Doctoral Dissertation

Fast Algorithms and Rich Models for Nonnegative Matrix Factorization

Nguyen Duy Khuong

Supervisor: Professor Tu-Bao Ho

School of Knowledge Science
Japan Advanced Institute of Science and Technology

March, 2016

To my family with all my love and honor.

Abstract

Experiments, observations, and numerical simulations in science with the major support of modern computers and sensor technologies are generating terabytes and petabytes of data. These datasets require rich models and fast algorithms to analyze large datasets to discover inside hidden knowledge for creating major breakthroughs in science, technology, industry, services and among others.

Nonnegative matrix factorization (NMF) is a linear powerful technique for dimension reduction, extracting latent factors and learning part-based representation. Specially, it reduces dimension of data making learning algorithms faster and more effective as they often work less effectively due to the curse of dimensionality. Moreover, latent factor extraction and part-based learning representation give concise interpretability of datasets to discover hidden knowledge. Although NMF and its applications have been developed for more than a decade, they have still several limitations of modeling and performance.

In this study, we designs rich models and fast algorithms for nonnegative matrix factorization. Specially, rich models provides concise interpretability describing data and enhance the quality of models by adding constraints to adapt the complexity of growing large datasets. In addition, fast algorithms are essential to find out these rich models for large datasets.

In summary, this study has the following contributions:

Firstly, concerning about rich NMF models, we propose a new rich NMF model as simplicial nonnegative matrix factorization and nonnegative matrix factorization with L_1 L_2 regularizations. Simplicial nonnegative matrix factorization can enhance smoothness and sparsity, and give more concise interpretability of the role of latent components over data instances. In addition, we generalize another rich NMF model as nonnegative matrix factorization with L_1 L_2 regularizations for Frobenius norm and KL divergence, which can enhance smoothness and sparsity of NMF models.

Secondly, we propose a fast parallel and distributed algorithm using limited internal memory for nonnegative matrix factorization with Frobenius norm with L_1 L_2 regularizations, which is based on the the accelerated anti-lopsided algorithm for nonnegative least squares. The proposed algorithm has fast over-bounded guaranteed convergence $O([(1-\frac{\mu}{L})(1-\frac{\mu}{rL})^{2r}]^k)$ in the space of passive variables, where convex parameter μ and Lipschitz constant L are bounded as $\frac{1}{2} \leq \mu \leq L \leq r$.

Thirdly, we propose a fast parallel randomized algorithm for NMF nonnegative matrix factorization with L_1 L_2 regularizations and KL divergence for large sparse datasets. The proposed algorithm has fast convergence, and utilize the sparse properties of data, model and representation. In addition, the experiments indicate that sparse models and sparse representation are archived for large sparse datasets, which is a significant milestone in this research problem.

Fourthly, we propose a fast parallel algorithm for simplicial nonnegative matrix factorization with Frobenius norm. The proposed algorithm has guaranteed instance inference with sub-linear convergence $\mathcal{O}(1/k)$, low iteration complexity, and easy sparsity control.

Finally, we propose a fast parallel algorithm for simplicial nonnegative matrix factorization with Kullback–Leibler divergence. The proposed algorithm has guaranteed instance inference with sub-linear convergence $\mathcal{O}(1/k)$, and easy sparsity control. The experiments indicate that this approach can achieve highly sparse representation with higher accuracy in comparison with equivalent approaches.

In summary, this thesis discusses two significant mutual aspects of nonnegative matrix factorization as rich models and fast algorithms. Specifically, we propose rich models and their four fast parallel algorithms for nonnegative matrix factorization for two divergences, which can adapt with large scale applications and various datasets.

Keywords: Rich models, fast algorithms, nonnegative matrix factorization, parallel and distributed, Frobenius norm, KL divergence

Acknowledgements

First of all, I would like to thank Professor Ho Tu Bao with my deep appreciation. His advices and comments have helped me become more mature. Furthermore, I am very grateful to his encouragement and support for me through this grad school. His inspiration and knowledge have been invaluable for my future academic career.

Secondly, sincere thanks go to Dam Hieu Chi and Nguyen Ngoc Binh. Professor Dam Hieu Chi have given me useful advice and insightful discussion to enhance my researches. Professor Nguyen Ngoc Binh have guided me a study of distributed computation for optimization for large datasets in Apache Spark, which is significant to become a data scientist in both academic and industry.

Thirdly, I would like to express my gratitude the committee members of my dissertation, including Takashi Hashimoto, Riichiro Mizoguchi, Hieu Chi Dam, and Hiroshi Motoda due to their helpful comments. In addition, I thank my friends and colleges for fruitful discussion and collaboration. I have studied a lot from the discussion with them including Than Quang Khoat, Tran Dinh Quoc, Nguyen Dang Thanh, Bui Ngoc Thang, Nguyen Vu Linh, Hoang Khanh Hung, and Dang Tran Thai.

I would like to thank my family for their continual supports and encouragements during my long-term study. Without their encouragement and support, it is hard for me to complete this study.

Finally, I want to express my gratitude to Vietnamese Ministry of Education and Training which financially supported me for four years of this study through 322 Scholarship Program; and Japan Advanced Institute of Science and Technology which supported me the tuition fee through Collaborative Education Program JAIST-FIVE. In addition, I would like to thank my school, School of Knowledge Science, and Asian Office of Aerospace R&D for additional financial support under research projects.

The whole of science is nothing more than a refinement of everyday thinking.

— Albert Einstein —

Contents

This dissertation was prepared according to the curriculum for the Collaborative Education Program organized by Japan Advanced Institute of Science and Technology, and University of Engineering and Technology, Vietnam National University, Hanoi.

List of Figures

List of Tables

Chapter 1

Introduction

Hiding within those mounds of data is knowledge that could change the life of a patient, or change the world.
— ATUL BUTTE, STANFORD

Nonnegative matrix factorization is a linear fast powerful dimensionality reduction, which can extract latent factors to effectively interpret and discover hidden knowledge in datasets. In this chapter, we introduce basic concepts for nonnegative matrix factorization.

1.1 Introduction to Nonnegative Matrix Factorization

Experiments, observations, and numerical simulations in science with the major support of modern computers and sensor technologies have been generating terabytes and petabytes of data [24, 79]. Finding hidden knowledge in these datasets has already led to major breakthroughs in science, technology, industry, services and among others [1, 24, 74]. For example, Google needs to search 30 trillion web pages and 100 billion times a month for giving useful suggestions for users [1]. The entire contents of the National Institutes of Health's 1000 Genomes Project are approximately all 200 terabytes to be analyzed to find out hidden disease patterns, which can be the answers for curing diseases as cancer and diabetes [2] [1].

Many datasets are formatted in pair-relationship nonnegative matrices such as documents, images, genes, log datasets, ... [22] In addition, latent factors are hidden inside these datasets and underlying the relationship. For example, latent topics are hidden between words and documents [3, 34]. Specifically, each document is a mixture of latent

[1]http://venturebeat.com
[2]http://www.zdnet.com

topics, while each topic contains a number of weighted words. The process of extracting these latent factors in these datasets is known as learning representation, which is one key problem in artificial intelligence for understanding knowledge inside datasets, dimensionality reduction, information retrieval, and many other tasks in machine learning [6, 30]. Furthermore, this process can be conducted by nonnegative matrix factorization which is a linear fast powerful dimensionality reduction to extract latent factors to effectively interpret and discover hidden knowledge and relationships between two sets of objects in the datasets. In this thesis, we study several significant issues of nonnegative matrix factorization to effectively extract latent factors in these datasets.

1.1.1 Notations and abbreviations

In this thesis, we use notations as follows:

n	Dimension of data instances
m	Number of data instances
r	Number of latent factors or number of latent components
V	$\in \mathcal{R}^{n\times m}$ as data nonnegative matrix
W	$\in \mathcal{R}^{r\times n}$ as basis vector matrix
F	$\in \mathcal{R}^{r\times m}$ as coefficient factor matrix
A_j	Column j^{th} vector
$\alpha, \alpha_1, ...$	L_2 regularization parameter
$\beta, \beta_1, ...$	L_1 regularization parameter
γ	Orthogonality regularization parameter
W_i, F_j	$\in \mathcal{R}^k$
$\mathbf{1}^n$	$[1, 1, ..., 1]^T \in \mathcal{R}^n$
$\mathbf{0}^n$	$[0, 0, ..., 0]^T \in \mathcal{R}^n$
$D(x\|y)$	Function to measure the difference between x and y
$\|x-y\|_1$	$= \sum_{i=1}^{n} \lvert x_i - y_i \rvert$, where $x, y \in \mathcal{R}^n$
$\|x\|_p^p$	$= \sum_{i=1}^{n} x_i^p$, where $x \in \mathcal{R}^n$ and $p \in 1, 2, 3, ...$

List of abbreviations:

NMF	Nonnegative matrix factorization
NTF	Nonnegative tensor factorization
sNMF	Simplicial nonnegative matrix factorization
KL	Kullback–Leibler
LS	Least squares
NNLS	Nonnegative least squares

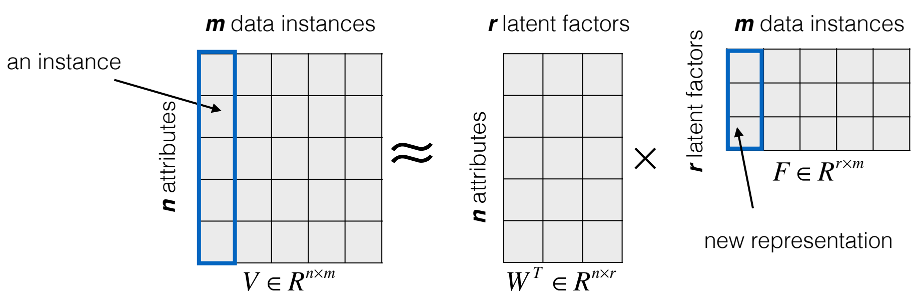

Fig. 1.1 Illustration of nonnegative matrix factorization

1.1.2 Nonnegative matrix factorization

Nonnegative matrix factorization (NMF) is a powerful technique widely used in applications of data mining, signal processing, computer vision, bioinformatics, etc. [92, 98]. Fundamentally, NMF has two main purposes. First, it reduces dimension of data making learning algorithms faster and more effective as they often work less effectively due to the curse of dimensionality [41]. Second, NMF helps extracting latent components and learning part-based representation, which are the significant distinction from other dimension reduction methods such as Principal Component Analysis (PCA), Independent Component Analysis (ICA), Vector Quantization (VQ), etc. This feature originates from transforming data into lower dimension of latent components and non-negativity constraints [29, 34, 58].

Mathematically, nonnegative matrix factorization (NMF) is formualated as follows:

Definition 1 *Given a dataset consisting of m vectors in n dimensions $V = [V_1, V_2, ..., V_m] \in R_+^{n \times m}$, where each vector presents a data instance. NMF seeks to approximately factorize V into a product of two nonnegative factorizing matrices W^T and F, where $W \in R_+^{r \times n}$ and $F \in R_+^{r \times m}$ are coefficient matrix and latent component matrix, respectively, $V \approx W^T F$.*

The illustration of approximate nonnegative matrix factorization is visually presented in Figure 1.1. Specifically, a nonnegative data matrix $V \in R_+^{n \times m}$, each column of which is a data instance and each row presents an attribute of data instances, is approximately factorized into a product of two nonnegative matrices as $W^T \in R^{n \times r}$ and $F \in R^{r \times m}$. In addition, each column of W^T is latent factors inside the dataset V, and each column of F is new presentation of an column in the data matrix V.

Furthermore, after determining the two factor matrices W^T and F, each instance is approximated by a linear combination of these latent components $V_i \approx \sum_{k=1}^{r} F_{ki} W_k^T$. Hence, the vectors in W^T are latent factors underlying the dataset V, and F_i is the new representation of vector V_i in the new space of latent factors W^T. In addition, F_i can be specified when W^T is fixed; thus, W^T can be considered the parameters of the NMF model. F_i is a representative of the data instance V_i in a higher abstract space of W^T instead of raw coordinates. Furthermore, datasets are assumed to follow an additive model [22], and non-negativity conditions are significant to successfully extract latent factors because they reduce overlapping of extracted hidden factors that are raised by lacking the orthogonality conditions of basis vectors.

1.1.3 Norms and divergences for NMF

In another mathematical way of understanding NMF, the factorization is the transformation of vectors in dataset V to a new space of vectors in W^T to obtain new representation F. Specifically, determining F is considered the projection of datasets into a new space of latent factors W^T. $W^T F$ is the result of projecting back into the original space of new coefficients F.

Furthermore, the whole of information in the original space can not be kept because m, n is often bigger than r. To retain the properties of V, the factorization needs to guarantee having at least information loss via minimizing an objective function $D(V\|W^T F) = \sum_{j=1}^{m} d(V_{ij}\|[W^T F]_{ij})$; where $d(x\|y)$ can be considered a function to measure the difference between x and y.

In practice, many functions are employed depending on the properties of datasets. For examples, Frobenius norm is suitable for image datasets [39], Kullback–Leibler (KL) divergence is employed for count sparse datasets such as documents, and Itakura-Saito (IS) divergence is more suitble for decomposing a power spectrogram in musical applications [95]. In this research, we focus on two of the most popular divergences:

- Frobenius norm: $d(x\|y) = (x - y)^2$
- KL divergence: $d(x\|y) = x \log(\frac{x}{y}) - x + y$

1.1.4 Applications for nonnegative matrix factorization

Nonnegative matrix factorization has been widely employed in many applications because it can automatically extract sparse and easily interpretable latent components [34]

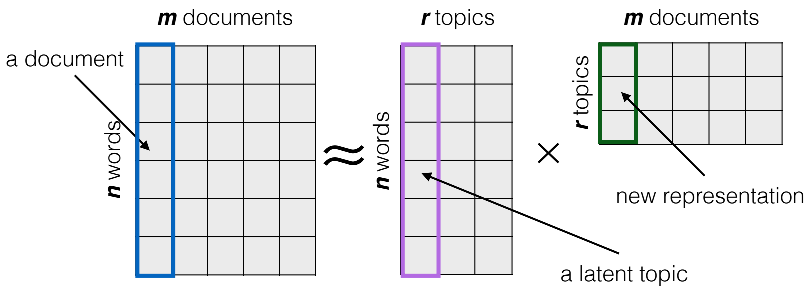

Fig. 1.2 Illustration of nonnegative matrix factorization

inside datasets to more effectively represent data instances and also discover hidden knowledge. In computer science, it have been applied in many research areas such as image processing for feature extraction [58, 69]; text mining for topic recovery [3], document classification [8], document clustering [53, 84], etc; hyperspectral imaging for identifying endmembers and classifying pixels [65, 68]; multimedia for blind source separation and music analysis [91]; social network for community detection [93]; etc. In other fields, NMF is a powerful tool to effectively analyze the experimental results to create significant breakthroughs such as in genomics for deciphering signatures of mutational processes operative in human cancer [1], in physics for understanding of the optical response of noble-metal nanoparticles [74], etc.

One example of NMF application is text processing [34], in which a collection of documents are given to extract latent topics and find the new document representation as a mixture of latent topics, see Figure 1.2. Specifically, each document is represented by weighted words in a column of V, and each latent factor is represented by weighted words in a column of W^T. Each new document representation as a column of F is a mixture of latent topics. This technique is widely applied in many applications of information retrieval, document classification, document clustering, etc...

1.1.5 Nonnegative matrix factorization in knowledge science

We have been interacting large volumes of data in scientific fields to discover knowledge for this digital era [24]. However, humans are always limited in processing the large mount and the high dimension of data. Hence, knowledge discovery must be essentially supported by dimensionality reduction and data processing methods which create and justify hypotheses in order to discover hidden knowledge inside observed data from experiments.

In comparison with dimensionality reduction like PCA, ICA, nonnegative factorization factorization has more concise interpretability because it can achieve linear part-base representation with high comprehension [86]. Specifically, observed instances are analyzed into small structure parts; then, these intermediate results are effectively interacted to create hypothesis. For example, documents can be represented by a combination of hidden topics [34]. Extracting hidden topics lead to understanding and managing knowledge of documents more effectively. Therefore, nonnegative matrix factorization is a powerful tool to manage and discovery knowledge from observed datasets in order to provide convenient services for us.

1.2 Research Context

Nonnegative matrix factorization has more than a decade of rapid development with numerous applications. Firstly, two milestones in early days of the NMF historical development were its mathematical formulations as positive matrix factorization with Byzantine algorithms [75] and as parts-based representation with a simple effective algorithm [58]. Then, various works contributed to NMF can be viewed in three major perspectives: variants of NMF, algorithms and applications. In particular, variants of NMF are based on either divergence functions [83,97], or constraints [43,76], or regularizations [19,59]. Most NMF algorithms were developed in two main directions: provable algorithms [2] and iterative multiplicative update algorithms [98].

Concerning provable algorithms, two major limitations of complexity and flexibility still exist for challenging current researches [2]. Currently, state-of-the-art algorithms running in a polynomial time $O((mn)^{O(r)})$ may reduce the feasibility for numerous large scale applications. In addition, it is difficult to add more constraints into the objective function and to change the kinds of objection functions because it is believed to need many practical settings [2]. Hence, this thesis only discusses rich models and their fast algorithms based on iterative multiplicative update algorithms to deal with big data.

The iterative multiplicative update algorithms can be divided into inexact block coordinate descent methods [7,12,57,58,63], and exact block coordinate descent methods [46, 49, 96], and accelerated block coordinate descent methods [35, 36]. These algorithms is classified based on the optimization strategies. The common characteristic of inexact block coordinate descent methods is their usage of gradient methods to seek an approximate solution for NNLS problems, which is neither optimal nor fulfilling of fast approximations and accelerated conditions. In addition, the specific characteristic of exact block coordinate descent methods is obtaining optimal solutions for two NNLS

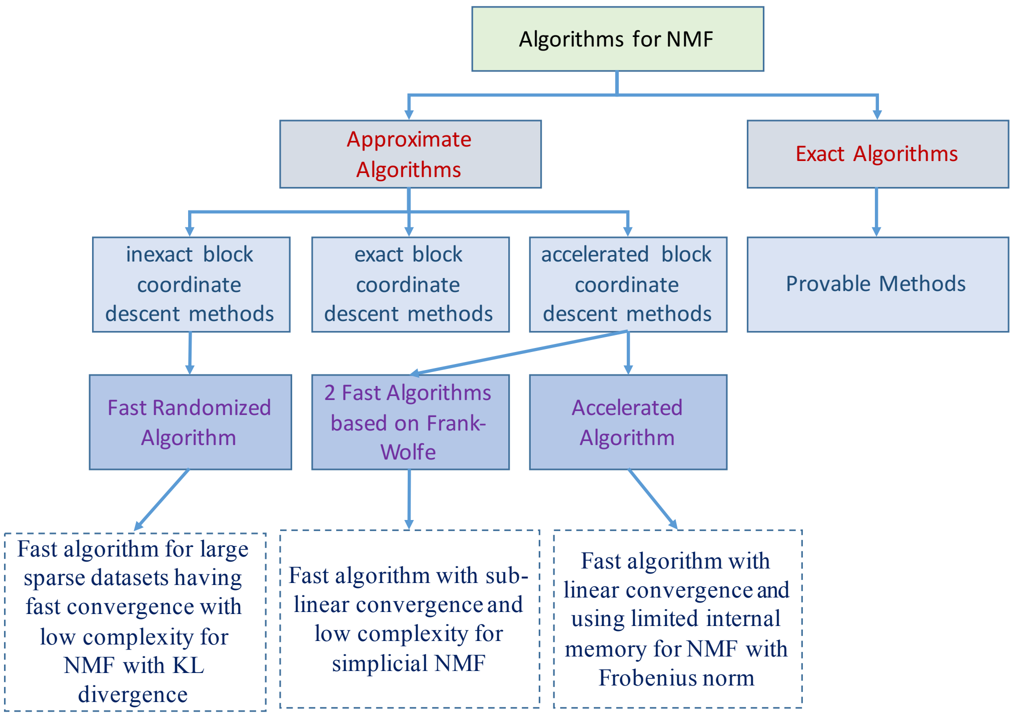
Algorithms for NMF
Approximate Algorithms
Exact Algorithms
inexact block coordinate descent methods
exact block coordinate descent methods
accelerated block coordinate descent methods
Provable Methods
Fast Randomized Algorithm
2 Fast Algorithms based on Frank-Wolfe
Accelerated Algorithm
Fast algorithm for large sparse datasets having fast convergence with low complexity for NMF with KL divergence
Fast algorithm with sub-linear convergence and low complexity for simplicial NMF
Fast algorithm with linear convergence and using limited internal memory for NMF with Frobenius norm

Fig. 1.3 Research context of proposed fast algorithms

problems in each iteration. Finally, the accelerated methods use fast solution approximations satisfying accelerated conditions to reduce the complexity and to keep fast convergence.

In this research, we propose rich models and fast algorithms for nonnegative matrix factorization. Concerning rich models, two rich models, namely NMF with L_1 L_2 regularizations and simplicial NMF (see Figure 1.4 that is extended from [92]), are proposed to enhance sparsity, smoothness and interpretability. In comparison with the previous models, the proposed rich models are more general, robust and interpretable to deal with various data. Regarding to fast algorithms, we propose four fast parallel algorithms for four NMF variants of two rich models and two divergences. Furthermore, the proposed algorithms are carefully designed to achieve low complexity and fast convergence to deal with big data.

1.3 Motivations and Challenges

Although nonnegative matrix factorization and its applications have been developed for more a decade, there remains many open problems for further studies. Some of major

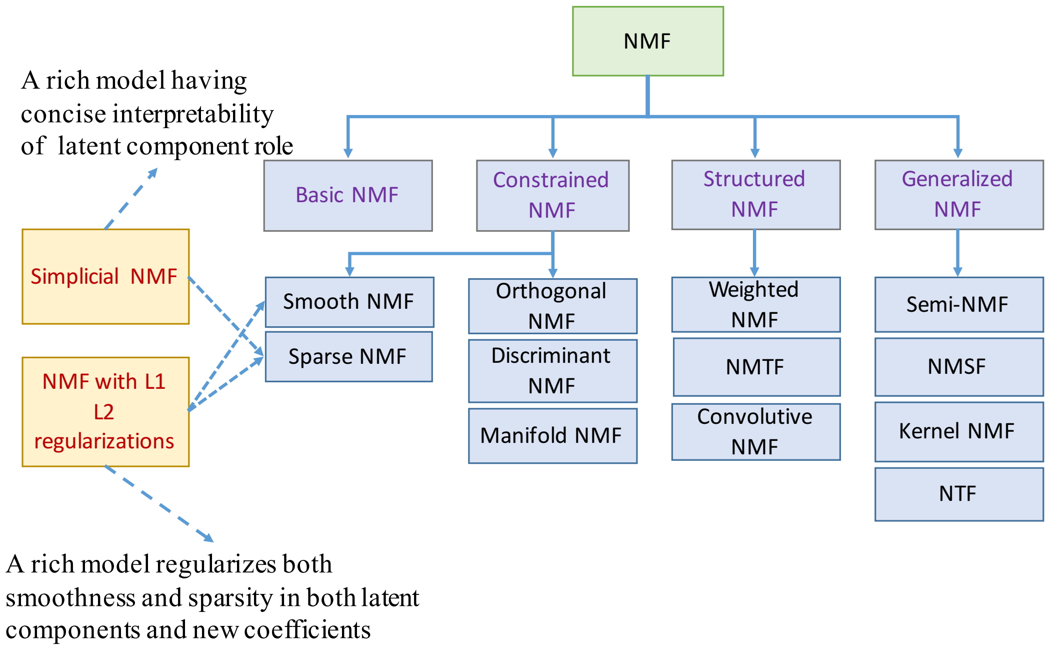
NMF
Basic NMF
Constrained NMF
Structured NMF
Generalized NMF
Smooth NMF
Sparse NMF
Orthogonal NMF
Discriminant NMF
Manifold NMF
Weighted NMF
NMTF
Convolutive NMF
Semi-NMF
NMSF
Kernel NMF
NTF
Simplicial NMF
NMF with L1 L2 regularizations
A rich model having concise interpretability of latent component role
A rich model regularizes both smoothness and sparsity in both latent components and new coefficients

Fig. 1.4 Research context of proposed rich models

challenges can be enumerated as follows:

C1. Rich models: Nonnegative matrix factorization is an ill-posed inverse problem. In other words, many local optimal solutions, called as stationary points, are existed. Hence, rich models are necessary to concisely interpret the relationship observed data and latent factors. Rich models will specially describe latent factors and new coefficients to enhance the factorization results. In addition, model-based approach provides significant advantages in creating highly tailored models for specific scenarios, rapidly prototyping and easily understanding various models [10].

C2. Fast convergence and low complexity: In practice, multiple iterative algorithms like EM algorithms containing a number of iterations are employed for NMF because NMF is often an ill-posed inverse problem and its objective function depends on separated sets of variables W^T and F. The number of used iterations and the iteration complexity mainly influence the speed of algorithm performance. In addition, NMF is a non-convex optimization problem, yet finding new W and F when fixing one of them is convex. Hence, algorithms having fast guaranteed convergence are preferred rather than heuristic unstable ones [36].

C3. Parallel and distributed computation: The development of technology and science has been generating huge data matrices [64], which is impossibly factorized by a single core or a computer. Hence, the computation must be parallelized and distributed to reduce the running time. The major challenges are how to decompose the computation into small independent computing units, deal with the limited internal memory and various constraints of smoothness, sparsity, and orthogonality [92, 98]. Furthermore, emerging new computing models such as Hadoop and Spark leads to redesign parallel NMF algorithms for distributed computation.

C4. Sparsity of models and representation: Sparsity is a natural property of data [5], which can be measured by the number of non-zeros elements. For example, the number of words in a topic and the number of topics in a document should be small [3,34]. Moreover, sparsity leads to save used internal memory, enhance the speed of algorithms, and void over-fitting problems [40, 89]. In nonnegative matrix factorization, algorithms need to attain sparse models W^T, sparse representation in F, and sparse computation that utilizes the sparsity of W^T and F to increase their speed.

Based on these challenges, we develop rich models having more concise interpretability with high quality of sparsity and smoothness. In addition, fast algorithms having fast convergence and low complexiy for these rich NMF models are proposed to deal with large datasets in parallel and distributed systems.

1.4 Thesis Structure

The thesis structure is depicted in Figure 1.5, which contains six main chapters discussing models and algorithms for nonnegative matrix factorization:

Chapter 2 propose two rich NMF models that give more concise interpretability and achieve high quality, namely NMF with L_1 L_2 regularizations and simplicial NMF for Frobenius norm and KL divergence.

Chapter 3 proposes an accelerated parallel and distributed algorithms for NMF with Frobenius norm, which inherits a novel accelerated anti-lopsided algorithm for non-negative least squares. Specifically, the proposed algorithm achieves over-bounded linear convergence rate of $\mathcal{O}([(1-\frac{\mu}{L})(1-\frac{\mu}{L})^{2r}]^k)$ in the sub-spaces of passive variables when fixing one of latent factor matrix; where μ and L are always bounded as $\frac{1}{2} \leq \mu \leq L \leq r$.

Chapter 4 introduces a fast parallel randomized algorithm for NMF with KL-divergence having fast convergence to solve major problems for count sparse datasets including sparse model, sparse representation, sparse algorithm, limited internal memory, and parallel algorithm.

Chapter 5 introduces a new formulation of NMF as simplicial NMF for Frobenius norm, which give a more concise interpretability of the roles of latent factors contribut-

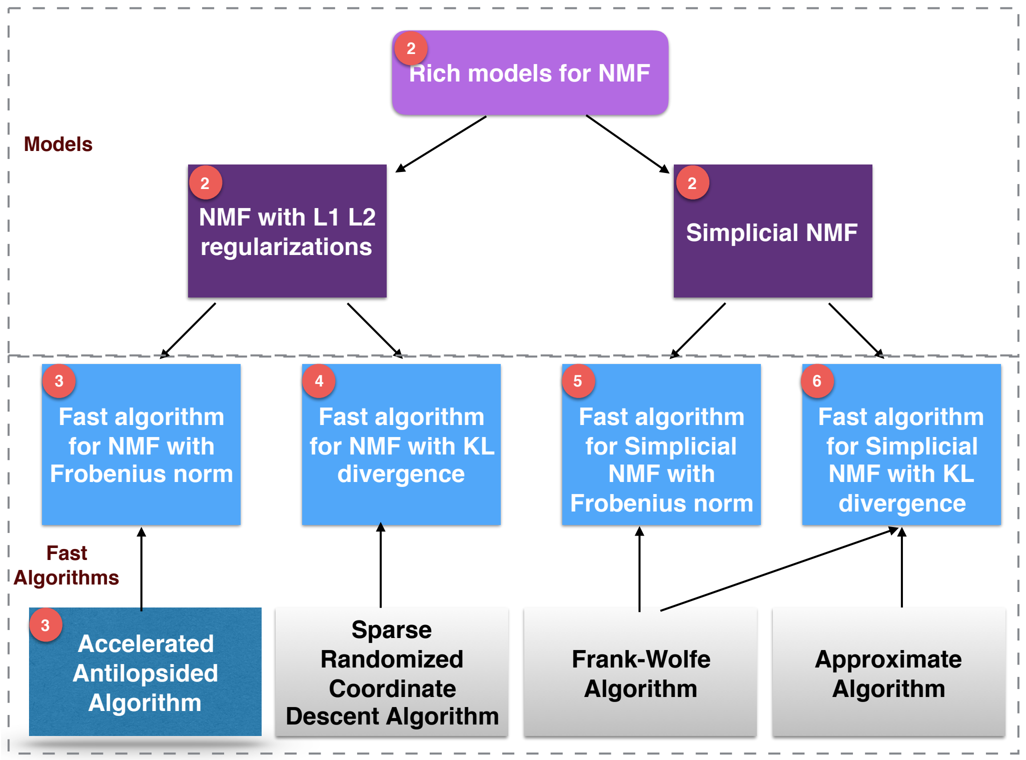
2 Rich models for NMF
Models
2 NMF with L1 L2 regularizations
2 Simplicial NMF
3 Fast algorithm for NMF with Frobenius norm
4 Fast algorithm for NMF with KL divergence
5 Fast algorithm for Simplicial NMF with Frobenius norm
6 Fast algorithm for Simplicial NMF with KL divergence
Fast Algorithms
3 Accelerated Antilopsided Algorithm
Sparse Randomized Coordinate Descent Algorithm
Frank-Wolfe Algorithm
Approximate Algorithm

Fig. 1.5 Thesis structure: numbers in circles refer chapters

ing on the data instances by assuming that each instance is a probabilistic contribution of latent components, while it still enhance sparsity and smoothness. In addition, the proposed inference algorithms based on Frank-Wolf algorithm has sub-linear convergence rate $O(1/k)$, which can achieve concise interpretability, fast convergence versus iterations, sparse representation, and high accuracy classification.

Chapter 6 introduce a new formulation of NMF as simplicical NMF KL divergence, which also give a more concise interpretability of the roles of latent factors contributing on the data instances by assuming that each instance is a probabilistic contribution of latent components, while it still enhance sparsity and smoothness. In addition, the proposed inference algorithms based on Frank-Wolf algorithm has sub-linear convergence rate $O(1/k)$, which can achieve fast convergence versus iterations, sparse representation, and high accuracy classification.

In summary, this thesis discusses two significant solid aspects of nonnegative matrix factorization as rich models and fast algorithms. Specifically, we propose rich models and their four fast parallel algorithms for nonnegative matrix factorization for two divergences, which can adapt with large scale applications and various datasets.

Chapter 2

Rich Models for Nonnegative Matrix Factorization

Essentially, all models are wrong, but some are useful.

— George Edward Pelham Box

Model and algorithm are the two significant mutual aspects for nonnegative matrix factorization (NMF). The model describes the processing of generating data, which the algorithm learns the model from training data and finds out the new representation of new data instances. In this chapter, we propose two rich models for nonnegative matrix factorization as simplicial NMF and NMF with L_1 L_2 regularizations, then the next chapters will discuss fast parallel algorithms for NMF variants of the rich models.

2.1 General NMF model

Nonnegative matrix factorization is a linear hidden factor model, which transfers high-dimensional observed data into a low-dimensional space [11] and extracts latent components inside data. Specially, V is approximately factorized by a product of two matrices: $V \approx W^T F$. W^T can be considered the parameters of NMF models, because the new coefficient F_i of V_i can be specified by $V_i \approx W^T F_i$ when W^T is fixed. W^T includes latent vectors or basis vectors; and F includes new coefficients as new low-rank representation of data instances. The quality of this factorization is guaranteed by a divergence $D(V \| W^T F)$ such as Frobenius norm, KL-divergence, etc [97, 98].

Although these divergences are often convex, NMF is a non-convex problem and has many local optimal solutions. Hence, various constraints and regularizations are added to control the quality of NMF [92, 98]. These constraints and regularizations are to enhance concise interpret-ability, sparsity, and orthogonality in order to provide better explanation, increase accuracy, and avoid over-fitting. Therefore, in practice, many variants of NMF are developed to be suitable for many real applications. In this

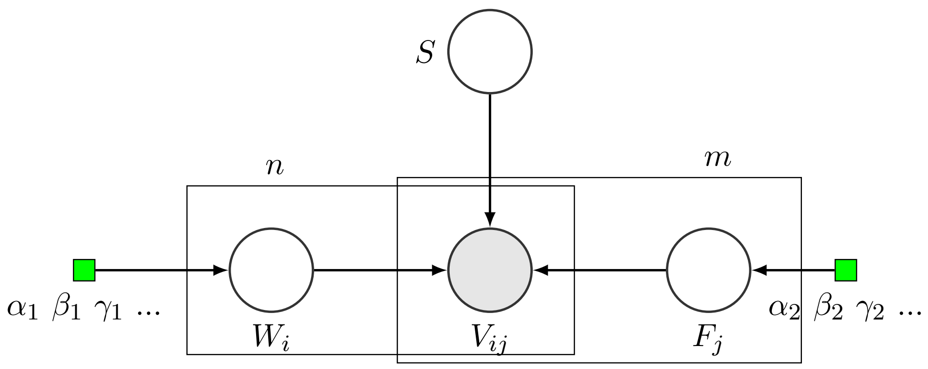
S
n
m
$\alpha_1\ \beta_1\ \gamma_1\ \ldots$
W_i
V_{ij}
F_j
$\alpha_2\ \beta_2\ \gamma_2\ \ldots$

Fig. 2.1 General model for nonnegative matrix factorization

section, we propose a general NMF model, which can catch many variants of NMF, see Figure 2.1. Based on this model, new researchers can conveniently comprehend existing NMF formulations during over a decade and also design new models for their various studies.

In the general model, it contains three main hidden variables S, W_i, and F_j; and several constraint parameters $\alpha_1, \alpha_2, \beta_1, \beta_2$, γ_1, γ_2, etc. W_i and F_j are latent factors contributing on the observation V_{ij}, which always appear in every NMF variant. In addition, because V is not normalized, S includes scaling factors for latent factors or provides additional interpretabilities. In many variants, these scaling factors are omitted since S can be considered to equal to the identify matrix in that case.

Although the mathematical roles of W and F are equivalent in the objective functions, they have different meanings. Hence, the parameters may have different meanings as latent components and new coefficients. Specifically, parameters $\alpha_1, \alpha_2, \beta_1, \beta_2$, γ_1, γ_2, etc are to control the quality of NMF including sparsity, smoothness, factor independence, etc. The roles of α_1 and α_2, β_1 and β_2 are equivalent for basis factors W and new coefficient F. However, the interpretability of γ_1 and γ_2 are different for W and F. γ_1 controls the orthogonality of basis vectors W^T, while γ_2 enhances the sparsity of F that leads to spectral clustering by the orthogonal unit constraints $F_i^T F_i = 1$ and $F_i^T F_{i'} = 0$, $\forall i, i', i \neq i'$ [34, 51, 78]. Usually, these quality parameters do not simultaneously appear in a NMF variant because the model may become thoroughly complicated over the requirements of real applications.

In real application, parameter variables can be omitted or added to construct simpler or more effective NMF models in real applications. In the next section, we review various customized models for many variants of NMF.

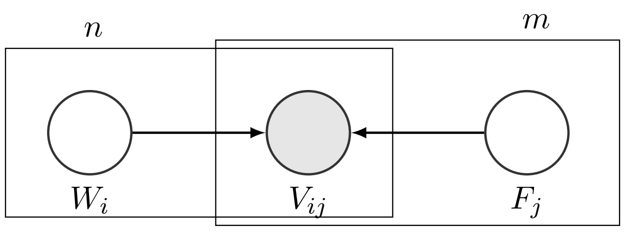

Fig. 2.2 Basic model for nonnegative matrix factorization

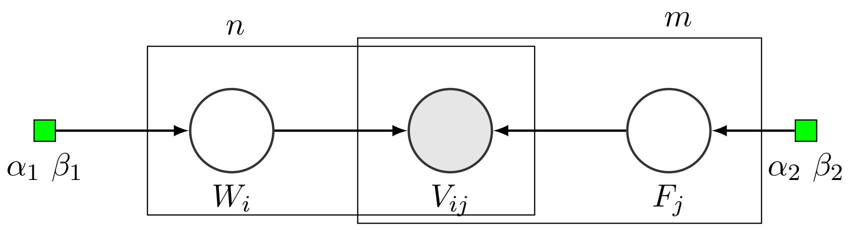

Fig. 2.3 L_1 and L_2 regularization model for nonnegative matrix factorization

2.2 Various NMF models and applications

For more a decade of development, NMF has many variants employed in numerous applications. In this section, we summarized several NMF models, which are widely applied in real applications.

In the most basic formulation as basic NMF [57], most the factors and the quality parameters are omitted, excepted W_i and F_j, see Figure 2.2. Many algorithms have been developed with different optimization strategies, namely inexact methods [46, 58, 63] and exact methods [35, 36, 44, 47, 49, 96]. Furthermore, NMF models with L_1 and L_2 regularizations are proposed to control sparsity and smoothness of NMF [36,44]. In this study, besides proposing fast algorithms, we generalize them to adapt with all the L_1 and L_2 NMF variants for Frobenius norm and KL-divergence with the general objective function $D(V\|W^TF) + \alpha_1\|W\|_1 + \beta_1\|W\|_2^2 + \alpha_2\|F\|_1 + \beta_2\|F\|_2^2$.

Furthermore, orthogonal NMF models are proposed to attain independence basis factors by adding orthogonal constraints on W [51], and spherical k-means by adding orthogonal constraints on F [78], see Figure 2.4.

In the previous NMF formulations, V is often not normalized, while scaling factors S is ignored since S equals to the identify matrix I. In addition, W and F are not normalized, which leads to unintended solutions that lacks of interpret-ability. Hence, several studies conduct nonnegative matrix tri-factorization $V \approx W^TSF$ with orthogonal conditions on W^T for orthogonality and F for clustering [27], see Figure 2.5. Although

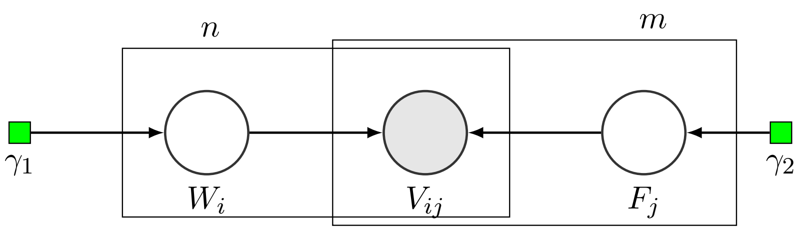

Fig. 2.4 Orthogonal model for nonnegative matrix factorization

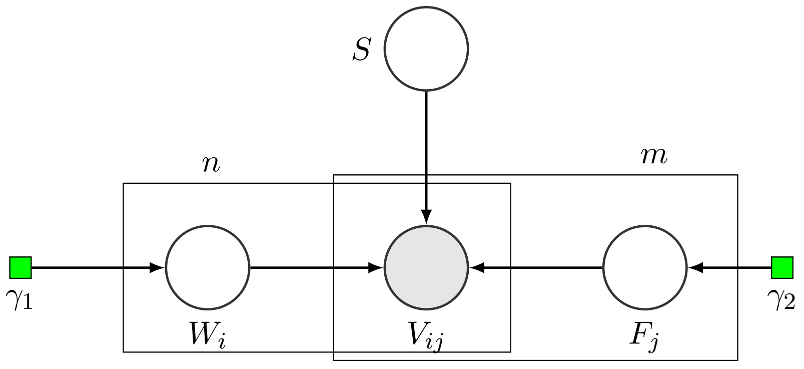

Fig. 2.5 Orthogonal nonnegative matrix tri-factorizations model for clustering

orthogonal conditions provides a poor matrix low-rank approximation, they give a concise interpretability of clustering and enhance factor independence.

2.3 Rich models for NMF

NMF is a non-convex problem and has stationary points of solutions. Hence, it is necessary to add more constraints into the basic NMF model based on prior knowledge to achieve rich models with more concise interpretability and high quality of sparsity, smoothness, and latent factor independence. In this thesis, we propose two rich models, namely NMF with L_1 L_2 regularizations (Figure 2.3) and simplicial NMF (Figure 2.6) for two popular divergences as Frobenius norm and KL divergence.

NMF with L_1 L_2 regularizations (Figure 2.3) generalizes models of the previous researches for NMF variants and simultaneously attains smoothness and sparsity. In comparison with the previous NMF formulations, NMF with L_1 L_2 regularizations enhance the sparsity and the smoothness of NMF in both latent components W and coefficients F. Specifically, the previous formulations lacks the generalization of algorithm to deal with all the possible cases of L_1 L_2 regularizations. In Chapter 3 and Chapter 4, we will propose fast parallel algorithms for all the NMF variants of L_1 L_2 regularizations for the

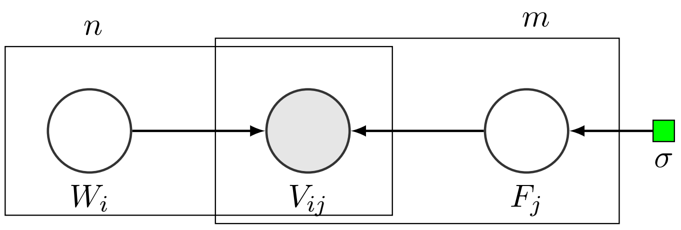

Fig. 2.6 Model for simplicial nonnegative matrix factorization

two most popular divergences as Frobenius norm and KL divergence.

Meanwhile, simplicial NMF (Figure 2.6) gives more concise interpretability of the role of latent components over data instances, in which each data instance is a probabilistic combination of latent components. The detailed formulations and proposed algorithms will be discussed in Chapter 5 and 6.

2.4 Conclusion

In summary, NMF is an opening flexible model, which can adapt with numerous applications. When studying on nonnegative matrix factorization, the model for NMF and the most suitable algorithm for the model should be carefully selected. In next chapters, we discuss the algorithm aspects for specific problems on nonnegative matrix factorization.

Chapter 3

Accelerated Parallel and Distributed Algorithm using Limited Internal Memory for NMF with L_1 L_2 Regularizations

Data is the new science. Big data holds the answers.
— Pat Gelsinger, CEO, EMC

Nonnegative matrix factorization (NMF) is a powerful technique for dimension reduction, extracting latent factors and learning part-based representation. For large datasets, NMF performance depends on some major issues: fast algorithms, fully parallel distributed feasibility and limited internal memory. This research aims to design a fast fully parallel and distributed algorithm using limited internal memory to reach high NMF performance for large datasets. Specially, we propose a flexible accelerated algorithm for NMF with all its L_1 L_2 regularized variants based on full decomposition, which is a combination of exact line search, greedy coordinate descent, and accelerated search. The proposed algorithm takes advantages of these algorithms to achieve over-bounded linear convergence rate of $\mathcal{O}[(1-\frac{\mu}{L})(1-\frac{\mu}{rL})^{2r}]^k$ in optimizing each factor matrix when fixing the other factor one in the sub-space of passive variables, where r is the number of latent components, and μ and L are bounded as $\frac{1}{2} \leq \mu \leq L \leq r$. In addition, the algorithm can exploit the data sparseness to run on large datasets with limited internal memory of machines. Furthermore, our experimental results are highly competitive with 7 state-of-the-art methods about three significant aspects of convergence, optimality and average of the iteration number. Therefore, the proposed algorithm is superior to fast block coordinate descent methods and accelerated methods.

3.1 Introduction

Nonnegative matrix factorization (NMF) is a powerful technique widely used in applications of data mining, signal processing, computer vision, bioinformatics, etc. [98]. Fundamentally, NMF has two main purposes. First, it reduces dimension of data making learning algorithms faster and more effective as they often work less effectively due to the curse of dimensionality [41]. Second, NMF helps extracting latent components and learning part-based representation, which are the significant distinction from other dimension reduction methods such as Principal Component Analysis (PCA), Independent Component Analysis (ICA), Vector Quantization (VQ), etc. This feature originates from transforming data into lower dimension of latent components and non-negativity constraints [29, 34, 58].

In the last decade of fast development, there were remarkable milestones. The two first milestones in early days of the NMF historical development were its mathematical formulations as positive matrix factorization with Byzantine algorithms [75] and as parts-based representation with a simple effective algorithm [58]. The last decade has witnessed the rapid NMF development [92, 98]. Various works on NMF can be viewed in three major perspectives: variants of NMF, algorithms and applications. In particular, variants of NMF are based on either divergence functions [83, 97], or constraints [43, 76], or regularizations [19, 59]. Most NMF algorithms were developed along two main directions: geometric greedy algorithms [90] and iterative multiplicative update algorithms. Although geometric greedy algorithms are usually fast, they are hard to trade off complexity, optimality, loss information and sparseness.

More recently, it is well recognized that the most challenging problems in iterative multiplicative update algorithms for NMF are fast learning, limited internal memory, parallel distributed computation, among others. In particular, fast learning is essential in learning NMF models from large datasets, and it is indeed difficult to carry out them when the number of variables is very large. In addition, the limited internal memory is one of the most challenging requirements for big data [38], because data has been exploring rapidly while the internal memory of nodes is always limited. Finally, parallel and distributed computation makes NMF applications feasible for big data [64].

To deal with these challenges, this work develops an accelerated algorithm for NMF and its L_1 L_2 regularized variants having several major advantages that are summarized in Table 3.1. The proposed approach has five significant properties as follows:

- *NMF and its variants*: We fully decompose NMF and its L_1 L_2 regularized variants

Table 3.1 Comparison Summary of NMF solvers

Criteria	Inexact		Exact				Accelerated			
	MUR	PrG	Qn	Nt	AcS	BlP	FCD	AcH	Ne	Alo
Guaranteed Convergence	✗	✗	✗	✗	✗	✗	✗	✗	✓ $\frac{1}{k^2}$	✓ $[(1-\frac{\mu}{L})(1-\frac{\mu}{rL})^{2r}]^k$
Exploit Data Spareness	✗	✗	✗	✗	✗	✗	✓	✓	✗	✓
Used Internal Memory	$\mathcal{O}(mn + r(r+n+m))$									$\mathcal{O}(r(r+n))$
Fully Parallel & Distributed	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓
Optimization Problem Size	$r(m,n)$			r	$r(n,m)$			(m,n)	$r(m,n)$	r

✓means considered, and ✗means not considered

$\frac{1}{2} \leq \mu \leq L \leq r$, $n \times m$ is the data matrix size, r is the number of latent components

$(m,n) = \max(m,n)$, and $r(m,n) = r.\max(m,n)$

Abbreviations: **MUR**: Multiplicative Update Rule [58]; **PrG**: Projected Gradient methods [63]; **Nt**: Newton-type methods [46]; **Qn**: Projected Quasi-Newton [96]; **AcS**: Fast Active-set-like method [47]; **BlP**: Block Principal Pivoting method [49]; **FCD**: Fast Coordinate Descent methods with variable selection [44]; **AcH**: Accelerated Hierarchical Alternating Least Squares [35]; **Nev**: Nesterov's optimal gradient method [36]; **Alo**: The proposed method.

into non-negative quadratic programming problems. This decomposition makes the proposed algorithm flexible to adapt all L_1 L_2 regularized NMF in a unified framework that can trade-off the quality of information loss, sparsity and smoothness.

• *Algortihm*: We employ an accelerated anti-lopsided algorithm in Chapter **??** for non-negative quadratic programming. The algorithm reduces variable scaling problems to achieve linear convergence rate of $\mathcal{O}([(1-\frac{\mu}{L})(1-\frac{\mu}{rL})^{2r}]^k)$ in optimizing each factor matrix in the sub-space of passive variables, where μ and L are bounded as $\frac{1}{2} \leq \mu \leq L \leq r$. This result is advanced to fast coordinate methods and accelerated methods in terms of efficiency as well as convergence rate. In addition, the size of optimization problem is reduced into r ($r \ll m, n$), which is the smallest among the state-of-the-art methods. Hence, the algorithm has the low complexity and converges very fast to the optimal solution, and it is highly potential to be applied in alternating least squares methods for factorization models.

• *Parallel and Distribution*: The proposed algorithms are fully parallel and distributed on limited internal memory systems, which is crucial for big data when computing nodes having limited internal memory that cannot hold the whole dataset.

• *Implementation*: The proposed algorithms are convenient to implement for hybrid multi-core distributed systems because this algorithm works on each individual instance and each latent feature.

• *Comparision*: This is the first time that state-of-the-art algorithms in different research directions for NMF are compared together.

The rest of chapter is organized as follows: Section 3.2 discusses the background and related works of NMF; Section 3.3 mentions our proposed algorithm; Section 3.4.2 gives a complexity analysis of our proposed algorithms; Section 3.5 experimentally compares our proposed algorithm with state-of-the-art algorithms for NMF among remarkable approaches; our conclusion is stated in Section 3.6.

3.2 Background and Related Works

3.2.1 Background

Mathematically, NMF in Frobenius norm is defined as follows:
Definition 1 [NMF]: *Given a dataset consisting of m vectors in a n-dimension space $V = [V_1, V_2, ..., V_m] \in R_+^{n\times m}$, where each vector presents a data instance. NMF seeks to decompose V into a product of two nonnegative factorizing matrices W and F, where $W = [W_1, ..., W_n] \in \mathcal{R}_+^{r\times n}$ and $F = [F_1, ..., F_m] \in \mathcal{R}_+^{r\times m}$ are the latent component matrix and the coefficient matrix respectively, $V \approx W^T F$, in which the quality of approximation can be guaranteed by the objective function in Frobenius norm: $D(V\|W^T F) = \|V - W^T F\|_2^2$.*

Although NMF is a non-convex problem, optimizing each factor matrix when fixing the other one is a convex problem. In other words, F can be traced when W is fixed, and vice versa. Furthermore, F and W have different roles although they are symmetric in the objective function. W are latent components to represent data instances V by coefficients F. Hence, NMF can be considered as a latent factor model of latent components W, and learning this model is equivalent to find out latent components W. Therefore, in this chapter, we propose an accelerated parallel and distributed algorithm to learn NMF models W for large datasets.

3.2.2 Related works

NMF algorithms can be divided into two groups: the greedy algorithms and the iterative multiplicative update algorithms. The greedy algorithms [90] are often based on geometric interpret-ability, and they can be extremely fast to deal with large datasets. However, it is hard to trade off complexity, optimality, loss information and sparseness. The iterative multiplicative update algorithms such as "two-block coordinate descent" often consist of two steps, each of them fixes one of two matrices to replace the other matrix for obtaining the convergence of the objective function. There are numerous studies on these algorithms, see Table 3.1, because NMF is nonconvex, though two steps corresponding to two non-negative least square (NNLS) sub-problems are convex [36,48]. In addition, various constraints and optimization strategies have been used to trade off the convexity, information loss, complexity, sparsity, and numerical instability.

Based on the optimization updating strategy, these iterative multiplicative update algorithms can be further divided into three sub-groups:

• *Inexact Block Coordinate Descent*: The algorithms' common characteristic is their usage of gradient methods to seek an approximate solution for NNLS problems, which is neither optimal nor fulfilling of fast approximations and accelerated conditions. Lee *et al.* [58] proposed the (basic) NMF problem and simple multiplicative updating rule (MUR) algorithm using first-order gradient method to learn the part-based representation. Seung *at al.* [83] concerned rescaling gradient factors with carefully selected learning rate to achieve a faster convergence rate. Subsequently, Lin [62] modified MUR, which is theoretically proved getting a stationary point. However, that algorithm cannot improve the convergence rate. Berry *et al.* [7] projected nonnegative least square (PNLS) solutions into nonnegative quadratic space by setting negative entries in the matrices to zero. Although this algorithm does not guarantee the convergence, it is widely applied in real applications. In addition, Bonettini *et al.* [12] used line search based on Amijo rule to obtain better solutions for matrices. Theoretically, this method can achieve optimal solutions for factor matrices as exact block coordinate descent group, but it very slowly tends to stationary points because the line search is time-consuming.

• *Exact Block Coordinate Descent*: In contrast to the first sub-group, the common characteristic in this group is obtaining optimal solutions for two NNLS problems in each iteration. Zdunek *et al.* [96] employed second-order quasi-Newton method with inverse of Hessian matrix to estimate the step size, aiming to a faster convergence than projected methods. However, this algorithm may be slow and non-stable because of the line search. Subsequently, Kim *et al.* [46] used rank-one to Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm to approximate the inverse of Hessian matrix. Furthermore, Chih-Jen Lin [63] proposed several algorithms based on projected gradient methods and exact line search. Theoretically, this method can obtain more accurate solutions, however it is time-consuming because of exact line search and the number of iterations increased by the large number of variables. Moreover, Kim *et al.* [47, 49] proposed two active-set methods based on Karush-Kuhn-Tucker (KKT) conditions, in which the variables are divided into two sets: a free set and an active set. Only the free set contains variables that can optimize the objective functions. Removing the number of redundant variables makes their algorithms improve the convergence rate significantly. However, the method still has heavy computation for large-scale problems.

• *Accelerated Block Coordinate Descent*: The accelerated methods use fast solution approximations satisfying accelerated conditions to reduce the complexity and to keep fast convergence. The accelerated conditions are different constraints in different methods to guarantee convergence to the optimal solution in comparison with the initial value. These accelerated methods are developed due to the limitation of inexact methods having slow convergence, and exact methods having high complexity in each iteration. Particularly, for inexact methods, they have slow convergence because of the high complexity of solution approximations in each iteration or a large number of iterations that leads to the highly expensive computation between two sequential iterations. Furthermore, the exact methods have high complexity in each iteration, however obtaining optimal solutions in every iteration is controversial because it can lead to zig-zag problems when optimizing a non-convex function of two independent sets of variables.

Firstly, Hsieh *et al.* [44] proposed a fast coordinate descent method with the best variable selection to reduce the objective function. The algorithm iteratively selects variables to update the approximate solution until the accelerated stopping condition $max_{ij} D^F_{ij} < \epsilon p_{\text{init}}$ satisfied, where D^F_{ij} is the reduction of the objective function based on the variable F_{ij}, and p_{init} is the maximum initial reduction over the matrix F. Although the greedy update method does not have guaranteed convergence, it has the fast convergence speed in many reports.

Subsequently, Gillis and Glineur [35] proposed a number of accelerated algorithms using fast approximation by fixing all variables but excepting a single column of factor matrices. This framework improved significantly the effectiveness of multiplicative updates [57], hierarchical alternating least squares (HALS) algorithms [21] and projected gradients [63]. These algorithms achieve the accelerated condition in each iteration such as that $\|F^{(k,l+1)} - F^{(k,l)}\|_2^2 \le \epsilon \|F^{(k,1)} - F^{(k,0)}\|_2^2$ is the stopping condition when optimizing the objective function on F if fixing W. Although these greedy algorithms does not have guaranteed convergence, their results are highly competitive with the inexact and exact methods.

More recently, Guan *et al.* [36] employed Nesterov's optimal methods to optimize NNLS with fast convergence rate $\mathcal{O}(1/k^2)$ to achieve the accelerated convergence condition $\|\frac{\partial f}{\partial F_{(k,l+1)}}\|_2^2 \le \epsilon \|\frac{\partial f}{\partial F_{(k,0)}}\|_2^2$. Although Guan *et al.*'s method [36] has a fast convergence rate $\mathcal{O}(1/k^2)$, it has several drawbacks such as working on the whole factor matrices, and less flexibility for regularized NMF variants. Furthermore, this approach does not consider the issues of parallel and distribution, and they require numerous iterations to satisfy the accelerated condition because the step size is limited by $\frac{1}{L}$, where L is Lipschitz constant.

To deal with the above issues of accelerated methods, in next section, we propose an accelerated parallel and distributed algorithm for NMF and its regularized L_1 L_2 variants with linear convergence in optimizing each factor matrix when fixing the other factor one.

3.3 Proposed Algorithm

To read easily, this section hierarchically presents our proposed algorithm. First, an iterative multiplicative update accelerated algorithm is introduced. Then, a transformational technique fully decomposes the objective functions of NMF into basic computation units as nonnegative quadratic programming (NQP) problems. After that, a modified version of the algorithm is proposed to deal with the issues of parallel and distributed systems. Subsequently, a combinational method of an anti-lopsided algorithm and a fast coordinate descent algorithm is developed to effectively solve NQP problems. Finally, extensions for L_1 L_2 regularized NMF is discussed.

3.3.1 Iterative multiplicative update accelerated algorithm

For solving NMF, we employ an iterative multiplicative update accelerated algorithm, like expectation-maximization (*EM*) algorithm, presented in Algorithm 1. This algorithm consists of two main steps: one for finding F^+ (F^+ is updated F in the iteration) when fixing W and the other for finding W^+ when fixing F. In the first step called *E-step*, we find F^+, each column of which F_i^+ is the new representation of a data instance V_i in the new space of latent components W. Meanwhile, the other one, called *M-step*, learns new latent components.

3.3.2 Full decomposition for NMF

This section discusses decomposing the objective function of NMF into non-negative quadratic programming (NQP) problems, which aims to fully parallelize and distribute the NMF computation. Particularly, in Algorithm 1, the E-step is to find new coordinates of data instances in the space of latent components W by minimizing $J(V\|W^TF) =$

Algorithm 1: Iterative Multiplicative Update Accelerated Algorithm

```
    Input: Data matrix $V = \{V_i\}_{i=1}^m \in \mathcal{R}_+^{n\times m}$ and $r$.
    Output: Latent components $W \in \mathcal{R}^{r\times n}$.
1   begin
2       Randomize $r$ nonnegative latent components $W \in R_+^{r\times n}$ ;
3       repeat
4           E-step: Fixing $W$ to find $F^+$ such that the accelerated condition is satisfied;
5           M-step: Fixing $F$ to find $W^+$ such that the accelerated condition is satisfied;
6       until Convergence condition is satisfied;
```

$\|V - W^TF\|_2^2 = \sum_{j=1}^m \|V_j - W^TF_j\|_2^2$. Hence, minimizing $J(V\|W^TF)$ is equivalent to independently minimizing $\|V_j - W^TF_j\|_2^2$ for each instance j since W is fixed. Similarly, the M-step is also equivalent to independently minimizing $\|V_i^T - F^TW_i\|_2^2$ for each feature i, where F is fixed. Hence, the basic computation units are nonnegative least-squares (NNLS) problems [56].

For large datasets $n, m \gg r$, we equivalently turn these problems into nonnegative quadratic programmings (NQP):

$$\begin{aligned} &\underset{x}{\text{minimize}} && \frac{1}{2}\|Ax - b\|_2^2 \\ &\text{subject to} && x \succeq 0 \in R^r \\ &\text{where} && A \in R_+^{nr}, b \in R_+^n \end{aligned} \tag{3.1}$$

equivalent to

$$\begin{aligned} &\underset{x}{\text{minimize}} && f(x) = \frac{1}{2}x^THx + h^Tx \\ &\text{subject to} && x \succeq 0. \\ &\text{where} && H = A^TA, h = -A^Tb \end{aligned} \tag{3.2}$$

Hence, finding new coefficients F^+ and new latent components W^+ can be fully paralleled and distributed into basic computation units as solving NQP problems.

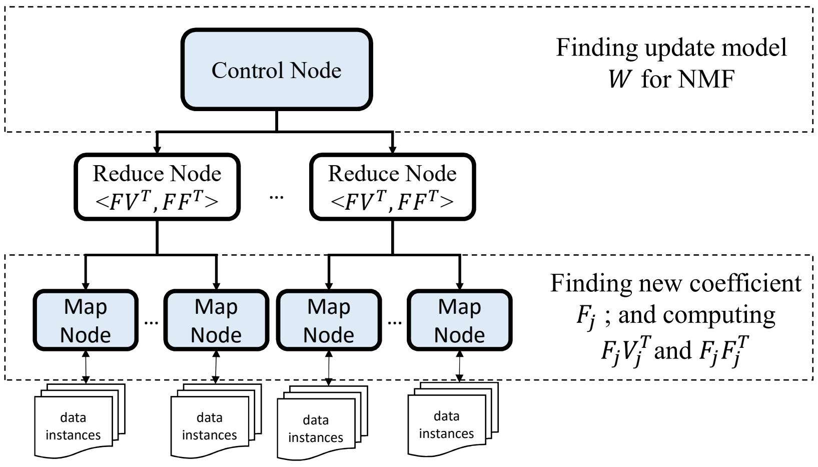
Control Node
Finding update model W for NMF
Reduce Node $<FV^T, FF^T>$
...
Reduce Node $<FV^T, FF^T>$
Map Node
Map Node
Map Node
Map Node
Finding new coefficient F_j ; and computing $F_jV_j^T$ and $F_jF_j^T$
data instances
data instances
data instances
data instances

Fig. 3.1 Distributed System Diagram for NMF

3.3.3 Parallel and distributed algorithm using limited internal memory

In this section, we design a parallel and distributed algorithm using limited internal memory for learning NMF model W, see Fig. 3.1, which is a modified version of Algorithm 1.

For large datasets, the computation can be not performed in a single process, so parallel and distributed algorithm environments are employed to speed up the computation. For parallel and distributed systems, we often face two major issues: dependency of computation units and limited internal memory computing nodes. In particular, computation units must be independently conducted as much as possible, since any dependency of computing elements will increase the complexity of implementation and the delay of data transfer over the network that reduces the performance of system. Furthermore, for these parallel distributed systems, computation units are executed on computing nodes within a limited internal memory. In addition, accessing external memory will increase the complexity and reduce the performance.

For our proposed approach, the computation can be fully paralleled and distributed, and use limited internal memory in computing nodes because the objective function is properly decomposed to NQP problems. Particularly, Algorithm 2 presents a modified version of iterative multiplicative update algorithms, in which computation units are fully paralleled and distributed. In addition, $Q = WW^T$ is precomputed to reduce the complexity, and finding new coefficients F_j can be independently computed and

Algorithm 2: Parallel and Distributed Algorithm

```
Input: Data matrix $V = \{V_j\}_{j=1}^{m} \in \mathcal{R}_+^{nm}$ and $r$.
Output: Latent components $W \in \mathcal{R}_+^{rn}$.
1  begin
2      Randomize $r$ nonnegative latent components $W \in \mathcal{R}_+^{rn}$;
3      repeat
4          $Y = 0 \in \mathcal{R}^{rn}$ /* $Y = FV^T$*/;
5          $H = 0 \in \mathcal{R}^{rr}$ /* $H = FF^T$ /*;
6          $Q = WW^T \in \mathcal{R}^{rr}$;
7          $maxStop = 0$;
8          /*Parallel and distributed*/;
9          for $j = 1$ to $m$ do
10             /*call Algorithm 3*/;
11             $F_j \approx \underset{x \in \mathcal{R}^r \succeq 0}{\operatorname{argmin}} \left(\frac{x^T Q x}{2} - V_j^T W^T x\right)$;
12             $Y = Y + F_j V_j^T$;
13             $H = H + F_j F_j^T$;
14         /*Parallel and distributed*/;
15         for $i = 1$ to $n$ do
16             /*call Algorithm 3*/;
17             $W_i \approx \underset{x \in \mathcal{R}^r \succeq 0}{\operatorname{argmin}} \left(\frac{x^T H x}{2} - Y_i^T x\right)$;
18     until Convergence condition is satisfied;
```

distributed. Remarkably, the most heavy computation of $Y = FV^T$ and $H = FF^T$ is divided into computing $F_jV_i^T$ and $F_jF_j^T$ to be parallel and distributed.

Particularly, the distributed system using MapReduce is described in Fig. 3.1. In this computing model, data instances and the instance projection are parallel and distributed over the Map nodes. The Reduce nodes sum up the results $F_jF_j^T$ and $F_jV_i^T$ of the Map nodes. Subsequently, in the M-step, the results FF^T and FV^T are employed to compute latent components W. This M-step computation can be conducted by a single machine or a distributed system, which depends on the dimension of problem because the time to distribute this computation over the network is usually considerable.

In comparison with the previous algorithms, this computing model is much more effective than the previous models [33, 64, 88] by the following reasons:

• The necessary memory used in computing nodes is $\mathcal{O}(size(W, Y, H)) = \mathcal{O}(r(r+n))$. The necessary memory used in the controlled node is $\mathcal{O}(size(W, Y, H, Q)) = \mathcal{O}(r(r+n))$. In practice, approximate solutions of NQP problems should be cached in hard disks in order to increase accuracy and reduce the number of iterations.

• At each distributed iteration, the computation is fully decomposed into basic computations units, which enhances the convergence speed to the optimal solution because the size of optimization is significantly reduced. Furthermore, the expensive computation FV^T and FF^T is fully parallelized and distributed over the computing nodes.

• The computational model is conveniently implemented because computing NMF model is divided into basic computation units as NQP problems that are independently solved, and the optimization is carried out on vectors instead of matrices.

In the next section, we propose a novel algorithm, Algorithm 3, to solve approximately NQP problems, which is robust and effective because it only uses the first derivative and does not consider the ill-condition of matrix inverse.

3.3.4 Fast algorithm for nonnegative quadratic programming

In this section, we briefly review the literature before proposing the novel algorithm to solve NQP Problem 3.2 for real large-scale NMF applications.

Regarding algorithms for NNLS and its equivalent problem NQP, numerous algorithms are proposed to deal with high dimension [18]. Generally, methods for solving NNLS can divided into two groups: active-set and iterative methods [18]. Active-set methods are traditional to solve accurately [15, 56]. However, they require heavy computation in repeatedly computing $(A^TA)^{-1}$ with different set of passive variables. Hence, iterative methods that can handle multiple active constraints in each iteration have more potential for fast NMF algorithms [18, 45, 50]. Hence, iterative methods can deal with more large-scale problems. Among the fast iterative methods, the coordinate descent method [31] has fast approximation, but has the zip-zag problem when the solution requires high accuracy. In addition, accelerated methods [72] has a fast convergence $\mathcal{O}(1/k^2)$ [36], which only require the first order derivative. However, one major disadvantage of the methods is that they require a big number of iterations because their step size is limited by $\frac{1}{L}$ that can be very small for large-scale problems; where L is Lipschitz constant. More recently, the accelerated anti-lopsided algorithm [73] re-scale variables to obtain fast linear convergence $\mathcal{O}([(1-\frac{\mu}{L})(1-\frac{\mu}{rL})^{2r}]^k)$ in the sub-space of passive variables, while the complexity of each iteration is kept in $\mathcal{O}(r^2)$.

Therefore, we employ the accelerated anti-lopsided algorithm having fast fast linear convergence to reduce the number of iterations. The proposed algorithm include four important parts to attain fast convergence:

- Part 1. Anti-lopsided transformation from Line 4 to Line 6: the variable vector x is transformed into a new space by $x = \varphi(y)$ as an inverse function. In the new space, the new equivalent objective function $g(y) = f(\varphi(y))$ has $\frac{\partial^2 g}{\partial y_i^2} = 1,\ \forall i$, or the acceleration of each variable equals 1. As a result, the role of variables become more balanced because the shape of the function becomes more spherical because $\frac{\partial^2 g}{\partial y_i^2} = 1,\ \forall i$, and $g(y)$ is convex. This part aims to make the post-processing parts more effective because it can implicitly exploit the second derivative information $\frac{\partial^2 g}{\partial y_i^2} = 1,\ \forall i$ to guarantee that μ and L are always bounded as $\frac{1}{2} \leq \mu \leq L \leq n$.

- Part 2: Exact line search from Line 11 to Line 13: this part optimizes the objective function with a guarantee of over-bounded convergence rate $\mathcal{O}((1-\frac{\mu}{L})^k)$ where $\frac{1}{2} \leq \mu \leq L \leq n$ over the space of passive variables, which has a complexity $\mathcal{O}(n^2)$. The part aims to reduce the objective functions exponentially and precisely, although it suffers from variable scaling problems and nonnegative constraints.

- Part 3. Greedy coordinate descent algorithm from Line 15 to Line 18 and repeated in Line 26: this part employs greedy coordinate descent using Gauss-Southwell rule with exact optimization to rapidly reduce the objective function with fast convergence $\mathcal{O}(1-\frac{\mu}{nL})$ for each update [71,80], which has a complexity of $\mathcal{O}(n^2)$. The part aims to reduce negative affects of variable scaling problems and nonnegative constraints, although it has zig-zagging problems because of optimizing the objective function over each single variable. Due to having fast convergence in practice and reducing negative affects of variable scaling problems and nonnegative constraints, this part is repeated one more time after Part 4.

- Part 4. Accelerated search from Line 22 to Line 25: This step performs a momentum search based on previous changes of variables in Part 2 and Part 3, which has a low complexity of $\mathcal{O}(n.nn(n))$ where $nn(n)$ is the number of negative elements in $(x_{k+1} - \alpha\triangle x)$, see Line 25 in Algorithm 3. This part relies on the global information of two distinct points to escape the local optimal information issues of the first derivative raised by the function complexity. This part originates from the idea that if the function is optimized from x_s to x_k by the exact line search and the coordinate descent algorithm, it is highly possible that the function value will be reduced along the vector $(x_k - x_s)$ because the NNLS objective function is convex and has (super) eclipse sharp.

Particularly, the proposed algorithm, Algorithm 3, contains two main steps: The first step, from Line 4 to Line 6, rescales variables to avoid rescaling problems of the first order methods by replacing $y = x. * \sqrt{diag(H)}$, we have:

$$f(x) = \frac{1}{2}x^T Hx + h^T x = \frac{1}{2}y^T Qy + q^T y \tag{3.3}$$

where $Q = \frac{H}{\sqrt{diag(H)diag(H)^T}}$ and $q = \frac{h}{\sqrt{diag(H)}}$ such that $\frac{\partial^2 f}{\partial^2 y_i} = Q_{ii} = \frac{H_{ii}}{\sqrt{H_{ii}H_{ii}}} = 1$ for $\forall i$. By the way, the rate of change of a quantity through variables equals to a constant to guarantee that the convex parameter μ and Lipschitz constant L are always bounded as $\frac{1}{2} \le \mu \le L \le r$.

The second step contains a loop of iterations, from Line 8 to Line 27, which combines three algorithms including exact line search, greedy coordinate descent, and accelerated search. The accelerated anti-lopsided algorithm guarantees the linear convergence $\mathcal{O}([(1-\frac{\mu}{L})(1-\frac{\mu}{rL})^{2r}]^k)$ in the sub-space of passive variables to avoid the zip-zag problem of the fast coordinate descent algorithm, while the coordinate block descent algorithm speeds up the convergence to the final optimal set of passive variables. The passive variables are variables belongs the set $P = \{x_i | x_i > 0 \ or \ \nabla f_i < 0\}$ that changes through iterations.

In addition, the complexity of each part is still kept in $\mathcal{O}(r^2)$. As a result, the proposed algorithm will utilize advantages of various algorithms to attain a fast convergence, while retaining the same low complexity $\mathcal{O}(r^2)$ of each iteration.

To comprehend the proposed algorithm's effectiveness, we consider optimizing Func-

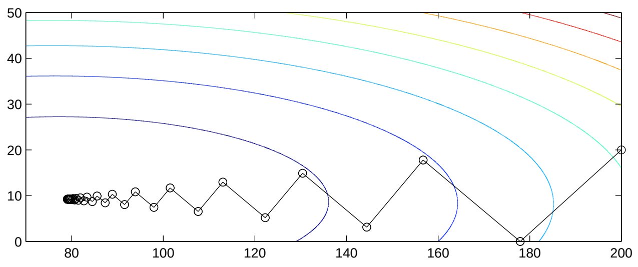

Fig. 3.2 59 optimizing steps in iterative exact line search method using the first order derivative for the function 3.4 starting at $x_0 = [200\ 20]^T$

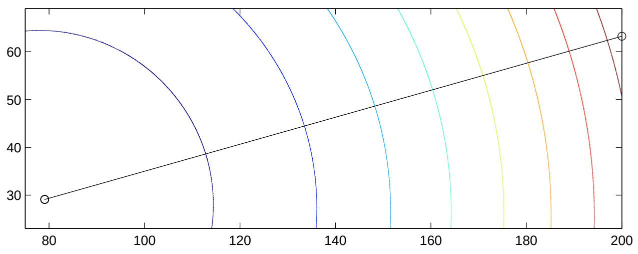

Fig. 3.3 1 optimizing steps in iterative exact line search method using the first order derivative for the function 3.5 starting at $y_0 = x_0\sqrt{diag(H)}$

tion 3.4:

$$f(x) = \frac{1}{2}x^T \begin{bmatrix} 1 & 0.1 \\ 0.1 & 10 \end{bmatrix} x + [-80 - 100]x \tag{3.4}$$

The exact search gradient algorithm, from Line 11 to Line 13, starting with $x_0 = [200\ 20]^T$ performs 59 iterations to reach the optimal solution, see Fig. 3.2. However, the proposed algorithm only needs 1 iterations to reach the optimal solution, see Fig. 3.3 because we optimize Function 3.5 instead of Function 3.4; where Function 3.5 is equivalently obtained by applying the steps from Line 11 to Line 13. The exact search gradient algorithm becomes much faster because the shape of Function 3.5 become more sphere, and its derivative is more effective to optimize the objective function.

$$f(y) = \frac{1}{2}y^T \begin{bmatrix} 1 & \frac{0.1}{\sqrt{10}} \\ \frac{0.1}{\sqrt{10}} & 1 \end{bmatrix} y + [\frac{-80}{\sqrt{10}} \quad \frac{-100}{\sqrt{10}}]y \tag{3.5}$$

Moreover, Algorithm 3 only attains approximate solutions because achieving the optimal solution is controversial for the reasons that its computation is expensive and it can leads to the zig-zag problem in optimizing a non-convex function. In addition, it is necessary to control and balance the quality of the convergence to the optimal solution.

Algorithm 3: Fast Combinational Algorithm for NQP

```
    Input: $H \in \mathcal{R}^{r \times r}$ and $h \in \mathcal{R}^r$ and $x_0$
    Output: $x \approx \underset{x \succeq 0}{\text{argmin}}\ \frac{1}{2}x^T Hx + h^T x$
1  begin
2      /*Having a variable $maxStop = 0$ for each thread of computation */;
3      /*Re-scaling variables*/;
4      $Q = \frac{H}{\sqrt{diag(H)diag(H)^T}}$; $q = \frac{h}{\sqrt{diag(H)}}$;
5      /*Solving NQP: minimizing$f(x) = \frac{1}{2}x^T Qx + q^T x$*/;
6      $x = x_0.*\sqrt{diag(H)}$;
7      $\nabla f = Qx + q$;
8      repeat
9          $x_s = x_{k-1}$ and $\nabla f_s = \nabla f$ ;
10         /*Exact line search over passive variables*/;
11         $\nabla\bar{f} = \nabla f$; and $\nabla\bar{f}[x = 0$ and $\nabla f > 0] = 0$;
12         $\alpha = \arg\underset{\alpha}{\min}\ f(x_k - \alpha\nabla\bar{f}) = \frac{\|\nabla\bar{f}\|_2^2}{\nabla\bar{f}^T Q\nabla\bar{f}}$;
13         $x_k = x_{k-1} - \alpha\nabla\bar{f}$; $\nabla f_k = \nabla f_k - \alpha Q\nabla\bar{f} - Q[x_k]_-$; $x_k = [x_k]_+$;
14         /*Greedy coordinate descent algorithm*/;
15         for $t$=1 to $n$ do
16             $p = \underset{i \in P(x)}{\text{argmax}}\ |\nabla_i f(x_k)|$;
17             $\triangle x_p = max(0, [x_k]_p - \frac{\nabla_p f}{Q_{pp}}) - [x_k]_p$;
18             $\nabla f = \nabla f + Q_p\triangle x_p$; $[x_k]_p = [x_k]_p + \triangle x_p$;
19         if $(\|\tilde{f}_k\|_2^2 \leq \epsilon\|\tilde{f}_0\|_2^2)$ or $(\|\tilde{f}_k\|_2^2 \leq maxStop)$ then
20             break;
21         /*Accelerated search carries a "momentum" based on the changes of
           variables in exact line search and greedy coordinate descent part*/;
22         $\triangle x = x_s - x_k$ /*$x_s$ and $\nabla f_s$ are assigned in Line 9*/;
23         $\alpha = \underset{\alpha}{\text{argmin}}\ f(x_k - \alpha\triangle x) = \frac{\nabla f^T\triangle x}{\triangle x^T Q\triangle x} = \frac{\nabla f^T\triangle x}{\triangle x^T(\nabla f_s - \nabla f)}$;
24         $x_k = x_k - \alpha\triangle x$;
25         $\nabla f = \nabla f - \alpha Q\triangle x - Q[x_k]_-$; $x_k = [x_k]_+$;
26         Repeat steps in the part of greedy coordinate descent algorithm;
27     until $(\|\tilde{f}_k\|_2^2 \leq \epsilon\|\tilde{f}_0\|_2^2)$ or $(\|\tilde{f}_k\|_2^2 \leq maxStop)$;
28     $maxStop = max(maxStop, \|\tilde{f}_k\|_2^2)$;
29     return $\frac{x_k}{\sqrt{diag(H)}}$
```

Hence, we employ an accelerated condition ($\|\tilde{f}_k\|_2^2 \leq \epsilon\|\tilde{f}_0\|_2^2$) to regulate the quality of the convergence to the optimal solutions of the NQP problems in comparison with initial values and a fast-break condition ($\|\tilde{f}_k\|_2^2 \leq maxStop$) to balance the quality of the convergence among variables in each thread of the computation. As a result, the objective function converges faster through iterations; and the complexity and average of the iteration number are reduced significantly.

3.3.5 Extensions for L_1 L_2 regularized NMF

In this section, we consider solutions for L_1 L_2 regularized NMF variants to control the quality of NMF. L_1 regularized NMF [42] aims to achieve sparse solutions in optimization problems. Usually, only the coefficient matrix F is penalized to control its sparsity. Meanwhile, concerning L_2 regularized NMF, the penalty terms of F and W are added to control smoothness of solutions in NMF [77]. Fortunately, the objective functions of L_1 L_2 regularized NMF can be turned into NQP problems, of which solutions are completely similar to the general NMF. Particularly, in the most general variant, the objective function $J(V\|W^TF)$ is formulated by:

$$\|V - W^TF\|_2^2 + \mu_1\|F\|_1 + \beta_1\|W\|_1 + \mu_2\|F\|_2^2 + \beta_2\|W\|_2^2 \tag{3.6}$$

where $\|.\|_1$ is the L_1-norm, $\|.\|_2$ is the L_2-norm, and $\mu_1, \mu_2, \beta_1, \beta_2$ are regularized parameters that tradeoff the sparsity and the smoothness of the information loss. Obviously, both the E-step and the M-step need to solve the same NNLS problems when one of the two matrices is fixed. For example, in a E-step, we can minimize the objective function by independently solving NQP problems when fixing W:

$$\begin{aligned} J(V\|W^TF) &= \frac{1}{2}\|V - W^TF\|_2^2 + \mu_1\|F\|_1 + \mu_2\|F\|_2^2 + \mathrm{C} \\ &= \sum_{j=1}^{m}(\frac{1}{2}\|V_j - W^TF_j\|_2^2 + \mu_1(1^K)^TF_j + \mu_2F_j^TIF_j) + \mathrm{C} \\ &= \sum_{j=1}^{m}(\frac{1}{2}F_j^TQF_j + q^TF_j) + \mathrm{C} \end{aligned} \tag{3.7}$$

where $Q = WW^T + 2\mu_1 I$, $q^T = -WV_j + \mu_2 1^K$ and C is a constant.

This transformation from minimizing the objective functions into solving NQP problems independently is comprehensive to understand and simplify the variants of NMF problems as much as possible. As a result, we can conveniently implement NMF and its L_1 L_2 regularized variants in parallel distributed systems as in sub-section 3.3.3.

In comparison with the previous algorithms that optimizing the objective function works on the whole of matrices, this approach decomposing the objective function is easier to parallelize and distribute the computation. Additionally, it is faster to reach the solutions because it only performs on a smaller set of variables.

3.4 Theoretical Analysis

In this section, we investigate the convergence of Algorithm 3 and the complexity of Algorithm 2 using Algorithm 3

3.4.1 Convergence

In this section, we only consider the convergence rate of Algorithm 3 for the general NMF for the two following reasons. Firstly, L_1 regularized coefficients do not affect on the complexity. Secondly, L_2 regularized coefficients are often small, and they change Lipschitz constants μ and L by adding a small positive value, where μ and L are the convex parameter and the Lipschitz constant of strongly convex function $f(x)$ satisfying $\mu I \preceq \frac{\partial^2 f}{\partial^2 x} \preceq LI$ and I is the identity matrix. Hence, L_2 regularized coefficients slightly change the convergence rate because it depends on $\frac{\mu}{L}$.

Based on [73], consider the complexity of Algorithm 3, we have:

Theorem 1 *The exact line search in Algorithm 3 linearly converges at the rate of* $\mathcal{O}(1-\frac{\mu}{L})^k$ *in the sub-space of passive variables, where* $\frac{1}{2} \leq \mu \leq L \leq r$*,* r *is the dimension of solutions or the number of latent factors, and* r *is the number of iterations.*

Proof:

From [73], we have:

Remark 1: After $(k+1)$ iterations, $f(x^{k+1}) - f^* \leq (1-\frac{\mu}{L})^k (f(x^0) - f^*)$, where $\mu I \preceq \nabla^2 f \preceq LI$, f^* is the minimum value of $f(x)$, and $f(x)$ is a strongly convex function of the passive variables.

We have $\nabla^2 f = Q$, and

$\frac{1}{2}x^T I x \leq \sum_{i=1}^{r}\sum_{j=1}^{r} Q_{ij}x_i x_j = x^T Q x, \quad \forall x$ since $Q_{ij} = \cos(W_i^T, W_j^T)$, and $Q_{ii} = 1$. $\Rightarrow \frac{1}{2} I \preceq Q$.

Moreover, based on Cauchy-Schwarz inequality, we have:

$$\begin{aligned}
&(\sum_{i=1}^{r}\sum_{j=1}^{r} Q_{ij}x_ix_j)^2 \leq (\sum_{i=1}^{r}\sum_{j=1}^{r} Q_{ij}^2)(\sum_{i=1}^{r}\sum_{j=1}^{r}(x_ix_j)^2) \\
\Rightarrow &\sum_{i=1}^{r}\sum_{j=1}^{r} Q_{ij}x_ix_j \leq \sqrt{\|Q\|_2^2(\sum_{i=1}^{r} x_i{}^2)^2} \\
\Leftrightarrow &x^TQx \leq \|Q\|_2 x^TIx \quad (\forall x) \quad \Leftrightarrow Q \preceq \|Q\|_2 I
\end{aligned}$$

Finally, $\sqrt{r} = \sqrt{\sum_{i=1}^{r} Q_{ii}^2} \leq \|Q\|_2 = \sqrt{\sum_{i=1}^{r}\sum_{j=1}^{r} Q_{ij}^2} \leq \sqrt{r^2} = r$ since $-1 \leq Q_{ij} = \cos(W_i^T, W_j^T) \leq 1$. Therefore, we have:

Remark 2: $\frac{1}{2} \leq \mu \leq L \leq r$.

From Remark 1 and Remark 2, we have Theorem 1. □

Actually, the exact line search step, from Line 11 to Line 13 in Algorithm 3, guarantees linear convergence of $\mathcal{O}(1-\frac{\mu}{L})^k$ in the sub-space of passive variables. However, the set of passive variables changes through iterations. Hence, we employ $2r$ times of the greedy coordinate descent update, which also has fast convergence $\mathcal{O}([(1-\frac{\mu}{2L})^{2r}]^k)$ [71, 80]. Hence, the greedy coordinate descent algorithm rapidly restricts the domain of solution to converge to the final optimal sub-space of passive variables of the solution. Hence, the proposed algorithm linearly converges of $\mathcal{O}([(1-\frac{\mu}{L})(1-\frac{\mu}{rL})^{2r}]^k)$, where μ and L are bounded as $\frac{1}{2} \leq \mu \leq L \leq r$. Therefore, the convergence rate is over-bounded $[(1-\frac{\mu}{L})(1-\frac{\mu}{rL})^{2r}]^k \leq [(1-\frac{1}{2n})(1-\frac{1}{2rn})^{2r}]^k$. Hence, we have:

Theorem 2 *Algorithm 3 has over-bounded linear convergence rate of* $\mathcal{O}([(1-\frac{\mu}{L})(1-\frac{\mu}{rL})^{2r}]^k)$ *in the sub-space of passive variables, where* $\frac{1}{2} \leq \mu \leq L \leq r$, $(1-\frac{\mu}{L})(1-\frac{\mu}{rL})^{2r} \leq (1-\frac{1}{2n})(1-\frac{1}{2rn})^{2r}$, *$r$ is the dimension of solutions or the number of latent factors, and k is the number of iterations.*

3.4.2 Complexity

In this section, we analyze the complexity of Algorithm 2 using Algorithm 3 to solve NQP problems. If we assume that the complexity for each iteration contains $\mathcal{O}(nr^2)$ in computing $Q = WW^T$, $\mathcal{O}(mnr)$ in computing $Y = VF^T$, $\mathcal{O}(mr^2)$ in computing $H = FF^T$, $\mathcal{O}(\overline{k}mr^2)$ in computing F and $\mathcal{O}(\overline{k}nr^2)$ in computing W, where $\overline{k}$ is the number of iterations, then we have the following Lemma 3:

Theorem 3 *The complexity of each iteration in Algorithm 2 using Algorithm 3 to solve NQP problems is* $\mathcal{O}((m+n)r^2 + mnr + \overline{k}(m+n)r^2)$. *In addition, it is* $\mathcal{O}((m+n)r^2 +$

$rS(mn) + \bar{k}(m+n)r^2)$ for sparse data, where $S(mn)$ is the number of non-zero elements in data matrix V.

Theorem 3 is significant for big data, because the data is usually big and sparse. In other words, mn is actually large, but $S(mn)$ is small; so $mn \gg (m+n)r^2 + rS(mn) + \bar{k}(m+n)r^2$. Hence, in experimental evaluation Section 3.5, we prove that our algorithm can run on large high-dimension sparse datasets such as Nytimes for an acceptable time. In that dataset, $mnr \gg rS(mn) \gg (m+n)r^2$, so the running time $T(m,n,r) \approx rS(mn)$ since $m, n \gg r$.

Moreover, Table 3.2 shows a comparison of the complexity in an iteration of our proposed algorithms (Alo) with other state-of-the-art algorithms' in the literature: Multiplicative Update Rule (MUR) [57], Projected Nonnegative Least Squares (PrN) [7], Projected Gradient (PrG) [63], Projected Quasi-Newton (PQN) [96], Active Set (AcS) [47], Block Principal Pivoting (BlP) [49], Accelerated Hierarchical Alternating Least Squares (AcH) , Fast Coordinate Descent Methods with Variable Selection (FCD) [44], and Nesterov's Optimal Gradient Method (Nev) [36]. It can be seen that the complexity of our proposed algorithm is highly comparable with that of other algorithms, and the speed of algorithms depend on the number of iterations. In the experimental evaluation, we will show that the iteration number of our algorithm is highly competitive with other algorithms'. Remarkably, moreover, our proposed algorithm has the following properties that other algorithms has yet considered:

- Exploit the sparseness of datasets,
- Runnable for big datasets in limited internal memory systems,
- Convenient to implement in fully paralleled and distributed systems.

3.5 Experimental Evaluation

In this section, we investigate the effectiveness of the proposed algorithm **Alo** by comparing it to 7 carefully selected state-of-the-art NMF solvers belongs to different approaches:

- **MUR**: Multiplicative Update Rule [58],
- **PrG**: Projected Gradient Methods [63],

Table 3.2 Complexity of an iteration in NMF solvers

Solver	**Complexity ($\mathcal{O}$)**
MUR [57]	$mnr + (m+n)r^2$
PrN [7]	$mnr + (m+n)r^2 + r^3$
PrG [63]	$(m+n)r^2 + rmn + \overline{k}\overline{t}(m+n)r^2$
PQN [96]	$\overline{k}(mnr + m^3r^3 + n^3r^3)$
BlP [49]	$(m+n)r^2 + mnr + \overline{k}(m+n)r^2$
AcS [47]	$(m+n)r^2 + rmn + \overline{k}(m+n)r^2$
FCD [44]	$(m+n)r^2 + rS(mn) + \overline{k}(m+n)r^2$
AcH [35]	$(m+n)r^2 + rS(mn) + \overline{k}(m+n)r^2$
Nev [36]	$(m+n)r^2 + mnr + \overline{k}(m+n)r^2$
Alo	$(m+n)r^2 + rS(mn) + \overline{k}(m+n)r^2$

where m, n is the matrix size, r is the number of latent components, $\overline{k}$ is the average number of iterations, $\overline{t}$ is the average number of internal iterations, and $S(mn)$ is the number of non-zero elements of data matrix V. To easily compare among the algorithms, we consider r update times for Algorithm FCD as one iteration because the complexity of one update is $\mathcal{O}(r)$, while the complexity of one iteration in other accelerated algorithms is $\mathcal{O}(r^2)$.

Table 3.3 Dataset Information

Data-sets	m	n	r	MaxIter
Faces	6977	361	60	300
Digits	6.10^4	784	80	300
Tiny Images	5.10^4	3,072	100	300
Nytimes	3.10^5	102,660	100,...,200	300

- **BlP**: Block Principal Pivoting method [49],
- **AcS**: Fast Active-set-like method [47],
- **FCD**: Fast Coordinate Descent methods with variable selection [44],
- **AcH**: Accelerated Hierarchical Alternating Least Squares [35],
- **Nev**: Nesterov's optimal gradient method [36].

Test cases: In this experiment, we design two tests using four datasets shown in Table 3.3. In the first test, 3 typical datasets with different sizes are used: Faces[1], Digits[2] and Tiny Images [3]. For these tests, the algorithms are compared in terms of convergence, optimality, and average of the iteration number to investigate their performance and effectiveness. Additionally, average of the the iteration number $\bar{k}$ for approximate solutions of the sub-problems as NNLS or NQP is to compare the complexity of algorithms. In the second test, a large dataset containing tf-idf values computed from the text dataset Nytimes[4] is used to verify the performance and the feasibility of our parallel algorithms on sparse large datasets.

Environment settings: To be fair in comparison, for the first test, the programs of compared algorithms are written in the same language Matlab 2013b, run by the same computer Mac Pro 8-Core Intel Xeon E5 3 GHz RAM 32 GB, and initialized by the same factor matrices W_0 and F_0. The maximum number of threads is set to 10 while keeping 2 threads for other tasks in the operation system. For the second test, the proposed algorithm is written in Java programming language to utilize the data sparseness.

[1]http://cbcl.mit.edu/cbcl/software-datasets/FaceData.html
[2]http://yann.lecun.com/exdb/mnist/
[3]http://horatio.cs.nyu.edu/mit/tiny/data/index.html
[4]https://archive.ics.uci.edu/ml/machine-learning-databases/bag-of-words/

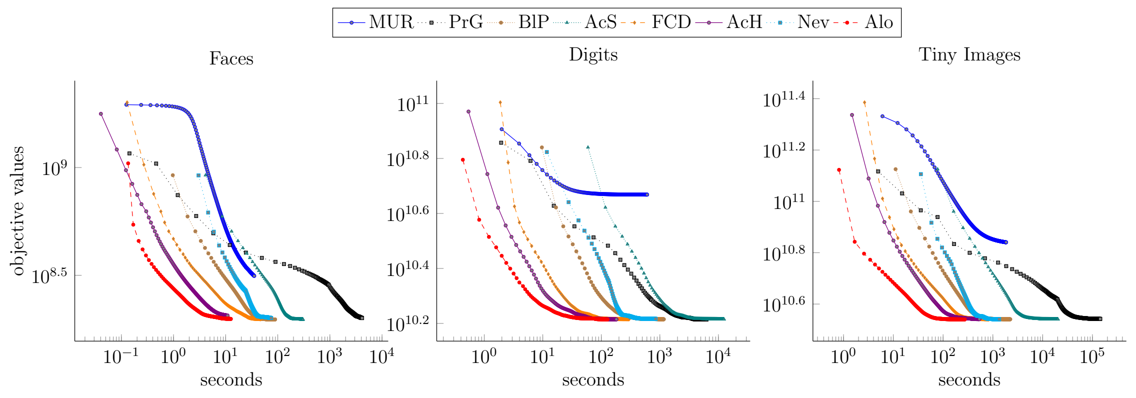

Fig. 3.4 Objective function values $\|V - W^T F\|_2^2/2$ versus CPU seconds for datasets: Faces, Digits, and Tiny Images

Source code: The source codes of **MUR**, **PrG**, **BlP**, **AcS**, **FCD**, **AcH**, and **Nev** are downloaded from [5], [6], [7], [8], and [9]. For convenient comparison in the future, we publish all the source codes and datasets in [10].

3.5.1 Convergence

In this experiment, we investigate the convergence of algorithms by information loss $\frac{1}{2}\|V - W^T F\|_2^2$ in terms of time and the iteration number. In terms of time, see Fig. 3.4, the proposed algorithm Alo is remarkably faster than the other algorithms for the three different-size datasets: Faces, Digits and Tiny Images. Especially, for the largest dataset Tiny Images, the distinction between the proposed algorithm and the runner-up algorithm AcH is easily recognized. Furthermore, in terms of the iteration number, see Fig. 3.5, the proposed algorithm converges to the stationary point of solutions faster than the others. This observation is clear for large datasets as Digits and Tiny Images. The results are significant in learning NMF models for big data because the proposed algorithm not only converges faster but also uses a less number of iterations, and the time of reading and optimization through a big dataset is actually considerable.

[5] `http://www.cs.toronto.edu/~dross/code/nnmf.m`
[6] `https://github.com/kimjingu/nonnegfac-matlab`
[7] `http://www.csie.ntu.edu.tw/~cjlin/nmf/`
[8] `http://dl.dropboxusercontent.com/u/1609292/Acc_MU_HALS_PG.zip`
[9] `https://sites.google.com/site/nmfsolvers/`
[10] `https://bitbucket.org/aaaaaa/matlabnmf`

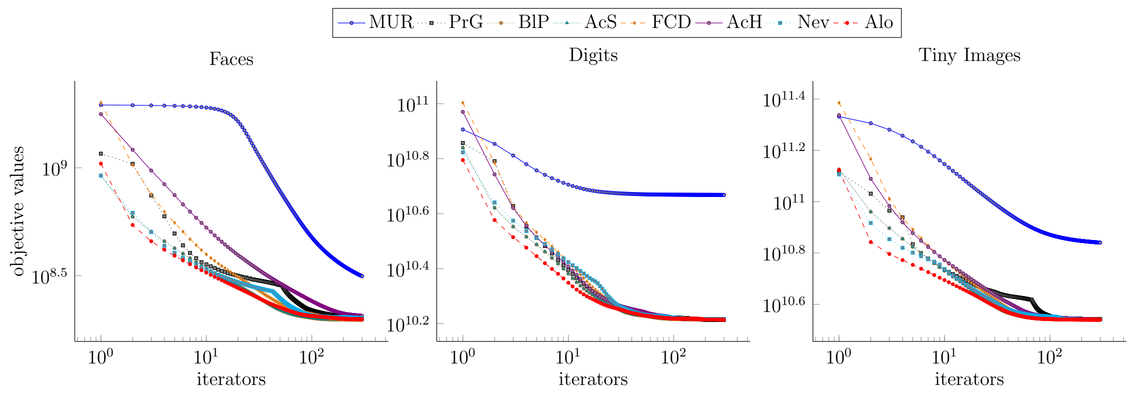

Fig. 3.5 Objective function values $\|V - W^T F\|_2^2/2$ in terms of the iteration number for datasets: Faces, Digits, and Tiny Images

Table 3.4 Optimal Values of NMF solvers

Dataset	MUR	PrG	BlP	AcS	FCD	AcH	Nev	Alo
Faces ($\times 10^8$)	3.142	2.003	**1.975**	**1.975**	1.983	2.058	2.003	1.985
Digits ($\times 10^{10}$)	4.659	1.639	1.641	1.641	1.644	1.640	1.646	**1.639**
Tiny Images ($\times 10^{10}$)	6.925	3.483	3.472	3.472	3.474	3.476	3.473	**3.467**

3.5.2 Optimality

After more a decade of rapid development, numerous algorithms have been proposed for solving NMF as a fundamental problem in dimension reduction and learning representation. Currently, the difference of the final loss information $\|V - W^T F\|_2^2$ among the state-of-the-art methods is inconsiderable in comparison to the square of information $\|V\|_2^2$. However, the small difference represents the effectiveness of the optimization methods because NMF algorithms often slowly converge when the approximate solution is close to the optimal local solution. Hence, in Table 3.4, the final values of the objective function $\frac{1}{2}\|V - W^T F\|_2^2$ investigate the optimality and the effectiveness of the optimization methods. Noticeably, Algorithm AcH fast converges over time and has a low average of the iteration number, but it has the optimal values much higher than the proposed algorithm because it uses a time-break technique to interrupt the optimization algorithm. In addition, the proposed algorithm achieves the best optimality for two largest of three datasets. This result additionally represents the robustness of the proposed method, which is highly competitive with the state-of-the-art methods.

Table 3.5 Average of Iteration Number $\bar{k}$

Dataset	MUR	PrG	BlP	AcS	FCD	AcH	Nev	Alo
Faces	1.00	321.12	1116.96	102.09	1.54	1.11	29.21	**1.01**
Digits	1.00	36.70	12503.75	305.94	1.00	1.05	23.36	**1.00**
Tiny Images	1.00	767.45	12869.12	1086.51	1.38	2.52	29.32	**1.02**

3.5.3 Average of iteration number

In this section, we investigate the complexity of the NMF solvers by average of the iteration number $\bar{k} = \frac{\text{number of internal iteration}}{\text{MaxIter} \times (m+n)}$ for approximate solutions of sub-problems as NNLS or NQP because the complexity of algorithms mainly depends on this number, see Table 3.2. Except for the original algorithm MUR with one update having the worst result, the proposed algorithm Alo employs at least average of the iteration number, see Table 3.5, especially for large datasets. In addition, the proposed algorithm does not employ any tricks to timely interrupt before one of the stopping conditions is satisfied, while the highly competitive algorithm AcH uses. Therefore, this result clearly represents the fast convergence of Algorithm 3 as it is verified by a large number of NQP problems.

3.5.4 Running on large datasets

In this section, we verify the feasibility of the proposed algorithm in learning NMF model for large datasets. Particularly, the proposed algorithm is implemented by Java programming language to exploit the data sparseness. Additionally, it runs on the large sparse text dataset Nytimes with different numbers of latent components, see Table 3.3. Interestingly, the proposed algorithm can run with hundreds of latent components by a single computer in an acceptable time.

Fig. 3.6 shows the performance of our algorithm running on the large sparse dataset Nytimes. Remarkably, the proposed algorithm only uses about 1 iteration on average to satisfy the accelerated condition of approximate solutions. Furthermore, the average of iteration time in learning NMF model linearly increases through the different numbers of latent components. This result totally fits the complexity analysis when $rnm \gg rS(mn) \gg (m+n)r^2 + \bar{k}(m+n)r^2$, so the complexity $T(m,n,r) \approx rS(mn)$ since $m, n \gg r$. Additionally, the objective function converges to the stationary point at about the 100th iteration within the different numbers of latent components r, which is the same with the previous datasets.

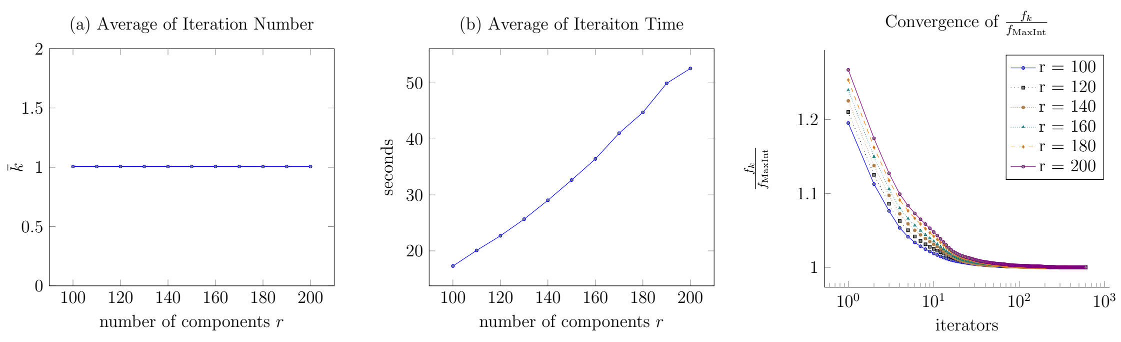

Fig. 3.6 average of the iteration number $\bar{k}$, average of iteration time, and convergence of $\frac{f_k}{f_{\text{MaxInt}}}$ in learning NMF model for the dataset Nytimes within the different numbers of latent components

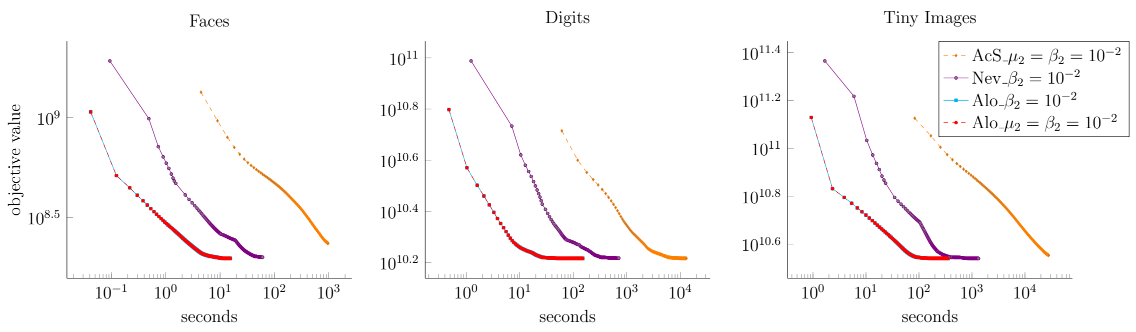

Fig. 3.7 Convergence of regularized NMF Extensions for algorithms AcS, Nev and Alo within two regularized cases: $\mu_2 = 10^{-2}$ and $\mu_2 = \beta_2 = 10^{-2}$

3.5.5 Regularized NMF extensions

In this section, we investigate the convergence of algorithms for regularized NMF extensions on three datasets: Faces, Digits, and Tiny Images. Due to the lack of available codes and the L_1 L_2 generalization of the other algorithms, only three algorithms AcS, Nev and Alo are compared within two regularized cases: $\mu_2 = 10^{-2}$ and $\mu_2 = \beta_2 = 10^{-2}$, see Fig. 3.7. In comparison with other algorithms for regularized NMF extensions, the proposed algorithm Alo converges much faster than algorithms AcS and Nev.

3.6 Conclusion and Discussion

In this chapter, we propose a general flexible algorithm in a unified framework for NMF and its L_1 L_2 regularized variants based on full decomposition, and employ a fast accelerated anti-lopsided algorithm for NMF. The proposed algorithm has over-bounded linear convergence rate of $\mathcal{O}[(1-\frac{\mu}{L})(1-\frac{\mu}{rL})^{2r}]^k$ in optimizing each matrix factor in the sub-space of passive variables when fixing the other matrix, where r is the number of latent components. The proposed algorithm is an advanced version of fast block coordinate descent methods and accelerated methods. In theory and practice, the proposed algorithm resolve some current major issues of NMF: fast learning algorithm, data sparseness exploit-ability, and parallel distributed feasibility using limited internal memory. Furthermore, the proposed algorithm flexibly adapts with all the variants of L_1 L_2 NMF regularizations.

In experimental comparative evaluation, our algorithm overcomes 7 of the most art-the-state algorithms in large datasets about three significant aspects of convergence, average of the iteration number and optimality. In addition, it can fully be parallelized and distributed because the computation using limited internal memory is decomposed into basic computation units as NQP problems. Concerning the feasibility in real applications, the proposed algorithm exploits the data sparseness to learn the huge sparse dataset Nytimes in an acceptable time by a single machine. Finally, the convergence of the proposed algorithm for L_1 L_2 regularized NMF variants is much faster than that of the existing algorithms.

Concerning the optimization techniques for alternating least squares methods, we propose a fast algorithm, Algorithm 3 for NQP problems, which not only has a linear convergence in theory but also is verified in practice about the three significant aspects by a large number of NQP problems conducted inside the NMF framework. Hence, we strongly believe that the algorithm can be effectively employed for alternating least square methods as the key problem in factorization methods. Hence, in further researches, we will generalize the proposed algorithm for nonnegative matrix factorization problems.

Publications

1. Nguyen, Duy Khuong, and Tu Bao Ho. Anti-lopsided Algorithm for Large-scale Nonnegative Least Square Problems. International Journal of Data Science and Analytics, Springer, (accepted to publish).

2. Nguyen, Duy-Khuong, and Tu-Bao Ho. Accelerated Parallel and Distributed Algorithm using Limited Internal Memory for Nonnegative Matrix Factorization. submitted to Journal of Global Optimization, (under revision).

Chapter 4

Fast Parallel Randomized Algorithm for NMF with L_1 L_2 Regularizations and KL Divergence for Large Sparse Datasets

We chose it because we deal with huge amounts of data.
Besides, it sounds really cool.
— Larry Page, founder of Google

Nonnegative Matrix Factorization (NMF) with Kullback-Leibler Divergence (NMF-KL) is one of the most significant NMF problems and equivalent to Probabilistic Latent Semantic Indexing (PLSI), which has been successfully applied in many applications. For sparse count data, a Poisson distribution and KL divergence provide sparse models and sparse representation, which describe the random variation better than a normal distribution and Frobenius norm. Specially, sparse models provide more concise understanding of the appearance of attributes over latent components, while sparse representation provides concise interpretability of the contribution of latent components over instances. However, minimizing NMF with KL divergence is much more difficult than minimizing NMF with Frobenius norm; and sparse models, sparse representation and fast algorithms for large sparse datasets are still challenges for NMF with KL divergence. In this chapter, we propose a fast parallel randomized coordinate descent algorithm having fast convergence for large sparse datasets to archive sparse models and sparse representation. The proposed algorithm's experimental results overperform the current studies' ones in this problem.

4.1 Introduction

The development of technology has been generating big datasets of count sparse data such as documents and social network data, which requires fast effective algorithms to manage this huge amount of information. One of these tools is nonnegative matrix factorization (NMF) with KL divergence, which is proved to be equivalent with Latent Semantic Indexing (PLSI) [26].

NMF is a powerful linear technique to reduce dimension and to extract latent topics, which can be readily interpreted to explain phenomenon in science [34, 58, 86]. NMF makes post-processing algorithms such as classification and information retrieval faster and more effective. In addition, latent factors extracted by NMF can be more concisely interpreted than other linear methods such as PCA and ICA [86]. In addition, NMF is flexible with numerous divergences to adapt a large number of real applications [92, 98].

For sparse count data, NMF with KL divergence and a Poisson distribution may provide sparse models and sparse representation describing better the random variation rather than NMF with Frobenius norm and a normal distribution [87]. For example, the appearance of words over latent topics and of topics over documents should be sparse. However, achieving sparse models and sparse representation is still a major challenge because minimizing NMF with KL divergence is much more difficult than NMF with Frobenius norm [53].

In the NMF-KL problem, a given nonnegative data matrix $V \in \mathcal{R}_+^{n\times m}$ must be factorized into a product of two nonnegative matrices, namely a latent component matrix $W \in \mathcal{R}_+^{r\times n}$ and a representation matrix $F \in \mathcal{R}_+^{r\times m}$, where n is the dimension of a data instance, m is the number of data instances, and r is the number of latent components or latent factors. The quality of this factorization is controlled by the objective function with KL divergence as follows:

$$D(V\|W^TF) = \sum_{i=1}^{n}\sum_{j=1}^{m}(V_{ij}\log\frac{V_{ij}}{(W^TF)_{ij}} - V_{ij} + (W^TF)_{ij}) \tag{4.1}$$

In the general form of L_1 L_2 regularization variants, the objective function is written as follows:

$$D(V\|W^TF) + \frac{\alpha_1}{2}\|W\|_2^2 + \frac{\alpha_2}{2}\|F\|_2^2 + \beta_1\|W\|_1 + \beta_2\|F\|_1 \tag{4.2}$$

NMF with KL divergence has been widely applied in many applications for dense datasets. For example, spatially localized, parts-based subspace representation of visual patterns is learned by local non-negative matrix factorization with a localization constraint (LNMF) [60]. In another study, multiple hidden sound objects from a single channel auditory scene in the magnitude spectrum domain can be extracted by NMF with KL divergence [85]. In addition, two speakers in a single channel recording can be separated by NMF with KL divergence and L_1 regularization on F [81].

However, the existing algorithms for NMF with KL divergence (NMF-KL) are extremely time-consuming for large count sparse datasets. Originally, Lee and Seung, 2001 [57] proposed the first multiple update iterative algorithm based on gradient methods for NMF-KL. Nevertheless, this technique is simple and ineffective because it requires a large number of iterations, and it ignores negative effects of nonnegative constraints. In addition, gradient methods have slow convergence for complicated logarithmic functions like KL divergence. Subsequently, Cho-Jui & Inderjit, 2011 [44] proposed a cycle coordinate descent algorithm having low complexity of one variable update. However, this method contains several limitations: first, it computes and stores the dense product matrix $W^T F$ although V is sparse; second, the update of $W^T F$ for the change of each cell in W and F is considerably complicated, which leads to practical difficulties in parallel and distributed computation; finally, the sparsity of data is not considered, while large datasets are often highly sparse.

In comparison with NMF with Frobenius norm, NMF with KL divergence is much more complicated because updating variables will influence derivatives of other variables; this computation is extremely expensive. Hence, it is difficult to employ fast algorithms having multiple variable updates, which limits the number of effective methods.

In this paper, we propose a new advanced version of coordinate descent methods with significant modifications for large sparse datasets. Regarding the contributions of this paper, we:

- Propose a fast sparse randomized coordinate descent algorithm using limited internal memory for nonnegative matrix factorization for huge sparse datasets, the full matrix of which can not stored in the internal memory. In this optimization algorithm, variables are randomly selected with uniform sampling to balance the order priority of variables. Moreover, the proposed algorithm effectively utilizes the sparsity of data, models and representation matrices to improve its performance. Hence, the proposed algorithm can be considered an advanced version of cycle coordinate descent for large sparse datasets proposed in [44].

- Design parallel algorithms for combinational variants of L_1 L_2 regularizations.
- Indicate that the proposed algorithm using limited memory can fast attain sparse models, sparse representation, and fast convergence by evaluational experiments, which is a significant milestone in this research problem for large sparse datasets.

The rest of the chapter is organized as follows. Section 4.2 presents the proposed algorithms. The theoretical analysis of convergence and complexity is discussed in Section 4.3. Section 4.4 shows the experimental results, and Section 4.5 summarizes the main contributions of this pape and discussion.

4.2 Proposed Algorithm

In this section, we propose a fast sparse randomized coordinate descent parallel algorithm for nonnegative sparse matrix factorization on Kullback-Leibler divergence. We employ a multiple iterative update algorithm like EM algorithm, see Algorithm 4, because $D(V\|W^TF)$ is a non-convex function although it is a convex function when fixing one of two matrices W and F. This algorithm contain a *while* loop containing two main steps: the first one is to optimize the objective function by F when fixing W; and the another one is to optimize the objective function by W when fixing F. Furthermore, in this algorithm, we need to minimize Function 4.3, the decomposed elements of which can be independently optimized in Algorithm 4:

$$D(V\|W^TF) = \sum_{i=1}^{m} D(V_i\|W^TF_i) = \sum_{j=1}^{n} D(V_j^T\|F^TW_j) \tag{4.3}$$

Specially, a number of optimization problems $D(V_i\|W^TF_i)$ or $D(V_j^T\|F^TW_j)$ in Algorithm 4 with the form $D(v\|Ax)$ can be independently and simultaneously solved by Algorithm 5. In this paper, we concern combinational variants of NMF KL divergence with L_1 L_2 regularizations in the general formula, Function 4.4:

$$f(x) = D(v\|Ax) = \sum_{i=1}^{n}(v_i \log\frac{v_i}{[Ax]_i} - v_i + [Ax]_i) + \frac{\alpha}{2}\|x\|_2^2 + \beta|x|_1 \tag{4.4}$$

where $v \in \mathcal{R}_+^n, A \in \mathcal{R}_+^{n\times r}, x \in \mathcal{R}_+^r$

Because the vector v is given, minimizing Function 4.4 is equivalent to minimizing

Algorithm 4: Iterative multiplicative update

```
   Input: $V \in \mathcal{R}_+^{n\times m}, r$, and $\alpha_1, \alpha_2, \beta_1, \beta_2 \geq 0$
   Output: $W \in \mathcal{R}_+^{n\times r}$, $F \in \mathcal{R}_+^{r\times m}$.
 1 begin
 2     Randomize $W \in \mathcal{R}_+^{r\times n}$;
 3     Randomize $F \in \mathcal{R}_+^{r\times m}$;
 4     while convergence condition is not satisfied do
 5         $ids$ = a randomized ordered set of values $\{1, 2, ..., r\}$;
 6         $sumW = W\mathbf{1}^n$;
 7         /*Optimizing the objective function by $F$ when fixing $W$*/;
 8         for $j$ = 1 to $m$ do
 9             /*Call Algorithm 5 in parallel*/;
10             $F_j^{k+1}$ = Algorithm 5 ($V_j$, $W^T$, $sumW$, $F_j^k$, $\alpha_2$, $\beta_2$, $ids$)
11         $sumF = F\mathbf{1}^m$;
12         /*Optimizing the objective function by $W$ when fixing $F$*/;
13         for $i$ = 1 to $n$ do
14             /*Call Algorithm 5 in parallel*/;
15             $W_i^{k+1}$ = Algorithm 5 ($V_i^T$ $F^T$, $sumW$, $W_i^k$, $\alpha_2$, $\beta_2$, $ids$);
16     return $(W^{k+1})^T$, $F^{k+1}$;
```

Function 4.5:

$$f(x) = D(v\|Ax) = \sum_{i=1}^{n}(-v_i \log{[Ax]_i} + [Ax]_i) + \frac{\alpha}{2}\|x\|_2^2 + \beta|x|_1 \tag{4.5}$$

From Equation 4.5, the first and second derivative of the variable x_k are computed by Formula 4.6:

$$\Rightarrow \begin{cases} \nabla f_k & = -\sum\limits_{i=1}^{n} v_i \frac{A_{ik}}{[Ax]_i} + \sum\limits_{i=1}^{n} A_{ik} + \alpha x_k + \beta \\ \nabla^2 f_{kk} & = \sum\limits_{i=1}^{n} v_i (\frac{A_{ik}}{[Ax]_i})^2 + \alpha \end{cases} \tag{4.6}$$

Based on Formula 4.6, we have several significant remarks:

- One update of x_k changes all elements of Ax, which are under the denominators of fractions. Hence, it is difficult to employ fast algorithms having simultaneous updates of multiple variables because it will require heavy computation. Hence, we employ coordinate descent methods to reduce the complexity of each update, and to avoid negative effects of nonnegative constraints.

- One update of x_k has complexity of maintaining ∇f_k and $\nabla^2 f_{kk}$ as $\mathcal{O}(k+\text{nnz}(v)+\text{nnz}(v)) = k + \text{nnz}(v)$ if $\sum_{i=1}^{n} A_{ik}$ is computed in advance. Specially, it employs $\mathcal{O}(k+\text{nnz}(v))$ of multiple and addition operators, and exactly $\mathcal{O}(\text{nnz}(v))$ of divide operators; where $\text{nnz}(v)$ is the number of non-zero elements in the vector v. Hence, for sparse datasets, the number of operators can be negligible.

- The used internal memory of Algorithm 4 and Algorithm 5 is $\mathcal{O}(\text{nnz}(V)+\text{size}(W)+\text{size}(F)) = \text{nnz}(V)+(n+m)r$, where $\text{nnz}(V)$ is the number of non-zero elements in the given matrix V, which is much smaller than $\mathcal{O}(mn+(n+m)r)$ for the existing algorithms [44, 57].

Hence, Algorithm 5 employs a coordinate descent algorithm based on projected Newton methods with quadratic approximation in Algorithm 5 is to optimize Function 4.5. Specially, because Function 4.5 is convex, a coordinate descent algorithm based on projected Newton method [66, 66] with quadratic approximation is employed to iteratively update with the nonnegative lower bound as follows:

$$x_k = \max(0, x_k - \frac{\nabla f_k}{\nabla^2 f_{kk}})$$

Considering the limited internal memory and the sparsity of x, we maintain $W^T F_j$ via computing Ax instead of storing the dense matrix $W^T F$ for the following reasons:

- The internal memory requirement will significantly decrease, so the proposed algorithm can stably run on limited internal memory machines.

- The complexity of computing Ax is always smaller than the complexity of computing and maintaining ∇f_k and $\nabla^2 f_{kk}$, so it does not cause the computation more complicated.

- The updating $Ax = Ax + \triangle x A_k$ as the adding with a scale of two vectors utilizes the speed of CPU cache because of accessing consecutive memory cells.

- The recomputing helps remove the complexity of maintaining the dense product matrix $W^T F$ as in [44,57], which is certainly considerable because this maintenance accesses memory cells far together and does not utilize CPU cache.

Algorithm 5: Randomized coordinate descent algorithm for sparse datasets

```
Input: $v \in \mathcal{R}^n$, $A \in \mathcal{R}^{n\times k}$, $sumA$, $x \in \mathcal{R}^k$, $\alpha \geq 0$, $\beta \geq 0$, and variable order $ids$
Output: $x$ is updated by
    $x \approx \underset{x\succeq 0}{\text{argmin}} \sum_{i=1}^{n} -v_i \log([Ax]_i + \epsilon) + [Ax]_i + \frac{\alpha}{2}\|x\|_2^2 + \beta\|x\|_1$.
1  begin
2      Compute $Ax \in \mathcal{R}^n$;
3      for $k$ in order $ids$ do
4          Compute $\nabla f_k$ and $\nabla^2 f_{kk}$ based on $\alpha, \beta, v, x, Ax, sumA$ and sparsity of $v$
           based on Formula 4.6;
5          while $(\nabla f_k < -\epsilon)$ or $(|\nabla f_k| > \epsilon$ and $x_k > \epsilon)$ do
6              $\triangle x = max(0, x_k - \frac{\nabla f_k}{\nabla^2 f_{kk}}) - x_k$;
7              Update $Ax$ via $Ax = Ax + \triangle x A_k$;
8              $xs = x_k$;
9              $x_k = x_k + \triangle x$;
10             if $(\triangle x < \epsilon_x xs)$ then
11                 break;
12             Update $\nabla f_k$ and $\nabla^2 f_{kk}$ based on $\alpha, \beta, v, x, sumA$ and sparsity of $v$
               based on Formula 4.6;
13     return $x$;
```

In summary, in comparison with the original coordinate algorithm [44] for NMK-KL, the proposed algorithm involve significant improvements as follows:

- Randomize the order of variables to optimize the objective function in Algorithm 5. Hence, the proposed algorithm can balance the order priority of variables,
- Remove duplicated computation of maintaining derivatives ∇f_k and $\nabla^2 f_{kk}$ by computing common elements $sumW = W\mathbf{1}^n$ and $sumF = F\mathbf{1}^m$ in advance, which led to that the complexity of computing ∇f_k and $\nabla^2 f_{kk}$ only depends on the sparsity of data,
- Effectively utilize the sparsity of W and F to reduce the running time of computing Ax,
- Effectively utilize CPU cache to improve the performance of maintaining $Ax = Ax + \triangle x A_k$,

- Recompute Ax but remove the maintenance of the dense matrix product $W^T F$. Hence, the proposed algorithm stably run on the limited internal memory systems with the required memory size $\mathcal{O}(\text{nnz}(V)+\text{size}(W)+\text{size}(F)) = \text{nnz}(V)+(m+n)r$, which is much smaller than $\mathcal{O}(mn+(n+m)r)$ for the existing algorithms [44, 57].

4.3 Theoretical Analysis

In this section, we analyze the convergence and complexity of Algorithm 4 and Algorithm 5.

In comparison with the previous algorithm of Hsieh & Dhillon, 2011 [44], the proposed algorithm has significant modifications for large sparse datasets by means of adding the order randomization of indexes and utilizing the sparsity of data V, model W, and representation F. These modifications does not affect on the convergence guarantee of algorithm. Hence, based on Theorem 1 in Hsieh & Dhillon, 2011 [44], Algorithm 5 converges to the global minimum of $f(x)$. Furthermore, based on Theorem 3 in Hsieh & Dhillon, 2011 [44], Algorithm 4 using Algorithm 5 will converge to a stationary point. In practice, we set $\epsilon_x = 0.1$ in Algorithm 5, which is more precise than $\epsilon_x = 0.5$ in Hsieh & Dhillon, 2011 [44].

Concerning the complexity of Algorithm 5, based on the remarks in Section 4.2, we have Theorem 4. Furthermore, because KL divergence is a convex function over one variable and the nonnegative domain, and project Newton methods with quadratic approximation for convex functions have superlinear rate of convergence [9, 66], the average number of iterations $\bar{t}$ is small.

Theorem 4 *The complexity of Algorithm 5 is $\mathcal{O}(n nnz(r) + \bar{t}r(r + n + nnz(n)))$, where $nnz(r)$ is the number of non-zero elements in x, $nnz(n)$ is the number of non-zero elements in v, and $\bar{t}$ is the average number of iterations. Then, the complexity of a while iteration in Algorithm 4 is $\mathcal{O}(\bar{t}(mnr + (m+n)r^2))$*

Proof: Consider the major computation in Algorithm 5, based on Formula 4.6, we have:

- The complexity of computing Ax in Line 2 is $\mathcal{O}(n\text{nnz}(r))$,
- The complexity of computing ∇f_k and $\nabla^2 f_{kk}$ in Line 4 is $\mathcal{O}(r + \text{nnz}(n))$ because Ax and $sumA$ are computed in advance,
- The complexity of updating ∇f_k and $\nabla^2 f_{kk}$ in Line 12 is $\mathcal{O}(r+n+\text{nnz}(n))$ because only one dimension of vector x is changed.

Table 4.1 Summary of datasets

Dataset (V)	n	m	nnz(V)	Sparsity (%)
Digits	784	60,000	8,994,156	80.8798
Reuters21578	8,293	18,933	389,455	99.7520
TDT2	9,394	36,771	1,224,135	99.6456
RCV1_4Class	9,625	29,992	730,879	99.7468

Hence, the complexity of Algorithm 5 is $\mathcal{O}(n\text{nnz}(r) + \bar{t}r(r + n + \text{nnz}(n)))$.

In addition, the complexity of computing $sumW$ and $sumF$ is $\mathcal{O}((m+n)r)$. Hence, the complexity of a *while* iteration in Algorithm 4 is $\mathcal{O}((m+n)r + mn\text{nnz}(r) + \bar{t}mr(r + n + \text{nnz}(n))) \approx (m+n)r + \bar{t}(mnr + (m+n)r^2) \approx \bar{t}(mnr + (m+n)r^2)$ Therefore, we have Theorem 4 □

For large sparse datasets, $m, n \gg r \Rightarrow \mathcal{O}(\bar{t}(mnr + (m+n)r^2)) \approx \bar{t}(mnr)$. This complexity is raised by the operators $Ax = Ax + \triangle x A_k$ in Algorithm 5. To reduce the running time of these operators, A_k must be stored in an array to utilize CPU cache memory by accessing continuous memory cells of Ax and A_k.

4.4 Experimental Evaluation

In this section, we investigate the effectiveness of the proposed algorithm via convergence and sparsity. Specially, we compare the proposed algorithm Sparse Randomized Coordinate Descent (**SRCD**) with state-of-the-art algorithms as follows:

- Multiplicative Update (**MU**) [57]: This algorithm is the original method for NMF with KL divergence.

- Cycle Coordinate Descent (**CCD**) [44]: This algorithm has the current fastest convergence because it has very low complexity of each update for one variable.

Datasets: To investigate the effectiveness of the algorithms compared, the 4 sparse datasets used are shown in Table 4.1. The dataset Digit is downloaded from [1], and the other tf-idf datasets Reuters21578, TDT2, and RCV1_4Class are downloaded from [2].

Environment settings: We develop the proposed algorithm SRCD in Matlab with embedded code C++ to compare them with other algorithms. We set system parameters

[1]http://yann.lecun.com/exdb/mnist/
[2]http://www.cad.zju.edu.cn/home/dengcai/Data/TextData.html

to use only 1 CPU for Matlab and the IO time is excluded in the machine Mac Pro 8-Core Intel Xeon E5 3 GHz 32GB. In addition, the initial matrices W^0 and F^0 are set to the same values. The source code will be published on our homepage [3].

4.4.1 Convergence

In this section, we investigate the convergence of the objective value $D(V||W^T F)$ versus running time by running all the compared algorithms on the four datasets with two numbers of latent factors $r = 10$ and $r = 20$. The experimental results are depicted in Figure 4.1 and Figure 4.2. From these figures, we realize two significant observations as follows:

- The proposed algorithm (SRCD) has much faster convergence than the algorithms CCD and MU,
- The sparser the datasets are, the greater the distinction between the convergence of the algorithm SRCD and the other algorithms CCD and MU is. Specially, for Digits with 81% sparsity, the algorithm SRCD's convergence is lightly faster than the convergence of the algorithms CCD and MU. However, for three more sparse datasets Reuters21578, TDT2, and RCV1_4Class with above 99% sparsity, the distance between these convergence speeds is readily apparent.

4.4.2 Sparsity of factor matrices

Concerning the sparsity of factor matrices W and F, the algorithms CCD and MU does not utilize the sparsity of factor matrices. Hence, these algorithms add a small number into these factor matrices to obtain convenience in processing special numerical cases. Hence, the sparsity of factor matrices W and F for the algorithms CCD and MU both are 0%. Although this processing may not affect other post-processing tasks such as classification and information retrieval, it will reduce the performance of these algorithms. The sparsity of (W, F) of the proposed algorithm's results is showed in Table 4.2. These results clearly indicate that the sparse model W and the sparse representation F are attained. The results also explain why the proposed algorithm runs very fast on the sparse datasets Reuters21578, TDT2 and RCV1_4Class, when it can obtain highly sparse models and sparse representation in these highly sparse datasets.

[3] http://khuongnd.appspot.com/

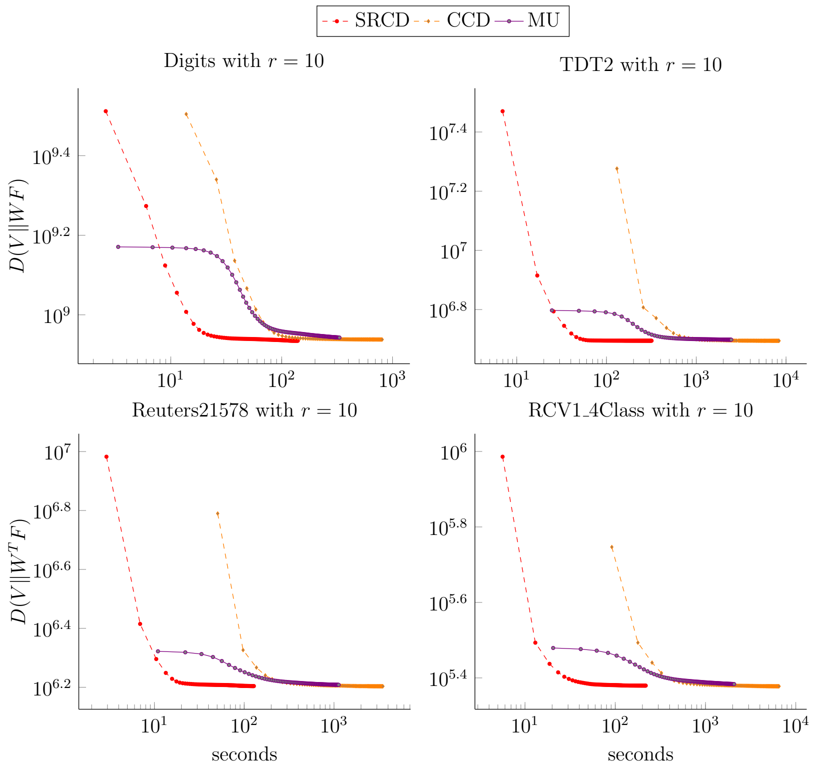

Fig. 4.1 Objective value $D(V\|W^TF)$ versus running time with $r = 10$

Table 4.2 Sparsity (%) of (W, F) for the algorithm SRCD's results

	Digits	Reuters21578	TDT2	RCV1_4Class
$r = 10$	(74.3, 49.2)	(75.6, 71.6)	(68.5, 71.3)	(81.2, 74.0)
$r = 20$	(87.8, 49.7)	(84.2, 80.4)	(78.6, 81.1)	(88.4, 83.0)

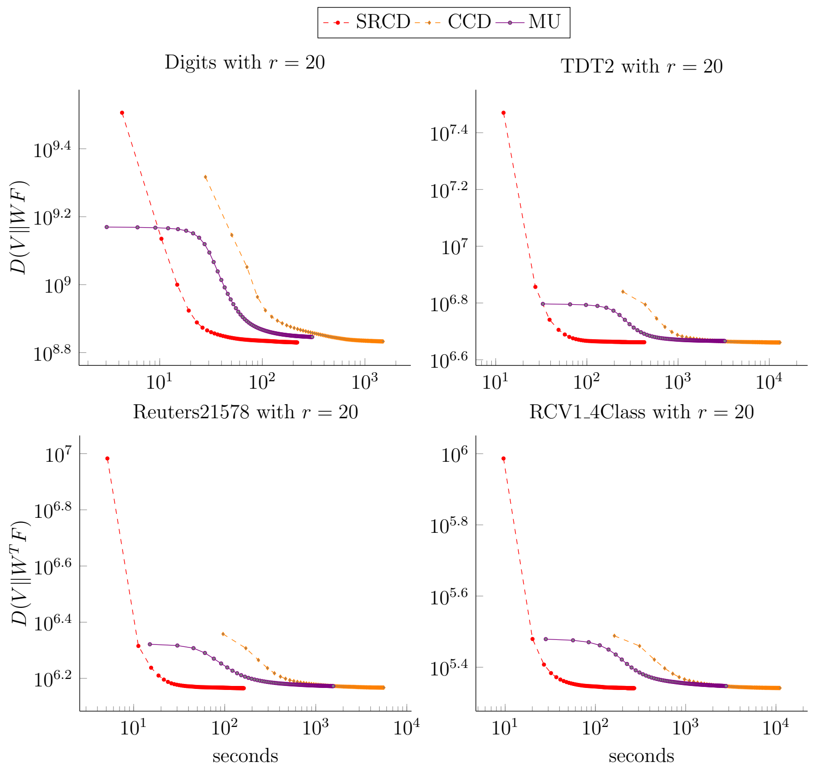

Fig. 4.2 Objective value $D(V\|W^TF)$ versus running time with $r = 20$

4.4.3 Used internal memory

Table 4.3 shows the internal memory used by algorithms. From the table, we have two significant observations:

- For the dense dataset Digits, the proposed algorithm SRCD uses more internal memory than the the algorithm CCD because a considerable amount of memory is used for the indexing of matrices.
- For the sparse datasets Reuters21578, TDT2, and RCV1_4Class, the internal

Table 4.3 Used internal memory (GB) for $r = 10$

Datasets	MU	CCD	SRCD
Digits	1.89	**0.85**	1.76
Reuters21578	5.88	2.46	**0.17**
TDT2	11.51	5.29	**0.30**
RCV1_4Class	9.73	4.43	**0.23**

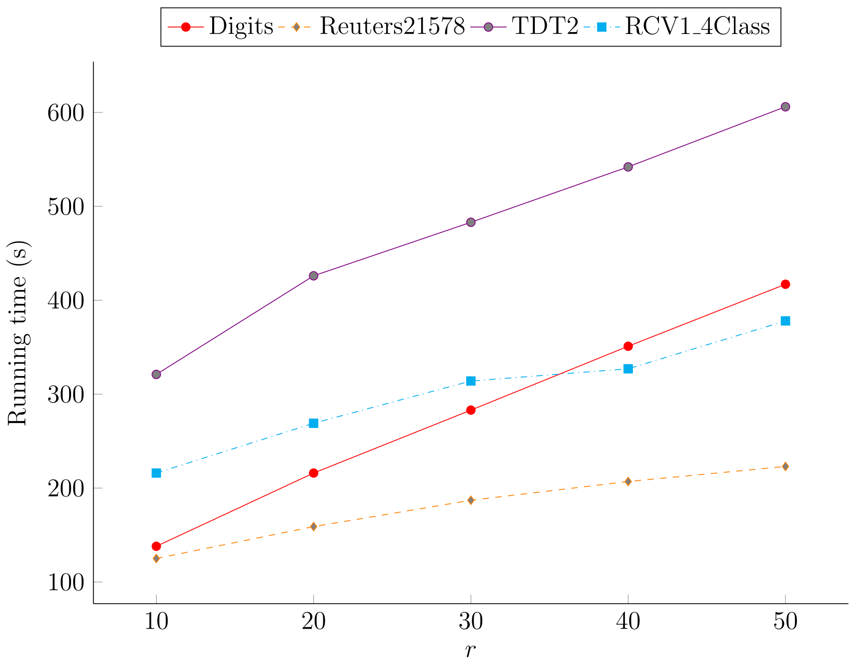

Fig. 4.3 Running time of 100 iterations with different number of latent component using 1 thread

memory for SRCD is remarkably smaller than the internal one for MU and CCD. These results indicate that we can conduct the proposed algorithm for huge sparse datasets with a limited internal memory machine is stable.

4.4.4 Running on large datasets

This section investigates running the proposed algorithm on large datasets with different settings. Figure 4.3 shows the running time of Algorithm SRCD for 100 iterations with different number of latent component using 1 thread. Clearly, the running time linearly increases, which fits the theoretical analyses about fast convergence and linear complexity for large sparse datasets in Section 4.3. Furthermore, concerning the parallel algorithm, the running time of Algorithm SRCD for 100 iterations significantly decreases when the number of used threads increases in Figure 4.4. In addition, the running time is acceptable for large applications. Hence, these results indicate that the proposed algorithm SRCD is feasible for large scale applications.

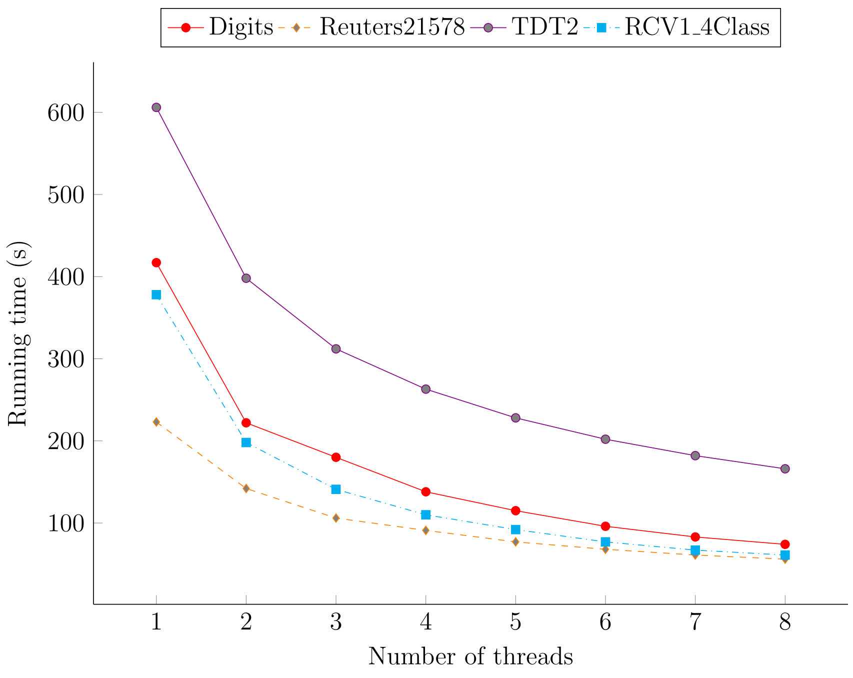

Fig. 4.4 Running time of 100 iterations with $r = 50$ and using different number of threads

4.5 Conclusion and Discussion

In this chapter, we propose a fast parallel randomized coordinate descent algorithm for NMF with KL divergence for large sparse datasets. The proposed algorithm attains fast convergence by means of removing duplicated computation, exploiting sparse properties of data, model and representation matrices, and utilizing the fast accessing speed of CPU cache. In addition, our method can stably run systems within limited internal memory by reducing internal memory requirements. Finally, the experimental results indicate that highly sparse models and sparse representation can be attained for large sparse datasets, a significant milestone in researching this problem. In future research, we will generalize this algorithm for nonnegative tensor factorization.

Publications

1. Nguyen, Duy-Khuong, 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 (Accepted to publish).

Chapter 5

Fast Parallel Algorithm for Simplicial NMF with Frobenius Norm

If your experiment needs statistics, you ought to have done a better experiment.
— Ernest Rutherford

Nonnegative matrix factorization (NMF) is a linear powerful method for dimension reduction and component analysis, which has been widely in many applications such as information retrieval, image processing, etc. It has more concise interpretability and more sparsity than other linear methods such as PCA and ICA because of its non-negativity constraints. However, these non-negativity constraints still have discursive interpretability of the role of latent components over data instances. In the chapter, we proposed simplicial NMF by adding simplicial constraints over coefficients that express the probabilistic combination role of latent components over data instances. In addition, we propose a fast parallel algorithms with instance inference guaranteed about sub-linear convergence $\mathcal{O}(1/k)$, low iteration complexity and sparsity control for simplicial NMF with Frobenius norm. The experimental results are highly competitive with the current results of nonnegative matrix factorization.

5.1 Introduction

Many algorithms in data mining cannot deal with large datasets because of their rawness, high dimension and complex distribution. To deal with this situation, two fundamental purposes have been raised in data processing: Transforming the data into a lower dimension space and extracting latent components and variables inside the datasets to represent data [92]. Nonnegative matrix factorization (NMF) is one of the most popular effective methods to pursue these two purposes. Many datasets in various fields have been formed as matrices with nonnegative values. NMF aims to factorize such a matrix V into a product of two matrices, $V \approx W^T F$, where W contains basis vectors in the new space and F contains new corresponding coefficients of data instances in V. In other words, this factorization transforms data instances into new space of basis vectors.

Many NMF methods have been developed in the last decade by using divergence functions, constraints and regularizations. Initially, basic NMF [58] only allows approximating the original nonnegative data by a product of two matrices. By this approximation, each object is represented as an additive combination of nonnegative parts or basis vectors. Following [58], many algorithms were proposed for different divergence functions [83]. Currently, by adapting requirements for data analysis problems and data types, many variants of NMF are being developed. Also, various new divergence functions are employed [97], and local constraints are added to improve the quality of matrix decompositions, which preserve the local features [60]. Sparse representation can be achieved by adding sparseness constraints [43, 76]. In addition, some work has implicitly or explicitly added orthogonality constraints [19, 59].

Usually, NMF is solved by iterative multiplicative updating algorithms because the objective function is non-convex, although each sub-problem when fixing one of latent matrices is convex. Minimizing object functions does not guarantee a unique solution, which often converges to stationary points. The existence of many stationary points makes the algorithms suffering from rotational ambiguities. If no prior information is known, the normalization of rows in W or latent components will help to reduce the effects of these ambiguities [22]. Specially, if $V \approx W^T F$ is a solution, $V \approx (D_W W')^T G = W^{T'}[D_W^T F]$ is also a solution; where $D_W = diag(\|W_1^T\|_p^{-1}, ..., \|W_r^T\|_p^{-1}),\quad p \in [1, \infty)$ and $W = D_W W'$. As a result, the latent components are normalized as basis vectors. Although this technique is advantageous to optimize the objective functions, the role of basis vectors is not easy to interpret directly via their coefficients values.

In this chapter, we propose a new NMF formulation by adding new prior information into NMF, in which each data instance is a convex combination of the latent components. In other words, each instance is a probabilistic distribution over the latent components. By this way, we associate a probabilistic model with the NMF problem. As a result, we obtain more advantages than the previous formulations such as easy interpretability, low complexity, convexity, sparsity, and distributability and parallelability. We also develop an effective algorithm for one of the most popular divergence functions as Frobenius norm [97].

5.2 Simplical Matrix Factorization with Frobenius norm

Mathematically, we can define NMF problem with Frobenius norm as follows:

Definition 2 ***[NMF]***: *Given a dataset consisting of m vectors in n dimensions $V = [V_1, V_2, ..., V_m] \in R_+^{n \times m}$, where each vector V_j presents a data instance. NMF seeks to factorize V into a product of two nonnegative factorizing matrices W^T and F, where $W \in \mathcal{R}_+^{r \times n}$ and $F \in \mathcal{R}_+^{r \times m}$ are coefficient matrix and latent component matrix, respectively, $V \approx W^T F$. In other words, each instance is approximately represented by a linear combination of these latent components $V_j \approx W^T F_i$. For Frobenius norm, NMF minimize the objective function: $\|V - W^T F\|_2^2$*

We assume that each instance is a probabilistic convex combination of the latent components obtained by adding simplicial constraints into NMF. Hence, we have:

Definition 3 ***[Simplicial NMF]***: *Simplicial NMF is NMF where each instance V_j is a convex combination of the latent components $V_j \approx W^T F_j$ and $\sum\limits_{k=1}^{r} F_{kj} = 1, \forall j$.*

By adding this new constraint, we have associated a probabilistic model with NMF problem, in which each instance is a probabilistic distribution over the latent components and represented as a convex combination of latent components. In other words, this convex combination provides explicitly the extent of contribution of each latent component, while other formulations of NMF do not have. Moreover, regarding to geometry meaning, each instance is projected as a point on the simplex of latent components. This projection is called instance inference. As a result, we obtained significant properties:

- **Sparsity**: Instance inference is a convex problem over the simplex of latent components. Furthermore, we can easily control the solution sparsity via greedy approximation algorithms such as Frank-Wolfe algorithm [23], which can guarantee theoretical convergence.

- **Convexity**: Obviously, inferring an instance is to find an approximation of the convex combination that is a convex optimization problem [13,14,23]. In addition, this simplicial convex problem can be solved by linearly guaranteed algorithms such as Frank-Wolfe algorithm [23, 55].

- **Computation**: The instance representation can be considered as a projection on the simplex of the latent components. Hence, the inference based on this projection can be guaranteed by sub-linear convergence rate $O(1/k)$ [23]. In comparison to other formulations, this one has significant computing advantages in the inference of instances, while the learning step is the same with the previous basic formulations because it solves the same optimization problem.

- **Interpretability**: The new formulation provides a more comprehensible interpretation of the contribution role of coefficients. Particularly, each data instance is a convex combination of the latent components, in which the sum of coefficients always equals to 1 through NMF. Hence, the contribution role of the latent components on instances can be concisely represented via values of coefficients. Otherwise, for other formulations, evaluating the contribution of components is forceful because of the lack of constraints between coefficients. Alternatively, a post-processing can be employed to find out the role of the latent components. However, it is independent and inconsistent with learning NMF model.

- **Distributability and parallelizability**: NMF problem contains two sub-problems: inference and learning, see Section 5.3. The learning problem is the same with other formulations and can be solved by distributed algorithms [20]. Meanwhile, the inference one of our formulation can be solved by a much faster algorithm comparing to the others', and it can be parallelized [13]. This favor is hard to be reached in other formulations.

To control the quality of NMF, various cost functions $f(V\|W^TF)$ are employed, which often contain two parts: The first part is a divergence function that measures the difference between original coordinates (V) inverted coordinates (W^TF); and the second one is possibly regularizations and constraints to control sparsity, smoothness, and orthogonality. In this formulation, we only consider the objective function containing a divergence function between original coordinates (V) and coordinates (W^TF) under simplicial contraints because these constraints can guarantee sparsity and smoothness to avoid over-fitting problems.

In this chapter, we focus on simplicial nonnegative matrix factorization with Frobenius norm:

- Frobenius norm:

$$D_{Fro}(x\|y) = \frac{1}{2}\|x-y\|_2^2 = \frac{1}{2}\sum_{i=1}^{n}(x_i - y_i)^2$$

- Simplicial NMF with Frobeius norm:

$$J(V\|W^TF) = \frac{1}{2}\sum_{j=1}^{m} D_{Fro}(V_j\|W^TF_j) = \frac{1}{2}\sum_{j=1}^{m}\|V_j - W^TF_j\|_2^2.$$

5.3 Proposed Algorithm

For solving simplicial NMF with Frobenius norm, we employ iterative multiplicative updates like *EM* algorithm, which is presented in Algorithm 6, because the objective function is non-convex, although sub-problems are strongly convex. This algorithm contains two main steps: one for finding F when fixing W, and the other for finding W when fixing F. In the first step, $F = \{F_j\}_{j=1}^m$ are updated to achieve a better objective value $W \approx \underset{F}{\operatorname{argmin}}\ D_{Fro}(V^T\|F^TW)$, where each of F_j is a probabilistic representation of data instances $\{V_j\}_{j=1}^m$. Hence, this step can be seen as *inference step* and the process is called as *inferring an data instance*. In the other step called *learning step*, the latent components are acquired by approximately minimizing $D_{Fro}(V\|W^TF)$ when fixing F: $W \approx \underset{W}{\operatorname{argmin}}\ D_{Fro}(V\|W^TF)$.

5.3.1 Inference Algorithm

Remark 1 *Inferring of data instances V in a new space of latent components by minimizing $J(V\|W^TF)$ can be conducted independently.*

In this step, we need to minimize

$$J(V\|W^TF) = \sum_{j=1}^{m} D_{Fro}(V_j\|W^TF_j) = \sum_{j=1}^{m}\|V_j - W^TF_j\|_2^2$$

Hence, since V and W are fixed in this step, minimizing $J(V\|W^TF)$ is equivalent to minimizing nonnegative least squares or its nonnegative quadratic programming problem $\frac{1}{2}x^TQx + q^Tx$, which is performed independently by Algorithm 7.

Specially, when setting $x = F_j$, the inference will become minimizing:

$$J(v\|W^Tx) = \frac{1}{2}\|V_j - W^Tx\|_2^2 = \frac{1}{2}x^TQx + q^Tx \tag{5.1}$$

Algorithm 6: Iterative multiplicative update for Frobenius norm

```
Input: Data matrix $V = \{V_j\}_{j=1}^{m} \in R_+^{n\times m}$ and $r$.
Output: Coefficients $F \in R_+^{r\times m}$ and latent components $W \in R_+^{r\times n}$
1 begin
2     Select randomly $r$ components from $m$ data instances;
3     repeat
4         $q = -WV$;
5         $Q = WW^T$;
6         Inference step: Fix $W$ to find $F \approx \underset{F\in\mathcal{R}^{r\times m}}{\operatorname{argmin}} J(V\|W^TF)$;
7         /*Call Algorithm 7 in parallel*/;
8         for $j$ = 1 to $m$ do
9             $F_j \approx \underset{x\ inR_+^r, x^T1^r=1}{\operatorname{argmin}} \|V_j - W^Tx\|_2^2 = \underset{x\ inR_+^r, x^T1^r=1}{\operatorname{argmin}} \frac{1}{2}x^TQx + q_j^Tx$
10        $q = -FV^T$;
11        $Q = FF^T$;
12        Learning step: Fix $F$ to find $W \approx \underset{W\in\mathcal{R}^{r\times n}}{\operatorname{argmin}} J(V^T\|F^TW)$;
13        /*Call solving NQP in parallel*/;
14        for $i$ = 1 to $n$ do
15            $W_i \approx \underset{x\ inR_+^r}{\operatorname{argmin}} \|V_i^T - F^Tx\|_2^2 = \underset{x\ inR_+^r}{\operatorname{argmin}} \frac{1}{2}x^TQx + q_i^Tx$
16        if convergence condition is satisfied then
17            break;
18    until False;
```

where: $\sum_{k=1}^{r} x_k = 1, Q = WW^T, q = -WV_j$.

This is a nonnegative quadratic programming problem adding a simplicial constraint.

Remark 2 *Inferring each data instance is equivalent to solving a nonnegative quadratic programming problem with a simplicial contraint.*

Moreover, adding the convex constraints leads to the existence of sparse solutions and it can avoid over-fitting problems. Specially, the convex constraints enable greedy algorithms, which is derived from Frank-Wolfe algorithm [23, 55], in order to control directly and effectively sparsity of solutions, see more details in Algorithm 7.

Algorithm 7: Inference for data instance x

```
    Input: $Q \in \mathcal{R}^{r \times r}, q \in \mathcal{R}^r$.
    Output: New coefficient $x \approx \underset{x \in \mathcal{R}^r}{\operatorname{argmin}} f(x) = \underset{x \in \mathcal{R}^r}{\operatorname{argmin}} \frac{1}{2} x^T Q x + q^T x$.
1   begin
2       Choose $k = \arg \min_k \frac{1}{2} e_k^T Q e_k + q^T e_k$, where $e_k$ is the $k^{th}$ basis vector;
3       Set $x = \mathbf{0}^k$; $x_k = 1$; $Qx = Qe_k$; $qx = q^T x$ and $\nabla f = Qx + q^T$;
4       repeat
5           Select $k = \underset{k \in \{1..r\}}{\operatorname{argmin}} \nabla f_k$;
6           Select $\alpha = \underset{\alpha}{\operatorname{argmin}} f(\alpha e_k + (1 - \alpha) x)$;
7           $\alpha = min(1, max(-1, max(\alpha, -\frac{f_k}{1 - f_k})))$;
8           if $\alpha == 0$ then
9               break;
10          $Qx = (1 - \alpha) Qx + \alpha Q e_k$;
11          $\nabla f = Qx + q$;
12          $qx = (1 - \alpha) qx + \alpha q e_k$;
13          $x = (1 - \alpha) x$;
14          $x_k = x_k + \alpha$;
15      until converged conditions are staisfied;
```

5.3.2 Learning Algorithm

Remark 3 *Learning components W by minimizing $J(V^T \| F^T W)$ can be independently conducted in each attribute.*

We also have $J(V^T \| F^T W) = \sum_{i=1}^{n} \frac{1}{2}\|V_i^T - F^T W_i\|_2^2$. Hence, minimizing $J(V^T \| F^T W)$ is equivalent to minimizing $\frac{1}{2}\|V_i - F^T W_i\|_2^2 = \frac{1}{2}x^T Q x + q^T x$ independently in each attribute since V and W are fixed, where $Q = FF^T, q = -FV_i^T$. Therefore, we can independently learn attributes of the latent components by solving nonnegative quadratic programming problem [56, 73].

5.4 Theoretical Analysis

5.4.1 Complexity

In this section, we discuss the complexity of instance inference because it effects on the performance of real applications. Consider the complexity of inference steps, we have:

Theorem 5 *The complexity of inferring a data instance is $\mathcal{O}(kr)$, and the complexity of inference step is $O(mnr + nr^2 + kmr)$, where k is the iteration number, n is the instance dimension, and m is the instance number.*

Proof: Excepted the statement in Line 6, all other statements in Algorithm 7 have complexity $O(1)$ or $O(r)$. Hence, the complexity of inferring a data instance is $\mathcal{O}(kr)$ if and only if $\alpha = \operatorname*{argmin}_{\alpha} f(\alpha e_k + (1-\alpha)x)$ has complexity of $O(r)$. Because $f(\alpha e_k + (1-\alpha)x)$ is a quadratic function of α, $\alpha = \operatorname*{argmin}_{\alpha} f(\alpha e_k + (1-\alpha)x) = -\frac{\nabla f_\alpha}{\nabla^2 f_\alpha}$.

In addition, we have:

$\nabla f_\alpha = (\mathbf{e}_k - x)^T Q((1-\alpha)x + \alpha\mathbf{e}_k) + (\mathbf{e}_k - x)^T q = (1-\alpha)(e_k - x)\nabla f + \alpha(e_k - x)(Qe_k + q)$

$\nabla^2 f_\alpha = (\mathbf{e}_k - x)^T Q(\mathbf{e}_k - x) = e_k^T Q e_k^T - 2e_k^T Q x + x^T Q x$

Hence, both of computing ∇f_α and $\nabla^2 f_\alpha$ can be computed in $O(r)$ because ∇f and Qx are computed in advance.

Therefore, the complexity of inferring a data instance is $\mathcal{O}(kr)$, and the complexity of inference step is $O(mnr + nr^2 + kmr)$ if we consider computing WV and WW^T is $O(mnr + nr^2)$. □

5.4.2 Convergence Guarantee of Inference

Concerning the convergence of inference, based on [23], we have

Theorem 6 *Let f be a twice differentiable convex function over simplex $\triangle$ and denote $C_f = sup_{y,z\in\triangle;\tilde{y}\in[y,z]}(y-z).\nabla^2 f(\tilde{y}).(y-z)^T$. After l iterations, the Frank-Wolfe algorithm will find an approximate solution x_l with at most $(l+1)$ non-zeros coefficients which satisfy*

$$max_{x\in\triangle} f(x_l) - f(x) \leq \frac{C_f}{l+1}$$

From this theorem, we have the following remarks:

- Convergence rate of inference is linear and the goodness of solutions is bounded, which are crucial in applications.
- Inference depends mostly on complexity of f and ∇f.
- We can easily tradeoff between sparsity and quality of solutions by stop finding new latent components to optimize the cost function. This property is valid for real applications, which the number of non-zero coefficients is limited.

5.4.3 Sparsity

Recently, sparse solutions receive much interests in machine learning by its abilities of improving accuracy and saving storage with low complexity. To obtain sparse solutions, most previous works employed different regularizations such as L_1 and L_2 ones. However, they are limited in controlling sparsity level of solutions. In other words, the number of non-zero coefficients in solutions is unpredictable.

Unlike previous approaches, we have imposed a greedy algorithm, e.g., Frank-Wolfe algorithm, which can control severely the solution's sparsity. From Algorithm 7, the number of non-zero coefficients can be restricted by do not employing new latent components to optimize the cost function, when the number of non-zeros ones reaches to the limitation. Moreover, the preference of selecting the best latent components to optimize allows our algorithm to achieve more sparse solutions than other algorithms while keeping the optimality of solution.

5.4.4 Distributability and Parallelizability

From the proposed algorithm, Algorithm 7, we have several remarks:

- Inference of data instances can be fully distributed over machines that is crucial for designing distributed algorithms.
- Running time of inference mostly depends on finding the best latent component. Furthermore, computing the partial derivative for each latent component is totally separated. As a result, this computation can be paralleled. Hence, the responding time in real applications can be effectively reduce.

5.5 Experimental Evaluation

This section investigates the effectiveness of our approach and the proposed algorithm for simplicial NMF (sNMF) with Frobenius norm by four criteria: interpretability, sparsity of solutions, performance in classification tasks and loss information measure. More particularly, our algorithm for Frobenius norm is compared to NMF [83], spNMF [82], oNMF [19], and cNMF [25]. The implemented codes are at [1].

Test-cases: In this investigation, we use one typical databases of images digit dataset, which has 4 000 random-selected samples from [2].

5.5.1 Interpretation

In conducting experiments for the digit dataset, we have run with different parameters $r \in \{10, 15, 20, 25, 30, 35, 40\}$. We realize that approaches begin finding out part-based representation of data instances from $r = 25$. Figure 5.3 shows latent components of NMF [58] and sNMF with $r = 25$ for the digit dataset. Obviously, latent components of sNMF are small part curves of digits, while ones of Lee 2001's NMF 5.3 also gives a part-based representation but they are bigger curves. The results indicates that the proposed approach obtains a better part-based representation for data instances because the latents components are more sparse and independent each other. Moreover, when factorizing matrices is comleted, sNMF's coefficients are in $[0, 1]$ with sum equal to 1, so they can represent the role of latent components in instances, while coefficients in

[1]http://www.ee.columbia.edu/~grindlay/code.html
[2]http://yann.lecun.com/exdb/mnist/

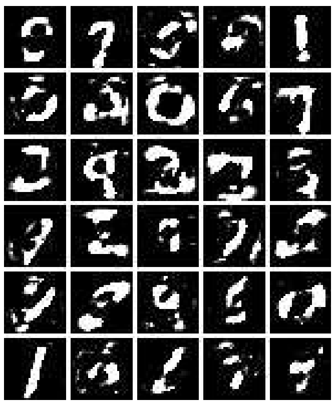
Fig. 5.1 Basic NMF

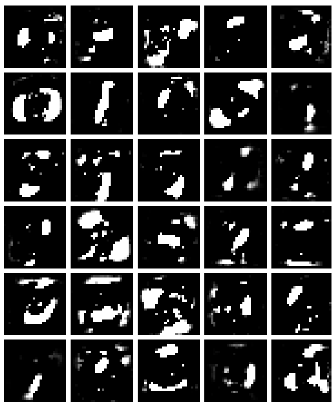
Fig. 5.2 Simplicial NMF

Fig. 5.3 Latent components with $K = 25$ for digit database

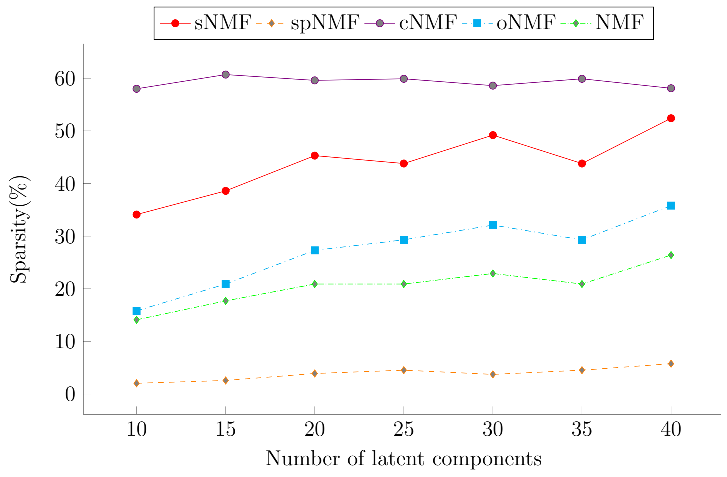

Fig. 5.4 Sparsity of new coefficients for Frobenius norm with $r = 30$

other formulations does not. They are nonnegative numbers, which only represent the measure of basis vectors.

5.5.2 Sparse Representation

In order to compare the sparsity of solutions, we compute the percentage of zero coefficients

$$\frac{\text{Number of zero coefficients}}{\text{Number of coefficients}} \times 100\%$$

The results are highly competitive with other methods. For Frobenius norm, although our algorithm's sparsity is only less than cNMF [25] (Figure 5.4), it has lower information loss and higher performance in classification.

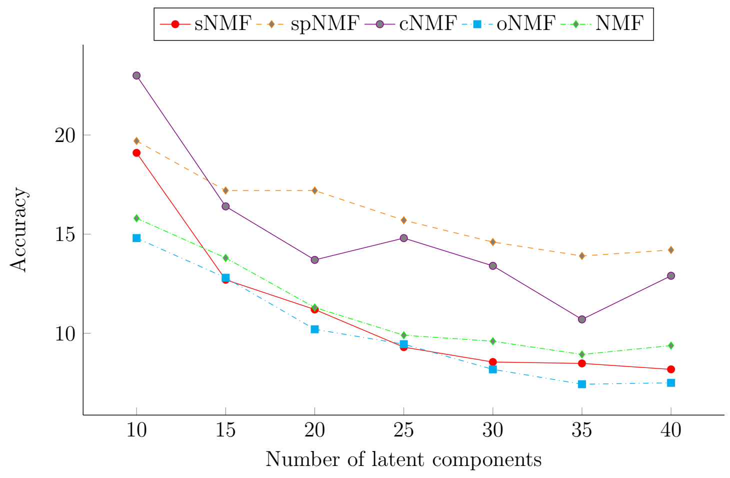

Fig. 5.5 Inaccuracy for Digit Classification

5.5.3 Performance for classification

Classification quality is one of measures that evaluates our method's effectiveness as NMF is often considered as a dimension redution technique used widely in classification. In this experiment, we use Random Forest[3], a robust algorithm for classification. Observing Figure 5.5, our method is one of methods with the lowest errors in testing. For Frobenius norm and the digit dataset, the result of our method is very close to the best method oNMF [19].

5.5.4 Convergence

For dimension reduction, information loss criterion is one of the most important measure. From observing Figure 5.6, our approach has the lowest information loss.

In addition, convergence speed is a significant measure to evaluate updating algorithms because algorithms having more iterations require more computation and also time for loading data. We can realize easily from Figure 5.6 that our algorithm has the fast convergence speed while it gains the best optimized solutions with at the least number of iterations.

[3]http://cran.r-project.org/web/packages/randomForest/

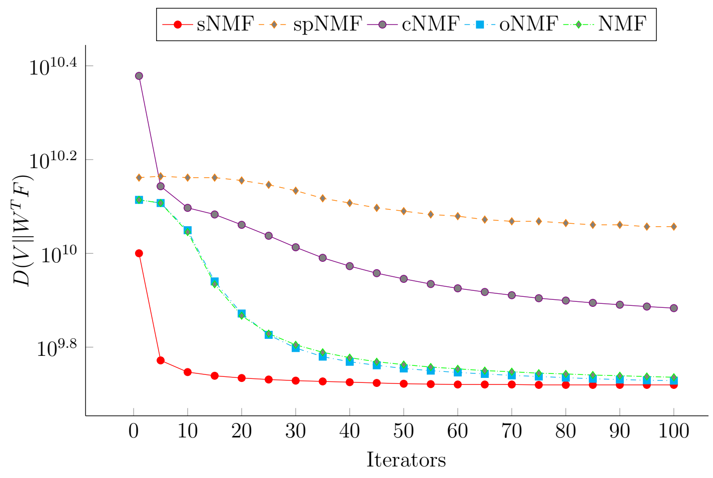

Fig. 5.6 Information Loss for Frobenius norm with $r = 30$

5.6 Conclusion and discussion

In this chapter, we have propose a new formulation as simplicial nonnegative matrix factorization (simplicial NMF) with Frobenius norm, in which each data instance is a probabilistic convex combination of latent components. A fast parallel algorithm derived from Frank-Wolfe algorithm is designed for simplicial NMF with Frobenius norm to control directly and effectively sparsity of solutions. The proposed algorithm has low complexity in inference, sparsity, and distributability and parallelization. Our experimental evaluation indicates the effectiveness of our approach via significant criteria such as interpretability, sparsity, performance in classification task and information loss. Our obtained results are highly competitive with equivalent approaches.

Publications

1. Nguyen, Duy Khuong, Khoat Than, and Tu Bao Ho. Simplicial nonnegative matrix factorization. In Computing and Communication Technologies, Research, Innovation, and Vision for the Future (RIVF), 2013 IEEE RIVF International Conference on, pp. 47-52. IEEE, 2013 (best student paper).

Chapter 6

Fast Parallel Algorithm for Simplicial NMF with KL Divergence

Data Science wisdom comes only through failed experimentation
— Damian Mingle

Nonnegative Matrix Factorization (NMF) with Kullback-Leibler(KL) Divergence (NMF-KL) can provide more sparse models and sparse representation than Nonnegative Matrix Factorization (NMF) with Frobenius norm. Hence, it is more suitable for sparse count data that follows Poisson distributions. In this chapter, we extend simplicial nonnegative matrix factorization for KL divergence by proposing a parallel algorithm having guaranteed convergence $O(1/k)$ in inference for large sparse datasets to archive sparse models and sparse representation. The proposed algorithm's experimental results are highly competitive with the results of equivalent methods.

6.1 Introduction

NMF is a powerful linear technique to reduce dimension and to extract latent factors, which can be concisely interpreted to explain phenomenon in science [34, 58, 86]. Hence, it has been widely used in many applications [92, 98]. For sparse count data, NMF with KL divergence and a Poisson distribution may provide sparse models and sparse representation describing better the random variation rather than NMF with Frobenius norm and a normal distribution [87]. However, nonnegative constraints are not enough to express the role of latent components over data instances. In this chapter, we extend the study of simplicial nonnegative matrix factorization (simplicial NMF) for KL divergence, in which the role of latent components is represented by coefficient values.

In simplicial NMF with KL divergence, a given nonnegative data matrix $V \in \mathcal{R}_+^{n\times m}$ must be factorized into a product of two nonnegative matrices, namely a latent component matrix $W \in \mathcal{R}_+^{r \times n}$ and a representation matrix $F \in \mathcal{R}_+^{r \times m}$, where n is the dimension of a data instance, m is the number of data instances, and r is the number of latent components or latent factors. In addition, each instance is a probabilistic combination of latent components, which is represented by $\sum_{r=1}^{r} F_{kj} = 1, \forall j$. Hence, the objective function of simplicial NMF with KL divergence is rewritten as follows:

$$J(V \| W^T F) = \sum_{j=1}^{m} D_{KL}(V_j \| W^T F_j) = \sum_{j=1}^{m} \sum_{i=1}^{n} (V_{ij} \log \frac{V_{ij}}{W_i^T F_j} - V_{ij} + W^T F_j), \qquad (6.1)$$

where $V, W, F \geq 0$; $\sum_{k=1}^{r} F_{kj} = 1$ for all j.

The rest of the chapter is organized as follows. Section 6.2 presents the proposed algorithms. The theoretical analysis of convergence and complexity is discussed in Section 6.3. Section 6.4 shows the experimental results, and Section 6.5 summarizes the main contributions of this chapter and discussion.

6.2 Proposed Algorithm

For solving simplicial NMF with KL divergence norm, sub-problems are convex, although the objective function is non-convex. Hence, we employ iterative multiplicative updates like *EM* algorithm, which is presented in Algorithm 8. This algorithm contains two main steps: one for finding F when fixing W, and the other for finding W when fixing F. In the first step, $F = \{F_j\}_{j=1}^{m}$ are updated to achieve a better objective value $W \approx \underset{F}{\text{argmin}}\, D_{KL}(V^T \| F^T W)$, where each of F_j is a probabilistic representation of data instances $\{V_j\}_{j=1}^{m}$. Hence, this step can be seen as *inference step* and the process is called as *inferring an data instance*. In the other step called *learning step*, the latent components are acquired by approximately minimizing $D_{KL}(V \| W^T F)$ when fixing F: $W \approx \underset{W}{\text{argmin}}\, D_{KL}(V \| W^T F)$.

6.2.1 Inference Algorithm

In this section, we discuss the inference algorithm which inherited Frank-Wolfe algorithm having sub-linear convergence $O(1/k)$. In this algorithm, the objective function is optimized by updating values of latent components $\{F_j\}_{j=1}^{m}$ when fixing latent components W^T. In addition, the inference in this divergence can be conducted independently for each data instance V_j because $D(V \| W^T F) = \sum_{j=1}^{m} D(V_j \| W^T F_j)$. Hence, inferring

Algorithm 8: Iterative multiplicative update for KL divergence

```
Input: Data matrix $V = \{V_j\}_{j=1}^m \in R_+^{n\times m}$ and $r$.
Output: Coefficients $F \in R_+^{r\times m}$ and latent components $W \in R_+^{r\times n}$
begin
    Select randomly $r$ components from $m$ data instances;
    repeat
        Inference step: Fix $W$ to find $F \approx \underset{F\in\mathcal{R}^{r\times m}}{\operatorname{argmin}} J(V\|W^TF)$;
        Compute $sumW = W\mathbf{1}^n$;
        /*Call Algorithm 9 in parallel*/;
        for $j$ = 1 to $m$ do
            $F_j \approx \underset{x\ inR_+^r, x^T1^r=1}{\operatorname{argmin}} D_{KL}(V_j\|W^Tx)$ with $sumW$;
        Learning step: Fix $F$ to find $W \approx \underset{W\in\mathcal{R}^{r\times n}}{\operatorname{argmin}} J(V^T\|F^TW)$;
        /*Learning new latent components by approximation algorithm*/;
        for $i$ = 1 to $n$ do
            $W_i \approx \underset{x\ inR_+^r}{\operatorname{argmin}} D_{KL}(V_i^T\|F^Tx) \Rightarrow W_{ki} = \frac{\sum_{j=1}^m V_{ij}}{\sum_{j=1}^m F_{kj}}$
        if convergence condition is satisfied then
            break;
    until False;
```

each data instance in KL divergence is also solving a convex optimization problem with simplicial constraints with the common form:

$$f(x) = \sum_{i=1}^{n}(v_i \log \frac{v_i}{A_i^T x} - v_i + A_i^T x) \tag{6.2}$$

The derivative of variable x is computed as follows:

$$\nabla f = -\sum_{i=1}^{n} v_i \frac{A_i^T}{A_i^T x} + \sum_{i=1}^{n} A_i^T \tag{6.3}$$

To seek the optimal solution, Algorithm 9 inherited from Frank-Wolfe algorithm is employed. This algorithm prefers optimizing the objective function along the steepest direction selected by:

$$k = \underset{k}{\operatorname{argmin}} < \mathbf{e}_k - x, \nabla f >= \underset{k}{\operatorname{argmin}} \nabla f_k \tag{6.4}$$

To optimize the objective function along the direction $e_k - x$, it is parameterized by α as follows:

$$f(y) = \sum_{i=1}^{n} (v_i \log \frac{v_i}{A_i^T y} - v_i + A_i^T y) \tag{6.5}$$

where : $y = (1-\alpha)x + \alpha \mathbf{e}_k$

Hence, we have:

$$\nabla f_\alpha = \frac{\partial f}{\partial y}\frac{\partial y}{\partial \alpha} = -\sum_{i=1}^{n} v_i \frac{A_i^T(\mathbf{e}_k - x_k)}{A_i^T((1-\alpha)x + \alpha \mathbf{e}_k)} + \sum_{i=1}^{n} A_i^T(\mathbf{e}_k - x) \tag{6.6}$$

Since the objective function is convex over a continuous variable α, we can employ bisection search method with $O(\log(\frac{1}{\epsilon}))$ iterations:

$$\alpha = \underset{\alpha \in [0,1]}{\operatorname{argmin}} f((1-\alpha)x + \alpha \mathbf{e}_k) \tag{6.7}$$

To avoid issues of numerical computation, we add a small positive number into the denominator. In addition, to reduce the complexity, Ax is computed and maintained through conducting the algorithm.

6.2.2 Learning Algorithm

Equivalent to Frobenius distance, learning components can be conducted separately in each column because $J(V^T \| F^T W) = \sum_{i=1}^{n} D_{KL}(V_i^T \| F^T W_i)$, and V and F are fixed.

In this step, to approximate the solution W_i, we have:

$$D_{KL}(V_i^T \| F^T W_i) = \sum_{j=1}^{m} (V_{ij} \log \frac{V_{ij}}{W_i^T F_j} - V_{ij} + W_i^T F_j) = \sum_{j=1}^{m} (W_i^T F_j - V_{ij} \log W_i^T F_j) + \text{const}$$

Hence, minimizing $D_{KL}(V_i^T \| F^T W_i)$ is equivalent to minimizing

$$h(W_{1i}, ..., W_{ri}) = \sum_{j=1}^{m} F_j^T W_i - \sum_{j=1}^{m} V_{ij} \log(F_j^T W_i) \tag{6.8}$$

Moreover, based on Josen's inequality for the concave function log and non-negative

Algorithm 9: Inference for data instance x

```
Input: Data instance $v \in \mathcal{R}_+^r$ and latent components $A \in \mathcal{R}_+^{r\times n}$; and
       $sumA = A\mathbf{1}^n$
1 , Output: New coefficient $x \approx \underset{x\in\mathcal{R}_+^r}{\operatorname{argmin}} \sum_{i=1}^{n}(v_i \log(\frac{v_i}{A_i^T x+\epsilon}) - v_i + A_i^T x$
2 begin
3     Choose component $W_i^T$ closest to $x$ in KL divergence;
4     Set $x = \mathbf{0}^r$; $x_k = 1$; and $Wx = W^T x$;
5     repeat
6         Computing $\nabla f$;
7         Select $k = \underset{i\in\{1..r\}}{\operatorname{argmin}} \nabla f_k$;
8         Select $\alpha = \underset{\alpha\in[0,1]}{\operatorname{argmin}} f(\alpha e_k + (1-\alpha)x)$;
9         $Wx = (1-\alpha)Wx + \alpha W_k^T e_k$;
10        Set $x = (1-\alpha)x$ and $x_k = x_k + \alpha$;
11    until Convergence condition satisfied;
```

coefficients $F_{1j}, ..., F_{rj}$ with $\sum_{k=1}^{r} F_{kj} = 1$, we have

$$logF_j^T W_i = \log(\sum_{k=1}^{r} F_{kj}W_{ki}) \leq \sum_{k=1}^{r} F_{kj}\log(W_{ki}) \tag{6.9}$$

Then:

$$h(W_{1i}, ..., W_{ri}) \leq h'(W_{1i}, ..., W_{ri}) = \sum_{j=1}^{m}\sum_{k=1}^{r} W_{ki}F_{jk} - \sum_{j=1}^{m} V_{ij}\sum_{k=1}^{r} F_{jk}log(W_{ki}) \tag{6.10}$$

W_k is approximated by minimizing $h'(W_{1i}, ..., W_{ki})$:

$$\frac{\partial h'}{\partial W_{ki}} = 0 \Leftrightarrow W_{ki} = \frac{\sum_{j=1}^{m} V_{ij}}{\sum_{j=1}^{m} F_{kj}} \tag{6.11}$$

Therefore, in the learning step for KL divergence, we can directly approximate the solution. Although this is only an approximate solution, it is really effective for KL divergence and this technique has been employed in numerous applications.

6.3 Theoretical Analysis

As discussed above, the main difference between our algorithms and the others is in the inference step because we solve the same optimization problem in the learning step. Hence, we will discuss complexity, convergence of inference, sparsity, and distributability and parallelizability in this section.

6.3.1 Complexity

Theorem 7 *Consider Algorithm 9 to infer a data instance having n-dimension by r latent components with k iterations. Then, its complexity is $\mathcal{O}(k[r.nnz(n) + n\log\frac{1}{\epsilon}])$, where $nnz(n)$ is the number of non-zero elements in v.*

*Proof:*In the Algorithm 9, for each iteration, base on Equation 6.3 of computing the derivative, the complexity of finding out the best coefficient k is $\mathcal{O}(r.nnz(n))$ when Ax and $sumA$ are computed in advanced. In addition, the complexity of estimating $a \in [0,1]$ is $\mathcal{O}(n\log\frac{1}{\epsilon})$, where ϵ is a small positive quantity for the required precision, because we use binary search algorithm. Overall, the complexity for k iterations is $\mathcal{O}(k[r.nnz(n) + n\log\frac{1}{\epsilon}])$ □

In addition, for the learning step with KL divergence, we employ an approximate algorithm with low complexity:

Theorem 8 *Let consider to learn new latent components after inferring coefficients of data instances. Then, its complexity is $\mathcal{O}(m[nnz(n) + nnz(r)])$.*

Proof: This theorem is implied from formula 6.11 □

6.3.2 Convergence Guarantee of Inference

Based on [23], we have

Theorem 9 *Let f be a twice differentiable convex function over simplex $\triangle$ and denote $C_f = sup_{y,z\in\triangle;\tilde{y}\in[y,z]}(y-z).\nabla^2 f(\tilde{y}).(y-z)^T$. After l iterations, the Frank-Wolfe algorithm will find an approximate solution x_l with at most $(l+1)$ non-zeros coefficients which satisfy*

$$max_{x\in\triangle} f(x_l) - f(x) \le \frac{C_f}{l+1}$$

From this theorem, we have the following remarks:

- The proposed algorithm has fast sub-linear convergence $O(1/k)$, which is a significant contribution because basic NMF with KL divergence does not guarantee this property. In addition, this convergence guarantee the bounded goodness of solutions.
- After training NMF models, fast inference is crucial for large scale applications.
- Sparsity and quality of solutions can easily trade-offed by stop finding new latent components to optimize the cost function, which is valid for real applications when limiting the number of non-zero coefficients.

6.3.3 Sparsity

Sparse solutions receive much interests in machine learning community because of improving computation performance and saving storage with low complexity. In this formation, sparse solutions are enhanced by three factors as simplicial constraints, KL divergence, and Frank-Wolfe algorithm. In addition, we can easily control the sparsity of solutions by limiting the number of non-zero elements, which is crucial for real applications, which prefer high sparse representation such as topic modeling and compressed sensing.

6.3.4 Parallelizability and distributability

In the proposed algorithm, the objective function is fully decomposed into small computation elements. Hence, the computation is easily computed and distributed over distributed systems. In addition, running time of inference mainly depends on finding the best latent component. Furthermore, computing the partial derivative for each latent component is totally separated. As a result, this computation can be paralleled. Hence, the responding time in real applications can be effectively reduce. Moreover, regarding to the learning algorithm for KL divergence, it is absolutely fast and can be easily distributed, which is highly suitable for parallel and distributed systems such as Hadoop and Apache Spark.

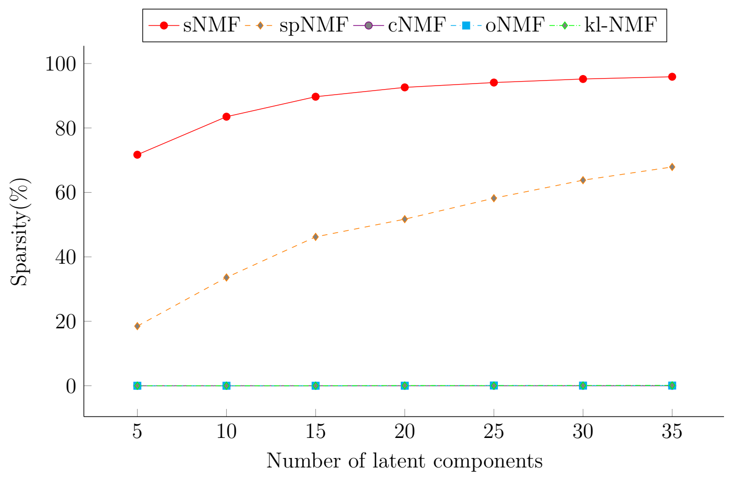

Fig. 6.1 Sparsity of new coefficients for KLdivergence with $r = 30$

6.4 Experimental Evaluation

In this section, we investigate the effectiveness of our approach and the proposed algorithm for simplicial NMF with KL divergence by four criteria: sparsity of solutions, performance in classification tasks and loss information measure. More particularly, our algorithm for simplicial NMF *KL*-divergence is compared to kl-NMF [83], local non-negative matrix factorization (locNMF) [60], convolutional NMF(conNMF) [85], and Nonsmooth Nonnegative Matrix Factorization (nsNMF) [76]. The implemented codes are at [1].

Test-cases: In this investigation, we use a typical database text, which contains 4 327 labeled spam emails are all downloaded from [2] . For the email dataset, after normalizing data such as numbers turning into *number* term and plural noun into single noun, we compute tf-idf [3] for 32 906 distinct terms as convenient features for data instances.

6.4.1 Sparse Representation

In this section, we investigate the sparse of instance representation in the new space of latent components. This sparsity of solutions is measured as the percentage of zero coefficients by the following formula:

[1]http://www.ee.columbia.edu/~grindlay/code.html
[2]http://csmining.org/index.php/spam-email-datasets-.html
[3]http://en.wikipedia.org/wiki/Tf-idf

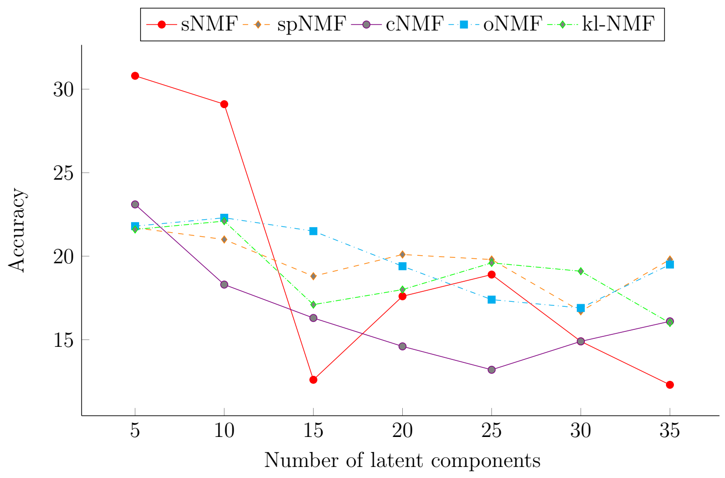

Fig. 6.2 Inaccuracy for Spam Classification

$$\frac{\text{Number of zero coefficients}}{\text{Number of coefficients}} \times 100\%$$

Figure 6.1 shows that the sparsity result is highly competitive with other methods' one. Specially, our approach retains the highest sparse solutions, while it still keeps one of the best result for the other measures. This result indicates that simplicial NMF with KL divergence can achieve both of highly sparse representation and high quality of other factors, which is crucial for real large scale application.

6.4.2 Performance for classification

In this section, our method is compared with other methods on classification performance, which is one important measure for dimension redution technique. In this experiment, we employ Random Forest[4], a robust algorithm for classification. Figure 6.2 indicates that our method obtains the lowest misclassification with $r = 15$ and $r = 35$.

6.4.3 Convergence

In this section, we investigate the convergence of information loss. Figure 6.3 has achieved lowest and fastest convergence versus iterations. This result is meaning for large scale applications because algorithms having more iterations require more computation and also time for loading data for distributed computation like Hadoop.

[4]http://cran.r-project.org/web/packages/randomForest/

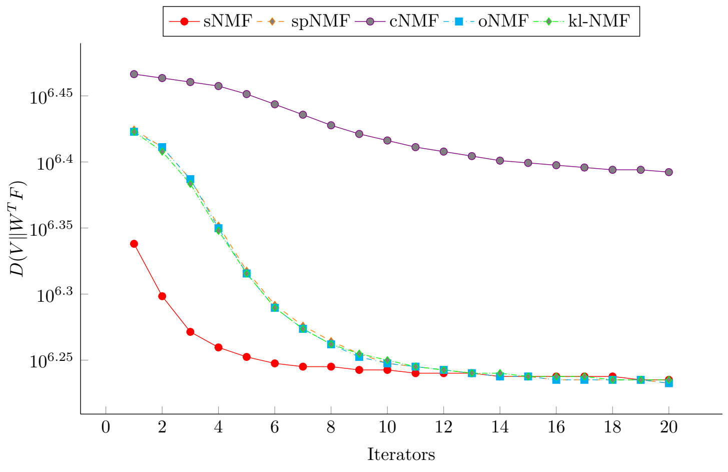

Fig. 6.3 Information Loss for KL divergence with $r = 30$

6.5 Conclusion and discussion

In this chapter, we have propose a fast parallel algorithm derived from Frank-Wolfe algorithm for simplicial NMF with KL divergence to control directly and effectively sparsity of solutions. The proposed algorithm has fast guaranteed convergence $O(1/k)$, sparsity, and parallelization. Our experimental evaluation shows the effectiveness of our approach via significant criteria such as sparsity, classification performance and information loss. The proposed approach has highly sparse representation, fast convergence and high accuracy, which are highly competitive with equivalent approaches and potentially feasible for large scale applications.

Publications

1. Nguyen, Duy Khuong, Khoat Than, and Tu Bao Ho. Simplicial nonnegative matrix factorization. In Computing and Communication Technologies, Research, Innovation, and Vision for the Future (RIVF), 2013 IEEE RIVF International Conference on, pp. 47-52. IEEE, 2013 (best student paper).

Chapter 7

Conclusion and Future Works

I have not failed. I've just found 10,000 ways that won't work.
— Thomas Edison

7.1 Conclusion

In summary, this thesis has systematically studied to rich models and fast algorithms for nonnegative matrix factorization in two popular objective functions as Frobenius norm and KL-divergence. Specially, the thesis has major contributions:

Chapter 2 introduces a unified NMF model by generalizing numerous variants of NMF, which has been developed and employed for many real applications. From this modeling, new researchers can easily implement and compare NMF models, conveniently comprehend existing NMF formulations during over a decade, and also create new formulations for their various researches. Base on this approach, we propose two rich models for NMF as NMF with L_1 L_2 regularizations and simplicial NMF.

Chapter 3 proposes an accelerated parallel and distributed algorithms for NMF with L_1 L_2 regularizations, which inherits a novel accelerated anti-lopsided algorithm for non-negative least squares. Specially, the proposed algorithm achieves over-bounded linear convergence rate of $\mathcal{O}([(1-\frac{\mu}{L})(1-\frac{\mu}{L})^{2r}]^k)$ in the sub-spaces of passive variables when fixing one of latent factor matrix; where μ and L are always bounded as $\frac{1}{2} \leq \mu \leq L \leq r$. In addition, the algorithm is a flexible framework for all the variants of NMF regularizations. In the experimental evaluation, our algorithm overcomes 7 of the most art-the-state algorithms in large datasets about three significant aspects of convergence, average of the iteration number and optimality. Furthermore, it can fully be parallelized and distributed because the computation using limited internal memory is decomposed into basic computation units as NQP problems.

Chapter 4 introduces a fast parallel randomized algorithm for nonnegative matrix factorization with KL-divergence having fast average convergence to solve major problems for count sparse datasets including sparse model, sparse representation, sparse algorithm, limited internal memory, and parallel algorithm. The algorithm can achieve sparse model and sparse representation, which leads to that the proposed algorithm effectively utilize sparsity prosperities of data and model to significantly improve its performance. In addition, the algorithm generally adapt all the variants of L_1 L_2 regularizations of NMF with KL-divergence. In experimental evaluation, the proposed algorithms' results outperform the results of the state-of-the-art algorithms.

Chapter 5 and Chapter 6 introduce a new formulation of NMF as simplicical NMF for Frobenius norm and KL divergence. The formulation give a more concise interpretability of the roles of latent factors contributing on the data instances by assuming that each instance is a probabilistic contribution of latent components, which it still enhance sparsity and smoothness. Furthermore, this formulation can utilize parallel and distributed algorithms, and current advanced optimization algorithms having guaranteed convergence and sparsity. In practice, the algorithm based on Frank-Wolf algorithm has sub-linear convergence rate $O(1/k)$. In addition, the proposed approaches can achieve concise interpretability, fast convergence versus iterations, sparse representation, and high accuracy classification.

In summary, this thesis discusses two significant mutual aspects of nonnegative matrix factorization as rich models and fast algorithms. Specifically, we propose rich models and their four fast parallel algorithms for nonnegative matrix factorization for two divergences, which can adapt with large scale applications and various datasets.

7.2 Future Works

In despite of these significant contributions that solved several fundamental problems in NMF with Frobenius norm and KL-divergence, many new challenges based on this study are open for further researches:

Fast parallel and distributed algorithms for orthogonal nonnegative matrix factorization

Nonnegative matrix factorization is a linear model like principal component analysis (PCA), which forces non-negativity constraints but relaxes orthogonality properties. These non-negativity constraints on both latent components and coefficients lead NMF to an additive model and reduce the overlapping of extracted latent factors. In addition, orthogonality provides more concise interpret-ability of clustering and latent factor independence [27, 28], which is necessary for real applications [78]. However, current algorithms [51, 67, 78, 94] for orthogonal nonnegative matrix factorization still is limited on convergence speed, and parallel and distributed computing paradigms to deal with big data.

Supervised algorithms for nonnegative matrix factorization

Nonnegative matrix factorization can be purely considered as an unsupervised learning method. Using supervised information such as label and clustering information will improve accuracy of supervised post-processes such as clustering and classification [61, 67, 70]. In future researches, we will extend the current studies of rich models and fast algorithms for supervised learning in nonnegative matrix factorization.

Rich models for stream data

Modern technologies such as the Internet of things, sensors, machines to machines systems, web logs, and computer network traffic, etc have been generating big stream data [4, 16, 24]. These big data are complicated by various properties of high-dimension datasets [24]. Current formulations for NMF have several limitations in interpretability and unpredictable forms of latent factors, which are necessary for data stream processing. Hence, rich models support fast algorithms, concise interpretability and normalized output, which are opening challenges for nonnegative matrix factorizations.

Fast online learning algorithms for nonnegative matrix factorization

Big and stream data require fast algorithms to deal with them [17]. In addition, nonnegative matrix factorization is a powerful and popular method for data analysis, while online learning algorithms can deal with big and stream data. Hence, online learning algorithms for new variants of NMF for real large-scale applications and stream data are opening problems for further researches [37].

Fast paparell and distributed algorithms for nonnegative tensor factorization

Tensor can be considered a higher-dimension version of matrix, and nonnegative tensor factorization (NTF) can model high-dimension and more complicated data [22, 32, 52].

Specially, NTF extract latent components to discover hidden knowledge inside high-dimension datasets. However, tensor is more complex than matrix because several algebra properties of matrix are incorrect for tensor [52]. Hence, upgrading researches of these results of NMF for tensor is necessary and still challenges for further researches [54].

Softwares

We publish three software packages as follows:

Accelerated anti-lopsided algorithm for nonnegative least squares

URL address: https://github.com/khuongnd/nnls_antilopsided

This package implements the proposed fast anti-lopsided algorithm for nonnegative least squares $\|Ax - b\|_2^2$ subject to $x \succeq 0$, which can be considered as a core algorithm for nonnegative matrix and tensor factorization.

Accelerated parallel algorithm for nonnegative matrix factorization with L_1 L_2 regularizations for Frobenius norm

URL address: https://github.com/khuongnd/NMF_APD

This package implements the proposed fast parallel algorithm for nonnegative matrix factorization with all the variants of L_1 L_2 regularizations and Frobenius norm, which optimizes the objective function: $\|V - WF\|_2^2 + \frac{\alpha_1}{2}\|W\|_2^2 + \frac{\alpha_2}{2}\|F\|_2^2 + \beta_1\|W\|_1 + \beta_2\|F\|_1$

Fast parallel algorithm for NMF with L_1 L_2 regularizations for KL-divergence for large sparse datasets

URL address: https://github.com/khuongnd/NMF_KL_FastParallel

This package implements the proposed fast parallel algorithm for nonnegative matrix factorization with KL-divergence for all the variants of L_1 L_2 regularizations for large sparse datasets, which optimizes the objective function: $\sum_{i=1}^{n}\sum_{j=1}^{m} V_{ij}\log(\frac{V_{ij}}{W_i^T F_j} - V_{ij} + W_i^T F_j) + \frac{\alpha_1}{2}\|W\|_2^2 + \frac{\alpha_2}{2}\|F\|_2^2 + \beta_1\|W\|_1 + \beta_2\|F\|_1$

List of Publications

[DKN15AN] Nguyen, Duy Khuong, and Tu Bao Ho. Anti-lopsided Algorithm for Large-scale Nonnegative Least Square Problems. International Journal of Data Science and Analytics, Springer, (accepted to publish).

[DKN15AC] Nguyen, Duy-Khuong, 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 (accepted to publish).

[DKN15AC] Nguyen, Duy-Khuong, and Tu-Bao Ho. Accelerated Parallel and Distributed Algorithm using Limited Internal Memory for Nonnegative Matrix Factorization. submitted to Journal of Global Optimization, (under revision).

[DKN15SIM] Nguyen, Duy Khuong, Khoat Than, and Tu Bao Ho. Simplicial nonnegative matrix factorization. In Computing and Communication Technologies, Research, Innovation, and Vision for the Future (RIVF), 2013 IEEE RIVF International Conference on, pp. 47-52. IEEE, 2013 (best paper student).

[QKT14AN] Than, Khoat, Tu Bao Ho, and Duy Khuong Nguyen. An effective framework for supervised dimension reduction. Neurocomputing 139 (2014): 397-407.

[QKT14SU] Than, Khoat, Tu Bao Ho, Duy Khuong Nguyen, and Pham Ngoc Khanh. Supervised dimension reduction with topic models. In ACML, pp. 395-410. 2012.

References

[1] Alexandrov, L.B., Nik-Zainal, S., Wedge, D.C., Campbell, P.J., Stratton, M.R.: Deciphering signatures of mutational processes operative in human cancer. Cell reports 3(1), 246–259 (2013)

[2] Arora, S., Ge, R., Kannan, R., Moitra, A.: Computing a nonnegative matrix factorization–provably. In: Proceedings of the forty-fourth annual ACM symposium on Theory of computing. pp. 145–162. ACM (2012)

[3] Arora, S., Ge, R., Moitra, A.: Learning topic models–going beyond svd. In: Foundations of Computer Science (FOCS), 2012 IEEE 53rd Annual Symposium on. pp. 1–10. IEEE (2012)

[4] Babcock, B., Babu, S., Datar, M., Motwani, R., Widom, J.: Models and issues in data stream systems. In: Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems. pp. 1–16. ACM (2002)

[5] Baraniuk, R.G.: Compressive sensing. IEEE signal processing magazine 24(4) (2007)

[6] Bengio, Y., Courville, A., Vincent, P.: Representation learning: A review and new perspectives. Pattern Analysis and Machine Intelligence, IEEE Transactions on 35(8), 1798–1828 (2013)

[7] Berry, M.W., Browne, M., Langville, A.N., Pauca, V.P., Plemmons, R.J.: Algorithms and applications for approximate nonnegative matrix factorization. Computational Statistics & Data Analysis 52(1), 155–173 (2007)

[8] Berry, M.W., Gillis, N., Glineur, F.: Document classification using nonnegative matrix factorization and underapproximation. In: Circuits and Systems, 2009. ISCAS 2009. IEEE International Symposium on. pp. 2782–2785. IEEE (2009)

[9] Bertsekas, D.P.: Projected newton methods for optimization problems with simple constraints. SIAM Journal on control and Optimization 20(2), 221–246 (1982)

[10] Bishop, C.M.: Model-based machine learning. Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences 371(1984), 20120222 (2013)

[11] Blei, D.M.: Build, compute, critique, repeat: Data analysis with latent variable models. Annual Review of Statistics and Its Application 1, 203–232 (2014)

[12] Bonettini, S.: Inexact block coordinate descent methods with application to non-negative matrix factorization. IMA journal of numerical analysis 31(4), 1431–1452 (2011)

[13] Boyd, S.: Alternating direction method of multipliers. In: Talk at NIPS Workshop on Optimization and Machine Learning (2011)

[14] Boyd, S., Vandenberghe, L.: Convex optimization. Cambridge university press (2004)

[15] Bro, R., De Jong, S.: A fast non-negativity-constrained least squares algorithm. Journal of chemometrics 11(5), 393–401 (1997)

[16] Chakravarthy, S., Jiang, Q.: Stream data processing: a quality of service perspective: modeling, scheduling, load shedding, and complex event processing, vol. 36. Springer Science & Business Media (2009)

[17] Chen, C.P., Zhang, C.Y.: Data-intensive applications, challenges, techniques and technologies: A survey on big data. Information Sciences 275, 314–347 (2014)

[18] Chen, D., Plemmons, R.J.: Nonnegativity constraints in numerical analysis. In: Symposium on the Birth of Numerical Analysis. pp. 109–140 (2009)

[19] Choi, S.: Algorithms for orthogonal nonnegative matrix factorization. In: Neural Networks, 2008. IJCNN 2008.(IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on. pp. 1828–1832. IEEE (2008)

[20] Chu, C., Kim, S.K., Lin, Y.A., Yu, Y., Bradski, G., Ng, A.Y., Olukotun, K.: Map-reduce for machine learning on multicore. Advances in neural information processing systems 19, 281 (2007)

[21] Cichocki, A., Zdunek, R., Amari, S.i.: Hierarchical als algorithms for nonnegative matrix and 3d tensor factorization. In: Independent Component Analysis and Signal Separation, pp. 169–176. Springer (2007)

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

[23] Clarkson, K.L.: Coresets, sparse greedy approximation, and the frank-wolfe algorithm. ACM Transactions on Algorithms (TALG) 6(4), 63 (2010)

[24] Council, N.: Frontiers in massive data analysis (2013)

[25] Ding, C., Li, T., Jordan, M.: Convex and semi-nonnegative matrix factorizations. Pattern Analysis and Machine Intelligence, IEEE Transactions on 32(1), 45–55 (2010)

[26] Ding, C., Li, T., Peng, W.: Nonnegative matrix factorization and probabilistic latent semantic indexing: Equivalence chi-square statistic, and a hybrid method. In: Proceedings of the national conference on artificial intelligence. vol. 21, p. 342. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999 (2006)

[27] Ding, C., Li, T., Peng, W., Park, H.: Orthogonal nonnegative matrix t-factorizations for clustering. In: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 126–135. ACM (2006)

[28] Ding, C.H., He, X., Simon, H.D.: On the equivalence of nonnegative matrix factorization and spectral clustering. In: SDM. vol. 5, pp. 606–610. SIAM (2005)

[29] Donoho, D., Stodden, V.: When does non-negative matrix factorization give a correct decomposition into parts? In: Advances in neural information processing systems 16: proceedings of the 2003 conference. MIT Press (2004)

[30] Fodor, I.K.: A survey of dimension reduction techniques (2002)

[31] Franc, V., Hlaváč, V., Navara, M.: Sequential coordinate-wise algorithm for the non-negative least squares problem. In: Computer Analysis of Images and Patterns. pp. 407–414. Springer (2005)

[32] Friedlander, M.P., Hatz, K.: Computing non-negative tensor factorizations. Optimisation Methods and Software 23(4), 631–647 (2008)

[33] Gemulla, R., Nijkamp, E., Haas, P.J., Sismanis, Y.: Large-scale matrix factorization with distributed stochastic gradient descent. In: Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 69–77. ACM (2011)

[34] Gillis, N.: The why and how of nonnegative matrix factorization. Regularization, Optimization, Kernels, and Support Vector Machines 12, 257 (2014)

[35] Gillis, N., Glineur, F.: Accelerated multiplicative updates and hierarchical als algorithms for nonnegative matrix factorization. Neural computation 24(4), 1085–1105 (2012)

[36] Guan, N., Tao, D., Luo, Z., Yuan, B.: Nenmf: an optimal gradient method for nonnegative matrix factorization. Signal Processing, IEEE Transactions on 60(6), 2882–2898 (2012)

[37] Guan, N., Tao, D., Luo, Z., Yuan, B.: Online nonnegative matrix factorization with robust stochastic approximation. Neural Networks and Learning Systems, IEEE Transactions on 23(7), 1087–1099 (2012)

[38] Guan, N., Wei, L., Luo, Z., Tao, D.: Limited-memory fast gradient descent method for graph regularized nonnegative matrix factorization. PloS one 8(10), e77162 (2013)

[39] Guillamet, D., Vitria, J.: Classifying faces with nonnegative matrix factorization. In: Proc. 5th Catalan conference for artificial intelligence. pp. 24–31 (2002)

[40] Hastie, T., Tibshirani, R., Friedman, J., Franklin, J.: The elements of statistical learning: data mining, inference and prediction. The Mathematical Intelligencer 27(2), 83–85 (2005)

[41] Helén, M., Virtanen, T.: Separation of drums from polyphonic music using non-negative matrix factorization and support vector machine. In: Proc. EUSIPCO. vol. 2005 (2005)

[42] Hoyer, P.O.: Non-negative sparse coding. In: Neural Networks for Signal Processing, 2002. Proceedings of the 2002 12th IEEE Workshop on. pp. 557–565. IEEE (2002)

[43] Hoyer, P.O.: Non-negative matrix factorization with sparseness constraints. J. Mach. Learn. Res. 5, 1457–1469 (Dec 2004)

[44] Hsieh, C.J., Dhillon, I.S.: 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. pp. 1064–1072. ACM (2011)

[45] Kim, D., Sra, S., Dhillon, I.S.: A new projected quasi-newton approach for the nonnegative least squares problem. Computer Science Department, University of Texas at Austin (2006)

[46] Kim, D., Sra, S., Dhillon, I.S.: Fast newton-type methods for the least squares nonnegative matrix approximation problem. In: SDM. pp. 343–354. SIAM (2007)

[47] Kim, H., Park, H.: Nonnegative matrix factorization based on alternating nonnegativity constrained least squares and active set method. SIAM Journal on Matrix Analysis and Applications 30(2), 713–730 (2008)

[48] Kim, J., He, Y., Park, H.: 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)

[49] Kim, J., Park, H.: Toward faster nonnegative matrix factorization: A new algorithm and comparisons. In: Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on. pp. 353–362. IEEE (2008)

[50] Kim Dongmin, S.S., Dhillon, I.S.: A non-monotonic method for large-scale nonnegative least squares. Optimization Methods and Software 28(5), 1012–1039 (2013)

[51] Kimura, K., Tanaka, Y., Kudo, M.: A fast hierarchical alternating least squares algorithm for orthogonal nonnegative matrix factorization. In: Proceedings of the Sixth Asian Conference on Machine Learning. pp. 129–141 (2014)

[52] Kolda, T.G., Bader, B.W.: Tensor decompositions and applications. SIAM review 51(3), 455–500 (2009)

[53] Kuang, D., Choo, J., Park, H.: Nonnegative matrix factorization for interactive topic modeling and document clustering. In: Partitional Clustering Algorithms, pp. 215–243. Springer (2015)

[54] Kuleshov, V., Chaganty, A.T., Liang, P.: Tensor factorization via matrix factorization. arXiv preprint arXiv:1501.07320 (2015)

[55] Lacoste-Julien, S., Jaggi, M.: An affine invariant linear convergence analysis for frank-wolfe algorithms. arXiv preprint arXiv:1312.7864 (2013)

[56] Lawson, C.L., Hanson, R.J.: Solving least squares problems, vol. 161. SIAM (1974)

[57] Lee, D.D., Seung, H.S.: Algorithms for non-negative matrix factorization. In: Advances in neural information processing systems. pp. 556–562 (2001)

[58] Lee, D., Seung, H., et al.: Learning the parts of objects by non-negative matrix factorization. Nature 401(6755), 788–791 (1999)

[59] Li, H., Adal, T., Wang, W., Emge, D., Cichocki, A.: Non-negative matrix factorization with orthogonality constraints and its application to raman spectroscopy. The Journal of VLSI Signal Processing Systems for Signal, Image, and Video Technology 48(1-2), 83–97 (2007)

[60] Li, S.Z., Hou, X.W., Zhang, H.J., Cheng, Q.S.: Learning spatially localized, parts-based representation. In: Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on. vol. 1, pp. I–207. IEEE (2001)

[61] Li, T., Ding, C., Jordan, M., et al.: Solving consensus and semi-supervised clustering problems using nonnegative matrix factorization. In: Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on. pp. 577–582. IEEE (2007)

[62] Lin, C.J.: On the convergence of multiplicative update algorithms for nonnegative matrix factorization. Neural Networks, IEEE Transactions on 18(6), 1589–1596 (2007)

[63] Lin, C.J.: Projected gradient methods for nonnegative matrix factorization. Neural computation 19(10), 2756–2779 (2007)

[64] Liu, C., Yang, H.c., Fan, J., He, L.W., Wang, Y.M.: Distributed nonnegative matrix factorization for web-scale dyadic data analysis on mapreduce. In: Proceedings of the 19th international conference on World wide web. pp. 681–690. ACM (2010)

[65] Lu, X., Wu, H., Yuan, Y., Yan, P.g., Li, X.: Manifold regularized sparse nmf for hyperspectral unmixing. Geoscience and Remote Sensing, IEEE Transactions on 51(5), 2815–2826 (2013)

[66] Luenberger, D.G., Ye, Y.: Linear and nonlinear programming, vol. 116. Springer Science & Business Media (2008)

[67] Ma, H., Zhao, W., Tan, Q., Shi, Z.: Orthogonal nonnegative matrix tri-factorization for semi-supervised document co-clustering. In: Advances in Knowledge Discovery and Data Mining, pp. 189–200. Springer (2010)

[68] Manolakis, D., Truslow, E., Pieper, M., Cooley, T., Brueggeman, M.: Detection algorithms in hyperspectral imaging systems: An overview of practical algorithms. Signal Processing Magazine, IEEE 31(1), 24–33 (2014)

[69] Mel, B.W.: Computational neuroscience: Think positive to find parts. Nature 401(6755), 759–760 (1999)

[70] Mohammadiha, N., Smaragdis, P., Leijon, A.: Supervised and unsupervised speech enhancement using nonnegative matrix factorization. Audio, Speech, and Language Processing, IEEE Transactions on 21(10), 2140–2151 (2013)

[71] Nesterov, Y.: Efficiency of coordinate descent methods on huge-scale optimization problems. SIAM Journal on Optimization 22(2), 341–362 (2012)

[72] Nesterov, Y.: A method of solving a convex programming problem with convergence rate $o(1/k^2)$. In: Soviet Mathematics Doklady. vol. 27, pp. 372–376 (1983)

[73] Nguyen, D.K., Ho, T.B.: Anti-lopsided algorithm for large-scale nonnegative least square problems. arXiv preprint arXiv:1502.01645 (2015)

[74] Nicoletti, O., de La Peña, F., Leary, R.K., Holland, D.J., Ducati, C., Midgley, P.A.: Three-dimensional imaging of localized surface plasmon resonances of metal nanoparticles. Nature 502(7469), 80–84 (2013)

[75] Paatero, P., Tapper, U.: Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values. Environmetrics 5(2), 111–126 (1994)

[76] Pascual-Montano, A., Carazo, J.M., Kochi, K., Lehmann, D., Pascual-Marqui, R.D.: Nonsmooth nonnegative matrix factorization (nsnmf). Pattern Analysis and Machine Intelligence, IEEE Transactions on 28(3), 403–415 (2006)

[77] Pauca, V.P., Piper, J., Plemmons, R.J.: Nonnegative matrix factorization for spectral data analysis. Linear algebra and its applications 416(1), 29–47 (2006)

[78] Pompili, F., Gillis, N., Absil, P.A., Glineur, F.: Two algorithms for orthogonal nonnegative matrix factorization with application to clustering. Neurocomputing 141, 15–25 (2014)

[79] Rudraraju, S., Salvi, A., Garikipati, K., Waas, A.M.: Experimental observations and numerical simulations of curved crack propagation in laminated fiber composites. Composites Science and Technology 72(10), 1064–1074 (2012)

[80] Schmidt, M., Friedlander, M.: Coordinate descent converges faster with the gauss-southwell rule than random selection. In: NIPS OPT-ML workshop (2014)

[81] Schmidt, M.N.: Speech separation using non-negative features and sparse non-negative matrix factorization (2007)

[82] Schmidt, M.N., Larsen, J., Hsiao, F.T.: Wind noise reduction using non-negative sparse coding. In: Machine Learning for Signal Processing, 2007 IEEE Workshop on. pp. 431–436. IEEE (2007)

[83] Seung, D., Lee, L.: Algorithms for non-negative matrix factorization. Advances in neural information processing systems 13, 556–562 (2001)

[84] Shahnaz, F., Berry, M.W., Pauca, V.P., Plemmons, R.J.: Document clustering using nonnegative matrix factorization. Information Processing & Management 42(2), 373–386 (2006)

[85] Smaragdis, P.: Non-negative matrix factor deconvolution; extraction of multiple sound sources from monophonic inputs. In: Independent Component Analysis and Blind Signal Separation, pp. 494–499. Springer (2004)

[86] Sotiras, A., Resnick, S.M., Davatzikos, C.: Finding imaging patterns of structural covariance via non-negative matrix factorization. NeuroImage 108, 1–16 (2015)

[87] Sra, S., Kim, D., Schölkopf, B.: Non-monotonic poisson likelihood maximization. Tech. rep., Tech. Rep. 170, Max Planck Institute for Biological Cybernetics (2008)

[88] Sun, Z., Li, T., Rishe, N.: Large-scale matrix factorization using mapreduce. In: Data Mining Workshops (ICDMW), 2010 IEEE International Conference on. pp. 1242–1248. IEEE (2010)

[89] Than, K., Ho, T.B.: Fully sparse topic models. In: Machine Learning and Knowledge Discovery in Databases, pp. 490–505. Springer (2012)

[90] Thurau, C., Kersting, K., Wahabzada, M., Bauckhage, C.: Convex non-negative matrix factorization for massive datasets. Knowledge and information systems 29(2), 457–478 (2011)

[91] Vincent, E., Bertin, N., Gribonval, R., Bimbot, F.: From blind to guided audio source separation: How models and side information can improve the separation of sound. IEEE Signal Processing Magazine 31(3), 107–115 (2014)

[92] Wang, Y.X., Zhang, Y.J.: Nonnegative matrix factorization: A comprehensive review. Knowledge and Data Engineering, IEEE Transactions on 25(6), 1336–1353 (2013)

[93] Xie, J., Kelley, S., Szymanski, B.K.: Overlapping community detection in networks: The state-of-the-art and comparative study. ACM Computing Surveys (csur) 45(4), 43 (2013)

[94] Yoo, J., Choi, S.: Orthogonal nonnegative matrix factorization: Multiplicative updates on stiefel manifolds. In: Intelligent Data Engineering and Automated Learning–IDEAL 2008, pp. 140–147. Springer (2008)

[95] Yoshii, K., Tomioka, R., Mochihashi, D., Goto, M.: Infinite positive semidefinite tensor factorization for source separation of mixture signals. In: Proceedings of the 30th International Conference on Machine Learning (ICML-13). pp. 576–584 (2013)

[96] Zdunek, R., Cichocki, A.: Non-negative matrix factorization with quasi-newton optimization. In: Artificial Intelligence and Soft Computing–ICAISC 2006, pp. 870–879. Springer (2006)

[97] Zhang, Z.Y.: Divergence functions of non negative matrix factorization: A comparison study. Communications in Statistics-Simulation and Computation 40(10), 1594–1612 (2011)

[98] Zhang, Z.Y.: Nonnegative matrix factorization: Models, algorithms and applications 2 (2011)