block coordinate descent (BCD) framework, however related convergence results [14] cannot be applied because

some blocks involve *nonconvex* constraints. If one uses a more expensive (non-cyclical) update schedule, then the following holds.

Property 2. *If the blocks are updated following the maximum block improvement (MBI) strategy (also known as the Gauss-Southwell rule), then every limit point of the solution sequence produced by JNKM, JTKM, and JVKM is a block-wise minimum of* (2.14)*,* (2.21)*, and* (2.13)*, respectively. A block-wise minimum is a point where it is not possible to reduce the cost by changing any single block while keeping the others fixed.*

The MBI update rule [28] is similar to BCD, but it does not update the blocks cyclically. Instead, it tries updating each block in each iteration, but actually updates only the block that brings the biggest cost reduction. The MBI result can be applied here since we solve each block subproblem to optimality. The drawback is that MBI is a lot more expensive per iteration. If one is interested in obtaining a converging subsequence, then a practical approach is to start with cyclical updates, and then only use MBI towards the end for 'last mile' refinement. We use Gauss-Seidel in our simulations, because it is much faster than MBI.

2.6.2 Complexity

Most block updates of the proposed algorithmic framework are identical or similar to the corresponding factorization models and clustering algorithms. Therefore, the complexity of the proposed joint optimization algorithms is no greater than performing factorization and clustering sequentially in a per-iteration complexity sense. For example, in the JNKM algorithm, the update of W is essentially a regualrized nonnegativity-constrained least squares. Using ADMM to solve this problem has per iteration complexity of $\mathcal{O}(FIJ)$ flops. The $\boldsymbol{S}$-subproblem and the $\boldsymbol{D}$-subproblem are from the Lloyd update of K-means, and have complexity of $\mathcal{O}(FKJ)$ and $\mathcal{O}(FJ)$ flops, respectively. Introducing $\boldsymbol{Z}$ and $\boldsymbol{D}$ to control scaling ambiguity is a unique feature of our proposed algorithm which enhances the performance greatly as will be shown. However, the updates of $\boldsymbol{Z}$ and $\boldsymbol{D}$ have closed-form solutions and thus the complexities are very light.

For the other combinations of factorization and clustering, the complexity can be analyzed in a similar fashion.

2.7 Synthetic data study

In this section, we use synthetic data to explore scenarios where the proposed algorithms show promising performance relative to the prior art. We will consider real datasets that are commonly used as benchmarks in the next section.

Our algorithms simultaneously *estimate* the latent factors and *cluster*, so we need a metric to assess estimation accuracy, and another for clustering accuracy. For the latter, we use the ratio of correctly clustered points over the total number of points (in $[0, 1]$, the higher the better) [23, 22]. Taking into account the inherent column permutation and scaling indeterminacy in estimating the latent factor matrices, estimation accuracy is measured via the following *matched* mean-square-error measure (MSE, for short)

$$\text{MSE} = \min_{\substack{\boldsymbol{\pi}\in\Pi,\\ c_1,\ldots,c_F\in\{\pm 1\}}} \frac{1}{F}\sum_{f=1}^{F}\left\| \frac{\boldsymbol{W}(:,f)}{\|\boldsymbol{W}(:,f)\|_2} - c_f \frac{\hat{\boldsymbol{W}}(:,\pi_f)}{\|\hat{\boldsymbol{W}}(:,\pi_f)\|_2}\right\|_2^2,$$

where $\boldsymbol{W}$ and $\hat{\boldsymbol{W}}$ represents the ground truth factor and the estimated factor, respectively, Π is the set of all permutations of the set $\{1, \cdots, F\}$, $\boldsymbol{\pi} = [\pi_1, \cdots, \pi_F]^\mathsf{T}$ is there to resolve the permutation ambiguity, and c_f to resolve the sign ambiguity when there is no nonnegativity constraint.

2.7.1 Joint Matrix Factorization and Latent K-means

We generate synthetic data according to

$$\boldsymbol{X} = \boldsymbol{W}(\boldsymbol{M}\boldsymbol{S} + \boldsymbol{E}_2) + \boldsymbol{E}_1, \tag{2.31}$$

$$= \boldsymbol{W}\boldsymbol{H} + \boldsymbol{E}_1,$$

where $\boldsymbol{X}$ is the data matrix, $\boldsymbol{W}$ is the basis matrix, $\boldsymbol{H} = \boldsymbol{M}\boldsymbol{S} + \boldsymbol{E}_2$ is the factor where the clustering structure lies in, $\boldsymbol{M}$ and $\boldsymbol{S}$ are the centroid matrix and the assignment matrix, respectively, and $\boldsymbol{E}_i$ for $i = 2, 1$ denote the modeling errors and the measurement noise, respectively. We define the Signal to Noise Ratio (SNR) in the data and latent space, respectively, as $\text{SNR}_1 = \frac{\|\boldsymbol{W}\boldsymbol{H}\|_F^2}{\|\boldsymbol{E}_1\|_F^2}$ and $\text{SNR}_2 = \frac{\|\boldsymbol{M}\boldsymbol{S}\|_F^2}{\|\boldsymbol{E}_2\|_F^2}$. SNR_1 is the 'conventional' SNR that characterizes the data corruption level, and SNR_2 is for characterizing the latent domain modeling errors. All the simulation results are obtained by averaging 100 Monte Carlo trials.

We employ several clustering and factorization algorithms as baselines, namely, the original K-means, the reduced K-means (RKM), the factorial K-means algorithm (FKM), and a simple two-stage combination of nonnegative matrix factorization (NMF) and K-means (NMF-KM).

We first test the algorithms under an NMF model. Specifically, $\boldsymbol{W} \in \mathbb{R}^{I \times F}$ is drawn from i.i.d. standard Gaussian distribution, with all negative entries set to zero. $\boldsymbol{H}$ is generated with the following steps:

1. Generate $\boldsymbol{M} \in \mathbb{R}^{F \times K}$ by setting the first F columns as unit vectors (to ensure identifiability of the NMF model), i.e. $\boldsymbol{M}(:, 1:F) = \boldsymbol{I}$, and entries in the remaining $K - F$ columns are randomly generated from an i.i.d. uniform distribution in $[0, 1]$; set $\boldsymbol{S} \in \mathbb{R}^{K \times J}$ as $\boldsymbol{S} = [\boldsymbol{I}, \boldsymbol{I}, \cdots, \boldsymbol{I}]$;

2. Draw $\boldsymbol{E}_2$ from an i.i.d. standard Gaussian distribution, set $\boldsymbol{H} \leftarrow \boldsymbol{M}\boldsymbol{S} + \boldsymbol{E}_2$ and perform the following steps

$$\boldsymbol{H} \leftarrow (\boldsymbol{H})_+, \tag{2.32a}$$

$$\boldsymbol{E}_2 \leftarrow \boldsymbol{H} - \boldsymbol{M}\boldsymbol{S}, \tag{2.32b}$$

$$\boldsymbol{E}_2 \leftarrow \gamma \boldsymbol{E}_2, \tag{2.32c}$$

$$\boldsymbol{H} \leftarrow \boldsymbol{M}\boldsymbol{S} + \boldsymbol{E}_2, \tag{2.32d}$$

where γ in (2.32c) is a scaling constant determined by the desired SNR_2, and $(\cdot)_+$ takes the nonnegative part of its argument. We may need to repeat the above steps (2.32a)$\sim$(2.32d) several times, till we get a nonnegative $\boldsymbol{H}$ with desired SNR_2 (in our experience, usually one repetition suffices).

We then multiply $\boldsymbol{W}$ and $\boldsymbol{H}$ and add $\boldsymbol{E}_1$, where $\boldsymbol{E}_1$ is drawn from an i.i.d. standard Gaussian distribution and scaled for the desired SNR_1. With this process, $\boldsymbol{W}$ and $\boldsymbol{H}$ are sparse and identifiable (when $\boldsymbol{E}_1 = \boldsymbol{0}$) with very high probability [39, 67]. Finally, we replace 3% of the columns of $\boldsymbol{X}$ with all-ones vectors that act as outliers, which are common in practice.

Table 2.2 presents the clustering accuracies of various algorithms using $I = 50$, $J = 1000$, $F = 7$, and $K = 10$. The MSEs of the estimated $\hat{\boldsymbol{W}}$ of the factorization-based algorithms are also presented. We set the parameters of the JNKM method to be $\lambda = 1$, $\mu = 100$ and $\eta = 10^{-1}$. Here, SNR_1 is fixed to be 15 dB and SNR_2 varies from 3 dB to 18 dB. The JNKM is initialized with an NMF solution [74], and the JVKM is initialized with the SISAL [15] algorithm. We see that, for all the SNR_2's under test, the proposed JNKM yields the best clustering accuracies. RKM and FKM give poor clustering results since they cannot resolve the distortion brought by $\boldsymbol{W}$, as we discussed before. Our proposed method works better than NMF-KM, since JNKM estimates $\boldsymbol{H}$ more accurately (cf. the MSEs of the estimated factor $\boldsymbol{W}$) – by making use of the cluster structure on $\boldsymbol{H}$ as prior information.

Note that in order to make the data fit the VolMin model, we normalized the columns of $\boldsymbol{X}$ to unit ℓ_1-norm as suggested in [48]. Due to noise, such normalization cannot ensure that the data follow the VolMin model exactly, however we observe that the proposed JVKM formulation still performs well in this case.

To better understand the reason why our method performs well, we present an illustrative example where the latent factor $\boldsymbol{H}$ lies in a two-dimensional subspace (so that it can be visualized),

Table 2.2: Clustering and factorization accuracy for identifiable NMF vs. SNR_2, for $I = 50, J = 1000, F = 7, K = 10$, $\text{SNR}_1 = 15$ dB.

SNR_2 [dB]		3	6	9	12	15	18
AC[%]	KM	77.43	81.5	82.9	81.47	82.68	84.5
	RKM	77.51	76.62	73.71	72.43	71.35	71.63
	FKM	15.12	15.68	16.6	17.14	37.5	59.74
	JNKM	**88.1**	**95.12**	**96.51**	**96.13**	**96.43**	**95.65**
	JVKM	75.84	83.87	87.87	89.96	90.27	89.36
	NMF-KM	84.72	86.62	88.96	90.95	90.87	92.34
MSE[dB]	JNKM	**-28.09**	**-27.82**	**-27.54**	**-26.59**	**-26.91**	**-26.26**
	JVKM	-16.41	-16.98	-16.37	-15.61	-15.19	-14.9
	NMF-KM	-27.09	-26.75	-26.7	-25.58	-26.05	-25.31
TIME[s]	KM	**0.14**	**0.05**	**0.05**	**0.07**	**0.06**	**0.06**
	RKM	0.18	0.13	0.16	0.17	0.17	0.18
	FKM	1.45	0.59	0.64	0.79	0.82	0.56
	JNKM	3.37	2.99	3.13	3.22	3.06	3.36
	JVKM	5.7	5.06	4.73	4.53	4.42	4.45
	NMF-KM	0.76	0.68	0.73	0.83	0.75	0.86

Table 2.3: Clustering and factorization accuracy for identifiable NMF vs. SNR_1, for $I = 50, J = 1000, F = 7, K = 10$, $\text{SNR}_2 = 10$ dB.

SNR_1 [dB]		5	10	15	20	25	30
AC[%]	KM	78.65	77.89	82.89	84.53	88.43	86.97
	RKM	79.54	72.84	72.87	71.15	71.37	72.06
	FKM	17.91	17.3	16.76	16.68	16.44	16.51
	JNKM	**93.28**	**94.69**	**95.73**	**96.33**	**96.43**	**96.04**
	JVKM	71.78	82.74	87.43	91.68	92.43	93.17
	NMF-KM	84.95	86.74	89.57	90.87	90.66	91.61
MSE[dB]	JNKM	**-18.03**	**-23.95**	**-26.71**	-27.17	-27.09	-26.19
	JVKM	-4.66	-12.19	-15.96	-20.21	-25.14	**-31.05**
	NMF-KM	-17.63	-23.36	-26.29	**-27.48**	**-27.76**	-27.13
TIME[s]	KM	**0.08**	**0.08**	**0.05**	**0.05**	**0.04**	**0.04**
	RKM	0.18	0.18	0.16	0.14	0.11	0.11
	FKM	0.68	0.7	0.66	0.69	0.61	0.67
	JNKM	3.1	3.29	3	3.13	2.99	2.92
	JVKM	3.21	3.83	4.57	5.12	5.01	4.94
	NMF-KM	0.74	0.75	0.7	0.66	0.64	0.62

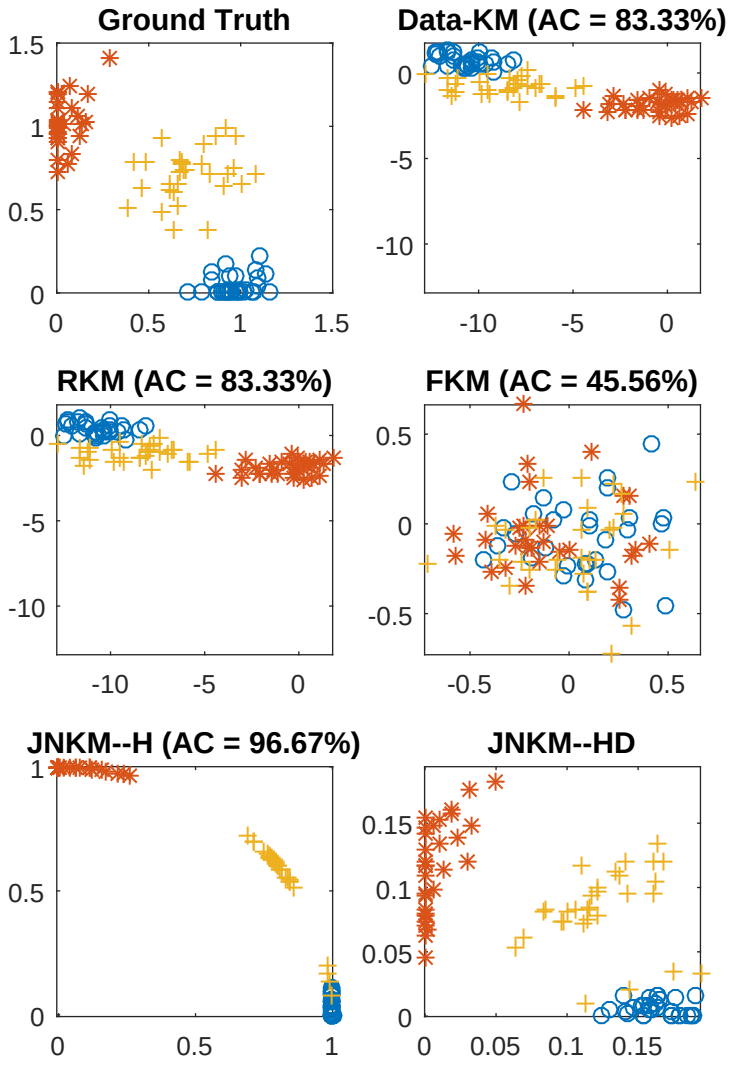


Figure 2.3: Illustration of how linear transformation obscures the latent cluster structure, and how identifiable models can recover this cluster structure. Top left: true latent factor $\boldsymbol{H}$; Top right: data domain $\boldsymbol{X} = \boldsymbol{W}\boldsymbol{H} + \boldsymbol{E}_1$, visualized using SVD (two principal components); Middle left: projected data found by RKM, $\boldsymbol{P}^\mathsf{T}\boldsymbol{X}$; Middle right: projected data found by FKM, $\boldsymbol{P}^\mathsf{T}\boldsymbol{X}$; Bottom left: $\boldsymbol{H}$ found by JNKM; Bottom right: $\boldsymbol{H}\boldsymbol{D}$ found by JNKM. In the top right subfigure, the clustering accuracy of running K-means directly on the data is shown; for other figures, the clustering accuracy given by corresponding method is shown.

Table 2.4: Simulation comparison of the clustering methods, identifiable NMF model. $I = 50$, $F = 7$, $J = 100K$.

K		5	6	7	8	9	10	11
AC[%]	RKM	79.97	78.57	76.6	76.16	75.48	75.22	75.54
	FKM	47.75	34.01	27.32	21.96	18.45	16.55	15.14
	JNKM	**97.6**	**97.5**	**97.43**	**97.37**	**96.88**	**96.63**	**95.48**
MSE[dB]	JNKM	-4.7	-7.52	-25.08	-24.77	-24.3	-24.17	-23.81

and has a clear cluster structure. The basis $\boldsymbol{W}$ is an 8×2 matrix. The factor are generated such that the NMF model is identifiable. Fig. 2.3 shows the true latent factors, together with those found by various methods. Clearly, $\boldsymbol{W}$ brings some distance distortion to the cluster structure in $\boldsymbol{H}$ (cf. top right subfigure). We see from this example that if the factorization model is identifiable, using the proposed approach helps greatly in removing the distance distortion brought by $\boldsymbol{W}$, as indicated by the last row of Fig. 2.3. On the other hand, the other semi-orthogonal projection-based algorithms do not have this salient feature.

Table 2.3 presents the results under various SNR_1's. Here, we fix $\text{SNR}_2 = 10$ dB, and the other settings are the same as in the previous simulation. We see that the clustering accuracies are not so sensitive to SNR_1, and the proposed JNKM outperforms other methods in AC and MSE in most of the cases. Table 2.4 presents the ACs and MSEs for fixed rank $F = 7$ as the number of clusters K varies from 5 to 11. Here $I = 50$, $J = 100K$, $\text{SNR}_1 = 6$dB, and $\text{SNR}_2 = 8$dB. We observe that the performance of all methods degrades when we add more clusters, which is expected. However, RKM and FKM suffer more than the proposed method.

In Fig. 2.4, we show the effect of changing λ in the JNKM formulation. We are particularly interested in this parameter since it plays an essential role in balancing the data fidelity and prior information. On the other hand, the parameter μ for enforcing $\boldsymbol{Z}$ to be close to $\boldsymbol{H}$ can be set to a large number, e.g., 1000, and η for balancing the scaling of the factors can usually be set to a small number – and the algorithm is not sensitive to these two according to our experience. Here, the setting is $I = 10,\ J = 100,\ F = 2$, and the number of clusters is $K = 2$. The SNRs

are set to $\text{SNR}_1 = 5\text{dB}$ and $\text{SNR}_2 = 30\text{dB}$. From Fig. 2.4, we see that both the MSE and AC performance of JNKM is reasonably good for all the λ's under test, although there does exist a certain λ giving the best performance ($\lambda = 3000$ in this case).

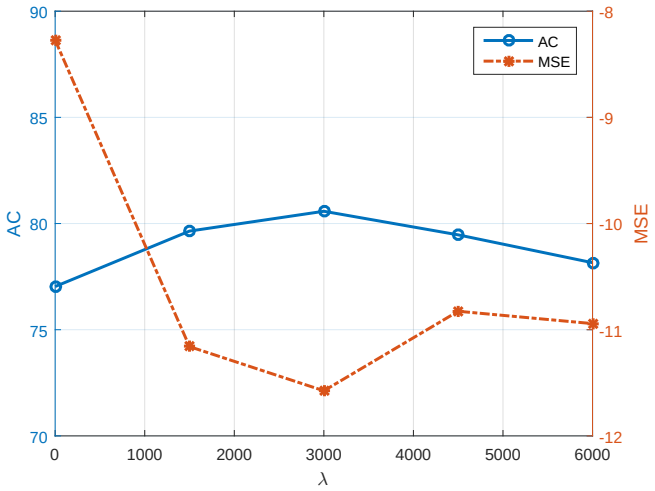


Figure 2.4: AC and MSE versus parameter λ

So far we have been working with sparse nonnegative factors. Let us consider a general $\boldsymbol{W}$ and a $\boldsymbol{H}$ with columns in a simplex, i.e. $\mathbf{1}^\mathsf{T}\boldsymbol{H} = \mathbf{1}^\mathsf{T},\ \boldsymbol{H} \geq \mathbf{0}$, which finds various applications in machine learning, e.g., document and hyperspectral pixel clustering / classification [48, 47]. We generate $\boldsymbol{H}$ using the same steps as in the previous simulations with the centroid matrix $\boldsymbol{M}(:,k)$'s being generated by putting a cluster near each unit vector, and several centroids randomly; under such setting, the identifiability conditions of the VolMin model are likely to hold [44]. Note that the entries of $\boldsymbol{W}$ are simply drawn from a zero-mean unit-variance i.i.d. Gaussian distribution, which means that $\boldsymbol{W}$ is a dense matrix and the identifiability of NMF does not hold – which differs from the previous simulations. Table 2.5 presents the results. We see that JNKM works worse relative to the previous simulation, since the generative model is not identifiable via NMF. However, JVKM works quite well since the VolMin identifiability holds regardless of the structure of $\boldsymbol{W}$, so long as it is full column rank. This is also reflected in the

Table 2.5: Clustering and factorization accuracy for identifiable VolMin vs. SNR_2, for $I = 50, J = 1000, F = 7, K = 10$.

SNR_2 [dB]		3	6	9	12	15	18
AC[%]	KM	61.01	73.82	74.59	73.53	74.66	75.04
	RKM	62.48	75.09	78.09	76.8	74.75	76.97
	FKM	14.28	14.47	14.66	14.98	15	15.28
	JNKM	58.46	75.68	85.52	89.74	91.43	92.56
	JVKM	**67.9**	**87.98**	**95.16**	**96.54**	**96.94**	**97.29**
	VolMin-KM	64.85	81.7	87.31	89.31	88.59	89.78
MSE[dB]	JNKM	-0.11	0.03	0.22	0.21	0.34	0.62
	JVKM	**-21.09**	**-27.75**	**-29.14**	**-29.79**	**-30.7**	**-30.98**
	VolMin-KM	-11.88	-15.18	-15.16	-15.58	-15.84	-15.86
TIME[s]	KM	**0.14**	**0.11**	**0.07**	**0.06**	**0.06**	**0.06**
	RKM	0.73	0.58	0.46	0.41	0.4	0.38
	FKM	1.89	2.03	1.69	1.67	1.68	1.87
	JNKM	5.49	6.08	5.81	5.83	5.66	6.09
	JVKM	3.02	2.5	2.17	2.1	1.99	2.24
	VolMin-KM	1.68	1.57	1.46	1.43	1.39	1.39

MSE performance of VolMin and NMF.

In Table 2.6, we test the joint tensor factorization and latent clustering algorithm (JTKM). We generate a three-way tensor $\underline{\boldsymbol{X}} \in \mathbb{R}^{I \times J \times L}$ with $I = J = L = 30$ and loading factors $\boldsymbol{A} \in \mathbb{R}^{I \times F}$, $\boldsymbol{B} \in \mathbb{R}^{J \times F}$, $\boldsymbol{C} \in \mathbb{R}^{L \times F}$. To obtain $\boldsymbol{A}$ with a cluster structure on its rows, we first generate a centroid matrix $\boldsymbol{M} = 2\boldsymbol{I} + \mathbf{1}\mathbf{1}^{\mathsf{T}}$, and then replicate its columns and add noise to create $\tilde{\boldsymbol{A}}$. This way, the rows of $\tilde{\boldsymbol{A}}$ randomly scatter around the rows of $\boldsymbol{M}$. Then we let $\boldsymbol{A} = \boldsymbol{D}\tilde{\boldsymbol{A}}$, where $\boldsymbol{D}$ is a diagonal matrix whose diagonal elements are uniformly distributed between zero and one. Here, $\boldsymbol{B}$ and $\boldsymbol{C}$ are randomly drawn from an i.i.d. uniform distribution between zero and one. Gaussian noise is finally added to the obtained tensor. As in the matrix case, SNR_1 denotes the SNR in the data domain, and SNR_2 the SNR in the latent domain. In this experiment we set $\text{SNR}_1 = 20$dB, $\text{SNR}_2 = 25$dB. As before, to create more severe modeling error so that the situation is more realistic, we finally replace two slabs (i.e., $\underline{\boldsymbol{X}}(:,:,i)$'s) with elements randomly distributed between zero and one; these slabs mimic outlying data that are

Table 2.6: MSE and clustering accuracy of JTKM vs. NTF for various $F = K$.

F		2	3	4	5	6	7	8
AC[%]	JTKM	**92.97**	**74.17**	**72.33**	**74.2**	**76.93**	**78.4**	**79.47**
	NTF	80.5	64.8	62	62.63	62	62.2	62.7
MSE[dB]	JTKM	**-22.54**	**-12.45**	**-10.04**	**-10.81**	**-12.08**	**-12.92**	**-13.84**
	NTF	-21.52	-11.51	-9.66	-10.47	-11.53	-12.28	-13.34

commonly seen in practice.

We apply the tensor version of the formulation in (2.14) to factor the synthesized tensors for various $F = K$. For each parameter setting, 100 independent trials are performed with randomly generated tensors, and the results are the average of all these trials. As shown in Tab. 2.6, the proposed approach consistently yields higher clustering accuracy, and lower MSEs than plain NTF. This suggests that the clustering regularization does help in better estimating the latent factors, and yields a higher clustering accuracy.

To conclude this section, we present a simulation where the latent data representations lie in different subspaces. We apply the joint factorization and latent subspace clustering algorithm to deal with this situation. As a baseline, the latent space sparse subspace clustering (LS3C) method [102] is employed. The idea of LS3C is closely related to FKM, except that the latent clustering part is replaced by sparse subspace clustering. We construct a dataset with data that lie in two independent two-dimensional subspaces. We set $I = 10$, $J = 200$, $F = 4$ and $K = 2$; each subspace contains 100 data columns. As before, we add noise in the latent domain, as well as the data domain. The SNR in data domain is fixed at $\text{SNR}_1 = 30$ dB, and the SNR in the latent domain varies. The parameters of our formulation are set to $\lambda = 1$ and $\mu = 0.5$. For LS3C, we used the code and parameters provided by the authors[5]. The results are shown in Table 2.7. As can be seen, our method recovers the factors well, and always gets higher clustering accuracy.

[5]Available online at http://www.rci.rutgers.edu/ vmp93/Software.html

Table 2.7: Simulation comparison of proposed JNKS with LS3C

SNR_2 [dB]		3	5	7	9	11	13	15
AC[%]	LS3C	82.9	87.45	89.51	91.9	93.74	94.14	95.81
	JNKS	**87.65**	**91.17**	**92.87**	**94.39**	**95.91**	**97.15**	**97.93**
MSE[dB]	JNKS	-13.84	-15.52	-15.96	-15.75	-17.37	-16.78	-17.24

2.8 Real-data validation

2.8.1 Document clustering

We first test our proposed approach on the document clustering problem. In this problem, the data matrix $\boldsymbol{X}$ is a word-document matrix, the columns of $\boldsymbol{W}$ represent F leading topics in this collection of documents, and $\boldsymbol{H}(f, \ell)$ denotes the weight that indicates how much document ℓ is related to topic f. We use a subset of the Reuters-21578 corpus[6] as in [22], which contains 8,213 documents from 41 categories. The number of words (features) is 18,933. Following standard pre-processing, each document is represented as a term-frequency-inverse-document-frequency (tf-idf) vector, and *normalized cut weighting* is applied; see [125, 93] for details.

We apply JNKM and JVKM to the pre-processed data. A regularized NMF-based approach, namely, *locally consistent concept factorization* (LCCF) [22] is employed as the baseline. LCCF makes use of data-domain similarity to enforce latent similarity, and it demonstrates superior performance compared to other algorithms on several document clustering tasks. We also compare with spectral clustering (SC) [98], [120], which constructs a similarity graph from the data, then performs an eigenvalue decomposition (EVD) on the constructed similarity matrix. The resulting principal eigenvectors are used to infer cluster membership with simple clustering algorithms such as K-means. A more recent method, namely the symmetric NMF-based clustering algorithm (SymNMF) [78], is included for comparison since good clustering performance on Reuters dataset was reported. SymNMF is similar to SC. The difference is that symmetric NMF (instead of EVD) is performed on the similarity matrix. In addition, we

[6]Online at: `http://www.daviddlewis.com/resources/testcollections/reuters21578/`

implemented two other related methods: NMF followed by K-means (NMF-KM) [126], and RKM [37]. FKM [117] is not applied here since this method is not scalable: each iteration of FKM requires taking an eigenvalue decomposition on a large matrix (in our case a 18933×18933 matrix).

Fig. 2.5 presents our experiment results. We test the above mentioned methods on the Reuters-21578 data for various K and use $F = K$ for the methods that perform factorization. For each K, we perform 50 Monte-Carlo trials by randomly selecting K clusters out of the total 41 clusters (categories), and report the performance by comparing the results with the ground truth. Clustering performance is measured by clustering accuracy. The averaged result shows that our proposed methods, i.e., JNKM and JVKM, outperform the other methods under test. For simpler clustering tasks, e.g., when the number of clusters is small, the difference in clustering accuracy is relatively small. However, when K becomes larger, the proposed methods get consistently higher accuracy than the others, as shown in the figure. For all the K's under test, JVKM performs slightly worse than JNKM in terms of accuracy, but it has a simpler update strategy and thus is faster.

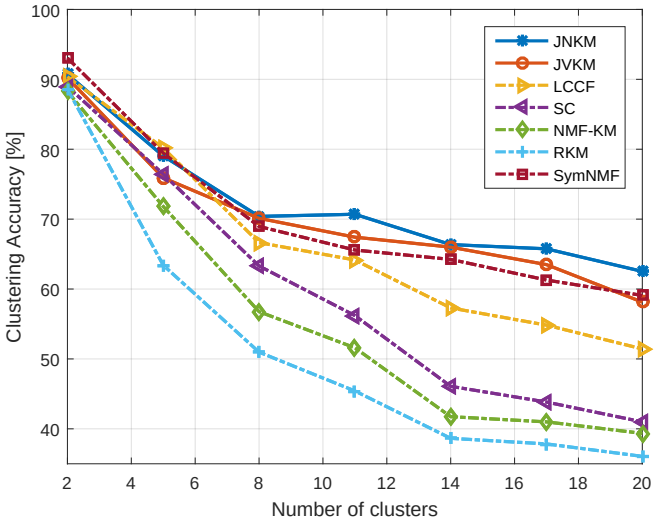


Figure 2.5: Clustering accuracy with different number of clusters on Reuters-21578 dataset

2.8.2 Image clustering

We also test the proposed approach using image data. Image data is known to be suitable for subspace clustering, and thus we apply the JNKS algorithm here. We compare our method with state-of-the-art subspace clustering methods, namely, the sparse subspace clustering (SSC) [41] and LS3C [102]. We evaluate these methods on the widely used MNIST [7] dataset. The MNIST dataset contains images of handwritten digits, from 0 to 9. We use only the testing subset of the dataset, which contains 10000 images, with each digit having ≈ 1000 images. Each 28×28 gray-level image is vectorized into a 784×1 vector. It was pointed out in [52] that vectors of each digit lie approximately in a 12-dimensional subspace (2.24) and we adopt this value in our experiments.

Fig. 2.6 shows how clustering accuracy changes with the number of clusters. For each number of clusters, we perform 50 trials, each time randomly picking digits to perform clustering. In each trial, we randomly pick 200 images for each digit. For example, for 2 clusters, each trial we will have in total 400 images. Note that JNKS outperforms SSC and LS3C when the number of cluster is moderate (5-7), and remains competitive with SSC, LS3C in all other cases.

Fig. 2.7 shows the results with all the 10 clusters under various number of samples of each cluster. We also average the results over 50 random trials as before. As can be seen, when the number of samples is small, JNKS and SSC get similar performance. With more data samples per cluster, however, the proposed method gets consistently higher accuracy.

2.8.3 Tensor social network data analysis

In this subsection, we apply the proposed JTKM algorithm to analyze the Enron email dataset, made public by the U.S. Department of Justice, as part of the Enron investigation. The data that we used contain communication counts between 184 employees over a period of 44 months, arranged in a three-way tensor of size $184 \times 184 \times 44$. The (i, j, l) entry of this tensor indicates

[7] Online available: http://yann.lecun.com/exdb/mnist/

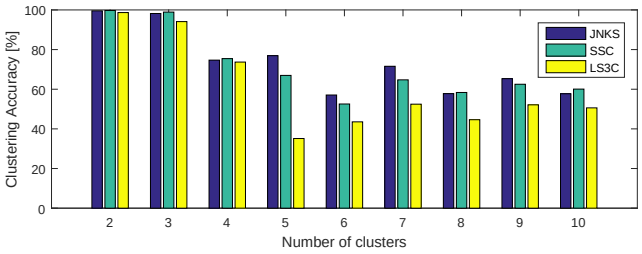


Figure 2.6: Clustering accuracy with different number of clusters on MNIST dataset, each cluster has 200 samples.

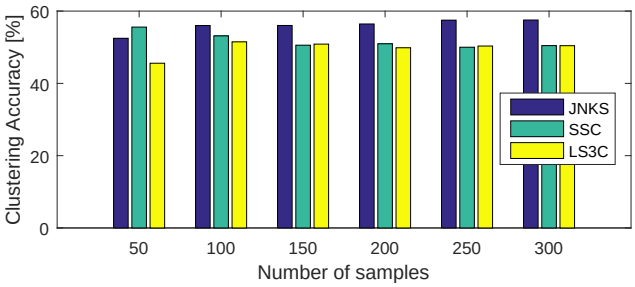


Figure 2.7: Clustering accuracy with different number of samples, all the 10 digits (clusters) from MNIST dataset.

the number of emails employee i sent to employee j in month l [8]. This dataset was analyzed with low-rank tensor factorization models [7], [101].

Enron filed for bankruptcy in late 2001, corresponding to months 36 $\sim$ 38 out of 44. Thus the 44 months can be roughly divided into pre-crisis, crisis, and post-crisis period. Therefore, we impose a cluster structure on the time-mode factor and come up with the following formulation:

$$\begin{aligned} \min_{\substack{\boldsymbol{A},\boldsymbol{B},\boldsymbol{C}\\ \boldsymbol{S},\boldsymbol{M}.}} \ & \left\| \boldsymbol{X}_{(1)} - (\boldsymbol{C} \odot \boldsymbol{B})\boldsymbol{A}^{\mathsf{T}} \right\|_F^2 + \eta \left(\|\boldsymbol{A}\|_1 + \|\boldsymbol{B}\|_1 \right) \\ & + \lambda \|\boldsymbol{C} - \boldsymbol{S}\boldsymbol{M}\|_F^2 \\ \text{s.t. } & \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C} \geq \boldsymbol{0}, \\ & \|\boldsymbol{S}(i,:)\|_0 = 1, \ \boldsymbol{S}(i,k) \in \{0,1\}, \ \forall i,k, \end{aligned} \tag{2.33}$$

[8] Online available at `http://cis.jhu.edu/~parky/Enron/enron.html`

Table 2.8: The Legal cluster identified by the three methods, Bold entries are miss-clustered.

Proposed	Co-clustering	NTF
Debra Perlingiere, Legal Specialist ENA Legal	Debra Perlingiere, Legal Specialist ENA Legal	Debra Perlingiere, Legal Specialist ENA Legal
Elizabeth Sager,VP and Asst Legal Counsel ENA Legal	Elizabeth Sager,VP and Asst Legal Counsel ENA Legal	Elizabeth Sager, VP and Asst Legal Counsel ENA Legal
Jason Williams ,Trader Central Desk Gas Trading	**Jason Williams ,Trader Central Desk Gas Trading**	**Eric Saibi, Trader**
Jeff Hodge, Asst General Counsel ENA Legal	Jeff Hodge, Asst General Counsel ENA Legal	Mark Taylor, Manager Financial Trading Group ENA Legal
Kay Mann , Lawyer	Kay Mann , Lawyer	**Jason Williams, Trader Central Desk Gas Trading**
Kim Ward, Manager West Gas Origination	**Kim Ward, Manager West Gas Origination**	Jeff Hodge, Asst General Counsel ENA Legal
Marie Heard,Senior Legal Specialist ENA Legal	Marie Heard,Senior Legal Specialist ENA Legal	Kay Mann, Lawyer
Mark Haedicke, Managing Director ENA Legal	Mark Haedicke, Managing Director ENA Legal	**Kevin Ruscitti, Trader Central Desk Gas Trading**
Mark Taylor, Manager Financial Trading Group ENA Legal	Mark Taylor, Manager Financial Trading Group ENA Legal	**Kim Ward, Manager West Gas Origination**
Sara Shackleton, Employee ENA Legal	Sara Shackleton, Employee ENA Legal	Marie Heard, Senior Legal Specialist ENA Legal
Stacy Dickson, Employee ENA Legal	Stacy Dickson, Employee ENA Legal	Mark Haedicke, Managing Director ENA Legal
Stephanie Panus , Senior Legal Specialist ENA Legal	Stephanie Panus , Senior Legal Specialist ENA Legal	Mark Haedicke, Managing Director ENA Legal
Susan Bailey, Legal Assistant ENA Legal	Susan Bailey, Legal Assistant ENA Legal	Mark Taylor, Manager Financial Trading Group ENA Legal
Tana Jones , Employee Financial Trading Group ENA Legal	Tana Jones , Employee Financial Trading Group ENA Legal	**Michelle Lokay , Admin. Asst. Transwestern Commercial Group**
		Sara Shackleton , Employee ENA Legal
		Stacy Dickson, Employee ENA Legal
		Stephanie Panus, Senior Legal Specialist ENA Legal
		Steven South,Director West Desk Gas Trading
		Susan Bailey, Legal Assistant ENA Legal
		Kim Ward, Manager West Gas Origination
		Tana Jones, Employee Financial Trading Group ENA Legal

where we set $K = 3$ and $\boldsymbol{C}$ denotes the time-mode factor. Following [101], we also use ℓ_1-norm regularizers on $\boldsymbol{A}$ and $\boldsymbol{B}$, to control scaling but also promote sparsity at the same time. We compare our latent clustering formulation with multi-way co-clustering [101] and plain NTF. As suggested by previous works on this dataset, we aim at identifying 5 groups of people, so we set $F = 5$ for all methods. Other parameters in formulation (2.33) are set to $\lambda = 500$, $\eta = 5$. The dataset is pre-processed as suggested in [7], i.e., the nonzero values are transformed using $x' = \log_2 x + 1$ to compress the dynamic range. After getting the results, we derive the clustering result from the estimated $\boldsymbol{A}$ factor. To measure the clustering quality, we first remove from the result the 71 employees who do not have clear roles, usually temporary employees and interns. The remaining 113 people have one or more of the four roles: legal (e.g., lawyers), executives (e.g. VPs, CEOs), trading, and pipeline operations.

We obtain qualitatively consistent cluster structure as reported in previous works. The 'Legal' cluster identified by the three methods is tabulated in Table 2.8. For this cluster, the proposed method gets the same result as the sparse co-clustering method [101], both of which are much cleaner than the result of NTF. In total, the proposed method gets 19 mis-classified employees compared to 21 for the sparse co-clustering [101] and 24 for plain NTF, respectively.

2.9 Chapter summary

We proposed a framework for joint factorization and latent clustering, motivated by the fact that many datasets exhibit better cluster structure in *some* reduced-dimension domain. The proposed framework leverages the identifiability of certain matrix and tensor factorization models together with a latent clustering prior to produce more discriminative latent representations that are suitable for clustering, and more accurate latent factors for estimation purposes. Carefully designed optimization objectives were proposed for joint factorization and K-means/K-subspace clustering. Alternating optimization-type algorithms were proposed for handling the proposed formulations, and judicious simulations as well as experiments with benchmark document, image, and social network data showed that the proposed approaches offer promising performance, and can outperform state-of-art methods for the respective datasets.

There are several open challenges / promising directions for future work. First, the proposed algorithms are not readily scalable to very large datasets. A lot of modern clustering tasks involve massive amounts of data, motivating work in this direction. A related question is how to adapt our methods to online or streaming settings – meaning that if the data points are acquired in a streaming fashion, can we use some low-complexity algorithm to update the factorization and clustering results without significant loss in performance? If the underlying clustering and factorization structure is changing over time, can we track the dynamics? These are important questions that deserve careful investigation, and we leave those for future work.

Chapter 3

Simultaneous Deep Learning and Clustering

3.1 Introduction

The K-means algorithm is suitable for clustering data samples that are evenly spread around some centroids (cf. the first subfigure in Fig. 3.1), but many real-life datasets do not exhibit this 'K-means-friendly' structure. Much effort has been spent on mapping high-dimensional data to a certain space that is suitable for performing K-means. Various techniques, including principal component analysis (PCA), canonical correlation analysis (CCA), nonnegative matrix factorization (NMF) and sparse coding (dictionary learning), were adopted for this purpose. In addition to these linear DR operators (e.g., a projection matrix), nonlinear DR techniques such as those used in spectral clustering [98] and sparse subspace clustering [41, 135] have also been considered.

In recent years, motivated by the success of deep neural networks (DNNs) in *supervised* learning, *unsupervised* deep learning approaches are now widely used for DR prior to clustering. For example, the stacked autoencoder (SAE) [119], deep CCA (DCCA) [3], and sparse

autoencoder [97] take insights from PCA, CCA, and sparse coding, respectively, and make use of DNNs to learn nonlinear mappings from the data domain to low-dimensional latent spaces. These approaches treat their DNNs as a preprocessing stage that is separately designed from the subsequent clustering stage. The hope is that the latent representations of the data learned by these DNNs will be naturally suitable for clustering. However, since no clustering-promoting objective is explicitly incorporated in the learning process, the learned DNNs do not necessarily output reduced-dimension data that are suitable for clustering – as will be seen in our experiments.

In [37, 102, 128], joint DR and clustering was considered. The rationale behind this line of work is that if there exists *some* latent space where the entities nicely fall into clusters, then it is natural to seek a DR transformation that reveals such structure, i.e., which yields a low K-means clustering cost. This motivates using the K-means cost in latent space as a prior that helps choose the right DR, and pushes DR towards producing K-means-friendly representations. By performing joint DR and K-means clustering, impressive clustering results have been observed in [128]. The limitation of these works is that the observable data is assumed to be generated from the latent clustering-friendly space via simple linear transformation. While simple linear transformation works well in many cases, there are other cases where the generative process is more complex, involving a nonlinear mapping.

Contributions In this chapter, we propose a joint DR and K-means clustering framework, where the DR part is implemented through learning a DNN, rather than a linear model. Unlike previous attempts that utilize this joint DNN and clustering idea, we propose customized co-design for this *unsupervised* task. Although implementing this idea is highly non-trivial (much more challenging than [37, 102, 128] where the DR part only needs to learn a linear model), our objective is well-motivated: by better modeling the data transformation process with a more general model, a much more K-means-friendly latent space can be learned – as we will demonstrate. A sneak peek of the kind of performance that can be expected using our proposed method can be seen

in Fig. 3.1, where we generate four clusters of 2-D data which are well separated in the 2-D Euclidean space and then transform them to a 100-D space using a complex non-linear mapping [cf. (3.9)] which destroys the cluster structure. One can see that the proposed algorithm outputs reduced-dimension data that are most suitable for applying K-means. Our specific contributions are as follows:

• **Optimization Criterion Design**: We propose an optimization criterion for joint DNN-based DR and K-means clustering. The criterion is a combination of three parts, namely, dimensionality reduction, data reconstruction, and cluster structure-promoting regularization. We deliberately include the reconstruction part and implement it using a decoding network, which is crucial for avoiding trivial solutions. The criterion is also flexible – it can be extended to incorporate different DNN structures (e.g. convolutional neural networks [83, 76]) and clustering criteria, e.g., subspace clustering.

• **Effective and Scalable Optimization Procedure**: The formulated optimization problem is very challenging to handle, since it involves layers of nonlinear activation functions and integer constraints that are induced by the K-means part. We propose a judiciously designed solution package, including empirically effective initialization and a novel alternating stochastic gradient algorithm. The algorithmic structure is simple, enables online implementation, and is very scalable.

• **Comprehensive Experiments and Validation**: We provide a set of synthetic-data experiments and validate the method on different real datasets including various document and image copora. Evidently visible improvement from the respective state-of-art is observed for all the datasets that we experimented with.

• **Reproducibility**: The code for the experiments is available at `https://github.com/boyangumn/DCN`.

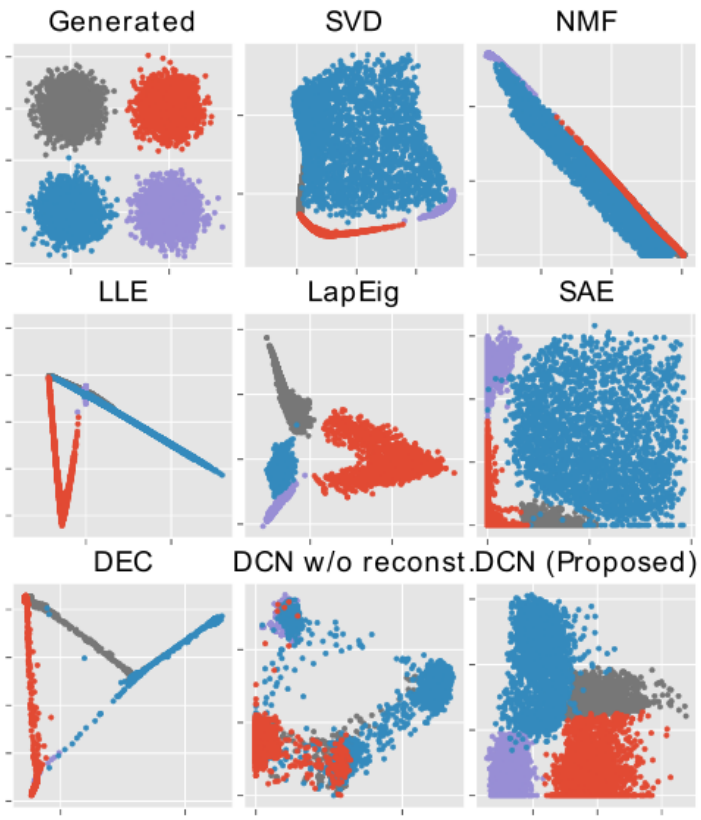


Figure 3.1: The learned 2-D reduced-dimension data by different methods. The observable data is in the 100-D space and is generated from 2-D data (cf. the first subfigure) through the nonlinear transformation in (3.9). The true cluster labels are indicated using different colors.

3.2 Background and related works

Given a set of data samples $\{\boldsymbol{x}_i\}_{i=1,\dots,N}$ where $\boldsymbol{x}_i \in \mathbb{R}^M$, the task of clustering is to group the N data samples into K categories. Arguably, K-means [90] is the most widely adopted algorithm. K-means approaches this task by optimizing the following cost function:

$$\min_{\boldsymbol{M}\in\mathbb{R}^{M\times K},\{\boldsymbol{s}_i\in\mathbb{R}^K\}} \sum_{i=1}^{N} \|\boldsymbol{x}_i - \boldsymbol{M}\boldsymbol{s}_i\|_2^2 \tag{3.1}$$
$$\text{s.t. } s_{j,i} \in \{0,1\},\ \mathbf{1}^T\boldsymbol{s}_i = 1\ \ \forall i,j,$$

where $\boldsymbol{s}_i$ is the assignment vector of data point i which has only one non-zero element, $s_{j,i}$ denotes the jth element of $\boldsymbol{s}_i$, and the kth column of $\boldsymbol{M}$, i.e., $\boldsymbol{m}_k$, denotes the centroid of the kth cluster.

K-means works well when the data samples are evenly scattered around their centroids in the feature space; we consider datasets which have this structure as being 'K-means-friendly' (cf. top-left subfigure of Fig. 3.1). However, high-dimensional data are in general not very K-means-friendly. In practice, using a DR pre-processing, e.g., PCA or NMF [126, 22], to reduce the dimension of $\boldsymbol{x}_i$ to a much lower dimensional space and then applying K-means usually gives better results. In addition to the above classic DR methods that essentially learn a linear generative model from the latent space to the data domain, nonlinear DR approaches such as those used in spectral clustering [98, 120] and DNN-based DR [58, 107, 55] are also widely used as pre-processing before K-means or other clustering algorithms, see also [119, 21].

Instead of using DR as a pre-processing, joint DR and clustering was also considered in the literature [37, 102, 128]. This line of work can be summarized as follows. Consider the generative model where a data sample is generated by $\boldsymbol{x}_i = \boldsymbol{W}\boldsymbol{h}_i$, where $\boldsymbol{W} \in \mathbb{R}^{M\times R}$ and $\boldsymbol{h}_i \in \mathbb{R}^R$, where $R \ll M$. Assume that the data clusters are well-separated in latent domain (i.e., where $\boldsymbol{h}_i$ lives) but distorted by the transformation introduced by $\boldsymbol{W}$. In reference [128] we

formulated the joint optimization problem as follows:

$$\begin{aligned} \min_{\boldsymbol{M},\{\boldsymbol{s}_i\},\boldsymbol{W},\boldsymbol{H}} \ & \|\boldsymbol{X}-\boldsymbol{W}\boldsymbol{H}\|_F^2+\lambda\sum_{i=1}^{N}\|\boldsymbol{h}_i-\boldsymbol{M}\boldsymbol{s}_i\|_2^2 \\ & +r_1(\boldsymbol{H})+r_2(\boldsymbol{W}) \\ \text{s.t. } & s_{j,i}\in\{0,1\},\ \mathbf{1}^T\boldsymbol{s}_i=1\ \ \forall i,j, \end{aligned} \tag{3.2}$$

where $\boldsymbol{X}=[\boldsymbol{x}_1,\ldots,\boldsymbol{x}_N]$, $\boldsymbol{H}=[\boldsymbol{h}_1,\ldots,\boldsymbol{h}_N]$, and $\lambda\geq 0$ is a parameter for balancing data fidelity and the latent cluster structure. In (3.2), the first term performs DR and the second term performs latent clustering. The terms $r_1(\cdot)$ and $r_2(\cdot)$ are regularizations (e.g., nonnegativity or sparsity) to prevent trivial solutions, e.g., $\boldsymbol{H}\to\mathbf{0}\in\mathbb{R}^{R\times N}$; see details in [128].

The data model $\boldsymbol{X}\approx\boldsymbol{W}\boldsymbol{H}$ in the above line of work may be oversimplified: the data generating process can be much more complex than this linear transform. Therefore, it is well justified to seek powerful non-linear transforms, e.g. DNNs, to model this data generating process, while at the same time make use of the joint DR and clustering idea. Two recent works, [124] and [132], made such attempts.

The idea of [124] and [132] is to connect a clustering module to the output layer of a DNN, and jointly learn DNN parameters and clusters. Specifically, the approaches look into an optimization problem of the following form

$$\min_{\mathcal{W},\Theta}\ \widehat{L}=\sum_{i=1}^{N}q(\boldsymbol{f}(\boldsymbol{x}_i;\mathcal{W});\Theta), \tag{3.3}$$

where $\boldsymbol{f}(\boldsymbol{x}_i;\mathcal{W})$ is the network output given data sample $\boldsymbol{x}_i$, $\mathcal{W}$ collects the network parameters, and Θ denotes parameters of some clustering model. For instance, Θ stands for the centroids $\boldsymbol{M}$ and assignments $\{\boldsymbol{s}_i\}$ if the K-means clustering formulation (3.1) is adopted. The $q(\cdot)$ in (3.3) denotes some clustering loss, e.g., the Kullback-Leibler (KL) divergence loss in [124] and agglomerative clustering loss in [132]. An illustration of this kind of approaches is shown

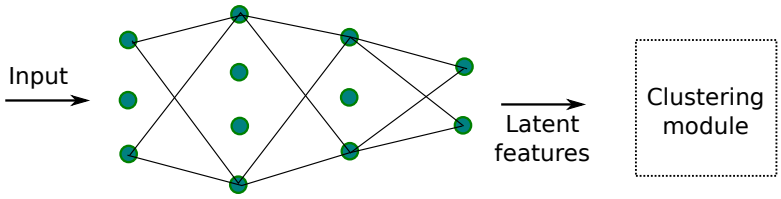


Figure 3.2: A problematic *joint* deep clustering structure. To avoid clutter, some links are omitted.

in Fig. 3.2. This idea seems reasonable, but is problematic. A *global optimal* solution to Problem (3.3) is $f(\boldsymbol{x}_i; \mathcal{W}) = \mathbf{0}$ and the optimal objective value $\widehat{L} = 0$ can always be achieved. Another type of trivial solutions are simply mapping *arbitrary* data samples to tight clusters, which will lead to a small value of $\widehat{L}$ – but this could be far from being desired since there is no provision for preserving the essential information in the data samples $\boldsymbol{x}_i$'s; see the bottom-middle subfigure in Fig. 3.1 [Deep Clustering Network (DCN) w/o reconstruction] and the bottom-left subfigure in Fig. 3.1 [DEC]. This issue also exists in [132].

3.3 Proposed formulation

We are motivated to model the relationship between the observable data $\boldsymbol{x}_i$ and its clustering-friendly latent representation $\boldsymbol{h}_i$ using a nonlinear mapping, i.e.,

$$\boldsymbol{h}_i = \boldsymbol{f}(\boldsymbol{x}_i; \mathcal{W}), \quad \boldsymbol{f}(\cdot; \mathcal{W}) : \mathbb{R}^M \to \mathbb{R}^R,$$

where $\boldsymbol{f}(\cdot; \mathcal{W})$ denotes the mapping function and $\mathcal{W}$ denote the set of parameters. In this chapter, we propose to employ a DNN as our mapping function, since DNNs have the ability of approximating any continuous mapping using a reasonable number of parameters [61].

We want to learn the DNN and perform clustering *simultaneously*. The critical question here is how to avoid trivial solutions in this *unsupervised* task. In fact, this can be resolved by taking insights from (3.2). The key to prevent trivial solution in the linear DR case lies

in the reconstruction part, i.e., the term $\|\boldsymbol{X} - \boldsymbol{W}\boldsymbol{H}\|_F^2$ in (3.2). This term ensures that the learned $\boldsymbol{h}_i$'s can (approximately) reconstruct the $\boldsymbol{x}_i$'s using the basis $\boldsymbol{W}$. This motivates incorporating a reconstruction term in the joint DNN-based DR and K-means. In the realm of unsupervised DNN, there are several well-developed approaches for reconstruction – e.g., the stacked autoencoder (SAE) is a popular choice for serving this purpose. To prevent trivial low-dimensional representations such as all-zero vectors, SAE uses a decoding network $\boldsymbol{g}(\cdot;\mathcal{Z})$ to map the $\boldsymbol{h}_i$'s back to the data domain and requires that $\boldsymbol{g}(\boldsymbol{h}_i;\mathcal{Z})$ and $\boldsymbol{x}_i$ match each other well under some metric, e.g., mutual information or least squares-based measures.

By the above reasoning, we come up with the following cost function:

$$\begin{aligned} \min_{\substack{\boldsymbol{W},\mathcal{Z},\\ \boldsymbol{M},\{\boldsymbol{s}_i\}}} \ & \sum_{i=1}^{N} \left(\ell\left(\boldsymbol{g}(\boldsymbol{f}(\boldsymbol{x}_i)), \boldsymbol{x}_i\right) + \frac{\lambda}{2} \left\| \boldsymbol{f}(\boldsymbol{x}_i) - \boldsymbol{M}\boldsymbol{s}_i \right\|_2^2 \right) \\ \text{s.t.} \ & \ s_{j,i} \in \{0,1\}, \ \mathbf{1}^T s_i = 1 \ \ \forall i, j, \end{aligned} \tag{3.4}$$

where we have simplified the notation $\boldsymbol{f}(\boldsymbol{x}_i;\mathcal{W})$ and $\boldsymbol{g}(\boldsymbol{h}_i;\mathcal{Z})$ to $\boldsymbol{f}(\boldsymbol{x}_i)$ and $\boldsymbol{g}(\boldsymbol{h}_i)$, respectively, for conciseness. The function $\ell(\cdot) : \mathbb{R}^M \to \mathbb{R}$ is a certain loss function that measures the reconstruction error. In this chapter, we adopt the least-squares loss $\ell(\boldsymbol{x},\boldsymbol{y}) = \|\boldsymbol{x}-\boldsymbol{y}\|_2^2$; other choices such as ℓ_1-norm based fitting and the KL divergence can also be considered. $\lambda \geq 0$ is a regularization parameter which balances the reconstruction error versus finding K-means-friendly latent representations.

Fig. 3.3 presents the network structure corresponding to the formulation in (3.4). Compare to the network in Fig. 3.2, our latent features are also responsible for reconstructing the input, preventing all the aforementioned trivial solutions. On the left-hand side of the 'bottleneck' layer are the so-called encoding or forward layers that transform raw data to a low-dimensional space. On the right-hand side are the 'decoding' layers that try to reconstruct the data from the latent space. The K-means task is performed at the bottleneck layer. The forward network, the decoding network, and the K-means cost are optimized simultaneously. In our experiments,

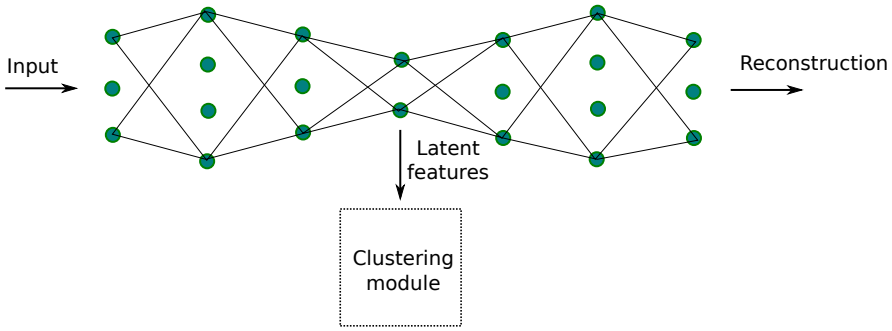


Figure 3.3: Proposed deep clustering network (DCN).

the structure of the decoding networks is a 'mirrored version' of the encoding network, and for both the encoding and decoding networks, we use *rectified linear unit* (ReLU) activation-based neurons [96]. Since our objective is to perform DNN-driven K-means clustering, we will refer to the network in Fig. 3.3 as the Deep Clustering Network (DCN) in the sequel.

We should remark that the proposed optimization criterion in (3.4) and the network in Fig. 3.3 are very flexible: Other types of networks, e.g., deep convolutional neural networks [83, 76], can be used. For the clustering part, other clustering criteria, e.g., K-subspace and soft K-means [82, 8], are also viable options. Nevertheless, we will concentrate on the proposed DCN in the sequel, as our interest is to provide a proof-of-concept rather than exhausting the possibilities of combinations.

3.4 Optimization procedure

Optimizing (3.4) is highly non-trivial since both the cost function and the constraints are non-convex. In addition, there are scalability issues that need to be taken into account. In this section, we propose a pragmatic optimization procedure including an empirically effective initialization method and an alternating optimization based algorithm for handling (3.4).

3.4.1 Initialization via layer-wise pre-training

For dealing with hard non-convex optimization problems like that in (3.4), initialization is usually crucial. To initialize the parameters of the network, i.e., $(\mathcal{W}, \mathcal{Z})$, we use the layer-wise pre-training method as in [12] for training autoencoders. This pre-training technique may be avoided in large-scale supervised learning tasks. For the proposed DCN which is completely unsupervised, however, we find that the layer-wise pre-training procedure is important no matter the size of the dataset. We refer the readers to [12] for an introduction of layer-wise pre-training. After pre-training, we perform K-means to the outputs of the bottleneck layer to obtain initial values of $\boldsymbol{M}$ and $\{\boldsymbol{s}_i\}$.

3.4.2 Alternating stochastic optimization

Even with a good initialization, handling Problem (3.4) is still very challenging. The commonly used stochastic gradient descent (SGD) algorithm cannot be directly applied to jointly optimize $\mathcal{W}, \mathcal{Z}, \boldsymbol{M}$ and $\{\boldsymbol{s}_i\}$ because the block variable $\{\boldsymbol{s}_i\}$ is constrained on a discrete set. Our idea is to combine the insights of alternating optimization and SGD. Specifically, we propose to optimize the subproblems with respect to (w.r.t.) one of $\boldsymbol{M}$, $\{\boldsymbol{s}_i\}$ and $(\mathcal{W}, \mathcal{Z})$ while keeping the other two sets of variables fixed.

Update network parameters

For fixed $(\boldsymbol{M}, \{\boldsymbol{s}_i\})$, the subproblem w.r.t. $(\mathcal{W}, \mathcal{Z})$ is similar to training an SAE – but with an additional penalty term on the clustering performance. We can take advantage of the mature tools for training DNNs, e.g., back-propagation based SGD and its variants. To implement SGD for updating the network parameters, we look at the problem w.r.t. the incoming data $\boldsymbol{x}_i$:

$$\min_{\mathcal{W},\mathcal{Z}} L^i = \ell\left(\boldsymbol{g}(\boldsymbol{f}(\boldsymbol{x}_i)), \boldsymbol{x}_i\right) + \frac{\lambda}{2}\left\|\boldsymbol{f}(\boldsymbol{x}_i) - \boldsymbol{M}\boldsymbol{s}_i\right\|_2^2. \tag{3.5}$$

The gradient of the above function over the network parameters is easily computable, i.e., $\nabla_{\mathcal{X}} L^i = \frac{\partial \ell(\boldsymbol{g}(\boldsymbol{f}(\boldsymbol{x}_i)), \boldsymbol{x}_i)}{\partial \mathcal{X}} + \lambda \frac{\partial \boldsymbol{f}(\boldsymbol{x}_i)}{\partial \mathcal{X}} (\boldsymbol{f}(\boldsymbol{x}_i) - \boldsymbol{M}\boldsymbol{s}_i)$, where $\mathcal{X} = (\mathcal{W}, \mathcal{Z})$ is a collection of the network parameters and the gradients $\frac{\partial \ell}{\partial \mathcal{X}}$ and $\frac{\partial \boldsymbol{f}(\boldsymbol{x}_i)}{\partial \mathcal{X}}$ can be calculated by back-propagation [103] (strictly speaking, what we calculate here is the *subgradient* w.r.t. $\mathcal{X}$ since the ReLU function is non-differentible at zero). Then, the network parameters are updated by

$$\mathcal{X} \leftarrow \mathcal{X} - \alpha \nabla_{\mathcal{X}} L^i, \tag{3.6}$$

where $\alpha > 0$ is a diminishing learning rate.

Update clustering parameters

For fixed network parameters and $\boldsymbol{M}$, the assignment vector of the current sample, i.e., $\boldsymbol{s}_i$, can be naturally updated in an online fashion. Specifically, we update $\boldsymbol{s}_i$ as follows:

$$s_{j,i} \leftarrow \begin{cases} 1, & \text{if } j = \underset{k=\{1,\ldots,K\}}{\arg\min} \left\| \boldsymbol{f}(\boldsymbol{x}_i) - \boldsymbol{m}_k \right\|_2, \\ 0, & \text{otherwise.} \end{cases} \tag{3.7}$$

When fixing $\{\boldsymbol{s}_i\}$ and $\mathcal{X}$, the update of $\boldsymbol{M}$ is simple and may be done in a variety of ways. For example, one can simply use $\boldsymbol{m}_k = (1/|\mathcal{C}_k^i|) \sum_{i \in \mathcal{C}_k^i} \boldsymbol{f}(\boldsymbol{x}_i)$, where $\mathcal{C}_k^i$ is the recorded index set of samples assigned to cluster k from the first sample to the current sample i. Although the above update is intuitive, it could be problematic for online algorithms, since the already appeared historical data (i.e., $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_i$) might not be representative enough to model the global cluster structure and the initial $\boldsymbol{s}_i$'s might be far away from being correct. Therefore, simply averaging the current assigned samples may cause numerical problems. Instead of doing the above, we employ the idea in [109] to adaptively change the learning rate of updating $\boldsymbol{m}_1, \ldots, \boldsymbol{m}_K$. The intuition is simple: assume that the clusters are roughly balanced in terms of the number of data samples they contain. Then, after updating $\boldsymbol{M}$ for a number of samples, one should update

the centroids of the clusters that already have many assigned members more gracefully while updating others more aggressively, to keep balance. To implement this, let c_k^i be the count of the number of times the algorithm assigned a sample to cluster k before handling the incoming sample $\boldsymbol{x}_i$, and update $\boldsymbol{m}_k$ by a simple gradient step:

$$\boldsymbol{m}_k \leftarrow \boldsymbol{m}_k - (1/c_k^i)\,(\boldsymbol{m}_k - \boldsymbol{f}(\boldsymbol{x}_i))\,s_{k,i}, \tag{3.8}$$

where the gradient step size $1/c_k^i$ controls the learning rate. The above update of $\boldsymbol{M}$ can also be viewed as an SGD step, thereby resulting in an overall alternating block SGD procedure that is summarized in Algorithm 1. Note that an epoch corresponds to a pass of all data samples through the network.

Algorithm 1 Alternating SGD

1: Initialization {Perform T epochs over the data}
2: **for** $t = 1 : T$ **do**
3: Update network parameters by (3.6)
4: Update assignment by (3.7)
5: Update centroids by (3.8)
6: **end for**

Algorithm 1 has many favorable properties. First, it can be implemented in a completely online fashion, and thus is very scalable. Second, many known tricks for enhancing performance of DNN training can be directly used. In fact, we have used a mini-batch version of SGD and batch-normalization [72] in our experiments, which indeed help improve performance.

3.5 Experiments

In this section, we use synthetic and real-world data to showcase the effectiveness of DCN. We implement DCN using the deep learning toolbox Theano [113].

3.5.1 Synthetic-data demonstration

Our settings are as follows: Assume that the data points have K-means-friendly structure in a two-dimensional domain (cf. the first subfigure of Fig. 3.1). This two-dimensional domain is a latent domain which we do not observe and we denote the latent representations of the data points as $\boldsymbol{h}_i$'s in this domain. What we observe is $\boldsymbol{x}_i \in \mathbb{R}^{100}$ that is obtained via the following transformation:

$$\boldsymbol{x}_i = \sigma\left(\boldsymbol{U}\sigma(\boldsymbol{W}\boldsymbol{h}_i)\right), \tag{3.9}$$

where $\boldsymbol{W} \in \mathbb{R}^{10\times 2}$ and $\boldsymbol{U} \in \mathbb{R}^{100\times 10}$ are matrices whose entries follow the zero-mean unit-variance i.i.d. Gaussian distribution, $\sigma(\cdot)$ is a sigmod function to introduce nonlinearity. Under the above generative model, recovering the K-means-friendly domain where $\boldsymbol{h}_i$'s live seems very challenging.

We generate four clusters, each of which has 2,500 samples and their geometric distribution on the 2-D plane is shown in the first subfigure of Fig. 3.1 that we have seen before. The other subfigures show the recovered 2-D data from $\boldsymbol{x}_i$'s using a number of DR methods, namely, NMF [84], local linear embedding (LLE) [105], Laplacian eigenmap (LapEig) [98] – the first step of spectral clustering, and DEC [124]. We also present the result of using the formulation in (3.3) (DCN w/o reconstruction) which is a similar idea as in [124]. For the three DNN-based methods (DCN, DEC, and SAE + KM), we use a four-layer forward network for dimensionality reduction, where the layers have 100, 50, 10 and 2 neurons, respectively; the reconstruction network used in DCN and SAE (and also in the per-training stage of DEC) is a mirrored version of the forward network. As one can see in Fig. 3.1, all the DR methods except the proposed DCN fail to map $\boldsymbol{x}_i$'s to a 2-D domain that is suitable for applying K-means. In particular, DEC and DCN w/o reconstruction indeed give trivial solutions: the reduced-dimension data are separated to four clusters, and thus $\hat{L}$ is small. But this solution is meaningless since the data partitioning

is arbitrary.

In the supplementary materials, two additional simulations with different generative model than (3.9) are presented, and similar results are observed. This further illustrates the DCN's ability of recovering clustering-friendly structure under different nonlinear generative models.

3.5.2 Real-data validation

In this section, we validate the proposed approach on several real-data sets which are all publicly available.

Baseline methods

We compare the proposed DCN with a variety of baseline methods:

1) **K-means (KM)**: The classic K-means [90].

2) **Spectral Clustering (SC)**: The classic SC algorithm [98].

3) **Sparse Subspace Clustering with Orthogonal Matching Pursuit (SSC-OMP)** [135]: SSC is considered very competitive for clustering images; we use the newly proposed greedy version here for scalability.

4) **Locally Consistent Concept Factorization (LCCF)** [22]: LCCF is based on NMF with a graph Laplacian regularization and is considered state-of-the-art for document clustering.

5) **XRAY** [80]: XRAY is an NMF-based document clustering algorithm that scales very well.

6) **NMF followed by K-means** (NMF+KM): This approach applies NMF for DR, and then applies K-means to the reduced-dimension data.

7) **Stacked Autoencoder followed by K-means (SAE+KM)**: This is also a two-stage approach. We use SAE for DR first and then apply K-means.

8) **Joint NMF and K-means (JNKM)** [128]: JNKM performs joint DR and K-means clustering as the proposed DCN does – but the DR part is based on NMF.

9) **Deep Embedded Clustering (DEC)** [124]: DEC performs joint DNN and clustering, where

the loss function contains only clustering loss, without penalty on reconstruction as in our method. We use the code[1] provided by the authors. For each experiment, we select the baselines that are considered most competitive and suitable for that application from the above pool.

Evaluation metrics

We adopt standard metrics for evaluating clustering performance. Specifically, we employ the following three metrics: normalized mutual information (NMI) [22], adjusted Rand index (ARI) [133], and clustering accuracy (ACC) [22]. In a nutshell, all the above three metrics are commonly used in the clustering literature, and all have pros and cons. But using them together suffices to demonstrate the effectiveness of the clustering algorithms. Note that NMI and ACC lie in the range of zero to one with one being the perfect clustering result and zero the worst. ARI is a value within -1 to 1, with one being the best clustering performance and minus one the opposite.

RCV1

We first test the algorithms on a large-scale text corpus, namely, the Reuters Corpus Volume 1 Version 2 (RCV1-v2). The RCV1-v2 corpus [85] contains 804,414 documents, which were manually categorized into 103 different topics. We use a subset of the documents from the whole corpus. This subset contains 20 topics and $365,968$ documents and each document has a single topic label. As in [99], we pick the 2,000 most frequently used words (in the tf-idf form) as the features of the documents.

We conduct experiments using different number of clusters. Towards this end, we first sort the clusters according to the number of documents that they have in a descending order, and then apply the algorithms to the first 4, 8, 12, 16, 20 clusters, respectively. Note that the first several clusters have many more documents compared to the other clusters (cf. Fig. 3.4). This way, we

[1] https://github.com/piiswrong/dec

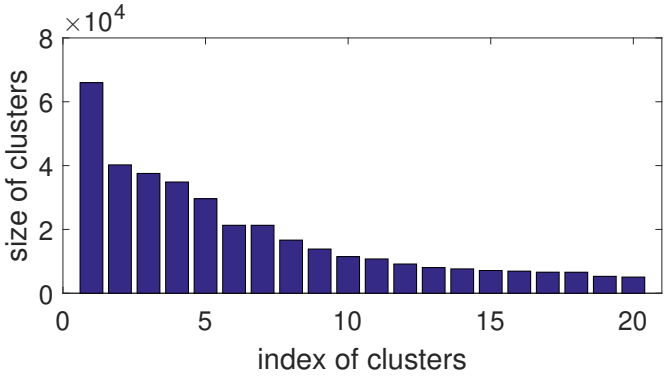


Figure 3.4: The sizes of 20 clusters in the experiment.

gradually increase the number of documents in our experiments and create cases with much more unbalanced cluster sizes for testing the algorithms – which means we gradually increase the difficulty of the experiments. To avoid unrealistic tuning, for all the experiments, we use a DCN whose forward network has five hidden layers which have $2000, 1000, 1000, 1000, 50$ neurons, respectively. The reconstruction network has a mirrored structure. We set $\lambda = 0.1$ for balancing the reconstruction error and the clustering regularization.

Table 3.1 shows the results given by the proposed DCN, SAE+KM, KM, and XRAY; other baselines are not scalable enough to handle the RCV1-v2 dataset and thus are dropped. One can see that for each case that we have tried, the proposed method gives clear improvement relative to the other methods. Particularly, the DCN approach outperforms the two-stage approach, i.e., SAE+KM, in almost all the cases and for all the evaluation metrics – this clearly demonstrates the advantage of using the joint optimization criterion. We notice that the performance of DEC in this experiment is unsatisfactory, possibly because 1) this dataset is highly unbalanced (cf. Fig. 3.4), while DEC is designed to produce balanced clusters; 2) DEC gets trapped in trivial solutions, as we discussed in Sec 3.2.

Fig. 3.5 shows how NMI, ARI, and ACC change when the proposed algorithm runs from epoch to epoch. One can see a clear ascending trend of every evaluation metric. This result shows that both the network structure and the optimization algorithm work towards a desired

Methods		DCN	SAE+KM	KM	DEC	XRAY
4 Clust.	NMI	**0.76**	0.73	0.62	0.11	0.12
	ARI	**0.67**	0.65	0.50	0.07	-0.01
	ACC	**0.80**	0.79	0.70	0.38	0.34
8 Clust.	NMI	**0.63**	0.60	0.57	0.10	0.24
	ARI	**0.46**	0.42	0.38	0.05	0.09
	ACC	**0.63**	0.62	0.59	0.24	0.39
12 Clust.	NMI	**0.67**	0.65	0.6	0.09	0.22
	ARI	**0.52**	0.51	0.37	0.02	0.05
	ACC	**0.60**	0.56	0.54	0.18	0.29
16 Clust.	NMI	**0.62**	0.60	0.56	0.09	0.23
	ARI	**0.36**	0.35	0.30	0.02	0.04
	ACC	**0.51**	0.50	0.48	0.17	0.29
20 Clust.	NMI	**0.61**	0.59	0.58	0.08	0.25
	ARI	**0.33**	**0.33**	0.29	0.01	0.04
	ACC	**0.47**	0.46	**0.47**	0.14	0.28

Table 3.1: Evaluation on the RCV1-v2 dataset

direction. In the future, it would be intriguing to derive (sufficient) conditions for guaranteeing such improvement using the proposed algorithm. Nevertheless, such empirical observation in Fig. 3.5 is already very interesting and encouraging.

We visualize the 50-D learned embeddings of our network on the RCV1 4-clusters dataset, using t-SNE [116], as shown in Fig. 3.7. We can see that the proposed DCN method learns much improved results compared to the initialization. Also, the DEC method does not get a desiable clustering result, possibly due to the imbalance clusters.

20Newsgroup

The 20Newsgroup corpus is a collection of 18,846 text documents which are partitioned into 20 different newsgroups. Using this corpus, we can observe how the proposed method works with a relatively small amount of samples. As the previous experiment, we use the tf-idf representation of the documents and pick the 2,000 most frequently used words as the features. Since this dataset is small, we include more baselines that are not scalable enough for RCV1-v2. Among them,

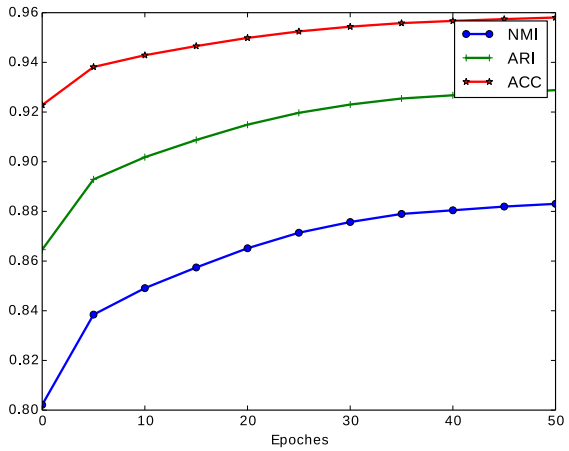


Figure 3.5: Clustering performance metrics v.s. training epochs.

Methods	DCN	SAE+KM	LCCF	NMF+KM	KM	SC	XARY	JNKM
NMI	**0.48**	0.47	0.46	0.39	0.41	0.40	0.19	0.40
ARI	**0.34**	0.28	0.17	0.17	0.15	0.17	0.02	0.10
ACC	**0.44**	0.42	0.32	0.33	0.3	0.34	0.18	0.24

Table 3.2: Evaluation on the 20Newsgroup dataset.

both JNKM and LCCF are considered state-of-art for document clustering. In this experiment, we use a DNN with three forward layers which have 250, 100, and 20 neurons, respectively. This is a relatively 'small network' since the 20Newsgroup corpus may not have sufficient samples to fit a large network. As before, the decoding network for reconstruction has a mirrored structure of the encoding part, and the baseline SAE+KM uses the same network for the autoencoder part.

Table 3.2 summarizes the results of this experiment. As one can see, LCCF indeed gives the best performance among the algorithms that do not use DNNs. SAE+KM improves ARI and ACC quite substantially by involving DNN – this suggests that the generative model may indeed be nonlinear. DCN performs even better by using the proposed joint DR and clustering criterion, which supports our motivation that a K-means regularization can help discover a clustering-friendly space.

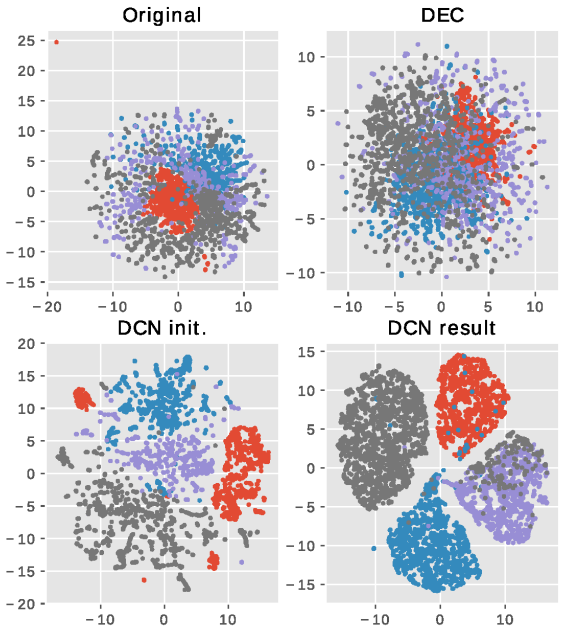


Figure 3.6: Visualization using t-SNE. From top-left to bottom-right: Original data, DEC result, DCN initialization, DCN result

Figure 3.7: Visualization on the 4-clusters subset of RCV1-v2

Raw MNIST

In this and next subsections, we present two experiments using two versions of the MNIST dataset. We first employ the raw MNIST dataset that has 70,000 data samples. Each sample is a 28×28 gray-scale image containing a handwritten digit, i.e., one of $\{0, 1, \ldots, 9\}$. Same as [124], we use a 4-layers forward network and set the number of neurons to be 500, 500, 2000, and 10, respectively. The reconstruction network is still a 'mirrored' version of the forward network. The hyperparameter λ is set to 1. We use SSC-OMP, which is a scalable version of SSC, and KM as a baseline for this experiment.

Table 3.3 shows results of applying DCN, SAE+KM, DEC, KM and SSC-OMP to the raw

Methods	DCN	SAE+KM	DEC	KM	SSC-OMP
NMI	**0.81**	0.73	0.80	0.50	0.31
ARI	**0.75**	0.67	**0.75**	0.37	0.13
ACC	0.83	0.80	**0.84**	0.53	0.30

Table 3.3: Evaluation on the raw MNIST dataset.

MNIST data – the other baselines are not efficient enough to handle 70,000 samples and thus are left out. One can see that our result is on par with the result of DEC reported in [124], and both methods outperform other methods by a large margin. The DEC method performs very competitively on this dataset, possibly because it is designed to favor balanced clusters, which is the case for MNIST dataset. On the dataset RCV1-v2 with unbalanced clusters, the result of DEC is not as satisfactory, see Fig. 3.7. It is also interesting to note that our method yields approximately same results as DEC in this balanced case, but DCN also works well in unbalanced cases, as we have seen.

Pre-processed MNIST

Besides the above experiment using the raw MNIST data, we also provide another interesting experiment using *pre-processed* MNIST data. The pre-processing is done by a recently introduced technique, namely, the scattering network (ScatNet) [21]. ScatNet is a cascade of multiple layers of wavelet transform, which is able to learn a good feature space for clustering / classification of images. Utilizing ScatNet, the work in [135] reported very promising clustering results on MNIST using SSC-OMP. Our objective here is to see if the proposed DCN can further improve the performance from SSC-OMP. Our idea is simple: SSC-OMP is essentially a procedure of constructing a similarity matrix of the data; after obtaining this matrix, it performs K-means on the rows of a matrix comprising several selected eigenvectors of the similarity matrix [98]. Therefore, it makes sense to treat the whole ScatNet + SSC-OMP procedure as pre-processing for performing K-means, and one can replace K-means by DCN to improve performance.

Methods	DCN	SAE+KM	KM (SSC-OMP)
NMI	**0.88**	0.86	0.85
ARI	**0.89**	0.86	0.82
ACC	**0.95**	0.93	0.86

Table 3.4: Evaluation on pre-processed MNIST

The results are shown in Table 3.4. One can see that the proposed method exhibits the best performance among the algorithms. We note that the result of using KM on the data processed by ScatNet and SSC-OMP is worse than that was reported in [135]. This is possibly because we use all the 70,000 samples, while only a subset was selected for conducting the experiments in [135].

This experiment is particularly interesting since it suggests that for any clustering algorithm that employs K-means as a key component, e.g., spectral clustering and sparse subspace clustering, one can use the proposed DCN to replace K-means and a better result can be expected. This is meaningful since many datasets are originally not suitable for K-means due to the nature of the data – but after pre-processing (e.g., kernelization and eigendecomposition), the pre-processed data is already more K-means-friendly, and using the proposed DCN at this point can further strengthen the result.

Parameter Selection

The parameter λ is important, since it trades off between the reconstruction objective and the clustering objective. As we see from the experiments, the proposed DCN works well with an appropriately chosen λ. Moreover, our experience suggests that the performance of our approach is insensitive to the exact value of λ. Fig. 3.8 shows how the proposed method performs with different λ on the MNIST dataset. As we can see, although there is degradation of performance as λ gets inappropriately large, the degradation is mild. The proposed method gives satisfactory result for a range of λ.

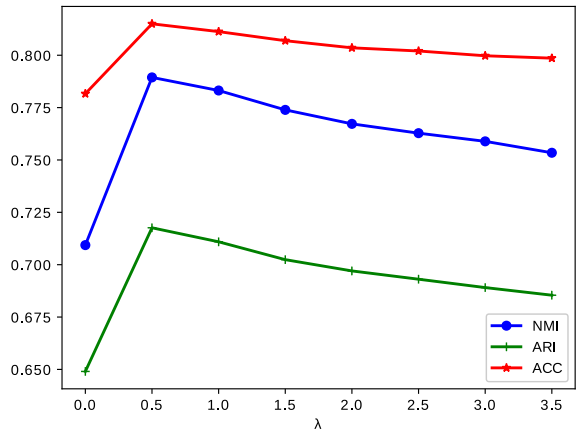


Figure 3.8: Clustering performance on MNIST with different λ.

3.6 Chapter summary

In this chapter, we proposed a joint DR and K-means clustering approach where the DR part is accomplished via learning a deep neural network. Our goal is to automatically map high-dimensional data to a latent space where K-means is a suitable tool for clustering. We carefully designed the network structure to avoid trivial and meaningless solutions and proposed an effective and scalable optimization procedure to handle the formulated challenging problem. Synthetic and real data experiments showed that the algorithm is very effective on a variety of datasets.

3.7 Appendix

3.7.1 Additional synthetic data experiments

In this section, we provide two more examples to illustrate the ability of DCN in recovering K-means-friendly spaces under different generative models. We first consider the transformation

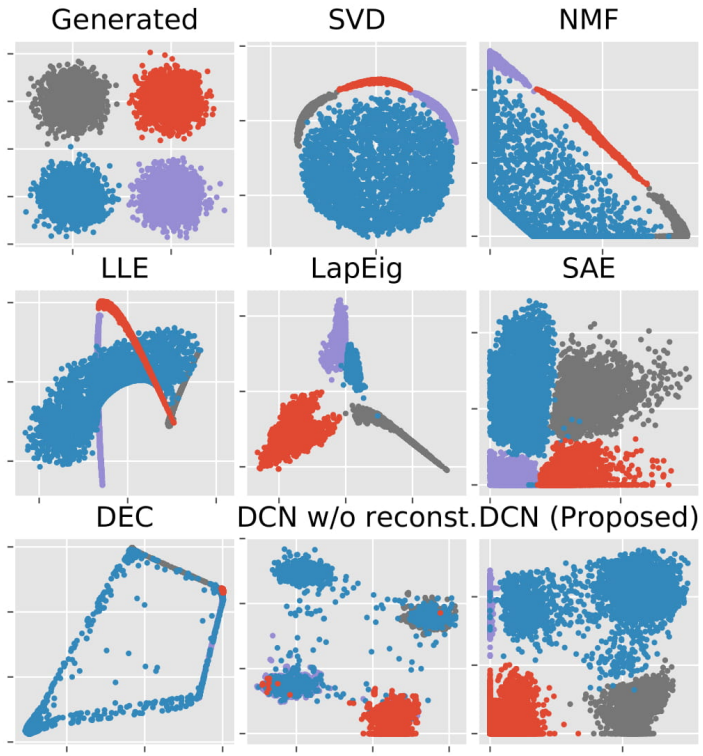


Figure 3.9: The generated latent representations $\{\boldsymbol{h}_i\}$ in the 2-D space and the recovered 2-D representations from $\boldsymbol{x}_i \in \mathbb{R}^{100}$, where $\boldsymbol{x}_i = (\sigma(\boldsymbol{W}\boldsymbol{h}_i))^2$.

as follows:

$$\boldsymbol{x}_i = (\sigma(\boldsymbol{W}\boldsymbol{h}_i))^2 , \tag{3.10}$$

where $\sigma(\cdot)$ is the sigmoid function as before and $\boldsymbol{W} \in \mathbb{R}^{100\times 2}$ is similarly generated as in the paper. We perform elementwise squaring on the result features to further complicate the generating process. The corresponding results can be seen in Fig. 3.9 of this supplementary document. One can see that a similar pattern as we have observed in the main text is also presented here: The proposed DCN recovers a 2-D K-means-friendly space very well and the

other methods all fail.

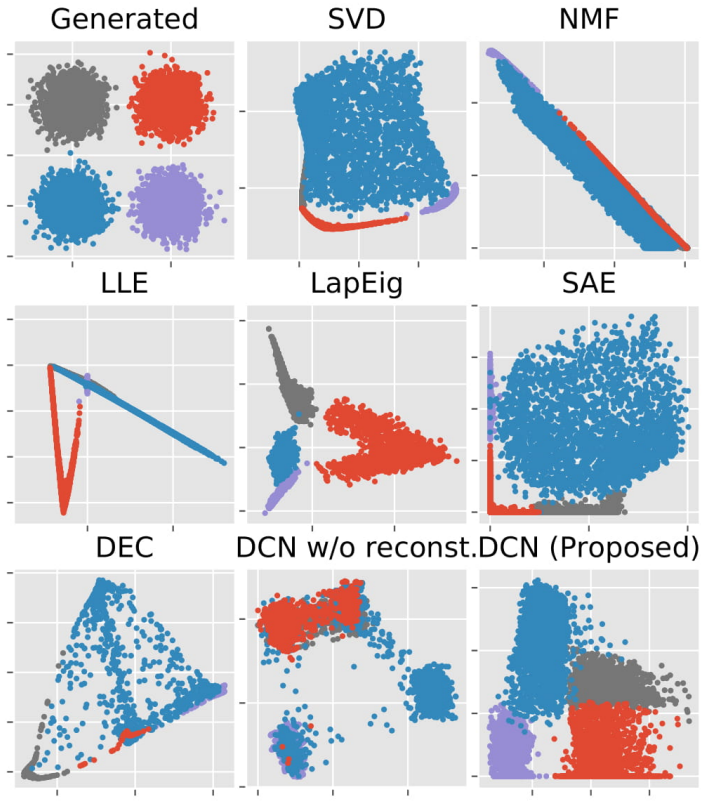


Figure 3.10: The generated latent representations $\{\boldsymbol{h}_i\}$ in the 2-D space of the recovered 2-D representations from $\boldsymbol{x}_i \in \mathbb{R}^{100}$, where $\boldsymbol{x}_i = \tanh\left(\sigma(\boldsymbol{W}\boldsymbol{h}_i)\right)$.

In Fig. 3.10, we test the algorithms under the generative model

$$\boldsymbol{x}_i = \tanh\left(\sigma(\boldsymbol{W}\boldsymbol{h}_i)\right), \tag{3.11}$$

where $\boldsymbol{W} \in \mathbb{R}^{100\times 2}$. Same as before, the proposed DCN gives very clear clusters in the recovered 2-D space.

The results in this section and the synthetic-data experiment presented in main text are encouraging: Under a variety of complicated nonlinear generative models, DCN can output clustering-friendly latent representations.

Table 3.5: Evaluation on the Pendigits dataset

Methods	DCN	SAE+KM	SC	KM
NMI	**0.69**	0.65	0.67	0.67
ARI	**0.56**	0.53	0.55	0.55
ACC	**0.72**	0.70	0.71	0.69

3.7.2 Additional real data experiments

Beside the real datasets in the paper, we also conduct experiment on the Pendigits dataset. The Pendigits dataset consists of 10,992 data samples. Each sample records 8 coordinates on a tablet, on which a subject is instructed to write the digits from 0 to 9. So each sample corresponds to a vector of length 16, and represents one of the digits. Note that this dataset is quite different from MNIST – each digit in MNIST is represented by an image (pixel values) while digits in Pendigits are represented by 8 coordinates of the stylus when a person was writing a certain digit. Since each digit is represented by a very small-size vector of length 16, we use a small network who has three forward layers which are with 16, 16, and 10 neurons. Table 3.5 shows the results: The proposed methods give the best clustering performance compared to the competing methods, and the methods using DNNs outperform the 'shallow' ones that do not use neural networks for DR.

Chapter 4

Learning Nonlinear Mixture Models

4.1 Introduction

Linear mixture models (LMMs) have found numerous applications in machine learning and signal processing, e.g., topic mining [60, 17], clustering, and source separation. When a LMM is used for parameter estimation, it is critical to ensure that the generative model is identifiable. This is crucial in many data mining problems [6, 49, 64, 94], as model identifiability is necessary for interpretability. However, a LMM is not identifiable in general – even in the ideal case without noise: a LMM boils down to matrix factorization (MF) that is known to be unidentifiable, unless additional constraints on the factors are imposed.

Identifiability research for LMMs has a long and fruitful history at the confluence of machine learning, statistics, and signal processing, see e.g., [16, 91, 42, 94, 46]. The arguably most notable line of work is independent component analysis (ICA) ([32, 70]), which was originally motivated by speech source separation. Statistical independence of latent parameters (i.e., signals or data streams corresponding to different sources) is exploited to establish identifiability in ICA. LMM unmixing with correlated latent parameters has also been extensively studied, e.g., in the context of *nonnegative matrix factorization* (NMF) [39, 81, 67, 44, 88, 94, 91], bounded

component analysis [34], and some other types of constrained MF models [45, 9].

Despite the relatively good understanding of the identifiability issues of different LMMs, the model is considered oversimplified in many applications. In many cases the observed data cannot be assumed to be linear mixtures of some basis vectors, since nonlinear distortions exist due to various reasons, for example, e.g., sensor saturation / clipping, or nonlinear amplification. A natural question then is: under a reasonable *nonlinear mixture model* (NMM), is it possible to identify the latent parameters of interest?

This question turns out to be highly nontrivial: most of the analytical tools in the linear mixture case do not apply. One exception is statistical independence of random variables. Based on this observation, many works [112, 1, 68, 69, 71] tackle nonlinear mixture model identification from a nonlinear ICA viewpoint. This line of work is elegant, but only partially answers our research question when the source signals are statistically independent. In many cases, statistical independence cannot be assumed, which is why there has been extensive study on correlated components / sources as mentioned above.

4.1.1 Contributions

In this work, we study the nonlinear mixture model learning problem, under a new setting that is very different from ICA. Specifically, we study a nonlinear mixture model where the observed data vectors are convex combinations of a set of basis vectors followed by nonlinear distortion.

Our specific contributions can be summarized as follows:

1. *Flexible generative model* We put forth a novel nonlinear mixture model, which is flexible enough to faithfully capture nonlinear effects naturally arising in many real world applications, e.g. hyperspectral unmixing [16] and MRI [121]. Specifically, the proposed model is related to a classical model in nonlinear ICA (nICA), while the independence assumption in nonlinear ICA is replaced with new assumptions from convex geometry. For source separation, our model can handle the case when sources are *dependent*, unlike

nICA.

2. *Identification criterion* We propose a model identification criterion for the considered problem and provide sufficient conditions under which the model is identifiable. The criterion, when its global optimality is achieved, guarantees that an inverse function (possibly scaled and shifted) of the unknown nonlinear distortion is learned, in a completely *unsupervised* fashion. Thus the nonlinear effects are rectified / equalized, and traditional methods for identifying LMMs can be applied subsequently. Our proof leverages perspectives from functional equations [40, 79], together with novel extensions of existing LMM identifiability results. This fortuitous union yields the right tools for our analysis, as we will see.

3. *Neural network-based implementation* We propose a neural network based formulation to implement the proposed criterion. The neural network acts as an adaptive feature extractor, which, after training with the proposed identification criterion, will be the (scaled) inverse function of the *unkown* nonlinear function in data generation. The employed neural network is judiciously designed, so as to ensure its invertibility, thus the specific constraints imposed by the identification criterion can be satisfied.

4. *Numerical validation* We reformulate the criterion into an easy-to-implement form and employ a trust region algorithm for solving the problem efficiently. We also test the algorithm on both synthetic and real data. In particular, the neural network is shown to be able to counter the nonlinear effects after training, and much better parameter estimation performance is observed. Evaluation on a real world hyperspectral image also confirms the effectiveness of the proposed theory and algorithm.

Another salient feature of our method is that it turns the *unsupervised* parameter estimation problem into a *supervised* regression problem, which requires little new algorithmic design – see Section 4.4 for more information.

4.1.2 Related work

The current work is closely related to the large body of work studying model identifiability. For a parametric statistical model, identifiability is a fundamental question which concerns whether the model parameters can be uniquely pinned down from observed data. In explanatory data analysis, where the goal is to explain / interpret the data, identifiability of the adopted model is critical, as otherwise there exist more than one model settings that fit the data equally well, in which case one does not even know which model to interpret [91, 49, 64]

Linear / Nonlinear Mixture Identifiability through statistical independence

ICA [32, 70] is a classical technique in unsupervised learning. Statistical independence and deviation from Gaussianity of components / sources are exploited to establish model identifiability. Several identification criteria have been proposed based on mutual information, entropy, as well as high-order cumulants, see e.g. [70, 53].

A crucial assumption in ICA is linear mixing: sources are mixed linearly by a so-called mixing matrix. This assumption can be over-simplified in some applications, and considerable effort has been put into developing methods that handle nonlinear effects in the framework of ICA. The work of [112, 1] put forth a post-nonlinear model, where on top of the classical assumption of a linear mixing system, a nonlinear transformation is also assumed. For this model, identifiability is established based on the assumptions of source independence, and some assumptions on the nonlinear function. A recent line of work on nonlinear mixture model includes [68, 69, 71], which exploit structures in the data, e.g. non-stationarity or temporal dependence, and form identification criterions tailored for these structures such that identifiability of sources is established.

Linear Mixture Identifiability through convex geometry

Another line of work establishes identifiability of LMM using geometric properties, instead of statistical properties as in ICA. To the best of our knowledge, this line of work originated in the hyperspectral imaging community, see e.g. [16, 91] and reference therein. In such applications, measurements of a geographical region on the ground are taken in multiple spectral bands, and the measurement for each pixel is assumed to be a *convex* combination of the spectral signatures of several materials, e.g., water and soil, present on the target ground. In these applications, it is of primary interest to identify the spectral signatures of the materials (known as "endmembers" in the remote sensing community), as well as those convex combination coefficients – in other words, identifiability of LMM is a central topic. Exploiting different model assumptions, considerable work has been done in this area, see [49, 91, 44, 88].

Interestingly, the same LMM also arises in probabilistic topic modeling, see [6, 4, 64]. Again, identifiability of LMM in this application is crucial, as one would like to identify the true topics.

4.1.3 Organization

The remainder of the paper is organized as follows. In Section 4.2, we review some preliminary materials that are essential for the presentation of this work, then introduce the considered nonlinear mixture model. In Section 4.3, we detail the main result of this work, i.e., the identifiability guarantees for the proposed nonlinear mixture model, as well as the existence of solution for the proposed identification criterion. In Section 4.4, we develop a companion learning method for the proposed identification criterion. Specifically, we put forth a neural network architecture that guarantees invertibility of the resulting mapping – which is crucial to implement the requirements set by the proposed identification criterion. We present numerical experiment results in Section 4.5. Conclusions are given in Section 4.6.

4.2 Problem statement

The new model considered in this work is a generalization of the LMM. In this section, we first briefly review the LMM and its related identifiability results, then we introduce the new model, and discuss its motivation. By design, the new model can be seen as having an extra nonlinear transformation added to the classical LMM. For this reason, in Section 4.3, we propose a natural two-stage learning strategy: in the first stage, our new method "removes" the nonlinear effects. After this step, the data model is reduced to LMM, and we employ existing techniques to deal with it in the second stage.

4.2.1 Review of linear mixture model

We briefly review existing parameter identification results concerning the LMM that are related to this work. Relevant concepts in convex geometry can be found in Appendix 4.7.1.

To facilitate discussion, we use $\Delta_M := \left\{\boldsymbol{x} | \boldsymbol{x} \in \mathbb{R}^M,\ \boldsymbol{x} \geq \mathbf{0},\ \mathbf{1}^\mathsf{T}\boldsymbol{x} = 1\right\}$ to denote the $(M-1)$-dimensional probability simplex. The LMM is defined as

$$\boldsymbol{x}_j = \boldsymbol{A}\boldsymbol{s}_j,\ \forall j \in [N], \tag{4.1}$$

where $\boldsymbol{A} \in \mathbb{R}^{M\times r}$ is often a tall matrix, i.e., $M > r$, and $\boldsymbol{s}_j \in \Delta_r$. Alternatively, we will also write $\boldsymbol{X} = \boldsymbol{A}\boldsymbol{S}$ by collecting all $\boldsymbol{x}_j$'s into $\boldsymbol{X}$, and $\boldsymbol{s}_j$'s into $\boldsymbol{S}$.

As mentioned earlier, the LMM finds applications in a plethora of scenarios: from signal processing tasks such as hyperspectral imaging [16, 91], MRI data analysis [121], to probabilistic topic modeling of text documents [17, 6, 64]. These important applications motivate the considerable amount of work studying its identifiability properties.

To recover factors $\boldsymbol{A}$ and $\boldsymbol{S}$ from data $\boldsymbol{X} = [\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N]$, the following so-called *Volume*

Minimization (VolMin, [33, 25, 44]) criterion is often employed:

$$\min_{\boldsymbol{B}\in\mathbb{R}^{M\times r},\boldsymbol{H}\in\mathbb{R}^{r\times N}} \ \mathrm{Vol}(\boldsymbol{B}) \\ \text{s.t. } \boldsymbol{X}=\boldsymbol{B}\boldsymbol{H},\ \boldsymbol{H}\geq\boldsymbol{0},\ \boldsymbol{H}^{\mathsf{T}}\boldsymbol{1}=\boldsymbol{1}, \tag{4.2}$$

where it is assumed that r is known. The term $\mathrm{Vol}(\boldsymbol{B})$ is a measure of the volume of the simplex formed by using columns of $\boldsymbol{B}$ as vertices, see [19, p. 408]. This criterion means that we want to find $\boldsymbol{B}$ and $\boldsymbol{H}$ that satisfy the LMM, and we pick the solution with the minimal volume, hence the name VolMin.

Several algorithms for dealing with (4.2) have been developed [15, 25, 88, 43], and we will use the so-called *minimum volume enclosing simplex* (MVES): Given data $\boldsymbol{X}$ and the rank parameter r, the MVES algorithm returns a solution $(\widehat{\boldsymbol{B}},\widehat{\boldsymbol{H}})$ of (4.2). We refer readers to [25] for more on MVES.

4.2.2 Proposed signal model

We consider the following signal model

$$\boldsymbol{x}_j=\boldsymbol{\phi}(\boldsymbol{A}\boldsymbol{s}_j),\ \forall j\in[N], \tag{4.3}$$

where $\boldsymbol{A}\in\mathbb{R}^{M\times r}$ satisfies $\boldsymbol{A}\geq\boldsymbol{0}$, and $\boldsymbol{s}_j\in\Delta_r,\ \forall j\in[N]$. The function $\boldsymbol{\phi}$ is a nonlinear mapping $\boldsymbol{\phi}:\mathbb{R}^M\to\mathbb{R}^M$, and we consider *element-wise* nonlinearity, i.e., $\boldsymbol{\phi}=[\phi_1,\phi_2,\cdots,\phi_M]^{\mathsf{T}}$, so that

$$\boldsymbol{\phi}(\boldsymbol{x})=[\phi_1(\boldsymbol{x}(1)),\cdots,\phi_M(\boldsymbol{x}(M))]^{\mathsf{T}}, \tag{4.4}$$

where $\boldsymbol{x}=[\boldsymbol{x}(1),\cdots,\boldsymbol{x}(M)]^{\mathsf{T}}$. For brevity, we use the shorthand notation $\boldsymbol{X}=\boldsymbol{\phi}(\boldsymbol{A}\boldsymbol{S})$ to denote (4.3), where it should be understood that the $\boldsymbol{\phi}$ is applied to each *column* of $\boldsymbol{A}\boldsymbol{S}$.

Model (4.3) is well motivated. It can be viewed as a generalization of (4.1), which is used in various applications. In hyperspectral unmixing (HU), each $\boldsymbol{x}_j$ is a hyperspectral pixel, each column of $\boldsymbol{A}$ represents the frequency signature of a certain material (e.g., soil, vegetation, water), and each $\boldsymbol{s}_j$ denotes the proportion of materials in that pixel $\boldsymbol{x}_j$, see e.g. [16, 91, 56, 38]. In MRI, LMM is used due to the so called "partial volume effect" [27, 121, 104], which gives rise to the condition $\boldsymbol{s}_j \in \Delta_r$. Both these applications are of great importance in their respective research fields, where considerable work has been done based on (4.1). Yet, it is widely recognized that in many real world scenarios, the LMM in (4.1) is oversimplified. For example, in HU, the measurements $\boldsymbol{x}_j$'s are obtained by sensors, which have inherent nonlinearity due to physical limitations of the measuring devices, see [16, 56, 38]. In MRI, empirical studies show that there exist (currently) unknown physical / biological processes, which cause nonlinear relationships between measurements (e.g., changes in cerebral blood flow) and the underlying signals of interest (brain network connectivity, brain local region energy consumption, etc.), see [115, 87, 86]. The existence of such nonlinear distortions in various problems has motivated us to impose the $\boldsymbol{\phi}$ function in (4.3) on top of the classical LMM in (4.1). This way, many nonlinear effects happening in practice yet ignored in the LMM literature may be captured.

However, it is clear that the additional *unknown* $\boldsymbol{\phi}$ brings considerable complication in recovering $\boldsymbol{A}$ and $\boldsymbol{S}$. Before pursuing a general result, let us make some simple observations. First, for many nonlinear $\boldsymbol{\phi}$, it is not possible to recover $\boldsymbol{A}$ and $\boldsymbol{S}$, e.g., $\boldsymbol{\phi}(\boldsymbol{x}) = \boldsymbol{0}$, $\forall\, \boldsymbol{x}$. Hence one of the tasks is to impose on $\boldsymbol{\phi}$ reasonable and practical conditions, under which recovery is possible. Second, if $\boldsymbol{\phi}$ is linear, by the element-wise assumption, we have $\boldsymbol{X} = \boldsymbol{DAS}$, where $\boldsymbol{D}$ is a diagonal matrix. From here, we can see that there are scaling ambiguities on the rows of $\boldsymbol{A}$, even for the simplest $\boldsymbol{\phi}$. In light of this, a crucial question about model (4.3) is which parts (or aspects) of $\boldsymbol{A}$ and $\boldsymbol{S}$ can be identified, and to what extend?

4.3 Identifiability analysis

As mentioned earlier, our overall identification strategy to tackle the new model has two stages. In the first stage, our method "removes" nonlinear effects. In the second stage, since the data model is reduced to the classical LMM, we adapt existing techniques to deal with it. In this section, we first present techniques to remove the nonlinearity; then in the last sub-section, we discuss a technique to identify the model after nonlinearity is dealt with.

4.3.1 A technical lemma

We aim at understanding the identifiability of (4.3), as well as designing a method to learn this model. It will be shown (see Theorem 1 and Lemma 2) that $\boldsymbol{S}$ can be identified uniquely, $\boldsymbol{A}$ can be identified up to linear transformation, and the inverse of $\boldsymbol{\phi}$ can be identified up to affine transformation. Towards identifying the model, we will try to learn an adjustable function $\boldsymbol{f}$, and denote

$$\boldsymbol{y}_j = \boldsymbol{f}(\boldsymbol{\phi}(\boldsymbol{A}\boldsymbol{s}_j)), \quad j \in [N]. \tag{4.5}$$

The remaining question is how to devise a learning method such that the resulting $\boldsymbol{f}$ will "counteract" the nonlinear effect brought by $\boldsymbol{\phi}$. If this can be done, we can then employ methods designed for LMM (4.1) to separate the latent factors. Towards this goal, we first introduce a technical lemma.

Consider the following functional equation concerning functions $\psi_1, \cdots, \psi_M$ and variables $\boldsymbol{s} \in \text{int } \Delta_r$

$$\sum_{i=1}^{M} \psi_i(\boldsymbol{a}_i^\mathsf{T}\boldsymbol{s}) = 1, \ \forall \boldsymbol{s} \in \text{int } \Delta_r, \tag{4.6}$$

where int Δ_r denotes the *interior* of Δ_r. To facilitate presentation, let $\boldsymbol{A} := [\boldsymbol{a}_1, \boldsymbol{a}_2, \cdots, \boldsymbol{a}_M]^\mathsf{T} \in$

$\mathbb{R}^{M\times r}$.

Lemma 1. *Suppose* (4.6) *holds, and* $M \geq r \geq 3$*. Let us further assume that (a) the functions* $\psi_1, \cdots, \psi_M$ *are twice differentiable, and are all convex (or all concave) in the domain* $(0, 1)$*; and (b)* $\boldsymbol{A}$ *is nonnegative and has two positive columns. Then the functions* $\psi_1, \cdots, \psi_M$ *are all affine.*

This lemma asserts that under some technical conditions, only affine functions satisfy (4.6). We use this result to learn a nonlinear function for each *element* of the nonlinear mixing function in (4.3), so that all the composite functions are affine, hence nonlinearity in the signal model is removed. The proof for Lemma 1 can be found in Appendix 4.7.3.

4.3.2 Nonlinear mixture model identification

To proceed, let us suppose that the learning function $\boldsymbol{f} : \mathbb{R}^M \to \mathbb{R}^M$ in (4.5) is also element-wise, i.e., $\boldsymbol{f} = [f_1, f_2, \cdots, f_M]^\mathsf{T}$, where f_i's are univariate functions. Denote $\boldsymbol{k} = [k_1, k_2, \cdots, k_M]^\mathsf{T} : \mathbb{R}^M \to \mathbb{R}^M$, where $k_i = f_i \circ \phi_i$, and $\circ$ denotes function composition. Let us make the following assumptions about the generative model (4.3).

(A1) The functions $\phi_1, \cdots, \phi_M$ are all invertible, and twice differentiable.

(A2) The matrix $\boldsymbol{A} \in \mathbb{R}^{M\times r}$ in (4.3) satisfies $\boldsymbol{A} \geq \boldsymbol{0}$, has two strictly positive columns, and is incoherent (see Definition 1). The dimensions satisfy $M \geq r \geq 3$.

(A3) The columns of $\boldsymbol{S}$ satisfy $\boldsymbol{s}_j \in \text{int } \Delta_r,\ \forall j \in [N]$.

Let us briefly discuss the roles of the assumptions. For (A1), the invertibility condition is important, as one in general cannot hope to recover the unknown parameters if they undergo non-invertible transformations. The twice differentiable condition on ϕ_i's requires the nonlinear functions in data generation to be smooth.

Assumption (A2) is the same as in Lemma 1, except for the additional incoherent assumption. Both conditions in (A2) are important in establishing the identifiability theory. For example, the incoherence assumption ensures that solutions that satisfy (4.7) exist; see detailed discussion in Section 4.3.3. We should mention that although (A2) seems technical, it is not hard to be satisfied; e.g., if $\boldsymbol{A}$ is generated from an absolutely continuous distribution, supported on the nonnegative orthant. In this case, both conditions are satisfied with probability one.

For brevity, let us define a matrix function that has $\boldsymbol{k}$ acting on the columns of its matrix argument, $\boldsymbol{T}_{\boldsymbol{k}}(\boldsymbol{X}) = [\boldsymbol{k}(\boldsymbol{x}_1), \boldsymbol{k}(\boldsymbol{x}_2), \cdots, \boldsymbol{k}(\boldsymbol{x}_N)]$ for $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_N]$. We are ready to state the following results.

Theorem 1. *(Main results) Under assumptions (A1), (A2), (A3), if $f_1, f_2, \cdots, f_M$ satisfy*

$$\sum_{i=1}^{M} k_i(\boldsymbol{a}_i^{\mathsf{T}} \boldsymbol{s}) = 1, \ \forall \boldsymbol{s} \in \textit{int}\ \Delta_r. \tag{4.7}$$

where $k_i = f_i \circ \phi_i$, and the composite functions k_i are all convex (or all concave), then the following hold

(a) The functions $k_1, k_2, \cdots, k_M$ are affine;

(b) The functions $\phi_1^{-1}, \cdots, \phi_M^{-1}$ are identified up to an affine transformation, i.e. $f_i(x) = d_i \phi_i^{-1}(x) + b_i, \ \forall i \in [M]$, where d_i's and b_i's are scalar constants.

The proof can be found in Appendix 4.7.4. A remark about function $\boldsymbol{T}_{\boldsymbol{k}}$ is in order.

Remark 2. *According to (a) in Theorem 1, we can write*

$$\boldsymbol{T}_{\boldsymbol{k}}(\boldsymbol{X}) = \boldsymbol{D}\boldsymbol{X} + \boldsymbol{b}\boldsymbol{1}_N^{\mathsf{T}}, \tag{4.8}$$

where $\boldsymbol{D} = diag(d_1, \cdots, d_M)$, and $\boldsymbol{b} = [b_1, \cdots, b_M]^{\mathsf{T}}$, and d_i and b_i are coefficients for the affine function k_i. Equation (4.8) *suggests that $\boldsymbol{T}_{\boldsymbol{k}}$ is an affine function in $\boldsymbol{X}$. However, we would*

like $\boldsymbol{T_k}$ to be linear *in $\boldsymbol{X}$, instead of affine, as later we show that it is possible to identify the LMM parameters under invertible linear transformation (Lemma 2).*

Fortunately, for data model (4.3) *satisfying (A1), (A2) and (A3), we can see that $\boldsymbol{T_k}(\boldsymbol{X})$ is indeed a linear function of $\boldsymbol{X}$ under mild conditions. Let us consider a matrix $\boldsymbol{X} \in \mathbb{R}^{M\times N}$. Due to equation* (4.7)*, we have $\mathbf{1}_M^\mathsf{T}\boldsymbol{T_k}(\boldsymbol{X}) = \mathbf{1}_M^\mathsf{T}\boldsymbol{D}\boldsymbol{X} + \mathbf{1}_M^\mathsf{T}\boldsymbol{b}\mathbf{1}_N^\mathsf{T} = \mathbf{1}_N^\mathsf{T}$, which means $\mathbf{1}_N^\mathsf{T} = \mathbf{1}_M^\mathsf{T}\boldsymbol{D}\boldsymbol{X}/(1-\mathbf{1}_M^\mathsf{T}\boldsymbol{b})$. Plugging this into the above equation, we have*

$$\begin{aligned}\boldsymbol{T_k}(\boldsymbol{X}) &= \boldsymbol{D}\boldsymbol{X} + \boldsymbol{b}\left(\frac{1}{1-\mathbf{1}_M^\mathsf{T}\boldsymbol{b}}\mathbf{1}_M^\mathsf{T}\boldsymbol{D}\boldsymbol{X}\right)\\ &= \left(\boldsymbol{I} + \frac{1}{1-\mathbf{1}_M^\mathsf{T}\boldsymbol{b}}\boldsymbol{b}\mathbf{1}_M^\mathsf{T}\right)\boldsymbol{D}\boldsymbol{X}\\ &= \boldsymbol{W}\boldsymbol{X}\end{aligned} \tag{4.9}$$

where we define $\boldsymbol{W} := \left(\boldsymbol{I} + \frac{1}{1-\mathbf{1}_M^\mathsf{T}\boldsymbol{b}}\boldsymbol{b}\mathbf{1}_M^\mathsf{T}\right)\boldsymbol{D}$, and $\mathbf{1}_M$ is an all-one vector of length M. The above equation suggests that $\boldsymbol{T_k}$ is linear *in $\boldsymbol{X}$. A subtle point is that the above calculation is invalid when $1 = \mathbf{1}_M^\mathsf{T}\boldsymbol{b}$ holds* exactly*, but this never happens in our extensive simulations..*

Remark 3. *Given the generative model* (4.3)*, Theorem 1 essentially asserts that if we require $\mathbf{1}^\mathsf{T}\boldsymbol{y} = 1$ for all input $\boldsymbol{s}$, then the learned functions $f_1, \cdots, f_M$ will remove the nonlinearity in functions $\phi_1, \cdots, \phi_M$. But our main goal is identifying parameters in the latent LMM; $\boldsymbol{T_k}$ being linear is not enough. To see this more clearly, suppose we get a solution for f_i's of this form*

$$f_i(x) = 1/M, \ \ \forall i \in [M]. \tag{4.10}$$

In this case, the k_i's are all constant functions, and hence convex. Moreover, for this solution (4.10)*, we have $\boldsymbol{k}(\boldsymbol{A}\boldsymbol{s}) = \boldsymbol{D}\boldsymbol{A}\boldsymbol{s} + \boldsymbol{b}$, where $\boldsymbol{D} = \mathbf{0}$ and $\boldsymbol{b} = (1/M)\mathbf{1}$; meaning that $\boldsymbol{f}$ maps all input $\boldsymbol{x} = \boldsymbol{\phi}(\boldsymbol{A}\boldsymbol{s})$ to the single point $\boldsymbol{y} = (1/M)\mathbf{1}$, which does satisfy* (4.7)*. The problem we exposed here is important: we need additional constraints on $\boldsymbol{y}$ beyond $\mathbf{1}^\mathsf{T}\boldsymbol{y} = 1$, so that $\boldsymbol{y}$ preserves information about the original data $\boldsymbol{x}$. Essentially, we require the f_i's to be invertible,*

see details in Section 4.4.

4.3.3 Existence of solutions

The results in Theorem 1 rely on equation (4.7). One could be wondering, given the conditions outlined in assumptions (A1), (A2), and (A3), does there exist $\boldsymbol{f}$ such that (4.7) holds? This amounts to studying feasibility of (4.7), which is not obvious. For instance, one might guess (incorrectly) that $\{\widehat{f}_i = \phi_i^{-1},\ \forall i\}$ satisfies (4.7). However, under such f_i's, $\sum_{i=1}^M k_i(\boldsymbol{a}_i^\mathsf{T}\boldsymbol{s}) = \sum_{i=1}^M \boldsymbol{a}_i^\mathsf{T}\boldsymbol{s} \neq 1$, unless we impose more assumptions on $\boldsymbol{A}$ or $\boldsymbol{S}$. This means, for this natural guess, (4.7) does not hold.

To study this feasibility issue, we first note that if there exists a diagonal matrix $\boldsymbol{D}$, such that $\mathbf{1}^\mathsf{T}\boldsymbol{D}\boldsymbol{A} = \mathbf{1}^\mathsf{T}$, then letting $\widetilde{f}_i = \phi_i^{-1}$, we have

$$\begin{aligned}\sum_{i=1}^M d_i \widetilde{f}_i(\phi_i(\boldsymbol{a}_i^\mathsf{T}\boldsymbol{s})) &= \sum_{i=1}^M d_i \boldsymbol{a}_i^\mathsf{T}\boldsymbol{s} \\ &= \mathbf{1}^\mathsf{T}\boldsymbol{D}\boldsymbol{A}\boldsymbol{s} \\ &= \mathbf{1}^\mathsf{T}\boldsymbol{s} \\ &= 1, \qquad \forall \boldsymbol{s} \in \text{int } \Delta, \end{aligned} \tag{4.11}$$

where d_i is the i-th diagonal element of $\boldsymbol{D}$. Hence, the functions $\left\{\widehat{f}_i(\cdot) = d_i\widetilde{f}_i(\cdot),\ i \in [M]\right\}$ satisfy (4.7). An additional requirement is that $\{d_i \neq 0, \forall i\}$, otherwise we can get a trivial solution, as explained in the above section.

Building on the above observation, the feasibility problem of (4.7) boils down to establishing existence of a *nonsingular* diagonal matrix $\boldsymbol{D}$ (i.e. $d_i \neq 0, \forall i$), such that $\mathbf{1}^\mathsf{T}\boldsymbol{D}\boldsymbol{A} = \mathbf{1}^\mathsf{T}$, for matrix $\boldsymbol{A}$ that satisfies assumption (A2). We present Proposition 1, which shows that with a mild incoherence condition (see Definition 1) on $\boldsymbol{A}$, such desired $\boldsymbol{D}$ indeed exists. We start by providing the following definition of incoherence.

Definition 1. *(Incoherence) A tall and full-rank matrix $\boldsymbol{A} \in \mathbb{R}^{m\times r}$ is a said to be incoherent if $\boldsymbol{e}_j \notin$ Range$(\boldsymbol{A})$, $\forall j \in [m]$.*

Note that here incoherence is defined in the same spirit as the incoherence found in well-known compressed sensing literature, see e.g. [24]. The physical meaning of incoherence is that the energy of each column is widespread across the rows.

We are now ready to state the following proposition. Here we write $\boldsymbol{A}^\mathsf{T}\boldsymbol{d} = \mathbf{1}$ instead of $\mathbf{1}^\mathsf{T}\boldsymbol{D}\boldsymbol{A} = \mathbf{1}^\mathsf{T}$ for conciseness: existence of nonsingular diagonal $\boldsymbol{D}$ is the same as existence of fully dense vector $\boldsymbol{d}$.

Proposition 1. *For a matrix $\boldsymbol{A} \in \mathbb{R}^{m\times r}$, assume that $m > r$, and $\boldsymbol{A}$ has full column-rank. Also assume that $\boldsymbol{A}$ is incoherent. There exists a vector $\boldsymbol{d} \in \mathbb{R}^m$, such that*

$$\boldsymbol{A}^\mathsf{T}\boldsymbol{d} = \mathbf{1}, \tag{4.12a}$$

$$\|\boldsymbol{d}\|_0 = m. \tag{4.12b}$$

Note that by assumption, $\boldsymbol{A}$ is tall and full rank, so there are infinitely many $\boldsymbol{d}$ vectors that satisfy (4.12a). However, it is not obvious if there is always a *fully dense* $\boldsymbol{d}$ (i.e., (4.12b)) such that (4.12a) holds for any $\boldsymbol{A}$ that is tall and full rank. For example, consider the matrix (which does not satisfy the incoherence condition)

$$\boldsymbol{A} = \begin{bmatrix} 1 & 1 \\ 0 & 0 \\ 1 & 0 \end{bmatrix}. \tag{4.13}$$

This matrix is tall and full rank. Let $\boldsymbol{d} = [d_1, d_2, d_3]^\mathsf{T}$, and from $\boldsymbol{A}^\mathsf{T}\boldsymbol{d} = \mathbf{1}$, we have

$$\begin{aligned} d_1 + d_3 &= 1 \\ d_1 &= 1, \end{aligned}$$

which means $d_3 = 0$ when $\boldsymbol{A}^\mathsf{T}\boldsymbol{d} = \mathbf{1}$. This suggests there does not exist a *fully dense* $\boldsymbol{d}$ for the particular $\boldsymbol{A}$ in (4.13), such that $\boldsymbol{A}^\mathsf{T}\boldsymbol{d} = \mathbf{1}$. The proof of Proposition 1 can be found in Appendix 4.7.5.

Remark 4. *We established that for an incoherent $\boldsymbol{A}$, there always exist solutions to make* (4.7) *hold. Moreover, we point out that even for some $\boldsymbol{A}$ that is not incoherent, solutions for* (4.7) *might also exist. For example, if one or more columns of $\boldsymbol{A}$ are some columns of an identity matrix, then $\boldsymbol{A}$ is not incoherent. However, if we have $\mathbf{1}^\mathsf{T}\boldsymbol{A} = \mathbf{1}^\mathsf{T}$ – which is true when all columns of $\boldsymbol{A}$ are some columns of an identity matrix – then we see that $\{f_i = \phi_i^{-1},\ \forall i\}$ is a feasible solution.*

4.3.4 Parameter identification after removing nonlinearity

To proceed with parameter learning, let us provide the following lemma, concerning parameter identifiability of LMM (4.1) under a linear transformation. This is needed in the second stage of our learning strategy, where nonlinearity is removed, and only a linear transformation of the LMM is remaining.

Lemma 2. *Consider the LMM model $\boldsymbol{X} = \boldsymbol{A}\boldsymbol{S}$, where $\boldsymbol{A} \in \mathbb{R}^{M\times r}$ and $\boldsymbol{S} \in \mathbb{R}^{r\times N}$ satisfies the SS condition (cf. Appendix 4.7.2), and rank$(\boldsymbol{A})$ = rank$(\boldsymbol{S}) = r$. Let $\boldsymbol{Y} = \boldsymbol{W}\boldsymbol{X}$, where $\boldsymbol{W} \in \mathbb{R}^{M\times M}$ is nonsingular. Then we can identify $\widetilde{\boldsymbol{A}} = \boldsymbol{W}\boldsymbol{A}$ and $\boldsymbol{S}$ up to column permutation by solving*

$$\min_{\boldsymbol{B}\in\mathbb{R}^{M\times r},\boldsymbol{H}\in\mathbb{R}^{r\times N}} \ \mathit{Vol}(\boldsymbol{B}) \quad \text{s.t. } \boldsymbol{Y} = \boldsymbol{B}\boldsymbol{H},\ \boldsymbol{H} \geq \mathbf{0},\ \boldsymbol{H}^\mathsf{T}\mathbf{1} = \mathbf{1}. \tag{4.14}$$

That is, if $(\boldsymbol{B}^*, \boldsymbol{H}^*)$ is an optimal solution of the above problem, then $\boldsymbol{B}^* = \widetilde{\boldsymbol{A}}\boldsymbol{\Pi}$ and $\boldsymbol{H}^* = \boldsymbol{\Pi}^\mathsf{T}\boldsymbol{S}$, where $\boldsymbol{\Pi}$ is a permutation matrix.

This lemma is a direct consequence of Theorem 2 (in Appendix 4.7.2, see also [44]). It suggests that when the original model $\boldsymbol{X} = \boldsymbol{AS}$ is identifiable, then after an invertible linear transformation $\boldsymbol{W}$, we can still identify $\boldsymbol{S}$ using VolMin; but it is not possible to identify $\boldsymbol{A}$ due to the linear transformation $\boldsymbol{W}$. This lemma also suggests that we can employ an algorithm designed to tackle LMM to identify $\boldsymbol{S}$, once the nonlinear effects in (4.3) have been removed, and only an unknown linear transformation is left.

4.4 Learning algorithm

Theorem 1 suggests the following optimization formulation to learn the desired $\boldsymbol{f}$

$$\begin{aligned} &\text{find} \;\; f_1, \cdots, f_M \\ &\text{s.t.} \;\; f_i \circ \phi_i \text{ is all convex (or all concave) } \forall i \in [M], \\ &\quad \sum_{i=1}^{M} f_i(\boldsymbol{x}_j(i)) = 1 \; \forall j \in [N]. \end{aligned} \tag{4.15}$$

For this formulation we have the following claim.

Corollary 1. *For problem* (4.15)*, suppose the data* $\boldsymbol{X} = [\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N] \in \mathbb{R}^{M \times N}$ *admit model* (4.3) *and assumptions (A1), (A2), (A3) hold. Assume further that* $\boldsymbol{s}_j$*'s are sampled from a Dirichlet distribution. When* $N \to +\infty$*, the optimal solutions to* (4.15) *satisfy* (4.7)*, and the resulting* $\{k_i = f_i \circ \phi_i, \; \forall i \in [M]\}$ *are all affine.*

This corollary follows from the distributional assumption on $\boldsymbol{s}_j$. As $N \to +\infty$, $\boldsymbol{s}_j$ will cover all the interior of Δ_r with probability 1, since every point in Δ_r has nonzero probability. Then the constraints in (4.15) become the same as the conditions in Theorem 1. Corollary 1 thus guarantees the nonlinear function identification property of formulation (4.15) in an *asymptotic* sense. It also suggests that the proposed method should be able to benefit from a large number of samples. In the following, we approximate problem (4.15) to make it amenable to numerical

algorithms.

There may be a number of ways to enforce the constraint that k_i's are all convex (or all concave). For example, we note

$$k_i''(x) = f_i''(\phi_i(x))[\phi_i'(x)]^2 + f_i'(\phi_i(x))\phi_i''(x). \tag{4.16}$$

To make sure k_i is convex (or concave), we need $k_i''(x) \geq 0$ (or $k_i''(x) \leq 0$). This can be done if we somehow know the sign of ϕ_i. For instance, if we know *a priori* that the nonlinear distortion is a concave function, i.e., $\phi_i'' \leq 0$, then we can pick a parametric family for f_i's, such that $f_i''(x) \geq 0$ and $f_i'(x) \leq 0$. Then we have $k_i''(x) \geq 0$, i.e. k_i is convex. Similarly, we can constrain f_i's for all $i \in [N]$ to make sure k_i's are all convex (or concave). To simplify implementation, we adopt an approximation: We only require f_i's to be invertible in this work. This leads to the following optimization problem.

$$\begin{aligned} &\text{find } \ f_1, \cdots, f_M \\ &\text{s.t. } \ f_i \text{ is invertible } \forall i \in [M], \\ &\quad \sum_{i=1}^{M} f_i(\boldsymbol{x}_j(i)) = 1 \ \forall j \in [N]. \end{aligned} \tag{4.17}$$

In other words, we aim at learning *invertible* functions that add to one. The invertibility condition is crucial, otherwise we can obtain trivial solutions, as explained before.

To parametrize functions f_j, we will adopt Neural Networks (NN) with one hidden layer, due to their universal approximation capability [61, 10]. In particular, we employ the following parametric function family

$$\mathcal{F} = \left\{ f \middle| f(x) = \sum_{k=1}^{K} \alpha_k \sigma(\beta_k x + \gamma_k) + \delta_k, \alpha_k > 0, \ \beta_k > 0, \ \forall k \in [K] \right\} \tag{4.18}$$

where K is the number of neurons, $\{\alpha_k, \beta_k, \gamma_k, \delta_k\}_{k=1}^K$ are the learnable parameters of this NN, and σ denotes the nonlinearity. Importantly, the constraints on α_k and β_k are to ensure invertibility, as stated below.

Lemma 3. *In* (4.18)*, if* $\sigma'(x) > 0,\ \forall x$*, the functions in* $\mathcal{F}$ *are all invertible.*

The above lemma can be easily seen to be true. By definition, we have

$$f'(x) = \sum_{k=1}^{K} \alpha_k \beta_k \sigma'(\beta_k x + \gamma_k).$$

For $\sigma'(x) > 0$, we have $f'(x) > 0$ if $\alpha_k >,\ \beta_k > 0,\ \forall k \in [K]$. Note that the requirement for $\sigma'(x) > 0$ is easily satisfied for commonly used neurons, e.g., $\tanh(\cdot)$ and the sigmoid function. For this reason, we set σ to be $\tanh(\cdot)$ in this work.

Utilizing the parametric family $\mathcal{F}$ in (4.18), we arrive at the following optimization problem

$$\begin{aligned} \min_{\substack{\{\alpha_k^i, \beta_k^i,\\ \gamma_k^i, \delta_k^i\}}} \ & \frac{1}{N} \sum_{j=1}^{N} \left(1 - \sum_{i=1}^{M} \sum_{k=1}^{K} \alpha_k^i \sigma(\beta_k^i \boldsymbol{x}_j(i) + \gamma_k^i) + \delta_k^i \right)^2 \\ \text{s.t. } & \alpha_k^i > 0,\ \beta_k^i > 0, \quad \forall k \in [K],\ i \in [M]. \end{aligned} \tag{4.19}$$

This is a nonlinear least-squares regression problem, with bound constraints. We employ a trust-region algorithm [31] for optimization.

After obtaining parameters $\{\widehat{\alpha}_k^i, \widehat{\beta}_k^i, \widehat{\gamma}_k^i, \widehat{\delta}_k^i\}$ via (4.19), we obtain $\widehat{f}_i(x) = \sum_{k=1}^K \widehat{\alpha}_k^i \sigma(\widehat{\beta}_k^i x + \widehat{\gamma}_k^i) + \widehat{\delta}_k^i$, and form the transformed data $\boldsymbol{Y} = \widehat{\boldsymbol{f}}(\boldsymbol{X})$. Theorem 1 predicts that $\boldsymbol{Y} \approx \boldsymbol{W}\boldsymbol{A}\boldsymbol{S}$ for some nonsingular matrix $\boldsymbol{W}$. From Lemma 2, we see that we can employ an algorithm for LMM to identify $\boldsymbol{S}$. For this purpose, we employ the classical MVES algorithm [25] for LMM, and obtain an estimate $\widehat{\boldsymbol{S}}$.

The overall procedure is summarized in Algorithm 2. We emphasize again that the method is *unsupervised*: the only training data is $\boldsymbol{X}$, not $\{\boldsymbol{x}_j, y_j\}_{j=1}^N$ (feature-label pairs) as in, e.g.,

Algorithm 2 Nonlinear mixture learning.

Require: Data $\boldsymbol{X} \in \mathbb{R}^{M \times N}$, number of neurons K, latent dimension r
Ensure: Learned functions $\widehat{f}_1, \cdots, \widehat{f}_M$, estimated $\widehat{\boldsymbol{S}}$
1: Learn parameters $\{\widehat{\alpha}_k^i, \widehat{\beta}_k^i, \widehat{\gamma}_k^i, \widehat{\delta}_k^i\}$ by solving (4.19)
2: Form functions $\widehat{f}_1, \cdots, \widehat{f}_M$ by $\widehat{f}_i(x) = \sum_{k=1}^K \widehat{\alpha}_k^i \sigma(\widehat{\beta}_k^i x + \widehat{\gamma}_k^i) + \widehat{\delta}_k^i$
3: Obtain transformed data by applying the learned functions on input data: $\boldsymbol{Y} = \widehat{\boldsymbol{f}}(\boldsymbol{X})$
4: Obtain $\widehat{\boldsymbol{S}}$ by calling MVES$(\boldsymbol{Y}, r)$
5: **return** $\widehat{f}_1, \cdots, \widehat{f}_M, \widehat{\boldsymbol{S}}$

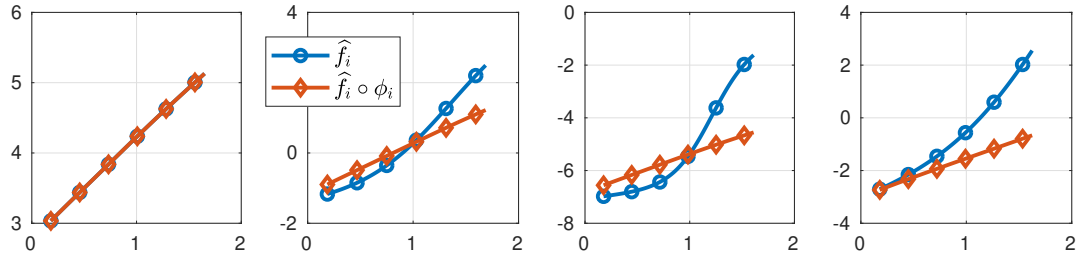


Figure 4.1: Learned functions ($\widehat{f}_i$'s, as parametrized by the neural network) and their composition with the ground truth nonlinear functions used for data generation. The four functions for data generation are $\phi_1(x) = x$, $\phi_2(x) = \sqrt{x}$, $\phi_3(x) = \sqrt[4]{x}$, $\phi_4(x) = \log(x+1)$. The ϕ_i's are kept secret in the learning stage.

the generalized additive models [53, Ch. 9] setting, or recent works on nonlinear estimation [134, 29].

4.5 Numerical experiments

We evaluate the presented theoretical results, as well as the learning algorithm in this section. To probe different aspects of the proposed method, we design and execute several numerical experiments, leveraging controlled synthetic data sets, as well as a real-world data set.

4.5.1 Synthetic data study

We start by providing a qualitative assessment of the proposed theory and algorithm. For this purpose, we will visualize the learned functions to see if nonlinearity in data generation is indeed

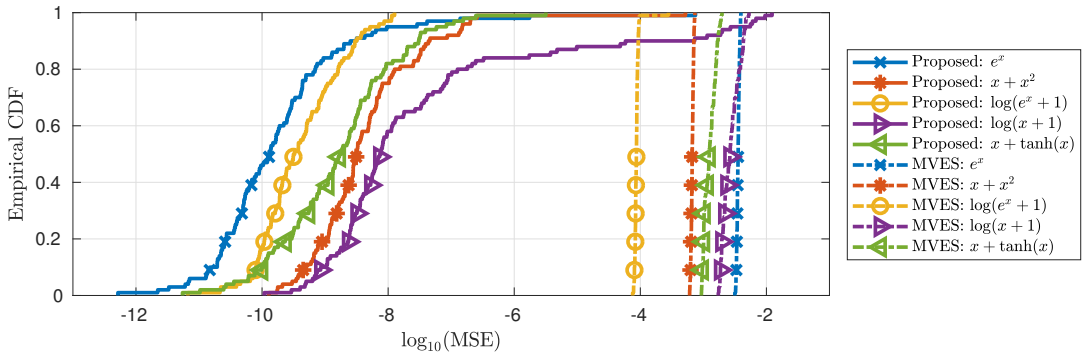


Figure 4.2: Empirical CDF of MSE: the legend shows the learning method, and the nonlinear function used in data generation. For each nonlinear function, 100 trials are generated. A point $(-6, 0.99)$ on a curve means the corresponding learning method yields MSE $\leq 10^{-6}$ in 99% of the 100 trials.

resolved. We randomly generate $\boldsymbol{S}$ according to a Dirichlet distribution – such that the generated $\boldsymbol{s}_j$'s are nonnegative and sum to one. The dimensions are $M = r = 4$ and $N = 1000$. The parameter of this Dirichlet distribution is set to $\boldsymbol{\mu} = [0.1, 0.1, 0.1, 0.1]$, so that the generated $\boldsymbol{s}_j$'s are well spread on the probability simplex, hence the sufficiently scattered condition (cf. Appendix 4.7.2) is likely to be satisfied. For this experiment, we take $\boldsymbol{A}$ to be $\boldsymbol{A} = 2\boldsymbol{I}_4$. The four nonlinear functions in data generation are $\phi_1(x) = x$, $\phi_2(x) = \sqrt{x}$, $\phi_3(x) = \sqrt[4]{x}$, and $\phi_4(x) = \log(x + 1)$. Note that these functions are *not* revealed to the learning algorithm, and are only used to visualize the results after learning is completed. For learning, each function f_i is parametrized by a constrained one-hidden-layer NN defined in (4.18), with $K = 20$ neurons. The learned functions $f_1 \cdots f_4$ and the composite functions $f_1 \circ \phi_1 \cdots f_4 \circ \phi_4$ are shown in Figure 4.1.

One can immediately see that the learned functions indeed resolve nonlinearity in data generating nonlinear functions: the learned f_1 is a linear function since ϕ_1 is a linear function; the other learned functions are all visually similar to the corresponding inverse functions of ϕ_i's. Moreover, one can clearly see that the composite functions all look affine.

Next, we test the parameter estimation performance. For this experiment, we generate data

with five different nonlinear functions: (a) e^x, (b) $x + x^2$, (c) $\log(e^x + 1)$, (d) $\log(x + 1)$, (e) $x + \tanh(x)$. For each case, one of the five functions is used for *all* coordinates (features), i.e. $\phi_1 = \cdots = \phi_M$. The parameter settings are $M = 10$, $N = 1000$, and $r = 4$. We generate $\boldsymbol{A} \in \mathbb{R}^{10\times 4}$ by sampling an i.i.d. normal distribution, and then take the absolute values, followed by a column normalization step. $\boldsymbol{S}$ is similarly generated as in the first experiment. For this experiment, the f_i functions are constrained to be the same: a constrained one-hidden-layer NN defined in (4.18), with $K = 40$ for all cases, to avoid unrealistic parameter tuning. In other words, all the NN share the same parameters. Since problem (4.19) is nonconvex, different initialization could lead to different results. For this reason, the formulation (4.19) is optimized five times with different random initialization, and the result of *smallest cost function value* is used for subsequent steps of Algorithm 1. The performance metric we employ is mean squared error (MSE): MSE $= \frac{\|\widehat{\boldsymbol{S}}-\boldsymbol{S}\|_F^2}{rN}$.

Since our method is the first work dealing with this nonlinear model, the only baseline we can employ is MVES without considering nonlinear effects. For each setting, 100 trials with different randomly generated data are performed, and the empirical cumulative distribution function (CDF) of the resulting MSEs is reported in Figure 4.2.

From Figure 4.2, one can see that the proposed method yields significant improvements over applying MVES directly, in all the cases. Note that the x-axis in Figure 4.2 is $\log_{10}(\text{MSE})$, hence our method yields several orders of magnitude improvement in accuracy over the baseline.

In the third experiment, we examine the estimation accuracy of matrix $\boldsymbol{S}$ with different number of data samples – as our theory offers asymptotic guarantees, we'd like to examine how the learning method performs in the case of finite data. The setting is similar to above: we generate $\boldsymbol{A}$ randomly, with $M = 10$ and $r = 4$. Then different number of data samples (columns of $\boldsymbol{S}$) are generated following the Dirichlet distribution. For each setting, we generate 50 data sets, and measure the resulting MSE of estimating $\boldsymbol{S}$. Two nonlinear functions, $\phi(x) = e^x$ and $\phi(x) = \log(x + 1)$ are employed in data generation. For all these experiments, the number of

neurons in f is fixed to $K = 40$.

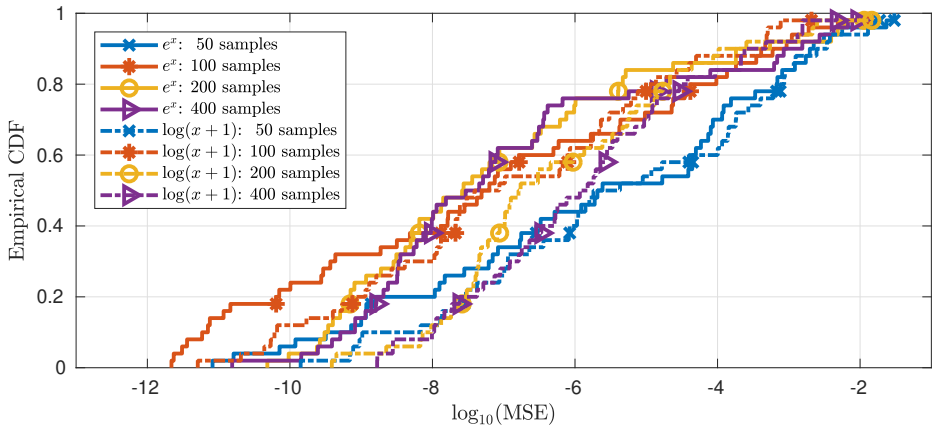


Figure 4.3: Empirical CDF of MSE of estimating $\boldsymbol{S}$ with different number of data samples.

The results are presented in Figure 4.3. As can be seen, when the number of data samples is small, larger MSEs of estimating $\boldsymbol{S}$ are observed. However, much better estimation performance is achieved with more data. Note also that the results presented in Figure 4.3 are relatively worse than those in Figure 4.2, which is mainly due to the small amount of available data.

In the next experiment, we vary the number of neurons in the neural network. The setting is similar to above: we generate $\boldsymbol{A}$ randomly, with $M = 10$ and $r = 4$. We draw columns of $\boldsymbol{S}$ by following a Dirichlet distribution, where the number of columns is $N = 1000$. The nonlinear function in data generation is $\phi(x) = e^x$. In total 50 data sets are generated. We run Algorithm 2 for different number of neurons K. The results are presented in Figure 4.4. As can be seen from the figure, very good estimation results can be achieved even when there are only 4 neurons, and better results are obtained when we employ more neurons in the network. However, using only 1 neuron does not work: in many cases we manually examined, the learned $\widehat{\boldsymbol{f}}(\boldsymbol{X})$ is almost constant – the entries are all the same, and no meaningful $\boldsymbol{S}$ can be estimated. This can be explained by the fact that a single neuron is unlikely to be able to approximate the (scaled) inverse of $\boldsymbol{\phi}$.

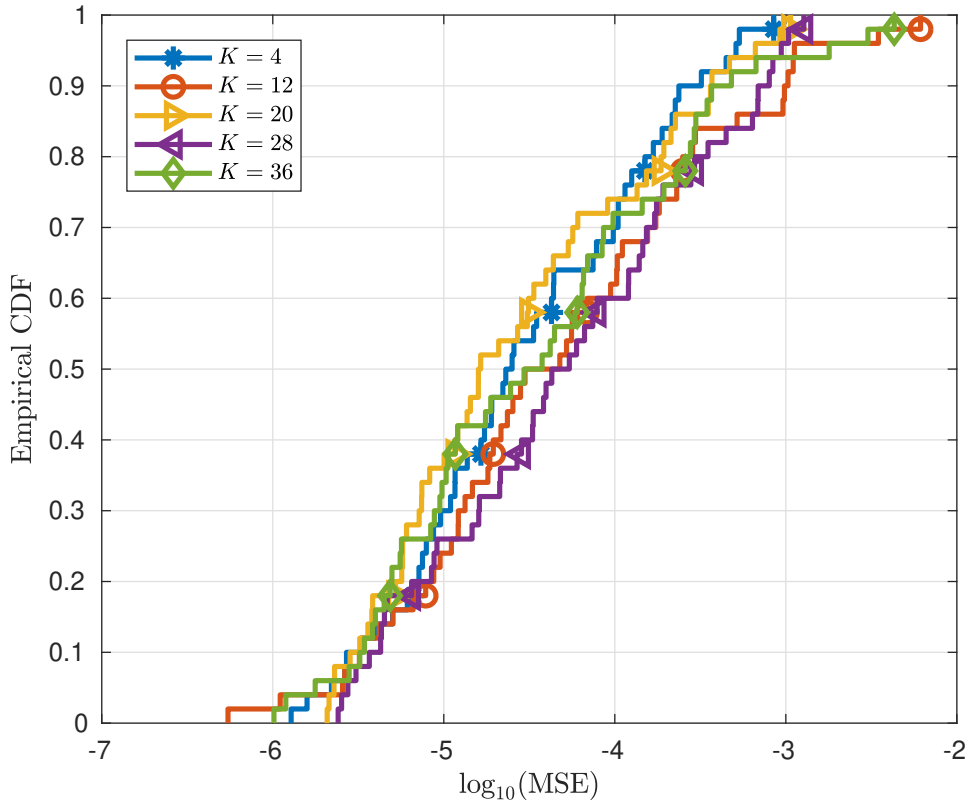


Figure 4.4: MSE results with different number of neurons.

In the last synthetic data experiment, we examine the estimation performance for different (but known) rank in the latent space. As before, we set $M = 10$, and $N = 1000$. The number of neurons are set to $K = 15$. We use $\phi(x) = e^x$ in data generation, and 50 data sets are generated for each rank setting. The results are presented in Figure 4.5. As can be seen from this figure, the estimation performance is insensitive to the different rank settings.

4.5.2 Case study with a hyperspectral image

We next perform an experiment on hyperspectral unmixing (HU). Unlike normal RGB images, a pixel in a hyperspectral image contains information on hundreds of spectral bands. With the

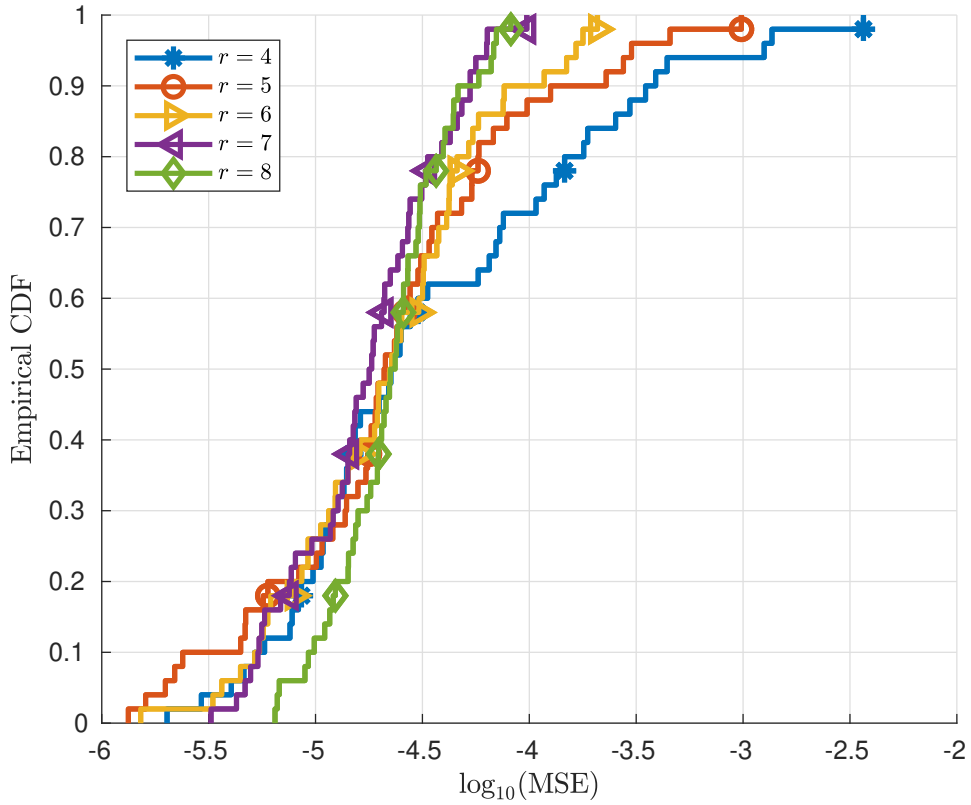


Figure 4.5: MSE results with different rank

more detailed spectral information, it is often assumed that different materials have their distinct spectral signatures (see e.g., [16] for details). Physically, each pixel is represented by a convex combination of materials that are present in that geographical patch. However, it is known that the collected measurement may encounter nonlinear distortion. The HU task involves identifying the spectra of the materials in a given region and the material concentrations in the different pixels (patches of the scene).

The image employed in this experiment is the Moffett Field captured in California, USA. The region has three main materials: water, soil, and vegetation. This scene is known for the existence of nonlinear mixture pixels, which usually poses a challenge to LMM-based HU algorithms such

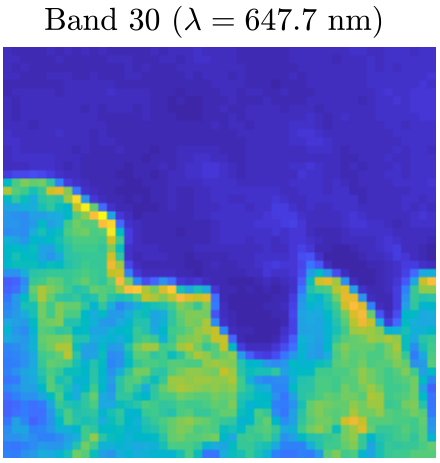


Figure 4.6: The Moffett Field hyperspectral image.

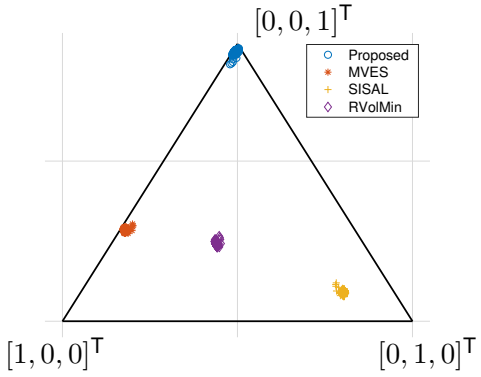


Figure 4.7: Visualizing columns of the estimated $\boldsymbol{S}$ corresponding to the water region.

as MVES. The size of the image is 50×50, hence we have 2500 pixels.

Each pixel is measured on 224 spectral bands, and we remove the water-absorbing bands following standard preprocessing [43], and end up with 200 spectral bands. In this case, each *pixel* $\{\boldsymbol{x}_j,\ j \in [2500]\}$ is measured as a vector of length 200, i.e. $\boldsymbol{x}_j \in \mathbb{R}^{200}$. A single band of the image is shown in Figure 4.6: the top part is water, while the bottom part is land (sand and vegetation). To apply our method, we use the same f_i on each of the 200 spectral bands, and fix $K = 40$. The latent dimension $r = 3$ is used since we know there are 3 materials in this scene. For comparison, in addition to MVES, we employ another two baselines that tackle

the nonlinearity problem from a robust LMM identification viewpoint: SISAL [15] and robust volume-minimization (RVolMin) [43].

We plot the estimated $\boldsymbol{S}$ in the known water region (top 15×50 part of Figure 4.6). We take this part to compare performance of the algorithms as it is clear that there is only one material (water) in this region, so the ground truth for each $\widehat{\boldsymbol{s}}_j$ is a permutation of $[1, 0, 0]^{\mathsf{T}}$, whereas the ground truth $\boldsymbol{s}_j$ for other regions is unavailable and hard to obtain. Since the resulting columns of $\widehat{\boldsymbol{S}}$ live in a dimension-2 simplex, we project all the points into a 2D space, with the vetices of the triangle corresponding to the original vetices in the 3D space, as shown in Figure 4.7.

From Figure 4.7, we see that results of the proposed method coalesce around a coordinate vector $[0, 0, 1]^{\mathsf{T}}$, which is almost identical to the ground truth. However results of the other methods are far away from any coordinate vector. This shows that by explicitly incorporating nonlinearity, the proposed method yields much improved results over prior art.

4.6 Chapter summary

In this chapter we proposed a novel and well-grounded nonlinear extension of the LMM data model, to account for nonlinear effects that naturally arise in real-world applications. The proposed model augments a widely used model, by allowing additional nonlinear distortions. It is an important problem to consider in practice, but a concrete study was sorely missing prior to this work. Much to one's surprise, the seemingly impossible mission of figuring out unknown nonlinearities in the new data model can actually be accomplished up to affine transformations, as we show in this paper. A learning algorithm leveraging the power of neural networks was proposed to equalize the unknown nonlinear functions. Numerical experiments show clear advantages of the proposed method over classic LMM learning algorithms when nonlinearity is present.

4.7 Appendix

4.7.1 Some definitions in convex geometry

Definition 2. *(Convex cone) The convex cone of* $\{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N\}$ *is defined as*

$$cone\{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N\} = \left\{ \boldsymbol{x} \middle| \boldsymbol{x} = \sum_{j=1}^{N} \boldsymbol{x}_j \theta_j,\ \theta_j \geq 0, \forall j \in [N] \right\}. \tag{4.20}$$

Definition 3. *(Convex hull) The convex hull of* $\{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N\}$ *is defined as*

$$conv\{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N\} = \left\{ \boldsymbol{x} \middle| \boldsymbol{x} = \sum_{j=1}^{N} \boldsymbol{x}_j \theta_j,\ \sum_{j=1}^{N} \theta_j = 1,\ \theta_j \geq 0, \forall j \in [N] \right\}. \tag{4.21}$$

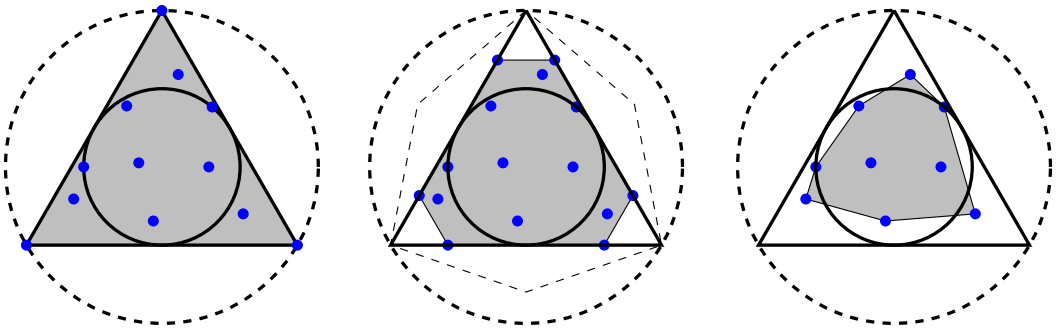
Definition 4. *(Simplex) A convex hull conv*$\{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N\}$ *is called a simplex if* $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N$ *are affinely independent, i.e.,* $\boldsymbol{x}_1 - \boldsymbol{x}_N, \cdots, \boldsymbol{x}_{N-1} - \boldsymbol{x}_N$ *are linearly independent.*

A probability simplex is a special simplex, with all vertex vectors being the coordinate vectors, i.e. $\forall i \in [N],\ \boldsymbol{x}_i = \boldsymbol{e}_j$ for some j, where $\boldsymbol{e}_j$ has 1 at its j-th coordinate, and 0 for all other coordinates.

4.7.2 Identifiability of LMM

We include some identifiability results for LMM, as they are important prior art that we build upon. In order to characterize identifiability of (4.1), let us introduce the following definition.

Definition 5. *(Sufficiently scattered, [44, 64]) Let matrix* $\boldsymbol{S} \in \mathbb{R}_+^{r \times N}$*, where* $\mathbb{R}_+^{r \times N}$ *is the nonnegative subset of* $\mathbb{R}^{r \times N}$*. Matrix* $\boldsymbol{S}$ *is said to be sufficiently scattered (SS) if cone*$(\boldsymbol{S})$ *satisfies: (a)* $\mathcal{C} \subseteq$ *cone*$(\boldsymbol{S})$*, where* $\mathcal{C}$ *is a second order cone:* $\mathcal{C} = \{\boldsymbol{x} \in \mathbb{R}^r | \mathbf{1}^\mathsf{T} \boldsymbol{x} \geq \sqrt{r-1}\|\boldsymbol{x}\|_2\}$*, (b) cone*$(\boldsymbol{S}) \subsetneq$ *cone*$(\boldsymbol{Q})$*, for any unitary matrix* $\boldsymbol{Q} \in \mathbb{R}^{r \times r}$ *that is not a permutation matrix.*



(a) Separability (b) Sufficiently scattered (c) Not sufficiently scattered

Figure 4.8: A geometric illustration of the sufficiently scattered condition (middle), a special case called separability (left), and a case that is not sufficiently scattered (right).

An illustration for $r = 3$ of this definition is given in Figure 4.8: We are viewing from the nonnegative orthant towards the origin. The 3 vertices of the triangle correspond to coordinates $[1, 0, 0],\ [0, 1, 0],\ [0, 0, 1]$, thus the triangle is the simplex where all columns of $\boldsymbol{S}$ reside. The inner circle corresponds to $\mathcal{C}$. The sufficient scattered condition, shown in Figure 4.8b, requires columns of $\boldsymbol{S}$ to be well-scatted on the simplex, so that cone$(\boldsymbol{S})$ encloses $\mathcal{C}$. This condition is in fact fairly relaxed, as discussed in [63].

We should also mention that a stronger condition on $\boldsymbol{S}$, termed "separability", which is shown in Figure 4.8a, can also help establish identifiability of LMM. We refer the interested readers to [26, 49, 5]. Here we present SS, as it is more relaxed than the separability condition.

Based on this VolMin criterion, the following theorem established identifiability of model(4.1).

Theorem 2. *[44] Let the matrices $\boldsymbol{A}$ and $\boldsymbol{S}$ satisfy rank$(\boldsymbol{A})$ = rank$(\boldsymbol{S})$ = r. Suppose $\boldsymbol{S}$ satisfies the SS condition. Under the generative model* (4.1)*, the VolMin criterion* (4.2) *uniquely identifies both $\boldsymbol{A}$ and $\boldsymbol{S}$ up to a permutation. Specifically, any optimal solution to* (4.2) *takes the form*

$$\boldsymbol{B} = \boldsymbol{A}\boldsymbol{\Pi},\ \boldsymbol{H} = \boldsymbol{\Pi}^{\mathsf{T}}\boldsymbol{S},$$

where $\boldsymbol{\Pi}$ is a permutation matrix.

A proof of this result can be found in [44]. We mention that by Theorem 2, given that $\boldsymbol{S}$ satisfies SS, the only remaining indeterminacy is a permutation of the columns (rows) of $\boldsymbol{A}$ (resp. $\boldsymbol{S}$), which is unavoidable – but also inconsequential in most applications.

4.7.3 Proof of Lemma 1

Proof: Assume without loss of generality that the two nonzero columns are the first and second column. Let us denote

$$\zeta(s_1, s_2, \cdots, s_{r-1}) := \sum_{i=1}^{M} \psi_i \left(\boldsymbol{a}_i^{\mathsf{T}} \boldsymbol{s} \right) = 1, \ \boldsymbol{s} \in \text{int } \Delta_r. \tag{4.22}$$

Note that ζ is a function of $(r-1)$ variables $s_1, \cdots, s_{r-1}$, since $\mathbf{1}^{\mathsf{T}} \boldsymbol{s} = 1$. Equation (4.22) suggests that ζ is a constant function on Δ_r. Taking derivative with respect to (w.r.t.) s_1 and s_2, we get

$$\frac{\partial \zeta}{\partial s_1} = \sum_{i=1}^{M} \boldsymbol{a}_i(1) \psi_i' \left(\boldsymbol{a}_i^{\mathsf{T}} \boldsymbol{s} \right), \tag{4.23}$$

and

$$\frac{\partial^2 \zeta}{\partial s_1 \partial s_2} = \sum_{i=1}^{M} \boldsymbol{a}_i(1) \boldsymbol{a}_i(2) \psi_i'' \left(\boldsymbol{a}_i^{\mathsf{T}} \boldsymbol{s} \right) = 0. \tag{4.24}$$

By the assumption on $\boldsymbol{A}$, we have $\boldsymbol{a}_i(1)\boldsymbol{a}_i(2) > 0, \ \forall i$. The assumption that ψ_i's are all convex (or concave) translates to $\psi_i'' \geq 0$ (or $\psi_i'' \leq 0$), for all $i \in [M]$. From (4.24), we conclude that $\psi_i'' = 0, \ \forall i$, which suggests that all the ψ_i's are affine.

While our result is novel and we conceived it for use in our work, other interesting results concerning functional equations can be found in [79, 40].

4.7.4 Proof of Theorem 1

Given assumptions (A2) and equation (4.7), (a) is a direct consequence of Lemma 2.

For (b), we note that from (a), $k_i(t) = d_i t + b_i$ for some constants d_i and b_i. Let $x = \phi_i(t)$, then $t = \phi^{-1}(x)$. Plugging into $f_i(\phi_i(t)) = d_i t + b_i$, we obtain $f_i(x) = d_i \phi^{-1}(x) + b_i$.

4.7.5 Proof of Proposition 1

To prove Proposition 1, we need Lemma 4 and Lemma 5, which are presented here followed by their respective proof.

Lemma 4. *Suppose $\boldsymbol{A} \in \mathbb{R}^{m \times r}$ is full rank and incoherent, i.e. $\boldsymbol{e}_i \notin$ Range$(\boldsymbol{A}), \forall\ i \in [m]$. Then $\widehat{\boldsymbol{A}} = \begin{bmatrix} \boldsymbol{A} \\ \mathbf{1}^\mathsf{T} \end{bmatrix}$ is incoherent.*

This lemma asserts that if a matrix $\boldsymbol{A}$ is incoherent, then appending a row of all 1's preserves incoherence.

Proof: The incoherence condition means that there is no such $\boldsymbol{y} \in \mathbb{R}^r$, such that $\boldsymbol{A}\boldsymbol{y} = \boldsymbol{e}_i$ for any $i \in [m]$. Suppose there is a $\widehat{\boldsymbol{y}} \in \mathbb{R}^r$, such that $\widehat{\boldsymbol{A}}\widehat{\boldsymbol{y}} = \boldsymbol{e}_j$ for some $j \in [m+1]$. There are two cases

1. $1 \leq j \leq m$: This means we have $\widehat{\boldsymbol{y}}$ such that $\boldsymbol{A}\widehat{\boldsymbol{y}} = \boldsymbol{e}_j$ for some $j \in [m]$ – a contradiction to the assumption that $\boldsymbol{A}$ is incoherent.

2. $j = m + 1$: This means that $\boldsymbol{A}\widehat{\boldsymbol{y}} = \mathbf{0}_m$ for $\widehat{\boldsymbol{y}} \neq \mathbf{0}$– a contradiction to the assumption that $\boldsymbol{A}$ is full rank.

Hence $\widehat{\boldsymbol{A}}$ is incoherent if $\boldsymbol{A}$ is full rank and incoherent.

Lemma 5. *For a tall and full rank matrix $\boldsymbol{A} \in \mathbb{R}^{m \times r}$, where $\boldsymbol{A}$ is incoherent, there exists a $\boldsymbol{d} \in \mathbb{R}^m$, such that*

$$\boldsymbol{A}^\mathsf{T}\boldsymbol{d} = \mathbf{0}, \tag{4.25a}$$

$$\|\boldsymbol{d}\|_0 = m. \tag{4.25b}$$

Proof: Let $\boldsymbol{U} \in \mathbb{R}^{m\times(m-r)}$ be a set of bases of the null space of $\boldsymbol{A}$, i.e.

$$\text{Range}(\boldsymbol{U}) = \text{Null}(\boldsymbol{A}). \tag{4.26}$$

By assumption, $\boldsymbol{A}$ is incoherent, hence $\boldsymbol{e}_j \notin \text{Range}(\boldsymbol{A}),\ \forall j \in [m]$. For any j, we have the decomposition

$$\boldsymbol{e}_j = \widehat{\boldsymbol{e}}_j + \overline{\boldsymbol{e}}_j, \tag{4.27}$$

where $\widehat{\boldsymbol{e}}_j \in \text{Range}(\boldsymbol{A})$ and $\overline{\boldsymbol{e}}_j \in \text{Range}(\boldsymbol{U})$. Since $\boldsymbol{e}_j \notin \text{Range}(\boldsymbol{A})$, we have $\boldsymbol{e}_j^\mathsf{T}\boldsymbol{U} = \overline{\boldsymbol{e}}_j^\mathsf{T}\boldsymbol{U} \neq \boldsymbol{0}_{m-r},\ \forall j \in [m]$, which means $\boldsymbol{U}$ does not have a row that is all-zero.

Let $\mathcal{I}_1, \cdots, \mathcal{I}_{m-r}$ be the index sets of nonzero entries in each column of $\boldsymbol{U}$, then we have $\cup_{j=1}^{m-r}\mathcal{I}_j = [m]$ since $\boldsymbol{U}$ does not have an all-zero row. Let us present the following useful fact.

Fact 1. *Let $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^m$, with sets $\mathcal{I}_{\boldsymbol{x}}$ and $\mathcal{I}_{\boldsymbol{y}}$ being the sets of indices of nonzero entries, then we can find a vector $\boldsymbol{z} \in$ Span$\{\boldsymbol{x}, \boldsymbol{y}\}$, such that $\mathcal{I}_{\boldsymbol{z}} = \mathcal{I}_{\boldsymbol{x}} \cup \mathcal{I}_{\boldsymbol{y}}$.*

Proof: Let $a = \frac{1}{\max_j |\boldsymbol{x}_j|}$ and $b = \frac{2}{\min_{j:\boldsymbol{y}_j\neq 0} |\boldsymbol{y}_j|}$. The denominator of b is the minimum of absolute value of the nonzero entries of $\boldsymbol{y}$. Consider the vector

$$\boldsymbol{z} = a\boldsymbol{x} + b\boldsymbol{y}. \tag{4.28}$$

By the choice of a and b, we have $\max_j |a\boldsymbol{x}_j| = 1$ and $\min_{j:\boldsymbol{y}_j\neq 0} |b\boldsymbol{y}_j| = 2$. Hence for any j where $\boldsymbol{x}_j \neq 0$ and $\boldsymbol{y}_j \neq 0$, we have $a\boldsymbol{x}_j + b\boldsymbol{y}_j \neq 0$. This shows that there exists a $\boldsymbol{z} \in \text{Span}\{\boldsymbol{x}, \boldsymbol{y}\}$, such that $\mathcal{I}_{\boldsymbol{z}} = \mathcal{I}_{\boldsymbol{x}} \cup \mathcal{I}_{\boldsymbol{y}}$.

We can now use Fact 1 to show that there exists a fully dense $\boldsymbol{d} \in \text{Range}(\boldsymbol{U})$. Consider the first two columns of $\boldsymbol{U}$: $\boldsymbol{U}_1$ and $\boldsymbol{U}_2$. From Fact 1, we can find a vector $\boldsymbol{u} \in \text{Span}\{\boldsymbol{U}_1, \boldsymbol{U}_2\}$,

such that $\mathcal{I}_{\boldsymbol{u}} = \mathcal{I}_1 \cup \mathcal{I}_2$. Now consider $\boldsymbol{u}$ and $\boldsymbol{U}_3$, invoking Fact 1 again, we can find a vector $\overline{\boldsymbol{u}} \in \text{Span}\{\boldsymbol{u}, \boldsymbol{U}_3\}$, such that $\mathcal{I}_{\overline{\boldsymbol{u}}} = \mathcal{I}_{\boldsymbol{u}} \cup \mathcal{I}_3 = \mathcal{I}_1 \cup \mathcal{I}_2 \cup \mathcal{I}_3$. Continuing this process, we can find a vector $\boldsymbol{d} \in \text{Span}\{\boldsymbol{U}_1, \cdots, \boldsymbol{U}_{m-r}\} = \text{Range}(\boldsymbol{U})$, such that $\mathcal{I}_{\boldsymbol{d}} = \cup_{j=1}^{m-r} \mathcal{I}_j = [m]$; meaning that $\boldsymbol{d} \in \text{Range}(\boldsymbol{U})$ and is fully dense. Since $\boldsymbol{d} \in \text{Range}(\boldsymbol{U})$, we have $\boldsymbol{A}^\mathsf{T}\boldsymbol{d} = \boldsymbol{0}$.

Proof of Proposition 1: Consider a matrix $\boldsymbol{A} \in \mathbb{R}^{m \times r}$ that is tall, full rank, and incoherent, we can rewrite (4.12a) as

$$\begin{bmatrix} \boldsymbol{A}^\mathsf{T} & \boldsymbol{1} \end{bmatrix} \begin{bmatrix} \boldsymbol{d} \\ -1 \end{bmatrix} = \boldsymbol{0} \tag{4.29}$$

Let us denote $\widehat{\boldsymbol{A}}^\mathsf{T} = \begin{bmatrix} \boldsymbol{A}^\mathsf{T} & \boldsymbol{1} \end{bmatrix}$. Then we can see that 1) $\widehat{\boldsymbol{A}} \in \mathbb{R}^{(m+1) \times r}$ is tall and full rank, 2) $\widehat{\boldsymbol{A}}$ is incoherent by Lemma 4. We see that $\widehat{\boldsymbol{A}}$ satisfies all the conditions in Lemma 5, hence there exists a $\boldsymbol{d} \in \mathbb{R}^{m+1}$ such that $\widehat{\boldsymbol{A}}^\mathsf{T}\boldsymbol{d} = \boldsymbol{0}$, and $\|\boldsymbol{d}\|_0 = m + 1$. Since $\boldsymbol{d}$ is fully dense, we construct a $\widehat{\boldsymbol{d}} \in \mathbb{R}^{m+1}$ as

$$\widehat{\boldsymbol{d}} := -\boldsymbol{d}/\boldsymbol{d}(m+1). \tag{4.30}$$

By this construction, we have $\widehat{\boldsymbol{d}}(m+1) = -1$. In addition, $\widehat{\boldsymbol{A}}^\mathsf{T}\widehat{\boldsymbol{d}} = \boldsymbol{0}$ as it is merely a scaled version of $\boldsymbol{d}$. Let $\overline{\boldsymbol{d}} = \widehat{\boldsymbol{d}}(1:m) \in \mathbb{R}^m$, then we have

$$\boldsymbol{A}^\mathsf{T}\overline{\boldsymbol{d}} = \boldsymbol{1}, \quad \|\overline{\boldsymbol{d}}\|_0 = m. \tag{4.31}$$

Hence we managed to show the existence of a $\boldsymbol{d}$ that satisfies both (4.12a) and (4.12b) for any $\boldsymbol{A}$ that satisfies the conditions in Proposition 1. ■

Chapter 5

Summary and Future Directions

This thesis studied the problem of learning latent structure in an unsupervised fashion. The work was motivated by the increasing need to process and cluster high-dimensional data, as well as the need for unsupervised learning methods in many scenarios where data labeling is expensive or impossible.

In Chapter 2, we proposed a joint factorization and clustering (JFC) framework. The JFC framework leverages identifiable matrix and tensor factorization models, to figure out the ground truth representation for each datum. Meanwhile, the underlying clustering structure is exploited to solve the factorization problem, which acts as the dimensionality reduction (DR) step. Our method builds upon the key observation that latent clustering structure can be distorted in the data generation process. This distortion cannot be rectified by PCA-based methods, as PCA finds a set of orthogonal bases, which is unlikely to be compatible with the linear transformation in data generation. In contrast, our method finds the ground truth representations, when the factorization models are identifiable. As such, the distorted latent clustering structure can be restored. We designed efficient algorithms for this method based on the alternating minimization framework, and showcased the effectiveness of the algorithms with synthetic data and real world data sets.

In Chapter 3, we developed a deep clustering network (DCN). This DCN method leverages recent advances in training deep neural networks (DNN), where impressive performance gain has been observed when solving supervised learning tasks, such as image classification [76, 54] and speech recognition [59]. These advancements motivated us to learn a complex nonlinear mapping from data space to latent space, such that the resultant latent space is naturally suitable for clustering. Our key observation is that, as an unsupervised learning method, clustering is fundamentally different than classification. Specifically, when training a DNN for classification, data labels act as guidance for representation learning – the DNN is trained to produce discriminative representations, so that the final classification layer produces good result. When training a DNN for clustering, however, lack of this explicit supervision can lead to trivial solutions. We identified this issue and proposed to include a data reconstruction loss term in the DNN loss function, which fixed the trivial solution problem that could emerge in existing works [124, 132]. In addition, we designed an algorithm building upon alternating minimization and online K-means algorithm [109]. The algorithm involves only minibatch-based updates, enabling scaling to large data sets.

In Chapter 4, we studied a nonlinear mixture model (NMM). Our work was motivated by the fact that nonlinear effects are ubiquitous in real world applications, thus modeling these effects can lead to performance enhancements. From a methodological perspective, albeit there have been many nonlinear representation learning methods, that show learning nonlinear representations of data is beneficial for many machine learning tasks, it is often unclear *when* will these methods work. That is, if indeed the data are generated from some nonlinear transformation on latent parameters, it is often unclear when can we recover these parameters. We studied this problem and established that under some mild conditions on the signal model, the proposed NMM is identifiable up to some trivial indeterminacy. Our key observation is that, if latent parameters exhibit a sum-to-one structure, it is possible to transform this unsupervised learning problem into a supervised regression problem. As a result, we leveraged existing algorithms to

solve this regression problem. Numerical experiments show that indeed latent parameters in this highly non-trivial model can be estimated to high accuracy.

Based on the current work, our ongoing work is concerned with the following extensions:

- *Tackling more general nonlinear mixture models*: The current NMM assumes that data are generated via a linear mixing step followed by an element-wise nonlinear transformation. While this is suitable for several important applications, it is desirable to consider more general nonlinear mappings. For example, in [68, 69], the authors study nonlinear independent component analysis (nICA) with more general nonlinear mappings. It is of interest to study parameter identification issues under such models.

- *Identifying connections between NMM and nICA:* A central assumption underpinning nICA identifiability results is the statistical independence of source signals. In our NMM work, however, the critical assumption is the latent LMM – this latent structure allows us to formulate the identification criterion. Identifiability of LMM is established based on geometrical properties such as the sufficiently scattered condition. It would be beneficial to cross-fertilize ideas in both frameworks. Besides nICA, recent studies show (empirically) that some deep unsupervised learning methods, e.g., variational autoencoders (VAE) [75], produce meaningful and *disentangled* representations. There, disentanglement is tantamount to statistical independence. Despite the many attempts [57, 30, 35], theoretical understanding of this phenomenon is still limited – it is unclear why meaningful and disentangled latent dimensions naturally emerge during training of these models. Building upon our success on establishing identifiability of the aforementioned NMM, we plan to apply and extend our theoretical understandings to the setting of VAEs, and possibly shield light on the intriguing experiment observations.

References

[1] S. Achard and C. Jutten, "Identifiability of post-nonlinear mixtures," *IEEE Signal Processing Letters*, vol. 12, no. 5, pp. 423–426, 2005.

[2] R. Agrawal, B. Golshan, and E. Papalexakis, "A study of distinctiveness in web results of two search engines," in *Proceedings of the 24th International Conference on World Wide Web Companion*. International World Wide Web Conferences Steering Committee, 2015, pp. 267–273.

[3] G. Andrew, R. Arora, J. Bilmes, and K. Livescu, "Deep canonical correlation analysis," in *Proceedings of the 30th International Conference on International Conference on Machine Learning*, 2013, pp. 1247–1255.

[4] S. Arora, R. Ge, Y. Halpern, D. Mimno, A. Moitra, D. Sontag, Y. Wu, and M. Zhu, "A practical algorithm for topic modeling with provable guarantees," in *Proceedings of the 30th International Conference on Machine Learning*, 2013, pp. 280–288.

[5] S. Arora, R. Ge, F. Koehler, T. Ma, and A. Moitra, "Provable algorithms for inference in topic models," in *Proceedings of the 33rd International Conference on Machine Learning*, 2016, pp. 2859–2867.

[6] S. Arora, R. Ge, and A. Moitra, "Learning topic models–going beyond SVD," in *Proceedings of the 53rd Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 2012, pp. 1–10.

[7] B. W. Bader, R. A. Harshman, and T. G. Kolda, "Temporal analysis of social networks using three-way DEDICOM," *Sandia National Laboratories TR SAND2006-2161*, vol. 119, 2006.

[8] A. Banerjee, S. Merugu, I. Dhillon, and J. Ghosh, "Clustering with Bregman divergences," *Journal of Machine Learning Research*, vol. 6, pp. 1705–1749, Oct 2005.

[9] B. Barak, J. A. Kelner, and D. Steurer, "Dictionary learning and tensor decomposition via the sum-of-squares method," in *Proceedings of the forty-seventh annual ACM Symposium on Theory of Computing*. ACM, 2015, pp. 143–151.

[10] A. R. Barron, "Universal approximation bounds for superpositions of a sigmoidal function," *IEEE Transactions on Information Theory*, vol. 39, no. 3, pp. 930–945, 1993.

[11] J. Behler and M. Parrinello, "Generalized neural-network representation of high-dimensional potential-energy surfaces," *Physical Review Letters*, vol. 98, no. 14, 2007.

[12] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," in *Advances in Neural Information Processing Systems 19 (NIPS 2006)*, vol. 19, 2007, pp. 153–160.

[13] M. Berman, H. Kiiveri, R. Lagerstrom, A. Ernst, R. Dunne, and J. F. Huntington, "ICE: A statistical approach to identifying endmembers in hyperspectral images," *IEEE transactions on Geoscience and Remote Sensing*, vol. 42, no. 10, pp. 2085–2095, 2004.

[14] D. P. Bertsekas, "Nonlinear programming," 1999.

[15] J. M. Bioucas-Dias, "A variable splitting augmented lagrangian approach to linear spectral unmixing," in *Proceedings of the First workshop on hyperspectral image and signal processing: Evolution in remote sensing*. IEEE, 2009, pp. 1–4.

[16] 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," *Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 5, no. 2, pp. 354–379, 2012.

[17] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirichlet allocation," *Journal of Machine Learning Research*, vol. 3, no. Jan, pp. 993–1022, 2003.

[18] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[19] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[20] R. Bro and N. Sidiropoulos, "Least squares algorithms under unimodality and i non-negativity constraints," *Journal of Chemometrics*, vol. 12, pp. 223–247, 1998.

[21] J. Bruna and S. Mallat, "Invariant scattering convolution networks," *IEEE Transaction on Pattern Analysis Machine Intelligence*, vol. 35, no. 8, pp. 1872–1886, 2013.

[22] D. Cai, X. He, and J. Han, "Locally consistent concept factorization for document clustering," *IEEE Transaction on Knowledge and Data Engineering*, vol. 23, no. 6, pp. 902–913, 2011.

[23] D. Cai, X. He, J. Han, and T. S. Huang, "Graph regularized nonnegative matrix factorization for data representation," *IEEE Transaction on Pattern Analysis Machine Intelligence*, vol. 33, no. 8, pp. 1548–1560, 2011.

[24] E. J. Candès and B. Recht, "Exact matrix completion via convex optimization," *Foundations of Computational Mathematics*, vol. 9, no. 6, p. 717, 2009.

[25] T.-H. Chan, C.-Y. Chi, Y.-M. Huang, and W.-K. Ma, "A convex analysis-based minimum-volume enclosing simplex algorithm for hyperspectral unmixing," *IEEE Transactions on Signal Processing*, vol. 57, no. 11, pp. 4418–4432, 2009.

[26] T.-H. Chan, W.-K. Ma, A. Ambikapathi, and C.-Y. Chi, "A simplex volume maximization framework for hyperspectral endmember extraction," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 49, no. 11, pp. 4177–4193, 2011.

[27] T.-H. Chan, W.-K. Ma, C.-Y. Chi, and Y. Wang, "A convex analysis framework for blind separation of non-negative sources," *IEEE Transactions on Signal Processing*, vol. 56, no. 10, pp. 5120–5134, 2008.

[28] B. Chen, S. He, Z. Li, and S. Zhang, "Maximum block improvement and polynomial optimization," *SIAM Journal on Optimization*, vol. 22, no. 1, pp. 87–107, 2012.

[29] S. Chen and A. Banerjee, "Sparse linear isotonic models," in *International Conference on Artificial Intelligence and Statistics*, 2018.

[30] T. Q. Chen, X. Li, R. B. Grosse, and D. K. Duvenaud, "Isolating sources of disentanglement in variational autoencoders," in *Advances in Neural Information Processing Systems*, 2018, pp. 2610–2620.

[31] T. F. Coleman and Y. Li, "An interior trust region approach for nonlinear minimization subject to bounds," *SIAM Journal on Optimization*, vol. 6, no. 2, pp. 418–445, 1996.

[32] P. Comon, "Independent component analysis, a new concept?" *Signal Processing*, vol. 36, no. 3, pp. 287–314, 1994.

[33] M. D. Craig, "Minimum-volume transforms for remotely sensed data," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 32, no. 3, pp. 542–552, 1994.

[34] S. Cruces, "Bounded component analysis of linear mixtures: A criterion of minimum convex perimeter," *IEEE Transactions on Signal Processing*, vol. 58, no. 4, pp. 2141–2154, 2010.

[35] B. Dai and D. Wipf, "Diagnosing and enhancing VAE models," *arXiv preprint arXiv:1903.05789*, 2019.

[36] I. Davidson, S. Gilpin, O. Carmichael, and P. Walker, "Network discovery via constrained tensor analysis of fMRI data," in *Proceedings of the 19th ACM SIGKDD international conference on Knowledge Discovery and Data Mining*. ACM, 2013, pp. 194–202.

[37] G. De Soete and J. D. Carroll, "K-means clustering in a low-dimensional Euclidean space," in *New Approaches in Classification and Data Analysis*. Springer, 1994, pp. 212–219.

[38] N. Dobigeon, J.-Y. Tourneret, C. Richard, J. C. M. Bermudez, S. McLaughlin, and A. O. Hero, "Nonlinear unmixing of hyperspectral images: Models and algorithms," *IEEE Signal Processing Magazine*, vol. 31, no. 1, pp. 82–94, 2014.

[39] D. Donoho and V. Stodden, "When does non-negative matrix factorization give a correct decomposition into parts?" in *Advances in Neural Information Processing Systems*, 2004, pp. 1141–1148.

[40] C. Efthimiou, *Introduction to Functional Equations*, 2010.

[41] E. Elhamifar and R. Vidal, "Sparse subspace clustering: Algorithm, theory, and applications," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 11, pp. 2765–2781, 2013.

[42] X. Fu, K. Huang, N. D. Sidiropoulos, and W.-K. Ma, "Nonnegative matrix factorization for signal and data analytics: Identifiability, algorithms, and applications," *IEEE Signal Processing Magazine*, vol. 36, pp. 59–80, 2019.

[43] 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 Transaction on Signal Processing*, vol. 64, no. 23, pp. 6254–6268, Dec. 2016.

[44] 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*, vol. 63, no. 9, pp. 2306–2320, 2015.

[45] P. Georgiev, F. Theis, and A. Cichocki, "Sparse component analysis and blind source separation of underdetermined mixtures," *IEEE Transactions on Neural Networks*, vol. 16, no. 4, pp. 992–996, 2005.

[46] G. B. Giannakis, Y. Shen, and G. V. Karanikolas, "Topology identification and learning over graphs: Accounting for nonlinearities and dynamics," *Proceedings of the IEEE*, vol. 106, no. 5, pp. 787–807, 2018.

[47] N. Gillis, "The why and how of nonnegative matrix factorization," *Regularization, Optimization, Kernels, and Support Vector Machines*, vol. 12, p. 257, 2014.

[48] N. Gillis and R. Luce, "Robust near-separable nonnegative matrix factorization using linear optimization," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1249–1280, 2014.

[49] N. Gillis and S. A. Vavasis, "Fast and robust recursive algorithmsfor separable nonnegative matrix factorization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 4, pp. 698–714, 2014.

[50] R. A. Harshman, "Foundations of the PARAFAC procedure: Models and conditions for an "explanatory" multi-modal factor analysis," 1970.

[51] J. A. Hartigan and M. A. Wong, "Algorithm as 136: A K-means clustering algorithm," *Applied Statistics*, pp. 100–108, 1979.

[52] T. Hastie and P. Y. Simard, "Metrics and models for handwritten character recognition," *Statistical Science*, pp. 54–65, 1998.

[53] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning*. Springer Series in Statistics, 2009, vol. 1.

[54] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[55] J. R. Hershey, Z. Chen, J. Le Roux, and S. Watanabe, "Deep clustering: Discriminative embeddings for segmentation and separation," in *Proceedings of 2016 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2016, pp. 31–35.

[56] R. Heylen, M. Parente, and P. Gader, "A review of nonlinear hyperspectral unmixing methods," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 7, no. 6, pp. 1844–1868, 2014.

[57] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner, "Beta-VAE: Learning basic visual concepts with a constrained variational framework," in *International Conference on Learning Representations*, vol. 3, 2017.

[58] G. E. Hinton and R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[59] G. Hinton, L. Deng, D. Yu, G. Dahl, A.-R. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition," *IEEE Signal Processing Magazine*, vol. 29, 2012.

[60] T. Hofmann, "Probabilistic latent semantic indexing," in *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 1999, pp. 50–57.

[61] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989.

[62] A. Huang, "Similarity measures for text document clustering," in *Proceddings of the sixth New Zealand Computer Science Research Student Conference (NZCSRSC2008)*, 2008, pp. 49–56.

[63] K. Huang, X. Fu, and N. D. Sidiropoulos, "Learning hidden Markov models from pairwise co-occurrences with applications to topic modeling," in *Proceedings of the 35th International Conference on Machine Learning*, 2018.

[64] ——, "Anchor-free correlated topic modeling: Identifiability and algorithm," in *Advances in Neural Information Processing Systems*, 2016, pp. 1786–1794.

[65] K. Huang and N. Sidiropoulos, "Putting nonnegative matrix factorization to the test: a tutorial derivation of pertinent Cramer-Rao bounds and performance benchmarking," *IEEE Signal Processing Magazine*, vol. 31, no. 3, pp. 76–86, 2014.

[66] K. Huang, N. D. Sidiropoulos, and A. P. Liavas, "A flexible and efficient algorithmic framework for constrained matrix and tensor factorization." *IEEE Transaction on Signal Processing*, vol. 64, no. 19, pp. 5052–5065, 2016.

[67] K. Huang, N. D. Sidiropoulos, and A. Swami, "Non-negative matrix factorization revisited: Uniqueness and algorithm for symmetric decomposition," *IEEE Transaction on Signal Processing*, vol. 62, no. 1, pp. 211–224, 2014.

[68] A. Hyvarinen and H. Morioka, "Unsupervised feature extraction by time-contrastive learning and nonlinear ICA," in *Advances in Neural Information Processing Systems*, 2016, pp. 3765–3773.

[69] ——, "Nonlinear ICA of temporally dependent stationary sources," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, 2017.

[70] A. Hyvärinen and E. Oja, "Independent component analysis: algorithms and applications," *Neural networks*, vol. 13, no. 4-5, pp. 411–430, 2000.

[71] A. Hyvarinen, H. Sasaki, and R. E. Turner, "Nonlinear ICA using auxiliary variables and generalized contrastive learning," *arXiv preprint arXiv:1805.08651*, 2018.

[72] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on Machine Learning*, 2015, pp. 448–456.

[73] I. Jeon, E. E. Papalexakis, U. Kang, and C. Faloutsos, "Haten2: Billion-scale tensor decompositions," in *Proceedings of the 31st International Conference on Data Engineering (ICDE)*. IEEE, 2015, pp. 1047–1058.

[74] J. Kim and H. Park, "Fast nonnegative matrix factorization: An active-set-like method and comparisons," *SIAM Journal on Scientific Computing*, vol. 33, no. 6, pp. 3261–3281, 2011.

[75] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[76] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2012, pp. 1097–1105.

[77] J. B. Kruskal, "Three-way arrays: rank and uniqueness of trilinear decompositions, with application to arithmetic complexity and statistics," *Linear Algebra and its Applications*, vol. 18, no. 2, pp. 95–138, 1977.

[78] D. Kuang, C. Ding, and H. Park, "Symmetric nonnegative matrix factorization for graph clustering," in *Proceedings of the 2012 SIAM International Conference on Data Mining*. SIAM, 2012.

[79] M. Kuczma, *An introduction to the theory of functional equations and inequalities: Cauchy's equation and Jensen's inequality*. Springer Science & Business Media, 2009.

[80] A. Kumar, V. Sindhwani, and P. Kambadur, "Fast conical hull algorithms for near-separable non-negative matrix factorization," in *Proceedings of 30th International Conference on Machine Learning*, 2013, pp. 231–239.

[81] 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.

[82] M. Law, A. Topchy, and A. K. Jain, "Model-based clustering with probabilistic constraints," in *Proceedings of the 2005 SIAM International Conference on Data Mining*. SIAM, 2005, pp. 641–645.

[83] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[84] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, p. 788, 1999.

[85] D. D. Lewis, Y. Yang, T. G. Rose, and F. Li, "RCV1: A new benchmark collection for text categorization research," *Journal of Machine Learning Research*, vol. 5, pp. 361–397, Apr 2004.

[86] X. Liang, A. Connelly, and F. Calamante, "Graph analysis of resting-state ASL perfusion MRI data: nonlinear correlations among CBF and network metrics," *Neuroimage*, vol. 87, pp. 265–275, 2014.

[87] A.-L. Lin, P. T. Fox, J. Hardies, T. Q. Duong, and J.-H. Gao, "Nonlinear coupling between cerebral blood flow, oxygen consumption, and ATP production in human visual cortex," *Proceedings of the National Academy of Sciences*, vol. 107, no. 18, pp. 8446–8451, 2010.

[88] 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*, vol. 53, no. 10, pp. 5530–5546, 2015.

[89] G. Liu, Z. Lin, and Y. Yu, "Robust subspace segmentation by low-rank representation." in *Proceedings of the 27th International Conference on Machine Learning*, 2010, pp. 663–670.

[90] S. Lloyd, "Least squares quantization in PCM," *IEEE Transaction on Information Theory*, vol. 28, no. 2, pp. 129–137, 1982.

[91] 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," *Signal Processing Magazine*, vol. 31, no. 1, pp. 67–81, 2014.

[92] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, vol. 1, no. 14. Oakland, CA, USA., 1967, pp. 281–297.

[93] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to information retrieval*. Cambridge university press Cambridge, 2008, vol. 1.

[94] X. Mao, P. Sarkar, and D. Chakrabarti, "On mixed memberships and symmetric nonnegative matrix factorizations," in *Proceedings of the 34th International Conference on Machine Learning*, 2017, pp. 2324–2333.

[95] K. Maruhashi, F. Guo, and C. Faloutsos, "Multiaspectforensics: Pattern mining on large-scale heterogeneous networks with tensor analysis," in *2011 International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*. IEEE, 2011, pp. 203–210.

[96] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proceedings of the 27th International Conference on Machine Learning*, 2010, pp. 807–814.

[97] A. Y. Ng, "Sparse autoencoder," *CS294A Lecture notes*, vol. 72, pp. 1–19, 2011.

[98] A. Y. Ng, M. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Advances in Neural Information Processing Systems 15 (NIPS 2002)*, 2002, pp. 849–856.

[99] S. Nitish, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[100] E. E. Papalexakis, K. Pelechrinis, and C. Faloutsos, "Location based social network analysis using tensors and signal processing tools," in *Proc. IEEE 6th Int. Workshop Comput. Adv. in Multi-Sensor Adaptive Process.* IEEE, 2015, pp. 93–96.

[101] E. E. Papalexakis, N. D. Sidiropoulos, and R. Bro, "From K-means to higher-way co-clustering: Multilinear decomposition with sparse latent factors," *IEEE Transaction on Signal Processing*, vol. 61, no. 2, pp. 493–506, 2013.

[102] V. M. Patel, H. Van Nguyen, and R. Vidal, "Latent space sparse subspace clustering," in *Proceedings of the 2013 IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2013, pp. 225–232.

[103] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Neurocomputing: Foundations of Research*, pp. 696–699, 1988.

[104] P. Santago and H. D. Gage, "Statistical models of partial volume effect," *IEEE Transactions on Image Processing*, vol. 4, no. 11, pp. 1531–1540, 1995.

[105] L. K. Saul and S. T. Roweis, "Think globally, fit locally: unsupervised learning of low dimensional manifolds," *Journal of Machine Learning Research*, vol. 4, pp. 119–155, 2003.

[106] B. Scholkopf and A. J. Smola, *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2001.

[107] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 815–823.

[108] P. Schwaller, T. Gaudin, D. Lanyi, C. Bekas, and T. Laino, ""Found in translation": predicting outcomes of complex organic chemistry reactions using neural sequence-to-sequence models," *Chemical Science*, vol. 9, no. 28, pp. 6091–6098, 2018.

[109] D. Sculley, "Web-scale K-means clustering," in *Proceedings of the 19th International Conference on World Wide Web (WWW)*. ACM, 2010, pp. 1177–1178.

[110] N. D. Sidiropoulos and R. Bro, "On the uniqueness of multilinear decomposition of N-way arrays," *Journal of Chemometrics*, vol. 14, no. 3, pp. 229–239, 2000.

[111] A. Strehl, J. Ghosh, and R. Mooney, "Impact of similarity measures on web-page clustering," in *Workshop on Artificial Intelligence for Web Search (AAAI 2000)*, 2000, pp. 58–64.

[112] A. Taleb and C. Jutten, "Source separation in post-nonlinear mixtures," *IEEE Transactions on Signal Processing*, vol. 47, no. 10, pp. 2807–2820, 1999.

[113] Theano Development Team, "Theano: A Python framework for fast computation of mathematical expressions," *arXiv e-prints*, vol. abs/1605.02688, May 2016. [Online]. Available: http://arxiv.org/abs/1605.02688

[114] P. Tseng, "Nearest q-flat to m points," *Journal of Optimization Theory and Applications*, vol. 105, no. 1, pp. 249–252, 2000.

[115] M. S. Vafaee and A. Gjedde, "Model of blood–brain transfer of oxygen explains nonlinear flow-metabolism coupling during stimulation of visual cortex," *Journal of Cerebral Blood Flow & Metabolism*, vol. 20, no. 4, pp. 747–754, 2000.

[116] L. Van der Maaten and G. E. Hinton, "Visualizing data using t-SNE," *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, Nov 2008.

[117] M. Vichi and H. A. Kiers, "Factorial K-means analysis for two-way data," *Computational Statistics & Data Analysis*, vol. 37, no. 1, pp. 49–64, 2001.

[118] R. Vidal, "A tutorial on subspace clustering," *IEEE Signal Processing Magazine*, vol. 28, no. 2, pp. 52–68, 2010.

[119] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P. A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *Journal of Machine Learning Research*, vol. 11, pp. 3371–3408, Dec 2010.

[120] U. Von Luxburg, "A tutorial on spectral clustering," *Statistics and Computing*, vol. 17, no. 4, pp. 395–416, 2007.

[121] F.-Y. Wang, C.-Y. Chi, T.-H. Chan, and Y. Wang, "Blind separation of positive dependent sources by non-negative least-correlated component analysis," in *Proceedings of the 16th IEEE Signal Processing Society Workshop on Machine Learning for Signal Processing*. IEEE, 2006, pp. 73–78.

[122] L. Wang, "Discovering phase transitions with unsupervised learning," *Physical Review B*, vol. 94, no. 19, 2016.

[123] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma, "Robust face recognition via sparse representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 2, pp. 210–227, 2009.

[124] J. Xie, R. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," in *Proceedings of the 33rd International Conference on Machine Learning*, 2016.

[125] W. Xu and Y. Gong, "Document clustering by concept factorization," in *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2004, pp. 202–209.

[126] W. Xu, X. Liu, and Y. Gong, "Document clustering based on non-negative matrix factorization," in *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*. ACM, 2003, pp. 267–273.

[127] 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*, vol. 6, no. 3, pp. 1758–1789, 2013.

[128] B. Yang, X. Fu, and N. D. Sidiropoulos, "Learning from hidden traits: Joint factor analysis and latent clustering," *IEEE Transaction on Signal Processing*, pp. 256–269, Jan. 2017.

[129] ——, "Joint factor analysis and latent clustering," in *Proc. IEEE 6th Int. Workshop Comput. Adv. in Multi-Sensor Adaptive Process.* IEEE, 2015, pp. 173–176.

[130] B. Yang, X. Fu, N. D. Sidiropoulos, and M. Hong, "Towards K-means-friendly spaces: Simultaneous deep learning and clustering," in *Proceedings of the 34th International Conference on Machine Learning*, 2017.

[131] B. Yang, X. Fu, N. D. Sidiropoulos, and K. Huang, "Learning nonlinear mixtures: Identifiability and algorithm," *arXiv preprint arXiv:1901.01568*, 2019.

[132] J. Yang, D. Parikh, and D. Batra, "Joint unsupervised learning of deep representations and image clusters," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 5147–5156.

[133] K. Y. Yeung and W. L. Ruzzo, "Details of the adjusted Rand index and clustering algorithms, supplement to the paper "An mpirical study on principal component analysis for clustering gene expression data"," *Bioinformatics*, vol. 17, no. 9, pp. 763–774, 2001.

[134] X. Yi, Z. Wang, C. Caramanis, and H. Liu, "Optimal linear estimation under unknown nonlinear transform," in *Advances in Neural Information Processing Systems*, 2015, pp. 1549–1557.

[135] C. You, D. Robinson, and R. Vidal, "Scalable sparse subspace clustering by orthogonal matching pursuit," in *Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, 2016.

[136] T. Zhang, B. Fang, W. Liu, Y. Y. Tang, G. He, and J. Wen, "Total variation norm-based nonnegative matrix factorization for identifying discriminant representation of image patterns," *Neurocomputing*, vol. 71, no. 10, pp. 1824–1831, 2008.