



HOKKAIDO UNIVERSITY

Title	A Study on Efficient Algorithms for Nonnegative Matrix/Tensor Factorization
Author(s)	木村, 圭吾
Degree Grantor	北海道大学
Degree Name	博士(情報科学)
Dissertation Number	甲第12623号
Issue Date	2017-03-23
DOI	https://doi.org/10.14943/doctoral.k12623
Doc URL	https://hdl.handle.net/2115/65379
Type	doctoral thesis
File Information	Keigo_Kimura.pdf



Doctoral Dissertation

A Study on Efficient Algorithms for
Nonnegative Matrix/Tensor Factorization.

Graduate School of Information Science and Technology,
Hokkaido University Japan,
Keigo Kimura

Abstract

Recently, machine learning and data mining technologies play important roles in many areas with data. For these areas, a dataset is often represented by a matrix whose each column or row vector represents a data point. Matrix Factorization that decomposes a matrix into a product of low-rank matrices, is a fundamental technique to analyze a matrix. It reveals the hidden structures behind the data and helps to accelerate the learning procedures in precisions and time.

Nonnegative Matrix Factorization (NMF) is one of matrix factorization methods whose elements are imposed to be nonnegative. In general, an intuitive data representations are only of nonnegative values, such as word counts on a document, brightness values of an image or connection strengths between users on a social network. This is because we, human being, rarely use negative values to represent natural phenomena. On NMF, the decomposed matrices are also nonnegativity, and thus, it is easy to give physical explanations and interpretations to the results. Due to this nature, NMF is most useful in exploratory data mining.

The problem of NMF is its complexity. It is known that the problem of finding a global optimal solution of NMF is NP-hard. Therefore, many heuristics to obtain local optimal solutions in iterative manners have been proposed so far. To factorize a large-scale matrix, this iterations may cause computational problem and sometime algorithms may become intractable due to its large computational time. In other words, the speed of convergence is one of main concerns to develop NMF algorithms.

In a similar stream, tensors, a multi-dimensional array, have been gathering more and more attentions as a generalization of matrices. Contrary to a matrix that can represent only binomial relationships, a tensor can represent multinomial relationship. For example, users tweets with timestamps on a social network service can be represented by a tensor of 3-dimensional of user, word and time axis. This data cannot be represented by a matrix. The necessity of such a high dimensional array has been increasing nowadays due to remarkable evolutions of data collection technologies and storages.

Nonnegative Tensor Factorization (NTF), decomposes a nonnegative tensor into a product of several nonnegative matrices, is one of generalizations of NMF to tensors. Similar to the matrix case, As the same in data represented by a matrix, nonnegativity is often required even in tensor factorization methods because nonnegativity plays crucial roles to understand data structures.

The optimization problem of NTF is much more difficult than that of NMF. Fortunately, NTF can be solved by an NMF algorithm by transforming the NTF problem to a set of NMF problems. However, this also means that the computational problem on NMF arises even on NTF, that is, a large number of optimizations procedures is repeated in iterations, what is worse, there is more serious computational problem called *Intermediate Data Explosion* (IDE) problem on NTF. The IDE problem refers a large matrix computation appears in the transformation of a NTF problem into a set of NMF problems. In more detail, a product of a transformed tensor and Khatri-rao product of matrices needs large matrices. This needs a large memory space and time at the same time. This IDE problem appears in every optimization step, and thus, no improvements can be expected without solve or alleviate this problem.

In this dissertation, we have develop efficient algorithms for both NMF and NTF with addressing above concerns. For NMF, we focused on NMF with orthogonal constraint called Orthogonal NMF. Orthogonal NMF has a connection to clustering algorithms and thus often used clustering or related tasks. I propose an efficient algorithm for Orthogonal NMF. I explicitly utilized nonnegativity to simplify the orthogonal constraint and developed column-wise block coordinate descent so as to bring faster convergence. On some experiments, we observed that the proposed algorithm performed five times faster than traditional algorithms on the average. In addition, we extended the proposed algorithm to be able to deal with Bregman divergence including the traditional Frobenius norm or KL-divergence. I also proved that the proposed method, Orthogonal NMF with Bregman divergence is equivalent Bregman Hard Clustering which contains K-means clustering as a special case.

For NTF, we proposed two efficient algorithms for standard NTF problems. As we stated before, the IDE problem is my main concern on NTF. To solve this IDE problem, we utilized the sparsity of resultant matrices of NTF. It is known that the resultant matrices of NTF are often sparse even when the original tensor is dense in general. I succeeded in solving smaller problems by ignoring parts that are unnecessary to be updated. To end this, we predict elements which will be zero after updating by using upper-bounds of the updated to zero values. This succeeded to solve the IDE problem.

Furthermore, we proposed a lazy coordinate descent on the basis of sampling methods. The method guarantees some amount of efficiency in the updating procedure. I used importance sampling for this goal. The proposed two algorithms are advantageous to traditional distributed method or sampling method in the sense of that convergence and better approximation accuracy. On some experiments, the proposed algorithms were about four times faster than traditional naive methods with one to five hundreds times less memory consumption.

Acknowledgment

First of all, I really appreciate Japan Society for the Promotion of Science giving me a research fellowship for young scientists (DC1) for the last three years. This financial supports helped me a lot and let me survive at this northern cold city, Sapporo. This study has been supported by JSPS KAKENHI Grant Number 25280079.

I would like to express my gratitude to Prof. Mineichi Kudo, my supervisor, the collaborator of PRML laboratory. He gave me many suggestive advices not only about researches but about life or way to live comfortably. These advices changed my mind and thought for how to live. He let me research as freely as possible. This three years were really comfortable to study and research about my interests. This research environment could not be realized without him. I also would like to express my thankfulness to associate Prof. Atsuyoshi Nakamura. He has helped me a lot from when I was a undergraduate student. Especially, in Vietnam, the discussion with him about research was really insightful for me. I also want to express my appreciation to all professors at the computer science division. In addition, I would like to thank Prof Akira Tanaka and Prof. Hideyuki Imai, semi-supervisors of mine. I have suffered from their invaluable but incisive comments to my researches.

I am also grateful to Lu Sun, a colleague at PRML laboratory. I have researched about Multi-Label Classification inspired by him for the last one year and half of this doctoral course. The discussion and researching with him and Prof. Mineichi Kudo were fun, invaluable and precious experiences in my life. We have published seven papers about Multi-Label Classification within one year and half. I cannot help wondering if we could work together much earlier. I also want to thank members at PRML laboratory and at MEME laboratory when I belong to. The student life with them were really exciting and I learned a lot of things from them.

I am also greatly thankful to Prof. Yuzuru Tanaka and Prof. Tetsuya Yoshida. I also would like to express my apologies to them for my ignorance and stupidity.

At last, I would like to express my thankfulness for my family to support

me for nine years of student life at Hokkaido university.

Contents

1	Introduction	17
1.1	Background	17
1.2	Contributions	18
1.3	Contents and Outline	19
2	Preliminaries	21
2.1	Basic Notations for Matrix and Tensor	21
2.2	Basic Operations in Matrix and Tensor Factorization	21
3	Related Work	25
3.1	Nonnegative Matrix Factorization	25
3.1.1	Optimization	27
3.1.2	Extensions and Variants of NMF	34
3.2	Tensor Factorization	35
3.2.1	Tucker Decomposition	37
3.2.2	CANDECOMP/PARAFAC Decomposition	38
3.2.3	Other Tensor Factorizations	41
3.2.4	Nonnegativity on Tensor	42
4	Column-Wise Update Algorithms for Orthogonal Nonnegative Matrix Factorization	47
4.1	Introduction	47
4.2	Related Work	48
4.2.1	Convergence Speed of Nonnegative Matrix Factorization	48
4.2.2	Orthogonal NMF	50
4.3	Hierarchical Alternating Least Squares Algorithm for ONMF .	52
4.3.1	Column-wise Orthogonal Constraint	53
4.3.2	HALS-ONMF	54
4.4	ONMF with Bregman divergence	55
4.4.1	Scalar Block Coordinate Descent Algorithm (sBCD) . .	55
4.4.2	Bregman Divergence ONMF	59

CONTENTS

4.4.3	Relation to Bregman Hard Clustering	60
4.5	Performance Evaluation	63
4.5.1	Datasets	63
4.5.2	Compared Algorithms and Evaluation Measures	63
4.5.3	Comparison on ONMF problems	64
4.5.4	Comparison on Bregman Divergence ONMF Problems	68
4.5.5	Clustering Experiments	71
4.6	Concluding Remarks	74
5	Variable Selection for Efficient Sparse Nonnegative Tensor Factorization	75
5.1	Introduction	75
5.1.1	Nonnegative Tensor Factorization	76
5.2	Related Works	77
5.3	NTF with A Variable Selection	79
5.3.1	Variable Selection	79
5.3.2	Sampling Variables with Weighted Distribution	82
5.3.3	Sparsity Constraint	84
5.3.4	Analysis of Algorithm	84
5.4	Experiments	86
5.4.1	Evaluation Measure	87
5.4.2	Methods and Parameters	87
5.4.3	Results	88
5.5	Conclusion and Discussion	90
6	Conclusion	93
6.1	Contributions	93
6.2	Discussions and Future Work	94

List of Figures

3.1	Illustrations of nonnegative matrix factorization and tensor factorizations	36
3.2	Illustrations of various tensor factorization methods	43
4.1	Comparison of five ONMF algorithms in the degree of approximation on six-real life datasets. The proposed HALS-ONMF algorithm converges faster than the other four conventional algorithms.	66
4.2	Orthogonality attained by three ONMF algorithms. The proposed HALS algorithm converges faster than the other four conventional algorithms, but the final state is worse than some of the others.	67
4.3	Comparison of KL-divergence, IS-divergence and β -divergence NMF on an artificial dataset ($\mathbf{X} \in \mathbb{R}^{2000 \times 1000}$). The first row shows the value of relative residual and the second row shows the degree of orthogonality.	70
5.1	Outline of the proposed NTF algorithm in case of three factors ($J = 3$). For each column, we predict non-zero variables and factorize the corresponding smaller tensor instead the original tensor. Then, we update the selected variables only after replacing the other variables with zero.	80
5.2	Results on an artificial dataset.	89
5.3	Sparsity and CPU time per iteration on four real-life datasets.	91
5.4	Relative space consumption and approximation error on four real-life datasets.	92

LIST OF FIGURES

List of Tables

2.1	A summary of notations and operations used in this thesis. . .	24
4.1	A summary of categorization of ONMF algorithms. Frobenius, KL and Bregman denotes distortions used for measuring the degree of approximation.	53
4.2	Examples of Bregman divergence	57
4.3	Datasets used in the experiments. Here $\#nnz$ is the number of non-zero values.	63
4.4	The results when the stopping criterion (4.23) with $\epsilon = 10^{-4}$ is applied on four datasets. NRV means Normalized Residual Values (4.20).	69
4.5	Datasets used in the clustering experiments.	71
4.6	The clustering results evaluated in NMI (the higher, the better) on four-real life datasets.	73
5.1	Datasets represented by 3-mode tensors	86
5.2	CPUtime with 30 iterations ($J = 30$)	90

LIST OF TABLES

List of Algorithms

1	Multiplicative Update rule for Frobenius NMF	28
2	Hierarchal Alternating Least Squares Algorithm for NMF . . .	30
3	Fast Hierarchal Alternating Least Squares Algorithm for NMF	31
4	Greedy Coordinate Descent Update for Matrix $\mathbf{U}$	32
5	Hi-Order Singular Value Decomposition (HOSVD)	38
6	Hi-Order Orthogonal Iteration (HOOI)	39
7	CANDECOMP/PARAFAC Decomposition (CPD)	41
8	Nonnegative CANDECOMP/PARAFAC Decomposition (NTF)	45
9	Fast HALS-Orthogonal NMF	56
10	sBCD-Orthogonal NMF (a generalized HALS-ONMF)	61
11	NTF with Variable Selections	83
12	Variable Selection for $\mathbf{A}_{:,j}$	83
13	Importance Sampling of Variables	84

LIST OF ALGORITHMS

Chapter 1

Introduction

1.1 Background

Nonnegative Matrix Factorization (NMF), decompose a nonnegative matrix into a product of two nonnegative matrices, has become popular due to its high interpretability and flexibility [LS00]. The nonnegativity is natural on many applications such as image analysis, document analysis and sound analysis [LS99,CZPA09]. The nonnegativity brings physical meaningful representation of the resultant matrices. Therefore, it is suitable for the exploratory data mining. In addition, NMF is superior to traditional matrix factorizations such as Singular Value Decomposition (SVD) or Principal Component Analysis (PCA) in the sense of that the nonnegativity often leads a sparse and part decomposition as a side effect [Hoy02,Hoy04,LS99,LS00,CZPA09].

However, in contrast to SVD or PCA can easily find the optimal factorization, NMF problem is NP-hard and cannot find a optimal solution efficiently [Vav09]. Thus many algorithms to find local minimum by iterative optimizations, that is, updating one matrix with fixing the other matrix and repeating this procedure for each matrix until convergence, have been proposed so far [LS00,Lin07,LLP12,HD11,CAH09]. On large-scale problem, a large number of iterations may become a critical problem and makes algorithms intractable due to these large computational time. Therefore, it is important to reduce the number of iterations to optimize, in other words, NMF algorithms which converge faster is demanded.

A tensor, a multi-dimensional array, has received a surge of interests recently [KB09]. This is because multi-aspect or multi-dimensional data has become available as a result of remarkable evolutions of sensors or computational capacities of machines [SH05,KB09]. Tensor Factorization (TF), decomposes a tensor into a product of a low-dimensional tensor and

several low-dimensional matrices, is a popular technique to analyze tensor data [KB09, AY09]. Since TF can be considered as a generalization of matrix factorization, TF is often used as a dimension reduction or a feature extraction for tensor data [CZPA09, BBB08, ADK09]. In addition, in general, TF produces a matrix for each mode and these matrices are useful for the mode-wise analysis. This connects tensor data to the traditional matrix based machine learning methods [SH05, DLDV00a].

Nonnegative Tensor Factorization (NTF) is TF with nonnegative constraint and it is a generalization of NMF [SH05, CZPA09, KB09]. NTF inherits the merit of NMF, that is, the factorization result has physical meanings and is easy to interpret or understand. NTF is also a popular technique to analyze tensor data, and thus, many applications are considered [BBB08, CAH09, ADK09, SH05, CZS⁺07, H⁺14, PKA10].

The large computational complexity is one of problems in NTF. Since NTF is solved as a set of NMF problems, a large number of iterations until convergence arises as a problem even on NTF. However, there is a more serious problem called *Intermediate Data Explosion* (IDE) problem [KPHF12, PFS12], that is, a calculation of large matrix operations in the gradient calculations. This prohibits to factorize large-scale tensors on personal machines. The IDE problem is addressed by separating the calculation to parallelable calculations [KPHF12, JPKF15] or using sparsity of tensor [CV14]. However, these are based on parallel computing environments and thus an efficient method w.r.t. both space and time on a single machine environment are demanded.

In this dissertation, I focus on these above computational problems for both NMF and NTF and develop efficient algorithms.

1.2 Contributions

The contributions of this dissertation are as follows:

(1) We have proposed a column-wise update algorithm for Orthogonal NMF problem with Frobenius norm [KTK14]. Orthogonal NMF, impose the one matrix to be orthogonal, is one of popular variants of NMF. Traditional methods for this problem uses matrix-wise update since the orthogonal constraint represented matrix-wisely. We decomposed a matrix-wise representation of orthogonal constraint to column-wise constraint by utilizing nonnegativity condition explicitly and derive column-wise update rules. As column-wise update algorithms converge faster than matrix-wise update algorithms in standard NMF problems, the proposed algorithm converges faster than traditional algorithms and shows better approximation accuracy.

On experiments the proposed algorithm performed about four times faster than the other algorithm with the best approximation accuracy in the best case.

(2) We extended the above orthogonal NMF problem with Frobenius to with Bregman divergence [KKT16]. Bregman divergence is a wide class of distortions which contains Frobenius norm Kullback-Leibler divergence, Itakura-Saito-divergence, β -divergence and so on [BGW05, BMDG05].

(3) We also proved that this problem is the equivalent to Bregman Hard Clustering proposed by Banerjee *et al.* [BMDG05], that is, a wide class of clustering methods which contains K-means clustering, Information Theoretic Co-clustering [DMM03] and so on. In addition, I also proposed a column-wise algorithm for this problem. On some experiments on synthetic datasets, the proposed algorithm can factorize the matrix with attaining high orthogonality.

(4) We have proposed two efficient algorithms for NTF [KK15]. We solve IDE problem by reducing the elements to be updated. The one method is variable selection. This method predicts the elements which will be zero after updating and reduce the calculation of these. Since the result of NTF with sparse constraints is sparse in general, and thus, it reduces both time and space complexity. The other one is importance sampling. This method is a kind of lazy coordinate descents and chooses elements to be updated wisely and forcibly. The variable selection cannot guarantee the reduction of memory consumption, on the other hand, this method can. In the best case, the proposed algorithm performed more than five times faster than naive algorithms with one to five hundred less memory consumption while keeping the approximation accuracy.

1.3 Contents and Outline

The rest of this dissertation is organized as follows. At first, I introduce common notations and basic operations for tensors and matrices in Chapter 2. In Chapter 3, I introduce related work of this dissertation and Chapter 4 I propose a column-wise update algorithm for Orthogonal NMF with Frobenius norm and extend this algorithm to Bregman divergence, a wider class of distances. We also derive a efficient column-wise algorithm for Orthogonal NMF with Bregman divergence. We confirms the efficiency of the proposed methods on experiments. In Chapter 5, I propose two efficient algorithms for NTF. We demonstrate that proposed algorithms are faster than the other state-of-the-art algorithms with smaller memory consumption with the same approximation accuracy. At last, conclusions come in Chapter 6.

Chapter 2

Preliminaries

In this chapter, I introduce notations and basics of matrix factorization and tensor factorization.

2.1 Basic Notations for Matrix and Tensor

In this dissertation, $\underline{\mathbf{X}}$, $\mathbf{X}$, $\mathbf{x}$ and x denote a tensor, a matrix, a vector and a scalar, respectively. Following MATLAB representation, $\mathbf{X}_{i:}$ and $\mathbf{X}_{:,j}$ denote i th row and j th column vector, respectively. In addition, I use $\mathbf{x}_j$ or $\mathbf{X}_j$ to denote j th column vector of $\mathbf{X}$.

For simplicity, in this dissertation, I consider 3-order tensors only. However, all algorithms explained in this dissertation can be generalized to N -order tensors easily. For a tensor, I use I_n for $n = 1, 2, 3$ as a dimension of the corresponding mode (*e.g.* $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$). Slices (matrices) of the tensor are denoted by $\underline{\mathbf{X}}_{::i_3} \in \mathbb{R}^{I_1 \times I_2}$, $\underline{\mathbf{X}}_{:,i_2:} \in \mathbb{R}^{I_1 \times I_3}$ and $\underline{\mathbf{X}}_{i_1::} \in \mathbb{R}^{I_2 \times I_3}$ for each mode respectively and also fibers of the tensor are denoted by $\underline{\mathbf{X}}_{:,i_2i_3} \in \mathbb{R}^{I_1}$, $\underline{\mathbf{X}}_{i_1:i_3} \in \mathbb{R}^{I_2}$ and $\underline{\mathbf{X}}_{i_1i_2:} \in \mathbb{R}^{I_3}$, respectively. In other words, I use $:$ as a wild card. An element of the tensor is denoted by $\underline{\mathbf{X}}_{i_1i_2i_3}$.

I use $\mathbf{X}^T$ for the transpose of $\mathbf{X}$ and $\mathbf{X}^{-1}$ as the inverse matrix of $\mathbf{X}$. $\mathbf{X}^\dagger$ denotes Moore-Penrose pseudo inverse matrix of $\mathbf{X}$. I also use $\mathbf{1}$ as a matrix filled with 1 in all elements.

2.2 Basic Operations in Matrix and Tensor Factorization

Here, I introduce some operations for tensors and matrices. Most of all these are based on [KB09].

Definition 1 (Outer Product). *Outer product of N vectors $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(N)}$ is denoted by $\mathbf{x}^{(1)} \circ \mathbf{x}^{(2)} \circ \dots \circ \mathbf{x}^{(N)}$, yields an N -order tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_N}$, by multiplying them element-wisely as $\underline{\mathbf{X}}_{i_1 i_2 \dots i_N} = \mathbf{x}_{i_1}^{(1)} \mathbf{x}_{i_2}^{(2)} \dots \mathbf{x}_{i_N}^{(N)}$*

Definition 2 (Kronecker Product). *Kronecker product of two matrices $\mathbf{A} \in \mathbb{R}^{J \times K}$ and $\mathbf{B} \in \mathbb{R}^{L \times M}$, denoted by $\mathbf{A} \otimes \mathbf{B}$, operates the following calculation:*

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} \mathbf{A}_{11}\mathbf{B} & \mathbf{A}_{12}\mathbf{B} & \cdots & \mathbf{A}_{1K}\mathbf{B} \\ \mathbf{A}_{21}\mathbf{B} & \mathbf{A}_{22}\mathbf{B} & \cdots & \mathbf{A}_{2K}\mathbf{B} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}_{J1}\mathbf{B} & \mathbf{A}_{J2}\mathbf{B} & \cdots & \mathbf{A}_{JK}\mathbf{B} \end{bmatrix} \in \mathbb{R}^{JL \times KM}.$$

Definition 3 (Khatri-Rao Product). *Khatri-Rao product of two matrices $\mathbf{A} \in \mathbb{R}^{J \times L}$ and $\mathbf{B} \in \mathbb{R}^{K \times L}$ with the same number L of columns, denoted by $\mathbf{A} \odot \mathbf{B}$, operates the following calculation:*

$$\mathbf{A} \odot \mathbf{B} = \begin{bmatrix} \mathbf{A}_{11}\mathbf{B}_{:1} & \mathbf{A}_{12}\mathbf{B}_{:2} & \cdots & \mathbf{A}_{1L}\mathbf{B}_{:L} \\ \mathbf{A}_{21}\mathbf{B}_{:1} & \mathbf{A}_{22}\mathbf{B}_{:2} & \cdots & \mathbf{A}_{2L}\mathbf{B}_{:L} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}_{J1}\mathbf{B}_{:1} & \mathbf{A}_{J2}\mathbf{B}_{:2} & \cdots & \mathbf{A}_{JL}\mathbf{B}_{:L} \end{bmatrix} \in \mathbb{R}^{JK \times L}.$$

Khatri-Rao product is a column-wise Kronecker product.

Definition 4 (Hadamard Product). *Hadamard product $\mathbf{A} * \mathbf{B}$ is obtained by element-wise multiplications of two matrices of the same size, that is, $(\mathbf{A} * \mathbf{B})_{ij} = \mathbf{A}_{ij}\mathbf{B}_{ij}$.*

In tensor factorization, the following equation connects Hadamard product with Khatri-Rao product and is often employed:

$$(\mathbf{A} \odot \mathbf{B})^T (\mathbf{A} \odot \mathbf{B}) = \mathbf{A}^T \mathbf{A} * \mathbf{B}^T \mathbf{B}. \quad (2.1)$$

Definition 5 (Matricization). *Mode- n matricization reorders an N -order tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times \dots \times I_N}$ into a matrix $\underline{\mathbf{X}}_{(n)} \in \mathbb{R}^{I_n \times (\prod_{k \neq n} I_k)}$ which has I_n rows and $(\prod_{k \neq n} I_k)$ columns. For example, by matricization $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ induces three matrices of $\underline{\mathbf{X}}_{(1)} \in \mathbb{R}^{I_1 \times I_2 I_3}$, $\underline{\mathbf{X}}_{(2)} \in \mathbb{R}^{I_2 \times I_1 I_3}$ and $\underline{\mathbf{X}}_{(3)} \in \mathbb{R}^{I_3 \times I_1 I_2}$.*

Definition 6 (Mode product). *Mode- n mode product is an operator which products a tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times \dots \times I_N}$ with a matrix $\mathbf{A} \in \mathbb{R}^{J \times I_n}$ to have another tensor $(\underline{\mathbf{X}} \times_n \mathbf{A}) \in \mathbb{R}^{I_1 \times \dots \times I_{n-1} \times J \times I_{n+1} \times \dots \times I_N}$ of which elements are calculated as*

$$(\underline{\mathbf{X}} \times_n \mathbf{A})_{i_1 i_2 \dots i_{n-1} j i_{n+1} \dots i_N} = \sum_{i_n=1}^{I_n} \underline{\mathbf{X}}_{i_1 i_2 \dots i_{n-1} i_n i_{n+1} \dots i_N} \mathbf{A}_{j i_n}.$$

In tensor factorization, the mode product and the matricization has the following relation:

$$\mathbf{A}\underline{\mathbf{X}}_{(n)} = (\underline{\mathbf{X}} \times_n \mathbf{A})_{(n)}.$$

This equation connects tensor factorization to matrix factorization. Table 2.1 shows the summary of notations and operations used in this dissertation.

Table 2.1: A summary of notations and operations used in this thesis.

Notation	Description
x	Scalar
$\boldsymbol{x}$	Vector
$\mathbf{X}$	Matrix
$\underline{\mathbf{X}}$	Tensor
N	Number of order
I_n	Number of dimensionality of n th mode
J	Number of low-dimensionality
$\circ$	Outer product
$\otimes$	Kronecker product
$\odot$	Khatri-rao product
$*$	Hadamard product
$\underline{\mathbf{X}}_{(n)}$	n th mode matricization
$\times_n$	n th mode product

Chapter 3

Related Work

3.1 Nonnegative Matrix Factorization

Nonnegative Matrix Factorization (NMF) was firstly proposed by Paatero and Tapper as Positive Matrix Factorization [PT94]. They proposed the concept of NMF and successful examples, however, they did not provide efficient algorithms to obtain nonnegative factor matrices. Lee and Seung firstly proposed efficient algorithms for NMF [LS99, LS00] and made NMF popular in a wide range of areas such as image [GSV02, GVS03, YO05, LS99], text [CHHH11, PSBP04, JG10, DHS05, SBPP06, XLG03, GG05, DLPP06] and so on [SMTT16, STM⁺16, CZPA09].

NMF decomposes a nonnegative matrix $\mathbf{X} \in \mathbb{R}^{I_1 \times I_2}$ into a product of two smaller nonnegative matrices $\mathbf{U} \in \mathbb{R}^{I_1 \times J}$ and $\mathbf{V} \in \mathbb{R}^{I_2 \times J}$ ($J \ll I_1, I_2$):

$$\mathbf{X} \simeq \mathbf{U}\mathbf{V}^T.$$

The difference from the other traditional matrix factorization methods such as Singular Value Decomposition (SVD) or Principal Component Analysis (PCA) is the nonnegativity of resultant matrices. This nonnegativity provides several advantages as follows.

First one is the interpretability of the resultant matrices of NMF [LS99, CZPA09]. Since many data representations contain only nonnegative values (*e.g.*, word frequencies in document analysis or a spectrogram in sound analysis), nonnegativity gives physical meanings in the resultant matrices. If resultant matrices contain negative values, it is not easy to interpret directly. For example, what do negative word frequencies in document analysis supposed to mean? It is not easy to answer this question since we cannot observe such negative values in the real world. Traditional matrix factorization methods contain negative values in general, and thus, it is hard to interpret

the factorization results intuitively. On the other hand, NMF keeps nonnegativity on the resultant factor matrices. This enables to interpret the result physically and understand the structures behind data intuitively. Therefore, NMF is often used for exploratory data mining [LS99, LS00, CZPA09].

Second, the resultant matrices of NMF are sparse in general [EK04, Hoy04, KP07]. This is because negative values are not allowed in the decomposition and thus zero values are approximated by only zero values. Obtaining sparse representation is well considered in many areas such as dictionary learning. NMF is also considered as one of dictionary learning methods with nonnegative constraints [MBPS10, Hoy02].

Third, NMF has relationships between several methods such as clustering methods [DHS05, DLPP06] or Topic models [DLP08] and probabilistic models [CZPA09, NKLR⁺10, SKAU12, LS00, GG05, SS12]. For example, with orthogonal constraint, NMF has a connection to k-means clustering method and thus this extension is well used for clustering tasks [DLPP06, YC08]. This allows us to design NMF methods with known backgrounds or domain knowledges.

Unfortunately, nonnegative constraints make the optimization problem of NMF complicated. Vavasis proved that the problem of finding the exact decomposition of NMF is NP-hard [Vav09]. This means that the problem of finding an optimal solution of NMF is NP-hard, what is worse, the problem of determination of the rank of NMF (the size of resultant matrices J) is also NP-hard. Thus, many researches have proposed measurements or heuristics to determine the optimal rank [YKT12] so far. In this dissertation, I do not consider this rank problem and use J as a user given parameter. Arora *et al.* proposed an exact NMF algorithm. However, unfortunately this algorithm is intractable in most cases [AGKM12]. This complexity is one of biggest disadvantages for NMF, since SVD and PCA are known that the solution satisfy the best approximation accuracy w.r.t. Euclidean distance [WRR03].

There are many distortions to formulate NMF problems. The most famous and popular distance is squared Frobenius norm. Squared Frobenius of a matrix $\mathbf{X}$ is defined as:

$$\|\mathbf{X}\|_F^2 = \sum_{i_1, i_2} \mathbf{X}_{i_1 i_2}^2 = \text{Tr}(\mathbf{X}^T \mathbf{X}),$$

where Tr is the trace-norm. The second popular distance is Kullback-Leibler divergence (KL-divergence). KL-divergence between two nonnegative matrices $d_{KL}(\mathbf{X}|\mathbf{Y})$ is defined as:

$$d_{KL}(\mathbf{X}|\mathbf{Y}) = \sum_{ij} \mathbf{X}_{ij} \log \frac{\mathbf{X}_{ij}}{\mathbf{Y}_{ij}} - \mathbf{X}_{ij} + \mathbf{Y}_{ij}. \quad (3.1)$$

NMF with KL-divergence is related to Poisson distribution [NKL⁺10], and thus, often used in count data such as a term-document matrix in document analysis or a user-item matrix in recommendation or collaborative filtering problems. This is also one of advantages of NMF since KL-divergence allows only nonnegative values, and thus, matrix factorization with KL-divergence must be NMF. There are more NMF algorithm with the other distances have been proposed so far [CZA06, FBD09, CPC08, SD05].

3.1.1 Optimization

Optimizations for NMF are well researched and developed. This is because NMF problems are NP-hard [Vav09] and thus need to consider an efficient way to find a local minimum. Here, I give detailed explanations for Frobenius norm NMF which I are focusing on mainly in this dissertation. The other distances such as KL-divergence, α -divergence and β -divergence are introduced only briefly. For Bregman divergence, see Chapter 4

Frobenius norm

As I stated, Frobenius norm is one of most popular distances for NMF formulation:

$$\begin{aligned} \min_{\mathbf{U}, \mathbf{V}} \|\mathbf{X} - \mathbf{UV}^T\|_F^2, \\ \text{subject to } \mathbf{U}, \mathbf{V} \geq 0, \end{aligned} \quad (3.2)$$

This is because Lee and Seung proposed the first efficient optimization called Multiplicative Update rule (MU) [LS00] with this distortion. This paper brought NMF to machine learning or data mining community, and thus, many researches followed [Hoy04, CHHH11, DLPP06]. MU can be obtained by Kurush-Kuhn-Tucker conditions (KKT). KKT conditions for matrix $\mathbf{U}$ of this problem are:

$$\begin{aligned} \mathbf{U} &\geq 0, \\ \frac{\partial \|\mathbf{X} - \mathbf{UV}^T\|_F^2}{\partial \mathbf{U}} &\geq 0, \\ \mathbf{U} * \frac{\partial \|\mathbf{X} - \mathbf{UV}^T\|_F^2}{\partial \mathbf{U}} &= 0, \end{aligned} \quad (3.3)$$

where the gradient is calculated as:

$$\frac{\partial \|\mathbf{X} - \mathbf{UV}^T\|_F^2}{\partial \mathbf{U}} = -2\mathbf{XV} + 2\mathbf{UV}^T\mathbf{V}.$$

Algorithm 1 Multiplicative Update rule for Frobenius NMF

Input: Nonnegative matrix $\mathbf{X}$, Number of components J

Output: Nonnegative factor matrices $\mathbf{U}$ and $\mathbf{V}$ such that $\mathbf{X} \simeq \mathbf{UV}^T$.

Initialize $\mathbf{U}$ and $\mathbf{V}$ with random positive values.

repeat

$$\mathbf{U} \leftarrow \mathbf{U} * \frac{\mathbf{XV}}{\mathbf{UV}^T\mathbf{V}}$$

$$\mathbf{V} \leftarrow \mathbf{V} * \frac{\mathbf{X}^T\mathbf{U}}{\mathbf{V}^T\mathbf{U}^T\mathbf{U}}$$

until Convergence criterion is satisfied.

The last equation of KKT conditions (3.3) gives the following update rule:

$$\mathbf{U} \leftarrow \mathbf{U} * \frac{\mathbf{XV}}{\mathbf{UV}^T\mathbf{V}},$$

where $/$ is an element-wise division. With the same manner, the update rule for $\mathbf{V}$ can be obtained as:

$$\mathbf{V} \leftarrow \mathbf{V} * \frac{\mathbf{X}^T\mathbf{U}}{\mathbf{VU}^T\mathbf{U}}.$$

Since these update rules use only matrix multiplications and Hadamard products(or divisions), if initial values $\mathbf{U}$ and $\mathbf{V}$ is nonnegative, the updated matrices are guaranteed to be nonnegative. The algorithm of MU is shown in Algorithm 1. The initialization of NMF is one of research topics like that of k-means algorithms [AV07] and many sophisticated methods have been proposed so far [DLPP06, XTCC08, BG08]. However, in this dissertation, these are out of focus and we use random initializations.

Since MU is efficient and easy to derive the update rules with additional constraints [CHHH11, DLPP06, SD05, YC08], it is often used. However, it is known that Mu needs large number of iterations to converge. Thus, more efficient algorithms have been considered. Lin proposed projected gradient descent for NMF (PGD) [Lin07]. This method uses a simple projected gradient descent for the updating:

$$\mathbf{U} \leftarrow \mathbf{U} - \eta(-2\mathbf{XV} + 2\mathbf{UV}^T\mathbf{V}),$$

where η is a step-size parameter. This is tuned by back tracking for each update. After updating, the matrix $\mathbf{U}$ is projected to be nonnegative as:

$$\mathbf{U} \leftarrow [\mathbf{U}]_+,$$

where $[x]_+ = \max(\epsilon, x)$ (ϵ is a sufficiently small and positive value). Since MU can be considered as a fixed step-size gradient descent not to be negative:

$$\mathbf{U} * \frac{\mathbf{X}\mathbf{V}}{\mathbf{U}\mathbf{V}^T\mathbf{V}} = \mathbf{U} - \eta * (-2\mathbf{X}\mathbf{V} + 2\mathbf{U}\mathbf{V}^T\mathbf{V}) \iff \eta = \mathbf{U} * \frac{1}{2\mathbf{U}\mathbf{V}^T\mathbf{V}}$$

PGD converges faster than MU since they search the step-size η with ignoring nonnegativity.

In contrast to the above two matrix-wise updating algorithms, Cichocki *et al.* proposed vector-wise update algorithm called Hierarchical Alternating Least Squares algorithm (HALS) [CAH09]. They found that a vector-wise problem can be solved in closed-form. With the residual without j th vectors of $\mathbf{U}$ and $\mathbf{V}$, the vector-wise problem can be written as:

$$\|\mathbf{X}^{(j)} - \mathbf{u}_j \mathbf{v}_j^T\|_F^2, \quad (3.4)$$

where $\mathbf{X}^{(j)} = \mathbf{X} - \sum_{k \neq j} \mathbf{u}_k \mathbf{v}_k^T$ is the residual without j th vectors. The gradient in $\mathbf{u}_j$ of (3.4) is calculated as:

$$\frac{\partial \|\mathbf{X}^{(j)} - \mathbf{u}_j \mathbf{v}_j^T\|_F^2}{\mathbf{u}_j} = -2\mathbf{X}^{(j)} \mathbf{v}_j + 2\mathbf{u}_j \mathbf{v}_j^T \mathbf{v}_j. \quad (3.5)$$

By setting gradient (3.5) to be zero, we have a optimal solution:

$$\mathbf{u}_j^* = \left[\frac{\mathbf{X}^{(j)} \mathbf{u}_j}{\mathbf{v}_j^T \mathbf{v}_j} \right]_+.$$

The explicit column normalization of one matrix $\|\mathbf{u}_j\|_2 = 1$ after updating enables to ignore scalar $\mathbf{v}_j^T \mathbf{v}_j$. In summary, the update rule of each vector of $\mathbf{U}$ and $\mathbf{V}$ are:

$$\begin{aligned} \mathbf{u}_j &= [\mathbf{X}^{(j)} \mathbf{v}_j]_+ \\ \mathbf{v}_j &= [\mathbf{X}^{(j)T} \mathbf{u}_j]_+ \end{aligned}$$

This algorithm converges faster than MU or PGD since each vector-wise update achieves optimal solutions in each update. The pseudo-code of HALS is shown in Algorithm 2. HALS algorithm converges faster than MU, however, it needs calculate the residual $\mathbf{X}^{(j)}$ for each vector update. This takes $O(I_1 I_2 J)$ and unfortunately inefficient. Cichocki *et al.* avoid this calculation by using $\mathbf{X}^{(j)} = \mathbf{X} - \mathbf{U}\mathbf{V}^T + \mathbf{u}_j \mathbf{v}_j^T$. Using this equation, update rules become:

$$\begin{aligned} \mathbf{u}_j &= [\mathbf{X} \mathbf{v}_j - \mathbf{U}(\mathbf{V}^T \mathbf{V})_j + \mathbf{u}_j \mathbf{v}_j^T \mathbf{v}_j]_+ \\ \mathbf{v}_j &= [\mathbf{X}^T \mathbf{u}_j - \mathbf{V}(\mathbf{U}^T \mathbf{U})_j + \mathbf{v}_j \mathbf{u}_j^T \mathbf{u}_j]_+ \end{aligned}$$

Algorithm 2 Hierarchal Alternating Least Squares Algorithm for NMF

Input: Nonnegative matrix $\mathbf{X}$, Number of components J

Output: Decomposing nonnegative matrices $\mathbf{U}$ and $\mathbf{V}$ such that $\mathbf{X} \simeq \mathbf{UV}^T$.

Initialize $\mathbf{U}$ and $\mathbf{V}$ arbitrary.

$\mathbf{R} = \mathbf{X} - \mathbf{UV}^T$

repeat

for $j = 1$ **to** J **do**

$\mathbf{X}^{(j)} = \mathbf{R} + \mathbf{u}_j \mathbf{v}_j^T$

$\mathbf{u}_j = [\mathbf{X}^{(j)} \mathbf{v}_j]_+$

$\mathbf{u}_j = \mathbf{u}_j / \|\mathbf{u}_j\|_2$

$\mathbf{v}_j = [\mathbf{X}^{(j)T} \mathbf{u}_j]_+$

$\mathbf{R} = \mathbf{X}^{(j)} - \mathbf{u}_j \mathbf{v}_j^T$

end for

until Convergence criterion is satisfied.

As shown in Algorithm 2, HALS updates vectors in order $\mathbf{u}_1, \mathbf{v}_1, \mathbf{u}_2, \mathbf{v}_2, \dots, \mathbf{u}_J, \mathbf{v}_J$. On the other hand, their improved method called Fast HALS updates vectors in order $\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_J, \mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_J$. This avoid the duplicative calculation of $\mathbf{XV}$ or $\mathbf{X}^T \mathbf{U}$ in their update rules. The pseudo-code of Fast HALS algorithm is shown in Algorithm 3. Ho *et al.* also independently developed a vector-wise update algorithm called Rank-one Residue Iteration (RRI). This is the same algorithm to HALS. However, it is slower than Fast HALS since they need to calculate the residual in each vector update.

Kim and Park proposed an active-set like method for NMF [KP11]. This algorithm also update column-wise but different from HALS. They uses active-like method to minimize the problem (3.4), however, the efficiency and approximation accuracies are almost the same, and thus, I omit that here. This is because both algorithms achieve optimal solutions in each vector update.

Recently, Hsieh and Dhillon proposed an element-wise update algorithm called Greedy Coordinate Descent (GCD) [HD11]. The element-wise optimal solution is also achieved in HALS or active-set like method, however, these algorithms update all elements at once for each iteration even it should be update again or should not be update anymore. Note that the element-wise optimal solution is achieved, however, elements in the same row of $\mathbf{U}$ or $\mathbf{V}$ depends on each other and thus optimal solution may change after updating the other elements in the same row. On the other hand, GCD updates

Algorithm 3 Fast Hierarchal Alternating Least Squares Algorithm for NMF

Input: Nonnegative matrix $\mathbf{X}$, Number of components J

Output: Decomposing nonnegative matrices $\mathbf{U}$ and $\mathbf{V}$ such that $\mathbf{X} \simeq \mathbf{UV}^T$.

Initialize $\mathbf{U}$ and $\mathbf{V}$ arbitrary.

repeat

$\mathbf{A} = \mathbf{XV}$

$\mathbf{B} = \mathbf{V}^T \mathbf{V}$

for $j = 1$ **to** J **do**

$\mathbf{u}_j = [\mathbf{a}_j - \mathbf{U}\mathbf{b}_j + \mathbf{u}_j\mathbf{B}_{jj}]_+$

$\mathbf{u}_j = \mathbf{u}_j / \|\mathbf{u}_j\|_2$

end for

$\mathbf{C} = \mathbf{X}^T \mathbf{F}$

$\mathbf{D} = \mathbf{F}^T \mathbf{F}$

for $j = 1$ **to** J **do**

$\mathbf{v}_j = [\mathbf{c}_j - \mathbf{V}\mathbf{d}_j + \mathbf{v}_j\mathbf{D}_{jj}]_+$

end for

until Convergence criterion is satisfied.

elements numerous times in one iteration. It calculates how elements update with closed-form solutions decreases the value of objective function. GCD considers element-wise objective functions as:

$$\min_s g_{i_1j}(s) \equiv \|\mathbf{X} - (\mathbf{U} + \mathbf{E}_{i_1j}s)\mathbf{V}^T\|_F^2, \quad (3.6)$$

where $\mathbf{E}_{i_1j}$ is a matrix have 1 for only its (i_1, j) element and 0 for the others. Thus, how update (i_1, j) element of $\mathbf{U}$ can be calculated as:

$$D_{i_1j} = g_{i_1j}(0) - g_{i_1j}(s).$$

Since $g_{i_1j}(s)$ can be expressed by Taylor expansion:

$$g_{i_1j}(s) \simeq g_{i_1j}(0) + (g_{i_1j})'(0)s + \frac{1}{2}(g_{i_1j})''(0)s^2.$$

Therefore,

$$D_{i_1j} = (\mathbf{UV}^T\mathbf{V} - \mathbf{XV})_{i_1j}s_{i_1j} + \frac{1}{2}(\mathbf{V}^T\mathbf{V})_{jj}s_{i_1j}^2.$$

Since s can be found optimally by the same way in HALS and both s and D_{i_1j} can be updated easily (each update takes only $O(J)$), GCD converges faster than the other methods. To the best of my knowledge, this algorithm

Algorithm 4 Greedy Coordinate Descent Update for Matrix $\mathbf{U}$

Input: Nonnegative matrix $\mathbf{X}$, Number of components J

Output: Find a nonnegative matrix $\mathbf{U}$ such that minimize $\|\mathbf{X} - \mathbf{U}\mathbf{V}^T\|^2$ with fixed $\mathbf{V}$.

tolerance $\epsilon = 1e - 4$, Max Number of Iteration $k = J^2$

$\mathbf{U}^{new} \leftarrow \mathbf{0}_{N \times J}$

$\mathbf{G}_{ij}^U = -2\mathbf{X}\mathbf{V}_{ij} + 2(\mathbf{U}\mathbf{V}^T\mathbf{V})_{ij}$

$\mathbf{S} \leftarrow \max(\mathbf{U}_{ij} - \frac{\mathbf{G}_{ij}^U}{(\mathbf{V}^T\mathbf{V})_{jj}}, 0) - \mathbf{U}_{ij}$ for all i, j

$\mathbf{D}_{ij} \leftarrow -\mathbf{G}_{ij}^U\mathbf{S}_{ij} - \frac{1}{2}(\mathbf{V}^T\mathbf{V})_{jj}(\mathbf{S}_{ij})^2$ for all i, j

$q_i \leftarrow \underset{j}{\operatorname{argmax}} \mathbf{D}_{ij}$ for all $i = 1, \dots, N$

$p^{init} \leftarrow \mathbf{D}_{i, q_i}$

for $i = 1$ **to** N **do**

while $\mathbf{D}_{i, q_i} < \epsilon p^{init}$ **do**

$s^* \leftarrow \mathbf{S}_{i, q_i}$

$\mathbf{U}_{i, q_i}^{new} \leftarrow \mathbf{U}_{i, q_i}^{new} + s^*$

$\mathbf{G}_{i:}^U \leftarrow \mathbf{G}_{i:}^U + s^*(\mathbf{V}^T\mathbf{V})_{q_i:}$

$\mathbf{S}_{ij} \leftarrow \max(\mathbf{U}_{ij} - \frac{\mathbf{G}_{ij}^U}{(\mathbf{V}^T\mathbf{V})_{jj}}, 0) - \mathbf{U}_{ij}$ for all j

$\mathbf{D}_{ij} \leftarrow -\mathbf{G}_{ij}^U\mathbf{S}_{ij} - \frac{1}{2}\mathbf{V}^T\mathbf{V}(\mathbf{S}_{ij})^2$ for all j

$q_i \leftarrow \underset{j}{\operatorname{argmax}} \mathbf{D}_{ij}$

end while

end for

$\mathbf{U} \leftarrow \mathbf{U} + \mathbf{U}^{new}$

is the fastest algorithm for NMF with Frobenius norm. The pseudo-code of GCD is shown in Algorithm 4.

In summary, efficient algorithms for NMF are focusing on the convergence speed and the small number of iterations to converge. Indeed, the computational cost for one iterations is almost the same $O(I_1 I_2 J)$ in all algorithms.

Biggs *et al.* proposed an interesting algorithm for NMF called Rank-1 residual iteration [BGV08]. This algorithm utilizes the fact that rank-1 SVD for a nonnegative matrix can find the best rank-1 nonnegative solution. Thus it finds one by one column with choosing indices not to make residual negative. But unfortunately, this algorithm is slower than the others.

The other distances and divergences

Another popular distance used for NMF is KL-divergence. Lee and Seung also proposed MU for KL-divergence in their paper [LS00]:

$$\begin{aligned} \mathbf{U}_{i1j} &\leftarrow \mathbf{U}_{i1j} * \frac{(((\mathbf{U}\mathbf{V}^T)/(\mathbf{X}))\mathbf{V})_{i1j}}{(\mathbf{1}\mathbf{V})_{i1j}} \\ \mathbf{V}_{i2j} &\leftarrow \mathbf{V}_{i2j} * \frac{(((\mathbf{U}\mathbf{V}^T)/(\mathbf{X}))^T\mathbf{U})_{i2j}}{(\mathbf{1}\mathbf{U})_{i2j}} \end{aligned}$$

Similar to Algorithm 1, I update $\mathbf{U}$ and $\mathbf{V}$ iteratively until these converge. This MU is also relatively inefficient since it is a matrix-wise update and thus it uses a fixed-size parameter. A vector-wise update was proposed by Hsieh *et al.* [HD11] like HALS for NMF with Frobenius norm. However, in contrast to HALS, the closed-form solution cannot be found even vector-wise problems. Thus, it needs projected gradient descent based optimizations like Lin did [Lin07]. Hsieh *et al.* proposed simple Newton-method called Cyclic Coordinate Descent (CCD). It solved vector-wise problem and update by Newton method until convergence. This method is simple, however, efficient to decrease objective functions. Nguyen *et al.* proposed improved CCD by utilizing sparsity of target matrix [NH16]. Since KL-divergence NMF is related to Poisson distribution and its often appears in count data such as document or user-item matrix in recommendation system. In general, these matrices are sparse so this is a suitable assumption. They also randomized the order of updating vectors. This guarantees the convergence of their algorithm.

To handle NMF problems in a general framework, NMF with general divergences have been proposed so far. Cichocki *et al.* proposed NMF with β -divergence and α divergence [CPC08, CZPA09]. They proposed a vector-wise problem with approximations, however, its computation cost is large [CAH09]. Févotte also proposed NMF with β -divergence [FI11]. Bregman divergence, a wide class of divergences, also contains KL-divergence and even Frobenius norm, have been considered so far. Sra *et al.* proposed MU algorithm for Bregman divergence [SD05] and Li *et al.* proposed a vector-wise algorithm [LLP12]. Since Bregman divergence contains Frobenius norm or KL-divergence, these algorithm also are available for these problem, however, GCD for Frobenius norm of CCD for KL-divergence is much faster than these methods.

There are still many variant of NMF such as Manhattan Distance [GTLST12] or Csizer divergence [CZA06]. In Robust NMF, designing distance were considered [KDH11, DLS12, GNCH15]. These methods deal with vector-wise out-lier data.

Online settings for NMF have been also considered [WLK11, CSS⁺07, WSL14, GTLY12, ZT16]. In Online NMF, a vector comes one by one and this is the same setting to online dictionary learning or online sparse coding [MBPS10]. Online NMF can be used for tracking of videos [WSL14, WSL14], documents [WLK11, CSS⁺07] and sounds [ZYXY11]. Hayashi *et al.* proposed an online NMF on a different setting [HMTK15]. They considered updates of information, therefore, a new nonnegative matrix comes one by one with the same index. This also can be considered as an online tensor factorization. These methods are efficient but out of focus of this dissertations. This is because the approximation accuracies of those methods are relatively lower than the that of batch NMF methods.

3.1.2 Extensions and Variants of NMF

Extendability and flexibility is also one of advantages of NMF. Since the objective function of NMF is simple and optimization and it is not based on eigen value problems, it is relatively easy to add constraints into the objective function. Thus, many variants and extensions have been proposed so far.

Sparse constraint is one of most considered extensions [Hoy02, Hoy04, EK04]. One of advantages of NMF is sparsity, however, this is a side effect of nonnegative constraint. Therefore, it is not guaranteed in all cases. Hence, several sparse NMF methods to guarantee or to enhance the sparsity have been proposed so far [KP07, PC11, Hoy04, MMH08]. They added constraint term with L_0 or L_1 norm to enhance sparsity into objective function. On nonnegativity constraint, manipulating L_1 constraint is relatively easy and thus simple optimization can be used [MMH08]. Hoyer proposed sparse NMF with enforcing sparsity given by users [Hoy04].

Graph regularization method is also one of well considered constraints. Cai *et al.* proposed Graph regularized NMF. They used graph Laplacian based regularizer called Graph regularization which is used in manifold learning methods such as Laplacian Eigen Map (LEM) [BN01] or Locality Preserving Projection (LPP) [Niy04]. This method ignores the approximation accuracy but preserving similarity given by a similarity graph [BN01]. This method can obtain coefficient matrix with preserving the graph structure and neighborhoods in the original feature space, and therefore, it performs well in clustering tasks and proposed many methods with this constraint [CHHH11, GZ09, TZBG14, ZZZ⁺14, GTLY11, SS10].

Ding *et al.* proposed Orthogonal constrained NMF (ONMF), which impose orthogonality to one-side of resultant matrices [DLPP06]. This brings sparse representation and connections to clustering methods [DLPP06, DHS05]. Since this orthogonal constraints is desirable in many tasks, many

researches are follows [GZ09, CWZ09, WLZ08, LYF⁺10, LWGH13, PKA10]. This method will be described later in Chapter 4.

Semi-NMF eases the nonnegativity and allows negative values on one-side matrix [DLJ10, GK15, YCI12]. This makes NMF enable to apply negative matrices. Trigeorgis *et al.* stacked this semi-NMF and construct Deep Semi-NMF which can extract different level of features on face image dataset [TBZS14]. This method is based on hierarchical NMF proposed by Song and Lee [SL13]. Ding *et al.* also proposed convex NMF [DLJ10], which decompose NMF with a linear combination of data point itself. This helps to find a convex set in the data [KWTB10, TKB09]. This method is also known as projective NMF [YO05, YYL07, YO10].

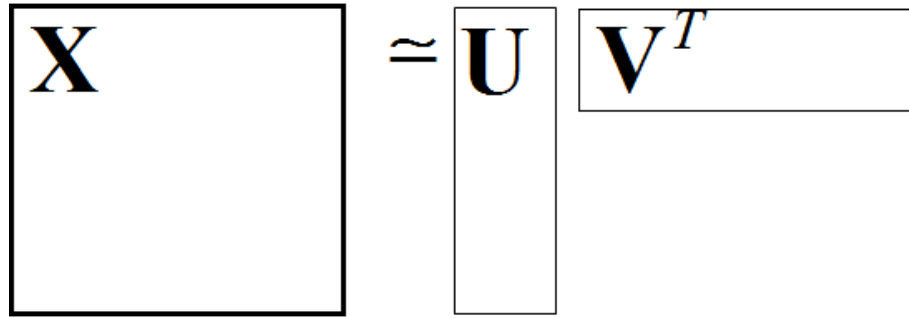
Binary Matrix Factorization or Boolean Matrix Factorization can be considered as one of variants of NMF [Mie10, SPK08, ZLDZ07]. Their values are restricted to be binary, and thus, these are more strict classes of NMF [Mie10]. These methods fit to the dyadic data with binary representations [MV14, MV11, ZLD⁺10]. It was proved that these problems are also NP-hard, in addition, the optimization is much more difficult to solve [EM13, Mie10, MMG⁺08].

Separable NMF considers the exact factorization with assuming that target matrix satisfy the separability condition [DS03], that is, all data points can be represented by a nonnegative linear combination anchor points. In other words, all data points are inside of a cone with some anchor points (chosen data points) [KSK13]. X-Ray can find anchor points efficiently one by one in greedily manners. Zhou *et al.* also proposed divide and conquer method [ZBT13]. A linear programming based method was also proposed [GL14, GV14].

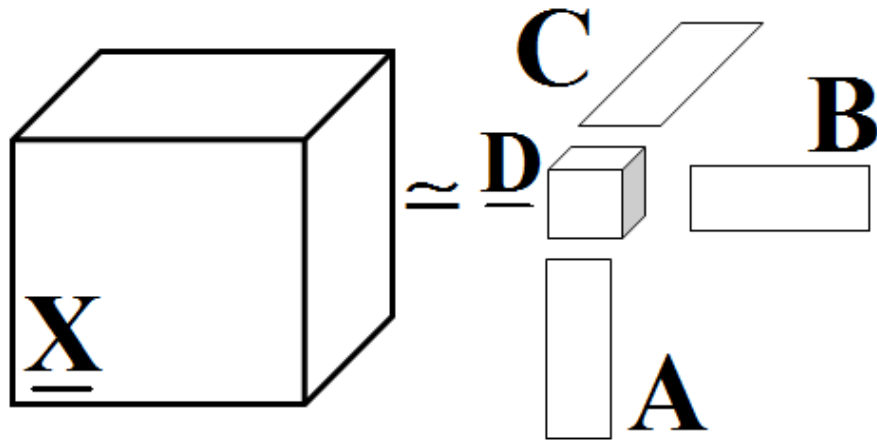
Convolutional NMF or Nonnegative Matrix Deconvolution (NMD) is NMF with a convolution, which considers time varying feature with convolution. In music analysis, time correlations are important and NMD can extract melodies as feature with time [WCC09, OP06, Sma04, SM06]. In the sense of the form of feature set, this method factorize a matrix with a product of tensor and a matrix with time shifting [FCC05]. Shifted NMF is also one of them [JFCR11, JFB⁺11].

3.2 Tensor Factorization

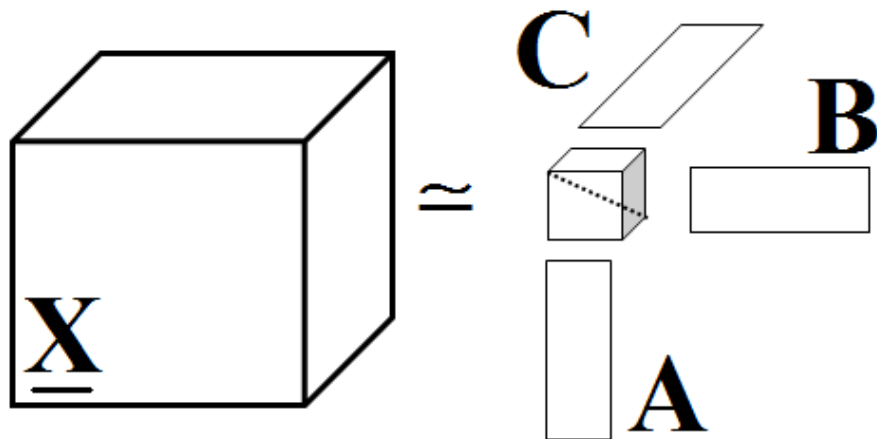
A tensor, a multidimensional array, has become popular to represent a polynomial relation data such as Electroencephalogram (EEG) [CZPA09, CPZ⁺12a], images with different angles or illuminations [VT03], videos (images with time) [LPV11, LPV08] and social networks [ADK09, BBB08]. Ten-



(a) Nonnegative Matrix Factorization



(b) Tucker Decomposition



(c) CANDECOMP/PARAFAC Decomposition

Figure 3.1: Illustrations of nonnegative matrix factorization and tensor factorizations

tensor factorization is one of most popular methods to analyze a tensor data [KB09, Cic14, SDLF⁺16]. This is because tensor factorization is a generalization of matrix factorization, and thus, it is easy to use as a feature extraction method, a dimension reduction method, and a data compression method for tensor data. A tensor can be decompose in various way unlike matrix factorization. In this chapter, I provide explanations for two most popular tensor factorization form, Tucker decomposition and CANDECOMP/PARAFAC Decomposition (CPD) in detail. We only introduce the other tensor factorization forms briefly.

3.2.1 Tucker Decomposition

Tucker decomposition decomposes a tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ into mode products of a low-dimensional tensor called *core tensor* $\underline{\mathbf{D}} \in \mathbb{R}^{J_1 \times J_2 \times J_3}$ ($J_1 \ll I_1$, $J_2 \ll I_2$ and $J_3 \ll I_3$) and matrices $\mathbf{A} \in \mathbb{R}^{I_1 \times J_1}$, $\mathbf{B} \in \mathbb{R}^{I_2 \times J_2}$ and $\mathbf{C} \in \mathbb{R}^{I_3 \times J_3}$:

$$\underline{\mathbf{X}} \simeq \underline{\mathbf{D}} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C}.$$

This factorization form is a general form of tensor factorizations and the minimization problem is defined as:

$$\min_{\mathbf{A}, \mathbf{B}, \mathbf{C}, \underline{\mathbf{D}}} \|\underline{\mathbf{X}} - \underline{\mathbf{D}} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C}\|_F^2.$$

This factorization problem with exact approximation is NP-hard even without nonnegativity constraint [DLDMV00a]. However, an efficient algorithm, called Hi-Order SVD (HOSVD), makes this method popular and useful. HOSVD conducts SVD for matricized tensor $\underline{\mathbf{X}}_{(1)}, \underline{\mathbf{X}}_{(2)}$ and $\underline{\mathbf{X}}_{(3)}$ for each mode and uses left singular valued matrices as factorization results:

$$\begin{aligned} \underline{\mathbf{X}}_{(1)} &\simeq \mathbf{A} \Sigma^{(1)} \mathbf{V}^{(1)}, \\ \underline{\mathbf{X}}_{(2)} &\simeq \mathbf{B} \Sigma^{(2)} \mathbf{V}^{(2)}, \\ \underline{\mathbf{X}}_{(3)} &\simeq \mathbf{C} \Sigma^{(3)} \mathbf{V}^{(3)}. \end{aligned}$$

After obtaining matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$. They calculate core tensor $\underline{\mathbf{D}}$ as:

$$\underline{\mathbf{X}} = \underline{\mathbf{D}} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C} \Rightarrow \underline{\mathbf{D}} = \underline{\mathbf{X}} \times_1 \mathbf{A}^T \times_2 \mathbf{B}^T \times_3 \mathbf{C}^T. \quad (3.7)$$

Note that this equation is hold since matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ are orthonormal $\mathbf{A}^T \mathbf{A} = \mathbf{I}$, $\mathbf{B}^T \mathbf{B} = \mathbf{I}$ and $\mathbf{C}^T \mathbf{C} = \mathbf{I}$. In addition, the dimension of core tensor $\underline{\mathbf{D}}$ for each mode is also obtained by the rank of corresponding SVD. Since it is not an iterative algorithm unlike the other tensor factorization and the dimension of tensor is easily obtained by each SVD, this method is one of most

Algorithm 5 Hi-Order Singular Value Decomposition (HOSVD)

Input: A Tensor $\underline{\mathbf{X}}$, the size of core tensor J_1 , J_2 and J_3 .

Output: Factor matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ and a core tensor $\underline{\mathbf{D}}$ such that $\underline{\mathbf{X}} \simeq \underline{\mathbf{D}} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C}$.

$\mathbf{A} \leftarrow$ the J_1 leading left singular vectors of $\underline{\mathbf{X}}_{(1)}$.

$\mathbf{B} \leftarrow$ the J_2 leading left singular vectors of $\underline{\mathbf{X}}_{(2)}$.

$\mathbf{C} \leftarrow$ the J_3 leading left singular vectors of $\underline{\mathbf{X}}_{(3)}$.

$\underline{\mathbf{D}} \leftarrow \underline{\mathbf{X}} \times_1 \mathbf{A}^T \times_2 \mathbf{B}^T \times_3 \mathbf{C}^T$

popular optimization methods for Tucker decomposition. Unfortunately, the approximation accuracy by HOSVD is not accurate, although, it has a theoretical lower bound of approximation error [DLDMV00a]. The pseudo-code of HOSVD is shown in Algorithm 5.

As stated above, HOSVD does not factorize a tensor accurately. This is because matrices are obtained independently. Thus, Lathauwer *et al.* proposed an iterative method called Hi-Order Orthogonal Iteration (HOOI) [DLDMV00b].¹

HOOI first conducts HOSVD by Algorithm 5 and obtain the initial conditions of $\underline{\mathbf{D}}$, $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$. After the initialization, HOOI conducts SVD with projected a smaller tensor with the other matrices. For example, to obtain matrix $\mathbf{A}$, HOOI projects the tensor $\underline{\mathbf{X}}$ by using $\mathbf{B}$ and $\mathbf{C}$ as follows:

$$\underline{\mathbf{Y}} = \underline{\mathbf{X}} \times_2 \mathbf{B}^T \times_3 \mathbf{C}^T$$

then decompose this small tensor $\underline{\mathbf{Y}} \in \mathbb{R}^{I_1 \times J_2 \times J_3}$ as:

$$\underline{\mathbf{Y}}_{(1)} \simeq \mathbf{A} \Sigma^{(1)} \mathbf{V}^{(1)}.$$

The other matrices $\mathbf{B}$ and $\mathbf{C}$ are also obtained likewise. HOOI repeats updating matrices until convergence. The core tensor $\underline{\mathbf{D}}$ is obtained by (3.7). The pseudo-code of HOOI is shown in Algorithm 6. Since the approximation accuracy is relatively good and the formulation is simple, this method is often used in feature extraction or tensor completion and as dimension reduction for tensors. However, this algorithm is iterative and thus it needs larger computational cost than HOSVD [KS08].

3.2.2 CANDECOMP/PARAFAC Decomposition

CANDECOMP/PARAFAC Decomposition (CPD) is a special form of tucker decomposition. CPD restricts the core tensor $\underline{\mathbf{D}}$ to be super-diagonal. Super-

¹This method is also known as Hi-Order Power Method

Algorithm 6 Hi-Order Orthogonal Iteration (HOOI)

Input: A Tensor $\underline{\mathbf{X}}$, the size of core tensor J_1 , J_2 and J_3 .

Output: Factor matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ and a core tensor $\underline{\mathbf{D}}$ such that $\underline{\mathbf{X}} \simeq \underline{\mathbf{D}} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C}$.

obtain $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ and $\underline{\mathbf{D}}$ by HOSVD (Algorithm 5).

repeat

$\underline{\mathbf{X}}' \leftarrow \underline{\mathbf{D}} \times_1 \mathbf{A}$

$\mathbf{A} \leftarrow$ the J_1 leading left singular vectors of $\underline{\mathbf{X}}'_{(1)}$.

$\underline{\mathbf{X}}' \leftarrow \underline{\mathbf{X}}' \times_1 \mathbf{A}^T \times_2 \mathbf{B}$

$\mathbf{B} \leftarrow$ the J_2 leading left singular vectors of $\underline{\mathbf{X}}'_{(2)}$.

$\underline{\mathbf{X}}' \leftarrow \underline{\mathbf{X}}' \times_2 \mathbf{B}^T \times_3 \mathbf{C}$

$\mathbf{C} \leftarrow$ the J_3 leading left singular vectors of $\underline{\mathbf{X}}'_{(3)}$.

until Coverage criterion is satisfied

$\underline{\mathbf{D}} \leftarrow \underline{\mathbf{X}}' \times_3 \mathbf{C}^T$

diagonal condition is a generalization of diagonal condition of matrix where

$$\underline{\mathbf{D}}_{i_1 i_2 i_3} = \begin{cases} \underline{\mathbf{D}}_{i_1 i_2 i_3} & \text{if } i_1 = i_2 = i_3, \\ 0 & \text{otherwise.} \end{cases}$$

Note that the core tensor $\underline{\mathbf{D}}$ must be the hyper-cube, that is, the condition of dimensions of all modes must be the same $J = J_1 = J_2 = J_3$. CPD can also be consider as one of generalizations of matrix factorization since CPD for the 2-order tensor (matrix) is the same to SVD.

The approximation accuracy of CPD is relatively lower than Tucker decomposition since the core tensor is restricted to be super-diagonal. However, this makes factorization result be easy to interpret. Like a matrix factorization can be written as the sum of outer-products of two vectors:

$$\mathbf{X} = \sum_{j=1}^J \mathbf{u}_1 \circ \mathbf{v}_1^T + \mathbf{u}_2 \circ \mathbf{v}_2^T + \cdots + \mathbf{u}_J \circ \mathbf{v}_J^T,$$

CPD can be written in the same manner as:

$$\underline{\mathbf{X}} = \sum_{j=1}^J \underline{\mathbf{D}}_{jjj} \mathbf{a}_j \circ \mathbf{b}_j \circ \mathbf{c}_j.$$

This representation is relatively easier to interpret than Tucker decomposition. For example, when time-varying network data $\underline{\mathbf{X}} \in \mathbb{R}^{\text{Node} \times \text{Node} \times \text{Time}}$

is considered, CPD decomposes $\underline{\mathbf{X}}$ into J products of membership vectors for nodes $\mathbf{a}_j \in \mathbb{R}^{Node}$ and $\mathbf{b}_j \in \mathbb{R}^{Node}$ and time activations $\mathbf{c}_j$, that is, the history of how j th community composed by $\mathbf{a}_j \circ \mathbf{b}_j$ be active on the past data [BBB08, ADK09]. This enables to understand the network structure behind the time-varying network. On the other hand, Tucker decomposition can also decompose this tensor $\underline{\mathbf{X}}$, although, the core tensor must be analyzed. Therefore, the result of Tucker decomposition is hard to interpret. Furthermore, this super-diagonal constraint brings uniqueness under some conditions [Kru77, tBS02, SB00]:

$$J_{\mathbf{A}} + J_{\mathbf{B}} + J_{\mathbf{C}} \geq 2J + 2, \quad (3.8)$$

where $J_{\mathbf{A}}, J_{\mathbf{B}}, J_{\mathbf{C}}$ are the rank of $\mathbf{A}, \mathbf{B}$ and $\mathbf{C}$, respectively. This condition is hold in general since $\mathbf{A}, \mathbf{B}$ and $\mathbf{C}$ are linear independent in most cases. Since Tucker decomposition is not unique and even matrix factorization is not unique [KB09, Kru77], this uniqueness property of CPD is the strong advantage to the others.

The super-diagonal constraint core tensor makes unable to use SVD based optimization, thus we need to optimize these problem by alternating updates. Since the diagonal elements are scaling of factor matrices and thus it can be ignored [CAH09], the optimization problem can be rewritten as:

$$\min_{\mathbf{A}, \mathbf{B}, \mathbf{C}} \|\underline{\mathbf{X}} - \underline{\mathbf{I}} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C}\|_F^2,$$

where

$$\underline{\mathbf{I}}_{j_1 j_2 j_3} = \begin{cases} 1 & \text{if } j_1 = j_2 = j_3, \\ 0 & \text{otherwise.} \end{cases}$$

We use the matricization and transform to solve the above problems as:

$$\begin{aligned} \|\underline{\mathbf{X}} - \underline{\mathbf{I}} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C}\|_F^2 &= \|\underline{\mathbf{X}}_{(1)} - \mathbf{A} \underline{\mathbf{I}}_{(1)} (\mathbf{C} \otimes \mathbf{B})^T\|_F^2 \\ &= \|\underline{\mathbf{X}}_{(2)} - \mathbf{B} \underline{\mathbf{I}}_{(2)} (\mathbf{B} \otimes \mathbf{A})^T\|_F^2 \\ &= \|\underline{\mathbf{X}}_{(3)} - \mathbf{C} \underline{\mathbf{I}}_{(3)} (\mathbf{C} \otimes \mathbf{A})^T\|_F^2. \end{aligned}$$

In addition, the super-diagonal tensor reduce the complexity of Kronecker product to the Khatri-rao product as:

$$\underline{\mathbf{I}}_{(1)} (\mathbf{C} \otimes \mathbf{B})^T = (\mathbf{C} \odot \mathbf{B})^T. \quad (3.9)$$

Therefore, the problem can be rewritten as:

$$\begin{aligned} \|\underline{\mathbf{X}} - \underline{\mathbf{I}} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C}\|_F^2 &= \|\underline{\mathbf{X}}_{(1)} - \mathbf{A} (\mathbf{C} \odot \mathbf{B})^T\|_F^2 \\ &= \|\underline{\mathbf{X}}_{(2)} - \mathbf{B} (\mathbf{B} \odot \mathbf{A})^T\|_F^2 \\ &= \|\underline{\mathbf{X}}_{(3)} - \mathbf{C} (\mathbf{C} \odot \mathbf{A})^T\|_F^2. \end{aligned}$$

Algorithm 7 CANDECOMP/PARAFAC Decomposition (CPD)

Input: A Tensor $\underline{\mathbf{X}}$, the size of core tensor J .

Output: Factor matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ such that $\underline{\mathbf{X}} \simeq \underline{\mathbf{I}} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C}$. Initialize $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ with random values.

repeat

$$\mathbf{A} \leftarrow \underline{\mathbf{X}}_{(1)}(\mathbf{C} \odot \mathbf{B})(\mathbf{B}^T \mathbf{B} * \mathbf{C}^T \mathbf{C})^\dagger$$

$$\mathbf{B} \leftarrow \underline{\mathbf{X}}_{(2)}(\mathbf{C} \odot \mathbf{A})(\mathbf{A}^T \mathbf{A} * \mathbf{C}^T \mathbf{C})^\dagger$$

$$\mathbf{C} \leftarrow \underline{\mathbf{X}}_{(3)}(\mathbf{B} \odot \mathbf{A})(\mathbf{A}^T \mathbf{A} * \mathbf{B}^T \mathbf{B})^\dagger$$

until Coverage criterion is satisfied

These problem are now only have matrices and it is the same to Least Squares problem and thus the optimal solution can be found easily. CPD updates $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ iteratively until these converges.

In contrast to Tucker decomposition with HOSVD, CPD has a computational problem. That is a product of Khatri-rao product of two matrices and the matricized target tensor. For example, the closed-form solution of $\mathbf{A}$ with fixed $\mathbf{B}$ and $\mathbf{C}$ is:

$$\underline{\mathbf{X}}_{(1)}(\mathbf{C} \odot \mathbf{B})(\mathbf{B}^T \mathbf{B} * \mathbf{C}^T \mathbf{C})^\dagger,$$

where comes from $(\mathbf{B}^T \mathbf{B} * \mathbf{C}^T \mathbf{C})$ the equation (2.1). The problem is in the term $\underline{\mathbf{X}}_{(1)}(\mathbf{C} \odot \mathbf{B})$. The Khatri-rao product $(\mathbf{C} \odot \mathbf{B}) \in \mathbb{R}^{I_1 I_2 \times J}$ needs computational complexity $O(I_1 I_2 J)$ for both space and time. This calculation is large and makes CPD intractable on large-scale problems. This is called *Intermediate Data Explosion* (IDE) problem [PFS12, P⁺13, JPKF15, KS08, KPHF12, CV14, JCW⁺14, SRSK15, LHCS16]. HOSVD and HOOI also needs larger calculation to obtain the core tensor $\underline{\mathbf{D}}$ with (3.7). However, these need only once in the algorithm, and thus, distributed or sequential computation can avoid this problem [KS08]. On the other hand, CPD needs this calculation in each update. Therefore, IDE problem must be solved or alleviated to achieve been proposed so far [PFS12, P⁺13, SRSK15, JPKF15, CV14, KPHF12]. This is a main concern on CPD or tensor factorization methods. The pseudo-code of CPD is shown in Algorithm 7.

3.2.3 Other Tensor Factorizations

Unlike matrix factorizations, there are many ways to decompose a tensor. Harshman proposed a method called DEcomposition into DIrectional COMponents (DEDICOM) which decompose a tensor into three matrices and two tensors with constraint that the only diagonal slice have values [Har78]. See

an illustration of DEDICOM in Figure 3.2(a). DEDICOM was designed to capture asynchronous relationships between two objects in the 1st mode and the 2nd mode. The 3rd mode is the time and 3-way DEDICOM captures the time-varying such relationships. Decomposing Tensor into Tensor (DTT) decomposes a tensor into a set of smaller tensors [FP14]. This enable to capture much more complicated relation between objects. See an illustration of DTT in Figure 3.2(b). PARAFAC2 decompose a tensor into two tensors and one matrix [BAK99,KTBB99]. This method can handle matrices with different size of row as a tensor and obtained a common coefficient matrix. This method can detect corresponding structures in different representations. For example, finding a sentences or words have the same meaning in a set of documents written in several different languages [CBKA07]. See an illustration of PARAFAC2 in Figure 3.2(c).

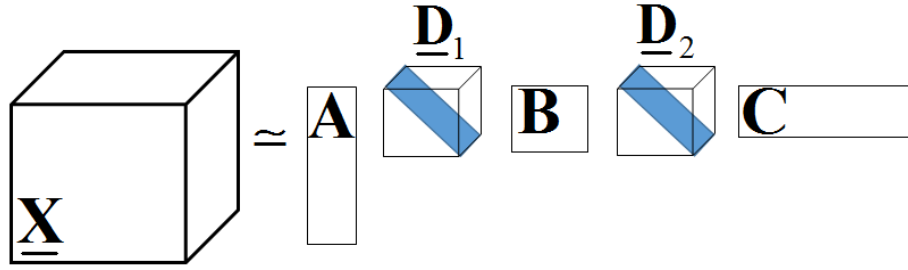
There are more various tensor factorization not introduced above such as Tensor Train Network [Ose11] and its extension Tensor Ring Decomposition [ZZX⁺16]. In this sense, the tensor factorization can be designed to specific tasks with user demands.

3.2.4 Nonnegativity on Tensor

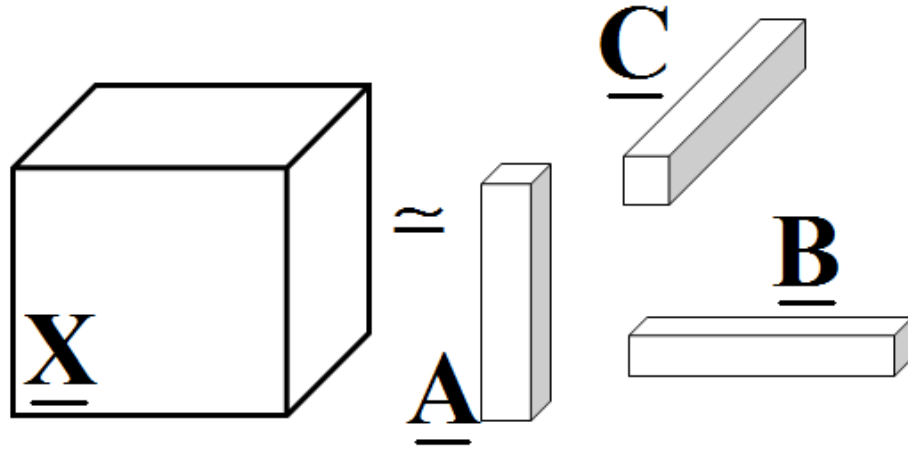
Nonnegative constraint also can be imposed in tensor factorization. Bro and De Jong imposed a nonnegativity to CPD [BDJ97]. Shashua *et al.* introduced NTF on machine learning community or data mining community with an image retrieval task [SH05]. The most NTF researches focused on adding nonnegativity constraint on CPD form tensor factorization. The reasons are twofolds. The one is the purpose of adding nonnegativity constraint is improving intepretability of the factorization results in general. In this sense, Tucker decomposition is not suitable for this purpose since it has dense core tensor $\underline{\mathbf{D}}$ need to be analyzed. The other one is the computational complexity of Nonnegative Tucker Decomposition (NTD) [KC07]. Since HOSVD or HOOI does not satisfy the nonnegativity because these are based on SVD which does not satisfy nonnegativity in general, NTD must solve following problems iteratively until three matrices and the core tensor converges:

$$\begin{aligned} \|\underline{\mathbf{X}} - \underline{\mathbf{I}} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C}\|_F^2 &= \|\underline{\mathbf{X}}_{(1)} - \mathbf{A}\underline{\mathbf{D}}_{(1)}(\mathbf{C} \otimes \mathbf{B})^T\|_F^2 \\ &= \|\underline{\mathbf{X}}_{(2)} - \mathbf{B}\underline{\mathbf{D}}_{(2)}(\mathbf{B} \otimes \mathbf{A})^T\|_F^2 \\ &= \|\underline{\mathbf{X}}_{(3)} - \mathbf{C}\underline{\mathbf{D}}_{(3)}(\mathbf{C} \otimes \mathbf{A})^T\|_F^2. \end{aligned}$$

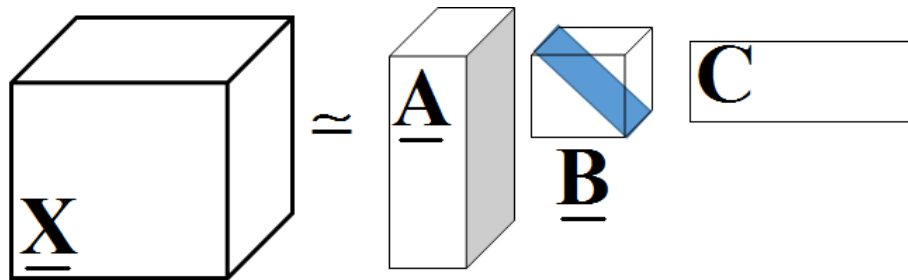
For the minimization of $\mathbf{A}$, the first matricized problem $\|\underline{\mathbf{X}}_{(1)} - \mathbf{A}\underline{\mathbf{D}}_{(1)}(\mathbf{C} \otimes \mathbf{B})^T\|_F^2$ is considered. As in NMF optimizations, we need to calculate the



(a) 3-Way DEDICOM



(b) DTT:Decomposing Tensor to Tensor



(c) PARAFAC2 Decomposition

Figure 3.2: Illustrations of various tensor factorization methods

gradient of matrix $\mathbf{A}$ and that is:

$$\frac{\partial \|\underline{\mathbf{X}} - \underline{\mathbf{I}} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C}\|_F^2}{\partial \mathbf{A}} = -2\underline{\mathbf{X}}_{(1)}(\mathbf{C} \otimes \mathbf{B})\underline{\mathbf{D}}_1^T + 2\mathbf{A}\underline{\mathbf{D}}_{(1)}(\mathbf{C} \otimes \mathbf{B})^T(\mathbf{C} \otimes \mathbf{B})\underline{\mathbf{D}}_{(1)}^T,$$

The calculation of Kronecker product $(\mathbf{C} \otimes \mathbf{B}) \in \mathbb{R}^{I_1 I_2 \times J_1 J_2}$ needs $O(I_1 I_2 J_1 J_2)$ for both space and time [KC07]. This is the same to the IDE problem in CPD. Furthermore, these calculation is larger than CPD. Kim *et al.* proposed the first update algorithm for NTD [KC07] and Phan *et al.* proposed an extend method of HALS for NTD although this calculation is unavoidable with these methods. In total, NTD can be used only to the small tensor and the application is limited [PC11, CPZ⁺12a, XY⁺11, MHA08].

Our main focus in this dissertation is Nonnegative CANDECOMP/PARAFAC Decomposition and I call this method NTF in order to discriminate from NTD. NTF is popular in machine learning community and used in image analysis [SH05, HPS05, CZPA09, BE08, YNRA12], time-varying network analysis [ADK09, BBB08, PFS12, CZS⁺07], music or sound analysis [BK10, FCC05], EEG data analysis [CAH09, CPC08, LKCC07, CPZ⁺12b, PTC12] and so on. Similar to NTD, NTF also needs to solve following problems:

$$\begin{aligned} \min_{\mathbf{A}, \mathbf{B}, \mathbf{C}} \|\underline{\mathbf{X}} - \underline{\mathbf{I}} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C}\|_F^2 &= \|\underline{\mathbf{X}}_{(1)} - \mathbf{A}(\mathbf{C} \odot \mathbf{B})^T\|_F^2 \\ &= \|\underline{\mathbf{X}}_{(2)} - \mathbf{B}(\mathbf{B} \odot \mathbf{A})^T\|_F^2 \\ &= \|\underline{\mathbf{X}}_{(3)} - \mathbf{C}(\mathbf{C} \odot \mathbf{A})^T\|_F^2, \end{aligned}$$

with nonnegative constraint on all tensors and matrices. Since NTF needs to minimize the above three problems with NMF algorithms and thus IDE problems appears in each minimization. This is also the problem in NTF, however, NTF is advantageous against NTD in this sense. The pseudo-code of NTF is shown in Algorithm 8.

In summary, from view point of efficiency of optimization, NMF has focused on the convergence speed of algorithms and tried to reduce the number of iterations. On the other hand, NTF has more difficult fundamental problem, that is IDE problem. In this dissertation, I focus on Orthogonal NMF, which considers an orthogonal condition on one-side matrix. As stated before, this is a popular setting since a close connection to clustering methods. We propose efficient algorithms for this problem which converges faster in Chapter 4. For NTF, I tackle IDE problem for the standard NTF problem and propose efficient algorithm in Chapter 5.

Algorithm 8 Nonnegative CANDECOMP/PARAFAC Decomposition (NTF)

Input: A Tensor $\underline{\mathbf{X}}$, the size of core tensor J .

Output: Nonnegative factor matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ such that $\underline{\mathbf{X}} \simeq \mathbf{I} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C}$.

Initialize $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ with positive random values.

repeat

 Solve an NMF problem $\|\underline{\mathbf{X}}_{(1)} - \mathbf{A}(\mathbf{C} \odot \mathbf{B})^T\|_F^2$ for $\mathbf{A}$

 Solve an NMF problem $\|\underline{\mathbf{X}}_{(2)} - \mathbf{B}(\mathbf{C} \odot \mathbf{A})^T\|_F^2$ for $\mathbf{B}$

 Solve an NMF problem $\|\underline{\mathbf{X}}_{(3)} - \mathbf{C}(\mathbf{B} \odot \mathbf{A})^T\|_F^2$ for $\mathbf{C}$

until Coverage criterion is satisfied

Chapter 4

Column-Wise Update Algorithms for Orthogonal Nonnegative Matrix Factorization

4.1 Introduction

Orthogonal Nonnegative Matrix Factorization (ONMF), firstly proposed by Ding *et al.* [DLPP06], factorizes a nonnegative matrix into two nonnegative matrices under a one-sided orthogonal constraint imposed on the first *factor* matrix. That is, ONMF is a minimization problem:

$$\begin{aligned} \min_{\mathbf{U}, \mathbf{V}} \|\mathbf{X} - \mathbf{UV}^T\|_F^2, \\ \text{subject to } \mathbf{U}, \mathbf{V} \geq 0, \mathbf{U}^T \mathbf{U} = \mathbf{I}, \end{aligned} \quad (4.1)$$

where $\mathbf{X} \in \mathbb{R}^{I_1 \times I_2}$, $\mathbf{U} \in \mathbb{R}^{I_1 \times J}$, $\mathbf{V} \in \mathbb{R}^{I_2 \times J}$ ($J \ll I_1, I_2$) and $\mathbf{I}$ is the identity matrix. In this formulation, $\mathbf{U}^T \mathbf{U} = \mathbf{I}$ is imposed as a condition, but the strict application of both nonnegativity and orthogonality is too strong. In fact, it yields a subset of orthonormal vectors in the standard basis. Therefore, in a practical sense, the optimization problem is stated as

$$\min_{\mathbf{U}, \mathbf{V}} \|\mathbf{X} - \mathbf{UV}^T\|_F^2 + \lambda \|\mathbf{U}^T \mathbf{U} - \mathbf{I}\| \quad (4.2)$$

with a positive coefficient λ . This corresponds to a Lagrangian formulation, as will be shown in the following section.

To the best of my knowledge, conventional algorithms for solving ONMF problems are all based on matrix-wise alternating block coordinate descent.

However, it is known that matrix-wise update algorithms require a relatively large number of iterations to converge. This is because those algorithms do not solve each conditional matrix-wise problem optimally [CAH09, KP11]. In NMF without the orthogonal constraint, some state-of-the-art algorithms update $\mathbf{U}$ and $\mathbf{V}$ column-wisely or element-wisely to gain faster convergence. In ONMF, however, it is difficult to incorporate the orthogonal constraint into column-wise or element-wise coordinate descent updates.

In this chapter, we propose a Fast Hierarchical Alternating Least Squares (HALS) algorithm for ONMF (HALS-ONMF). Our algorithm is based on a column-wise update algorithm for NMF proposed by Cichocki and Phan [CAH09]. To enable such a column-wise update even in ONMF, we derive a set of column-wise orthogonal constraints, taking into consideration both nonnegativity and orthogonality at the same time. Furthermore, we show that the column-wise orthogonal constraint can also be applied to column-wise update algorithms called scalar Block Coordinate Descent for solving Bregman divergence NMF (sBCD-NMF) [LLP12] where the Frobenius norm in (4.1) is replaced with more general Bregman divergence [LLP12]. This sBCD-ONMF algorithm is the first algorithm to solve ONMF with Bregman divergence.

4.2 Related Work

4.2.1 Convergence Speed of Nonnegative Matrix Factorization

As we introduced in Chapter 3, various iterative algorithms for NMF have been proposed so far [LS00, CZPA09, KP11, HD11]. Their interests is in the convergence speed and reducing the number of iterations. In general, their convergence speeds depend on their unit of updates, matrix-wise, vector-wise and element-wise. We briefly introduce them again and discuss applicability for Orthogonal NMF problems.

Matrix-wise Update Algorithms

Lee and Seung proposed a Multiplicative Update (MU) algorithm [LS00]. This MU algorithm is one of the efficient algorithms for NMF proposed in the early stage, and thus many extensions followed (*e.g.*, [CHHH11, CZPA09]). However, from the viewpoint of convergence, they were not sufficient [KHP14]. Lin *et al.* proposed a Project Gradient Descent (PGD) algorithm for NMF [Lin07]. This algorithm solves an NMF problem by solving Non-

negative Least Squares (NLS) problems for $\mathbf{U}$ and $\mathbf{V}$ alternatively and, gains faster convergence than MU algorithms. The difference in these algorithms is that the MU algorithm uses a fixed step size in the gradient descent, while PGD uses a flexible step size.

Vector-wise Update Algorithm

Cichocki and Phan proposed a Hierarchical Alternating Least Squares (HALS) algorithm [CAH09]. The HALS algorithm solves a set of column-wise NLS problems for each column and updates $\mathbf{U}$ and $\mathbf{V}$ column-wisely. Since column-wise NLS problems can be solved at a high accuracy and efficiency, HALS converges very fast. Kim *et al.* proposed an active-set like algorithm that also decomposes a matrix NLS problem into a set of column-wise sub-problems [KP11]. The difference between HALS and the active-set like method lies in the way to solve a column-wise sub-problem. The former uses the gradient to solve a sub-problem, while the latter uses an active-set method to solve a sub-problem. The active-set method consists of two stages: first, it finds a feasible point in standard NMF, as a nonnegative point, and then it solves a column-wise NLS problem while maintaining feasibility. Li *et al.* recently proposed scalar Block Coordinate Descent (sBCD) algorithm [LLP12]. The sBCD algorithm is applicable to not only NMF with Frobenius norm but also NMF with more general Bregman divergence. They used Taylor series expansion to derive the element-wise problem. Since the sBCD algorithm uses the column-wise residual in their update rule, its complexity is the same as that of column-wise update algorithms. Therefore, in this paper, we consider sBCD as a column-wise update algorithm (see Section 4.4.1). All of these vector-wise update algorithms can be regarded as state-of-the-art algorithms, because they converge empirically faster than matrix-wise update algorithms. However, addition of matrix-based constraints such as $\mathbf{U}^T \mathbf{U} = \mathbf{I}$ is still challenging in such column-wise updates.

Element-wise Update Algorithms

Hsieh *et al.* proposed an element-wise update algorithm called a Greedy Coordinate Descent (GCD) algorithm. To the authors' knowledge, it is the fastest algorithm for NMF [HD11]. The GCD algorithm takes a greedy strategy to decrease the value of the objective function. It selects and updates the most contributable variables for minimization. The low computational cost of GCD is due to the fact that it does not update unnecessary elements. Unfortunately, the GCD algorithm cannot work with such a constraint that affects all elements of one column at the same time. One example of such

a constraint is the graph regularized constraint that appears when we minimize $\alpha(\text{tr}(\mathbf{U}^T \mathbf{L} \mathbf{U}))$, where $\mathbf{L}$ is a graph Laplacian matrix of $\mathbf{X}^T \mathbf{X}$. The GCD relies on the fact that, with a fixed $\mathbf{V}$, updating of an element U_{i_1j} of $\mathbf{U}$ changes only the gradients of elements in the same row $\mathbf{U}_{i_1}$ because the gradient in $\mathbf{U}$ is given by $(-2\mathbf{X}\mathbf{V} + 2\mathbf{U}\mathbf{V}^T\mathbf{V})$. In more detail, GCD iteratively selects and updates the most contributable variable U_{i_1j} in the i_1 th row. Unfortunately, the GCD is not applicable to ONMF because the orthogonal condition requires an interaction between different rows.

4.2.2 Orthogonal NMF

An additional orthogonal constraint, $\mathbf{U}^T \mathbf{U} = \mathbf{I}$, is imposed in ONMF. At first, we briefly review the first ONMF algorithm proposed by Ding *et al.* [DLPP06] and reveal the problem behind ONMF.

The goal of ONMF is to find a nonnegative orthogonal matrix $\mathbf{U}$ and a nonnegative matrix $\mathbf{V}$ minimizing the following objective function with a Lagrangian multiplier λ ,

$$L(\mathbf{U}, \mathbf{V}) = \|\mathbf{X} - \mathbf{U}\mathbf{V}^T\|_F^2 + \text{Tr}[\lambda(\mathbf{U}^T \mathbf{U} - \mathbf{I})]. \quad (4.3)$$

The KKT complementary condition gives¹

$$(-2\mathbf{X}\mathbf{V} + 2\mathbf{U}\mathbf{V}^T\mathbf{V} + 2\mathbf{U}\lambda)_{i_1j} \mathbf{U}_{i_1j}^2 = 0, \quad i_1 = 1, 2, \dots, I_1, \quad j = 1, 2, \dots, J. \quad (4.4)$$

Then the update rule of the constrained matrix $\mathbf{F}$ is derived as

$$\mathbf{U}_{i_1j} \leftarrow \mathbf{U}_{i_1j} * \sqrt{\frac{(\mathbf{X}\mathbf{V})_{i_1j}}{[\mathbf{U}(\mathbf{V}^T\mathbf{V} + \lambda)]_{i_1j}}}. \quad (4.5)$$

The point is how to determine the value of the Lagrange multiplier λ . Since it is not easy to solve this problem for every value of λ , [DLPP06] ignored the nonnegativity and relied only on $\mathbf{U}^T \mathbf{U} = \mathbf{I}$ to have approximate values of off-diagonal elements. By multiplying $\mathbf{U}^T$ from the left in (4.4), we have

$$\lambda = \mathbf{U}^T \mathbf{X}\mathbf{V} - \mathbf{V}^T \mathbf{V}.$$

Thus, inserting this λ into (4.5), we have the final update form as

$$\mathbf{U}_{i_1j} \leftarrow \mathbf{U}_{i_1j} * \sqrt{\frac{(\mathbf{X}\mathbf{V})_{i_1j}}{(\mathbf{U}\mathbf{U}^T \mathbf{X}\mathbf{V})_{i_1j}}}.$$

¹Hereafter, we will not state the nonnegative constraint explicitly.

Note that their formulation with the specific values of λ do not strictly satisfy the orthogonality. Nevertheless, this specification is useful in avoiding the *zero-lock* problem appearing both in ONMF and in NMF: Once an element becomes zero in the middle of iterations, the element will not be recasted in the following steps (see the multiplicative update rule (4.5)). Besides, when the orthogonality is strictly posed with nonnegativity, each row vector of $\mathbf{U}$ must have only one non-zero value. That is, any algorithm using a multiplicative update rule falls easily into a hole of the zero-lock problem. Therefore, ONMF algorithms put the first priority on the approximation while loosening the degree of orthogonality.

As a result of compromise, an ONMF algorithm can be seen as an algorithm that balances the trade-off between orthogonality and approximation with a weighting parameter λ as seen in (4.2). We do not categorize ONMF algorithms by the unit of updates because all conventional ONMF algorithms are based only on matrix-wise updates. Rather, those algorithms should be categorized according to whether the algorithm employs a weighting parameter or not. If an algorithm minimizes an objective function with a weighting parameter and if the value is not appropriately chosen, then the algorithm would fail in either acceptable degree of approximation or orthogonality. Such a failure has often been reported in past experimental results [LWP10, Mir14, PGAG12].

Algorithms without weighting parameters

Ding *et al.* [DLPP06] proposed the first ONMF algorithm based on the MU algorithm [LS00]. This algorithm has no weighting parameter. It solves approximately the Lagrangian (4.3) as I reviewed. Yoo *et al.* also proposed an MU-based algorithm. They used the gradient on the Stiefel manifold that is the set of all orthogonal matrices [YC08]. The gradient on the Stiefel manifold is compatible with that of the MU algorithm because the manifold constrains every matrix to be orthogonal and the employed MU algorithm guarantees nonnegative values.²

²In general, the resultant constrained matrix by ONMF in [YC08] also does not satisfy strict orthogonality because the MU algorithm is gradient descent with a fixed step size, and thus, it may undershoot or overshoot.

Algorithms with weighting parameters

Mirzal proposed a convergent algorithm that is also based on the MU algorithm in practice. He proposed two algorithms in [Mir14], one of which is the same as the one by Li *et al.* [LWP10]. The first algorithm introduced a weighting parameter α instead of the Lagrangian multiplier λ in (4.3). The second algorithm was a convergent algorithm. The convergence of the algorithm is proved, but the computational cost is high. In this algorithm, the zero-lock problem was forcibly avoided by replacing zero values with a small positive value ϵ . There are algorithms that put the first priority on nonnegativity rather than orthogonality. Pompili *et al.* tackled directly the zero-lock problem [PGAG12]. They employed the Augmented Lagrangian method. In more detail, they used the gradient on the Stiefel manifold and explicitly introduced a Lagrangian multiplier ψ for nonnegativity. The initial value of the Lagrangian was approximated to a smaller value in order to avoid the zero-lock problem. They increase the value of ψ gradually to strengthen the nonnegativity while the iteration is repeated. As a result, the nonnegativity was not strictly guaranteed in the algorithm. In addition, it has three parameters to be set appropriately for orthogonality, nonnegativity and step size.

There are mainly two problems to be solved in order to develop fast ONMF algorithms. First, we have to incorporate the matrix-type orthogonal condition $\mathbf{U}^T \mathbf{U} = \mathbf{I}$ into column-wise or element-wise updating NMF algorithms. This is necessary to obtain efficiency. Next, we need to solve the zero-lock problem. This is necessary to find an appropriate balance between orthogonality and nonnegativity without a weighting parameter. This problem prevents us from using the Lagrangian and alternatively forces us to take a balance between orthogonality and nonnegativity appropriately. In this paper, I show a way to realize two things in ONMF algorithms.

4.3 Hierarchical Alternating Least Squares Algorithm for ONMF

In this section, I show a way of utilizing the HALS for ONMF. First, I briefly review the HALS for standard NMF and then describe how to incorporate the orthogonal constraint column-wisely to propose HALS-ONMF.

CHAPTER 4. COLUMN-WISE UPDATE ALGORITHMS FOR ORTHOGONAL NONNEGATIVE MATRIX FACTORIZATION

Table 4.1: A summary of categorization of ONMF algorithms. Frobenius, KL and Bregman denotes distortions used for measuring the degree of approximation.

Author(Year)	Updates		Weighting (YES/NO)	Distortion measure
	Matrix	Vector		
Ding <i>et al.</i> (2006) [DLPP06]	MU		NO	Frobenius
Yoo <i>et al.</i> (2008) [YC08]	MU		NO	Frobenius
Li <i>et al.</i> (2010) [LWP10]	MU		YES	Frobenius/KL
Pompili <i>et al.</i> (2012) [PGAG12]	PGD		YES	Frobenius
Mirzal (2014) [Mir14]	MU		YES	Frobenius
Proposed (HALS-ONMF) [KTK14]		HALS	NO	Frobenius
Proposed (sBCD-ONMF) [KKT16]		sBCD	NO	Bregman

4.3.1 Column-wise Orthogonal Constraint

Since a column vector affects the other column vectors in $\mathbf{U}^T\mathbf{U}$, the orthogonal constraint cannot be directly introduced into the HALS algorithm introduces. In this dissertation, I exploit a simple fact that if the sum of nonnegative values is zero, then all of the values are zero. Since the orthogonal condition $\mathbf{U}^T\mathbf{U} = \mathbf{I}$ means $\mathbf{u}_k^T\mathbf{u}_j = 0$ for every $k \neq j$, we can use a single condition $\sum_{k \neq j} \mathbf{u}_k^T\mathbf{u}_j = 0$ for fixed j coupled with $\mathbf{u}_k^T\mathbf{u}_j \geq 0$, instead of $J - 1$ conditions $\mathbf{u}_k^T\mathbf{u}_j = 0$ for every $k (\neq j)$. That is, one matrix condition $\mathbf{U}^T\mathbf{U} = \mathbf{I}$ is equivalently replaced with $2J$ column-wise constraints of $\mathbf{u}_j^T\mathbf{u}_j = 1$ and $\sum_{k \neq j} \mathbf{u}_k^T\mathbf{u}_j = 0$ for every j . As will be shown, the newly derived column-wise constraints can be updated with $O(I_1)$ for each column (I_1 being the number of rows of $\mathbf{X}$ to be factorized).

Now it suffices to impose the conditions

$$\mathbf{U}^{(j)T}\mathbf{u}_j \triangleq \sum_{k \neq j} \mathbf{u}_k^T\mathbf{u}_j = 0, \quad j = 1, 2, \dots, J. \quad (4.6)$$

In addition, I normalize each column vector so as to $\|\mathbf{u}_j\|^2 = \mathbf{u}_j^T\mathbf{u}_j = 1$ to satisfy $\mathbf{U}^T\mathbf{U} = \mathbf{I}$. Thus, I introduce constraint $\mathbf{U}^{(j)T}\mathbf{u}_j = 0$ ($j = 1, 2, \dots, J$) into (4.3) as the column-wise orthogonal constraint. The nonnegativity of the elements is preserved with the ϵ -truncate function $[\]_+$.

4.3.2 HALS-ONMF

With the derived column-wise constraint (4.6), the localized objective function is formulated as a Lagrangian:

$$\begin{aligned} L(\mathbf{u}_j, \mathbf{v}_j, \lambda_j) &= \|\mathbf{X}^{(j)} - \mathbf{u}_j \mathbf{v}_j^T\|_F^2 + \lambda_j (\mathbf{U}^{(j)T} \mathbf{u}_j), \quad \text{where} \\ \mathbf{X}^{(j)} &= \mathbf{X} - \sum_{k \neq j} \mathbf{u}_k \mathbf{v}_k^T, \\ \mathbf{U}^{(j)} &= \sum_{k \neq j} \mathbf{u}_k, \quad \lambda_j \geq 0. \end{aligned}$$

The gradient is given as

$$\frac{\partial L}{\partial \mathbf{u}_j} = -2\mathbf{X}^{(j)} \mathbf{v}_j + 2\mathbf{u}_j \mathbf{v}_j^T \mathbf{v}_j + \lambda_j \mathbf{U}^{(j)}. \quad (4.7)$$

By solving $\partial L / \partial \mathbf{u}_j = \mathbf{0}$ and forcibly keeping the nonnegativity, we obtain the update rule, under the assumption of normalization of $\mathbf{u}_j^T \mathbf{u}_j = 1$, as post-processing:

$$\mathbf{u}_j \leftarrow \left[\mathbf{X}^{(j)} \mathbf{v}_j - \frac{\lambda_j}{2} \mathbf{U}^{(j)} \right]_+. \quad (4.8)$$

Unfortunately, the setting of the value of λ still remains as a problem. In this study, I take the same way as Ding *et al.* did in [DLPP06]. By multiplying $\mathbf{U}^{(j)}$ from the left in (4.7) and using $\mathbf{U}^{(j)T} \mathbf{u}_j = \mathbf{0}$, we obtain

$$\lambda_j = \frac{2\mathbf{U}^{(j)T} \mathbf{X}^{(j)} \mathbf{v}_j}{\mathbf{U}^{(j)T} \mathbf{U}^{(j)}}.$$

Hence, (4.8) becomes

$$\mathbf{u}_j \leftarrow \left[\mathbf{X}^{(j)} \mathbf{v}_j - \frac{\mathbf{U}^{(j)T} \mathbf{X}^{(j)} \mathbf{v}_j}{\mathbf{U}^{(j)T} \mathbf{U}^{(j)}} \mathbf{U}^{(j)} \right]_+. \quad (4.9)$$

Since the orthogonal constraint $\mathbf{U}^{(j)T} \mathbf{u}_j = 0$ does not affect $\mathbf{v}_j$, we can use the same update rule as HALS-NMF,

$$\begin{aligned} \mathbf{u}_j &\leftarrow \left[\mathbf{X}^{(j)} \mathbf{v}_j - \frac{\mathbf{U}^{(j)T} \mathbf{X}^{(j)} \mathbf{v}_j}{\mathbf{U}^{(j)T} \mathbf{U}^{(j)}} \mathbf{U}^{(j)} \right]_+, \quad \text{and} \\ \mathbf{v}_j &\leftarrow \left[\mathbf{X}^{(j)T} \mathbf{u}_j \right]_+. \end{aligned}$$

Using $\mathbf{X}^{(j)} = \mathbf{X} - \sum_{k \neq j} \mathbf{u}_k \mathbf{v}_k^T = \mathbf{X} - \mathbf{U}\mathbf{V}^T + \mathbf{u}_j \mathbf{v}_j^T$, we have the final form of updating rules:

$$\begin{aligned} \mathbf{u}_j &\leftarrow \left[\mathbf{h} - \frac{\mathbf{U}^{(j)T} \mathbf{h}}{\mathbf{U}^{(j)T} \mathbf{U}^{(j)}} \mathbf{U}^{(j)} \right]_+, \\ \mathbf{u}_j &\leftarrow \mathbf{u}_j / \|\mathbf{u}_j\|_2, \text{ and} \\ \mathbf{v}_j &\leftarrow [(\mathbf{X}^T \mathbf{U})_j - \mathbf{V}(\mathbf{U}^T \mathbf{U})_j + \mathbf{v}_j \mathbf{u}_j^T \mathbf{u}_j]_+, \text{ where} \\ \mathbf{h} &= (\mathbf{X}\mathbf{V})_j - \mathbf{U}(\mathbf{V}^T \mathbf{V})_j + \mathbf{u}_j \mathbf{v}_j^T \mathbf{v}_j. \end{aligned}$$

The zero-lock problem is resolved by $[\]_+$ operation as it is in [Mir14]. The proposed HALS-ONMF algorithm is shown in Algorithm 9.

This vector-wise update algorithm is faster than conventional matrix-wise update algorithms for the following reason. The matrix-wise update rule (4.5) is derived from (4.4), while the vector-wise update rule (4.9) of the proposed HALS-ONMF is derived from (4.7). The former comes from the KKT complementary condition which is just a necessary condition for the solution to minimize (4.3). Therefore, there is no guarantee for the updating to be optimum in each iteration. While, in the latter, the corresponding optimization problem can be solved analytically in a closed form. Therefore, the updating is always optimal in each iteration.

4.4 ONMF with Bregman divergence

In this section, I consider a wider class of ONMF problems; that is, Bregman divergence is introduced instead of the Frobenius norm to measure the degree of approximation. In the case of NMF, Li *et al.* [LLP12] already proposed a column-wise update algorithm called scalar Block Coordinate Descent (sBCD) to solve Bregman divergence NMF. In this paper, I develop Bregman divergence ONMF, by incorporating the column-wise orthogonal constraint into their sBCD algorithm. We first briefly review the sBCD-algorithm [LLP12] and then explain how my column-wise orthogonal constraint can be incorporated in sBCD.

4.4.1 Scalar Block Coordinate Descent Algorithm (sBCD)

The objective function is now given as

$$\begin{aligned} \min_{\mathbf{U}, \mathbf{V}} D_\phi(\mathbf{X} \| \mathbf{U}\mathbf{V}^T), \\ \text{subject to } \mathbf{U} \geq 0, \mathbf{V} \geq 0, \end{aligned} \tag{4.10}$$

Algorithm 9 Fast HALS-Orthogonal NMF

Input: Nonnegative matrix $\mathbf{X}$, Number of components J

Output: Decomposing nonnegative matrices $\mathbf{U}$ and $\mathbf{V}$ such that $\mathbf{X} \simeq \mathbf{U}\mathbf{V}^T$ and $\mathbf{U}^T\mathbf{U} \cong \mathbf{I}$.

Initialize $\mathbf{U}$ and $\mathbf{V}$ arbitrary.

$\mathbf{F} = \mathbf{U}\mathbf{1}_J$

repeat

$\mathbf{A} = \mathbf{X}\mathbf{V}$

$\mathbf{B} = \mathbf{V}^T\mathbf{V}$

for $j = 1$ **to** J **do**

$\mathbf{U}^{(j)} = \mathbf{F} - \mathbf{u}_j$

$\mathbf{h} = \mathbf{A}_j - \mathbf{U}\mathbf{B}_j + \mathbf{B}_{jj}\mathbf{u}_j$

$\mathbf{f}_j = [\mathbf{h} - \frac{\mathbf{U}^{(j)}\mathbf{h}}{\mathbf{U}^{(j)T}\mathbf{U}^{(j)}}\mathbf{U}^{(j)}]_+$

$\mathbf{f}_j = \mathbf{u}_j / \|\mathbf{u}_j\|^2$

$\mathbf{F} = \mathbf{U}^{(j)} + \mathbf{u}_j$

end for

$\mathbf{C} = \mathbf{X}^T\mathbf{U}$

$\mathbf{D} = \mathbf{F}^T\mathbf{U}$

for $j = 1$ **to** J **do**

$\mathbf{v}_j \leftarrow [\mathbf{C}_j - \mathbf{V}\mathbf{D}_j + \mathbf{D}_{jj}\mathbf{v}_j]_+$

end for

until Convergence criterion is satisfied.

CHAPTER 4. COLUMN-WISE UPDATE ALGORITHMS FOR
ORTHOGONAL NONNEGATIVE MATRIX FACTORIZATION

Table 4.2: Examples of Bregman divergence

Description	Function $\phi(a)$	$D_\phi(a b)$
Frobenius norm	$\frac{a^2}{2}$	$(a - b)^2/2$
KL-divergence	$a \log a$	$a \log \frac{a}{b} - a + b$
IS-divergence	$-\log a$	$\frac{a}{b} - \log \frac{a}{b}$
β -divergence	$\frac{a^{\beta+1} - (\beta+1)a + \beta}{\beta(\beta+1)}$	$\frac{1}{\beta(\beta+1)}(a^{\beta+1} - b^{\beta+1} - (\beta+1)b^\beta(a - b))$

where $D_\phi(\mathbf{A}||\mathbf{B})$ is a Bregman divergence between matrices $\mathbf{A}$ and $\mathbf{B}$ using a strictly convex function ϕ . The definition of Bregman divergence is as follows.

Definition 7 (Bregman Divergence). *Let $\phi : S \subseteq \mathbb{R} \rightarrow \mathbb{R}$ be a strictly convex function with the continuous first derivation $\nabla\phi$. Then, Bregman divergence corresponding to ϕ , $D_\phi : S \times \text{int}(S) \rightarrow \mathbb{R}_+$, is defined as $D_\phi(x, y) = \phi(x) - \phi(y) - \nabla\phi(y)(x - y)$. Here $\text{int}(S)$ is the interior of S .*

A Bregman divergence for scalars is extended to the one for matrices by $D_\phi(\mathbf{A}||\mathbf{B}) = \sum_{i_1, i_2} D_\phi(\mathbf{A}_{i_1 i_2}||\mathbf{B}_{i_1 i_2})$. Bregman divergences include many well-known divergences such as Frobenius norm and KL-divergence (Table 4.2). Recently, Li *et al.* proposed a column-wise update algorithm for Bregman divergence NMF.³ Their key idea of the update rules is Taylor series of Bregman divergences.

Let $E_t(a||b) \triangleq |a - b|^t$ and $E_t(\mathbf{X}||\mathbf{X}') \triangleq \sum_{i_1, i_2} |x_{i_1 i_2} - x'_{i_1 i_2}|^t$ be the t th power of t -norm distance. Then, for $\mathbf{X}' = \mathbf{U}\mathbf{V}^T$, we have

$$\min_{\mathbf{X}'} E_t(\mathbf{X}||\mathbf{X}') = \min_{\mathbf{U}\mathbf{V}} E_t(\mathbf{X}||\mathbf{U}\mathbf{V}^T) = \min_{\forall j \mathbf{u}_j, \mathbf{v}_j} E_t(\mathbf{X}^{(j)}||\mathbf{u}_j \mathbf{v}_j^T).$$

We want to connect $E_t(\mathbf{X}^{(j)}||\mathbf{u}_j \mathbf{v}_j^T)$ with Bregman divergence $D_\phi(\mathbf{X}||\mathbf{U}\mathbf{V}^T)$ to minimize $E_t(\mathbf{X}^{(j)}||\mathbf{u}_j \mathbf{v}_j^T)$. In the scalar case, by applying the Taylor series

³They proposed the scalar Block Coordinate Descent algorithm as an element-wise update algorithm. However, their update rules need to re-calculate the residual column-wisely. Therefore, I consider their algorithm as a column-wise update algorithm.

of $\phi(x)$ at $x = b$ to $\phi(a)$, we have

$$\begin{aligned}
D_\phi(a||b) &= \phi(a) - \phi(b) - \nabla\phi(b)(a - b) \\
&= \nabla\phi(b)(a - b) + \sum_{t=2}^{\infty} \frac{\nabla^t\phi(b)}{t!}(a - b)^t - \nabla\phi(b)(a - b) \\
&\quad \text{(Taylor series of the first two terms)} \\
&= \sum_{t=2}^{\infty} \frac{\nabla^t\phi(b)}{t!}(a - b)^t \\
&= \sum_{t=2}^{\infty} \frac{\nabla^t\phi(b)}{t!}(-\text{sgn}(b - a))^t E_t(a||b), \tag{4.11}
\end{aligned}$$

where $\nabla^t\phi(b)$ is the t -order derivative of $\phi(x)$ at $x = b$. The last equation comes from the relation: $(a - b)^t = (\text{sgn}(a - b))^t |a - b|^t$. Hence, as a natural extension, $D_\phi(\mathbf{X}||\mathbf{UV}^T)$ can be re-written as

$$D_\phi(\mathbf{X}||\mathbf{UV}^T) = \sum_{i_1 i_2} \sum_{t=2}^{\infty} \frac{\nabla^t\phi(x'_{i_1 i_2})}{t!} (-\text{sgn}(x'_{i_1 i_2} - x_{i_1 i_2}))^t E_t(x_{i_1 i_2}^{(j)} || u_{i_1 j} v_{i_2 j}),$$

where $x_{i_1 i_2}^{(j)} = (\mathbf{X} - \sum_{k \neq j} \mathbf{u}_k \mathbf{v}_k^T)_{i_1 i_2}$ and $x'_{i_1 i_2} = (\mathbf{UV}^T)_{i_1 i_2}$. Thus, we can use the partial derivation of $E_t(x_{i_1 i_2}^{(j)} || u_{i_1 j} v_{i_2 j})$ instead of that of $D_\phi(\mathbf{X}||\mathbf{UV}^T)$. Since

$$\begin{aligned}
&\frac{\partial}{\partial f_{i_1 j}} \left(\frac{\nabla^t\phi(x'_{i_1 i_2})}{t!} (-\text{sgn}(u_{i_1 j} v_{i_2 j} - x_{i_1 i_2}^{(j)}))^t E_t(x_{i_1 i_2}^{(j)} || u_{i_1 j} v_{i_2 j}) \right) \\
&= -v_{i_2 j} \frac{\nabla^t\phi(x'_{i_1 i_2})}{t-1!} (x_{i_1 i_2}^{(j)} - u_{i_1 j} v_{i_2 j})^{t-1} + v_{i_2 j} \frac{\nabla^{t+1}\phi(x'_{i_1 i_2})}{t!} (x_{i_1 i_2}^{(j)} - u_{i_1 j} v_{i_2 j})^t,
\end{aligned}$$

with (4.11), we have

$$\begin{aligned}
\frac{\partial D_\phi(x_{i_1 i_2} || x'_{i_1 i_2})}{f_{i_1 j}} &= v_{i_2 j} \nabla^2\phi(x'_{i_1 i_2})(u_{i_1 j} v_{i_2 j} - x_{i_1 i_2}^{(j)}) \\
&\quad + \sum_{t=2}^{\infty} \left(-v_{i_1 j} \frac{\nabla^{t+1}\phi(x'_{i_1 i_2})}{t!} (x_{i_1 i_2}^{(j)} - u_{i_1 j} v_{i_2 j})^t \right. \\
&\quad \left. + v_{i_2 j} \frac{\nabla^t\phi(x'_{i_1 i_2})}{t!} (x_{i_1 i_2}^{(j)} - u_{i_1 j} v_{i_2 j})^t \right) \\
&= v_{i_2 j} \nabla^2\phi(x'_{i_1 i_2})(u_{i_1 j} v_{i_2 j} - x_{i_1 i_2}^{(j)}).
\end{aligned}$$

Taking the sum over the rows and columns, we obtain the gradient of $D_\phi(\mathbf{X}||\mathbf{UV}^T)$ in $u_{i_1 j}$:

$$\frac{\partial D_\phi(\mathbf{X}||\mathbf{UV}^T)}{\partial u_{i_1 j}} = \sum_{i_1=1}^{I_1} v_{i_2 j} \nabla^2\phi(x'_{i_1 i_2})(u_{i_1 j} v_{i_2 j} - x_{i_1 i_2}^{(j)}). \tag{4.12}$$

Finally, the update rule of sBCD is given by

$$u_{i_1 j} \leftarrow \left[\frac{\sum_{i_2=1}^{I_2} \nabla^2 \phi(x'_{i_1 i_2}) x_{i_1 i_2}^{(j)} v_{i_2 j}}{\sum_{i_2=1}^{I_2} \nabla^2 \phi(x'_{i_1 i_2}) v_{i_2 j} v_{i_2 j}} \right]_+. \quad (4.13)$$

This sBCD algorithm (4.13) needs to calculate column-wise residual $\mathbf{X}^{(j)} = \mathbf{X} - \sum_{k \neq j} \mathbf{u}_k \mathbf{v}_k^T$ for $x_{i_1 i_2}^{(j)}$ in (4.13). Therefore, instead of the element-wise update (4.13), I adopt the following:

$$\mathbf{u}_j \leftarrow \left[\frac{(\nabla^2 \phi(\mathbf{U}\mathbf{V}^T) * \mathbf{X}^{(j)}) \mathbf{v}_j}{\nabla^2 \phi(\mathbf{U}\mathbf{V}^T) \mathbf{v}_j^2} \right]_+. \quad (4.14)$$

4.4.2 Bregman Divergence ONMF

Now, I introduce the orthogonal constraint into Bregman divergence NMF to have Bregman divergence ONMF. The minimization problem of Bregman divergence ONMF is given by

$$\begin{aligned} \min_{\mathbf{U}, \mathbf{V}} D_\phi(\mathbf{X} \| \mathbf{U}\mathbf{V}^T) \\ \text{subject to } \mathbf{U} \geq 0, \mathbf{V} \geq 0, \mathbf{U}^T \mathbf{U} = \mathbf{I}. \end{aligned} \quad (4.15)$$

For the same reason as that stated before, I solve its relaxed version:

$$\begin{aligned} \min_{\mathbf{U}, \mathbf{V}} D_\phi(\mathbf{X} \| \mathbf{U}\mathbf{V}^T) + \lambda \|\mathbf{U}^T \mathbf{U} - \mathbf{I}\| \\ \text{subject to } \mathbf{U} \geq 0, \mathbf{V} \geq 0. \end{aligned}$$

This problem can be re-written equivalently and column-wisely as

$$\begin{aligned} O = \min_{\mathbf{u}_j, \mathbf{v}_j, \lambda_j} D_\phi(\mathbf{X} \| (\sum_{k \neq j} \mathbf{u}_k \mathbf{v}_k^T) + \mathbf{u}_j \mathbf{v}_j^T) + \lambda \mathbf{U}^{(j)T} \mathbf{u}_j, \\ \text{where } \mathbf{U}^{(j)} = \sum_{k \neq j} \mathbf{u}_k, \lambda_j \geq 0. \end{aligned} \quad (4.16)$$

Note that the first term of RHS of (4.16) is equivalent to (4.15). Hence, we have

$$\frac{\partial O}{\partial \mathbf{u}_j} = (\nabla^2 \phi(\mathbf{X}'_{i_1 i_2}) \odot (\mathbf{u}_j \mathbf{v}_j^T - \mathbf{X}^{(j)})) \mathbf{v}_j + \lambda_j \mathbf{U}^{(j)}. \quad (4.17)$$

To determine the value of the Lagrangian multiplier λ_j , I again assume $\mathbf{U}^{(j)T} \mathbf{u}_j = 0$ and multiply $\mathbf{U}^{(j)T}$ from the left to (4.17) to be zero. This gives

$$\lambda_j \mathbf{U}^{(j)T} \mathbf{U}^{(j)} = -\mathbf{U}^{(j)T} (\mathbf{u}_j \mathbf{v}_j^T \odot \nabla^2 \phi(\mathbf{U}\mathbf{V}^T)) \mathbf{v}_j + \mathbf{U}^{(j)T} (\mathbf{X}^{(j)} \odot \nabla^2 \phi(\mathbf{U}\mathbf{V}^T)) \mathbf{v}_j.$$

CHAPTER 4. COLUMN-WISE UPDATE ALGORITHMS FOR
ORTHOGONAL NONNEGATIVE MATRIX FACTORIZATION

Under nonnegativity $\mathbf{u}_j \geq 0$ and $\mathbf{U}^{(j)} \geq 0$, with the assumption $\mathbf{U}^{(j)T} \mathbf{u}_j = 0$, we have

$$\mathbf{u}_j, \mathbf{U}^{(j)} \geq 0 \quad \text{and} \quad \mathbf{U}^{(j)T} \mathbf{u}_j = 0 \Rightarrow \mathbf{U}^{(j)T} (\mathbf{u}_j \mathbf{v}_j^T \odot \nabla^2 \phi(\mathbf{U} \mathbf{V}^T)) \mathbf{v}_j = 0.$$

This is because the row indices of zero values of $(\mathbf{u}_j \mathbf{v}_j^T \odot \nabla^2 \phi(\mathbf{U} \mathbf{V}^T)) \mathbf{v}_j$ are the same as those of $\mathbf{u}_j$. Hence, I may set λ_j to

$$\lambda_j = \frac{\mathbf{U}^{(j)T} (\mathbf{X}^{(j)} \odot \nabla^2 \phi(\mathbf{U} \mathbf{V}^T)) \mathbf{v}_j}{\mathbf{U}^{(j)T} \mathbf{U}^{(j)}}.$$

Then the update rule of sBCD-ONMF becomes

$$\mathbf{u}_j \leftarrow \left[\frac{(\nabla^2 \phi(\mathbf{U} \mathbf{V}^T) * \mathbf{X}^{(j)}) \mathbf{v}_j - \frac{\mathbf{U}^{(j)T} (\mathbf{X}^{(j)} \odot \nabla^2 \phi(\mathbf{U} \mathbf{V}^T)) \mathbf{v}_j \mathbf{U}^{(j)}}{\mathbf{U}^{(j)T} \mathbf{U}^{(j)}}}{\nabla^2 \phi(\mathbf{U} \mathbf{V}^T) \mathbf{v}_j^2} \right]_+. \quad (4.18)$$

If I use $\phi(x) = x^2/2$ corresponding to the Frobenius norm, it is easy to verify $\nabla^2 \phi(\mathbf{U} \mathbf{V}^T) = \mathbf{1}$. It implies that (4.18) is equivalent to (4.9) with post-processing normalization $\|\mathbf{u}_j\|_2^2 = 1$.

This sBCD-ONMF algorithm is an extension of the previous HALS-ONMF algorithm, but its convergence is slower because sBCD-ONMF needs to update the column-wise residual in addition to the updating of each column, while HALS-ONMF does not need to do so for the residual. The proposed sBCD-ONMF algorithm is shown in Algorithm 10.

4.4.3 Relation to Bregman Hard Clustering

The original ONMF is known to be related to k -means clustering [DLPP06]. So, in this section, I make clear the relationship between Bregman divergence ONMF and Bregman Hard Clustering proposed by Banerjee *et al.* [BMDG05]. The criterion of Bregman hard clustering to minimize is a natural extension of that of k -means clustering as shown below:

$$\min_{\pi_j=1,2,\dots,J} \sum_{j=1}^J \sum_{n \in \pi_j} D_\phi(\mathbf{x}_n || \boldsymbol{\mu}_j), \quad (4.19)$$

where π_j for $j = 1, 2, \dots, J$ is a set of disjoint clusters and $\boldsymbol{\mu}_j = \sum_{n \in \pi_j} \frac{1}{|\pi_j|} \mathbf{x}_n$ is the centroid of cluster π_j . Then, we have the following theorem.

Theorem 1 (Equivalence between Bregman divergence ONMF and Bregman hard clustering). *The minimization problem of Bregman divergence ONMF (4.15) is equivalent to that of the Bregman hard clustering defined in (4.19).*

Algorithm 10 sBCD-Orthogonal NMF (a generalized HALS-ONMF)

- 1: **Input:** Nonnegative matrix $\mathbf{X}$, Number of components J and function ϕ for a Bregman divergence
 - 2: **Output:** Decomposing nonnegative matrices $\mathbf{U}$ and $\mathbf{V}$ such that $\mathbf{X} \simeq \mathbf{UV}^T$ and $\mathbf{U}^T\mathbf{U} \simeq \mathbf{I}$.
 - 3:
 - 4: Initialize $\mathbf{U}$ and $\mathbf{V}$ arbitrary.
 - 5: $\mathbf{R} = \mathbf{U}\mathbf{1}_J$
 - 6: $\mathbf{X}' = \mathbf{UV}^T$
 - 7: $\mathbf{E} = \mathbf{X} - \mathbf{X}'$
 - 8: **repeat**
 - 9: $\mathbf{B} = \nabla^2\phi(\mathbf{X}')$
 - 10: **for** $j = 1$ **to** J **do**
 - 11: $\mathbf{X}^{(j)} = \mathbf{E} + \mathbf{u}_j\mathbf{v}_j^T$
 - 12: $\mathbf{U}^{(j)} = \mathbf{R} - \mathbf{u}_j$
 - 13: $\mathbf{h} = (\mathbf{B} \odot \mathbf{X}^{(j)})\mathbf{v}_j$
 - 14: $\mathbf{u}_j = \left[\frac{\mathbf{h} - \frac{\mathbf{U}^{(j)T}\mathbf{h}}{\mathbf{U}^{(j)T}\mathbf{U}^{(j)}}\mathbf{U}^{(j)}}{\mathbf{B}\mathbf{v}_j^2} \right]_+$
 - 15: $\mathbf{v}_j = \left[\frac{(\mathbf{B} \odot \mathbf{X}^{(j)})^T\mathbf{u}_j}{\mathbf{B}^T\mathbf{u}_j^2} \right]_+$
 - 16: $\mathbf{R} = \mathbf{U}^{(j)} + \mathbf{u}_j$
 - 17: $\mathbf{E} = \mathbf{X}^{(j)} - \mathbf{u}_j\mathbf{v}_j^T$
 - 18: **end for**
 - 19:
 - 20: $\mathbf{X}' = \mathbf{UV}^T$
 - 21: **until** Convergence criterion is satisfied.
-

CHAPTER 4. COLUMN-WISE UPDATE ALGORITHMS FOR
ORTHOGONAL NONNEGATIVE MATRIX FACTORIZATION

Proof. Let us suppose a given data matrix $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N] \in \mathbb{R}^{M \times N}$ and impose the orthogonal constraint into $\mathbf{V}$ instead of $\mathbf{U}$, that is, $\mathbf{V}^T \mathbf{V} = \mathbf{I}$.⁴ We first consider the minimization problem for ONMF with Bregman divergence:

$$D_\phi(\mathbf{X} || \mathbf{U}\mathbf{V}^T) = \sum_{n=1}^N D_\phi(\mathbf{x}_n || (\mathbf{U}\mathbf{V}^T)_n),$$

where $(\mathbf{U}\mathbf{V}^T)_n$ denotes the n th column vector of matrix $\mathbf{U}\mathbf{V}^T$. As stated before, each row vector of the orthogonal nonnegative matrix $\mathbf{V}$ has only one non-zero value. In a clustering task, this non-zero value corresponds to the clustering index that the data belong to. Therefore, we can rewrite the minimization problem as

$$\sum_{n=1}^N \sum D_\phi(\mathbf{x}_n || (\mathbf{U}\mathbf{V}^T)_n) = \sum_{j=1}^J \sum_{n:v_{nj} \neq 0} D_\phi(\mathbf{x}_n || v_{nj} \mathbf{u}_j).$$

According to [DLP08], let us impose the row normalization condition

$$v_{nj} = 1.$$

Then, it suffices to minimize

$$\sum_{\pi_j=1,2,\dots,J} \sum_{n \in \pi_j} D_\phi(\mathbf{x}_n || \mathbf{u}_j).$$

The last thing I need to show is $\mathbf{u}_j = \mu_j = \sum_{n \in \pi_j} \frac{1}{|\pi_j|} \mathbf{x}_n$, but this has already been proved in previous studies [BMDG05, BGW05]: the best predictor in Bregman divergence is the arithmetic mean of the data. Therefore, the optimal solution $\mathbf{u}_j^*$ with a fixed $\mathbf{V}$ is given by

$$\mathbf{u}_j^* = \sum_{n:v_{nj} \neq 0} \frac{1}{|\pi_j|} \mathbf{x}_n = \sum_{n \in \pi_j} \frac{1}{|\pi_j|} \mathbf{x}_n = \mu_j.$$

□

Since Bregman hard clustering is applicable to various data types with appropriate choices of $\phi(x)$ (*e.g.*, text data with KL-divergence and speech data with IS-divergence), Bregman divergence ONMF has a wider variety of applications than does the standard ONMF.

⁴This formulation is acceptable since the problem is equivalent to problem (4.1) with transpose $\mathbf{X} \leftarrow \mathbf{X}^T$.

Table 4.3: Datasets used in the experiments. Here #nnz is the number of non-zero values.

Dataset	Size	#nnz	Source
20Newsgroup	61188×18774	2435219	Document
TDT	36771×9394	1224135	Document
RCV	29992×9625	730879	Document
Reuters21678	18993×8293	389455	Document
MNIST	784×70000	10505375	Image
ORL64x64	4096×400	1638400	Image

4.5 Performance Evaluation

4.5.1 Datasets

We compared the performance of those algorithms for six real-life datasets and one artificial dataset. For the artificial dataset, I followed the setting in [LLP12].⁵ A summary of the datasets is given in Table 4.3.⁶

4.5.2 Compared Algorithms and Evaluation Measures

In ONMF problems, I compared the proposed HALS-ONMF with four traditional Frobenius norm ONMF algorithms. In Bregman divergence ONMF problems, I compared sBCD-ONMF with Li’s KL-divergence ONMF. In addition, in order to investigate the trade-off of orthogonality and approximation in Bregman divergence, I also compared sBCD-ONMF with sBCD-NMF, although sBCD-NMF does not have an imposed orthogonality constraint. This is because sBCD-ONMF is the first and only one ONMF algorithm workable with any Bregman divergence. We used IS-divergence ($\phi(x) = -\log x$) and β -divergence ($(\frac{1}{\beta(\beta+1)})(x^{\beta+1} - (\beta+1)x + \beta)$ with $\beta = 2$) in this comparison. We measured the orthogonality and approximation accuracy. We compared all of the algorithms shown in Table 4.1 except for Pompili’s ONMF [PGAG12]. Pompili’s ONMF algorithm [PGAG12] was not used because their algorithm attains orthogonality but not nonnegativity. It is also reported already in their own comparison [PGAG12]. We also noted that their algorithm was

⁵ The code of their sBCD-NMF algorithm [LLP12] and the data generator are available at <http://www.cc.gatech.edu/grads/l/li86/sbcd.zip>

⁶These datasets are downloadable from <http://www.cad.zju.edu.cn/home/dengcai/Data/TextData.html> [CWH09].

CHAPTER 4. COLUMN-WISE UPDATE ALGORITHMS FOR ORTHOGONAL NONNEGATIVE MATRIX FACTORIZATION

slowest among all the compared algorithms. For the weighting parameter α in Li's ONMF [LWP10] and Mirzal's ONMF [Mir14], I used $\alpha = 1$, because the value worked satisfactorily on most datasets.

We employed the same evaluation setting as that in [LLP12]. Ten trials with different initial values are conducted and the average values of measurements were shown here. We fixed the number of iterations to 100 for all of the algorithms. We evaluated the degree of approximation and the degree of orthogonality by

$$\text{Normalized Residual Value: } \frac{\|\mathbf{X} - \mathbf{U}\mathbf{V}^T\|_F^2}{\|\mathbf{X}\|_F^2} \quad (\text{for Frobenius ONMF}) \quad (4.20)$$

$$\text{Relative Residual Value: } \log_{10} \frac{D_\phi(\mathbf{X} \|\mathbf{U}\mathbf{V}^T)}{D_\phi(\mathbf{X} \|\mathbf{U}_0\mathbf{V}_0^T)} \quad (\text{for Bregman ONMF}) \quad (4.21)$$

$$\text{Orthogonality: } \|\mathbf{U}^T\mathbf{U} - \mathbf{I}\|_F^2 \quad (4.22)$$

Here, $\mathbf{U}_0$ and $\mathbf{V}_0$ are the matrices used for initialization. In addition, I evaluated the computation time (seconds), the normalized residual value (4.20), and the degree of orthogonality (4.22) for Frobenius norm ONMF. For Bregman divergence ONMF, I evaluated the relative residual value (4.21) and (4.22) since (4.20) cannot be appropriately normalized for Bregman divergence.

4.5.3 Comparison on ONMF problems

Figure 4.1 shows the values of the normalized residual for $J = 30$ (number of components) for the six real-life datasets. The proposed HALS-ONMF converges faster than do the other ONMF algorithms. HALS-ONMF converges before 250 seconds for all six datasets. This is because HALS-ONMF needs a smaller number of iterations, because of the fact that HALS-ONMF solves vector-wise problems with the analytical solutions (4.8). Figure 4.2 shows the degrees of orthogonality attained. The HALS-ONMF achieves almost the highest degree of orthogonality among the algorithms in the early stage, though the final degree of orthogonality is slightly worse than that of others. In the ORL dataset (Figure 4.2(e)), a dense dataset, only HALS-ONMF succeeded in achieving an acceptable degree of orthogonality.

To show the speed of convergence, I defined the stopping criterion of the iteration according to the way conventional researches adopted [PGAG12, KP08] as:

$$\frac{\|\mathbf{X} - \mathbf{U}^{t-1}\mathbf{V}^{t-1T}\|_F^2 - \|\mathbf{X} - \mathbf{U}^t\mathbf{V}^{tT}\|_F^2}{\|\mathbf{X}\|_F^2} < \epsilon, \quad (4.23)$$

CHAPTER 4. COLUMN-WISE UPDATE ALGORITHMS FOR ORTHOGONAL NONNEGATIVE MATRIX FACTORIZATION

where ϵ is a threshold and $\mathbf{U}^t$ and $\mathbf{V}^t$ are matrices after t th update. In this experience, I set the threshold ϵ to 10^{-4} in all datasets.

Table 4.4 shows the result on four datasets when I terminated the calculation with the stopping criterion (4.23).⁷ The proposed HALS-ONMF is the fastest with the smallest number of iterations. The proposed algorithm converged about 1.6 to 4.1 times faster than the others, keeping comparable approximation accuracy and orthogonality.

⁷We omit the results on *RCV* dataset and *ORL* dataset because, on *RCV* dataset, Li's ONMF [LWP10] and Mirzal's ONMF [Mir14] and, on *ORL* dataset, all ONMF algorithms except the proposed HALS-ONMF decreased the approximation error too slowly and made the stopping criterion (4.23) satisfied in only a few iterations. See Fig.1(d)(e).

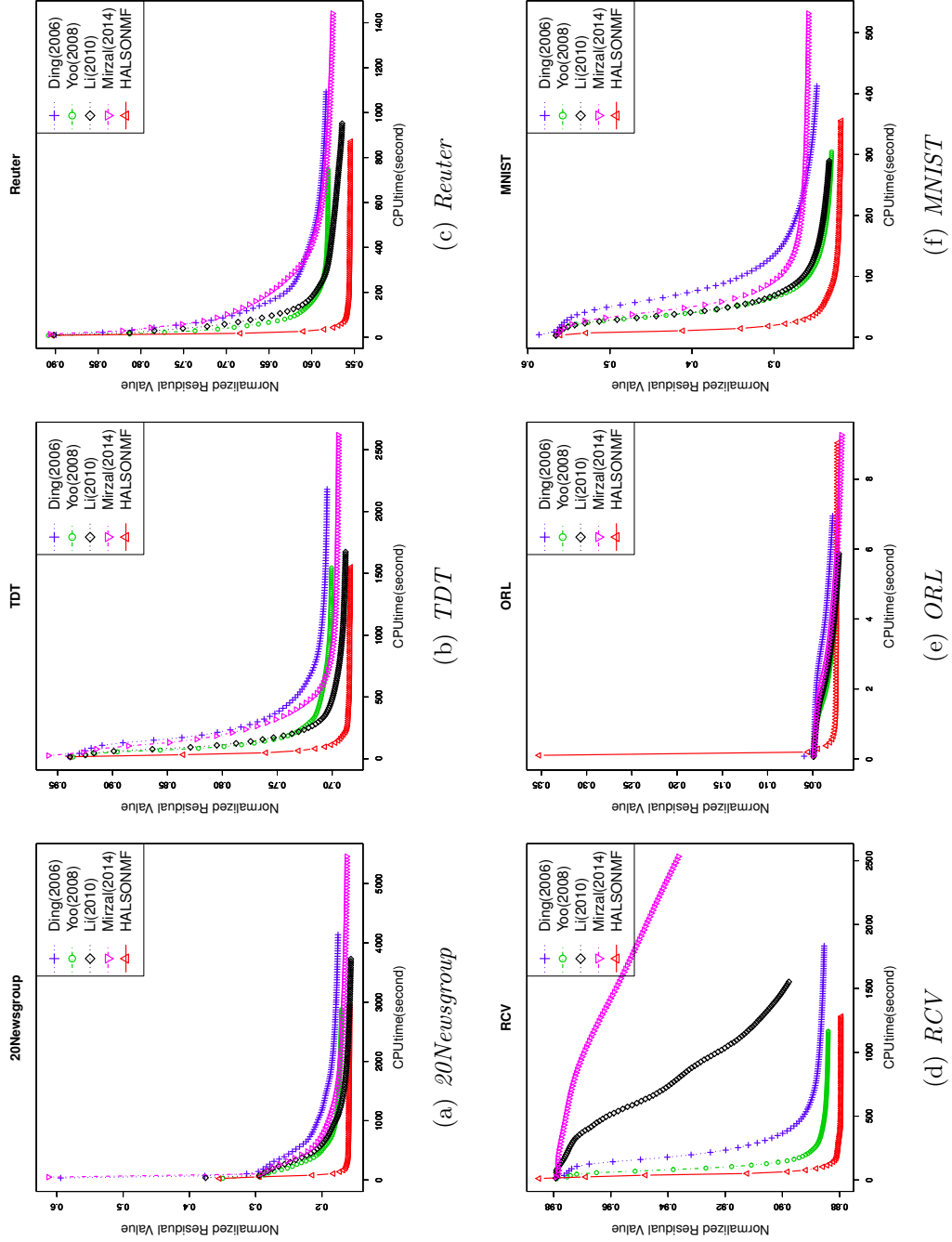


Figure 4.1: Comparison of five ONMF algorithms in the degree of approximation on six-real life datasets. The proposed HALS-ONMF algorithm converges faster than the other four conventional algorithms.

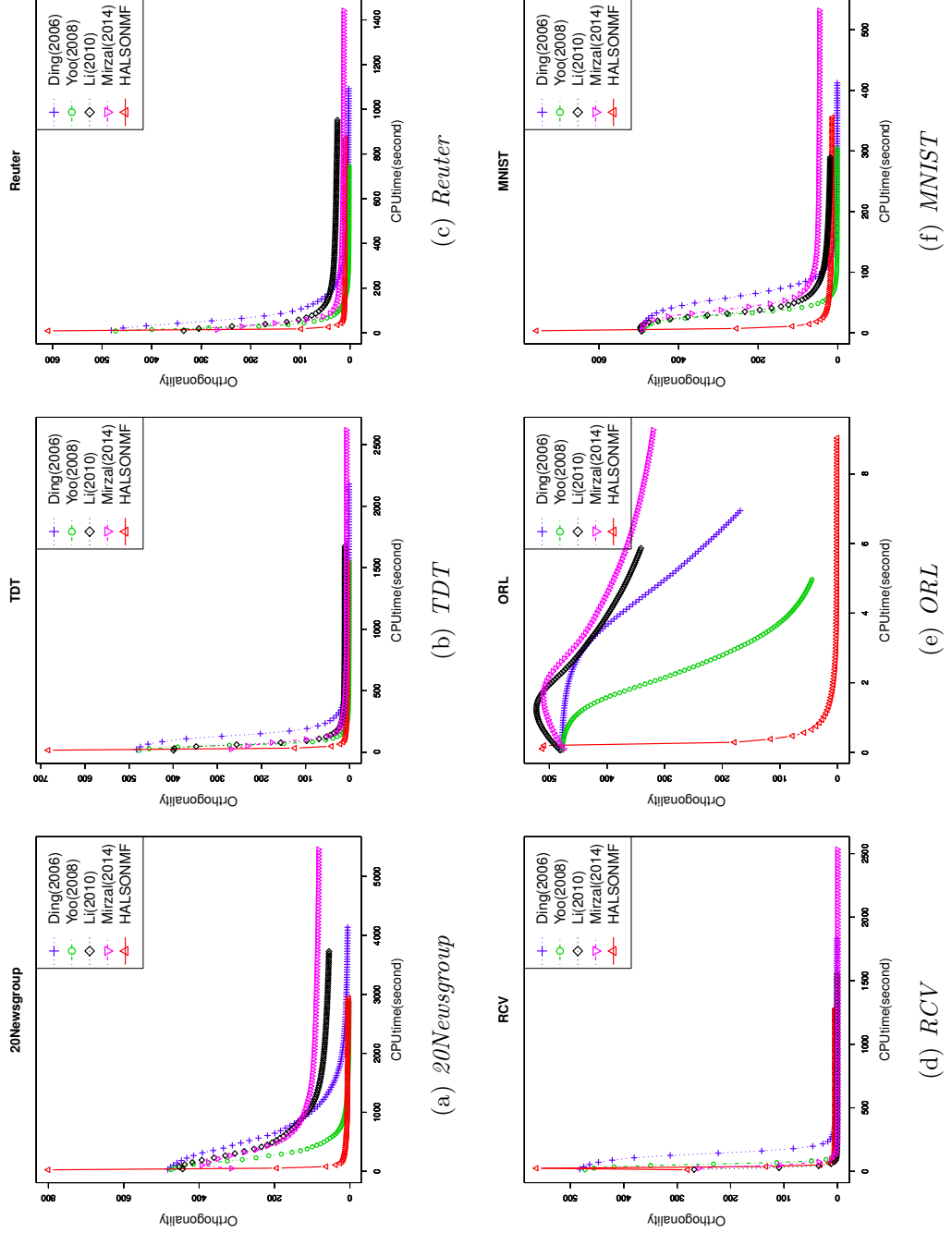


Figure 4.2: Orthogonality attained by three ONMF algorithms. The proposed HALS algorithm converges faster than the other four conventional algorithms, but the final state is worse than some of the others.

4.5.4 Comparison on Bregman Divergence ONMF Problems

In Bregman divergence ONMF problems, I compared sBCD-ONMF with Li's KL-divergence ONMF [LWP10] and sBCD-NMF with KL-divergence. In addition, I compared sBCD-ONMF with sBCD-NMF for IS- or β -divergence. Unfortunately, since KL-divergence and IS-divergence do not allow zero values ($0 \notin \text{dom}_\phi$), most datasets were not suitable for this comparison. Besides, sBCD-NMF or sBCD-ONMF does not scale because of their high computational costs (see, for example, Step (11) and Step (17) in Algorithm 10). Therefore, I dealt with only one artificial dataset of $\mathbf{X} \in \mathbb{R}^{2000 \times 1000}$.

The results are shown in Figure 4.3. As predicted, the sBCD-NMF algorithm without an orthogonal constraint achieved better approximation than did the algorithms with orthogonal constraints, Li's ONMF and sBCD-ONMF, while the latter two achieved a higher degree of orthogonality. In comparison of convergence speeds, sBCD-ONMF is almost the same as sBCD-NMF or even faster. Li's ONMF is inferior to sBCD-ONMF in convergence speed.⁸

In total, I can say that sBCD-ONMF is a fast algorithms to find a solution in Bregman divergence ONMF problems with a sufficient degree of orthogonality at the expense of a little amount of degradation of approximation.

⁸In [LWP10], it is reported that Li's KL-divergence ONMF algorithm needs a large number of iterations to attain a sufficient level of orthogonality.

CHAPTER 4. COLUMN-WISE UPDATE ALGORITHMS FOR
ORTHOGONAL NONNEGATIVE MATRIX FACTORIZATION

Table 4.4: The results when the stopping criterion (4.23) with $\epsilon = 10^{-4}$ is applied on four datasets. NRV means Normalized Residual Values (4.20).

Dataset: *20Newsgroup* ($\epsilon = 10^{-4}$)

Algorithm	Evaluation criterion			
	#iteration	CPU time(s)	NRV	Orthogonality
Ding [DLPP06]	89	3638.784	0.177	6.960
Yoo [YC08]	65	1899.256	0.173	3.606
Li [LWP10]	83	3101.354	0.157	58.566
Mirzal [Mir14]	78	4289.880	0.164	85.314
HALSONMF	16	459.936	0.159	9.642

Dataset: *TDT* ($\epsilon = 10^{-4}$)

Algorithm	Evaluation criterion			
	#iteration	CPU time(s)	NRV	Orthogonality
Ding [DLPP06]	68	1471.580	0.706	1.957
Yoo [YC08]	66	1008.414	0.702	1.242
Li [LWP10]	68	1127.494	0.689	11.228
Mirzal [Mir14]	48	1267.000	0.697	8.031
HALSONMF	20	326.622	0.685	5.441

Dataset: *Reuter* ($\epsilon = 10^{-4}$)

Algorithm	Evaluation criterion			
	#iteration	CPU time(s)	NRV	Orthogonality
Ding [DLPP06]	93	1014.908	0.584	2.751
Yoo [YC08]	52	383.856	0.582	2.221
Li [LWP10]	100	951.008	0.564	25.394
Mirzal [Mir14]	93	1337.524	0.576	13.128
HALSONMF	23	197.200	0.556	9.825

Dataset: *MNIST* ($\epsilon = 10^{-4}$)

Algorithm	Evaluation criterion			
	#iteration	CPU time(s)	NRV	Orthogonality
Ding [DLPP06]	100	412.206	0.248	1.469
Yoo [YC08]	94	285.862	0.230	1.312
Li [LWP10]	100	289.862	0.233	19.011
Mirzal [Mir14]	68	360.948	0.261	8.031
HALSONMF	48	170.702	0.221	5.441

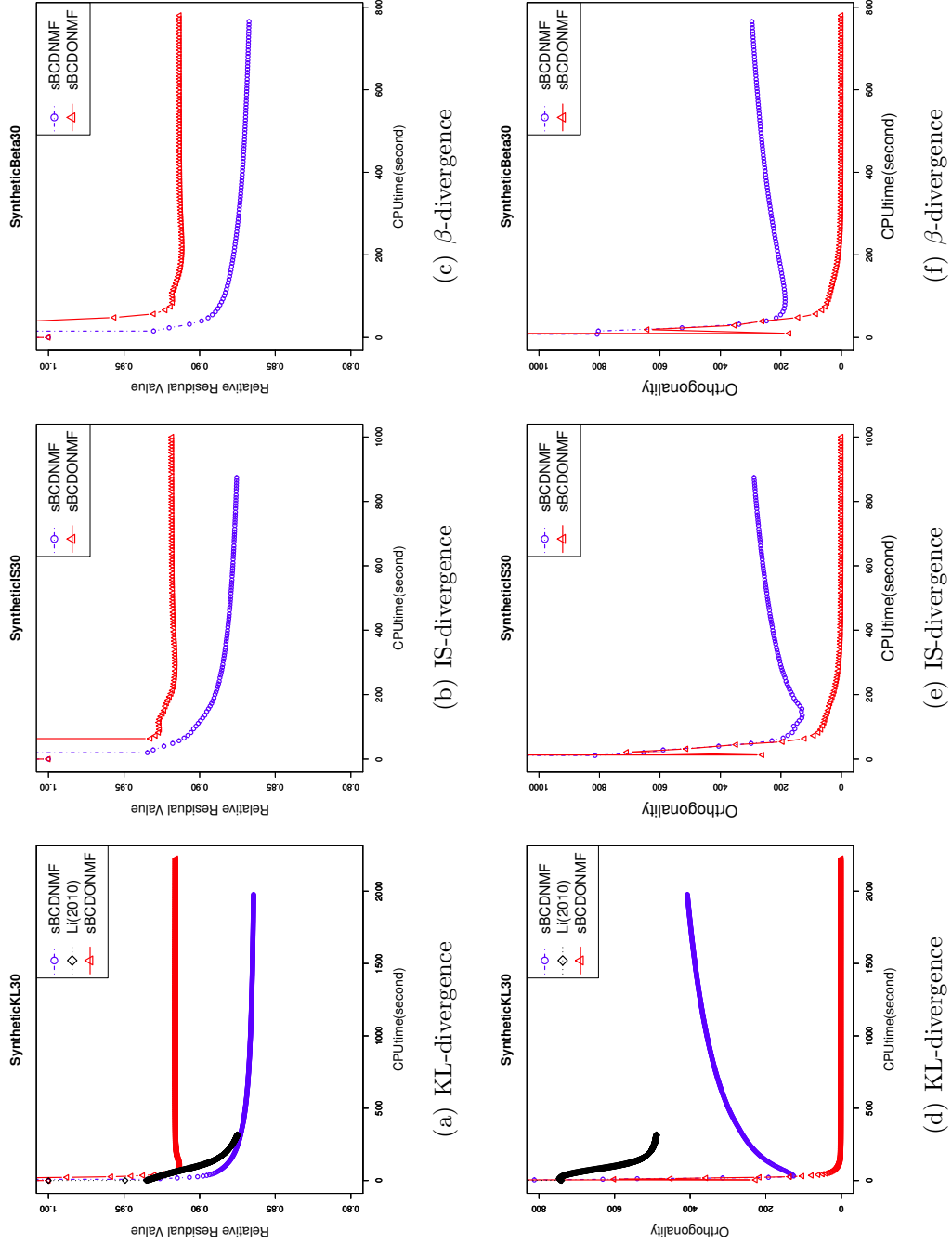


Figure 4.3: Comparison of KL-divergence, IS-divergence and β -divergence NMF on an artificial dataset ($\mathbf{X} \in \mathbb{R}^{2000 \times 1000}$). The first row shows the value of relative residual and the second row shows the degree of orthogonality.

Table 4.5: Datasets used in the clustering experiments.

Dataset	# instance	# feature	#class
k1b	2340	21819	6
reviews	4069	18483	5
sports	8580	14870	7
hitech	2301	10080	7

4.5.5 Clustering Experiments

As I stated before, ONMF is suitable for clustering tasks more than standard NMF. This is because the constrained matrix $\mathbf{U}$ can be considered as an indicator matrix in ONMF. Let $\mathbf{X}$ be an *instance* $\times$ *feature* matrix factorized by $\mathbf{UV}^T$. Then i th row of $\mathbf{U}$ can be considered as a membership vector of instance i to J groups (features). Especially, a solution $\mathbf{UV}^T$ in ONMF is expected to have a crisp membership of a single one. We assign the i th instance to k th cluster such as

$$k = \underset{j}{\operatorname{argmax}} \mathbf{U}_{ij}.$$

We compared the proposed HALS-ONMF and sBCD-ONMF with one standard NMF algorithm (HALS-NMF) and conventional ONMF algorithms (Ding’s ONMF [DLPP06] and Yoo’s ONMF [YC08]). We set the number iteration to 30 which was sufficient for convergence in the previous experiments in Sections 5.3 and 5.4. In addition, I conducted k -means algorithm as a base-line method. We used four TREC document classification datasets (see Table 4.5 for the detail). Since these dataset have class labels, I hid them for clustering and then evaluated the difference between the true clustering induced by the class labels and the obtained clustering.

We measured Normalized Mutual Information (NMI) defined as

$$\text{NMI} : \frac{I(\hat{C}; C)}{(H(\hat{C}) + H(C))/2},$$

where $\hat{C}$ is the predicted clustering and C is the ground truth. Here, $H(\cdot)$ is Shanon Entropy, and $I(\cdot)$ is the Mutual Information. We averaged the results for ten trials with different initial points.

Unfortunately, the general advantage of ONMF over NMF was not confirmed⁹, as long as their algorithms are with Frobenius norm. Nevertheless,

⁹Note that the orthogonal constraint gives more crisp membership, however, this does not mean better clustering accuracy.

the proposed HALS-ONMF achieved the best score in NMI among them. Rather, I confirmed the advantage of KL-divergence over Frobenius norm and IS-divergence. This is not an unexpected result because it is known that the document data is well explained by Multinomial distribution models and minimizing KL-divergence is corresponding to maximum likelihood with Multinomial distribution model [BMDG05, LLP12]. The best choice is one of conventional SBCD-NMF with KL-divergence, sBCD-ONMF with KL-divergence, and HALS-ONMF with Frobenius norm.

Table 4.6: The clustering results evaluated in NMI (the higher, the better) on four-real life datasets.

Criterion	Algorithm	Dataset			
		klb	reviews	sports	hitech
Frobenius norm	K-means	0.503 $\pm$ 0.018	0.338 $\pm$ 0.040	0.235 $\pm$ 0.022	0.197 $\pm$ 0.019
	HALS-NMF [CZPA09, LLP12]	0.527 $\pm$ 0.081	0.337 $\pm$ 0.040	0.268 $\pm$ 0.024	0.299 $\pm$ 0.020
	Ding's ONMF [DLPP06]	0.520 $\pm$ 0.039	0.333 $\pm$ 0.082	0.242 $\pm$ 0.029	0.292 $\pm$ 0.017
	Yoo's ONMF [YC08]	0.559 $\pm$ 0.035	0.370 $\pm$ 0.065	0.271 $\pm$ 0.040	0.315 $\pm$ 0.018
	HALS-ONMF	0.540 $\pm$ 0.050	0.420 $\pm$ 0.095	0.390 $\pm$ 0.048	0.334 $\pm$ 0.015
KL- divergence	sBCD-NMF [LLP12]	0.593 $\pm$ 0.032	0.530 $\pm$ 0.071	0.544 $\pm$ 0.032	0.271 $\pm$ 0.048
	sBCD-ONMF	0.587 $\pm$ 0.019	0.474 $\pm$ 0.095	0.610 $\pm$ 0.040	0.228 $\pm$ 0.017
IS- divergence	sBCD-NMF [LLP12]	0.032 $\pm$ 0.045	0.022 $\pm$ 0.017	0.070 $\pm$ 0.059	0.016 $\pm$ 0.017
	sBCD-ONMF	0.025 $\pm$ 0.010	0.048 $\pm$ 0.024	0.030 $\pm$ 0.020	0.034 $\pm$ 0.020

4.6 Concluding Remarks

We have proposed a fast algorithm for solving one-sided orthogonal nonnegative matrix factorization problems in the Frobenius norm and in Bregman divergence. Orthogonal NMF algorithms proposed so far suffered from slow convergence mainly due to their matrix-wise updates. By decomposing the matrix-type orthogonality condition into a set of column-wise orthogonality conditions, I succeeded in speeding up the convergence. One of the proposed algorithms is the first algorithm to solve a Bregman divergence NMF problem with an orthogonal constraint. In addition, I showed that Bregman divergence ONMF problem is equivalent to Bregman hard clustering. Experiments for six real-life datasets and an artificial dataset demonstrated that the proposed algorithms are in fact faster than state-of-the-art algorithms in convergence while keeping a satisfactory level of orthogonality. In the best case, the proposed algorithm converged more than four times faster than state-of-the-art algorithms.

Chapter 5

Variable Selection for Efficient Sparse Nonnegative Tensor Factorization

5.1 Introduction

Recently, many kinds of data are necessary to be analyzed from multi-aspect and multi-viewpoint. A tensor, multi-dimensional array, is chosen as an appropriate representation to this end. On the other hand, analyzing tensors is naturally more difficult and costly compared with their low-dimensional version such as vectors and matrices. Nonnegative Tensor Factorization (NTF) is a popular technique to analyze tensor data. It decomposes a nonnegative tensor into a product of nonnegative factor matrices. The non-negativity is natural in many applications such as video analysis, text analysis and network analysis as seen in the literature [CZS⁺07, CZPA09, KB09, LS99].

Typically, an NTF problem is solved by solving a set of Nonnegative Matrix Factorization (NMF) problems that are transformed from the tensor problem by matricization (reshaping a tensor into a matrix). In more detail, NTF solves NMF problems iteratively. Hence, the computational cost depends on how fast we can solve NMF problems [CZPA09]. The dominant computation is used for a multiplication of a matricized tensor and a Khatri-Rao product of matrices, which is needed in the middle of calculation of gradients. This calculation cost is larger than those of the other calculations. Therefore, no essential improvement cannot be expected as long as this part is included in the algorithm.

In this chapter, I propose a variable selection algorithm to cope with this problem and reduce both computational cost and memory consumption. We

explicitly utilize the sparse property of NTF or NMFs, that is, I assume that the factor matrices of NTF or NMFs are of many zeros. Such sparsity is often observed in several previous works [CZPA09, KHP14, KB09, SH05, PFS12, LS99, MHA08]. Relying on this sparsity, I use only variables/elements which have non-zero values. We calculate in each iteration step an upper-bound of the degree of contribution of a single variable for approximation and then discard the variables for which a sufficient degree of contribution by updating cannot be expected. In addition, in case the factor matrices are not sufficient sparse, I propose a lazy coordinate descent algorithm in which I update only a fixed number of variables/elements chosen by importance sampling.

5.1.1 Nonnegative Tensor Factorization

Given a 3-order tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ and an integer J , my goal is to find three matrices $\mathbf{A} \in \mathbb{R}^{I_1 \times J}$, $\mathbf{B} \in \mathbb{R}^{I_2 \times J}$ and $\mathbf{C} \in \mathbb{R}^{I_3 \times J}$ minimizing

$$\|\underline{\mathbf{X}} - \underline{\mathbf{I}} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C}\|_F^2, \quad (5.1)$$

where $\|\cdot\|_F^2$ is the squared Frobenius norm and $\underline{\mathbf{I}} \in \mathbb{R}^{J \times J \times J}$ is the super-diagonal identity tensor with one's as the diagonal elements. This special type of formulation is called CANDECOMP/PARAFAC decomposition [KB09] and has been studied intensively. Such a decomposition with super-diagonal tensor is useful for factor based analysis. This is because we can rewrite the problem (5.1) to a component-wise (column) formulation with outer products (see Fig 5.1):

$$\|\underline{\mathbf{X}} - \underline{\mathbf{I}} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C}\|_F^2 \iff \|\underline{\mathbf{X}} - \sum_{j=1}^J \underline{\mathbf{I}}_{jjj} * \mathbf{a}_j \circ \mathbf{b}_j \circ \mathbf{c}_j\|_F^2.$$

Imposing the diagonal elements to be one's does not lose generality. If it is not the case, we may pass the values to matrix $\mathbf{C}$ to ensure the assumption. We may impose more constraints in $\mathbf{A}$ and $\mathbf{B}$ such that each column of $\mathbf{A}$ and $\mathbf{B}$ has norm one in L_1 distance such as $\|\mathbf{a}_j\|_1^2 = 1$ and $\|\mathbf{b}_j\|_1^2 = 1$.

A problem of NTF can be translated into a set of problems of Nonnegative Matrix Factorization (NMF) by matricization as follows. A composed 3-order tensor in (5.1) can be written as

$$(\underline{\mathbf{I}} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C})_{(1)} \iff \mathbf{A}(\mathbf{C} \odot \mathbf{B})^T.$$

Hence, minimization problem for $\mathbf{A}$ becomes

$$\min_{\mathbf{A}} \|\underline{\mathbf{X}}_{(1)} - \mathbf{A}(\mathbf{C} \odot \mathbf{B})^T\|_F^2. \quad (5.2)$$

With fixed $\mathbf{B}$ and $\mathbf{C}$, $(\mathbf{C} \odot \mathbf{B})$ is a (large) matrix and, thus, this problem is reduced to an NMF problem to minimize (5.2) in $\mathbf{A}$ [KB09, LS99]. Similarly, we have two more NMF problems: $\min_{\mathbf{B}} \|\underline{\mathbf{X}}_{(2)} - \mathbf{B}(\mathbf{C} \odot \mathbf{A})^T\|_F^2$ and $\min_{\mathbf{C}} \|\underline{\mathbf{X}}_{(3)} - \mathbf{C}(\mathbf{A} \odot \mathbf{B})^T\|_F^2$.

To solve above three problems in an iterative manner, we calculate the gradients of (5.2) in $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ in order. For example, the gradient in $\mathbf{A}$ is given by

$$\frac{\partial \|\underline{\mathbf{X}}_{(1)} - \mathbf{A}(\mathbf{C} \odot \mathbf{B})^T\|_F^2}{\partial \mathbf{A}} = -2\underline{\mathbf{X}}_{(1)}(\mathbf{C} \odot \mathbf{B}) + 2\mathbf{A}(\mathbf{B}^T \mathbf{B} * \mathbf{C}^T \mathbf{C}). \quad (5.3)$$

The calculation of R.H.S. of (5.3) requires a high computational cost. Especially, the first term $\underline{\mathbf{X}}_{(1)}(\mathbf{C} \odot \mathbf{B})$ ($\underline{\mathbf{X}}_{(1)} \in \mathbb{R}^{I_1 \times I_2 I_3}$, $(\mathbf{C} \odot \mathbf{B}) \in \mathbb{R}^{I_2 I_3 \times J}$) needs $O(I_1 I_2 I_3 J)$ in time and $O(I_1 I_2 I_3)$ in space. On CANDECOMP/PARAFAC solvers without nonnegative constraint, $\mathbf{C} \odot \mathbf{B}$ is a matrix of an intractable size, with millions or billions rows, because $\mathbf{B}$ and $\mathbf{C}$ are often dense in that case. This intractable size of temporally generated intermediate matrices is called *intermediate data explosion* (IDE) [KPHF12]. On NTF problems, matrices $\mathbf{C}$ and $\mathbf{B}$ are often sparse, so that the size of IDE can be smaller than those of TF problems. Even so, this calculation of $O(I_1 I_2 I_3 J)$ is still far larger than the other calculations of $O(I_1 J^2)$ ($J \ll I_2, I_3$).

5.2 Related Works

Most of studies in acceleration of NTF are based on fast optimization algorithms for NMF. This is natural because NTF solves NMF problems iteratively as I stated before. Cichocki *et al.* proposed Hierarchical Alternating Least Squares (HALS) algorithm, a vector-wise coordinate descent algorithm, for NTF [CAH09]. Park *et al.* also applied block pivoting methods, one of the state-of-the-art algorithms of NMF, for NTF [KHP14]. These optimization algorithms contributed in speeding up of convergence and succeeded in reducing the number of iterations. However, the theoretical complexity of time and space necessary for updating has not been reduced and unchanged. In other words, they did not solve the IDE problem.

Another type of fast NTF algorithms are based on parallel computing. Phan *et al.* proposed a distributed algorithm called *GRID-PARAFAC* [PTC12]. This algorithm separates the target tensor into grids and factorizes smaller tensors in parallel while sharing the information for the gradient. By sharing the same information for the gradients, they succeeded to factorize the tensor without loss of approximation accuracy. On the other hand, *Par-Cube* [PFS12] chose some indices by sampling for each mode and generated

CHAPTER 5. VARIABLE SELECTION FOR EFFICIENT SPARSE NONNEGATIVE TENSOR FACTORIZATION

smaller tensors. After factorization of those smaller tensors, they merged the factorization results into one. This algorithm factorized the tensor faster than the other algorithms with a smaller amount of space, but the approximation accuracy is sacrificed instead. *PARACOMP* [S⁺14] is another sampling algorithm. Before factorizing smaller tensors, they compressed a tensor with a randomized matrix. The compression and decomposition processes reduced largely the space and brought the speed improvement. After factorization, they gathered the factorization results and uncompressed it. This algorithm, however, also suffers from a large approximation error as ParCube does.

The essential problem to be solved is IDE, that is, the multiplication of a matricized tensor and a Khatri-Rao product in the first term of R.H.S. of (5.3). To cope with IDE, Kolda and Sun proposed a scalable tensor factorization method [KS08] which makes the IDE calculation matrix- or vector-wisely so as to be made in the memory size. Instead of a smaller memory size, their approach needs a larger computation time. *Gigatensor* [KPHF12] decomposed the multiplication of the matricized tensor and the Khatri-Rao products of matrices into two different calculations. This succeeded to avoid calculating Khatri-Rao product ($\mathbf{C} \odot \mathbf{B}$). Gigatensor worked with a small amount of memory space suitable to parallel computation. Choi and Vishwanathan proposed a distributed calculation method called *DFacto* and coped with IDE by transforming the multiplication into two operations of reshaping of tensor and matrix multiplication [CV14]. They utilized the sparsity of the target tensor to avoid the unnecessary calculation, and succeeded to improve the computation speed. On the other hand, DFacto requires the matricized tensors for all mode to be kept instead. Gigatensor and DFacto have developed algorithms for CANDECOMP/PARAFAC factorization, assuming that composing matrices are dense. Indeed, unless constraint of nonnegativity is imposed, factor matrices become dense typically. However, with nonnegativity, we can expect the sparsity of factor matrices and the nature for improving algorithms. Usage of sparsity is the first ingredient of the proposed algorithm.

In viewpoint of variable selection, feature eliminations are related to the proposed methods. SAFE [GVR10] predicts features which will become zero after solving their problem, sparse regressions such as Lasso or Elastic net, and eliminates such features before solving. In the sense of eliminating unnecessary features, SAFE and the proposed methods are close to each other. However, the proposed methods needs to select variables in each update unlike SAFE does only once before solving there problem. Therefore, settings are somehow different. Indeed, the best of my knowledge, there are no methods for feature eliminations for alternating update procedures. In this sense, my method is the first trial for such problems.

In the NMF literature, the closest work to the proposed methods is a greedy algorithm of variable selection called Greedy Coordinate Descent (GCD) proposed for NMF, not for NTF, by Hsieh *et al.* [HD11]. This algorithm selects variables for updating according to the degree of contribution to the approximation error reduction. This degree is calculated by Taylor Expansion. Since they update only selected variables, the convergence speed was greatly improved compared with the other NMF algorithms. The dominant computation is consumed in the calculation of gradients. Therefore, their GCD algorithm cannot be expected to speed up NTF algorithms, because GCD needs to calculate the gradient first to evaluate the contribution degree of variables. In other words, IDE problem was not solved. In this paper, as the second ingredient, we estimate the degree of contribution using an upper-bound without actual calculation of the gradient. That is, for each variable we estimate from above the degree of contribution without solving the original IDE problem, and reduce the size of IDE by choosing variables necessary for updating.

5.3 NTF with A Variable Selection

We show the outline of the proposed algorithm in Fig. 5.1 and the pseudo-code in Algorithm 11.

The key idea is very simple. In each round of updating of one component, say, $\mathbf{b}_1$ (see the second line of Fig.5.1), while the others are fixed, I estimate the zero or less elements in $\mathbf{b}_1$ and update only the other (positive) elements. In addition, I choose a fixed number K of elements by importance sampling according to the estimated degree of contribution by one updating for responding to memory limitation. This is made only if the number of positive elements is larger than K . Once variables for updating are chosen, I calculate the gradients only in these variables. That is, I solve the core NMF problem (5.2) causing IDE with smaller matrices than original matrices as seen in Step 10 in Algorithm 11.

In the following, I describe the details of two procedures of variable selection and importance sampling.

5.3.1 Variable Selection

We explain the procedure on $\mathbf{A}$ only. The same procedure is applied to $\mathbf{B}$ and $\mathbf{C}$ in this order. Our goal is to select non-zero variables column-wisely, $\mathbf{A}_{:j}$ for each $j \in \{1, 2, \dots, J\}$, aiming at reducing the cost of calculation of the column-wise gradient $\underline{\mathbf{X}}_{(1)}(\mathbf{C}_{:j} \odot \mathbf{B}_{:j})$.

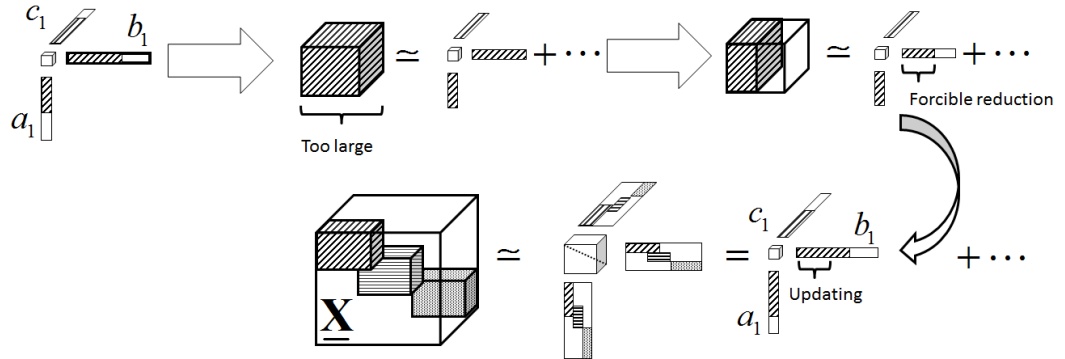
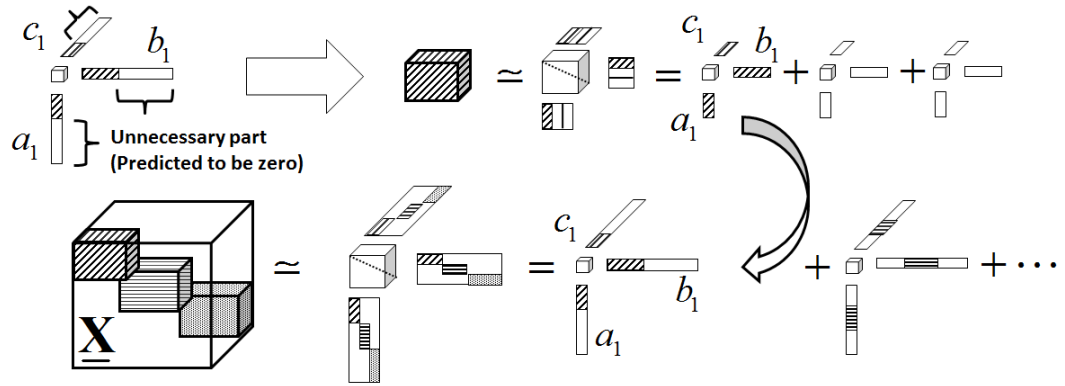
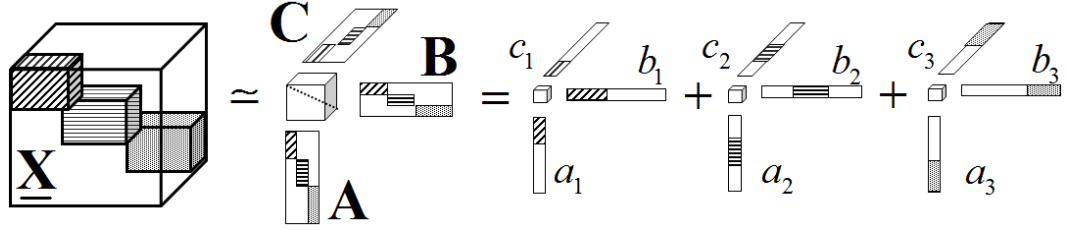


Figure 5.1: Outline of the proposed NTF algorithm in case of three factors ($J = 3$). For each column, we predict non-zero variables and factorize the corresponding smaller tensor instead the original tensor. Then, we update the selected variables only after replacing the other variables with zero.

CHAPTER 5. VARIABLE SELECTION FOR EFFICIENT SPARSE
NONNEGATIVE TENSOR FACTORIZATION

For a fixed j , the column-wise sub-problem of (5.2) is solved by setting gradient (5.3) to be zero [HD11]:

$$\mathbf{A}_{ij} \leftarrow [\mathbf{A}_{ij} + \frac{(\underline{\mathbf{X}}_{(1)}(\mathbf{C} \odot \mathbf{B}))_{ij} - (\mathbf{A}(\mathbf{B}^T \mathbf{B} * \mathbf{C}^T \mathbf{C}))_{ij}}{(\mathbf{B}^T \mathbf{B} * \mathbf{C}^T \mathbf{C})_{jj}}]_+. \quad (5.4)$$

Here, $[\cdot]_+$ enforces nonnegativity with a small positive value ϵ . If the value in the square brackets is zero or negative, then I do not update $\mathbf{A}_{ij}$. In other words, I skip (5.4) and replace with zero if

$$\mathbf{A}_{ij} * (\mathbf{B}^T \mathbf{B} * \mathbf{C}^T \mathbf{C})_{jj} + (\underline{\mathbf{X}}_{(1)}(\mathbf{C} \odot \mathbf{B}))_{ij} - (\mathbf{A}(\mathbf{B}^T \mathbf{B} * \mathbf{C}^T \mathbf{C}))_{ij} \leq 0, \quad (5.5)$$

To predict the value of the L.H.S. of (5.5), I use an upper-bound of $(\underline{\mathbf{X}}_{(1)}(\mathbf{C} \odot \mathbf{B}))_{ij}$ instead a direct calculation of the term. The upper-bound is obtained by Cauchy-Schwartz inequality as

$$(\underline{\mathbf{X}}_{(1)}(\mathbf{C} \odot \mathbf{B}))_{ij} \leq \|\underline{\mathbf{Y}}_{i1:}\|_1 * \|\mathbf{C}_{:j}\|_1 * \|\mathbf{B}_{:j}\|_1,$$

where $\underline{\mathbf{Y}}$ is given by

$$\underline{\mathbf{Y}}_{i_1 i_2 i_3} = \begin{cases} \underline{\mathbf{X}}_{i_1 i_2 i_3} & (\mathbf{B}_{i_2 j} \neq 0 \text{ and } \mathbf{C}_{i_3 j} \neq 0), \\ 0 & (\text{otherwise}). \end{cases} \quad (5.6)$$

That is, either $\mathbf{B}$ or $\mathbf{C}$ is sparse, so $\underline{\mathbf{Y}}$ is. It is clear from (5.6) that $\underline{\mathbf{Y}}$ is smaller than the original tensor $\underline{\mathbf{X}}$. Then condition (5.5) is replaced with

$$\mathbf{A}_{ij} * (\mathbf{B}^T \mathbf{B} * \mathbf{C}^T \mathbf{C})_{jj} + \|\underline{\mathbf{Y}}_{i:}\|_1 * \|\mathbf{C}_{:j}\|_1 - (\mathbf{A}(\mathbf{B}^T \mathbf{B} * \mathbf{C}^T \mathbf{C}))_{ij} \leq 0. \quad (5.7)$$

This is my solution for IDE problem, we do not need to construct $\mathbf{C} \odot \mathbf{B}$ nor to multiply it with $\underline{\mathbf{X}}_{(1)}$ anymore.

By this upper-bound, we can estimate positive variables without calculation of the multiplication in IDE problem. By Ω_{A_j} , I denote the set of indices of non-zero variables in the j th column of $\mathbf{A}$ (likewise, I denote Ω_{B_j} and Ω_{C_j} for the j th columns of $\mathbf{B}$ and $\mathbf{C}$, respectively). Then, collecting non-zero variables, I construct tensor $\underline{\mathbf{Z}}^{(j)} \in \mathbb{R}^{|\Omega_{A_j}| \times |\text{nnz}(\mathbf{B}_{:j})| \times |\text{nnz}(\mathbf{C}_{:j})|}$

$$\underline{\mathbf{Z}}^{(j)} = \underline{\mathbf{Y}}_{\Omega_{A_j}::} \in \mathbb{R}^{|\Omega_{A_j}| \times |\text{nnz}(\mathbf{B}_{:j})| \times |\text{nnz}(\mathbf{C}_{:j})|}, \quad (5.8)$$

where $\text{nnz}(\cdot)$ means the index set of non-zero elements and $|\text{nnz}(\cdot)|$ means the cardinality of $\text{nnz}(\cdot)$. We use $\text{nnz}(\mathbf{B}_{:j})$ and $\text{nnz}(\mathbf{C}_{:j})$ because vectors $\mathbf{B}_j$ and $\mathbf{C}_j$ are fixed in this stage and only $\text{nnz}(\mathbf{B}_{:j})$ and $\text{nnz}(\mathbf{C}_{:j})$ are necessary in the calculation $\underline{\mathbf{X}}_{(1)}(\mathbf{C}_{:j} \odot \mathbf{B}_{:j})$.

Using $\underline{\mathbf{Z}}^{(j)}$, we calculate the gradient (5.3) column-wisely with $\underline{\mathbf{Z}}_{(1)}^{(j)}(\mathbf{C}_{nnz(\mathbf{C}_{:j})j} \odot \mathbf{B}_{nnz(\mathbf{B}_{:j})j})$ instead of $\underline{\mathbf{X}}_{(1)}(\mathbf{C}_{:j} \odot \mathbf{B}_{:j})$. Note that the size of the tensor is reduced from $\underline{\mathbf{X}}_{(1)} \in \mathbb{R}^{I_1 \times I_2 I_3}$ to $\underline{\mathbf{Z}}_{(1)}^{(j)} \in \mathbb{R}^{|\Omega_{A_j}| \times |nnz(\mathbf{B}_{:j})| \times |nnz(\mathbf{C}_{:j})|}$ and the size of Khatri-Rao product is reduced from $(\mathbf{C}_{:j} \odot \mathbf{B}_{:j}) \in \mathbb{R}^{I_2 I_3 \times 1}$ to $(\mathbf{C}_{nnz(\mathbf{C}_{:j})j} \odot \mathbf{B}_{nnz(\mathbf{B}_{:j})j}) \in \mathbb{R}^{|nnz(\mathbf{B}_{:j})| |nnz(\mathbf{C}_{:j})| \times 1}$. As a result, the number of multiplications is reduced from $O(I_1 I_2 I_3)$ to $O(|\Omega_{A_j}| |nnz(\mathbf{B}_{:j})| |nnz(\mathbf{C}_{:j})|)$. It reduces both the space and time complexity largely if factor matrices are sparse enough. For example, if half of variables are zero in all matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$, the calculation cost of the gradient is reduced at ratio of $(1/2)^3$. The pseudo-code of variable selection is shown in Algorithm 12.¹

5.3.2 Sampling Variables with Weighted Distribution

Although variable selection is effective for reducing both space and time, it does not mean that the reduced size is sufficient for efficient usage of memory. Therefore, I control the size of $\underline{\mathbf{Z}}^{(j)}$ explicitly. That is, I choose a pre-determined number of variables. We adopt a lazy strategy for this end, that is, I update only a part of variables. This step is made after variable selection (Algorithm 12). Unlike typical coordinate descent approaches such as stochastic gradient descent or parallel coordinate descent, I choose variables according to their importance degrees. We measure the degree to which the approximation error is reduced in (5.1) by one updating of the variables, according to the way Hsieh *et al.* used [HD11]. We calculate the degree of importance D_{ij} of element $\mathbf{A}_{ij}$ as

$$D_{ij} = \nabla_{A_{ij}} s + \nabla_{A_{ij}}^2 * s^2, \quad (5.9)$$

where $s = |\mathbf{A}_{ij}^{(new)} - \mathbf{A}_{ij}^{(old)}|$, that is, the difference value between before and after one updating. Since s is the second term of (5.4), $\nabla_{A_{ij}}$ is the gradient in $\mathbf{A}_{ij}$ and $\nabla_{A_{ij}}^2$ is the second-order derivative, we have

$$D_{ij} = (\underline{\mathbf{X}}_{(1)}(\mathbf{C} \odot \mathbf{B}))_{ij} - (\mathbf{A}(\mathbf{B}^T \mathbf{B} * \mathbf{C}^T \mathbf{C}))_{ij}.$$

Note that index i is chosen from only Ω_{A_j} .

In practice, I use an upper-bound of the first term as

$$D_{ij} \leq \|\underline{\mathbf{Y}}_{i1:}\|_1 * \|\mathbf{C}_{:j}\|_1 - (\mathbf{A}(\mathbf{B}^T \mathbf{B} * \mathbf{C}^T \mathbf{C}))_{ij}. \quad (5.10)$$

In this calculation, (5.10) is already obtained in the stage of variable selection (see (5.7)), so that I do not need to calculate (5.10) again. The sampling algorithm is shown in Algorithm 13.

¹ In implementation, I keep only coordinates Ω_{A_j} , Ω_{B_j} and Ω_{C_j} and do not reconstruct tensor $\underline{\mathbf{Y}}$ explicitly.

Algorithm 11 NTF with Variable Selections

- 1: **Input:** A nonnegative tensor $\underline{\mathbf{X}}$; the number of factors J ; the number of variables K ;
 - 2: **Output:** Factor matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$;
 - 3: Initialize $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ for all mode n with random positive values; Normalize all columns of matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ by L1 norm;
 - 4: Initialize index sets Ω_{A_j} , Ω_{B_j} and Ω_{C_j} for $j = 1, 2, \dots, J$ (Algorithm 12);
 - 5: **repeat**
 - 6: **for** $j = 1$ **to** J **do**
 - 7: Select positive variables to have new Ω_{A_j} (Algorithm 12);
 - 8: Select K variables by importance sampling to reduce Ω_{A_j} (Algorithm 13) ;
 - 9: Construct a smaller tensor $\underline{\mathbf{Z}}^{(j)}$ with the index set Ω_{A_j} as (5.8);
 - 10: Solve NMF $\|\underline{\mathbf{Z}}_{(1)}^{(j)} - \mathbf{A}_{\Omega_{A_j}}(\mathbf{C}_{nnz(\mathbf{C}_{:j})} \odot \mathbf{B}_{nnz(\mathbf{B}_{:j})})\|_F^2$ for $\mathbf{A}_{\Omega_{A_j}j}$;
 - 11: **end for**
 - 12: Repeat Steps 6-11 for $\mathbf{B}$ and $\mathbf{C}$ instead of $\mathbf{A}$;
 - 13: **until** Converge criterion is met
-

Algorithm 12 Variable Selection for $\mathbf{A}_{:j}$

- 1: **Input:** A nonnegative tensor $\underline{\mathbf{X}}$; factor matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$;
 - 2: **Output:** Index set Ω_{A_j} ; Updated factor matrix $\mathbf{A}_{:j}$;
 - 3: $\Omega_{A_j} = \{\}$
 - 4: $\alpha = (\mathbf{B}^T \mathbf{B} * \mathbf{C}^T \mathbf{C})$
 - 5: $\beta = \|\underline{\mathbf{Y}}_{i_1:}\|_1 * \|\mathbf{C}_{:j}\|_1$
 - 6: $\delta = \mathbf{A}\alpha$
 - 7: **for** $i = 1$ **to** I_1 **do**
 - 8: **if** $(\mathbf{A}_{ij} * \alpha_{jj} + \beta_i - \delta_{ij}) \geq 0$ **then**
 - 9: $\Omega_{A_j} = \Omega_{A_j} \cup \{i\}$
 - 10: **else**
 - 11: $\mathbf{A}_{ij} = 0$
 - 12: **end if**
 - 13: **end for**
-

CHAPTER 5. VARIABLE SELECTION FOR EFFICIENT SPARSE
NONNEGATIVE TENSOR FACTORIZATION

Algorithm 13 Importance Sampling of Variables

- 1: **Input:** Index set Ω_{A_j} ; A factor matrix $\mathbf{A}$; The number K of sampling variables;
- 2: **Output:** Reduced index set Ω'_{A_j} ;
- 3: Compute the distribution of importance degree:

$$D = \|\underline{\mathbf{Y}}_{i_1::}\|_1 * \|\mathbf{C}_{:,j}\|_1 - (\mathbf{A}_{\Omega_{A_j}}(\mathbf{B}^T \mathbf{B} * \mathbf{C}^T \mathbf{C})).$$

- 4: Drawn K indices from Ω_{A_j} at random, without duplication, according to probability

$$D_i / \sum_{i \in \Omega_{A_j}} \|D_i\|_1.$$

5.3.3 Sparsity Constraint

We have discussed how effectively we can reduce the time and space when the factor matrices are sparse. Here, I turn my attention into a methodology taking a balance in the trade-off between the approximation accuracy and sparsity of factor matrices. Typically, such a solution is obtained by introducing additional sparse constraints into a tensor factorization problem such as

$$\|\underline{\mathbf{X}} - \underline{\mathbf{I}} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C}\|_F^2 + \pi(\|\mathbf{A}\|_1^2 + \|\mathbf{B}\|_1^2 + \|\mathbf{C}\|_1^2),$$

where π is a positive parameter. In the proposed algorithm, this formulation changes (5.7) to

$$\mathbf{A}_{ij} * \alpha_{jj} + (\underline{\mathbf{X}}_{(1)}(\mathbf{C} \odot \mathbf{B}))_{ij} - (\mathbf{A}(\mathbf{B}^T \mathbf{B} * \mathbf{C}^T \mathbf{C}))_{ij} \leq \alpha_{jj} \pi. \quad (5.11)$$

Hence, the larger value of π is, the smaller number of variables are chosen in variable selection. We use this condition instead of (5.7) in the following experiment.

5.3.4 Analysis of Algorithm

Sparsity of Factors

The proposed algorithm assumes that the resultant factors obtained by NTF algorithms are sparse. This assumption is not strong. Indeed, such a sparsity was reported by many works [KB09, SH05, PFS12, P⁺13, MHA08, LS99, LSC⁺09]. In addition, sparsity is often enhanced in NTF compared with NMF [SH05, CAH09].

Convergence Property

Convergence to a stationary point is guaranteed in the proposed algorithm. This is because my algorithm is one of block coordinate descent algorithms and the update rules for non-zero values are the same with the other update algorithms [CAH09, HD11, BV04]. Since the upper-bound (5.7) does not hurt the update accuracy and convergence property, the same quality of solutions is obtained as the case obtained without variable selection. In fact, if the proposed algorithm is carried out without importance sampling algorithm nor initialization (Step 4 in Algorithm 11), we have the same stationary point as HALS [CZPA09] reaches.

Initialization

The performance of column-wise algorithms, in general, depends on the initialization strongly. For example, if the last columns $\mathbf{a}_J$, $\mathbf{b}_J$ and $\mathbf{c}_J$ of $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$ are dense and have very large values such that $(\mathbf{a}_J \circ \mathbf{b}_J \circ \mathbf{c}_J)_{i_1 i_2 i_3} \geq \underline{\mathbf{X}}_{i_1 i_2 i_3}$ for all i_1, i_2 and i_3 then, all variables of the other columns $\mathbf{a}_j$, $\mathbf{b}_j$ and $\mathbf{c}_j$ for $j \neq J$ become zero in the stage of variable selection, because the third term of criterion (5.7) becomes very large. Therefore, I normalize all columns not so as to take a large value in every column in the initialization stage.

Parallelization

Unfortunately, the proposed algorithm as a whole is not suitable for parallelization unlike many others. This is because the proposed algorithm works column-wisely and columns affects to each other. Nevertheless all parts are easily parallelized, respectively. For example, variable selection and sampling of variables are parallelizable row-wisely. Therefore, the parallelization of a whole procedure is one of the future works.

Time Complexity

The standard NTF algorithms need $O(I_1 I_2 I_3)$ time in each column updating in the calculation of $(\underline{\mathbf{X}}_{(1)}(\mathbf{C} \odot \mathbf{B}))_{:j}$. In contrast, my algorithm needs only $O(I_1 |\text{nnz}(\mathbf{B}_{:j})| |\text{nnz}(\mathbf{C}_{:j})|)$ in the calculation of upper-bound (5.7). We can expect a large amount of reduction in space and time when $|\Omega_{A_j}|, |\text{nnz}(\mathbf{A}_{:j})| \ll I_1$, $|\Omega_{B_j}|, |\text{nnz}(\mathbf{B}_{:j})| \ll I_2$ and $|\Omega_{C_j}|, |\text{nnz}(\mathbf{C}_{:j})| \ll I_3$. As I stated, in many practical situations, one or two conditions would be satisfied and I confirm them in the next section.

Table 5.1: Datasets represented by 3-mode tensors

Dataset	Mode Description	Size	Sparsity
Synthetic-Fac	(Object,Object,Object)	$3000 \times 3000 \times 3000$	0.003%
Enron	(User,Word,Week)	$151 \times 10793 \times 52$	0.450%
DiggW	(Article,Keyword,Topic)	$44005 \times 13714 \times 51$	0.040%
DiggC	(User,Article,Comment)	$9583 \times 44005 \times 95325$	0.00002%
Beeradovocate	(User,Item,Word)	$5840 \times 66055 \times 263359$	0.002%

Space Complexity

Our algorithm needs additional $O((I_1 + I_2 + I_3)J)$ space for keeping $\mathbf{\Omega}_{A_j}$, $\mathbf{\Omega}_{B_j}$ and $\mathbf{\Omega}_{C_j}$. This amount is ignorable comparing with the space $O(I_1 I_2 I_3)$ necessary for the calculation of gradients in the standard NTF algorithms. In the proposed algorithm, I need $O(|\mathbf{\Omega}_{A_j}| |\text{nnz}(\mathbf{B}_{:j})| |\text{nnz}(\mathbf{C}_{:j})|)$ instead of $O(I_1 I_2 I_3)$ for each column. This is exactly smaller than that of standard NTF. Furthermore, with sampling of K variables, the space cost is limited to $O(K |\text{nnz}(\mathbf{B}_{:j})| |\text{nnz}(\mathbf{C}_{:j})|)$. Therefore, if I upper-bound the size of $|\mathbf{\Omega}_{A_j}|$ by K (Algorithm 13), the space complexity is reduced to $O(K |\text{nnz}(\mathbf{B}_{:j})| |\text{nnz}(\mathbf{C}_{:j})|)$.

5.4 Experiments

We evaluated the proposed algorithm on four real-life datasets and an artificial dataset. The data were represented by 3-mode tensors shown in Table 5.1 with the sparsity of the tensors. We conducted all experiments on Matlab with tensor toolbox [BK04].² On Enron dataset, I extracted the 2000-2001 year data and removed stop-words and deleted words which appeared less than 5 times in total. For the detail of other three real-life datasets, see [LSC⁺09, CV14].

We generated an artificial datasets assuming multi-relational data. This dataset consists of 30 non-overlapped clusters generated by a stochastic block model with link probability $p = 0.3$ [K⁺06]. We did not add noise in this dataset to confirm the fundamental effectiveness of the proposed algorithm in an ideal case.

²<http://www.sandia.gov/~tgkolda/TensorToolbox/index-2.6.html>

5.4.1 Evaluation Measure

We evaluated the approximation error by Normalized Residual Value (NRV) defined as

$$\text{Approximation error} = \frac{\| \underline{\mathbf{X}}_{(1)} - \mathbf{A}(\mathbf{C} \odot \mathbf{B})^T \|_F^2}{\| \underline{\mathbf{X}} \|_F^2}. \quad (5.12)$$

We measured the CPU time consumed for updating of factor matrices as the time cost. On the other hand, with respect to space consumption, since I am interested in the size of the intermediate matrices, I measured the ratio $|\underline{\mathbf{Z}}^{(j)}|/|\underline{\mathbf{X}}|$ on the average:

$$\text{Relative Space Consumption} = \frac{\sum_{nj} |\underline{\mathbf{Z}}^{(j)}| / I_1 I_2 I_3}{JN}, \quad (5.13)$$

where $n = 1, 2, 3$, $N = 3$ and $j = 1, 2, \dots, J$. Note that the size of the tensor $\underline{\mathbf{Z}}^{(j)}$ changes according with the mode which I update. For example, for $\underline{\mathbf{Z}}^{(j)} \in \mathbb{R}^{|\Omega_{A_j}| \times |\text{nnz}(\mathbf{B}_{:j})| \times |\text{nnz}(\mathbf{C}_{:j})|}$ for $\mathbf{A}_{:j}$ and $\underline{\mathbf{Z}}^{(j)} \in \mathbb{R}^{|\text{nnz}(\mathbf{A}_{:j})| \times |\Omega_{B_j}| \times |\text{nnz}(\mathbf{C}_{:j})|}$ for $\mathbf{B}_{:j}$. Therefore, I calculated the size of tensor $\underline{\mathbf{Z}}^{(j)}$ before updating for each column vector and averaged the results.

For evaluation of sparsity, I evaluated column-wise sparsity for each column $\mathbf{A}_{:j}$, $\mathbf{B}_{:j}$ and $\mathbf{C}_{:j}$ and calculated average of them as

$$\text{Average Sparsity} = \frac{\sum_j \left(\frac{|\text{nnz}(\mathbf{A}_{:j})|}{I_1} + \frac{|\text{nnz}(\mathbf{B}_{:j})|}{I_2} + \frac{|\text{nnz}(\mathbf{C}_{:j})|}{I_3} \right)}{JN}.$$

5.4.2 Methods and Parameters

We compared the following four NTF algorithms including two variants of the proposed algorithm:

- Hierarchical ALS (**HALS**): Column-wise update algorithm [CAH09].³
- Greedy Coordinate Descent (**GCD**): Fastest algorithm for solving NMF problem [HD11].⁴
- NTF + variable selection (**Proposal-VS**): The proposed algorithm with variable selection only.

³ http://www.cc.gatech.edu/~hpark/software/ntf_package.zip

⁴ <http://www.cs.utexas.edu/~cjhsieh/nmf>

- NTF + variable selection + importance sampling (**Proposal-VSS**): The proposed algorithm with variable selection and importance sampling. The number of sampled variables was set to 10% of the corresponding mode.

We implemented DFacto and GigaTensor too on matlab, but they were omitted from comparison because they were too slow in the implementation using the tensor toolbox. This is likely due to the language and architecture or both.⁵

We fixed the number of iterations to 30 at which all algorithms converged in my experiments. We repeated 10 trials with different initializations and calculated the average values of the results. In all algorithms, I set the parameter for sparsity constraint as $\pi = 0.1$.

5.4.3 Results

The results on the artificial dataset are shown in Fig. 5.2. As I expected, the sparsity of the factor matrices obtained in the middle of process were very high even in the early stage of iterations. After a few times of iterations, the factor matrices reached at sparsity of 10% (90% zero variables), and reached at the theoretical value (1%) in 5 to 15 iterations (Fig.5.2(a)). The proposed algorithms converge to that sparsity faster than the other two conventional algorithms. At fifth iteration, the sizes of intermediate matrices were already one hundred smaller than the original size (Fig.5.2(c)). Even the approximation accuracy of the proposed algorithms were better than two standard algorithms (Fig.5.2(d)), although the performance depends strongly on the initialization (random positive values) regardless of algorithms.

The results on four real-life datasets are shown in Fig. 5.3 and Fig. 5.4. The sparsity of resultant factor matrices ranges from 10^{-3} to 0.5 (Fig. 5.3(a)). Among four algorithms, **GCD** and **Proposal-VS** attained better sparsity in three data sets except **Enron**.

The proposed algorithms were about three times faster than the two standard NTF algorithms in updating (Fig. 5.3(b)).⁶ On **Enron** dataset, the importance sampling **Proposal-VSS** was faster than **Proposal-VS** unlike

⁵ DFacto is implemented on C++ and openMP in [CV14] but, on MATLAB, the reshaping of their intermediate matrix needs large computational time. GigaTensor is based on MAP/Reduce environment [KPHF12]. On a single machine, it is known that the GigaTensor is relatively slow [CV14].

⁶In tensor toolbox, standard NTF algorithms use **mttkrp** function, that is, a fast Khatri-Rao calculation for sparse tensor format. The **mttkrp** avoids direct calculations of Khatri-Rao product and matricization by utilizing sparsity not of factors but of tensor itself [KB09].

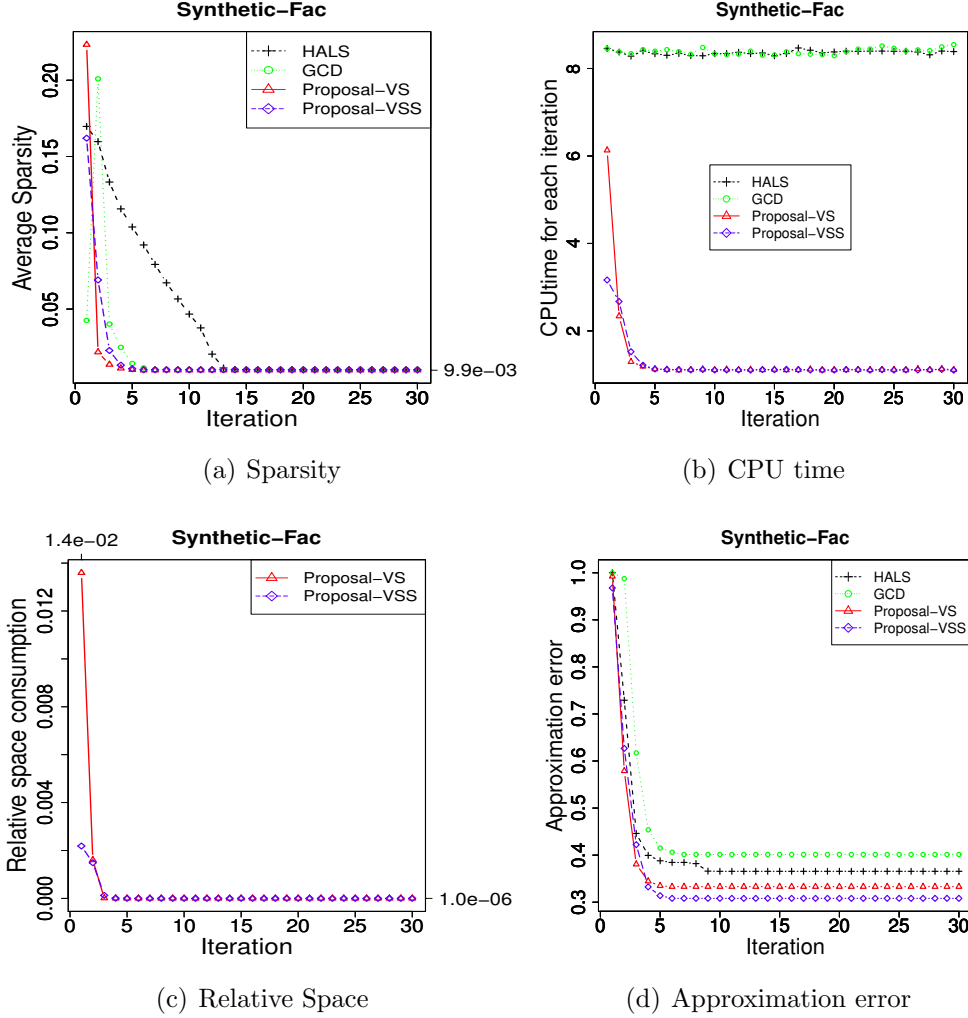


Figure 5.2: Results on an artificial dataset.

on the other dataset. This is natural because the sparsity of factors are higher than the number K of sampled variables (10% variables are forcibly chosen to be updated). On the other hand, on **DiggC** dataset, the proposed two algorithms showed the same results. This is interpretable from the fact that the sparsity of factors became less than K after first variable selection (Step 4 in Algorithm 11). These observations imply that the importance sampling works well even if the obtained factors are relatively dense.

The size of intermediate matrices was reduced by a factor of 100 to 500 after 5 to 20 iterations in the proposed algorithms (Fig. 5.4(a)). Note that the size of tensor is not reduced in the standard algorithms. Especially, on

Table 5.2: CPUtime with 30 iterations ($J = 30$)

Dataset	Method			
	HALS	GCD	Proposal -VS	Proposal -VSS
Synthetic-Fac	250.9	251.8	39.9	37.5
Enron	188.4	154.6	85.6	66.06
DiggW	107.6	124.4	27.1	29.9
DiggC	174.0	178.5	29.5	31.4
Beeradovocate	6.6e+03	7.6e+03	2.5e+03	2.5e+03

Enron dataset, the importance sampling adopted in **Proposal-VSS** worked well. In spite of such a large reduction of time and space, the approximation accuracy of the proposed algorithms was comparable with standard NTF algorithms. Table 5.2 shows the total CPU time consumed for 30 iterations. The proposed two algorithms were three to five times faster than the two standard NTF algorithms in total.

5.5 Conclusion and Discussion

We have proposed variable selection for obtaining a fast and memory efficient NTF algorithm. We succeeded to solve the intermediate data explosion problem of concern by deriving an upper-bound of the amount of contribution obtained by one update, and with is by evaluating which variable is unnecessary to be updated. In addition, I introduced a lazy coordinate descent algorithm to forcibly choose a fixed number of variables according to their degrees of importance. The proposed two algorithms factorized tensor three to five times faster than two traditional NTF algorithms on four real-life datasets. The necessary memory was also reduce by up to a factor of 500.

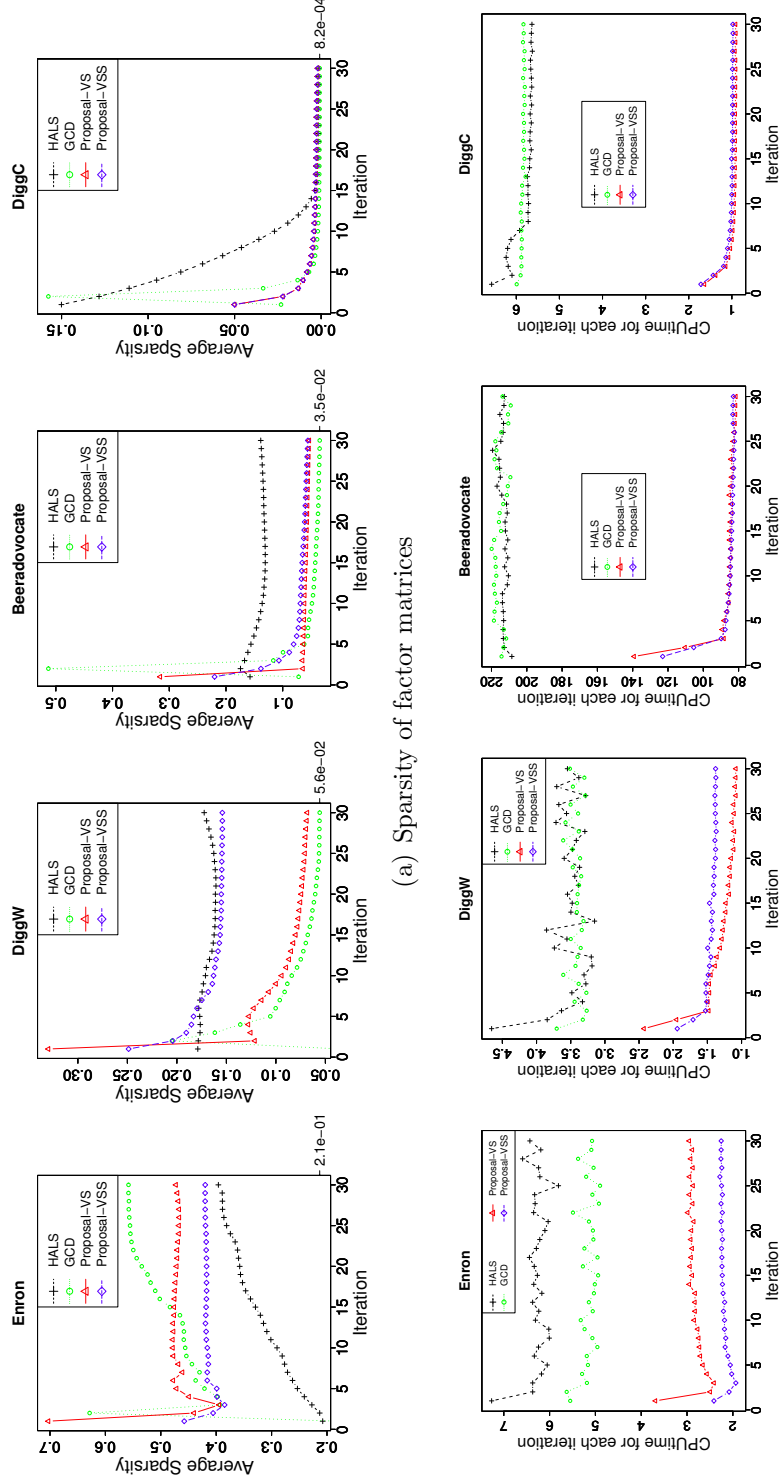


Figure 5.3: Sparsity and CPU time per iteration on four real-life datasets.

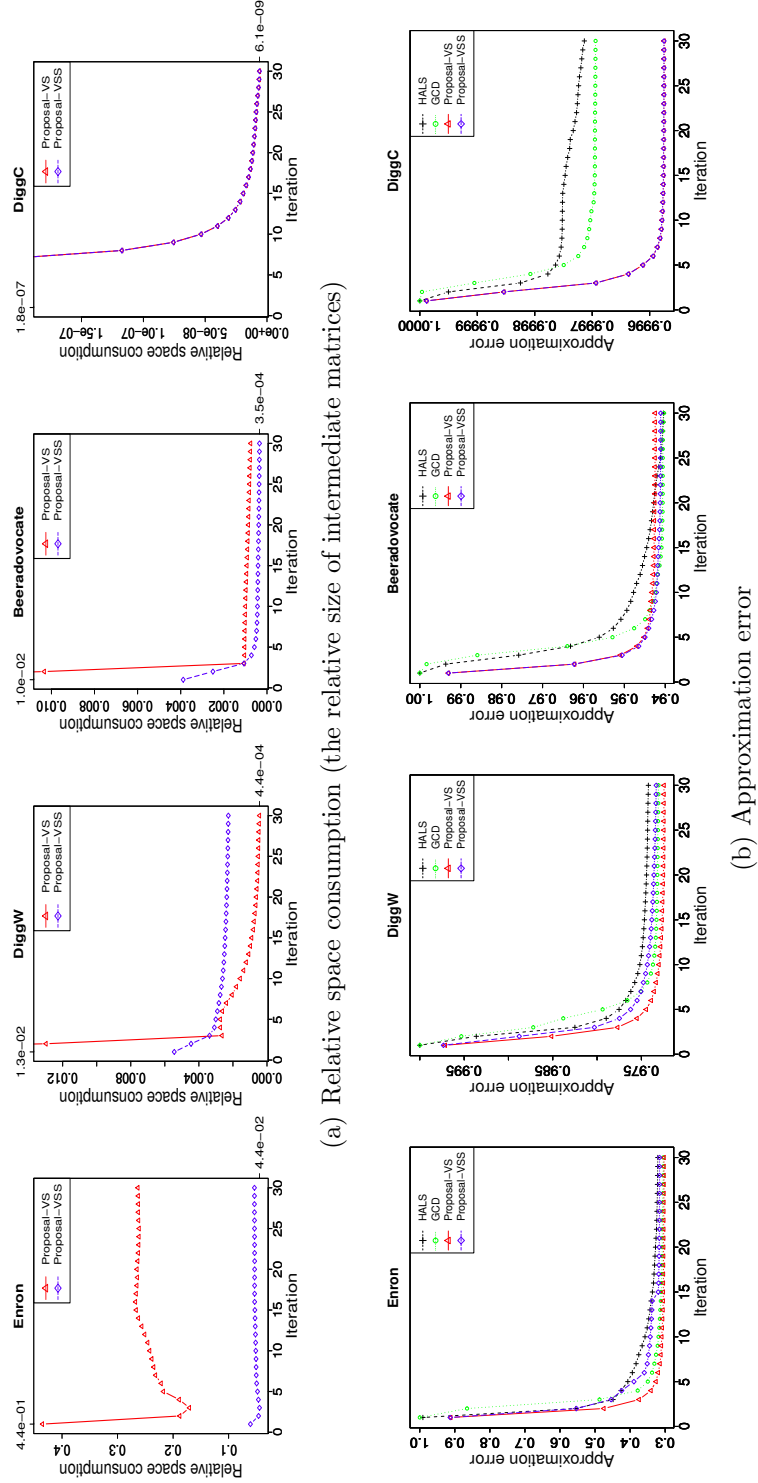


Figure 5.4: Relative space consumption and approximation error on four real-life datasets.

Chapter 6

Conclusion

6.1 Contributions

We have proposed two efficient algorithms for Orthogonal NMF which update matrices column-wisely [KKT16, KTK14] in Chapter 4. The one is Hierarchical Alternating Least Squares algorithm (HALS) for Frobenius norm which minimize Frobenius approximation error with orthogonal constraints. This method performed the fastest computational time until the convergence with the best approximation accuracy among traditional methods with acceptable degrees of orthogonality. The other one is scalar Block Coordinate Descent method for Bregman divergence which minimizes Bregman divergence approximation error with orthogonal constraints. This method is the first method to solve Orthogonal NMF with Bregman divergence. In addition, I prove the equivalence between the problem of Orthogonal NMF with Bregman divergence and Bregman Hard Clustering [BMDG05], that is, a wide class of clustering methods. This connects Orthogonal NMF to the other clustering methods such as K-means clustering and Information Theoretic Co-Clustering [DMM03]. The effectiveness of this method has been also confirmed on experiments.

In Chapter 5, I have proposed two efficient algorithms for NTF. The one is variable selection for NTF which predict unnecessary elements which becomes zero after updating and reduce calculation related to those. This method is related to feature elimination methods such as SAFE [GVR10] on Lasso problems though the theoretical backgrounds on these methods are different. This is the first trial to use such methods on alternating update procedures. This reduce the computational time and memory consumption to calculate intermediate matrices in gradient calculations. The other one is importance sampling for NTF. This method can be considered as one of

lazy coordinate descents, which does not update all elements necessarily in each iteration. The variable selection method does not guarantee to reduce memory consumption enough to fit the memory used in the machine, this importance sampling can reduce exactly how users want. The main advantages of these methods are the guarantee of approximation accuracy like safe screening rule. The proposed algorithm will achieve the same approximation accuracy.

6.2 Discussions and Future Work

Column-wise Update Algorithms for Orthogonal Nonnegative Matrix Factorization

Column-wise update algorithms for orthogonal NMF are available not only for the discussed problem in Chapter 4 but the other problems such as NTF problem with orthogonal constraints like stated in Chapter 5. In this sense, the proposed are methods can be used for clustering of tensor data or high-order co-clustering [PSB13]. This flexibility and a wide range of usage are strong advantages against the other clustering methods. The problem on these methods is still the balancing the degree of orthogonality and approximation accuracy. Since most of all orthogonal NMF algorithms do not satisfy the strict orthogonality $\mathbf{U}^T \mathbf{U} = \mathbf{I}$, orthogonal NMF needs to take a trade-off between them. The problem is how orthogonality should be achieved. Our methods, HALS-ONMF and sBCD ONMF weight these automatically unlike most of all the other methods depends on user's decision and user defined-parameter. This can be considered as one of advantages but also as one of disadvantages. For example, on Table 4.4 the proposed algorithm did not achieve high degree of orthogonality compared to the other algorithms. On the other hand, the proposed algorithm achieved better approximation accuracy. In this observation, the proposed algorithms consider the approximation accuracy more importantly than the degree of orthogonality. On specific problems which consider the orthogonality is the most important. Thus, in this sense, we need to choose the methods carefully.

The convergence to a local minimum of the proposed methods are guaranteed, however, the superiority to the others w.r.t. convergence speed or the goodness of approximation are not proved and there are based only on empirical observations. Theoretical analysis of convergence speeds is one of future works related to this topic. However, to the best of my knowledge, such analysis has not been done yet even with a standard NMF algorithms.

Variable Selection for Efficient Nonnegative Tensor Factorization

To the best of my knowledge, this is the first trial to use feature elimination like methods on alternating update procedures and to reduce the complexity of gradient calculation itself. The experimental results show if the resultant matrices are sparse, the proposed methods work well. However, if it is not the effectiveness and improvements would be limited. Unfortunately, the effectiveness or improvements are unpredictable since the sparsity of the resultant matrices are also unpredictable. The resultant matrices possibly become dense even if the decomposed tensor is sparse. This problem also lies on feature eliminations methods on Lasso [GVR10]. This is the main concern of this work and future work.

There is a method which greedily eliminates features with damaging accuracy such as the strong rule in regression literature [TBF⁺12]. The proposed method uses upper-bound to keep approximation accuracy, however, it is possibly to accelerate the method with sacrificing that. This is also one of approaches remaining to consider.

In analysis of tensor data, users often demand sparse representation to understand structures behind data. Indeed, many sparse learning methods and sparse factorization have been proposed so far [MMH08, MHA08, Hoy04, PFS12, MBPS10]. In this study, I use a simple L1 regularization term, however, there are more sophisticated sparse constraints [Hoy04]. Thus, aggregation of this constraint and derive elimination rules is also one of future works.

The importance sampling has a trade-off between convergence speed and computational time since it chooses the small number of elements to be updated. However, in most cases, the number of positive elements in a factor (a column vector) is still small. Therefore, in the experimental section I could not observe the trade-off. This convergence analysis is also one of future works.

In summary, I have efficient algorithms for orthogonal NMF and NTF. Column-wise update algorithms for orthogonal NMF converge faster than the others with a satisfactory degree of orthogonality. Variable selection method and importance sampling method for NTF also perform faster computation and smaller memory consumption than the other state-of-the-arts.

Bibliography

- [ADK09] Evrim Acar, Daniel M Dunlavy, and Tamara G Kolda. Link prediction on evolving data using matrix and tensor factorizations. In *IEEE International Conference on Data Mining Workshops, 2009.*, pages 262–269. IEEE, 2009.
- [AGKM12] Sanjeev Arora, Rong Ge, Ravindran Kannan, and Ankur Moitra. Computing a nonnegative matrix factorization—provably. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 145–162. ACM, 2012.
- [AV07] David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007.
- [AY09] Evrim Acar and Bülent Yener. Unsupervised multiway data analysis: A literature survey. *IEEE Transactions on Knowledge and Data Engineering*, 21(1):6–20, 2009.
- [BAK99] Rasmus Bro, Claus A Andersson, and Henk AL Kiers. Parafac2-part ii. modeling chromatographic data with retention time shifts. *Journal of Chemometrics*, 13(3-4):295–309, 1999.
- [BBB08] Brett W Bader, Michael W Berry, and Murray Browne. Discussion tracking in enron email using parafac. In *Survey of Text Mining II*, pages 147–163. Springer, 2008.
- [BDJ97] Rasmus Bro and Sijmen De Jong. A fast non-negativity-constrained least squares algorithm. *Journal of chemometrics*, 11(5):393–401, 1997.

BIBLIOGRAPHY

- [BE08] Ori Bryt and Michael Elad. Compression of facial images using the k-svd algorithm. *Journal of Visual Communication and Image Representation*, 19(4):270–282, 2008.
- [BG08] Christos Boutsidis and Efstratios Gallopoulos. Svd based initialization: A head start for nonnegative matrix factorization. *Pattern Recognition*, 41(4):1350–1362, 2008.
- [BGV08] Michael Biggs, Ali Ghodsi, and Stephen Vavasis. Nonnegative matrix factorization via rank-one downdate. In *Proceedings of the 25th international conference on Machine learning*, pages 64–71. ACM, 2008.
- [BGW05] Arindam Banerjee, Xin Guo, and Hui Wang. On the optimality of conditional expectation as a bregman predictor. *IEEE Transactions on Information Theory*, 51(7):2664–2669, 2005.
- [BK04] Brett W Bader and Tamara G Kolda. Matlab tensor classes for fast algorithm prototyping. *Sandia Report SAND2004-5187*, 2004.
- [BK10] Emmanouil Benetos and Constantine Kotropoulos. Non-negative tensor factorization applied to music genre classification. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(8):1955–1967, 2010.
- [BMDG05] Arindam Banerjee, Srujana Merugu, Inderjit S Dhillon, and Joydeep Ghosh. Clustering with bregman divergences. *The Journal of Machine Learning Research*, 6:1705–1749, 2005.
- [BN01] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in Neural Information Processing Systems*, volume 14, pages 585–591, 2001.
- [BV04] Stephen Poythress Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.
- [CAH09] Andrzej Cichocki and PHAN Anh-Huy. Fast local algorithms for large scale nonnegative matrix and tensor factorizations. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 92(3):708–721, 2009.

- [CBKA07] Peter A Chew, Brett W Bader, Tamara G Kolda, and Ahmed Abdelali. Cross-language information retrieval using parafac2. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 143–152. ACM, 2007.
- [CHHH11] Deng Cai, Xiaofei He, Jiawei Han, and Thomas S Huang. Graph regularized nonnegative matrix factorization for data representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(8):1548–1560, 2011.
- [Cic14] Andrzej Cichocki. Era of big data processing: A new approach via tensor networks and tensor decompositions. *arXiv preprint arXiv:1403.2048*, 2014.
- [CPC08] Andrzej Cichocki, Anh Huy Phan, and Cesar Caiafa. Flexible hals algorithms for sparse non-negative matrix/tensor factorization. In *Machine Learning for Signal Processing, 2008. MLSP 2008. IEEE Workshop on*, pages 73–78. IEEE, 2008.
- [CPZ⁺12a] Fengyu Cong, Anh Phan, Qibin Zhao, Qiang Wu, Tapani Ristaniemi, and Andrzej Cichocki. Feature extraction by non-negative tucker decomposition from eeg data including testing and training observations. In *Neural Information Processing*, pages 166–173. Springer, 2012.
- [CPZ⁺12b] Fengyu Cong, Anh Huy Phan, Qibin Zhao, Asoke K Nandi, Vinoo Alluri, Petri Toivainen, Hanna Poikonen, Minna Huotilainen, Andrzej Cichocki, and Tapani Ristaniemi. Analysis of ongoing eeg elicited by natural music stimuli using nonnegative tensor factorization. In *Proceedings of the 20th European Signal Processing Conference*, pages 494–498. IEEE, 2012.
- [CSS⁺07] Bin Cao, Dou Shen, Jian-Tao Sun, Xuanhui Wang, Qiang Yang, and Zheng Chen. Detect and track latent factors with online nonnegative matrix factorization. In *IJCAI*, volume 7, pages 2689–2694, 2007.
- [CV14] Joon Hee Choi and S Vishwanathan. Dfacto: Distributed factorization of tensors. In *Advances in Neural Information Processing Systems*, pages 1296–1304, 2014.
- [CWH09] Deng Cai, Xuanhui Wang, and Xiaofei He. Probabilistic dyadic data analysis with local and global consistency. In

BIBLIOGRAPHY

- Proceedings of the 26th Annual International Conference on Machine Learning*, pages 105–112. ACM, 2009.
- [CWZ09] Gang Chen, Fei Wang, and Changshui Zhang. Collaborative filtering using orthogonal nonnegative matrix tri-factorization. *Information Processing & Management*, 45(3):368–379, 2009.
- [CZA06] Andrzej Cichocki, Rafal Zdunek, and Shun-ichi Amari. Csiszars divergences for non-negative matrix factorization: Family of new algorithms. In *International Conference on Independent Component Analysis and Signal Separation*, pages 32–39. Springer, 2006.
- [CZPA09] Andrzej Cichocki, Rafal Zdunek, Anh Huy Phan, and Shun-ichi Amari. *Nonnegative matrix and tensor factorizations: applications to exploratory multi-way data analysis and blind source separation*. John Wiley & Sons, 2009.
- [CZS⁺07] Yun Chi, Shenghuo Zhu, Xiaodan Song, Junichi Tatemura, and Belle L Tseng. Structural and temporal analysis of the blogosphere through community factorization. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 163–172. ACM, 2007.
- [DHS05] Chris HQ Ding, Xiaofeng He, and Horst D Simon. On the equivalence of nonnegative matrix factorization and spectral clustering. In *SDM*, volume 5, pages 606–610. SIAM, 2005.
- [DLDMV00a] Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle. A multilinear singular value decomposition. *SIAM journal on Matrix Analysis and Applications*, 21(4):1253–1278, 2000.
- [DLDMV00b] Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle. On the best rank-1 and rank-($r_1, r_2, \dots, r_n$) approximation of higher-order tensors. *SIAM Journal on Matrix Analysis and Applications*, 21(4):1324–1342, 2000.
- [DLJ10] Chris HQ Ding, Tao Li, and Michael I Jordan. Convex and semi-nonnegative matrix factorizations. *IEEE transactions on pattern analysis and machine intelligence*, 32(1):45–55, 2010.
- [DLP08] Chris Ding, Tao Li, and Wei Peng. On the equivalence between non-negative matrix factorization and probabilistic latent semantic indexing. *Computational Statistics & Data Analysis*, 52(8):3913–3927, 2008.

- [DLPP06] Chris Ding, Tao Li, Wei Peng, and Haesun Park. Orthogonal nonnegative matrix t-factorizations for clustering. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data mining*, pages 126–135. ACM, 2006.
- [DLS12] Liang Du, Xuan Li, and Yi-Dong Shen. Robust nonnegative matrix factorization via half-quadratic minimization. In *2012 IEEE 12th International Conference on Data Mining*, pages 201–210. IEEE, 2012.
- [DMM03] Inderjit S Dhillon, Subramanyam Mallela, and Dharmendra S Modha. Information-theoretic co-clustering. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 89–98. ACM, 2003.
- [DS03] David Donoho and Victoria Stodden. When does non-negative matrix factorization give a correct decomposition into parts? In *Advances in neural information processing systems*, page None, 2003.
- [EK04] Julian Eggert and Edgar Korner. Sparse coding and nmf. In *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on*, volume 4, pages 2529–2533. IEEE, 2004.
- [EM13] Dóra Erdos and Pauli Miettinen. Walk’n’merge: A scalable algorithm for boolean tensor factorization. In *2013 IEEE 13th International Conference on Data Mining*, pages 1037–1042. IEEE, 2013.
- [FBD09] Cédric Févotte, Nancy Bertin, and Jean-Louis Durrieu. Non-negative matrix factorization with the itakura-saito divergence: With application to music analysis. *Neural computation*, 21(3):793–830, 2009.
- [FCC05] Derry FitzGerald, Matt Cranitch, and Eugene Coyle. Non-negative tensor factorisation for sound source separation. 2005.
- [FI11] Cédric Févotte and Jérôme Idier. Algorithms for nonnegative matrix factorization with the β -divergence. *Neural Computation*, 23(9):2421–2456, 2011.

BIBLIOGRAPHY

- [FP14] Xiaomin Fang and Rong Pan. Fast dtt: a near linear algorithm for decomposing a tensor into factor tensors. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 967–976. ACM, 2014.
- [GG05] Eric Gaussier and Cyril Goutte. Relation between pls and nmf and implications. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 601–602. ACM, 2005.
- [GK15] Nicolas Gillis and Abhishek Kumar. Exact and heuristic algorithms for semi-nonnegative matrix factorization. *SIAM Journal on Matrix Analysis and Applications*, 36(4):1404–1424, 2015.
- [GL14] Nicolas Gillis and Robert Luce. Robust near-separable non-negative matrix factorization using linear optimization. *Journal of Machine Learning Research*, 15(1):1249–1280, 2014.
- [GNCH15] Hongchang Gao, Feiping Nie, Weidong Cai, and Heng Huang. Robust capped norm nonnegative matrix factorization: Capped norm nmf. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 871–880. ACM, 2015.
- [GSV02] David Guillaumet, Bernt Schiele, and Jordi Vitria. Analyzing non-negative matrix factorization for image classification. In *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, volume 2, pages 116–119. IEEE, 2002.
- [GTLST12] Naiyang Guan, Dacheng Tao, Zhigang Luo, and John Shawe-Taylor. Mahnmf: Manhattan non-negative matrix factorization. *arXiv preprint arXiv:1207.3438*, 2012.
- [GTLY11] Naiyang Guan, Dacheng Tao, Zhigang Luo, and Bo Yuan. Manifold regularized discriminative nonnegative matrix factorization with fast gradient descent. *IEEE Transactions on Image Processing*, 20(7):2030–2048, 2011.
- [GTLY12] Naiyang Guan, Dacheng Tao, Zhigang Luo, and Bo Yuan. Online nonnegative matrix factorization with robust stochastic approximation. *IEEE Transactions on Neural Networks and Learning Systems*, 23(7):1087–1099, 2012.

- [GV14] Nicolas Gillis and Stephen A Vavasis. Fast and robust recursive algorithms for separable nonnegative matrix factorization. *IEEE transactions on pattern analysis and machine intelligence*, 36(4):698–714, 2014.
- [GVR10] Laurent El Ghaoui, Vivian Viallon, and Tarek Rabbani. Safe feature elimination for the lasso and sparse supervised learning problems. *arXiv preprint arXiv:1009.4219*, 2010.
- [GVS03] David Guillaumet, Jordi Vitria, and Bernt Schiele. Introducing a weighted non-negative matrix factorization for image classification. *Pattern Recognition Letters*, 24(14):2447–2454, 2003.
- [GZ09] Quanquan Gu and Jie Zhou. Co-clustering on manifolds. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 359–368. ACM, 2009.
- [H⁺14] Joyce C Ho et al. Marble: high-throughput phenotyping from electronic health records via sparse nonnegative tensor factorization. In *ACM SIGKDD*, pages 115–124, 2014.
- [Har78] Richard A Harshman. Models for analysis of asymmetrical relationships among n objects or stimuli. In *First Joint Meeting of the Psychometric Society and the Society for Mathematical Psychology, McMaster University, Hamilton, Ontario*, volume 5, 1978.
- [HD11] Cho-Jui Hsieh and Inderjit S Dhillon. Fast coordinate descent methods with variable selection for non-negative matrix factorization. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data mining*, pages 1064–1072. ACM, 2011.
- [HMTK15] Kohei Hayashi, Takanori Maehara, Masashi Toyoda, and Ken-ichi Kawarabayashi. Real-time top-r topic detection on twitter with topic hijack filtering. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 417–426. ACM, 2015.
- [Hoy02] Patrik O Hoyer. Non-negative sparse coding. In *Proceedings of the 2002 12th IEEE Workshop on Neural Networks for Signal Processing, 2002.*, pages 557–565. IEEE, 2002.

BIBLIOGRAPHY

- [Hoy04] Patrik O Hoyer. Non-negative matrix factorization with sparseness constraints. *Journal of machine learning research*, 5(Nov):1457–1469, 2004.
- [HPS05] Tamir Hazan, Simon Polak, and Amnon Shashua. Sparse image coding using a 3d non-negative tensor factorization. In *Tenth IEEE International Conference on Computer Vision Volume 1*, volume 1, pages 50–57. IEEE, 2005.
- [JCW⁺14] Meng Jiang, Peng Cui, Fei Wang, Xinran Xu, Wenwu Zhu, and Shiqiang Yang. Fema: flexible evolutionary multi-faceted analysis for dynamic behavioral pattern discovery. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1186–1195. ACM, 2014.
- [JFB⁺11] Rajesh Jaiswal, Derry FitzGerald, Dan Barry, Eugene Coyle, and Scott Rickard. Clustering nmf basis functions using shifted nmf for monaural sound source separation. In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 245–248. IEEE, 2011.
- [JFCR11] Rajesh Jaiswal, Derry Fitzgerald, Eugene Coyle, and Scott Rickard. Shifted nmf using an efficient constant-q transform for monaural sound source separation. In *22nd IET Irish Signals and Systems Conference*, pages 23–24, 2011.
- [JG10] Andreas GK Janecek and Wilfried N Gansterer. Utilizing non-negative matrix factorization for email classification problems. *Text mining: Applications and theory*, pages 57–80, 2010.
- [JPKF15] Inah Jeon, Evangelos E Papalexakis, U Kang, and Christos Faloutsos. Hatent2: Billion-scale tensor decompositions. In *2015 IEEE 31st International Conference on Data Engineering*, pages 1047–1058. IEEE, 2015.
- [K⁺06] Charles Kemp et al. Learning systems of concepts with an infinite relational model. In *AAAI*, page 5, 2006.
- [KB09] Tamara G Kolda and Brett W Bader. Tensor decompositions and applications. *SIAM review*, 51(3):455–500, 2009.

- [KC07] Yong-Deok Kim and Seungjin Choi. Nonnegative tucker decomposition. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007.
- [KDH11] Deguang Kong, Chris Ding, and Heng Huang. Robust non-negative matrix factorization using l21-norm. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 673–682. ACM, 2011.
- [KHP14] Jingu Kim, Yunlong He, and Haesun Park. Algorithms for nonnegative matrix and tensor factorizations: A unified view based on block coordinate descent framework. *Journal of Global Optimization*, 58(2):285–319, 2014.
- [KK15] Keigo Kimura and Mineichi Kudo. Variable selection for efficient nonnegative tensor factorization. In *2015 IEEE International Conference on Data Mining*, pages 805–810. IEEE, 2015.
- [KKT16] Keigo Kimura, Mineichi Kudo, and Yuzuru Tanaka. A column-wise update algorithm for nonnegative matrix factorization in bregman divergence with an orthogonal constraint. *Machine Learning*, 103(2):285–306, 2016.
- [KP07] Hyunsoo Kim and Haesun Park. Sparse non-negative matrix factorizations via alternating non-negativity-constrained least squares for microarray data analysis. *Bioinformatics*, 23(12):1495–1502, 2007.
- [KP08] Jingu Kim and Haesun Park. Toward faster nonnegative matrix factorization: A new algorithm and comparisons. In *Proceedings of the 8th IEEE International Conference on Data Mining*, pages 353–362. IEEE, 2008.
- [KP11] Jingu Kim and Haesun Park. Fast nonnegative matrix factorization: An active-set-like method and comparisons. *SIAM Journal on Scientific Computing*, 33(6):3261–3281, 2011.
- [KPHF12] U Kang, Evangelos Papalexakis, Abhay Harpale, and Christos Faloutsos. Gigatensor: scaling tensor analysis up by 100 times-algorithms and discoveries. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 316–324. ACM, 2012.

BIBLIOGRAPHY

- [Kru77] Joseph B Kruskal. Three-way arrays: rank and uniqueness of trilinear decompositions, with application to arithmetic complexity and statistics. *Linear algebra and its applications*, 18(2):95–138, 1977.
- [KS08] Tamara G Kolda and Jimeng Sun. Scalable tensor decompositions for multi-aspect data mining. In *Eighth IEEE International Conference on Data Mining, 2008.*, pages 363–372. IEEE, 2008.
- [KSK13] Abhishek Kumar, Vikas Sindhwani, and Prabhanjan Kam-badur. Fast conical hull algorithms for near-separable non-negative matrix factorization. In *ICML (1)*, pages 231–239, 2013.
- [KTBB99] Henk AL Kiers, Jos MF Ten Berge, and Rasmus Bro. Parafac2-part i. a direct fitting algorithm for the parafac2 model. *Journal of Chemometrics*, 13(3-4):275–294, 1999.
- [KTK14] Keigo Kimura, Yuzuru Tanaka, and Mineichi Kudo. A fast hierarchical alternating least squares algorithm for orthogonal nonnegative matrix factorization. In *ACML*, 2014.
- [KWTB10] Kristian Kersting, Mirwaes Wahabzada, Christian Thureau, and Christian Bauckhage. Hierarchical convex nmf for clustering massive data. *ACML*, 10:253–268, 2010.
- [LHCS16] Xinsheng Li, Shengyu Huang, K Selçuk Candan, and Maria Luisa Sapino. 2pcp: Two-phase cp decomposition for billion-scale dense tensors. In *2016 IEEE 32nd International Conference on Data Engineering.*, pages 835–846. IEEE, 2016.
- [Lin07] Chih-Jen Lin. Projected gradient methods for nonnegative matrix factorization. *Neural Computation*, 19(10):2756–2779, 2007.
- [LKCC07] Hyekyoung Lee, Yong-Deok Kim, Andrzej Cichocki, and Seungjin Choi. Nonnegative tensor factorization for continuous eeg classification. *International journal of neural systems*, 17(04):305–317, 2007.
- [LLP12] Liangda Li, Guy Lebanon, and Haesun Park. Fast bregrman divergence nmf using taylor expansion and coordinate descent.

- In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data mining*, pages 307–315. ACM, 2012.
- [LPV08] Haiping Lu, Konstantinos N Plataniotis, and Anastasios N Venetsanopoulos. MPCA: Multilinear principal component analysis of tensor objects. *IEEE Transactions on Neural Networks*, 19(1):18–39, 2008.
- [LPV11] Haiping Lu, Konstantinos N Plataniotis, and Anastasios N Venetsanopoulos. A survey of multilinear subspace learning for tensor data. *Pattern Recognition*, 44(7):1540–1551, 2011.
- [LS99] Daniel D Lee and H Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.
- [LS00] Daniel D Lee and H Sebastian Seung. Algorithms for non-negative matrix factorization. In *Advances in Neural Information Processing Systems 13*, pages 556–562, 2000.
- [LSC⁺09] Yu-Ru Lin, Jimeng Sun, Paul Castro, Ravi Konuru, Hari Sundaram, and Aisling Kelliher. Metafac: community discovery via relational hypergraph factorization. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 527–536. ACM, 2009.
- [LWGH13] Jialu Liu, Chi Wang, Jing Gao, and Jiawei Han. Multi-view clustering via joint nonnegative matrix factorization. In *Proc. of SDM*, volume 13, pages 252–260. SIAM, 2013.
- [LWP10] Zhao Li, Xindong Wu, and Hong Peng. Nonnegative matrix factorization on orthogonal subspace. *Pattern Recognition Letters*, 31(9):905–911, 2010.
- [LYF⁺10] Chao Liu, Hung-chih Yang, Jinliang Fan, Li-Wei He, and Yi-Min Wang. Distributed nonnegative matrix factorization for web-scale dyadic data analysis on mapreduce. In *Proceedings of the 19th international conference on World wide web*, pages 681–690. ACM, 2010.
- [MBPS10] Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. Online learning for matrix factorization and sparse

BIBLIOGRAPHY

- coding. *Journal of Machine Learning Research*, 11(Jan):19–60, 2010.
- [MHA08] Morten Mørup, Lars Kai Hansen, and Sidse M Arnfred. Algorithms for sparse nonnegative tucker decompositions. *Neural computation*, 20(8):2112–2131, 2008.
- [Mie10] Pauli Miettinen. Sparse boolean matrix factorizations. In *2010 IEEE International Conference on Data Mining*, pages 935–940. IEEE, 2010.
- [Mir14] Andri Mirzal. A convergent algorithm for orthogonal non-negative matrix factorization. *Journal of Computational and Applied Mathematics*, 260:149–166, 2014.
- [MMG⁺08] Pauli Miettinen, Taneli Mielikäinen, Aristides Gionis, Gautam Das, and Heikki Mannila. The discrete basis problem. *IEEE Transactions on Knowledge and Data Engineering*, 20(10):1348–1362, 2008.
- [MMH08] Morten Morup, Kristoffer Hougaard Madsen, and Lars Kai Hansen. Approximate l0 constrained non-negative matrix and tensor factorization. In *2008 IEEE International Symposium on Circuits and Systems*, pages 1328–1331. IEEE, 2008.
- [MV11] Pauli Miettinen and Jilles Vreeken. Model order selection for boolean matrix factorization. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 51–59. ACM, 2011.
- [MV14] Pauli Miettinen and Jilles Vreeken. mdl4bmf: Minimum description length for boolean matrix factorization. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 8(4):18, 2014.
- [NH16] Duy-Khuong Nguyen and Tu-Bao Ho. Fast parallel randomized algorithm for nonnegative matrix factorization with kl divergence for large sparse datasets. *International Journal of Machine Learning and Computing*, 6(2):111, 2016.
- [Niy04] X Niyogi. Locality preserving projections. In *Neural information processing systems*, volume 16, page 153. MIT, 2004.

- [NKLR⁺10] Masahiro Nakano, Hirokazu Kameoka, Jonathan Le Roux, Yu Kitano, Nobutaka Ono, and Shigeki Sagayama. Convergence-guaranteed multiplicative algorithms for nonnegative matrix factorization with β -divergence. *choice*, 10:1, 2010.
- [OP06] Paul D O’grady and Barak A Pearlmutter. Convolutional non-negative matrix factorisation with a sparseness constraint. In *2006 16th IEEE Signal Processing Society Workshop on Machine Learning for Signal Processing*, pages 427–432. IEEE, 2006.
- [Ose11] Ivan V Oseledets. Tensor-train decomposition. *SIAM Journal on Scientific Computing*, 33(5):2295–2317, 2011.
- [P⁺13] Evangelos E Papalexakis et al. Scoup-smt: Scalable coupled sparse matrix-tensor factorization. *arXiv preprint arXiv:1302.7043*, 2013.
- [PC11] Anh Huy Phan and Andrzej Cichocki. Extended hals algorithm for nonnegative tucker decomposition and its applications for multiway analysis and classification. *Neurocomputing*, 74(11):1956–1969, 2011.
- [PFS12] Evangelos E Papalexakis, Christos Faloutsos, and Nicholas D Sidiropoulos. Parcube: sparse parallelizable tensor decompositions. In *Machine Learning and Knowledge Discovery in Databases*, pages 521–536. Springer, 2012.
- [PGAG12] Filippo Pompili, Nicolas Gillis, P-A Absil, and François Glineur. Two algorithms for orthogonal nonnegative matrix factorization with application to clustering. *arXiv preprint arXiv:1201.0901*, 2012.
- [PKA10] Yannis Panagakis, Constantine Kotropoulos, and Gonzalo R Arce. Non-negative multilinear principal component analysis of auditory temporal modulations for music genre classification. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(3):576–588, 2010.
- [PSB13] Evangelos E Papalexakis, Nicholas D Sidiropoulos, and Rasmus Bro. From k-means to higher-way co-clustering: Multilinear decomposition with sparse latent factors. *IEEE transactions on signal processing*, 61(2):493–506, 2013.

BIBLIOGRAPHY

- [PSBP04] V Paul Pauca, Farial Shahnaz, Michael W Berry, and Robert J Plemmons. Text mining using non-negative matrix factorizations. In *SDM*, volume 4, pages 452–456. SIAM, 2004.
- [PT94] Pentti Paatero and Unto Tapper. Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics*, 5(2):111–126, 1994.
- [PTC12] Anh-Huy Phan, Petr Tichavský, and Andrzej Cichocki. Fast alternating ls algorithms for high order candecomp/parafac tensor factorizations. *IEEE TSP, CoRR*, 2012.
- [S⁺14] N Sidiropoulos et al. Parallel randomly compressed cubes: A scalable distributed architecture for big tensor decomposition. *Signal Processing Magazine, IEEE*, 31(5):57–70, 2014.
- [SB00] Nicholas D Sidiropoulos and Rasmus Bro. On the uniqueness of multilinear decomposition of n-way arrays. *Journal of chemometrics*, 14(3):229–239, 2000.
- [SBPP06] Farial Shahnaz, Michael W Berry, V Paul Pauca, and Robert J Plemmons. Document clustering using nonnegative matrix factorization. *Information Processing & Management*, 42(2):373–386, 2006.
- [SD05] Suvrit Sra and Inderjit S Dhillon. Generalized nonnegative matrix approximations with bregman divergences. In *Advances in neural information processing systems*, pages 283–290, 2005.
- [SDLF⁺16] Nicholas D Sidiropoulos, Lieven De Lathauwer, Xiao Fu, Kejun Huang, Evangelos E Papalexakis, and Christos Faloutsos. Tensor decomposition for signal processing and machine learning. *arXiv preprint arXiv:1607.01668*, 2016.
- [SH05] Amnon Shashua and Tamir Hazan. Non-negative tensor factorization with applications to statistics and computer vision. In *Proceedings of the 22nd international conference on Machine learning*, pages 792–799. ACM, 2005.
- [SKAU12] Hiroshi Sawada, Hirokazu Kameoka, Shoko Araki, and Naonori Ueda. Efficient algorithms for multichannel extensions of itakura-saito nonnegative matrix factorization. In

- 2012 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 261–264. IEEE, 2012.
- [SL13] Hyun Ah Song and Soo-Young Lee. Hierarchical representation using nmf. In *International Conference on Neural Information Processing*, pages 466–473. Springer, 2013.
- [SM06] Mikkel N Schmidt and Morten Mørup. Nonnegative matrix factor 2-d deconvolution for blind single channel source separation. In *International Conference on Independent Component Analysis and Signal Separation*, pages 700–707. Springer, 2006.
- [Sma04] Paris Smaragdis. Non-negative matrix factor deconvolution; extraction of multiple sound sources from monophonic inputs. In *International Conference on Independent Component Analysis and Signal Separation*, pages 494–499. Springer, 2004.
- [SMTT16] Motoki Shiga, Shunsuke Muto, Kazuyoshi Tatsumi, and Koji Tsuda. Matrix factorization for automatic chemical mapping from electron microscopic spectral imaging datasets. *Transactions of the Materials Research Society of Japan*, 41(4):333–336, 2016.
- [SPK08] Václav Snášel, Jan Platoš, and Pavel Krömer. Developing genetic algorithms for boolean matrix factorization. *Databases, Texts*, 61, 2008.
- [SRSK15] Shaden Smith, Niranjay Ravindran, Nicholas D Sidiropoulos, and George Karypis. Splatt: Efficient and parallel sparse tensor-matrix multiplication. In *2015 IEEE International Parallel and Distributed Processing Symposium*,, pages 61–70. IEEE, 2015.
- [SS10] Bin Shen and Luo Si. Non-negative matrix factorization clustering on multiple manifolds. In *AAAI*, 2010.
- [SS12] Ankan Saha and Vikas Sindhwani. Learning evolving and emerging topics in social media: a dynamic nmf approach with temporal regularization. In *Proceedings of the 5th International Conference on Web Search and Data Mining*, pages 693–702. ACM, 2012.

BIBLIOGRAPHY

- [STM⁺16] Motoki Shiga, Kazuyoshi Tatsumi, Shunsuke Muto, Koji Tsuda, Yuta Yamamoto, Toshiyuki Mori, and Takayoshi Tanji. Sparse modeling of eels and edx spectral imaging data by nonnegative matrix factorization. *Ultramicroscopy*, 170:43–59, 2016.
- [TBF⁺12] Robert Tibshirani, Jacob Bien, Jerome Friedman, Trevor Hastie, Noah Simon, Jonathan Taylor, and Ryan J Tibshirani. Strong rules for discarding predictors in lasso-type problems. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 74(2):245–266, 2012.
- [tBS02] Jos MF ten Berge and Nikolaos D Sidiropoulos. On uniqueness in candecomp/parafac. *Psychometrika*, 67(3):399–409, 2002.
- [TBZS14] George Trigeorgis, Konstantinos Bousmalis, Stefanos Zafeiriou, and Bjoern Schuller. A deep semi-nmf model for learning hidden representations. In *ICML*, pages 1692–1700, 2014.
- [TKB09] Christian Thureau, Kristian Kersting, and Christian Bauckhage. Convex non-negative matrix factorization in the wild. In *2009 Ninth IEEE International Conference on Data Mining*, pages 523–532. IEEE, 2009.
- [TZBG14] Lei Tong, Jun Zhou, Xiao Bai, and Yongsheng Gao. Dual graph regularized nmf for hyperspectral unmixing. In *2014 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*,, pages 1–8. IEEE, 2014.
- [Vav09] Stephen A Vavasis. On the complexity of nonnegative matrix factorization. *SIAM Journal on Optimization*, 20(3):1364–1377, 2009.
- [VT03] M Alex O Vasilescu and Demetri Terzopoulos. Multilinear subspace analysis of image ensembles. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, volume 2, pages II–93. IEEE, 2003.
- [WCC09] Wenwu Wang, Andrzej Cichocki, and Jonathon Chambers. A multiplicative algorithm for convolutive non-negative matrix factorization based on squared euclidean distance. 2009.

- [WLK11] Fei Wang, Ping Li, and Arnd Christian König. Efficient document clustering via online nonnegative matrix factorizations. In *SDM*, volume 11, pages 908–919. SIAM, 2011.
- [WLZ08] Fei Wang, Tao Li, and Changshui Zhang. Semi-supervised clustering via matrix factorization. In *SDM*, pages 1–12. SIAM, 2008.
- [WRR03] Michael E Wall, Andreas Rechtsteiner, and Luis M Rocha. Singular value decomposition and principal component analysis. In *A practical approach to microarray data analysis*, pages 91–109. Springer, 2003.
- [WSL14] Yi Wu, Bin Shen, and Haibin Ling. Visual tracking via online nonnegative matrix factorization. *IEEE Transactions on Circuits and Systems for Video Technology*, 24(3):374–383, 2014.
- [XLG03] Wei Xu, Xin Liu, and Yihong 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*, pages 267–273. ACM, 2003.
- [XTCC08] Yun Xue, Chong Sze Tong, Ying Chen, and Wen-Sheng Chen. Clustering-based initialization for non-negative matrix factorization. *Applied Mathematics and Computation*, 205(2):525–536, 2008.
- [XY⁺11] Zenglin Xu, Feng Yan, et al. Infinite tucker decomposition: Nonparametric bayesian models for multiway data analysis. *arXiv preprint arXiv:1108.6296*, 2011.
- [YC08] Jiho Yoo and Seungjin Choi. Orthogonal nonnegative matrix factorization: Multiplicative updates on stiefel manifolds. In *Intelligent Data Engineering and Automated Learning*, pages 140–147. Springer, 2008.
- [YCI12] Naoto Yokoya, Jocelyn Chanussot, and Akira Iwasaki. Generalized bilinear model based nonlinear unmixing using semi-nonnegative matrix factorization. In *2012 IEEE International Geoscience and Remote Sensing Symposium*, pages 1365–1368. IEEE, 2012.

BIBLIOGRAPHY

- [YKT12] Sayaka Yamauchi, Masanori Kawakita, and Junichi Takeuchi. Botnet detection based on non-negative matrix factorization and the mdl principle. In *International Conference on Neural Information Processing*, pages 400–409. Springer, 2012.
- [YNRA12] X-X Yin, BW-H Ng, Kotagiri Ramamohanarao, and Derek Abbott. Tensor based sparse decomposition of 3d shape for visual detection of mirror symmetry. *Computer methods and programs in biomedicine*, 108(2):629–643, 2012.
- [YO05] Zhijian Yuan and Erkki Oja. Projective nonnegative matrix factorization for image compression and feature extraction. In *Scandinavian Conference on Image Analysis*, pages 333–342. Springer, 2005.
- [YO10] Zhirong Yang and Erkki Oja. Linear and nonlinear projective nonnegative matrix factorization. *IEEE Transactions on Neural Networks*, 21(5):734–749, 2010.
- [YYL07] Zhirong Yang, Zhijian Yuan, and Jorma Laaksonen. Projective non-negative matrix factorization with applications to facial image processing. *International Journal of Pattern Recognition and Artificial Intelligence*, 21(08):1353–1362, 2007.
- [ZBT13] Tianyi Zhou, Wei Bian, and Dacheng Tao. Divide-and-conquer anchoring for near-separable nonnegative matrix factorization and completion in high dimensions. In *2013 IEEE 13th International Conference on Data Mining*, pages 917–926. IEEE, 2013.
- [ZLD⁺10] Zhong-Yuan Zhang, Tao Li, Chris Ding, Xian-Wen Ren, and Xiang-Sun Zhang. Binary matrix factorization for analyzing gene expression data. *Data Mining and Knowledge Discovery*, 20(1):28–52, 2010.
- [ZLDZ07] Zhongyuan Zhang, Tao Li, Chris Ding, and Xiangsun Zhang. Binary matrix factorization with applications. In *Seventh IEEE International Conference on Data Mining*, pages 391–400. IEEE, 2007.
- [ZT16] Renbo Zhao and Vincent YF Tan. Online nonnegative matrix factorization with outliers. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 2662–2666. IEEE, 2016.

- [ZYXY11] Guoxu Zhou, Zuyuan Yang, Shengli Xie, and Jun-Mei Yang. Online blind source separation using incremental nonnegative matrix factorization with volume constraint. *IEEE transactions on neural networks*, 22(4):550–560, 2011.
- [ZZX⁺16] Qibin Zhao, Guoxu Zhou, Shengli Xie, Liqing Zhang, and Andrzej Cichocki. Tensor ring decomposition. *arXiv preprint arXiv:1606.05535*, 2016.
- [ZZZ⁺14] Xianchao Zhang, Long Zhao, Linlin Zong, Xinyue Liu, and Hong Yu. Multi-view clustering via multi-manifold regularized nonnegative matrix factorization. In *2014 IEEE International Conference on Data Mining*, pages 1103–1108. IEEE, 2014.