# NONNEGATIVE MATRIX FACTORIZATION FOR CLUSTERING

A Thesis
Presented to
The Academic Faculty

by

Da Kuang

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Computational Science and Engineering

Georgia Institute of Technology
August 2014

# NONNEGATIVE MATRIX FACTORIZATION FOR CLUSTERING

*To my mom and dad*

# ACKNOWLEDGEMENTS

First of all, I would like to thank my research advisor, Professor Haesun Park. When I was at the beginning stage of a graduate student and knew little about scientific research, Dr. Park taught me the spirit of numerical computing and guided me to think about nonnegative matrix factorization, a challenging problem that keeps me wondering about for five years. I greatly appreciate the large room she offered me to choose the research topics which I believe are interesting and important, and meanwhile her insightful advice that has always helped me make a better choice. I am grateful for her trust that I can be an independent researcher and thinker.

I would like to thank the PhD Program in Computational Science and Engineering and the professors at Georgia Tech who had the vision to create it. With a focus on numerical methods, it brings together the fields of data science and high-performance computing, which has proven to be the trend as time went by. I benefited a lot from the training I received in this program.

I would like to thank the computational servers I have been relying on and the ones who manage them, without whom this thesis would be all dry theories. I thank Professor Richard Vuduc for his generosity. Besides the invaluable and extremely helpful viewpoints he shared with me on high-performance computing, he also allowed me to use his valuable machines. I thank Peter Wan who always solved the system issues immediately upon my request. I thank Dr. Richard Boyd and Dr. Barry Drake for the inspiring discussions and their kindness to sponsor me to use the GPU servers at Georgia Tech Research Institute.

I also thank all the labmates and collaborators. Jingu Kim helped me through the messes and taught me how NMF worked intuitively when I first joined the lab. Jiang

Bian was my best neighbor in the lab before he graduated and we enjoyed many meals on and off campus. Dr. Lee Cooper led me through a fascinating discovery of genomics at Emory University. My mentor at Oak Ridge National Lab, Dr. Cathy Jiao, taught me the wisdom of managing a group of people, and created a perfect environment for practicing my oral English. I thank Jaegul Choo, Nan Du, Yunlong He, Fuxin Li, Yingyu Liang, Ramki Kannan, Mingxuan Sun, and Bo Xie for the helpful discussions and exciting moments. I also thank my friends whom I met during internships for their understanding for my desire to get a PhD.

Finally, I would like to thank my fiancée, Wei, for her love and support. I would like to thank my mom and dad, without whom I could not have gone so far.

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# SUMMARY

This dissertation shows that nonnegative matrix factorization (NMF) can be extended to a general and efficient clustering method. Clustering is one of the fundamental tasks in machine learning. It is useful for unsupervised knowledge discovery in a variety of applications such as text mining and genomic analysis. NMF is a dimension reduction method that approximates a nonnegative matrix by the product of two lower rank nonnegative matrices, and has shown great promise as a clustering method when a data set is represented as a nonnegative data matrix. However, challenges in the widespread use of NMF as a clustering method lie in its correctness and efficiency: First, we need to know why and when NMF could detect the true clusters and guarantee to deliver good clustering quality; second, existing algorithms for computing NMF are expensive and often take longer time than other clustering methods. We show that the original NMF can be improved from both aspects in the context of clustering. Our new NMF-based clustering methods can achieve better clustering quality and run orders of magnitude faster than the original NMF and other clustering methods.

Like other clustering methods, NMF places an implicit assumption on the cluster structure. Thus, the success of NMF as a clustering method depends on whether the representation of data in a vector space satisfies that assumption. Our approach to extending the original NMF to a general clustering method is to switch from the vector space representation of data points to a graph representation. The new formulation, called Symmetric NMF, takes a pairwise similarity matrix as an input and can be viewed as a graph clustering method. We evaluate this method on document clustering and image segmentation problems and find that it achieves better clustering accuracy. In addition, for the original NMF, it is difficult but important to choose the right number of clusters. We show that the widely-used consensus NMF in genomic analysis for choosing the number of clusters have critical flaws and can produce misleading results. We propose a variation of the prediction strength measure arising from statistical inference to evaluate the stability of clusters and select the right number of clusters. Our measure shows promising performances in artificial simulation experiments.

Large-scale applications bring substantial efficiency challenges to existing algorithms for computing NMF. An important example is topic modeling where users want to uncover the major themes in a large text collection. Our strategy of accelerating NMF-based clustering is to design algorithms that better suit the computer architecture as well as exploit the computing power of parallel platforms such as the graphic processing units (GPUs). A key observation is that applying rank-2 NMF that partitions a data set into two clusters in a recursive manner is much faster than applying the original NMF to obtain a flat clustering. We take advantage of a special property of rank-2 NMF and design an algorithm that runs faster than existing algorithms due to continuous memory access. Combined with a criterion to stop the recursion, our hierarchical clustering algorithm runs significantly faster and achieves even better clustering quality than existing methods. Another bottleneck of NMF algorithms, which is also a common bottleneck in many other machine learning applications, is to multiply a large sparse data matrix with a tall-and-skinny dense matrix. We use the GPUs to accelerate this routine for sparse matrices with an irregular sparsity structure. Overall, our algorithm shows significant improvement over popular topic modeling methods such as latent Dirichlet allocation, and runs more than 100 times faster on data sets with millions of documents.

# CHAPTER I

# INTRODUCTION

This dissertation shows that *nonnegative matrix factorization* (NMF), a dimension reduction method proposed two decades ago [87, 66], can be extended to a general and efficient clustering method. Clustering is one of the fundamental tasks in machine learning [32]. It is useful for unsupervised knowledge discovery in a variety of applications where human label information is scarce or unavailable. For example, when people read articles, they can easily place the articles into several groups such as science, art, and sports based on the text contents. Similarly, in text mining, we are interested in automatically organizing a large text collection into several clusters where each cluster forms a semantically coherent group. In genomic analysis and cancer study, we are interested in finding common patterns in the patients' gene expression profiles that correspond to cancer subtypes and offer personalized treatment. However, clustering is a difficult, if not impossible, problem. Many clustering methods have been proposed but each of them has tradeoffs in terms of clustering quality and efficiency. The new NMF-based clustering methods that will be discussed in this dissertation can be applied to a wide range of data sets including text, image, and genomic data, achieve better clustering quality, and run orders of magnitude faster than other existing NMF algorithms and other clustering methods.

## 1.1 Nonnegative Matrix Factorization

In nonnegative matrix factorization, given a nonnegative matrix $X \in \mathbb{R}_+^{m \times n}$ and $k \leq \min(m, n)$, $X$ is approximated by a product of two nonnegative matrices $W \in \mathbb{R}_+^{m \times k}$ and $H \in \mathbb{R}_+^{k \times n}$:

$$X \approx WH \tag{1}$$

where $\mathbb{R}_+$ denotes the set of nonnegative real numbers.

In the above formulation, the matrix $X$ is a given data matrix, where rows correspond to features and the columns of $X = [\mathbf{x_1}, \cdots, \mathbf{x_n}]$ represent $n$ nonnegative data points in the $m$-dimensional space. Many types of data have such representation as high-dimensional vectors. For example, a document in the bag-of-words model is represented as a distribution of all the words in the vocabulary; a raw image (without feature extraction) is represented as a vectorized array of pixels. In high-dimensional data analysis, rather than training or making prediction relying on these high-dimensional data directly, it is often desirable to discover a small set of latent factors using a dimension reduction method. In fact, high-dimensional data such as documents and images are usually embedded in a space with much lower dimensions [23].

Nonnegative data frequently occur in data analysis, such as texts [110, 88, 90], images [66, 17], audio signal [21], and gene expression profiles [16, 35, 52]. These types of data can all be represented as a nonnegative data matrix, and NMF has become an important technique for reducing the dimensionality for such data sets. The columns of $W$ form a basis of a latent space and are called *basis vectors*. The matrix $H$ contains coefficients that reconstruct the input matrix by linear combinations of the basis vectors. The $i$-th column of $H$ contains $k$ nonnegative linear coefficients that represent $\mathbf{x_i}$ in the latent subspace spanned by the columns of $W$. In other words, the second low-rank matrix explains the original data points in the latent space. Typically we have $k << \min(m, n)$, i.e. the original data points in the $m$-dimensional space are approximated in a much lower-dimensional space of dimension $k$.

Nonnegativity on the matrices $W$ and $H$ is the key constraint that distinguishes NMF from other low-rank matrix approximation methods. Because the latent factors found by NMF preserve the nonnegativity of the original data points, we can interpret their meanings in order to facilitate human understanding of a data set.

NMF was first proposed by Paatero and Tapper [87], and became popular after Lee and Seung [66] published their work in Nature in 1999. Lee and Seung applied this technique to a collection of human face images, and discovered that NMF extracted facial organs (eyes, noses, lips, etc.) as a set of basic building blocks for these images. This result was in contrast to previous dimension reduction methods such as singular value decomposition (SVD), which did not impose nonnegativity constraints and generated latent factors not easily interpretable by human beings. They called previous methods "holistic" approaches for dimension reduction, and correspondingly referred to NMF as a "parts-based" approach: Each original face image can be approximately represented by additively combining several "parts".

There has been a blossom of papers extending and improving the original NMF in the past two decades, and NMF has been successfully applied to many areas such as bioinformatics [16, 35, 52], blind source separation [21, 100], and recommender systems [117]. In particular, NMF has shown excellent performances as a clustering method. For the time being, let us assume that the given parameter $k$ is the actual number of clusters in a data set; we will consider the case where $k$ is unknown *a priori* in later chapters. Because of the nonnegativity constraints in NMF, one can use the basis vectors directly as cluster representatives, and the coefficients as soft clustering memberships. More precisely, the $i$-th column of $H$ contains fractional assignment values of $\mathbf{x_i}$ corresponding to the $k$ clusters. To obtain a hard clustering result for $\mathbf{x_i}$, we may choose the index that corresponds to the largest element in the $i$-th column of $H$. This clustering scheme has been shown to achieve promising clustering quality in texts [110], images [17], and genomic data [16, 52]. For example, text data can be represented as a term-document matrix where rows correspond to words, columns correspond to documents, and each entry is the raw or weighted frequency of a word in a document. In this case, we can interpret each basis vector as a topic, whose elements are importance values for all the words in a vocabulary. Each document is modeled as a $k$-vector of topic proportions over the $k$ topics, and these topic proportions can be used to derive clustering assignments.

## 1.2 The Correctness of NMF for Clustering

Although NMF has already had many success stories in clustering, one challenge in the widespread use of NMF as a clustering method lie in its correctness. First, we need to know why and when NMF could detect the *true* clusters and guarantee to deliver good clustering quality. From both theoretical and practical standpoints, it is important to know the advantages and limitation of NMF as a clustering method. While dimension reduction and clustering are closely related, they have different goals and different objective functions to optimize. The goal of NMF is to approximate the original data points in a latent subspace, while the goal of clustering is to partition the data points into several clusters so that within-cluster variation is small and between-cluster variation is large. In order to use NMF as a clustering method in the right circumstances, we need to know first when the latent subspace corresponds well to the actual cluster structures.

The above issue, namely the limited understanding of NMF as a clustering method, is partly attributed to the ill-defined nature of clustering. Clustering is often quoted as a technique that discovers "natural grouping" of a set of data points. The word "natural" implies that the true clusters are determined by the discretion of human beings, sometimes visual inspection, and the evaluation of clustering results is subjective [31]. Kleinberg [58] defined three axioms as desired properties for any reasonable clustering method, and showed that these axioms were in themselves contradictory, i.e. no clustering method could satisfy all of them.

From a pessimistic view, Kleinberg's result may suggest that it is worthless to study a clustering method. Talking about the correctness of a clustering method is tricky because there is no *correct* clustering method in its technical sense. However, clustering methods have proved to be very useful for exploratory data analysis in practice. From an optimistic view, what we need to study is the conditions in which a clustering method can perform well and discover the true clusters. Each clustering method places an implicit assumption on the distribution of the data points and the cluster structures. Thus, the success of a clustering method depends on whether the representation of data satisfies that assumption. The same applies to NMF. We investigate the assumption that NMF places on the vector space representation of data points, and extend the original NMF to a general clustering method.

## *1.3 Efficiency of NMF Algorithms for Clustering*

Another issue that may prevent NMF from widespread use in large-scale applications is its computational burden. A popular way to define NMF is to use the Frobenius norm to measure the difference between $X$ and $WH$ [53]:

$$\min_{W,H\geq 0} \|X - WH\|_F^2 \tag{2}$$

where $\|\cdot\|_F$ denotes the Frobenius norm and "$\geq 0$" indicates entrywise nonnegativity. Algorithms for NMF solve (2) as a constrained optimization problem.

A wide range of numerical optimization algorithms have been proposed for minimizing the formulation of NMF (2). Since (2) is nonconvex, in general we cannot expect an algorithm to reach the global minimum; a reasonable convergence property is to reach a stationary point solution [12], which is a necessary condition to be a local or global minimum. Lee and Seung's original algorithm, called multiplicative update rules [66], has been a very popular choice (abbreviated as "update rule" in the following text). This algorithm consists of basic matrix computations only, and thus is very simple to implement. Though it was shown to always reduce the objective function value as the iteration proceeds, its solution is not guaranteed to be a stationary point [37], which is a drawback concerning the quality of the solution. More principled algorithms can be explained using the *block coordinate descent* framework [71, 53], and optimization theory guarantees the stationarity of solutions. In this framework, NMF is reduced to two or more convex optimization problems. Algorithms differ in the respects of how to partition the unknowns into blocks, which correspond to solutions to convex problems, and how to solve these convex problems. Existing methods include projected gradient descent [71], projected quasi-Newton [51], active set [53], block pivoting [56], hierarchical alternating least squares [21], etc. Numerical experiments have shown that NMF algorithms following the block coordinate descent framework are more efficient and produce better solutions than update rule algorithms in terms of the objective function value [71, 53, 57]. For a comprehensive review, see [55].

Despite the effort in developing more efficient algorithms for computing NMF, the computational complexity of these algorithms is still larger than that of classical clustering methods (e.g. K-means, spectral clustering). Applying NMF to data sets with very large $m$ and/or $n$, such as clustering the RCV1 data set [68] with more than 800,000 documents, is still very expensive and costs several hours at the minimum. Also, when $m$ and $n$ are fixed, the computational complexity of most algorithms in the block coordinate descent framework increases superlinearly as $k$, the number of clusters a user requests, increases. Thus, we can witness a demanding need for faster algorithms for NMF in the specific context of clustering. We may increase the efficiency by completely changing the existing framework for "flat" NMF-based clustering.

## *1.4 Contributions, Scope, and Outline*

In this dissertation, we propose several new approaches to improve the quality and efficiency of NMF in the context of clustering. Our contributions include:

1. We show that the original NMF, when used as a clustering method, assumes that different clusters can be represented by linearly independent vectors in a vector space; therefore the original NMF is not a general clustering method

that can be applied everywhere regardless of the distribution of data points and the cluster structures. We extend the original NMF to a general clustering method by switching from the vector space representation of data points to a graph representation. The new formulation, called *Symmetric NMF*, takes a pairwise similarity matrix as an input instead of the original data matrix. Symmetric NMF can be viewed as a graph clustering method and is able to capture nonlinear cluster strutures. Thus, Symmetric NMF can be applied to a wider range of data sets compared to the original NMF, including those that cannot be represented in a finite-dimensional vector space. We evaluate Symmetric NMF on document clustering and image segmentation problems and find that it achieves better clustering accuracy than the original NMF and spectral clustering.

2. For the original NMF, it is difficult but important to choose the right number of clusters. We investigate consensus NMF [16], a widely-used method in genomic analysis that measures the stability of clusters generated under different $k$'s for choosing the number of clusters. We discover that this method has critical flaws and can produce misleading results that suggest cluster structures when they do not exist. We argue that the geometric structure of the low-dimensional representation in a single NMF run, rather than the consensus result of many NMF runs, is important for determining the presence of well-separated clusters. We propose a new framework for cancer subtype discovery and model selection. The new framework is based on a variation of the prediction strength measure arising from statistical inference to evaluate the stability of clusters and select the right number of clusters. Our measure shows promising performances in artificial simulation experiments. The combined methodology has theoretical implications in genomic studies, and will potentially drive more accurate discovery of cancer subtypes.

3. We accelerate NMF-based clustering by designing algorithms that better suit the computer architecture. A key observation is that the efficiency of NMF-based clustering can be tremendously improved by recursively partitioning a data set into two clusters using rank-2 NMF, that is, NMF with $k = 2$. In this case, the overall computational complexity is linear instead of superlinear with respect to the number of clusters in the final clustering result. We focus on a particular type of algorithms, namely *active-set-type* algorithms. We take advantage of a special property of rank-2 NMF solved by active-set-type algorithms and design an algorithm that runs faster than existing algorithms due to continuous memory access. This approach, when used for hierarchical document clustering, generates a tree structure which provides a topic hierarchy in contrast to a flat partitioning. Combined with a criterion to stop the recursion, our hierarchical clustering algorithm runs significantly faster than the original NMF with comparable clustering quality. The leaf-level clusters can be transformed back to a flat clustering result, which turns out to have even better clustering quality. Thus, our algorithm shows significant improvement over popular topic modeling methods such as latent Dirichlet allocation [15].

4. Another bottleneck of NMF algorithms, which is also a common bottleneck in many other machine learning applications, is to multiply a large sparse data matrix with a tall-and-skinny dense matrix (SpMM). Existing numerical libraries that implement SpMM are often tuned towards other applications such as structural mechanics, and thus cannot exploit the full computing capability for machine learning applications. We exploit the computing power of parallel platforms such as the graphic processing units (GPUs) to acclerate this routine. We discuss the performance of SpMM on GPUs and propose a cache blocking strategy that can take advantage of memory locality and increase memory throughput. We develop an out-of-core SpMM routine on GPUs for sparse

matrices with an arbitrary sparsity structure. We optimize its performance specifically for multiplying a large sparse matrix with two dense columns, and apply it to our hierarchical clustering algorithm for large-scale topic modeling. Overall, our algorithm runs more than 100 times faster than the original NMF and latent Dirichlet allocation on data sets with millions of documents.

The primary aim of this dissertation is to show that the original NMF is not sufficient for clustering, and the extensions and new approaches that will be presented in later chapters are necessary and important to establish NMF as a clustering method, in terms of its correctness and efficiency. We focus ourselves on the context of large-scale clustering. When developing the algorithms for the new formulations, we focus on shared memory computing platforms, possibly with multiple cores and accelerators such as the GPUs. We believe that algorithms on shared memory platforms are a required component in any distributed algorithm and thus their efficiency is also very important. Development of efficient distributed NMF algorithms for clustering is one of our future plans and is not covered in this dissertation.

The rest of the dissertation is organized as follows. We first briefly review several existing clustering algorithms in Chapter 2. In Chapter 3, we present Symmetric NMF as a general graph clustering method. In Chapter 4, we introduce our method for choosing the number of clusters and build a new NMF-based framework for cancer subtype discovery. In Chapter 5, we design a hierarchical scheme for clustering that completely changes the existing framework used by NMF-based clustering methods and runs significantly faster. Topic modeling is an important use case of NMF where the major themes in a large text collection need to be uncovered. In Chapter 6, we further accelerate the techniques proposed in the previous chapter by developing a GPU routine for sparse matrix multiplication and culminate with a highly efficient topic modeling method.

# CHAPTER II

# REVIEW OF CLUSTERING ALGORITHMS

## *2.1 K-means*

K-means is perhaps the most widely-used clustering algorithm by far [89, 86]. Given $n$ data points $\mathbf{x}_1, \cdots, \mathbf{x}_n$, a distance function $d(\mathbf{x}_i, \mathbf{x}_j)$ between all pairs of data points, and a number of clusters $k$, the goal of K-means is to find a non-overlapping partitioning $C_1, \cdots, C_k$ of all the data points that minimizes the sum of within-cluster variation of all the partitionings:

$$J = \sum_{j=1}^{k} \frac{1}{2|C_j|} \sum_{i,i' \in C_j} d(\mathbf{x}_i, \mathbf{x}_{i'}), \tag{3}$$

where $|C_j|$ is the cardinality of $C_j$. The squared Euclidean distance is the most frequently used distance function, and K-means clustering that uses Euclidean distances is called *Euclidean K-means.* The sum of within-cluster variation in Euclidean K-means can be written in terms of $k$ "centroids":

$$J = \sum_{j=1}^{k} \frac{1}{2|C_j|} \sum_{i,i' \in C_j} \|\mathbf{x}_i - \mathbf{x}_{i'}\|_2^2 = \sum_{j=1}^{k} \frac{1}{2|C_j|} \sum_{i \in C_j} \|\mathbf{x}_i - \mathbf{c}_j\|_2^2 \tag{4}$$

where

$$\mathbf{c}_j = \frac{1}{|C_j|} \sum_{i'=1}^{|C_j|} \mathbf{x}_{i'} \tag{5}$$

is the centroid of all the data points in $C_j$. (4) is referred to as the *sum of squared error.*

Euclidean K-means is often solved by a heuristic EM-style algorithm, called the *Lloyd's algorithm* [73]. The algorithm can only reach a local minimum of $J$ and cannot be used to obtain the global minimum in general. In the basic version, it starts with a random initialization of centroids, and then iterate the following two steps until convergence:

1. Form a new partitioning $C_1, \cdots, C_k$ by assigning each data point $\mathbf{x}_i$ to the centroid closest to $\mathbf{x}_i$, that is, $\arg\min_j \|\mathbf{x}_i - \mathbf{c}_j\|_2^2$;

2. Compute a new set of centroids $\mathbf{c}_1, \cdots, \mathbf{c}_k$.

This procedure is guaranteed to converge because $J$ is nonincreasing throught the iterations and lower bounded by zero.

The most expensive step of the above algorithm comes from the computation of the Euclidean distances of each pair $(\mathbf{x}_i, \mathbf{c}_j)$ to determine the closest centroid for each data point, which costs $O(mnk)$ where $m$ is the dimension of the data points. In a naïve implementation such as a for-loop, this step can be prohibitively slow and prevent the application of K-means to large data sets. However, the Euclidean distance between two data points can be transformed into another form [83]:

$$\|\mathbf{x}_i - \mathbf{c}_j\|_2^2 = \|\mathbf{x}_i\|_2^2 - 2\mathbf{x}_i^T\mathbf{c}_j + \|\mathbf{c}_j\|_2^2 \tag{6}$$

The cross-term $\mathbf{x}_i^T\mathbf{c}_j$ for all the $(i, j)$ pairs can be written as a matrix form $X^T C$ and computed as a matrix product. The terms $\|\mathbf{x}_i\|_2^2$ and $\|\mathbf{c}_j\|_2^2$ need to be computed only once for each $i$ and each $j$. This way of implementing K-means is much faster because matrix-matrix multiplication is BLAS3 computation and has efficient of the CPU cache. Note that though rewriting the Euclidean distance as (6) is mathematically equivalent, we found that the numerical values may not remain the same, which may lead to different clustering results.

The procedure described above is also called the *batch-update phase* of K-means, in which the data points are re-assigned to their closest centroids all at once in each iteration. Some implementations such as the Matlab `kmeans` employ an additional *online-update phase* that is much more time-consuming [32]. In each iteration of the online-update phase, a single data point is moved from one cluster to another if such a move reduces the sum of squared error $J$, and this procedure is done for every data point in a cyclic manner until the objective function would be increased by moving any single data point from one cluster to another.

## 2.2 Baseline Evaluation of NMF for Clustering

We have introduced the application of NMF to clustering and its interpretation in Chapter 1. Now we present some baseline experimental results that support NMF as a clustering method. We compare the clustering quality between K-means and NMF; zooming into the details of NMF algorithms, we compare the multiplicative updating (MU) algorithm [66] and an alternating nonnegative least squares (ANLS) algorithm [56, 57] in terms of their clustering quality and convergence behavior as well as sparseness in the solution.

### 2.2.1 Data Sets and Algorithms

We used text data sets in our experiments. All these corpora have ground-truth labels for evaluating clustering quality.

1. **TDT2** contains 10,212 news articles from various sources (e.g., NYT, CNN, and VOA) in 1998.

2. **Reuters**[1] contains 21,578 news articles from the Reuters newswire in 1987.

3. **20 Newsgroups**[2] (20News) contains 19,997 posts from 20 Usenet newsgroups. Unlike previous indexing of these posts, we observed that many posts have duplicated paragraphs due to cross-referencing. We discarded cited paragraphs and signatures in a post by identifying lines starting with ">" or "--". The resulting data set is less tightly-clustered and much more difficult to apply clustering or classification methods.

[1] `http://www.daviddlewis.com/resources/testcollections/reuters21578/` (retrieved in June 2014)

[2] `http://qwone.com/~jason/20Newsgroups/` (retrieved in June 2014)

Table 1: Data sets used in our experiments.

| **Data set** | **# Terms** | **# Documents** | **# Ground-truth clusters** |
|---|---|---|---|
| TDT2 | 26,618 | 8,741 | 20 |
| Reuters | 12,998 | 8,095 | 20 |
| 20 Newsgroups | 36,568 | 18,221 | 20 |
| RCV1 | 20,338 | 15,168 | 40 |
| NIPS14-16 | 17,583 | 420 | 9 |

4. From the more recent Reuters news collection **RCV1**[3] [68] that contains over 800,000 articles in 1996-1997, we selected a subset of 23,149 articles. Labels are assigned according to a topic hierarchy, and we only considered leaf topics as valid labels.

5. The research paper collection **NIPS14-16**[4] contains NIPS papers published in 2001-2003 [36], which are associated with labels indicating the technical area (algorithms, learning theory, vision science, etc).

For all these data sets, documents with multiple labels are discarded in our experiments. In addition, the ground-truth clusters representing different topics are highly unbalanced in their sizes for TDT2, Reuters, RCV1, and NIPS14-16. We selected the largest 20, 20, 40, and 9 ground-truth clusters from these data sets, respectively. We constructed term-document matrices using tf-idf features [77], where each row corresponds to a term and each column to a document. We removed any term that appears less than three times and any document that contains less than five words. Table 1 summarizes the statistics of the five data sets after pre-processing. For each data set, we set the number of clusters to be the same as the number of ground-truth clusters.

We further process each term-document matrix $X$ in two steps. First, we normalize each column of $X$ to have a unit $L_2$-norm, i.e., $\|\mathbf{x}_i\|_2 = 1$. Conceptually, this

[3]`http://jmlr.org/papers/volume5/lewis04a/lyrl2004_rcv1v2_README.htm` (retrieved in June 2014)

[4]`http://chechiklab.biu.ac.il/~gal/data.html` (retrieved in June 2014)

makes all the documents have equal lengths. Next, following [110], we compute the normalized-cut weighted version of $X$:

$$D = \mathrm{diag}(X^T X \mathbf{1}_n), \quad X \leftarrow X D^{-1/2}, \tag{7}$$

where $\mathbf{1}_n \in \mathbb{R}^{n \times 1}$ is the column vector whose elements are all 1's, and $D \in \mathbb{R}_+^{n \times n}$ is a diagonal matrix. This column weighting scheme was reported to enhance the clustering quality of both K-means and NMF [110].

For K-means clustering, we used the standard K-means with Euclidean distances. We used both the batch-update and online-update phases and rewrote the Matlab `kmeans` function using BLAS3 operations and boosted its efficiency substantially.[5] For the ANLS algorithm for NMF, we used the block principal pivoting algorithm[6] [56, 57].

### 2.2.2 Clustering Quality

We used two measures to evaluate the clustering quality against the ground-truth clusters. Note that we use *classes* and *clusters* to denote the ground-truth knowledge and the labels given by a clustering algorithm, respectively.

*Clustering accuracy* is the percentage of correctly clustered items given by the maximum bipartite matching (see more details in [110]). This matching associates each cluster with a ground-truth cluster in an optimal way and can be found by the Kuhn-Munkres algorithm [60].

*Normalized mutual information* (NMI) is an information-theoretic measure of the similarity between two flat partitionings [77], which, in our case, are the ground-truth clusters and the generated clusters. It is particularly useful when the number of generated clusters is different from that of ground-truth clusters or when the ground-truth clusters have highly unbalanced sizes or a hierarchical labeling scheme. It is

[5] `http://www.cc.gatech.edu/~dkuang3/software/kmeans3.html` (retrieved in June 2014)
[6] `https://github.com/kimjingu/nonnegfac-matlab` (retrieved in June 2014)

Table 2: The average clustering accuracy given by the four clustering algorithms on the five text data sets.

| | **K-means** | **NMF/MU** | **NMF/ANLS** | **Sparse NMF/ANLS** |
|---|---|---|---|---|
| TDT2 | 0.6711 | 0.8022 | 0.8505 | 0.8644 |
| Reuters | 0.4111 | 0.3686 | 0.3731 | 0.3917 |
| 20News | 0.1719 | 0.3735 | 0.4150 | 0.3970 |
| RCV1 | 0.3111 | 0.3756 | 0.3797 | 0.3847 |
| NIPS14-16 | 0.4602 | 0.4923 | 0.4918 | 0.4923 |

calculated by:

$$\text{NMI} = \frac{I(C_{\text{ground-truth}}, C_{\text{computed}})}{\left[H(C_{\text{ground-truth}}) + H(C_{\text{computed}})\right]/2} = \frac{\sum_{h,l} n_{h,l} \log \frac{n \cdot n_{h,l}}{n_h n_l}}{\left(\sum_h n_h \log \frac{n_h}{n} + \sum_l n_l \log \frac{n_l}{n}\right)/2}, \quad (8)$$

where $I(\cdot,\cdot)$ denotes mutual information between two partitionings, $H(\cdot)$ denotes the entropy of a partitioning, and $C_{\text{ground-truth}}$ and $C_{\text{computed}}$ denote the partitioning corresponding to the ground-truth clusters and the computed clusters, respectively. $n_h$ is the number of documents in the $h$-th ground-truth cluster, $n_l$ is the number of documents in the $l$-th computed cluster, and $n_{h,l}$ is the number of documents in both the $h$-th ground-truth cluster and the $l$-th computed cluster.

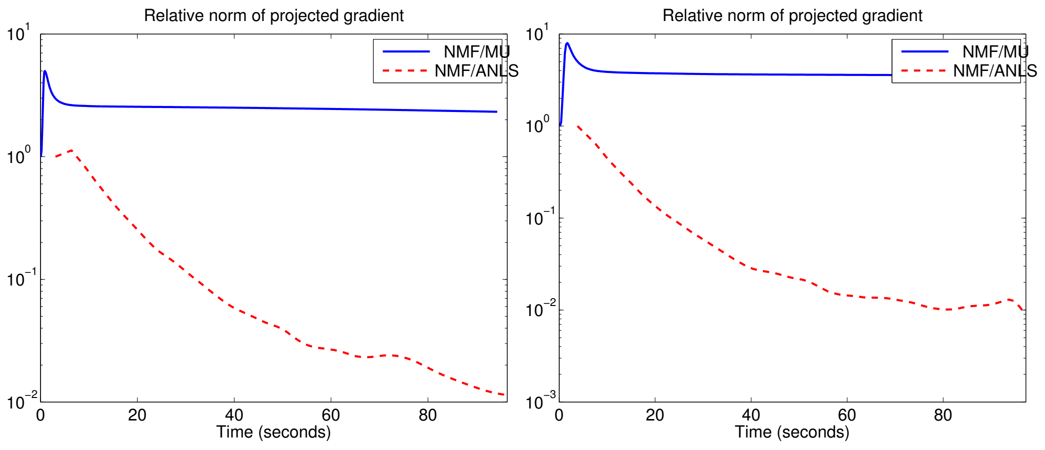
Tables 2 and 3 show the clustering accuracy and NMI results, respectively, averaged over 20 runs with random initializations. All the NMF algorithms have the same initialization of $W$ and $H$ in each run. We can see that all the NMF algorithms consistently outperform K-means except one case (clustering accuracy evaluated on the Reuters data set). Considering the two algorithms for standard NMF, the clustering quality of NMF/ANLS is either similar to or much better than that of NMF/MU. The clustering quality of the sparse NMF is consistently better than that of NMF/ANLS except on the 20 Newsgroups data set and always better than NMF/MU.

### 2.2.3 Convergence Behavior

Now we compare the convergence behaviors of NMF/MU and NMF/ANLS. We employ the *projected gradient* to check stationarity and determine whether to terminate

Table 3: The average normalized mutual information given by the four clustering algorithms on the five text data sets.

| | **K-means** | **NMF/MU** | **NMF/ANLS** | **Sparse NMF/ANLS** |
|---|---|---|---|---|
| TDT2 | 0.7644 | 0.8486 | 0.8696 | 0.8786 |
| Reuters | 0.5103 | 0.5308 | 0.5320 | 0.5497 |
| 20News | 0.2822 | 0.4069 | 0.4304 | 0.4283 |
| RCV1 | 0.4092 | 0.4427 | 0.4435 | 0.4489 |
| NIPS14-16 | 0.4476 | 0.4601 | 0.4652 | 0.4709 |



(a) 20 Newgroups (b) RCV1

Figure 1: The convergence behavior of NMF/MU and NMF/ANLS on the 20 Newsgroups data set ($k = 20$) and RCV1 data set ($k = 40$).

the algorithms [71], which is defined as:

$$(\nabla^P f_W)_{ij} = \begin{cases} \nabla(f_W)_{ij}, & \text{if } (\nabla f_W)_{ij} < 0 \text{ or } W_{ij} > 0; \\ 0, & \text{otherwise,} \end{cases} \tag{9}$$

and the projected gradient norm is defined as:

$$\Delta = \sqrt{\|\nabla^P f_W\|_F^2 + \|\nabla^P f_H\|_F^2}. \tag{10}$$

We denote the projected gradient norm computed from the first iterate of $(W, H)$ as $\Delta(1)$. Fig. 1 shows the relative norm of projected gradient $\Delta/\Delta(1)$ as the algorithms proceed on the 20 Newsgroups and RCV1 data sets. The quantity $\Delta/\Delta(1)$ is not monotonic in general; however, on both data sets, it has a decreasing trend for

NMF/ANLS and eventually reached the given tolerance $\epsilon$, while NMF/MU did not converge to stationary point solutions. This observation is consistent with the result that NMF/ANLS achieved better clustering quality and sparser low-rank matrices.

### 2.2.4 Sparse Factors

With only nonnegativity constraints, the resulting factor matrix $H$ of NMF contains the fractional assignment values corresponding to the $k$ clusters represented by the columns of $W$. Sparsity constraints on $H$ have been shown to facilitate the interpretation of the result of NMF as a hard clustering result and improve the clustering quality [43, 52, 54]. For example, consider two different scenarios of a column of $H \in \mathbb{R}_+^{3\times n}$: $(0.2, 0.3, 0.5)^T$ and $(0, 0.1, 0.9)^T$. Clearly, the latter is a stronger indicator that the corresponding data point belongs to the third cluster.

To incorporate sparsity constraints into the NMF formulation (2), we can adopt the $L_1$-norm regularization on $H$ [52, 54], resulting in *Sparse NMF*:

$$\min_{W,H\geq 0} \|X - WH\|_F^2 + \alpha\|W\|_F^2 + \beta \sum_{i=1}^{n} \|H(:,\, i)\|_1^2, \tag{11}$$

where $H(:,\, i)$ represents the $i$-th column of $H$. The Frobenius-norm regularization term in (11) is used to suppress the entries of $W$ from being too large. Scalar parameters $\alpha$ and $\beta$ are used to control the strength of regularization. The choice of these parameters can be determined by cross validation, for example, by tuning $\alpha, \beta$ until the desired sparseness is reached. Following [52, 53], we set $\alpha$ to the square of the maximum entry in $X$ and $\beta = 0.01$ since these choices have been shown to work well in practice.

We compare the sparseness in the $W$ and $H$ matrices among the solutions of NMF/MU, NMF/ANLS, and the Sparse NMF/ANLS. Table 4 shows the percentage of zero entries for the three NMF versions. Compared to NMF/MU, NMF/ANLS does not only lead to better clustering quality and smaller objective values, but also facilitates sparser solutions in terms of both $W$ and $H$. Recall that each column of $W$

Table 4: The average sparseness of $W$ and $H$ for the three NMF algorithms on the five text data sets. $\%(\cdot)$ indicates the percentage of the matrix entries that satisfy the condition in the parentheses.

| | **NMF/MU** | | **NMF/ANLS** | | **Sparse NMF/ANLS** | |
|---|---|---|---|---|---|---|
| | $\%(w_{ij}=0)$ | $\%(h_{ij}=0)$ | $\%(w_{ij}=0)$ | $\%(h_{ij}=0)$ | $\%(w_{ij}=0)$ | $\%(h_{ij}=0)$ |
| TDT2 | 21.05 | 6.08 | 55.14 | 50.53 | 52.81 | 65.55 |
| Reuters | 40.92 | 12.87 | 68.14 | 59.41 | 66.54 | 72.84 |
| 20News | 46.38 | 15.73 | 71.87 | 56.16 | 71.01 | 75.22 |
| RCV1 | 52.22 | 16.18 | 77.94 | 63.97 | 76.81 | 76.18 |
| NIPS 14-16 | 32.68 | 0.05 | 50.49 | 48.53 | 49.90 | 54.49 |

is interpreted as the term distribution for a topic. With a sparser $W$, the keyword-wise distributions for different topics are more orthogonal, and one can select important terms for each topic more easily. A sparser $H$ can be interpreted as clustering indicators more easily. Table 4 also validates that the sparse NMF generates an even sparser $H$ in the solutions and often better clustering results.

# CHAPTER III

# SYMMETRIC NMF FOR GRAPH CLUSTERING

## *3.1 Limitations of NMF as a Clustering Method*

Although NMF has been widely used in clustering and often reported to have better clustering quality than classical methods such as K-means, it is not a general clustering method that performs well in every circumstance. The reason is that the clustering capability of an algorithm and its limitation can be attributed to its assumption on the cluster structure. For example, K-means assumes that data points in each cluster follow a spherical Gaussian distribution [32]. In the case of NMF, let us consider an exact low-rank factorization where $X = WH$. The columns of $W = [\mathbf{w}_1, \cdots, \mathbf{w}_k]$ form a simplicial cone [30]:

$$\Gamma_W = \{\mathbf{x} | \mathbf{x} = \sum_{j=1}^{k} \alpha_j \mathbf{w}_j, \alpha_j \geq 0\}, \tag{12}$$

and NMF finds a simplicial cone $\Gamma_W$ such that $\mathbf{x}_i \in \Gamma_W, \forall 1 \leq i \leq n$, where each column of $H$ is composed of the nonnegative coefficients $\alpha_1, \cdots, \alpha_k$. Because the cluster label assigned to $\mathbf{x}_i$ is the index of the largest element in the $i$-th column of $H$, a necessary condition for NMF to produce good clustering results is:

> There exists a simplicial cone $\Gamma$ in the positive orthant, such that each of the $k$ vectors that span $\Gamma$ represents a cluster.

If $k \leq \text{rank}(X)$, the columns of $W$ returned by NMF are linearly independent due to $\text{rank}(X) \leq \text{nonnegative-rank}(X)$ [9]. Thus another necessary condition for NMF to produce good clustering results is:

> The $k$ clusters can be represented by linearly independent vectors.

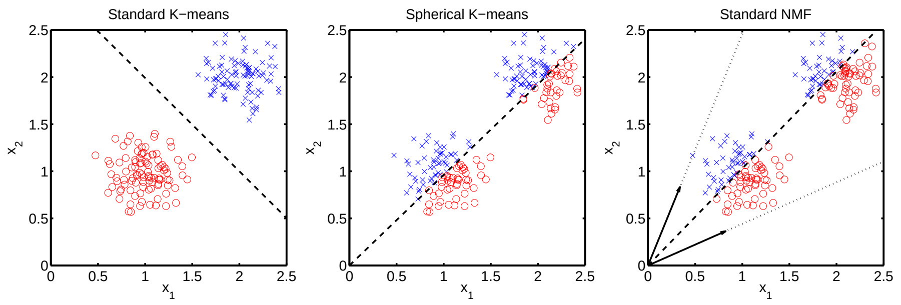


Figure 2: An example with two ground-truth clusters, with different clustering results.

In the case of a low-rank approximation instead of an exact factorization, it was shown that the approximation error $\min_{W\in\mathbb{R}_+^{m\times k},H\in\mathbb{R}_+^{k\times n}}\|X-WH\|_F^2$ decreases with $k$ [55], and thus the columns of $W$ are also linearly independent. In fact, if the columns of $W$ in the result of NMF with lower dimension $k$ were linearly dependent, there always exist matrices $\tilde{W}\in\mathbb{R}_+^{m\times(k-1)}$ and $\tilde{H}\in\mathbb{R}_+^{(k-1)\times n}$ such that $\min_{W\in\mathbb{R}_+^{m\times k},H\in\mathbb{R}_+^{k\times n}}\|X-WH\|_F^2=\left\|X-[\tilde{W}_0\ \ 0][\tilde{H}_0^T\ \ 0]^T\right\|_F^2\geq\min_{W\in\mathbb{R}_+^{m\times(k-1)},H\in\mathbb{R}_+^{(k-1)\times n}}\|X-WH\|_F^2$, which contradicts that $\min_{W\in\mathbb{R}_+^{m\times k},H\in\mathbb{R}_+^{k\times n}}\|X-WH\|_F^2<\min_{W\in\mathbb{R}_+^{m\times(k-1)},H\in\mathbb{R}_+^{(k-1)\times n}}\|X-WH\|_F^2$ [55]. Therefore, we can use NMF to generate good clustering results only when the $k$ clusters can be represented by linearly independent vectors.

Although K-means and NMF have the equivalent form of objective function, $\|X-WH\|_F^2$, each has its best performance on different kinds of data sets. Consider the example in Fig. 2, where the two cluster centers are along the same direction therefore the two centroid vectors are linearly dependent. While NMF still approximates all the data points well in this example, no two linearly independent vectors in a two-dimensional space can represent the two clusters shown in Fig. 2. Since K-means and NMF have different conditions under which each of them does clustering well, they may generate very different clustering results in practice. We are motivated by Fig. 2 to mention that the assumption of spherical K-means is that data points in each cluster follow a von Mises-Fisher distribution [5], which is similar to that of NMF.

NMF, originally a dimension reduction method, is not always a preferred clustering method. The success of NMF as a clustering method depends on the underlying data set, and its most success has been around document clustering [110, 88, 90, 69, 54, 29]. In a document data set, data points are often represented as unit-length vectors [77] and embedded in a linear subspace. For a term-document matrix $X$, a basis vector $\mathbf{w}_j$ is interpreted as the term distribution of a single topic. As long as the representatives of $k$ topics are linearly independent, which are usually the case, NMF can extract the ground-truth clusters well. However, NMF has not been as successful in image clustering. For image data, it was shown that a collection of images tends to form multiple 1-dimensional nonlinear manifolds [99], one manifold for each cluster. This does not satisfy NMF's assumption on cluster structures, and therefore NMF may not identify correct clusters.

In this chapter, we study a more general formulation for clustering based on NMF, called *Symmetric NMF* (SymNMF), where an $n \times n$ nonnegative and symmetric matrix $A$ is given as an input instead of a nonnegative data matrix $X$. The matrix $A$ contains pairwise similarity values of a similarity graph, and is approximated by a lower-rank matrix $BB^T$ instead of the product of two lower-rank matrices $WH$. High-dimensional data such as documents and images are often embedded in a low-dimensional space, and the embedding can be extracted from their graph representation. We will demonstrate that SymNMF can be used for graph embedding and clustering and often performs better than spectral methods in terms of standard evaluation measures for clustering.

The rest of this chapter is organized as follows. In Section 3.2, we review previous work on nonnegative factorization of a symmetric matrix and introduce the novelty of the directions proposed in this chapter. In Section 3.3, we present our new interpretation of SymNMF as a clustering method. In Section 3.4, we show the difference between SymNMF and spectral clustering in terms of their dependence on the spectrum. In Sections 3.5 & 3.6, we propose two algorithms for SymNMF: A

Newton-like algorithm and an alternating nonnegative least squares (ANLS) algorithm, and discuss their efficiency and convergence properties. In Section 3.7.4, we report competitive experiment results on document and image clustering. In Section 3.8, we apply SymNMF to image segmentation and show the unique properties of the obtained segments. In Section 3.9, we discuss future research directions.

## *3.2 Related Work*

In Symmetric NMF (SymNMF), we look for the solution $B \in \mathbb{R}_+^{n \times k}$,

$$\min_{B \geq 0} f(B) = \|A - BB^T\|_F^2, \tag{13}$$

given $A \in \mathbb{R}_+^{n \times n}$ with $A^T = A$ and $k$. The integer $k$ is typically much smaller than $n$. In our graph clustering setting, $A$ is called a *similarity matrix*: The $(i, j)$-th entry of $A$ is the similarity value between the $i$-th and $j$-th node in a similarity graph.

The above formulation has been studied in a number of previous papers. Ding et al. [28] transformed the formulation of NMF (2) to a symmetric approximation $\|A - BB^T\|_F^2$ where $A$ is a positive semi-definite matrix, and showed that it has the same form as the objective function of spectral clustering. Li et al. [69] used this formulation for semi-supervised clustering where the similarity matrix was modified with prior information. Zass and Shashua [115] converted a completely positive matrix [10] to a symmetric doubly stochastic matrix $A$ and used the formulation (13) to find a nonnegative $B$ for probabilistic clustering. They also gave a reason why the nonnegativity constraint on $B$ was more important than the orthogonality constraint in spectral clustering. He et al. [41] approximated a completely positive matrix directly using the formulation (13) with parallel update algorithms. In all of the above work, $A$ was assumed to be a positive semi-definite matrix. For other related work that imposed additional constraints on $B$, see [2, 112, 111].

The SymNMF formulation has also been applied to non-overlapping and overlapping community detection in real networks [105, 75, 84, 119, 118]. For example, Nepusz [84] proposed a formulation similar to (13) with sum-to-one constraints to detect soft community memberships; Zhang [119] proposed a binary factorization model for overlapping communities and discussed the pros and cons of hard/soft assignments to communities. The adjacency matrix $A$ involved in community detection is often an indefinite matrix.

Additionally, Catral et al. [18] studied the symmetry of $WH$ and the equality between $W$ and $H^T$, when $W$ and $H$ are the global optimum for the problem $\min_{W,H \geq 0} \|A - WH\|_F^2$ where $A$ is nonnegative and symmetric. Ho [42] in his thesis related SymNMF to the exact symmetric NMF problem $A = BB^T$. Both of their theories were developed outside the context of graph clustering, and their topics are beyond the scope of this thesis. Ho [42] also proposed a $2n$-block coordinate descent algorithm for (13). Compared to our two-block coordinate descent framework described in Section 3.6, Ho's approach introduced a dense $n \times n$ matrix which destroys the sparsity pattern in $A$ and is not scalable.

Almost all the work mentioned above employed multiplicative update algorithms to optimize their objective functions with nonnegativity constraints. However, this type of algorithms does not have the property that every limit point is a stationary point [37, 70], and accordingly their solutions are not necessarily local minima. In fact, though the papers using multiplicative update algorithms proved that the solutions satisfied the KKT condition, their proof did not include all the components of the KKT condition, for example, the sign of the gradient vector (we refer the readers to [26] as an example). Of the three papers [84, 118, 42] that used gradient descent methods for optimization and *did* reach stationary point solutions, they performed the experiments only on graphs with up to thousands of nodes.

In this chapter, we study the formulation (13) from a different angle:

1. We focus on a more general case where $A$ is a symmetric indefinite matrix representing a general graph. Examples of such an indefinite matrix include a

similarity matrix for high-dimensional data formed by the self-tuning method [116] as well as the pixel similarity matrix in image segmentation [91]. Real networks have additional structures such as the scale-free properties [95], and we will not include them in this work.

2. We focus on hard clustering and will give an intuitive interpretation of SymNMF as a graph clustering method. Hard clustering offers more explicit membership and easier visualization than soft clustering [119]. Unlike [28], we emphasize the difference between SymNMF and spectral clustering instead of their resemblance.

3. We will propose two optimization algorithms that converge to stationary point solutions for SymNMF, namely Newton-like algorithm and ANLS algorithm. We also show that the new ANLS algorithm scales to large data sets.

4. In addition to experiments on document and image clustering, we apply SymNMF to image segmentation using 200 images in the Berkeley Segmentation Data Set [1]. To the best of our knowledge, our work is the first attempt to perform a comprehensive evaluation of nonnegativity-based methods for image segmentation.

Overall, we conduct a comprehensive study of SymNMF in this chapter, covering from foundational justification for SymNMF for clustering, convergent and scalable algorithms, to real-life applications for text and image clustering as well as image segmentation.

## *3.3 Interpretation of SymNMF as a Graph Clustering Method*

Just as the nonnegativity constraint in NMF makes it interpretable as a clustering method, the nonnegativity constraint $B \geq 0$ in (13) also gives a natural interpretation of SymNMF. Now we provide an intuitive explanation of why this formulation is expected to extract cluster structures.

Fig. 3 shows an illustrative example of SymNMF, where we have reorganized the rows and columns of $A$ without loss of generality. If a similarity matrix has a clear cluster structure embedded in it, several diagonal blocks (two diagonal blocks in Fig. 3) that contain large similarity values will appear after the rows and columns of $A$ are permuted so that graph nodes in the same cluster are contiguous to each other in $A$. In order to approximate this similarity matrix with low-rank matrices and simultaneously extract cluster structures, we can approximate each of these diagonal blocks by a rank-one nonnegative and symmetric matrix because each diagonal block indicates one cluster. As shown in Fig. 3, it is straightforward to use an outer product $\mathbf{b}\mathbf{b}^T$ to approximate a diagonal block. Because $\mathbf{b}$ is a nonnegative vector, it serves as a cluster membership indicator: Larger values in $\mathbf{b}$ indicate stronger memberships to the cluster corresponding to the diagonal block. When multiple such outer products are added up together, they approximate the original similarity matrix, and each column of $B$ represents one cluster.

Due to the nonnegativity constraints in SymNMF, only "additive", or "non-subtractive", summation of rank-1 matrices is allowed to approximate both diagonal and off-diagonal blocks. On the contrary, Fig. 4 illustrates the result of low-rank approximation of $A$ without nonnegativity constraints. In this case, when using multiple outer products $\mathbf{b}\mathbf{b}^T$ to approximate $A$, cancellations of positive and negative numbers are allowed. The large diagonal blocks and small off-diagonal blocks could still be well approximated. However, without nonnegativity enforced on $\mathbf{b}$'s, the diagonal blocks need not be approximated separately, and all the elements in a vector $\mathbf{b}$ could be large, thus $\mathbf{b}$ cannot serve as a cluster membership indicator. In this case, the rows of the low-rank matrix $B$ contain both positive and negative numbers and can be used for graph embedding. In order to obtain hard clusters, we need to post-process

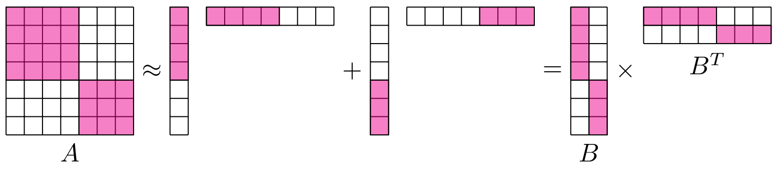


Figure 3: An illustration of SymNMF formulation $\min_{B \geq 0} \|A - BB^T\|_F^2$. Each cell is a matrix entry. Colored region has larger values than white region. Here $n = 7$ and $k = 2$.

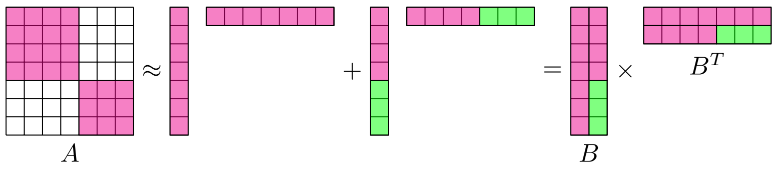


Figure 4: An illustration of $\min \|A - BB^T\|_F^2$ or $\min_{BB^T=I} \|A - BB^T\|_F^2$. Each cell is a matrix entry. Colored region has larger magnitudes than white region. Magenta cells indicate positive entries, green indicating negative. Here $n = 7$ and $k = 2$.

the embedded data points such as applying K-means clustering. This reasoning is analogous to the contrast between NMF and SVD (singular value decomposition) [66].

Compared to NMF, SymNMF is more flexible in terms of choosing similarities between data points. We can choose any similarity measure that describes the cluster structure well. In fact, the formulation of NMF (2) can be related to SymNMF when $A = X^T X$ in (13) [28]. This means that NMF implicitly chooses inner products as the similarity measure, which is not always suitable to distinguish different clusters.

## *3.4 SymNMF and Spectral Clustering*

### 3.4.1 Objective Functions

Spectral clustering represents a large class of graph clustering methods that rely on eigenvector computation [19, 91, 85]. Now we will show that spectral clustering and SymNMF are closely related in terms of the graph clustering objective but fundamentally different in optimizing this objective.

Many graph clustering objectives can be reduced to a trace maximization form

[24, 61]:

$$\max \operatorname{trace}(\tilde{B}^T A \tilde{B}), \tag{14}$$

where $\tilde{B} \in \mathbb{R}^{n \times k}$ (to be distinguished from $B$ in the SymNMF formulation) satisfies $\tilde{B}^T\tilde{B} = I, \tilde{B} \geq 0$, and each row of $\tilde{B}$ contains one positive entry and at most one positive entry due to $\tilde{B}^T\tilde{B} = I$. Clustering assignments can be drawn from $\tilde{B}$ accordingly.

Under the constraints on $\tilde{B}$, we have [28]:

$$\begin{aligned}
& \max \operatorname{trace}(\tilde{B}^T A \tilde{B}) \\
\Leftrightarrow \quad & \min \operatorname{trace}(A^T A) - 2\operatorname{trace}(\tilde{B}^T A \tilde{B}) + \operatorname{trace}(I) \\
\Leftrightarrow \quad & \min \operatorname{trace}[(A - \tilde{B}\tilde{B}^T)^T (A - \tilde{B}\tilde{B}^T)] \\
\Leftrightarrow \quad & \min \|A - \tilde{B}\tilde{B}^T\|_F^2.
\end{aligned}$$

This objective function is the same as (13), except that the constraints on the low-rank matrices $B$ and $\tilde{B}$ are different. The constraint on $\tilde{B}$ makes the graph clustering problem NP-hard [91], therefore a practical method relaxes the constraint to obtain a tractable formulation. In this respect, spectral clustering and SymNMF can be seen as two different ways of relaxation: While spectral clustering retains the constraint $\tilde{B}^T\tilde{B} = I$, SymNMF retains $\tilde{B} \geq 0$ instead. These two choices lead to different algorithms for optimizing the same graph clustering objective (14), which are shown in Table 5.

### 3.4.2 Spectral Clustering and the Spectrum

Normalized cut is a widely-used objective for spectral clustering [91]. Now we describe some scenarios where optimizing this objective may have difficulty in identifying correct clusters while SymNMF could be potentially better.

Although spectral clustering is a well-established framework for graph clustering, its success relies on the properties of the leading eigenvalues and eigenvectors of the

Table 5: Algorithmic steps of spectral clustering and SymNMF clustering.

| | **Spectral clustering** | **SymNMF** |
|---|---|---|
| Objective | $\min_{\hat{B}^T\hat{B}=I} \|A - \hat{B}\hat{B}^T\|_F^2$ | $\min_{B\geq 0} \|A - BB^T\|_F^2$ |
| Step 1 | Obtain the global optimal $\hat{B}_{n\times k}$ by computing $k$ leading eigenvectors of $A$ | Obtain a solution $B$ using an optimization algorithm |
| Step 2 | Scale each row of $\hat{B}$ | (no need to scale rows of $B$) |
| Step 3 | Apply a clustering algorithm to the rows of $\hat{B}$, a $k$-dimensional embedding | The largest entry in each row of $B$ indicates the clustering assignments |

similarity matrix $A$. It was pointed out in [94, 85] that the $k$-dimensional subspace spanned by the leading $k$ eigenvectors of $A$ is stable only when $|\lambda_k(A) - \lambda_{k+1}(A)|$ is sufficiently large, where $\lambda_i(A)$ is the $i$-th largest eigenvalue of $A$. Now we show that spectral clustering could fail when this condition is not satisfied but the cluster structure is perfectly represented in the block-diagonal structure of $A$. Suppose $A$ is composed of $k = 3$ diagonal blocks, corresponding to three clusters:

$$A = \begin{bmatrix} A_1 & 0 & 0 \\ 0 & A_2 & 0 \\ 0 & 0 & A_3 \end{bmatrix}. \tag{15}$$

If we construct $A$ as in the normalized cut, then each of the diagonal blocks $A_1, A_2, A_3$ has a leading eigenvalue 1. We further assume that $\lambda_2(A_i) < 1$ for all $i = 1, 2, 3$ in exact arithmetic. Thus, the three leading eigenvectors of $A$ correspond to the diagonal blocks $A_1, A_2, A_3$ respectively. However, when $\lambda_2(A_1)$ and $\lambda_3(A_1)$ are so close to 1 that it cannot be distinguished from $\lambda_1(A_1)$ in finite precision arithmetic, it is possible that the computed eigenvalues $\tilde{\lambda}_j(A_i)$ satisfy $\tilde{\lambda}_1(A_1) > \tilde{\lambda}_2(A_1) > \tilde{\lambda}_3(A_1) > \max(\tilde{\lambda}_1(A_2), \tilde{\lambda}_1(A_3))$. In this case, three subgroups are identified within the first cluster; the second and the third clusters cannot be identified, as shown in Fig. 5 where all the data points in these two clusters are mapped to $(0, 0, 0)$. Therefore, eigenvectors computed in a finite precision cannot always capture the correct low-dimensional graph embedding.

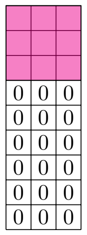


Figure 5: Three leading eigenvectors of the similarity matrix in (15) when $\tilde{\lambda}_3(A_1) > \max(\tilde{\lambda}_1(A_2), \tilde{\lambda}_1(A_3))$. Here we assume that all the block diagonal matrices $A_1, A_2, A_3$ have size $3 \times 3$. Colored region has nonzero values.

Table 6: Leading eigenvalues of the similarity matrix based on Fig. 6 with $\sigma = 0.05$.

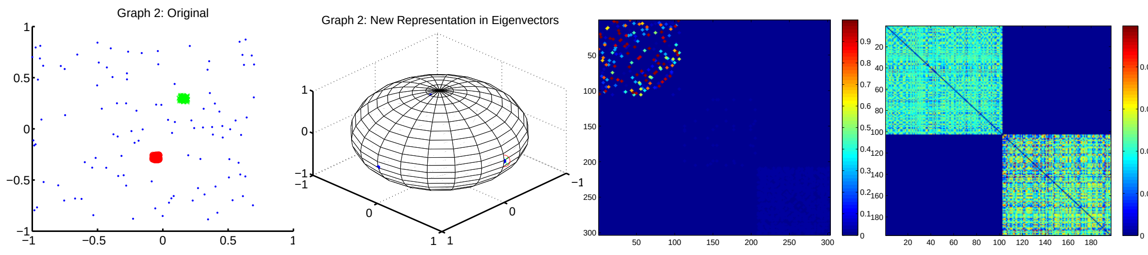
| | |
|---|---|
| 1st | 1.000000000000001 |
| 2nd | 1.000000000000000 |
| 3rd | 1.000000000000000 |
| 4th | 0.999999999998909 |

Now we demonstrate the above scenario using a concrete graph clustering example. Fig. 6 shows (a) the original data points; (b) the embedding generated by spectral clustering; and (c-d) plots of the similarity matrix $A$. Suppose the scattered points form the first cluster, and the two tightly-clustered groups correspond to the second and third clusters. We use the widely-used Gaussian kernel [102] and normalized similarity values [91]:

$$\begin{aligned} e_{ij} &= \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{\sigma^2}\right), \\ A_{ij} &= e_{ij} d_i^{-1/2} d_j^{-1/2}, \end{aligned} \tag{16}$$

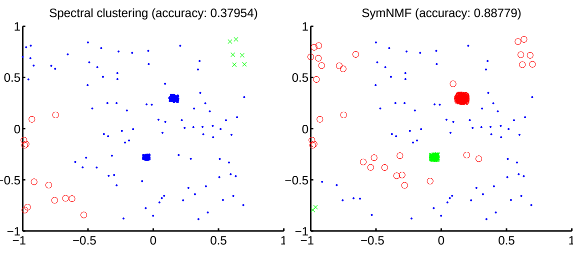
where $\mathbf{x}_i$'s are the two-dimensional data points, $d_i = \sum_{s=1}^{n} e_{is}$ $(1 \le i \le n)$, and $\sigma$ is a parameter set to 0.05 based on the scale of data points. In spectral clustering, the rows of the leading eigenvectors determine a mapping of the original data points, shown in Fig. 6b. In this example, the original data points are mapped to three unique points in a new space. However, the three points in the new space do not correspond to the three clusters in Fig. 6a. In fact, out of the 303 data points in total, 290 data points are mapped to a single point in the new space.

Let us examine the leading eigenvalues, shown in Table 6, where the fourth largest



(a) (b) (c) (d)

Figure 6: A graph clustering example with three clusters (original data from [116]). (a) Data points in the original space. (b) 3-dimensional embedding of the data points as rows of three leading eigenvectors. (c) Block-diagonal structure of $A$. (d) Block-diagonal structure of the submatrix of $A$ corresponding to the two tightly-clustered groups in (a). Note that the data points in both (a) and (b) are marked with ground-truth labels.



(a) (b)

Figure 7: Clustering results for the example in Fig. 6: (a) Spectral clustering. (b) SymNMF.

eigenvalue of $A$ is very close to the third largest eigenvalue. This means that the second largest eigenvalue of a cluster, say $\lambda_2(A_1)$, would be easily identified as one of $\lambda_1(A_1)$, $\lambda_1(A_2)$, and $\lambda_1(A_3)$. The mapping of the original data points shown in Fig. 6b implies that the computed three largest eigenvalues come from the first cluster. This example is a noisier case of the scenario in Fig. 5.

On the contrary, we can see from Figs. 6c and 6d that the block-diagonal structure of $A$ is clear, though the within-cluster similarity values are not on the same scale. Fig. 7 shows the comparison of clustering results of spectral clustering and SymNMF in this case. SymNMF is able to separate the two tightly-clustered groups more accurately.

### 3.4.3 A Condition on SymNMF

We have seen that the solution of SymNMF relies on the block-diagonal structure of $A$, thus it does not suffer from the situations in Section 3.4.2. We will also see in later sections that algorithms for SymNMF do not depend on eigenvector computation. However, we do emphasize a condition on the spectrum of $A$ that SymNMF must satisfy in order to make the formulation (13) valid. This condition is related to the spectrum of $A$, specifically the number of nonnegative eigenvalues of $A$. Note that $A$ is assumed to be symmetric and nonnegative, and is not necessarily positive semi-definite, therefore may have both positive and negative eigenvalues. On the other hand, in the approximation $\|A - BB^T\|_F$, $BB^T$ is always positive semi-definite and has rank at most $k$, therefore $BB^T$ would not be a good approximation if $A$ has fewer than $k$ nonnegative eigenvalues. We assume that $A$ has at least $k$ nonnegative eigenvalues when the given size of $B$ is $n \times k$.

This condition on $A$ could be expensive to check. Here, by a simple argument, we claim that it is practically reasonable to assume that this condition is satisfied given a similarity matrix and an integer $k$, the nubmer of clusters, which is typically small. Again, we use the similarity matrix $A$ in (15) as an example. Suppose we know the actual number of clusters is three, and therefore $B$ has size $n \times 3$. Because $A$ is nonnegative, each of $A_1, A_2, A_3$ has at least one nonnegative eigenvalue according to Perron-Frobenius theorem [9], and $A$ has at least three nonnegative eigenvalues. In a real data set, $A$ may become much noisier with small entries in the off-diagonal blocks of $A$. The eigenvalues are not dramatically changed by a small perturbation of $A$ according to matrix perturbation theory [94], hence $A$ is likely to have at least $k$ nonnegative eigenvalues if its noiseless version does. In practice, the number of positive eigenvalues of $A$ is usually much larger than that of negative eigenvalues, which is verified in our experiments.

**Algorithm 1** Framework of the Newton-like algorithm for SymNMF: $\min_{B \geq 0} f(x) = \|A - BB^T\|_F^2$

```
1: Input: number of data points $n$, number of clusters $k$, $n \times n$ similarity matrix
   $A$, reduction factor $0 < \beta < 1$, acceptance parameter $0 < \sigma < 1$, and tolerance
   parameter $0 < \mu << 1$
2: Initialize $x$, $x^{(0)} \leftarrow x$
3: repeat
4:    Compute scaling matrix $S$
5:    Step size $\alpha = 1$
6:    while true do
7:       $x_{\text{new}} = [x - \alpha S \nabla f(x)]^+$
8:       if $f(x_{\text{new}}) - f(x) \leq \sigma \nabla f(x)^T (x_{\text{new}} - x)$ then
9:          break
10:      end if
11:      $\alpha \leftarrow \beta\alpha$
12:   end while
13:   $x \leftarrow x_{\text{new}}$
14: until $\|\nabla^P f(x)\| \leq \mu \|\nabla^P f(x^{(0)})\|$
15: Output: $x$
```

## *3.5 A Newton-like Algorithm for SymNMF*

In this section, we will present an optimization algorithm to compute SymNMF where $A$ is nonnegative and symmetric. The objective function in (13) is a fourth-order non-convex function with respect to the entries of $B$, and has multiple local minima. For this type of problem, it is difficult to find a global minimum; thus a good convergence property we can expect is that every limit point is a stationary point [12]. We could directly apply standard gradient search algorithms, which lead to stationary point solutions; however, they suffer from either slow convergence or expensive computational cost.

### 3.5.1 Algorithm Framework

First, we introduce several notations for clarity. Let $B = [\mathbf{b}_1, \cdots, \mathbf{b}_k] \in \mathbb{R}_+^{n \times k}$. A vector $x$ of length $nk$ is used to represent the vectorization of $B$ by column, i.e. $x = \text{vec}(B) = [\mathbf{b}_1^T, \cdots, \mathbf{b}_k^T]^T \in \mathbb{R}_+^{nk \times 1}$. For simplicity, functions applied on $x$ have the same notation as functions applied on $B$, i.e. $f(x) \equiv f(B)$. $[\cdot]^+$ denotes the

Table 7: Comparison of PGD and PNewton for solving $\min_{B\geq 0} \|A - BB^T\|_F^2$, $B \in \mathbb{R}_+^{n\times k}$.

| | **Projected gradient descent (PGD)** | **Projected Newton (PNewton)** |
|---|---|---|
| Scaling matrix | $S^{(t)} = I_{nk\times nk}$ | $S^{(t)} = \left(\nabla^2_{\mathcal{E}} f(x^{(t)})\right)^{-1}$ |
| Convergence | Linear (zigzagging) | Quadratic |
| Complexity | $O(n^2k)$ / iteration | $O(n^3k^3)$ / iteration |

projection to the nonnegative orthant, i.e. replacing any negative element of a vector to be 0. Superscripts denote iteration indices, e.g. $x^{(t)} = \text{vec}(B^{(t)})$ is the iterate of $x$ in the $t$-th iteration. For a vector $v$, $v_i$ denotes its $i$-th element. For a matrix $M$, $M_{ij}$ denotes its $(i,j)$-th entry; and $M_{[i][j]}$ denotes its $(i,j)$-th $n \times n$ block, assuming that both the numbers of rows and columns of $M$ are multiples of $n$. $M \succ 0$ refers to positive definiteness of $M$. We define the *projected gradient* $\nabla^P f(x)$ at $x$ as [71]:

$$\left(\nabla^P f(x)\right)_i = \begin{cases} \left(\nabla f(x)\right)_i, & \text{if } x_i > 0; \\ [\left(\nabla f(x)\right)_i]^+, & \text{if } x_i = 0. \end{cases} \tag{17}$$

Algorithm 1 describes a framework of gradient search algorithms applied to SymNMF, based on which we will develop our Newton-like algorithm. This description does not specify iteration indices, but updates $x$ in-place. The framework uses the "scaled" negative gradient direction as search direction. Except the scalar parameters $\beta, \sigma, \mu$ in Algorithm 1, the $nk \times nk$ scaling matrix $S^{(t)}$ is the only unspecified quantity. Table 7 lists two choices of $S^{(t)}$ that lead to different gradient search algorithms: projected gradient descent (PGD) [71] and projected Newton (PNewton) [12].

PGD sets $S^{(t)} = I$ throughout all the iterations. It is known as one of steepest descent methods, and does not scale the gradient using any second-order information. This strategy often suffers from the well-known zigzagging behavior, thus has slow convergence rate [12]. On the other hand, PNewton exploits second-order information provided by the Hessian $\nabla^2 f(x^{(t)})$ as much as possible. PNewton sets $S^{(t)}$ to be the inverse of a reduced Hessian at $x^{(t)}$. The reduced Hessian with respect to index set

$R$ is defined as:

$$(\nabla_R^2 f(x))_{ij} = \begin{cases} \delta_{ij}, & \text{if } i \in R \text{ or } j \in R; \\ (\nabla^2 f(x))_{ij}, & \text{otherwise}, \end{cases} \tag{18}$$

where $\delta_{ij}$ is the Kronecker delta. Both the gradient and the Hessian of $f(x)$ can be computed analytically:

$$\begin{aligned} \nabla f(x) &= \mathrm{vec}(4(BB^T - A)B), \\ (\nabla^2 f(x))_{[i][j]} &= 4\left(\delta_{ij}(BB^T - A) + \mathbf{b}_j\mathbf{b}_i^T + (\mathbf{b}_i^T\mathbf{b}_j)I_{n\times n}\right). \end{aligned}$$

We introduce the definition of an index set $\mathcal{E}$ that helps to prove the convergence of Algorithm 1 [12]:

$$\mathcal{E} = \{i | 0 \le x_i \le \epsilon, (\nabla f(x))_i > 0\}, \tag{19}$$

where $\epsilon$ depends on $x$ and is usually small ($0 < \epsilon < 0.01$) [50]. In PNewton, $S^{(t)}$ is formed based on the reduced Hessian $\nabla_{\mathcal{E}}^2 f(x^{(t)})$ with respect to $\mathcal{E}$. However, because the computation of the scaled gradient $S^{(t)}\nabla f(x^{(t)})$ involves the Cholesky factorization of the reduced Hessian, PNewton has very large computational complexity of $O(n^3k^3)$, which is prohibitive. Therefore, we propose a Newton-like algorithm that exploits second-order information in an inexpensive way.

### 3.5.2 Improving the Scaling Matrix

The choice of the scaling matrix $S^{(t)}$ is essential to an algorithm that can be derived from the framework described in Algorithm 1. We propose two improvements on the choice of $S^{(t)}$, yielding new algorithms for SymNMF. Our focus is to efficiently collect partial second-order information but meanwhile still effectively guide the scaling of the gradient direction. Thus, these improvements seek a tradeoff between convergence rate and computational complexity, with the goal of accelerating SymNMF algorithms as an overall outcome.

Our design of new algorithms must guarantee the convergence. Since the algorithm framework still follows Algorithm 1, we would like to know what property of the scaling matrix $S^{(t)}$ is essential in the proof of the convergence result of PGD and PNewton. This property is described by the following lemma:

**Definition 1.** *A scaling matrix $S$ is diagonal with respect to an index set $R$, if $S_{ij} = 0, \forall i \in R$ and $j \neq i$. [11]*

**Lemma 1.** *Let $S$ be a positive definite matrix which is diagonal with respect to $\mathcal{E}$. If $x \geq 0$ is not a stationary point, there exists $\bar{\alpha} > 0$ such that $f\left([x - \alpha S \nabla f(x)]^+\right) < f(x), \forall 0 < \alpha < \bar{\alpha}$. (modified from [11])*

Lemma 1 states the requirement on $S^{(t)}$, which is satisfied by the choices of $S^{(t)}$ in both PGD and PNewton. It guides our development of new ways to choose $S^{(t)}$.

#### *3.5.2.1 Improvement 1: Fewer Hessian Evaluations*

A common method for reducing computational cost related to $S^{(t)}$ is to periodically update $S^{(t)}$ or evaluate $S^{(t)}$ only at the 1st iteration (chord method) [50]. However, this method cannot be directly used in the framework of Algorithm 1, because $S^{(t)}$ is not necessarily diagonal with respect to $\mathcal{E}^{(t)}$ if $\mathcal{E}^{(t)} \neq \mathcal{E}^{(1)}$, and the requirement for convergence is violated.

Our way to delay the update of $S^{(t)}$ is to evaluate $S^{(t)}$ only when $\mathcal{E}^{(t)}$ changes. More precisely,

$$S^{(t)} = \begin{cases} S^{(t-1)}, \text{ if } \mathcal{E}^{(t)} = \mathcal{E}^{(t-1)}; \\ \left(\nabla^2_{\mathcal{E}} f(x^{(t)})\right)^{-1}, \text{ if } \mathcal{E}^{(t)} \neq \mathcal{E}^{(t-1)} \\ \qquad\qquad\qquad \text{and } \nabla^2_{\mathcal{E}} f(x^{(t)}) \succ 0; \\ I_{nk \times nk}, \text{ otherwise.} \end{cases} \tag{20}$$

Note that because $f(x)$ is non-convex, we have to set $S^{(t)} = I$ when $\nabla^2_{\mathcal{E}} f(x^{(t)})$ is not positive definite, which can be checked during its Cholesky factorization. We expect that this improvement can reduce the number of Hessian evaluations and Cholesky factorizations.

#### *3.5.2.2 Improvement 2: Cheaper Hessian Evaluations*

The second improvement in choosing $S^{(t)}$ is inspired by the recently proposed *coordinate gradient descent* (CGD) method for solving covariance selection [114]. When CGD is directly applied to SymNMF, it updates one column of $B$ in each iteration while the other columns are fixed, and the search direction is typically determined by solving a quadratic programming problem. The CGD method introduces additional overhead when determining the search direction; however, it implies a possibility of using second-order information without evaluating the entire Hessian.

Inspired by the incremental update framework of CGD, we propose to choose $S^{(t)}$ to be a block-diagonal matrix in our batch update framework in Algorithm 1. Specifically,

$$S^{(t)}_{[i][j]} = \begin{cases} 0, \text{ if } i \neq j; \\ \left(\nabla^2_{\mathcal{E}} f(x^{(t)})_{[i][j]}\right)^{-1}, \text{if } i = j \\ \qquad\qquad\qquad \text{and } \nabla^2_{\mathcal{E}} f(x^{(t)})_{[i][j]} \succ 0; \\ I_{n\times n}, \text{ otherwise.} \end{cases} \tag{21}$$

Intuitively speaking, the $i$-th $n \times n$ diagonal block of $S^{(t)}$ corresponds to variables in the $i$-th column of $B$, and $S^{(t)}$ only involves second-order information within each column of $B$. This choice of $S^{(t)}$ has two advantages over the choice in PNewton algorithm: 1. The computational complexity in each iteration is $O(n^3k)$, much lower than the complexity of PNewton if $k$ is not too small; 2. We can exploit partial second-order information even though the $n$ diagonal blocks of $\nabla^2_{\mathcal{E}} f(x^{(t)})$ are not all positive definite, whereas PNewton requires the positive definiteness of all the $n$ diagonal blocks.

Our final strategy for solving SymNMF (13) is to combine Improvement 1 and Improvement 2. Note that the requirement on $S^{(t)}$ described in Lemma 1 is satisfied in both of the improvements, and also in their combination. Thus, convergence is guaranteed in all of these variations.

## *3.6 An ANLS Algorithm for SymNMF*

In this section, we propose another optimization algorithm for SymNMF that converges to stationary points, a necessary condition for local minima. The algorithm is based on an alternative formulation of SymNMF, where it is straightforward to use the two-block coordinate descent framework that has been shown efficient for standard NMF.

### 3.6.1 Two-block Coordinate Descent Framework

We first briefly review the *two-block coordinate descent framework* [71, 53, 55] for standard NMF problems, which has our desired convergence property that every limit point is a stationary point. Separating the $(m+n)k$ unknowns in the NMF formulation (2) into two blocks, we obtain the following subproblems:

1. Fix $H$ and solve $\min_{W \geq 0} \|H^T W^T - X^T\|_F^2$.
2. Fix $W$ and solve $\min_{H \geq 0} \|WH - X\|_F^2$.

Each subproblem is a nonnegative least squares problem with multiple right-hand sides (NNLS for short), and many efficient procedures have been developed to solve NNLS, e.g. active-set method [65, 53], block pivoting [56], PGD [71], etc. The key requirement in this framework is to obtain the optimal solution in each subproblem (see more discussions in [53]). This way, the original NMF formulation (2) has been reduced to an alternating NNLS problem (ANLS for short).

### 3.6.2 A Nonsymmetric Formulation for SymNMF

In SymNMF, it is difficult to separate the $nk$ unknowns in a straightforward way as in NMF, because the two factors $B$ and $B^T$ contain the same set of unknowns. We propose to re-formulate SymNMF in the context of NMF [42]:

$$\min_{C,B\geq 0} g(C,B) = \|A - CB^T\|_F^2 + \alpha\|C - B\|_F^2, \tag{22}$$

where $A$ still represents the $n \times n$ similarity matrix, $C, B$ are two low-rank factors of size $n \times k$, and $\alpha > 0$ is a scalar parameter for the tradeoff between the approximation error and the difference of $C$ and $B$. Here we *force* the separation of unknowns by associating the two factors with two different matrices. If $\alpha$ has a large enough value, the solutions of $C$ and $B$ will be close enough so that the clustering results will not be affected whether $C$ or $B$ are used as the clustering assignment matrix.

If $C$ or $B$ is expected to indicate more distinct cluster structures, sparsity constraints on rows of $B$ can also be incorporated into the nonsymmetric formulation easily, by adding $L_1$ regularization terms [52, 53]:

$$\min_{C,B\geq 0} \tilde{g}(C,B) = \|A - CB^T\|_F^2 + \alpha\|C - B\|_F^2 + \beta\sum_{i=1}^{n}\|c_i\|_1^2 + \beta\sum_{i=1}^{n}\|b_i\|_1^2, \tag{23}$$

where $\alpha, \beta > 0$ are regularization parameters, $c_i, b_i$ are the $i$-th rows of $C, B$ respectively, and $\|\cdot\|_1$ denotes vector 1-norm.

The nonsymmetric formulation can be easily cast into the two-block coordinate descent framework after some restructuring. In particular, we have the following subproblems for (23) (and (22) is a special case where $\beta = 0$):

$$\min_{C\geq 0} \left\| \begin{bmatrix} B \\ \sqrt{\alpha}I_k \\ \sqrt{\beta}\mathbf{1}_k^T \end{bmatrix} C^T - \begin{bmatrix} A \\ \sqrt{\alpha}B^T \\ 0 \end{bmatrix} \right\|_F^2, \tag{24}$$

$$\min_{B\geq 0} \left\| \begin{bmatrix} C \\ \sqrt{\alpha}I_k \\ \sqrt{\beta}\mathbf{1}_k^T \end{bmatrix} B^T - \begin{bmatrix} A \\ \sqrt{\alpha}C^T \\ 0 \end{bmatrix} \right\|_F^2, \tag{25}$$

where $\mathbf{1}_k \in \mathbb{R}^{k\times 1}$ is a column vector whose elements are all 1's, and $I_k$ is the $k \times k$ identity matrix. Note that we have assumed $A = A^T$. Solving subproblems (24) and

**Algorithm 2** Framework of the ANLS algorithm for SymNMF: $\min_{C,B\geq 0} \|A - CB^T\|_F^2 + \alpha\|C - B\|_F^2$

```
1: Input: number of data points $n$, number of clusters $k$, $n \times n$ similarity matrix $A$,
   regularization parameter $\alpha > 0$, and tolerance parameter $0 < \mu << 1$
2: Initialize $B$, $B^{(0)} \leftarrow B$, $C^{(0)} \leftarrow B$
3: repeat
4:     $C \leftarrow B$
5:     Solve an NNLS problem: $B \leftarrow \arg\min_{B\geq 0} \left\| \begin{bmatrix} C \\ \sqrt{\alpha} I_k \end{bmatrix} B^T - \begin{bmatrix} A \\ \sqrt{\alpha} C^T \end{bmatrix} \right\|_F^2$
6: until $\|\nabla^P g(C,B)\|_F \leq \mu \|\nabla^P g(C^{(0)}, B^{(0)})\|_F$
7: Output: $B$
```

(25) in an alternate fashion will lead to a stationary point solution, as long as an optimal solution is returned for every NNLS subproblem encountered. We simplify and summarize this algorithm in Algorithm 2 for the formulation (22), i.e. without sparsity constraints.

### 3.6.3 Implementation

Now we describe an efficient implementation of the ANLS algorithm for SymNMF. Our algorithm reduces to solving the NNLS problem in line 5 of Algorithm 2. Consider a form of NNLS with simplified notation: $\min_{G\geq 0} \|FG^T - X\|_F^2$. In many algorithms for NNLS, the majority of time cost comes from the computation of $F^TF$ and $X^TF$. For example, in the active-set method [53] and block pivoting method [56], we need to form the normal equation:

$$F^TFG^T = F^TX.$$

In PGD [71], we need to compute the gradient:

$$\nabla_G = 2G(F^TF) - 2X^TF.$$

For more details of these algorithms for NNLS, please refer to the papers [71, 53, 56, 57]. Our strategy to solve the NNLS problem in Algorithm 2 is to precompute $F^TF$ and $X^TF$:

$$F^TF = C^TC + \alpha I_k, \quad X^TF = A^TC + \alpha C$$

without forming $X = \begin{bmatrix} A \\ \sqrt{\alpha} C^T \end{bmatrix}$ directly. Though this change sounds trivial, forming $X$ directly is very expensive when $A$ is a large and sparse matrix, especially when $A$ is stored in the "compressed sparse column" format such as in Matlab and the Python `scipy` package. In our experiments, we observed that our strategy had considerable time savings in the iterative Algorithm 2.

For choosing the parameter $\alpha$, we can gradually increase $\alpha$ from 1 to a very large number, for example, by setting $\alpha \leftarrow 1.01\alpha$. We can stop increasing $\alpha$ when $\|C - B\|_F / \|B\|_F$ is negligible (say, $< 10^{-8}$).

Conceptually, both the Newton-like algorithm and the ANLS algorithm work for any nonnegative and symmetric matrix $A$ in SymNMF. In practice, however, a similarity matrix $A$ is often very sparse and the efficiencies of these two algorithms become very different. The Newton-like algorithm does not take into account the structure of SymNMF formulation (13), and a sparse input matrix $A$ cannot contribute to speeding up the algorithm because of the formation of the dense matrix $BB^T$ in intermediate steps. On the contrary, in the ANLS algorithm, many algorithms for the NNLS subproblem [71, 53, 56] can often benefit from the sparsity of similarity matrix $A$ automatically. This benefit comes from sparse-dense matrix multiplication inside these algorithms such as $A \cdot B$ as well as the absence of large dense matrices such as $BB^T$. Therefore, we recommend using the ANLS algorithm for a sparse input matrix $A$.

## *3.7 Experiments on Document and Image Clustering*

In this section, we show the performances of SymNMF on a number of text and image data sets, and compare SymNMF with the standard forms and variations of NMF, spectral clustering, and K-means. Throughout the experiments, we use Matlab 7.9 (R2009b) with an Intel Xeon X5550 quad-core processor and 24GB memory.

### 3.7.1 Data Preparation

We construct a sparse graph for each data set. Using sparse graphs makes large-scale clustering possible in terms of efficiency. We take the following three steps to form the similarity matrix:

1. *Construct a complete graph.* The edge weights between graph nodes are defined according to the type of data set.

   - For text data, all the document vectors are normalized to have unit 2-norm. The edge weight is the cosine similarity between two document vectors:

   $$e_{ij} = \mathbf{x}_i^T \mathbf{x}_j, \ (i \neq j). \tag{26}$$

   - For image data, the self-tuning method [116] is used:

   $$e_{ij} = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{\sigma_i \sigma_j}\right), \ (i \neq j), \tag{27}$$

     where each data point has a *local scale* $\sigma_i$, as opposed to a global scale $\sigma$ in (16). $\sigma_i$ is set to be the Euclidean distance between $\mathbf{x}_i$ and its $\hat{k}$-th neighbor. We use $\hat{k} = 7$ as suggested in [116].

   Note that we enforce self-edge weights $e_{ii} = 0$ $(1 \leq i \leq n)$ in all cases [85].

2. *Sparsify the graph.* We only keep the edges that connect a node to its $k_n$ nearest neighbors. More precisely, let

   $$N(i) = \{j | \mathbf{x}_j \text{ is one of the } k_n \text{ nearest neighbors of } \mathbf{x}_i, j \neq i\}. \tag{28}$$

   Edge weights in the sparse graph are defined as:

   $$\hat{e}_{ij} = \begin{cases} e_{ij}, \ \text{if } i \in N(j) \ \textit{or} \ j \in N(i); \\ 0, \ \text{otherwise.} \end{cases} \tag{29}$$

   We choose $k_n = \lfloor \log_2 n \rfloor + 1$ as suggested in [102].

3. *Form the similarity matrix.* We compute the normalized similarity values as in the normalized cut [85]:

$$A_{ij} = \hat{e}_{ij} d_i^{-1/2} d_j^{-1/2}, \tag{30}$$

where $d_i = \sum_{s=1}^{n} \hat{e}_{is}$ $(1 \leq i \leq n)$.

Note that the similarity matrix $A$ constructed as above is symmetric, nonnegative, and usually indefinite.

### 3.7.2 Data Sets

Document clustering was conducted on the following labeled corpuses: 1. **TDT2**[1] contains 10,212 news articles from various sources (e.g. NYT, CNN, VOA) in 1998. 2. **Reuters-21578**[2] contains 21,578 news articles from the Reuters newswire in 1987. 3. From the newly-released Reuters news collection **RCV1-v2**[3] [68] that contains over 800,000 articles in 1996-1997, we selected the training set containing 23,149 articles. Labels are assigned according to a topic hierarchy, and we only considered leaf topics as valid labels. 4. The research paper collection **NIPS14-16**[4] contains NIPS papers in 2001-2003 [36], which are associated with labels indicating the technical area (algorithms, learning theory, vision science, etc). For all these data sets, documents with multiple labels are discarded in our experiments. In addition, clusters representing different topics are highly unbalanced in size. We selected the largest 20, 20, 40, 9 clusters from each of these data sets respectively. While TDT2 and the two Reuters data sets were well maintained, the NIPS data set was extracted from PS and PDF files, resulting in very noisy texts, which can be seen from the list of terms available online[4]. For example, its vocabulary includes many symbols

[1]`https://catalog.ldc.upenn.edu/LDC2001T57` (retrieved in June 2014)

[2]`http://www.daviddlewis.com/resources/testcollections/reuters21578/` (retrieved in June 2014)

[3]`http://jmlr.org/papers/volume5/lewis04a/lyrl2004_rcv1v2_README.htm` (retrieved in June 2014)

[4]`http://chechiklab.biu.ac.il/~gal/data.html` (retrieved in June 2014)

frequently used in formulas which are not semantically meaningful.

Image clustering was conducted on object and face recognition data sets: 1. **COIL-20**[5] contains gray-scale images of 20 objects, rescaled to $64 \times 64$ size. The viewpoints are equally spaced in the entire $360^o$ range, resulting in 72 images for each object. 2. **ORL**[6] contains 400 face images of 40 persons with different facial expressions and slightly-varing pose. 3. From **Extended YaleB**[7] face data set (with the original YaleB data included) [67], we selected 2,414 frontal face images of 38 persons, with different illumination conditions. 4. From **PIE**[8] face data set [92], we selected 232 frontal face images of 68 persons, with different facial expressions. Compared to other variations in PIE data set such as illumination and lighting conditions, different facial expressions represent more variations in faces and the images are embedded on multiple manifolds [99]; moreover, only $3 \sim 4$ images are available for each person, which makes clustering more challenging. Though ORL and the selected subset of PIE are not of large-scale, they share the same characteristics: High variations within each class, with a handful of images per class. For all the image data sets, the identity information of the objects or faces is used as ground-truth labels. The statistics of the processed document and image data sets are summarized in Table 8.

### 3.7.3 Algorithms for Comparison

We experimented with a large variety of clustering algorithms for a comprehensive comparison. The algorithms in our experiment can be divided into four categories:

1. **K-means variants** (All these K-means variants include a *batch-update* phase and an additional *online-update* phase in each run [32]. We use both phases.)

[5]`http://www.cs.columbia.edu/CAVE/software/softlib/coil-20.php` (retrieved in June 2014)

[6]`http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html` (retrieved in June 2014)

[7]`http://vision.ucsd.edu/~leekc/ExtYaleDatabase/ExtYaleB.html` (retrieved in June 2014)

[8]`http://www.ri.cmu.edu/research_project_detail.html?project_id=418&menu_id=261` (retrieved in June 2014)

Table 8: Data sets used in experiments.

| **Data set** | **Dimension** | **# Data points** | **# Clusters** |
|---|---|---|---|
| TDT2 | 26,618 | 8,741 | 20 |
| Reuters-21578 | 12,998 | 8,095 | 20 |
| RCV1-v2 | 20,338 | 15,168 | 40 |
| NIPS14-16 | 17,583 | 420 | 9 |
| COIL-20 | $64 \times 64$ | 1,440 | 20 |
| ORL | $69 \times 84$ | 400 | 40 |
| Extended YaleB | $56 \times 64$ | 2,414 | 38 |
| PIE-expression | $64 \times 64$ | 232 | 68 |

- *Standard K-means (KM)*: The input matrix is constructed as follows. For text data, each column of the tf-idf matrix $X$ [77] is scaled to have unit 2-norm; in addition, $X$ is transformed into its normalized-cut weighted version $XD^{-1/2}$ [110], where $D$ is defined in Section 3.3 with $e_{ij} = \mathbf{x}_i^T \mathbf{x}_j$. For image data, each column of $X$ is scaled to the $[0, 1]$ interval.

- *Spherical K-means (SKM)*: Unlike standard K-means that uses Euclidean distance as the dissimilarity measure, spherical K-means uses $1-\cos(\mathbf{x}_i, \mathbf{x}_j)$; therefore any scaling of columns of $X$ does not take effect. Spherical K-means was proposed for document clustering, where cosine similarity is often a better measure than Euclidean distance [25]. As mentioned in Section 3.1, we believe that spherical K-means has a closer relationship to NMF than standard K-means.

- *Kernel K-means (KKM)*: Kernel K-means is a graph clustering method based on K-means. We use the weighted kernel K-means algorithm described in [61, 24] that minimizes the normalized cut objective. Because $A$ is generally indefinite, the condition for convergence is violated. We terminate the algorithm as soon as the objective function value stops decreasing.

2. **NMF variants**

- *NMF*: We use the ANLS algorithm with block pivoting method for NMF [56]. The same input matrix as in standard K-means is used. The clustering result is indicated by the largest entry in each column of $H$.
- *GNMF*: Cai et al. [17] proposed Graph-regularized NMF (GNMF) by adding a graph-theoretic penalty term to (2) that takes neighboring relationship into account, so that the resulting method is better at clustering on manifolds. We use the algorithm and the suggested parameters in [17]. The input matrix is constructed in the same way as in standard K-means. However, the neighboring relationship based on the sparse graph is generated using the original data matrix, i.e. without the scaling of each $\mathbf{x}_i$. The clustering result is obtained by treating the columns of $B$ as graph embedding and applying spherical K-means to the embedded points.

3. **Spectral clustering variants**

   - *NJW algorithm (SpNJW)*: This refers to the algorithm proposed in Ng et al. [85]. The rows of the $k$ leading eigenvectors of $A$, where each row is normalized to have unit 2-norm, are used as the graph embedding of data points. Standard K-means is used in the final step to obtain clustering results, which is initialized by randomly choosing $k$ samples as centroids.
   - *YS algorithm (SpYS)*: This refers to the algorithm proposed in Yu and Shi [113]. The clustering results are obtained by finding the optimal orthogonal transformation of $\tilde{B} = D^{-1/2}\hat{B}$ into a partition matrix [113], where columns of $\hat{B}$ are the $k$ leading eigenvectors of $A$. As a result, the additional step of clustering is not needed.

4. **SymNMF**: We observed that the Newton-like algorithm for SymNMF gives better clustering quality on image data (more details in Section 3.7.5). On text data, however, the Newton-like algorithm is not efficient enough due to large

problem sizes, and only the ANLS algorithm is applicable. When reporting the results, we use the general name "SymNMF" to refer to the Newton-like algorithm for image data and the ANLS algorithm for text data.

In the Newton-like algorithm, we set parameters $\beta = 0.1, \sigma = 0.1$ in the context of Algorithm 1. We also empirically observe that choosing $\epsilon$ in (19) to be a fixed value $10^{-16}$ makes the Newton-like algorithm faster while having little influence on the clustering quality. For the ANLS algorithm, we solve the formulation (22), i.e. without sparsity constraints on $C, B$. We empirically observe that it is sufficient to use a fixed parameter $\alpha = 1$ in (22) to obtain a negligible $\|C - B\|_F / \|B\|_F$. Note that the choice of a large enough value of $\alpha$ should be aligned with the scale of the similarity values in $A$. In our experiments, the matrix $A$ contains normalized similarity values (30), thus the maximum possible value in $A$ is 1, and most of the entries of $A$ are smaller than 1. Finally, in both of our algorithms, the tolerance parameter $\mu$ in the stopping criteria is set to $10^{-4}$ and the maximum iteration count is set to 10,000.

For each data set, we run each algorithm 20 times with different random initializations and the known number of clusters $k$ as input. Algorithms in the same category have the same initializations. Although the data sets are labeled, the labels are used only when evaluating the clustering quality, not by the clustering algorithms.

### 3.7.4 Clustering Quality

We use *clustering accuracy*, the percentage of correctly clustered items given by the maximum bipartite matching, to evaluate the clustering quality (see more details in [110]). The average and maximum clustering accuracy are shown in Tables 9 and 10, respectively. We have the following observations:

1. Among all the methods in our experiments, SymNMF performs well in terms of *both* average and maximum clustering accuracy, and achieves the best clustering quality more frequently than other methods.

Table 9: Average clustering accuracy for document and image data sets. For each data set, the highest accuracy and any other accuracy within the range of 0.01 from the highest accuracy are marked bold.

| | **KM** | **SKM** | **KKM** | **NMF** | **GNMF** | **SpNJW** | **SpYS** | **SymNMF** |
|---|---|---|---|---|---|---|---|---|
| TDT2 | 0.6711 | 0.6755 | 0.6837 | 0.8505 | 0.7955 | 0.7499 | **0.9050** | 0.8934 |
| Reuters-21578 | 0.4111 | 0.3332 | 0.3489 | 0.3731 | 0.4460 | 0.3114 | 0.4986 | **0.5094** |
| RCV1-v2 | 0.3111 | **0.3888** | **0.3831** | **0.3797** | 0.3592 | 0.2723 | 0.2743 | 0.2718 |
| NIPS14-16 | 0.4602 | 0.4774 | 0.4908 | 0.4918 | 0.4908 | **0.4987** | **0.5026** | **0.5086** |
| COIL-20 | 0.6184 | 0.5611 | 0.2881 | 0.6312 | 0.6304 | 0.6845 | **0.7899** | 0.7258 |
| ORL | 0.6499 | 0.6500 | 0.6858 | 0.7020 | 0.7282 | 0.7127 | **0.7752** | **0.7798** |
| Extended YaleB | 0.0944 | 0.0841 | 0.1692 | 0.1926 | 0.2109 | 0.1862 | **0.2254** | **0.2307** |
| PIE-expression | 0.7358 | 0.7420 | 0.7575 | 0.7912 | **0.8235** | 0.7966 | 0.7375 | 0.7517 |
| *ALL* | *0.4940* | *0.4890* | *0.4759* | *0.5515* | *0.5606* | *0.5265* | ***0.5886*** | ***0.5839*** |

Table 10: Maximum clustering accuracy for document and image data sets. For each data set, the highest accuracy and any other accuracy within the range of 0.01 from the highest accuracy are marked bold.

| | **KM** | **SKM** | **KKM** | **NMF** | **GNMF** | **SpNJW** | **SpYS** | **SymNMF** |
|---|---|---|---|---|---|---|---|---|
| TDT2 | 0.7878 | 0.7930 | 0.7884 | 0.9037 | 0.8768 | 0.8064 | 0.9145 | **0.9298** |
| Reuters-21578 | 0.5001 | 0.4011 | 0.5313 | 0.4127 | 0.5408 | 0.4033 | 0.4989 | **0.5633** |
| RCV1-v2 | 0.3531 | **0.4160** | **0.4153** | 0.4009 | 0.4001 | 0.2939 | 0.2747 | 0.2902 |
| NIPS14-16 | 0.5357 | 0.5190 | **0.5905** | 0.5357 | 0.5643 | 0.5548 | 0.5095 | 0.5619 |
| COIL-20 | 0.6937 | 0.6278 | 0.3743 | 0.6729 | 0.7111 | 0.7590 | 0.8014 | **0.8153** |
| ORL | 0.7075 | 0.7025 | 0.7675 | 0.7375 | 0.7900 | 0.7700 | 0.7925 | **0.8025** |
| Extended YaleB | 0.1023 | 0.0928 | 0.2133 | 0.2154 | 0.2229 | 0.2440 | 0.2307 | **0.2564** |
| PIE-expression | 0.7974 | 0.8147 | 0.7931 | 0.8233 | **0.8578** | 0.8362 | 0.7888 | 0.7802 |
| *ALL* | *0.5597* | *0.5459* | *0.5592* | *0.5878* | ***0.6205*** | *0.5835* | *0.6014* | ***0.6250*** |

2. SpYS and SymNMF show comparable performances in terms of average clustering accuracy. Note that SpYS was proposed as a more principled way to obtain hard clustering from the $k$ leading eigenvectors of $A$ [113]; however, our result shows that it is not always better than SpNJW that uses K-means to obtain hard clustering.

3. GNMF and SymNMF show comparable performances in terms of maximum clustering accuracy. GNMF in our experiments does not show as dramatic improvement over SpNJW as the results reported in [17] where only maximum clustering accuracy was reported. One possible reason is that in [17], full graphs with cosine similarity are used, whereas we use sparse graphs and different similarity measures for better scalability and clustering quality (Section 3.7.1).

4. The K-means variants give exceedingly high accuracy on the RCV1-v2 data set. We need more study to have a good explanation of their performances, for example, in what cases cosine dissimilarity is a better choice of distance measure than Euclidean distance. Note that RCV1-v2 is the only data set where spherical K-means has the highest accuracy, and also the only data set where NMF performs better than almost all the other low-rank approximation methods (GNMF, SpNJW, SpYS, SymNMF). This consistency corroborated with our observation that spherical K-means has a closer relationship to NMF than standard K-means, and seems to explain why spherical K-means is often used as an initialization strategy for NMF [106].

### 3.7.5 Convergence and Efficiency of SymNMF Algorithms

We mentioned in Section 3.7.3 that the ANLS algorithm for SymNMF handles large data sets more efficiently, and the Newton-like algorithm achieves higher clustering accuracy. Here we discuss this tradeoff between efficiency and quality. The different properties exhibited by the two algorithms can be attributed to their different

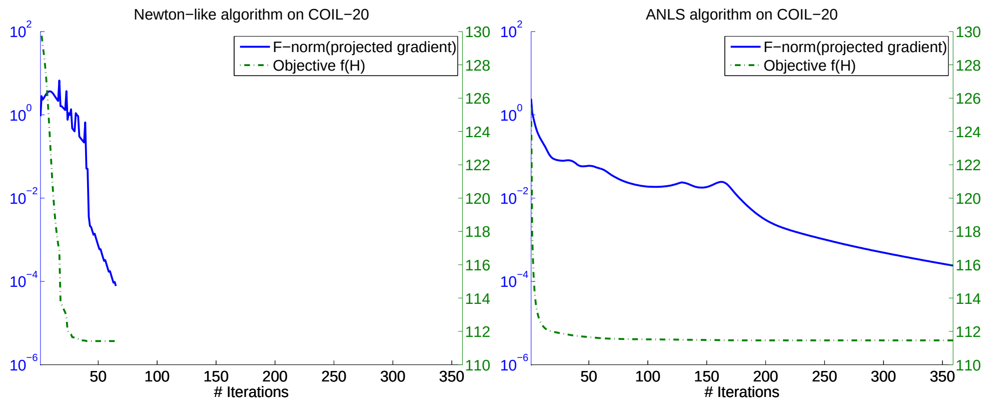


Figure 8: Convergence behaviors of SymNMF algorithms, generated from a single run on COIL-20 data set with the same initialization.

convergence behaviors, though both algorithms converge to stationary point solutions. In Fig. 8, we use COIL-20 data set to study their convergence by plotting the objective function $f(B)$ and the projected gradient $\|\nabla^P f(B)\|_F$ throughout the iterations. As we could expect, $f(B)$ is non-inreasing in both algorithms; on the contrary, $\|\nabla^P f(B)\|_F$ is not guaranteed to drop in every iteration but is used to check stationarity.

The Newton-like algorithm shows a divergent behavior in the initial stage of iterations, because the formulation (13) is nonconvex and the search step degrades to a steepest descent direction. However, when the intermediate iterate becomes close to a local minimum, the Hessian matrix becomes positive definite and the second-order information begins to help guide the search. Thus after this point, the algorithm converges very quickly to an accurate stationary point. In contrast, the ANLS algorithm shows a quick drop in both $\|\nabla^P f(B)\|_F$ and $f(B)$ when the algorithm starts. However, near the final stage, it slowly converges to the appointed stationarity level. Overall, the Newton-like algorithm produces more accurate solutions and better clustering quality; however, it is overall less efficient than the ANLS algorithm due to heavier computational cost per iteration. We compare their clustering quality and timing performance in Table 11, with $\mu = 10^{-4}$ in the stopping criterion in both

Table 11: Clustering accuracy and timing of the Newton-like and ANLS algorithms for SymNMF. Experiments are conducted on image data sets with parameter $\mu = 10^{-4}$ and the best run among 20 initializations.

| | **Newton-like algorithm** | | **ANLS algorithm** | |
|---|---|---|---|---|
| | **Accuracy** | **Time (s)** | **Accuracy** | **Time (s)** |
| COIL-20 | 0.8153 | 18.91 | 0.7833 | 5.63 |
| ORL | 0.8025 | 6.52 | 0.7975 | 2.45 |
| Extended YaleB | 0.2564 | 158.2 | 0.2535 | 18.97 |
| PIE-expression | 0.7802 | 16.49 | 0.7155 | 6.70 |

algorithms.

## *3.8 Image Segmentation Experiments*

In this section, we explore the application of SymNMF to image segmentation. Image segmentation methods have been heavily relying on spectral clustering [76, 79, 34, 22, 1]. We will demonstrate that SymNMF produces segmentation results that are closer to human-marked boundaries compared to spectral clustering. To the best of our knowledge, this is the first systematic evaluation of SymNMF applied to image segmentation.

### 3.8.1 Overview

Image segmentation is an important task in computer vision that organizes an image into a non-overlapping set of closed regions. It can be viewed as a graph clustering problem: The input is a nonnegative and symmetric matrix that contains similarity values between pairs of pixels; the output is a clustering of pixels where each cluster corresponds to a region.

In the graph represented by a pixel similarity matrix $A$, a pixel is only connected to the pixels within some neighborhood. Thus, the input matrix $A$ is typically a sparse matrix. The similarity value between two neighboring pixels can be computed based on brightness, color, and texture cues [76, 79]. The similarity value characterizes the discontinuity along the line connecting the two pixels and can be trained by a logistic

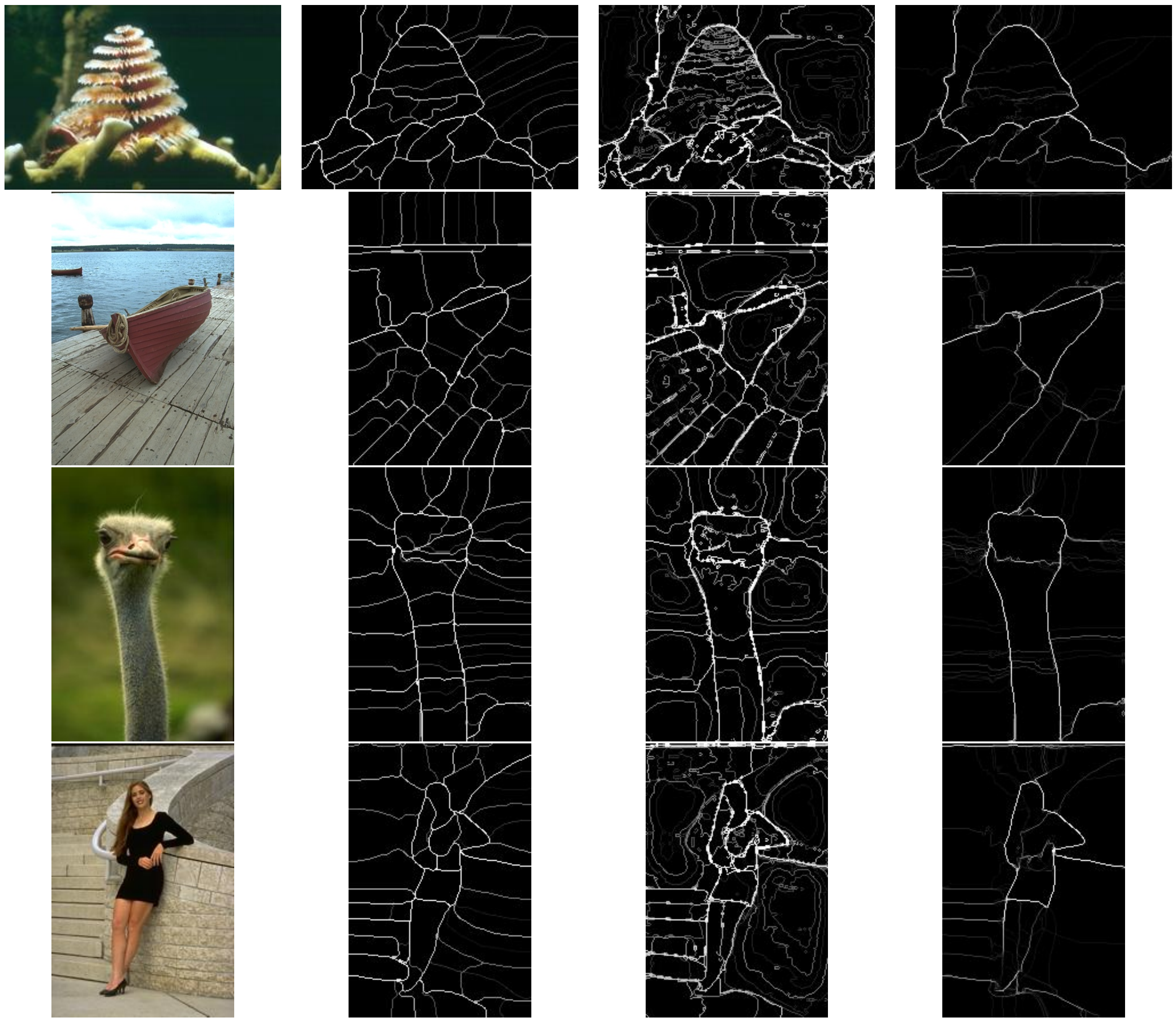

(a) Original (b) `Spectral-Embed` (c) `SymNMF-Embed` (d) `SymNMF-Clust`

Figure 9: Examples of the original images and $P_b$ images from BSDS500. Pixels with brighter color in the $P_b$ images have higher probability to be on the boundary.

model using human-marked boundaries as ground-truth [34].

Spectral clustering is one of the most common methods that solve the graph clustering problem in image segmentation. As we explained in Sections 3.3 and 3.4, because eigenvectors contain both positive and negative numbers in general, they cannot be used as cluster indicators directly. A variety of methods have been proposed to post-process the graph embedding – the continuous-valued eigenvectors – to obtain closed regions. In contrast, the low-rank matrix $B$ in the solution of SymNMF can not only be used as graph embedding, but also derive graph clustering results directly.

In the current chapter, our focus is the gain in segmentation quality by replacing spectral clustering with SymNMF. We follow the steps in an early paper [34] to construct the similarity matrix as well as post-process the graph embedding when the produced low-rank matrix is viewed as graph embedding. The post-processing steps are:

1. Run K-means on the embedded points to generate an oversegmentation of an image. The oversegmentations are called *superpixels* and denoted as $o_1, \cdots, o_K$, where $K$ is an integer larger than the rank $k$ of the low-rank matrix.

2. Build a contracted graph on the superpixels and represent it by a $K \times K$ similarity matrix $W$. The edge weight between the $I$-th and $J$-th superpixels ($1 \leq I, J \leq K$) is defined as:

$$W_{IJ} = \sum_{i \in o_I} \sum_{j \in o_J} A_{ij}. \tag{31}$$

3. Recursively split the contracted graph to produce a hierarchy of regions [76].

We note that the baseline segmentation algorithm [34] used in our comparison between spectral clustering and SymNMF is not the best algorithm to date (for example, see [1]). However, we chose this baseline algorithm in order to simplify the experiment setting and make the comparison more visible. In our current workflow, both spectral and SymNMF use the same similarity matrix as an input; the resulting low-rank matrices are interpreted as either graph embedding to produce a hierarchy of regions or graph clustering to produce a flat partitioning of an image into regions. With more recent segmentation algorithms such as [1], the low-rank matrices would be interpreted in a more sophisticated way so that we do not know which component of the segmentation algorithm contributes to the gain in segmentation quality. We expect that the comparison result shown in this section will carry on to other segmentation algorithms.

### 3.8.2 Data and Software

We use the Berkeley Segmentation Data Set 500[9] (BSDS500) [1] and choose the 200 color images used in [34]. The size of the original images is $481 \times 321$. We resized the images to $240 \times 160$ to be consistent with the experiments in [79, 34].

We compute the pixel similarity matrices and post-process the embedded points using the Berkeley Segmentation Engine[10]. We use the default settings: The number of eigenvectors in spectral clustering $k$ (and also the lower rank in SymNMF) is set to 16; the number of oversegmentations $K$ is set to 51. The neighborhood of a pixel is modified from default to a round disk centered at the pixel with radius of 20 pixels. The resulting similarity matrix has size $n \times n$ where $n = 38400$ and 44 million nonzeros. The same similarity matrix is given as an input to both spectral clustering and SymNMF.

### 3.8.3 Evaluation Methods

The evaluation of segmentation results is based on the evaluation of boundary detection. In the experiments on document and image clustering, solving SymNMF and interpreting the low-rank result matrix as a cluster indicator yield a hard clustering of items. In order to evaluate SymNMF in the context of image segmentation and compare its performance with that of spectral clustering, we introduce our way to transform the hard clustering results to soft boundaries. First, we generate a probability of boundary ($P_b$) image from multiple segmentations of an image. Second, we evaluate the $P_b$ image against human-marked boundaries.

- We consider the following three ways to obtain multiple segmentations:

    1. `Spectral-Embed`: Compute the eigenvectors associated with the 16 largest

[9] `http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/resources.html` (retrieved in June 2014)

[10] `http://www.cs.berkeley.edu/~fowlkes/BSE/` (retrieved in June 2014)

eigenvalues and treat them as a graph embedding. Generate a hierarchy of regions following the procedures in Section 3.8.1. Each level of the hierarchy determines a segmentation of the image.

2. `SymNMF-Embed`: Solve SymNMF with $k = 16$ and treat the rows of $B$ as a graph embedding. Generate a hierarchy of regions following the procedures in Section 3.8.1. Each level of the hierarchy determines a segmentation of the image.

3. `SymNMF-Clust`: Solve SymNMF with $k = 2, 3, \cdots, 16$ and treat each matrix $B$ as a cluster indicator. For each $k$, the clustering result corresponds to a segmentation.

`Spectral-Embed` and `SymNMF-Embed` produce 50 segmentations for each image. `SymNMF-Clust` produces 15 segmentations for each image. The $P_b$ value of a pixel is defined as the proportion of times the pixel lies on the boundary determined by the regions in a segmentation. Note that `SymNMF-Clust` does not enforce hierarchies in its segmentations. Among these three ways of post-processing, only `Spectral-Embed` was used for evaluation against human-marked boundaries in existing work.

- The data set includes human-marked boundaries by about 10 human subjects for each image for evaluation. The $P_b$ image has values in the $[0, 1]$ interval. We can produce a binary boundary image using a threshold value $t$ $(0 < t < 1)$. Then the *precision* $P$ is calculated as the fraction of true boundary pixels among all the detected boundary pixels; the *recall* $R$ is calculated as the fraction of detected boundary pixels among all the true boundary pixels. The F-measure is defined as $2PR/(P + R)$. We can draw a precision-recall curve using a series of threshold values (see more details in [78]). The best F-measure on this curve is regarded as a summary performance metric.

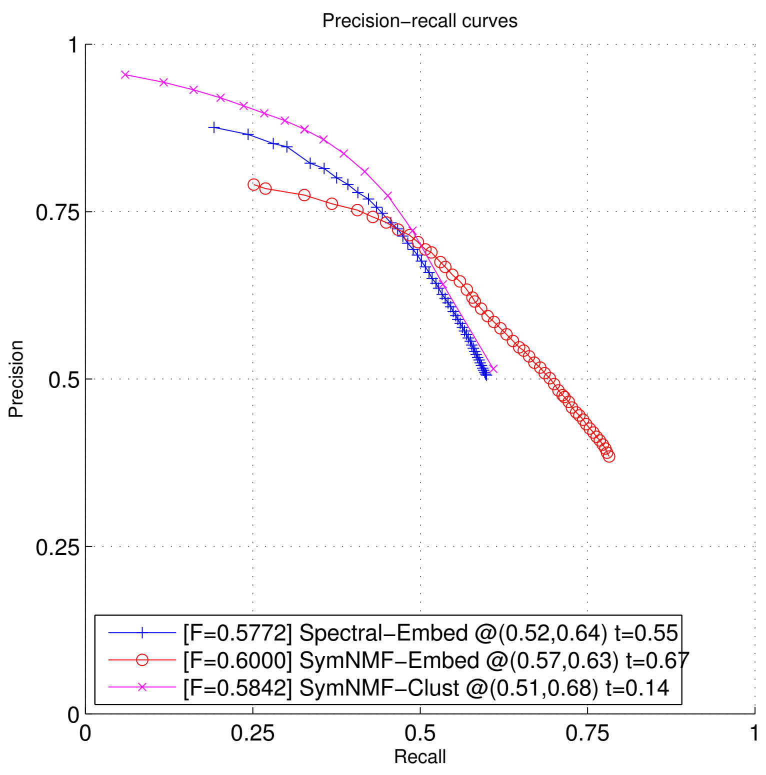


Figure 10: Precision-recall curves for image segmentation.

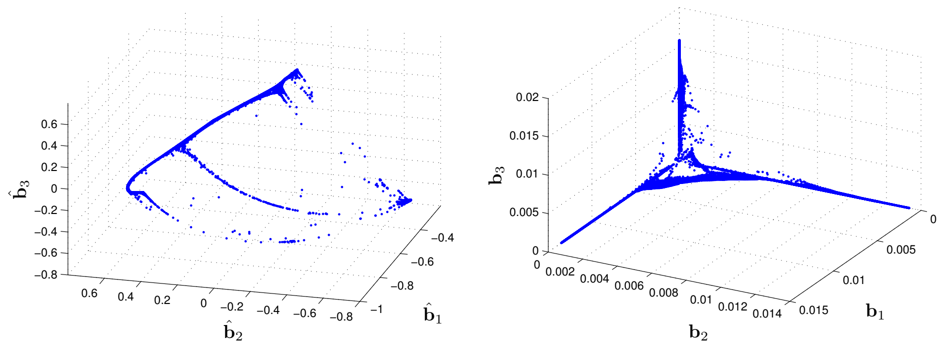
### 3.8.4 Results

We show the precision-recall curves for `Spectral-Embed`, `SymNMF-Embed`, and `SymNMF-Clust` in Fig. 10. Using the best F-measure as the summary metric, both SymNMF versions have better segmentation quality than either of the spectral clustering methods.

`SymNMF-Embed` is much better than `Spectral-Embed` in the high-recall low-precision area, with the highest recall approaching 0.8.

`SymNMF-Clust` is much better than `Spectral-Embed` in the high-precision low-recall area, and consistently better than `Spectral-Embed` along the curve. When the threshold value $t$ is close to 1, we can be much more confident about the detected regions using `SymNMF-Clust` than using `Spectral-Embed`.

Fig. 9 shows several exemplar images from the BSDS500 data set. The segmentation results are consistent with our findings in the precision-recall curve. We



(a) Spectral clustering (b) SymNMF

Figure 11: Illustration of different graph embeddings produced by spectral clustering and SymNMF for the third color image in Fig. 9. (a) The rows of the first three eigenvectors $\hat{B} \in \mathbb{R}^{n\times 3}$ are plotted. (b) The rows of $B \in \mathbb{R}_+^{n\times 3}$ in the result of SymNMF with $k = 3$ are plotted. Each dot corresponds to a pixel.

notice that `Spectral-Embed` often subdivides a large flat area with uniform colors into multiple regions (grass, sky, etc.). This is a well-known problem of image segmentation methods that rely on K-means to post-process the eigenvectors, and the reason is that the embedded points for the pixels in those areas vary smoothly [1]. On the contrary, `SymNMF-Clust` often leaves those areas intact, which implies that the low-rank matrix generated by SymNMF is a better cluster indicator. Fig. 11 shows the pixels plotted in the lower dimensional space produced by spectral clustering and SymNMF for a single image, which seems to support our reasoning above. We also notice that `SymNMF-Clust` tends to identify a few very small regions that correspond to noise in an image. This means that setting $k$ larger than needed will not degrade its segmentation quality. If we remove the regions whose areas are smaller than some threshold, we will see that many of the remaining regions correspond to meaningful objects.

In summary, we can use `SymNMF-Clust` to detect salient objects and use `SymNMF-Embed` to discover more detailed segments.

## *3.9 Discussion*

In this chapter, we studied Symmetric NMF (SymNMF): $\min_{B \geq 0} \|A - BB^T\|_F^2$ as a graph clustering method that is suitable for clustering data points embedded in linear and nonlinear manifolds. Our method extends the applicability of NMF to more general cases, where data relationship is not described by distances in vector space but by similarity values in a latent space. Unlike previous work on SymNMF that imposed various additional constraints on the matrix $B$, we showed that with nonnegativity constraints only, $B$ can be well interpreted as a cluster indicator matrix. We justified SymNMF to be a valid graph clustering method by showing that it originates from the same formulation as spectral clustering but relaxes the constraint on $B$ differently. While spectral clustering methods require post-processing the eigenvector-based data representation to obtain hard clusters, SymNMF does not depend on the spectrum and finds cluster memberships directly from $B$. Compared to previous work on the extension of NMF to a positive semi-definite and nonnegative matrix, our approach only assumes that $A$ is symmetric and nonnegative.

We developed two algorithms for SymNMF, a Newton-like algorithm and an ANLS-based algorithm, which have different properties but both are guaranteed to converge to stationary point solutions. We discuss the tradeoff between clustering quality and efficiency when choosing an algorithm for SymNMF. On one hand, the Newton-like algorithm often produces more accurate solutions and higher-quality clustering results, but is more appropriate when the problem size $n$ is small, e.g. $n < 3000$. On the other hand, the ANLS algorithm is especially efficient for a sparse input matrix $A$ and is scalable to very large data sets, e.g. $n \approx 10^6$. For large-scale clustering, we have to construct a sparse similarity matrix instead of a dense one. For example, with $n = 10^5$ data points, it is difficult to store a dense similarity matrix ($\sim$ 75 GB) into the main memory of a contemporary machine.

We have shown the promise of SymNMF in document clustering and image clustering. We also conducted a comprehensive evaluation of SymNMF for image segmentation on 200 natural images. Overall, we developed a *general* framework in this chapter, one with minimal constraints and flexible enough for extension. One limitation of our formulation is that an indefinite matrix $A$ could be approximated by a positive semi-definite matrix $BB^T$. Its effect requires further study; however, we have not seen evidences that the clustering performance degraded due to this limitation. The proposed algorithms can be easily parallelized, for example, in the Newton-like algorithm, the evaluation and Cholesky factorization of different diagonal blocks of the Hessian can run in parallel; and in the ANLS algorithm, the nonnegative least squares problem with different right-hand sides can run in parallel as well.

# CHAPTER IV

# CHOOSING THE NUMBER OF CLUSTERS AND THE APPLICATION TO CANCER SUBTYPE DISCOVERY

In the previous chapter, we considered an extension of the formulation of the standard NMF that applies to a symmetric graph similarity matrix. Starting from this chapter, we focus our discussion on the standard NMF again, where data points are represented in a vector space, forming a generally nonsymmetric nonnegative matrix $X$. As the analysis in Section 3.1 suggests, we keep our attention on the application scenarios where each cluster can be represented as linearly independent vectors and NMF performs well as a clustering method.

## *4.1 Motivation*

As in any clustering method, it is critical and often difficult to choose the right number of clusters when using NMF for clustering. This issue is referred to as *cluster validation* in the literature [40]. For NMF in particular, the matrices $W$ and $H$ in the solutions of NMF under different $k$'s have no direct relationships with each other. If we chose a wrong value of $k$ for some data set and compute NMF, the clustering result would provide misleading information about the true clusters, and it would be expensive to compute another NMF with a different $k$, unlike in other dimension reduction methods that are based on the SVD or symmetric eigenvalue decomposition. In the previous chapters, we have assumed in our discussion that $k$ is the actual number of clusters. It is interesting to study how an incorrect $k$ will influence the clustering result, which is the focus in this chapter.

A reliable method for cluster validation is especially useful for genomic data analysis such as cancer subtype discovery. Recall the formulation of the standard NMF:

$$X \approx WH. \tag{32}$$

Here each column of $X$ is the expression profile of a patient, and each row of $X$ correspond to a gene. The expression profiles can be obtained by either DNA microarray or RNA sequencing (RNASeq) technologies, measuring the abundances of the expression of all the genes. Our task is to find a handful of representative expression profiles that correspond to different cancer subtypes, represented by the columns of $W$, and to group the patients into those subtypes which can be determined by the columns of $H$. Physicians can offer personalized treatments based on the specific cancer subtype for a patient. Therefore, determining the right number of cancer subtypes is a key requirement in delivering accountable healthcare. Genomic data gathered from cancer patients have a much larger volume nowadays, such as in The Cancer Genome Atlas (TCGA), thus providing a great opportunity for comprehensive analysis of cancer subtypes.

Various existing methods for cluster validation are based on the notion of *stability* of the clusters. The intuition is that if a data set has $k$ **well-separated** clusters generated by a clustering algorithm, applying the same algorithm to a sub-sample of the data set will yield the same clustering assignment of the data points in that sub-sample. Ben-Hur et al. [8] evaluated the agreement of the clustering results generated on two random samples over multiple times of sub-sampling. Monti et al. [82] aggregated the clustering results generated on many random samples and reached a "consensus" of clustering results. They provided a visualization of the consensus clustering in a symmetric *consensus matrix*. In particular, Brunet et al. [16] adapted the consensus clustering method to the context of NMF for cluster validation, called *consensus NMF*, which has been very popular in gemonic analysis and employed as one of the default methodologies in the FIREHOSE data pipeline at TCGA Genome

Data Analysis Centers[1].

However, in this chapter, we will demonstrate by a surprisingly simple example that consensus NMF for cluster validation as described in Brunet et al. [16] has a critical flaw and may produce misleading results that suggest cluster structures when they do not exist. We will introduce the details of consensus NMF in Section 4.2 and its flaws in Section 4.3. In Section 4.4, we will propose a variation of another measure assessing the stability of clusters, called Gap of Prediction Strength (GPS), for cluster validation for NMF-based clustering. In Section 4.5, we show the effectiveness of our measure on artificially simulated data. Our discussion also suggests that an extended formulation of NMF, called affine NMF [64], is more appropriate for clustering genomic data, which will be presented in Section 4.6. In Section 4.7, we conduct a case study on *lung adenocarcinoma*, a common type of lung cancer. Our new methodology has theoretical implications in genomic studies, and will potentially drive more accurate discovery of cancer subtypes.

## *4.2 Consensus NMF*

We first review the consensus clustering for a generic clustering algorithm proposed by Monti et al. [82]. This method relies on the stability of the clusters with respect to random sampling of the data points. Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be two subsets sampled randomly from a data set of $n$ data points. Suppose two data points $\mathbf{x}_i, \mathbf{x}_j$ appear in both subsets generated by random sampling, that is to say, $\mathbf{x}_i, \mathbf{x}_j \in \mathcal{A}_1 \cap \mathcal{A}_2$. Let us run a clustering algorithm on both $\mathcal{A}_1$ and $\mathcal{A}_2$, assuming the correct number of clusters $k$ is given. Conceptually, we expect that if $\mathbf{x}_i, \mathbf{x}_j$ belong to the same cluster derived from $\mathcal{A}_1$, they also belong to the same cluster derived from $\mathcal{A}_2$. Based on this reasoning, we can aggregate the results of a clustering algorithm over many runs and achieve a consensus partitioning of the data points.

[1]`https://confluence.broadinstitute.org/display/GDAC/Home` (retrieved in June 2014)

Now we follow [82] and formalize the idea of a *consensus matrix*. For $n$ data points, the $(i, j)$-th entry of a consensus matrix $\tilde{C} \in \mathbb{R}^{n\times n}$ is the co-clustering frequency of the $i$-th and $j$-th data points over multiple runs of a clustering algorithm. More rigorously, let $r$ be the sampling rate, the fraction of data points selected in each random sample. We generate $T$ subsets $\mathcal{A}_1, \cdots, \mathcal{A}_T$ by random sampling, each with sampling rate $r$, and cluster each sub-sample with the same number of clusters $k$. Define the matrices $C^{(t)}$ and $S^{(t)}$ as the following $(1 \leq t \leq T)$:

$$c_{ij}^{(t)} = \begin{cases} 1, & \text{if the } i\text{-th and } j\text{-th data points belong to the same cluster using } \mathcal{A}_t; \\ 0, & \text{otherwise,} \end{cases} \tag{33}$$

$$s_{ij}^{(t)} = \begin{cases} 1, & \text{if both the } i\text{-th and } j\text{-th data points appear in } \mathcal{A}_t; \\ 0, & \text{otherwise.} \end{cases} \tag{34}$$

Clearly, $c_{ij}^{(t)} = 1$ implies $s_{ij}^{(t)} = 1$. Then we can define the consensus matrix $\tilde{C}$:

$$\tilde{c}_{ij} = \frac{\sum_{t=1}^{T} c_{ij}^{(t)}}{\sum_{t=1}^{T} s_{ij}^{(t)}}. \tag{35}$$

The entries of $\tilde{C}$ are co-clustering frequencies – the level of agreement of the clustering labels generated by multiple runs of a clustering algorithm – and have values in the interval $[0, 1]$. In the perfect scenario where no ambiguity exists for the co-membership of any two data points, the entries of $\tilde{C}$ could be 0 or 1 only. Thus, the histogram or cumulative distribution of the entries of $\tilde{C}$ on the interval $[0, 1]$ can be used to assess the stability of the clusters [82, 52].

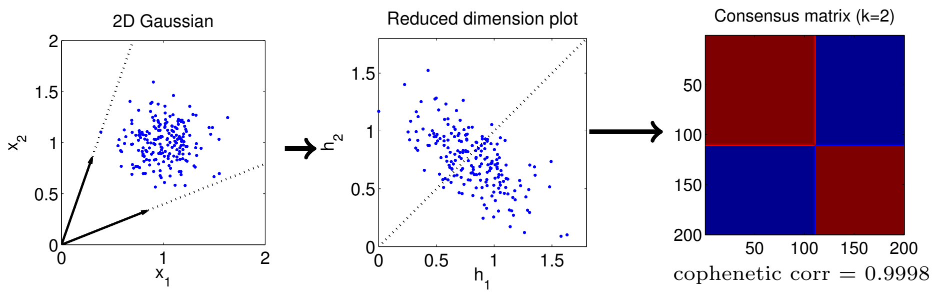
The consensus matrix can also serve as a visualizable evidence of stable or unstable clusters. $\tilde{C}$ can be viewed as a similarity matrix between all pairs of the data points, and can be provided as an input to a hierarchical clustering algorithm such as the average linkage, generating a binary tree called a dendrogram [33]. Given the same parameter $k$, we can cut the tree and produce $k$ non-overlapping partitions, i.e. clusters. Based on these cluster memberships, we can reorder the rows of $\tilde{C}$ so that the data points in the same cluster correspond to contiguous rows, and the same reordering is applied to the columns of $\tilde{C}$. If the cluster structures are stable with respect to random sampling, the reordered consensus matrix will exhibit $k$ diagonal blocks.

Brunet et al. [16] exploited the idea of consensus clustering in the context of NMF. They proposed to use another quantitative measure of clustering stability, called the *cophenetic correlation coefficient*, which is the correlation coefficient between the vectorized matrix $I - \tilde{C}$ and the distance metrics induced by the dendrogram tree [93]. The parameter $k$ corresponding to the largest cophenetic correlation was chosen as the right number of clusters. We refer to this consensus clustering method by Brunet et al. [16] as consensus NMF. A key characteristic that distinguishes consensus NMF from Monti et al.'s consensus clustering is that the random sampling procedure was replaced by random initialization in NMF. Virtually all of the algorithms for solving the optimization problem $\min_{W,H \geq 0} \|X - WH\|_F^2$ start from a random initialization of $W$ and $H$. Thus, consensus NMF chooses the number of clusters based on the stability of clustering results with respect to random initialization.

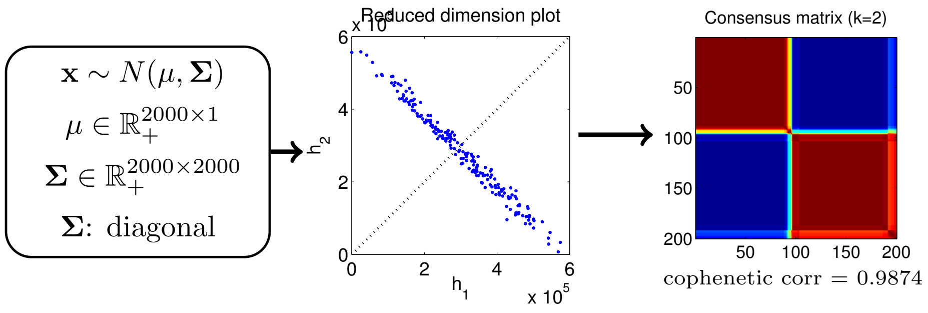
## 4.3 A Flaw in Consensus NMF

Here we show that consensus NMF is an erroneous method for cluster validation. The stability of clusters with respect to random initialization cannot suggest anything about the cluster structures. Consider a simple data set generated by a single Gaussian distribution in Fig. 12(a)(b), where no cluster structures are present; in other words, the right number of clusters is one. However, when using $k = 2$, consensus NMF generates a perfect consensus matrix with two diagonal blocks and clearly indicates two clusters.

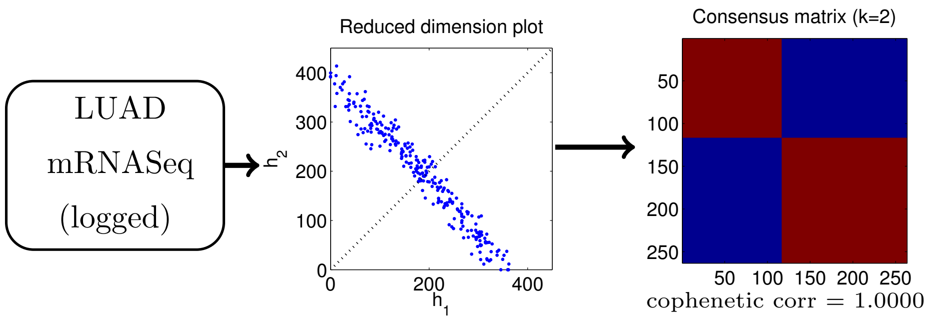
The reason for this misleading result given by consensus NMF is that a clustering



(a) Example on 2-D Gaussian data, $X \in \mathbb{R}_+^{2\times 200}$.



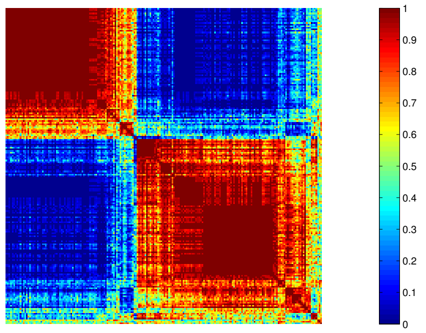
(b) Example on high-dimensional Gaussian data, $X \in \mathbb{R}_+^{2000\times 200}$.



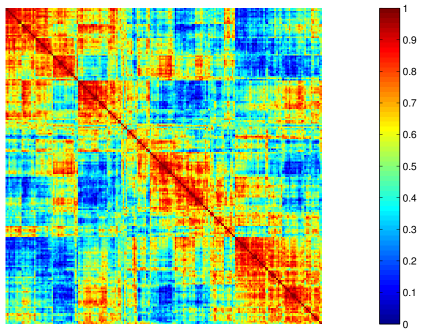
(c) Example on logged LUAD RNASeq data, $X \in \mathbb{R}_+^{5000\times 263}$.

Figure 12: Misleading results of consensus NMF on artificial and real RNASeq data. In each row: The left figure describes a data set in a plot or in words; the middle figure is a plot of the data set in the reduced dimensional space found by standard NMF with $k = 2$, where each column of $H$ is regarded as the 2-D representation of a data point; the right figure is the consensus matrix computed from 50 runs of standard NMF.

algorithm simply *partitions* a data set. Given a parameter $k$, most clustering algorithms generate $k$ partitions that minimize some underlying objective function. The clustering quality depends on whether those $k$ partitions happen to match the true cluster structures. In the above counterexample, the mechanism of NMF as a clustering method is to find two linearly independent vectors and treat them as cluster



(a) $X \in \mathbb{R}_+^{3\times 200}$



(b) $X \in \mathbb{R}_+^{2000\times 200}$

Figure 13: Reordered consensus matrices using Monti et al.'s method [82] and NMF as the clustering algorithm. The consensus matrices are constructed by computing 50 runs of the standard NMF on two artificial data sets, each generated by a single Guassian distribution. These results show that Monti et al.'s method based on random sampling does not suffer from the flaw in consensus NMF that is based on random initialization.

representatives to partition the data set into two groups. However, we have shown in Section 3.1 that a necessary condition for NMF to produce good clustering results is that the linearly independent basis vectors it finds correspond to the true clusters. The cases in Fig. 12(a)(b) do not satisfy this condition. Although the partitions given by NMF in this example are stable with respect to random initialization, they are not stable with respect to random sampling, shown in Fig. 13. We will also show more complicated simulation experiments in Section 4.5 that reaffirms our discovery.

Misleading conclusions can also be drawn from consensus NMF on real genomic data, such as in Fig. 12(c). The logarithm of the raw abundances of gene expression is often used for genomic analysis, for example, in the FIREHOSE pipeline[2]. Note that the result of NMF (middle figure) and the reordered consensus matrix (right figure) for the LUAD mRNASeq data closely resemble those for the artificial cases composed of a single Gaussian. Thus, we cannot deduce any useful information about the LUAD cancer subtypes from such analyses.

[2]`http://gdac.broadinstitute.org/runs/analyses__2014_04_16/reports/cancer/LUAD/mRNAseq_Clustering_CNMF/nozzle.html` (retrieved in June 2014)

Our discovery has several implications:

(1) The stability of clusters with respect to random initialization of a clustering algorithm is not an indicator of well-separated cluster structures. The stability result given by consensus NMF is determined not by the separability of the clusters but by the number of local optima in the NMF formulation. Therefore, the result of consensus NMF reflects the characteristic of the clustering algorithm, rather than the characteristic of the clusters themselves.

(2) Consensus clustering that evaluates the stability with respect to random sampling, as described by Monti et al. [82] is a reasonble cluster validation method. We can also compute the cophenetic correlation based on the consensus matrix and the dendrogram tree under different $k$'s, and choose the one with the largest cophenetic correlation as the number of clusters. However, we have found empirically that this measure often has similar values for a range of $k$'s and thus is not very informative for cluster validation. We will develop a new measure for cluster validation in the next section.

(3) The geometric structure of the low-dimensional representation of the patients in a single NMF run is also important for cluster validation. Ideally, when NMF discovers basis vectors that correspond well to well-separated clusters, the data points represented in the $k$-dimensional space should be located on or close to an axis in the $k$-dimensional coordinate system. In the middle figures in Fig. 12, the majority of data points in the low-dimensional space are located near the separation boundary, and it is not justifiable to assign cluster labels to those data points. We will propose to use affine NMF for genomic analysis to remedy this issue in Section 4.6.

(4) Contrary to common practice in genomic analysis, we have seen that NMF cannot reveal well-separated clusters from the logarithm of expression profiles.

We will use the raw expression profiles without taking the logarithm in our following experiments.

## 4.4 A Variation of Prediction Strength

We first review two existing measures for cluster validation, namely the *gap statistic* [97] and the *prediction strength* [96], and then propose our new cluster validation method based on these two measures.

### 4.4.1 Gap Statistic

The gap statistic [97] is a quantity defined for any distance-based clustering algorithm, i.e. an algorithm that relies on a distance function between pairs of data points. The intuition is that if a data set has $k$ well-separated clusters generated by a clustering algorithm, the within-class variation for these clusters should be sufficiently small compared to the within-class variation for the partitioning of the data points under a null distribution. Let $P = \{C_1, \cdots, C_k\}$ be a non-overlapping partitioning of $n$ data points, and $n_j = |C_j|$ $(1 \leq j \leq k)$ be the cardinality of the $j$-th cluster. The within-class variation is:

$$J_k = \sum_{j=1}^{k} \frac{1}{2n_r} \sum_{i,i' \in C_j} d_{ii'}, \tag{36}$$

'where $d_{ii'}$ is the distance between the $i$-th and $i'$-th data points. Then the gap statistic is defined as

$$\text{Gap}(k) = \mathbf{E}[(J_k)_{\text{null}}] - J_k, \tag{37}$$

where $(J_k)_{\text{null}}$ is the within-class variation under a null distribution, and $\mathbf{E}[\cdot]$ denotes the expectation of a random variable. The number of clusters is chosen as the smallest $k$ such that $\text{Gap}(k) \geq \text{Gap}(k+1) - \sigma[(J_k)_{\text{null}}]$, where $\sigma(\cdot)$ denotes the standard deviation and is included for a conservative estimate of the number of clusters.

### 4.4.2 Prediction Strength

The prediction strength measure [96] takes a prediction-based approach to the estimation of $k$. It evaluates the stability of cluster structures using repeated two-fold cross validation. In each iteration of the cross validation procedure, we randomly split the data set into a training set and a testing set, and apply a clustering algorithm to both the subsets and obtain the partitionings $P_{\text{train}}$ and $P_{\text{test}}$. Then, the cluster centers/representatives on the training set is used to predict the cluster labels of the data points in the testing set, resulting in $P_{\text{predict}}$. The prediction strength is a measure that evaluates the agreement between $P_{\text{predict}}$ and $P_{\text{test}}$. We simply denote this measure as PS and omit its exact definition here. PS is averaged over all the cross validation iterations and takes a value in the interval $[0, 1]$.

### 4.4.3 Gap of Prediction Strength

NMF used as a clustering method is not directly related to a distance function. Thus, prediction strength is a well suited measure for cluster validation for NMF-based clustering because it does not rely on a distance function. In each iteration of cross validation, we apply NMF to both the training and testing sets:

$$\begin{aligned} &\min_{W_{\text{train}} \geq 0, H_{\text{train}} \geq 0} \|X_{\text{train}} - W_{\text{train}} H_{\text{train}}\|_F^2, \\ &\min_{W_{\text{test}} \geq 0, H_{\text{test}} \geq 0} \|X_{\text{test}} - W_{\text{test}} H_{\text{test}}\|_F^2. \end{aligned} \tag{38}$$

We can obtain the partitioning $P_{\text{predict}}$ from $H_{\text{predict}}$, the solution of a nonnegative least squares problem:

$$H_{\text{predict}} = \arg\min_{H \geq 0} \|W_{\text{train}} H - X_{\text{test}}\|_F^2. \tag{39}$$

A simple strategy to choose the number of clusters is to pick $k$ with the largest PS value. However, we have found empirically that the prediction strength measure favors a smaller $k$ over a larger one. In [96], the authors suggested choosing the largest $k$ with a PS value larger than $0.8 \sim 0.9$, which was an *ad hoc* strategy and did not

**Algorithm 3** Generating a null distribution for a nonnegative data matrix $X \in \mathbb{R}_+^{m \times n}$

```
1: Input: $X = [\mathbf{x}_1, \cdots, \mathbf{x}_n]$
2: Compute the mean of the input data points: $\boldsymbol{\mu} = (1/n) \sum_{i=1}^{n} \mathbf{x}_i$
3: Translate the data points so that each feature has zero mean: $\mathbf{x}_i \leftarrow \mathbf{x}_i - \boldsymbol{\mu}, \forall i$
4: Compute the singular value decomposition: $X = U \Sigma V^T \equiv UY$, where $Y = [\mathbf{y}_1, \cdots, \mathbf{y}_n]$ is the data points in the principal component space
5: Get the $m$-dimensional bounding box for $[\mathbf{y}_1, \cdots, \mathbf{y}_n]$: $[\mathbf{y}_{\min}, \mathbf{y}_{\max}]$
6: Generate $n$ vectors uniformly in the bounding box: $\mathbf{z}_i \sim U(\mathbf{y}_{\min}, \mathbf{y}_{\max})$, forming a matrix $Z = [\mathbf{z}_1, \cdots, \mathbf{z}_n]$
7: Project $Z$ back to the original space: $Z \leftarrow UZ$
8: Translate $Z$ by $\boldsymbol{\mu}$: $\mathbf{z}_i \leftarrow \mathbf{z}_i + \boldsymbol{\mu}, \forall i$
9: Translate each row of $Z$ by the minimal amount such that $Z \in \mathbb{R}_+^{m \times n}$
10: Output: $Z$
```

generalize to an arbitrary data set. Therefore, we need to devise a more principled strategy for reliable cluster validation.

Inspired by the gap statistic, we propose to use the Gap of Prediction Strength (GPS) defined as:

$$\text{GPS}(k) = \text{PS}(k) - \mathbf{E}[\text{PS}(k)_{\text{null}}], \tag{40}$$

where $\text{PS}_{\text{null}}$ is the PS value for the data points under a null distribution. A possible null distribution can be obtained by generating each feature uniformly over the range in the principal components of the data points [97]. The benefit of generating the features in the transformed space instead of the original space is to capture the "shape" of the data set. The algorithm for obtaining the null distribution is summarized in Algorithm 3.

Our cluster validation method works as follows. For a range of $k$ values $2, 3, \cdots, k_{\max}$, we pick $k_{\text{opt}}$ with the largest GPS value as a candidate number of clusters. We accept the null hypothesis and decide the number of clusters is one if:

$$\text{PS}(k_{\text{opt}}) \leq \mathbf{E}[\text{PS}(k_{\text{opt}})_{\text{null}}] + \sigma[\text{PS}(k_{\text{opt}})_{\text{null}}]; \tag{41}$$

we reject the null hypothesis and decide the number of clusters is $k_{\text{opt}}$ if:

$$\text{PS}(k_{\text{opt}}) > \mathbf{E}[\text{PS}(k_{\text{opt}})_{\text{null}}] + \sigma[\text{PS}(k_{\text{opt}})_{\text{null}}]; \tag{42}$$

The standard deviation term is included here for reaching a conservative conclusion. Note that the way we incorporate the standard deviation into our decision rule is different from that for gap statistic.

## 4.5 Simulation Experiments

We perform empirical comparison on artificially simulated data sets where we have ground-truth knowledge of the cluster memberships, in order to evaluate the effectiveness of the cluster validation measures. The measures in our comparison include cophenetic correlation coefficients based on random initialization [16] and random sampling [82], the gap statistic [97], and the gap of prediction strength (GPS).

We construct artificial data sets that simulate the properties of gene expression profiles. Given the number of genes $M$, the number of patients $N$, and the number of clusters $K$, we construct a nonnegative data matrix $X \in \mathbb{R}_+^{M \times N}$ with $K$ clusters. First, generate the cluster centers $\boldsymbol{\mu}_1, \cdots, \boldsymbol{\mu}_K$. Each entry in these $k$ vectors is chosen uniformly from the $[0, 1]$ interval. Then, we generate the columns of $X$ as a mixture of Gaussians; that is, for each $1 \leq i \leq N$, we randomly pick a cluster center $\boldsymbol{\mu}_j$ and generate $\mathbf{x}_i \sim N(\boldsymbol{\mu}_j, \sigma^2)$. Two additional parameters control the simulation: The number of nuisance features, denoted as $M_{\text{noise}}$, simulating the non-informative genes that contain basically Gaussian noise; and the fraction of outlier entries, denoted as a scalar $z$, simulating inaccurate measurements and abnormal gene expression with no observable patterns. Finally, we translate the coordinate system so that $X$ is a nonnegative matrix.

We use the following parameters in the simulation: $M = 200$, $K = 1, 2, 3, 4, 5, 6$, $\sigma = 0.25, 0.3, 0.35, 0.4, 0.45, 0.5$, $M_{\text{noise}} = 80$, $z = 0.15$. We set $N = 60K$, that is, each ground-truth cluster has 60 data points.

For cophenetic correlation based on random initialization, we run NMF starting

Table 12: Accuracy of four cluster validation measures in the simulation experiments using standard NMF.

(a) Cophenetic correlation based on random initialization

| | $K = 1$ | $K = 2$ | $K = 3$ | $K = 4$ | $K = 5$ | $K = 6$ |
|---|---|---|---|---|---|---|
| $\sigma = 0.25$ | 0 | 1.0 | 0.4 | 0.8 | 1.0 | 0.7 |
| $\sigma = 0.3$ | 0 | 1.0 | 0.4 | 0.6 | 0.9 | 1.0 |
| $\sigma = 0.35$ | 0 | 1.0 | 0.3 | 0.8 | 1.0 | 1.0 |
| $\sigma = 0.4$ | 0 | 1.0 | 0.2 | 0.4 | 0.5 | 0.9 |
| $\sigma = 0.45$ | 0 | 1.0 | 0.1 | 0.1 | 0.5 | 0.6 |
| $\sigma = 0.5$ | 0 | 1.0 | 0 | 0 | 0.5 | 0.4 |

(b) Cophenetic correlation based on random sampling

| | $K = 1$ | $K = 2$ | $K = 3$ | $K = 4$ | $K = 5$ | $K = 6$ |
|---|---|---|---|---|---|---|
| $\sigma = 0.25$ | 0 | 1.0 | 0.7 | 1.0 | 1.0 | 0.9 |
| $\sigma = 0.3$ | 0 | 1.0 | 0.9 | 0.9 | 1.0 | 1.0 |
| $\sigma = 0.35$ | 0 | 1.0 | 0.8 | 1.0 | 0.9 | 1.0 |
| $\sigma = 0.4$ | 0 | 1.0 | 0.5 | 0.8 | 0.9 | 1.0 |
| $\sigma = 0.45$ | 0 | 1.0 | 0.4 | 0.6 | 1.0 | 0.9 |
| $\sigma = 0.5$ | 0 | 1.0 | 0.2 | 0.5 | 0.9 | 0.4 |

(c) Gap statistic

| | $K = 1$ | $K = 2$ | $K = 3$ | $K = 4$ | $K = 5$ | $K = 6$ |
|---|---|---|---|---|---|---|
| $\sigma = 0.25$ | 0.3 | 0.3 | 0.3 | 0.4 | 0.4 | 0.8 |
| $\sigma = 0.3$ | 0.5 | 0.1 | 0.4 | 0.5 | 0.7 | 0.8 |
| $\sigma = 0.35$ | 0.2 | 0.5 | 0.4 | 0.6 | 0.8 | 0.8 |
| $\sigma = 0.4$ | 0.4 | 0.2 | 0.3 | 0.3 | 0.6 | 0.8 |
| $\sigma = 0.45$ | 0.3 | 0.2 | 0.2 | 0.4 | 0.8 | 0.7 |
| $\sigma = 0.5$ | 0.3 | 0.2 | 0.3 | 0.1 | 0.7 | 0.6 |

(d) Gap of Prediction Strength (GPS)

| | $K = 1$ | $K = 2$ | $K = 3$ | $K = 4$ | $K = 5$ | $K = 6$ |
|---|---|---|---|---|---|---|
| $\sigma = 0.25$ | 0.2 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| $\sigma = 0.3$ | 0.1 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| $\sigma = 0.35$ | 0.1 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| $\sigma = 0.4$ | 0.3 | 1.0 | 0.9 | 1.0 | 1.0 | 1.0 |
| $\sigma = 0.45$ | 0.2 | 1.0 | 0.9 | 1.0 | 1.0 | 0.7 |
| $\sigma = 0.5$ | 0.3 | 1.0 | 0.9 | 0.9 | 0.7 | 0.4 |

from 50 different initializations. For cophenetic correlation based on random sampling, we build 50 random samples, and 80% of the data points are selected for each sub-sample. There is no mechanism to determine whether any cluster structure exists, i.e. $K = 1$, using these cophenetic correlation measures. For the gap statistic, we use Euclidean distance as the distance function in within-class variation (36) because the data points are generated by a mixture of Gaussians ignoring the nuisance features and outliers. For the GPS measure we proposed in this chapter, we conduct 50 iterations of two-fold cross validation. For the evaluation of all the four measures, NMF is used as the clustering algorithm, and we consider $k = 2, 3, 4, 5, 6, 7$ as candidate numbers of clusters.

For each set of parameters, we repeat the simulation for 10 times and report the proportion of times that the true number of clusters is detected, which we call the *accuracy* of a cluster validation measure. Table 12 show the accuracy results for the four measures. We can see that consensus NMF as described in Brunet et al. [16] performs much worse than re-sampling based consensus clustering [82], and GPS clearly outperforms the other three.

## *4.6 Affine NMF*

Inspired by the discussion in previous sections, we propose to remove the nuisance features in gene expression measurements to achieve more well-separated cluster structures. It is impractical to identify beforehand which genes are informative for cancer subtype discovery and which are not; in fact, this is one of the ultimate goals of genomic study. Instead, we assume that each gene expression measurement is composed of a nuisance part and an informative part.

In the standard NMF, each data point $\mathbf{x}_i$ is approximated in the latent subspace spanned by $k$ basis vectors, that is,

$$\mathbf{x}_i \approx \mathbf{w}_1 h_{1i} + \cdots + \mathbf{w}_k h_{ki}. \tag{43}$$

The basis vectors $\mathbf{w}_1, \cdots, \mathbf{w}_k$ are also called *metagenes* in genomic analysis. Now we explicitly model the nuisance part as a vector $\mathbf{w}_0$:

$$\mathbf{x}_i \approx \mathbf{w}_0 + \mathbf{w}_1 h_{1i} + \cdots + \mathbf{w}_k h_{ki}. \tag{44}$$

$\mathbf{w}_0$ can be viewed as a baseline expression profile of all the genes, or baseline metagene. Written in a matrix form, the new model is the standard NMF plus an offset:

$$X \approx \begin{bmatrix} \mathbf{w}_0 & W \end{bmatrix} \begin{bmatrix} \mathbf{e}^T \\ H \end{bmatrix}, \tag{45}$$

where $\mathbf{e} \in \mathbb{R}^{n\times 1}$ is a vector of all 1's. The formulation (45) was known as the *affine NMF* in the signal processing community. When we assume that the nuisance part contains Gaussian noise, the Frobenius norm can be used to define an optimization problem:

$$\min_{\mathbf{w}_0 \geq 0, W \geq 0, H \geq 0} \left\| X - \begin{bmatrix} \mathbf{w}_0 & W \end{bmatrix} \begin{bmatrix} \mathbf{e}^T \\ H \end{bmatrix} \right\|_F^2 . \tag{46}$$

The formulation (46) can be readily solved by an algorithm in the two-block coordinate descent framework [71, 53, 56, 57, 55]. We expect that the matrix $H$ found by affine NMF are sparser than that found by standard NMF and the metagenes $\mathbf{w}_1, \cdots, \mathbf{w}_k$ are more orthogonal to each other.

The four measures for cluster validation in our simulation experiments, namely cophenetic correlation based on random initialization and random sampling, the gap statistic, and the gap of prediction strength (GPS), are shown in Table 13, where affine NMF is used as the clustering algorithm. Comparing these results with those in the previous section using standard NMF, we can see that affine NMF successfully boosts the accuracy for all the four measures, especially when determining whether to accept or reject the null hypothesis ($K = 1$) in GPS.

Table 13: Accuracy of four cluster validation measures in the simulation experiments using affine NMF.

(a) Cophenetic correlation based on random initialization

| | $K = 1$ | $K = 2$ | $K = 3$ | $K = 4$ | $K = 5$ | $K = 6$ |
|---|---|---|---|---|---|---|
| $\sigma = 0.25$ | 0 | 1.0 | 0.9 | 0.7 | 0.3 | 0.1 |
| $\sigma = 0.3$ | 0 | 1.0 | 0.9 | 0.6 | 0.8 | 0.3 |
| $\sigma = 0.35$ | 0 | 1.0 | 1.0 | 0.5 | 0.4 | 0.3 |
| $\sigma = 0.4$ | 0 | 1.0 | 0.9 | 0.7 | 0.4 | 0.4 |
| $\sigma = 0.45$ | 0 | 0.9 | 0.9 | 0.6 | 0.4 | 0.4 |
| $\sigma = 0.5$ | 0 | 1.0 | 0.6 | 0.3 | 0.3 | 0.2 |

(b) Cophenetic correlation based on random sampling

| | $K = 1$ | $K = 2$ | $K = 3$ | $K = 4$ | $K = 5$ | $K = 6$ |
|---|---|---|---|---|---|---|
| $\sigma = 0.25$ | 0 | 1.0 | 1.0 | 0.8 | 1.0 | 1.0 |
| $\sigma = 0.3$ | 0 | 1.0 | 1.0 | 0.9 | 0.9 | 1.0 |
| $\sigma = 0.35$ | 0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| $\sigma = 0.4$ | 0 | 1.0 | 1.0 | 1.0 | 0.9 | 0.9 |
| $\sigma = 0.45$ | 0 | 1.0 | 1.0 | 1.0 | 1.0 | 0.9 |
| $\sigma = 0.5$ | 0 | 1.0 | 0.9 | 1.0 | 0.9 | 0.7 |

(c) Gap statistic

| | $K = 1$ | $K = 2$ | $K = 3$ | $K = 4$ | $K = 5$ | $K = 6$ |
|---|---|---|---|---|---|---|
| $\sigma = 0.25$ | 0.2 | 0.5 | 0.8 | 0.8 | 0.8 | 0.9 |
| $\sigma = 0.3$ | 0.3 | 0.6 | 0.7 | 1.0 | 0.6 | 0.8 |
| $\sigma = 0.35$ | 0 | 0.3 | 0.7 | 0.2 | 0.9 | 0.8 |
| $\sigma = 0.4$ | 0.4 | 0.1 | 0.5 | 0.5 | 0.5 | 0.8 |
| $\sigma = 0.45$ | 0.5 | 0 | 0.3 | 0.2 | 0.5 | 0.7 |
| $\sigma = 0.5$ | 0.3 | 0.1 | 0 | 0.2 | 0.9 | 0.7 |

(d) Gap of Prediction Strength (GPS)

| | $K = 1$ | $K = 2$ | $K = 3$ | $K = 4$ | $K = 5$ | $K = 6$ |
|---|---|---|---|---|---|---|
| $\sigma = 0.25$ | 1.0 | 0.9 | 1.0 | 1.0 | 1.0 | 1.0 |
| $\sigma = 0.3$ | 0.9 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| $\sigma = 0.35$ | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| $\sigma = 0.4$ | 0.9 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| $\sigma = 0.45$ | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 0.7 |
| $\sigma = 0.5$ | 0.9 | 1.0 | 1.0 | 1.0 | 0.8 | 0.3 |

Table 14: Average entropy $E(k)$ computed on the LUAD data set, for the evaluation of the separability of data points in the reduced dimensional space.

|  | $k = 2$ | $k = 3$ | $k = 4$ | $k = 5$ |
|---|---|---|---|---|
| Standard NMF | 0.5778 | 0.8655 | 1.1378 | 1.3155 |
| Affine NMF | 0.2918 | 0.6892 | 0.9608 | 1.2322 |

## 4.7 Case Study: Lung Adenocarcinoma

In this chapter, we have shown that a popular framework for cancer subtype discovery, namely the consensus NMF [16], is flawed. We have proposed a new framework based on affine NMF for clustering and Gap of Prediction Strength (GPS) for cluster validation. Now we conduct a case study on *lung adenocarcinoma* (LUAD), a specific type of lung cancer. The data set we collected was a snapshot of LUAD RNASeq data on the TCGA website. We selected 5,000 genes with the largest coefficients of variation ($\sigma/\mu$) [16] and formed a data matrix $X \in \mathbb{R}_+^{5000\times 263}$.

First, we apply both standard NMF and affine NMF to the LUAD data matrix and evaluate the separability of the cluster structures. Besides visual inspection, we use the average entropy as a quantitative measure of separability. The average entropy over all the columns of $H$ is computed by treating each column of $H$ after normalization as a probability distribution in the low-dimensional space:

$$E(k) = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{k} P_{ji}\log P_{ji} \tag{47}$$

where $P \in \mathbb{R}^{k\times n}$ and $P_{ji} = H_{ji}/\sum_{j'=1}^{k} H_{j'i}$. A smaller value of average entropy correspond to better separability.

The low-dimensional representation of the data points, i.e. the columns of $H$, are plotted for the $k = 2$ case in Fig. 14. The average entropy values for $k = 2, 3, 4, 5$ are shown in Table 14. We can clearly see that affine NMF generates more well-separated clusters. In the result of affine NMF, more data points are located on or close to an axis, which means that the metagenes are better cluster representatives than those generated by standard NMF, and we are more confident in assigning hard cluster

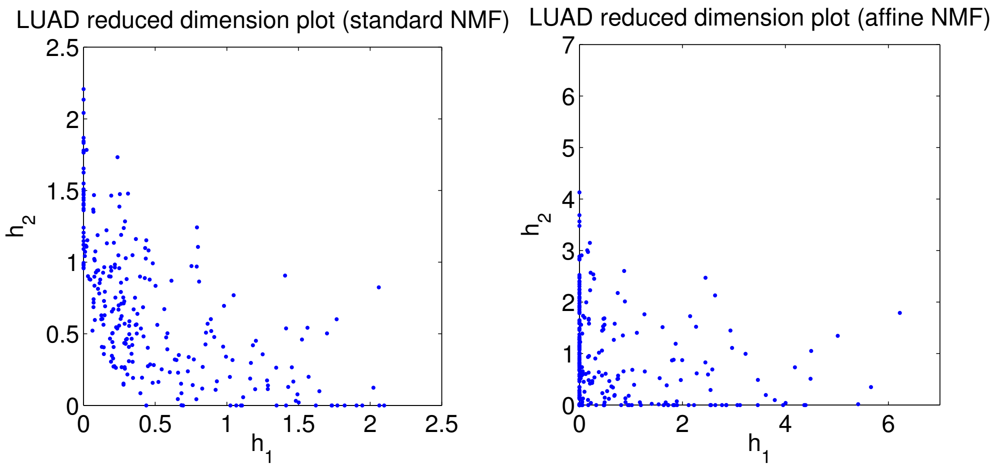


Figure 14: Reduced dimensional plots generated by standard NMF and affine NMF.

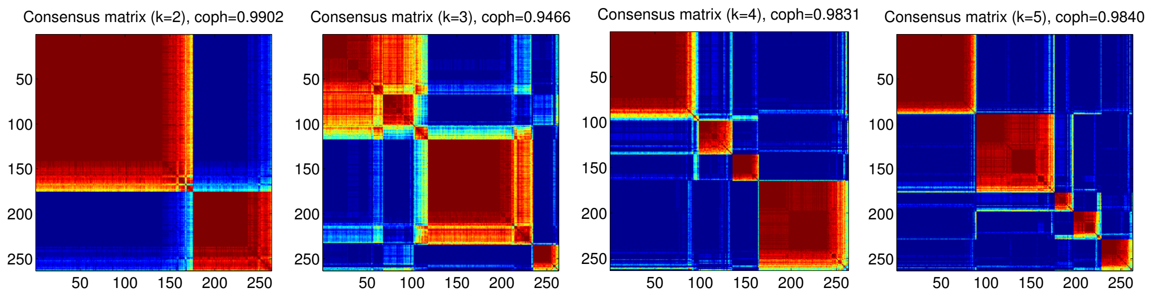


Figure 15: Reordered consensus matrix and cophenetic correlation based on random sampling [82] when using standard NMF on the LUAD data set for $k = 2, 3, 4, 5$. Results generated by affine NMF are similar. A block diagonal structure appears in three out of the four cases with different $k$'s.

labels to the data points.

Next, we compare the cophenetic correlation based on random sampling and GPS for cluster validation. The gap statistic is not included because no suitable distance function can be found. The parameter configuration is the same as that in simulation experiments (Section 4.5). When computing PS for each $k$, we use the median over all the cross validation iterations instead of the mean in order to have a robust estimate of PS.

The reordered consensus matrices for $k = 2, 3, 4, 5$ are shown in Fig. 15. We can see block diagonal patterns in all four cases, and the cophenetic correlation values are also very similar to each other, indicating that it is not a good metric for cluster

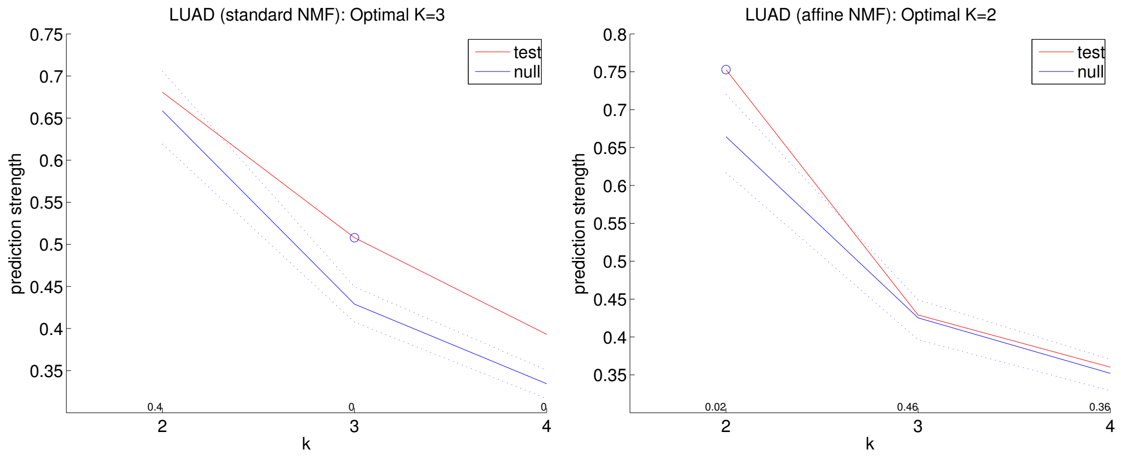


Figure 16: Prediction strength measures for the LUAD data set (red curve, labeled as 'test') as well as the data under a null distribution generated by Algorithm 3 (blue curve, labeled as 'null'). Results for both standard NMF and affine NMF are shown. The blue dotted curves indicate the 1-standard-deviation of PS values under the null distribution. The blue circles indicate the number $K$ with the largest GPS. The numbers displayed above the horizontal axis are empirical p-values for the observed PS under the null distribution. These results show that GPS is an effective measure for cluster validation.

validation.

For the GPS measure, we show the PS values for both the actual data matrix and the data under a null distribution in Fig. 16. The curves corresponding to the null distribution (blue) show the preference towards a smaller $k$ by the prediction strength measure, which verifies our motivation to propose the GPS measure. GPS with affine NMF determines two LUAD subtypes while GPS with standard NMF determines three LUAD subtypes. This means that removing the baseline expression profiles has an impact on the number of subtypes that NMF-based clustering discovers. Looking into more details at the clustering results, one of the three clusters found by standard NMF is very small and contains about 10% of the patients. This finding makes us believe that the three-cluster result is an artifact and the two clusters given by affine NMF are a better clustering. Further analysis with additional molecular data and clinical data is required to evaluate the biological significance of the clusters.

# CHAPTER V

# FAST RANK-2 NMF FOR HIERARCHICAL DOCUMENT CLUSTERING

## *5.1 Flat Clustering Versus Hierarchical Clustering*

Most of the previous work on clustering with NMF has been focused on flat partitioning of a data set. However, hierarchical clustering often reveals additional structures in the data. For example, a tree structure often provides a more detailed taxonomy or a better description of natural phenomena than a flat partitioning. In the widely-used text corpus RCV1 [68], a hierarchy of topics was defined, with 103 leaf nodes under four super categories (Corporate/Industrial, Economics, Government/Social, Markets). Online retailers such as Amazon and eBay also maintain their product catalogues as a hierarchical tree structure. In this chapter, we will explore hierarchical clustering with NMF and present a highly efficient algorithm with competitive clustering quality.

The lower rank $k$ in standard NMF, which represents the number of clusters in a clustering setting, is often assumed to be given or predetermined according to prior knowledge about the data set or the embedding of the data points. Selecting the number of clusters $k$ is an important and difficult issue in practice. Though model selection methods for selecting $k$ have been proposed in the context of NMF [16, 47], it is expensive to compute solutions of NMF for each $k$ in general [55]. In the NMF-based hierarchical approach we propose in this chapter, a data set is recursively divided into small subsets and the number of clusters does not need to be predetermined by a user.

We will design a hierarchical clustering method based on rank-2 NMF, i.e. NMF with $k = 2$. The hierarchical structure we will generate is a binary tree, and our method does not require any input on the level of the tree or the total number of clusters. Our motivation for hierarchical clustering with binary tree structure is based on our fast algorithm for rank-2 NMF proposed in this chapter. We will exploit the special properties of NMF with $k = 2$, and propose a very fast algorithm. We will study a particular type of existing algorithms for standard NMF, namely *active-set-type algorithms* [53, 56], and propose a simple and efficient active-set-type algorithm for rank-2 NMF, which has additional benefits when implemented on parallel platforms due to "non-random" memory access.

When applying rank-2 NMF to the recursive splitting of a text corpus, our empirical results reveal that if a balanced tree is constructed, its clustering quality is often worse than that of a flat partitioning. Thus we need to adaptively determine the next node to split. Our strategy is to compute a score for each leaf node to evaluate whether it is composed of two well-separated clusters based on the two basis vectors generated by rank-2 NMF before deciding which one to split. Compared to existing strategies that rely on an $n \times n$ document-document similarity matrix [27], our methodology never generates a large dense matrix thus is more time/space efficient. Although the proposed hierarchical clustering workflow contains redundant computation, our methodology is still very efficient overall due to the high efficiency of our rank-2 NMF algorithm.

Our contributions in this chapter include:

- We propose an active-set-type algorithm for rank-2 NMF, which is fast, guaranteed to converge, and easy to parallelize.
- By combining rank-2 NMF with a scoring function for every leaf node, we develop an efficient workflow for hierarchical document clustering. Our methodology is able to determine both the tree structure and the depth of the tree on-the-fly and detect outliers, in contrast to hierarchical probabilistic modeling methods [13] that require the depth of the tree be specified by the user.

- We present promising empirical results of our methodology in terms of efficiency, clustering quality, as well as semantic quality in the topic modeling context. To the best of our knowledge, our work is the first attempt to cluster the full RCV1 data set [68] which contains approximately 800,000 documents. Our method finished in about 7 minutes on a shared-memory machine with two quad core CPUs and achieved better quality than standard NMF which costs 4.5 hours.

The rest of the chapter is organized as follows. We conduct detailed analysis of existing active-set-type algorithms for NMF in the special case of $k = 2$ in Section 5.2, and present our new algorithm for rank-2 NMF in Section 5.3. In Section 5.4, we describe our measure for scoring tree nodes and the hierarchical document clustering workflow. In Section 5.5, we show the difference between clustering approaches and topic modeling approaches when applied to flat and hierarchical document clustering. In Section 5.6, we demonstrate the promising efficiency, clustering quality, and semantic quality of our methodology empirically on large-scale data sets. In Section 5.7, we summarize the advantages and shortcomings of this work. Although we focus on document clustering, the proposed hierarchical clustering method is not limited to documents.

Throughout this chapter, $\|\cdot\|$ denotes the Euclidean norm, and $\|\cdot\|_F$ denotes the Frobenius norm.

## 5.2 Alternating Nonnegative Least Squares for NMF

In this chapter, we consider the algorithms for NMF that fit into the two-block coordinate descent framework [71, 53, 56, 55] due to better theoretical guarantee in convergence properties. In this framework, starting from some initialization, the matrices $W$ and $H$ are updated in an iterative manner, until some stopping criterion is satisfied. The overall nonconvex problem (2) is thus reduced to two convex subproblems:

$$\min_{W\geq 0} \quad \|H^T W^T - X^T\|_F^2, \tag{48}$$

$$\min_{H\geq 0} \quad \|WH - X\|_F^2. \tag{49}$$

When an optimal solution is obtained for each subproblem in each iteration, this iterative procedure is guaranteed to converge to a stationary point [38], which is a good convergence property for nonconvex problems such as (2). Each subproblem is a nonnegative least squares problem (NNLS) with multiple right-hand sides. Consider the following generic form for simplicity:

$$\min_{G\geq 0} \|BG - Y\|_F^2, \tag{50}$$

where $B \in \mathbb{R}_+^{m\times k}, Y \in \mathbb{R}_+^{m\times n}$ are given, and $G \in \mathbb{R}_+^{k\times n}$ is to be solved.

Various types of algorithms can be used to solve the NNLS problem and can be categorized into standard optimization algorithms and active-set-type algorithms. A classical active-set algorithm for NNLS with a single right-hand side was introduced in [65]. In the context of NMF, Lin [71] claimed that it would be expensive to solve NNLS with multiple right-hand sides using the active-set algorithm repeatedly, and proposed a projected gradient descent (PGD) algorithm. However, Kim & Park [53] proposed several improvements for the original active-set algorithm, and achieved an NMF algorithm with overall efficiency comparable to PGD. Later, a block-pivoting algorithm for NNLS [56] was proposed, which proved to be more efficient than the active-set algorithm when $k$ is large. We call both active-set based and block-pivoting based algorithms for NMF as *active-set-type algorithms*.

In active-set-type algorithms for NNLS, we need to identify a partitioning of variables in $G$ into an *active set* $\mathcal{A}$ and a *passive set* $\mathcal{P}$. At each step of searching for the optimal active set, we need to solve an unconstrained least squares problem. Because the number of possible active sets is exponential, a well-guided search of the optimal active sets is important, such as presented by the active-set and block-pivoting methods. To improve the efficiency of solving NNLS with multiple right-hand sides (50), the columns of $G$ with the same active set pattern are grouped together for lower computational complexity and more cache-efficient computation [53, 98], and the grouping of columns changes when the active set is re-identified in each iteration of NNLS. Practically, the grouping step is implemented as a sorting of the columns of $G$, with complexity $O(n \log n)$ which is expensive when $n$ is large. Other steps, such as checking the optimality of the active sets, also introduces additional overheads.

When the underlying application restricts the value of $k$ to be 2, such as hierarchical clustering that generates a binary tree, the number of possible active sets is reduced to $2^2 = 4$ for each column of $G$, and it is practical to enumerate all of them. Conceptually, active-set-type algorithms search for the optimal active set in a finite set of possible active sets, and the size of the finite search space is 4 in the special case of $k = 2$. On the contrary, standard optimization algorithms require an indefinite number of iterations before convergence, and the actual number of iterations depends on the required accuracy. Therefore, when $k = 2$, standard optimization algorithms such as PGD are not able to exploit the special property of NNLS, and active-set-type algorithms become the better choice. In the next section, we will propose a new algorithm and its efficient implementation based on active-set-type algorithms, which will avoid all the overheads of switching between active sets.

Both flat clustering based on standard NMF and hierarchical clustering based on rank-2 NMF can produce $k$ non-overlapping groups of a data set. In the following, we argue that the hierarchical approach is the preferred choice in terms of efficiency by conducting an analysis of computational complexity for different $k$ values, using the active-set based algorithm [53] as an exemplar algorithm. Given an $m \times n$ sparse data matrix $X$ and the number of clusters $k$, the complexity of one NNLS run for standard NMF is upper bounded by:

$$4kN + 2(m+n)k^2 + t[(1/3)k^3 + 2(m+n)k^2] \text{ flops}, \tag{51}$$

where $N$ is the number of nonzero entries in $X$, and $t$ is the number of search steps towards the optimal active set. In hierarchical clustering, we need to perform rank-2 NMF for $k-1$ times, and the complexity of one NNLS run summed over all the $k-1$ steps is at most:

$$(k-1) \cdot [8N + 8(m+n) + t(8/3 + 8m + 8n)] \text{ flops}. \tag{52}$$

The actual flops (floating point operations) in hierarchical clustering must be smaller than (52), because any splitting other than the first step is executed on a subset of the data set only. Thus, the expression (51) is superlinear with respect to $k$, while (52) is linear with respect to $k$. Assuming the number of search steps $t$ is the same in both cases, the hierarchical approach is expected to be much less expensive. With our new algorithm specifically for rank-2 NMF in this chapter, the efficiency of NMF-based hierarchical clustering will be boosted even more.

## 5.3 A Fast Algorithm for Nonnegative Least Squares with Two Columns

In ANLS, the problem of solving NMF is reduced to the problem of solving NNLS with multiple right-hand sides: $\min_{G \geq 0} \|BG - Y\|_F^2$. In the context of NMF, $Y$ is set to be either the data matrix $X$, or $X^T$. Let $Y = [\mathbf{y}_1, \cdots, \mathbf{y}_n], G = [\mathbf{g}_1, \cdots, \mathbf{g}_n]$. We emphasize that $\mathbf{y}_i \geq 0$ $(1 \leq i \leq n)$ in the NNLS problem we are solving since the data is nonnegative.

Since the formulation (50) for NNLS with multiple right-hand sides can be rewritten as

$$\min_{\mathbf{g}_1, \cdots, \mathbf{g}_n \geq 0} \|B\mathbf{g}_1 - \mathbf{y}_1\|^2 + \cdots + \|B\mathbf{g}_n - \mathbf{y}_n\|^2, \tag{53}$$

Table 15: Four possible active sets when $B \in \mathbb{R}_+^{m\times 2}$.

| $\mathcal{A}$ | $\mathcal{P}$ | $J(\mathbf{g})$ |
|---|---|---|
| $\{1,2\}$ | $\emptyset$ | $\|\mathbf{y}\|^2$ |
| $\{1\}$ | $\{2\}$ | $\|\mathbf{b}_2 g_2 - \mathbf{y}\|^2$ |
| $\{2\}$ | $\{1\}$ | $\|\mathbf{b}_1 g_1 - \mathbf{y}\|^2$ |
| $\emptyset$ | $\{1,2\}$ | $\|\mathbf{b}_1 g_1 + \mathbf{b}_2 g_2 - \mathbf{y}\|^2$ |

the solution of each column of $G$ is independent of each other, and we obtain $n$ NNLS problems each with a single right-hand side. We first study the solution of NNLS with a single right-hand side, and then consider the issues when combining multiple right-hand sides.

### 5.3.1 Solution of $\min_{\mathbf{g}\geq 0} \|B\mathbf{g} - \mathbf{y}\|$ with $B \in \mathbb{R}_+^{m\times 2}, \mathbf{y} \geq 0$

In general, when $B$ has more than two columns, an active-set-type algorithm has to search for an optimal active set as discussed in Section 5.2. We denote the two parts of $\mathbf{g}$ that correspond to the active set and the passive set as $\mathbf{g}_{\mathcal{A}}$ and $\mathbf{g}_{\mathcal{P}}$, respectively. At each iteration of the algorithm, $\mathbf{g}_{\mathcal{A}}$ is set to zero, and $\mathbf{g}_{\mathcal{P}}$ is solved by unconstrained least squares using a subset of columns of $B$ corresponding to $\mathcal{P}$. The optimal active set is the one where the solution of unconstrained least squares is feasible, i.e. $\mathbf{g}_{\mathcal{P}} \geq 0$, and meanwhile $\|B\mathbf{g} - \mathbf{y}\|^2$ is minimized.

When $k = 2$, we have

$$J(\mathbf{g}) \equiv \|B\mathbf{g} - \mathbf{y}\|^2 = \|\mathbf{b}_1 g_1 + \mathbf{b}_2 g_2 - \mathbf{y}\|^2, \tag{54}$$

where $B = [\mathbf{b}_1, \mathbf{b}_2] \in \mathbb{R}_+^{m\times 2}$, $\mathbf{y} \in \mathbb{R}_+^{m\times 1}$, and $\mathbf{g} = [g_1, g_2]^T \in \mathbb{R}^{2\times 1}$.

Considering the limited number of possible active sets, our idea is to avoid the search of the optimal active set at the cost of some redundant computation. The four possibilities of the active set $\mathcal{A}$ is shown in Table 15. We simply enumerate all the possibilities of $(\mathcal{A}, \mathcal{P})$, and for each $\mathcal{P}$, minimize the corresponding objective function $J(\mathbf{g})$ in Table 15 by solving the unconstrained least squares problem. Then, of all the feasible solutions of $\mathbf{g}$ (i.e. $\mathbf{g} \geq 0$), we pick the one with the smallest $J(\mathbf{g})$. Now we

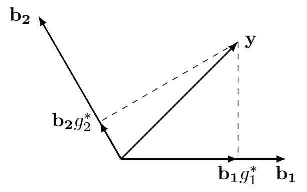


Figure 17: An illustration of one-dimensional least squares problems $\min \|\mathbf{b}_1 g_1^* - \mathbf{y}\|^2$ and $\min \|\mathbf{b}_2 g_2^* - \mathbf{y}\|^2$.

study the properties of the solutions of these unconstrained least squares problems, which will lead to an efficient algorithm to find the optimal active set.

First, we claim that the two unconstrained problems $\min \|\mathbf{b}_1 g_1 - \mathbf{y}\|$, $\min \|\mathbf{b}_2 g_2 - \mathbf{y}\|$ always yield feasible solutions. Take $\min \|\mathbf{b}_1 g_1 - \mathbf{y}\|^2$ as an example. Its solution is:

$$g_1^* = \frac{\mathbf{y}^T \mathbf{b}_1}{\mathbf{b}_1^T \mathbf{b}_1}. \tag{55}$$

If $\mathbf{b}_1 \neq 0$, we always have $g_1^* \geq 0$ since $\mathbf{y} \geq 0, \mathbf{b}_1 \geq 0$. In the context of rank-2 NMF, the columns of $W$ and the rows of $H$ are usually linearly independent when nonnegative-rank$(X) \geq 2$, thus $\mathbf{b}_1 \neq 0$ holds in most cases. Geometrically (see Fig. 17), the best approximation of vector $\mathbf{y}$ in the one-dimensional space spanned by $\mathbf{b}_1$ is the orthogonal projection of $\mathbf{y}$ onto $\mathbf{b}_1$.

If $\mathbf{g}^\emptyset \equiv \arg\min \|\mathbf{b}_1 g_1 + \mathbf{b}_2 g_2 - \mathbf{y}\|^2$ is nonnegative, then $\mathcal{A} = \emptyset$ is the optimal active set because the unconstrained solution $\mathbf{g}^\emptyset$ is feasible and neither $\min \|\mathbf{b}_1 g_1 - \mathbf{y}_1\|^2$ nor $\min \|\mathbf{b}_2 g_2 - \mathbf{y}_2\|^2$ can be smaller than $J(\mathbf{g}^\emptyset)$. Otherwise, we only need to find the smallest objective $J(\mathbf{g})$ among the other three cases since they all yield feasible solutions. We claim that $\mathcal{A} = \{1, 2\}$, i.e. $\mathcal{P} = \emptyset$, can be excluded. Using $g_1^*$, the solution of $\min \|\mathbf{b}_1 g_1 - \mathbf{y}\|^2$, we have

$$\|\mathbf{b}_1 g_1^* - \mathbf{y}\|^2 = \|\mathbf{y}\|^2 - (\mathbf{y}^T \mathbf{b}_1)^2 / (\mathbf{b}_1^T \mathbf{b}_1) \leq \|\mathbf{y}\|^2. \tag{56}$$

In fact, $\mathcal{P} = \{1\}$ includes $\mathcal{P} = \emptyset$ as a special case when $\mathbf{b}_1 \perp \mathbf{y}$.

To compare $\|\mathbf{b}_1 g_1^* - \mathbf{y}\|^2$ and $\|\mathbf{b}_2 g_2^* - \mathbf{y}\|^2$, we note that in the illustration in Fig.

**Algorithm 4** Algorithm for solving $\min_{\mathbf{g}\geq 0} \|B\mathbf{g} - \mathbf{y}\|^2$, where $B = [\mathbf{b}_1, \mathbf{b}_2] \in \mathbb{R}_+^{m\times 2}, \mathbf{y} \in \mathbb{R}_+^{m\times 1}$

```
1: Solve unconstrained least squares $\mathbf{g}^{\emptyset} \leftarrow \min \|B\mathbf{g} - \mathbf{y}\|^2$ by normal equation
   $B^T B\mathbf{g} = B^T\mathbf{y}$
2: if $\mathbf{g}^{\emptyset} \geq 0$ then
3:    return $\mathbf{g}^{\emptyset}$
4: else
5:    $g_1^* \leftarrow (\mathbf{y}^T\mathbf{b}_1)/(\mathbf{b}_1^T\mathbf{b}_1)$
6:    $g_2^* \leftarrow (\mathbf{y}^T\mathbf{b}_2)/(\mathbf{b}_2^T\mathbf{b}_2)$
7:    if $g_1^*\|\mathbf{b}_1\| \geq g_2^*\|\mathbf{b}_2\|$ then
8:       return $[g_1^*, 0]^T$
9:    else
10:      return $[0, g_2^*]^T$
11:   end if
12: end if
```

17, $\mathbf{b}_1 g_1^* - \mathbf{y} \perp \mathbf{b}_1 g_1^*$ and $\mathbf{b}_2 g_2^* - \mathbf{y} \perp \mathbf{b}_2 g_2^*$, therefore we have

$$\|\mathbf{b}_j g_j^*\|^2 + \|\mathbf{b}_j g_j^* - \mathbf{y}\|^2 = \|\mathbf{y}\|^2 \tag{57}$$

for $j = 1, 2$. Thus choosing the smaller objective amounts to choosing the larger value from $g_1^*\|\mathbf{b}_1\|$ and $g_2^*\|\mathbf{b}_2\|$.

Our algorithm for NNLS with a single right-hand side is summarized in Algorithm 4. Note that $B^T B$ and $B^T\mathbf{y}$ need only to be computed once for lines 1,5,6.

### 5.3.2 Solution of $\min_{G\geq 0} \|BG - Y\|_F$ with $B \in \mathbb{R}_+^{m\times 2}, Y \geq 0$

When Algorithm 4 is applied to NNLS with multiple right-hand sides, computing $\mathbf{g}^{\emptyset}, g_1^*, g_2^*$ for different vectors $\mathbf{y}$ separately is not cache-efficient. In Algorithm 5, we solve NNLS with $n$ different vectors $\mathbf{y}$ simultaneously, and the analysis in Section 5.3.1 becomes important. Note that the entire for-loop (lines 5-15, Algorithm 2) is embarrassingly parallel and can be vectorized. To achieve this, unconstrained solutions for all the three possible active sets are computed before entering the for-loop. Some computation is redundant, for example, the cost of solving $u_i$ and $v_i$ is wasted when $\mathbf{g}_i^{\emptyset} \geq 0$ (c.f. line 5-6, Algorithm 4). However, Algorithm 5 represents

**Algorithm 5** Algorithm for solving $\min_{G \geq 0} \|BG - Y\|_F^2$, where $B = [\mathbf{b}_1, \mathbf{b}_2] \in \mathbb{R}_+^{m \times 2}, Y \in \mathbb{R}_+^{m \times n}$

```
1: Solve unconstrained least squares $G^\emptyset = [\mathbf{g}_1^\emptyset, \cdots, \mathbf{g}_n^\emptyset] \leftarrow \min \|BG - Y\|^2$ by normal
   equation $B^T BG = B^T Y$
2: $\beta_1 \leftarrow \|\mathbf{b}_1\|$, $\beta_2 \leftarrow \|\mathbf{b}_2\|$
3: $\mathbf{u} \leftarrow (Y^T \mathbf{b}_1)/\beta_1^2$
4: $\mathbf{v} \leftarrow (Y^T \mathbf{b}_2)/\beta_2^2$
5: for $i = 1$ to $n$ do
6:    if $\mathbf{g}_i^\emptyset \geq 0$ then
7:       return $\mathbf{g}_i^\emptyset$
8:    else
9:       if $u_i \beta_1 \geq v_i \beta_2$ then
10:         return $[u_i, 0]^T$
11:      else
12:         return $[0, v_i]^T$
13:      end if
14:   end if
15: end for
```

a non-random pattern of memory access, and we expect that it is much faster for rank-2 NMF than applying existing active-set-type algorithms directly.

Note that a naïve implementation of comparing $\|\mathbf{b}_1 g_1^* - \mathbf{y}\|^2$ and $\|\mathbf{b}_2 g_2^* - \mathbf{y}\|^2$ for $n$ different vectors $\mathbf{y}$ requires $O(mn)$ complexity due to the creation of the $m \times n$ dense matrix $BG - Y$. In contrast, our algorithm only requires $O(m+n)$ complexity at this step (line 9, Algorithm 5), because $\mathbf{b}_1, \mathbf{b}_2$ are the same across all the $n$ right-hand sides. Assuming $Y$ is a sparse matrix with $N$ nonzeros, the overall complexity of Algorithm 5 is $O(N)$, which is the same as the complexity of existing active-set-type algorithms when $k = 2$ (see Eq. 51). The dominant part comes from computing the matrix product $Y^T B$ in unconstrained least squares.

## *5.4 Hierarchical Document Clustering Based on Rank-2 NMF*

Rank-2 NMF can be recursively applied to a data set, generating a hierarchical tree structure. In this section, we focus on text corpus and develop an overall efficient approach to hierarchical document clustering. In particular, we propose a strategy of

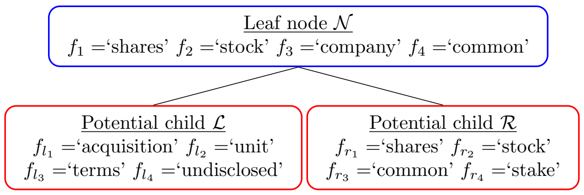

Leaf node $\mathcal{N}$
$f_1$ =‘shares’ $f_2$ =‘stock’ $f_3$ =‘company’ $f_4$ =‘common’
Potential child $\mathcal{L}$
$f_{l_1}$ =‘acquisition’ $f_{l_2}$ =‘unit’
$f_{l_3}$ =‘terms’ $f_{l_4}$ =‘undisclosed’
Potential child $\mathcal{R}$
$f_{r_1}$ =‘shares’ $f_{r_2}$ =‘stock’
$f_{r_3}$ =‘common’ $f_{r_4}$ =‘stake’

Figure 18: An illustration of a leaf node $\mathcal{N}$ and its two potential children $\mathcal{L}$ and $\mathcal{R}$.

choosing an existing leaf node at each splitting step. Extensive criteria for selecting the next leaf node to split were discussed in previous literature for general clustering methods [27], mainly relying on cluster labels induced by the current tree structure. In the context of NMF, however, we have additional information about the clusters: Each column of $W$ is a cluster representative. In text data, a column of $W$ is the term distribution for a topic [110], and the largest elements in the column correspond to the *top words* for this topic. We will exploit this information to determine the next node to split.

In summary, our strategy is to compute a score for each leaf node by running rank-2 NMF on this node and evaluating the two columns of $W$. Then we select the current leaf node with the highest score as the next node to split. The score for each node needs only to be computed once when the node first appears in the tree. For an illustration of a leaf node and its two potential children, see Fig. 18. We split a leaf node $\mathcal{N}$ if at least two well-separated topics can be discovered within the node. Thus we expect that $\mathcal{N}$ receives a high score if the top words for $\mathcal{N}$ is a well-balanced combination of the top words for its two potential children, $\mathcal{L}$ and $\mathcal{R}$. We also expect that $\mathcal{N}$ receives a low score if the top words for $\mathcal{L}$ and $\mathcal{R}$ are almost the same.

We utilize the concept of normalized discounted cumulative gain (NDCG) [46] from the information retrieval community. Given a perfect ranked list, NDCG measures the quality of an actual ranked list which always has value between 0 and 1. A leaf node $\mathcal{N}$ in our tree is associated with a term distribution $w_{\mathcal{N}}$, given by a column of $W$ from the rank-2 NMF run that generates the node $\mathcal{N}$. We can obtain a ranked list of terms for $\mathcal{N}$ by sorting the elements in $w_{\mathcal{N}}$ in descending order, denoted by $f_{\mathcal{N}}$. Similarly, we can obtain ranked lists of terms for its two potential children, $\mathcal{L}$ and $\mathcal{R}$, denoted by $f_{\mathcal{L}}$ and $f_{\mathcal{R}}$. Assuming $f_{\mathcal{N}}$ is a perfect ranked list, we compute a modified NDCG (mNDCG) score for each of $f_{\mathcal{L}}$ and $f_{\mathcal{R}}$. We describe our way to compute mNDCG in the following. Recall that $m$ is the total number of terms in the vocabulary. Suppose the perfectly ordered terms corresponding to $f_{\mathcal{N}}$ is

$$f_1, f_2, \cdots, f_m,$$

and the shuffled orderings in $f_{\mathcal{L}}$ and $f_{\mathcal{R}}$ are respectively:

$$f_{l_1}, f_{l_2}, \cdots, f_{l_m};$$
$$f_{r_1}, f_{r_2}, \cdots, f_{r_m}.$$

We first define a *position discount factor* $p(f_i)$ and a *gain* $g(f_i)$ for each term $f_i$:

$$\begin{aligned} p(f_i) &= \log\left(m - \max\{i_1, i_2\} + 1\right), && (58)\\ g(f_i) &= \frac{\log(m - i + 1)}{p(f_i)}, && (59) \end{aligned}$$

where $l_{i_1} = r_{i_2} = i$. In other words, for each term $f_i$, we find its positions $i_1, i_2$ in the two shuffled orderings, and place a large discount in the gain of term $f_i$ if this term is high-ranked in both shuffled orderings. The sequence of gain $\{g(f_i)\}_{i=1}^m$ is sorted in descending order, resulting in another sequence $\{\hat{g}_i\}_{i=1}^m$. Then, for a shuffled ordering $f_{\mathcal{S}}$ ($f_{\mathcal{S}} = f_{\mathcal{L}}$ or $f_{\mathcal{R}}$), mNDCG is defined as:

$$\begin{aligned} \text{mDCG}(f_{\mathcal{S}}) &= g(f_{s_1}) + \sum_{i=2}^{m} \frac{g(f_{s_i})}{\log_2(i)}, && (60)\\ \text{mIDCG} &= \hat{g}_1 + \sum_{i=2}^{m} \frac{\hat{g}_i}{\log_2(i)}, && (61)\\ \text{mNDCG}(f_{\mathcal{S}}) &= \frac{\text{mDCG}(f_{\mathcal{S}})}{\text{mIDCG}}. && (62) \end{aligned}$$

As we can see, mNDCG is basically computed in the same way as the standard NDCG measure, but with a modified gain function. Also note that $\hat{g}_i$ instead of $g(f_i)$ is used in computing the ideal mDCG (mIDCG) so that mNDCG always has a value in the $[0, 1]$ interval.

Finally, the score of the leaf node $\mathcal{N}$ is computed as:

$$\text{score}(\mathcal{N}) = \text{mNDCG}(f_{\mathcal{L}}) \times \text{mNDCG}(f_{\mathcal{R}}). \tag{63}$$

To illustrate the effectiveness of this scoring function, let us consider some typical cases.

1. When the two potential children $\mathcal{L}, \mathcal{R}$ describe well-separated topics, a top word for $\mathcal{N}$ is high-ranked in one of the two shuffled orderings $f_{\mathcal{L}}, f_{\mathcal{R}}$, and low-ranked in the other. Thus the top words will not suffer from a large discount, and both mNDCG($f_{\mathcal{L}}$) and mNDCG($f_{\mathcal{R}}$) will be large.

2. When both $\mathcal{L}$ and $\mathcal{R}$ describe the same topic as that of $\mathcal{N}$, a top word for $\mathcal{N}$ is high-ranked in both the shuffled orderings. Thus the top words will get a large discount, and both mNDCG($f_{\mathcal{L}}$) and mNDCG($f_{\mathcal{R}}$) will be small.

3. When $\mathcal{L}$ describes the same topic as that of $\mathcal{N}$, and $\mathcal{R}$ describes a totally unrelated topic (e.g. outliers in $\mathcal{N}$), then mNDCG($f_{\mathcal{L}}$) is large and mNDCG($f_{\mathcal{R}}$) is small, and score($\mathcal{N}$) is small.

The overall hierarchical document clustering workflow is summarized in Algorithm 6, where we refer to a node and the documents associated with the node exchangably. The while-loop in this workflow (lines 8-15) defines an outlier detection procedure, where $T$ trials of rank-2 NMF are allowed in order to split a leaf node $\mathcal{M}$ into two well-separated clusters. At each trial, two potential children nodes $\mathcal{N}_1, \mathcal{N}_2$ are created, and if we believe that one of them (say, $\mathcal{N}_2$) is composed of outliers, we discard $\mathcal{N}_2$ from $\mathcal{M}$ at the next trial. If we still cannot split $\mathcal{M}$ into two well-separated clusters

**Algorithm 6** Hierarchical document clustering based on rank-2 NMF

```
1: Input: A term-document matrix $X \in \mathbb{R}_+^{m \times n}$ (often sparse), maximum number of
   leaf nodes $k$, parameter $\beta > 1$ and $T \in \mathbb{N}$ for outlier detection
2: Create a root node $\mathcal{R}$, containing all the $n$ documents
3: score($\mathcal{R}$) $\leftarrow \infty$
4: repeat
5:     $\mathcal{M} \leftarrow$ a current leaf node with the highest score
6:     Trial index $i \leftarrow 0$
7:     Outlier set $Z \leftarrow \emptyset$
8:     while $i < T$ do
9:         Run rank-2 NMF on $\mathcal{M}$ and create two potential children $\mathcal{N}_1, \mathcal{N}_2$, where
           $|\mathcal{N}_1| \geq |\mathcal{N}_2|$
10:        if $|\mathcal{N}_1| \geq \beta|\mathcal{N}_2|$ and score($\mathcal{N}_2$) is smaller than every positive score of current
           leaf nodes then
11:            $Z \leftarrow Z \cup \mathcal{N}_2$, $\mathcal{M} \leftarrow \mathcal{M} - Z$, $i \leftarrow i + 1$
12:        else
13:            break
14:        end if
15:    end while
16:    if $i < T$ then
17:        Split $\mathcal{M}$ into $\mathcal{N}_1$ and $\mathcal{N}_2$ (hard clustering)
18:        Compute score($\mathcal{N}_1$) and score($\mathcal{N}_2$)
19:    else
20:        $\mathcal{M} \leftarrow \mathcal{M} \cup Z$ (recycle the outliers and do not split $\mathcal{M}$)
21:        score($M$) $\leftarrow -1$ (set $\mathcal{M}$ as a permanent leaf node)
22:    end if
23: until # leaf nodes = $k$
24: Output: A binary tree structure of documents, where each node has a ranked list
    of terms
```

after $T$ trials, $\mathcal{M}$ is marked as a permanent leaf node. Empirically, without the outlier detection procedure, the constructed tree would end up with many tiny leaf nodes, which do not correspond to salient topics and degrade the clustering quality. We have not specified the best moment to exit and stop the recursive splitting process, but simply set an upper limit of leaf nodes $k$. Other strategies can be used to determine when to exit, such as specifying a score threshold $\sigma$ and exiting the program when none of the leaf nodes have scores above $\sigma$; $\sigma = 0$ means that the recursive splitting process is not finished until all the leaf nodes become permanent leaf nodes.

Compared to other criteria for choosing the next node to split, such as those relying on the self-similarity of each cluster and incurring $O(n^2)$ overhead [27], our method is more efficient. In practice, the binary tree structure that results from Algorithm 6 often has meaningful hierarchies and leaf clusters. We will evaluate its performance by clustering quality measures in the experiment section.

## 5.5 Related Work

NMF can be regarded as both a clustering method and, with certain additional constraints, a probabilistic topic modeling method [3]. Both document clustering and topic modeling can be thought of as some sort of dimension reduction; however, these two tasks have fundamental differences. This chapter is focused on the clustering aspect, and now we compare NMF-based clustering to a popular topic modeling method, latent Dirichlet allocation (LDA) [15].

We start with comparing flat clustering and flat topic modeling. First, LDA builds a probabilistic model that generates the text corpus, and intends to predict the probability of new documents, while the goal of NMF-based clustering is to derive a partitioning that well-organizes an existing text corpus. Second, LDA models each word as a discrete random variable, thus is not compatible with tf-idf weighting when forming the term-document matrix. On the contrary, NMF-based clustering finds an algebraic latent subspace and is able to leverage the benefit of tf-idf weighting which has proved to be useful in a wide range of tasks such as information retrieval [77].

Now we discuss the difference between hierarchical clustering based on rank-2 NMF and hierarchical LDA (hLDA) [13]. hLDA builds a hierarchy of topics and each document is generated by sampling from the topics along a path with length $L$ from the root to a leaf node. On the contrary, hierarchical clustering builds a hierarchy of documents, and the documents associated with each node are a mixture of two topics extracted from this node. In practice, hLDA requires all the leaf nodes be on the same level of the tree, and the depth $L$ of the tree is chosen beforehand, while hierarchical clustering adaptively chooses a node at each splitting step.

In general, both methods have pros and cons: Topic modeling has a probabilistic interpretation, and clustering approaches are more flexible. In the next section, we include LDA in our experiments and compare its performance with the performances of clustering based on NMF and rank-2 NMF. hLDA or other hierarchical probabilistic models are not considered in the experiments because they represent a quite different scenario of text modeling.

## *5.6 Experiments*

In this section, we describe our experimental settings and demonstrate both the efficiency and quality of our proposed algorithm. All the experiments except LDA are run in Matlab 7.9 (R2009b) with two Intel Xeon X5550 quad-core processors and 24GB memory.

### 5.6.1 Data Sets

Four text data sets with ground-truth classes are used in our experiments: 1. **Reuters-21578**[1] contains news articles from the Reuters newswire in 1987. We discarded documents with multiple class labels, and then selected the 20 largest classes. 2. **20 Newsgroups**[2] contains articles from Usenet newsgroups and has a defined hierarchy of 3 levels. Unlike previous indexing, we observed that many articles have duplicated paragraphs due to cross-referencing. We discarded cited paragraphs and signatures. 3. **Cora** [80] is a collection of research papers in computer science, from which we extracted the title, abstract, and reference-contexts. Although this data set defines a topic hierarchy of 3 levels, we observed that some topics, such as "AI – NLP" and

[1]`http://www.daviddlewis.com/resources/testcollections/reuters21578/` (retrieved in June 2014)

[2]`http://qwone.com/~jason/20Newsgroups/` (retrieved in June 2014)

Table 16: Data sets used in our experiments.

| Data sets | Has label | Has hierarchy | Size | # terms | # docs | # nodes at each level |
|---|---|---|---|---|---|---|
| Reuters-21578 | Y | N | medium | 12,411 | 7,984 | 20 |
| 20 Newsgroups | Y | Y | medium | 36,568 | 18,221 | 6/18/20 |
| Cora | Y | N | large | 154,134 | 29,169 | 70 |
| RCV1-labeled | Y | Y | large | 115,679 | 496,756 | 4/15/28/39/40 |
| RCV1-full | N | - | large | 149,113 | 764,751 | - |

Table 17: Timing results of NMF-based clustering.

| Data sets | `r2-nmf-hier` | `nmf-hier` | `nmf-flat` |
|---|---|---|---|
| Reuters-21578 | 4.34 sec | 12.8 sec | 51.2 sec |
| 20 Newsgroups | 18.8 sec | 58.4 sec | 289 sec |
| Cora | 144 sec | 335 sec | 0.92 hrs |
| RCV1-labeled | 213 sec | 384 sec | 1.11 hrs |
| RCV1-full | 419 sec | 702 sec | 4.48 hrs |

"IR – Extraction", are very related but reside in different subtrees. Thus we ignored the hierarchy and obtained 70 ground-truth classes as a flat partitioning. 4. **RCV1** [68] is a much larger collection of news articles from Reuters. It contains over 800,000 articles in the time period of 1996-1997 and defines a sophisticated topic hierarchy with 103 labels. We discarded documents with multiple class labels, and then selected the 40 largest classes, named as **RCV1-labeled**. The full data set, named as **RCV1-full**, is also included in our experiments with no ground-truth classes.

We summarize these data sets in Table 16. All the data sets except 20 Newsgroups have very unbalanced sizes of ground-truth classes. We constructed the normalized-cut weighted version of term-document matrices as in [110].

### 5.6.2 Methods for Comparison

The clustering methods in our experiments are named as follows[3]:

[3]We also compared our method with the off-the-shelf clustering software CLUTO [49]. In most cases, our method is faster than CLUTO configured by default, with comparable clustering quality. When both methods are terminated after 10 iterations, our method costs 104 seconds on RCV1-full data set while CLUTO costs 398 seconds.

- `r2-nmf-hier`: Hierarchical clustering based on rank-2 NMF with the algorithm proposed in this chapter.

- `nmf-hier`: Hierarchical clustering based on standard NMF with active-set based algorithm [53]. In our experiments, multiplicative update rule algorithms [66] for standard NMF are always slower and give similar quality compared to active-set-type algorithms, thus are not included in our results.

- `nmf-flat`: Flat clustering based on standard NMF with block-pivoting based algorithm [56].

- `kmeans-hier`: Hierarchical clustering based on standard K-means. We use the hierarchical clustering workflow described in Algorithm 6; however, the term distribution associated with each node is given by the centroid vector from the K-means run that generates this node.

- `kmeans-flat`: Flat clustering based on standard K-means.

- `lda`: Flat clustering using the Gibbs sampling algorithm for LDA. We use a highly-optimized implementation in the software MALLET[4] written in Java. 1000 iterations are used by default. LDA is not run for RCV1-labeled and RCV1-full due to efficiency reasons.

Hierarchical clustering and flat clustering cannot be compared against each other directly. We evaluate the hierarchy by taking snapshots of the tree as leaf nodes are generated, and because leaf nodes are non-overlapping, we treat all the leaf nodes in each snapshot as a flat partitioning. Thus, if the maximum number of leaf nodes is $c$, we produce $c-1$ flat partitionings forming a hierarchy.

For each method, we perform 20 runs on medium-scale data sets and 5 runs on large-scale data sets starting from random initializations. Average measurements are

[4]`http://mallet.cs.umass.edu/` (retrieved in June 2014)

reported. Note that for flat clustering methods, each run consists of $c-1$ separate executions with the number of clusters set to $2, 3, \cdots, c$.

The maximum number of leaf nodes $c$ is set to be the number of ground-truth classes at the deepest level for labeled data sets (see Table 16); and we set $c = 60$ for RCV1-full. The hierarchical clustering workflow (Algorithm 6) runs with parameters $\beta = 9$, $T = 3$ (for `r2-nmf-hier` and `nmf-hier`) or 5 (for `kmeans-hier`). The Matlab `kmeans` function has a batch-update phase and a more time-consuming online-update phase. We rewrote this function using BLAS3 operations and boosted its efficiency substantially[5]. We use both phases for medium-scale data sets and only the batch-update phase for large-scale data sets. For NMF, we use the projected gradient as the stopping criterion and $\epsilon = 10^{-4}$ where a tolerance parameter $\epsilon$ is defined in [71]. All the methods are implemented with multi-threading.

### 5.6.3 Evaluation Measures

Each of the six methods described above can be regarded as both a clustering method and a topic modeling method. We use the following two measures to evaluate their quality:

1. *Normalized mutual information (NMI)*: This is a measure of the similarity between two flat partitionings. It is used to evaluate clustering quality and is only applicable to data sets with ground-truth classes. It is particularly useful when the number of generated clusters is different from that of ground-truth classes and can be used to determine the optimal number of clusters. More details can be found in [77]. For data sets with defined hierarchy, we compute NMI between a generated partitioning and the ground-truth classes at each level of the tree; if the tree has depth $L$, then we compute $L$ measures corresponding to each level. For hierarchical clustering following Algorithm 6, we treat all the outliers as one separate cluster for

[5]`http://www.cc.gatech.edu/~dkuang3/software/kmeans3.html` (retrieved in June 2014)

fair evaluation.

2. *Coherence*: This is a measure of intra-topic similarity in topic models [81, 3]. Given the top words $f_1, \cdots, f_K$ for a topic, coherence is computed as

$$\text{coherence} = \sum_{i=1}^{K} \sum_{j=i}^{K} \left( \log \frac{D(f_i, f_j) + \mu}{D(f_i)} \right), \tag{64}$$

where $D(f_i)$ is the document frequency of $f_i$. $D(f_i, f_j)$ is the number of documents that contain both $f_1$ and $f_2$, and $\mu$ is a smoothing parameter. We use $\mu = 1$ and $K = 20$ [81]. The coherence averaged over all the topics is reported.

### 5.6.4 Timing Results

Timing results of the six methods are shown in Fig. 19. Hierarchical clustering based on rank-2 NMF is much faster than flat clustering using NMF or LDA. These results have verified our complexity analysis in Section 5.2, that flat clustering based on standard NMF exhibits a superlinear trend while hierarchical clustering based on rank-2 NMF exhibits a linear trend of running time as $k$ increases. The first two plots correspond to medium-scale data sets, and `r2-nmf-hier` only requires about 1/3 the time needed by `nmf-hier`. The other three plots correspond to large-scale data sets, where we use logarithmic scale. K-means with only the batch-update phase also runs fast; however, their clustering quality is not as good, which will be shown later.

The difference between our proposed algorithm and the original active-set based algorithm for rank-2 NMF is less substantial as the data size increases. The performance is mainly bounded by the computation of $Y^T B$ in Algorithm 5. Because $B \in \mathbb{R}_+^{m \times 2}$ is a very long-and-thin matrix, $Y^T B$ essentially behaves like a sparse matrix-vector multiplication, which is a memory-bound operation. However, `r2-nmf-hier` is still much faster than all the other methods: On RCV1-full data set, `r2-nmf-hier`, `nmf-hier`, and `nmf-flat` cost about 7 minutes, 12 minutes, and 4.5 hours, respectively.

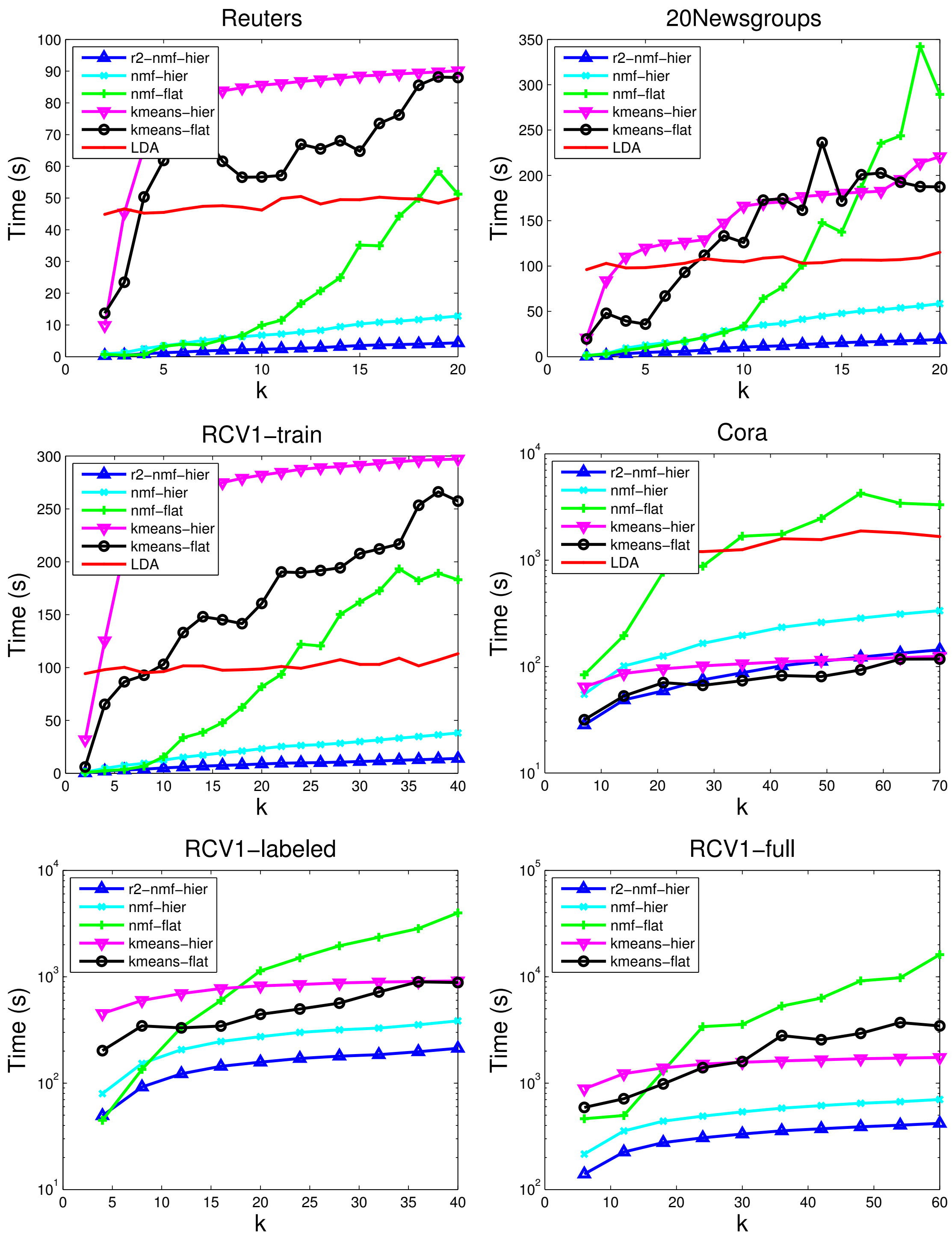


Figure 19: Timing results in seconds.

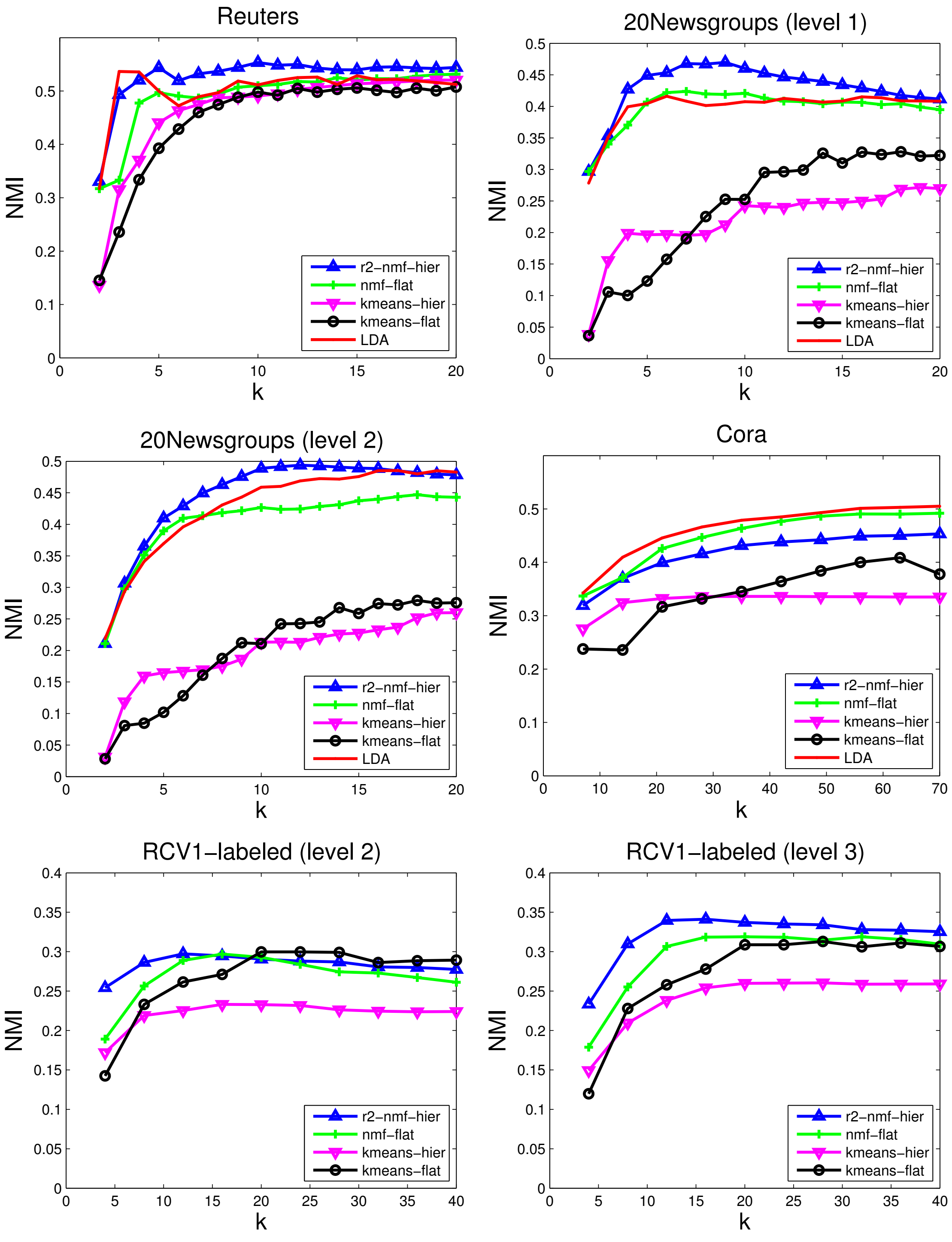


Figure 20: NMI on labeled data sets. Scales of y-axis for the same data set are set equal.

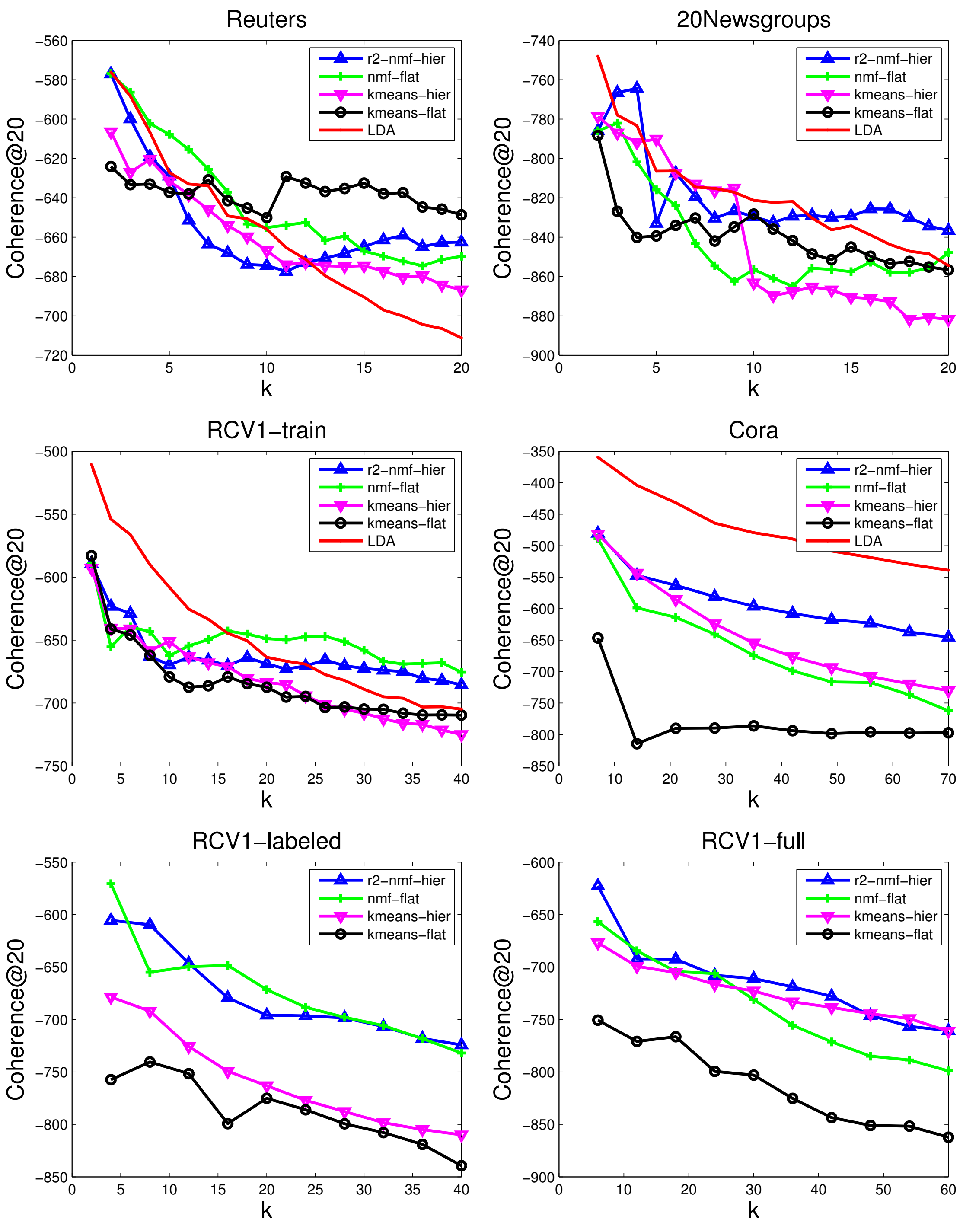


Figure 21: Coherence using the top 20 words for each topic.

### 5.6.5 Clustering Quality

Clustering quality is evaluated on labeled data sets, shown in Fig. 20. The plot for the Cora data set is omitted for space reasons. For data sets with a defined hierarchy, ground-truth classes on the first 3 levels are used for evaluation, and those on deeper levels produce similar results. `nmf-hier` has identical results with `r2-nmf-hier`, thus is not shown here.

`r2-nmf-hier` is a very competitive method in general. NMF-based methods give stably good clustering quality using both the flat and hierarchical schemes. Compared to `nmf-flat`, we can clearly see the improved NMI values of `r2-nmf-hier`. Although `kmeans-flat` achieves comparable performances on RCV1-labeled, it performs poorly on other data sets. A general trend is that the improvement in clustering quality by `r2-nmf-hier` is more substantial when a deeper level of the defined hierarchy is used for evaluation, which correspond to more elaborated ground-truth classes.

We note that if NMI values are used for selecting the best number of clusters for a data set, `r2-nmf-hier` and `nmf-flat` frequently give different numbers (see the last two plots in Fig. 20). Thus they tend to interpret a data set in different ways. We also note that although `kmeans-hier` uses the same hierarchical clustering workflow as `r2-nmf-hier`, it performs poorly in most cases.

### 5.6.6 Semantic Quality of Topics

The coherence results for all the data sets are shown in Fig. 21. None of these methods have consistent performances when the number of clusters $k$ is small; when $k$ is large, `r2-nmf-hier` gives the highest coherence value in 3 out of 5 cases. On RCV1 data set, `r2-nmf-hier` is a stably good method in terms of topic coherence, while `nmf-flat` and `kmeans-hier` have comparable performances sometimes but perform very poorly otherwise. More study is needed to understand the benefits of each method in terms of topic coherence.

## 5.7 *Discussion*

Hierarchical document clustering has a rich history in data analysis and management [107]. In this chapter, we considered the divisive approach, which splits a data set in the top-down fashion and offers a global view of the data set compared to agglomerative clustering methods. In divisive hierarchical clustering, a clustering method is needed at each splitting step. However, it is not as easy as recursively applying any flat clustering method available to generate a tree structure. As can be seen in our experiments, the widely-used K-means clustering, when applied to hierarchical clustering, frequently generates very unbalanced clusters that lead to a poor organization of a corpus.

A good combination of a flat clustering method and a way to determine the next node to split is important for efficient and practical hierarchical clustering. In this chapter, we proposed such a combination and showed its promising performance compared to other clustering methods such as NMF and LDA. For the efficiency of each splitting step, we designed a fast active-set-type algorithm for rank-2 NMF. Our algorithm has redundant computation but has continuous memory access, allowing better use of the cache; thus, it is faster than existing active-set-type algorithms. We also proposed a scoring method in the hierarchical clustering workflow, which provides a way to evaluate the potential of each leaf node to be split into two well-separated clusters and can be used to determine when to stop splitting. Outlier detection is also included in the overall workflow. Our method generated a binary tree structure of the full RCV1 data set in 7 minutes on a shared-memory machine with 2 quad-core CPUs, compared to standard NMF which costs 4.5 hours.

We conclude by listing several shortcomings of the current method for further research. First, after a node is split, each document has a hard assignment to one of the two generated leaf nodes. It would be more flexible to enable soft assignments. Second, the performance of our proposed algorithm for rank-2 NMF is bounded by that of sparse matrix-vector multiplication (SpMV) when the data size is very large. The efficiency of our algorithm can be further boosted by using a more efficient SpMV implementation or moving to a distributed platform. Currently, our method can be used to build a hierarchical organization of documents efficiently on a single machine, possibly as part of a large machine learning infrastructure with many machines.

# CHAPTER VI

# NMF FOR LARGE-SCALE TOPIC MODELING

## 6.1 NMF-Based Clustering for Topic Modeling

Nowadays text data have overwhelming volumes and are ubiquitous thanks to the explosion of the Internet. Long articles in user-contributed encyclopedia and short text snippets such as tweets are two examples: The current English Wikipedia contains about 4.5 million articles[1]; Twitter users worldwide generate over 400 million tweets every single day. Useful information can be extracted from these online texts. For example, decision makers and researchers interested in the area of sustainability would learn how energy technology and policies receive public attention and affect daily lives from tweets. Analyzing the huge and increasing volume of online text data efficiently has become an important data analytics problem.

We focus on unsupervised methods for analyzing text data in this chapter. Many online texts have no label information; other documents such as Wikipedia articles are often tagged with multiple labels from a user-generated taxonomy thus do not fit into traditional supervised learning framework well. Therefore, unsupervised clustering and topic modeling methods have become important tools for browsing and organizing a large text collection [14]. The goal of these unsupervised methods is to find a number of document clusters, say $k$ clusters, where each cluster contains semantically connected documents and forms a coherent topic. Among those, latent Dirichlet allocation (LDA) is a prominent and widely-used method so far [15]. The key idea in LDA is that each document is modeled as a mixture of $k$ topics and each topic is represented by a distribution over words. An algorithm for LDA searches for the

[1] `http://en.wikipedia.org/wiki/Wikipedia:Size_of_Wikipedia` (retrieved in March 2014)

$k$ topics and the topic mixtures that maximize the likelihood of a given document collection. However, most algorithms for LDA are expensive and cannot scale to millions of documents such as those on Wikipedia or Twitter [109].

Though topic modeling methods such as LDA are based on probabilistic models, they can be explained in a matrix approximation framework. Let $\mathbb{R}_+$ denote the set of nonnegative real numbers. In clustering and topic modeling, text data are commonly represented as a term-document matrix $X \in \mathbb{R}_+^{m \times n}$ [77]. The $m$ rows of $X$ correspond to a vocabulary of $m$ terms, and the $n$ columns correspond to $n$ documents. Consider a factorization of $X$:

$$X = WH, \tag{65}$$

where $W \in \mathbb{R}_+^{m \times k}$ and $H \in \mathbb{R}_+^{k \times n}$. Note that $k$ is the number of topics we want to find, and $k << n$. Eq. (65) means that each of the $n$ documents, say the $i$-th document $\mathbf{a}_i$, can be reconstructed by a linear combination of the $k$ columns of $W$, where the $k$ linear coefficients are contained in the $i$-th column of $H$. Suppose the given matrix $X$ further satisfies $\|\mathbf{x}_i\|_1 = 1$ $(1 \le i \le n)$, i.e. $\mathbf{x}_i$ represents the distribution over words in the $i$-th document. If we normalize $W$ so that each column of $W$ sums to one, then clearly each column of $H$ would sum to one. In this case, we can interpret the columns of $W$ as distributions over words for the $k$ topics and the columns of $H$ as distributions over the $k$ topics. To obtain a hard clustering result for the $i$-th document, we can select the topic corresponding to the largest entry in the $i$-th column of $H$. This way, clustering and topic modeling can be unified in the same model where $W$ shows the topics and $H$ shows the clusters.

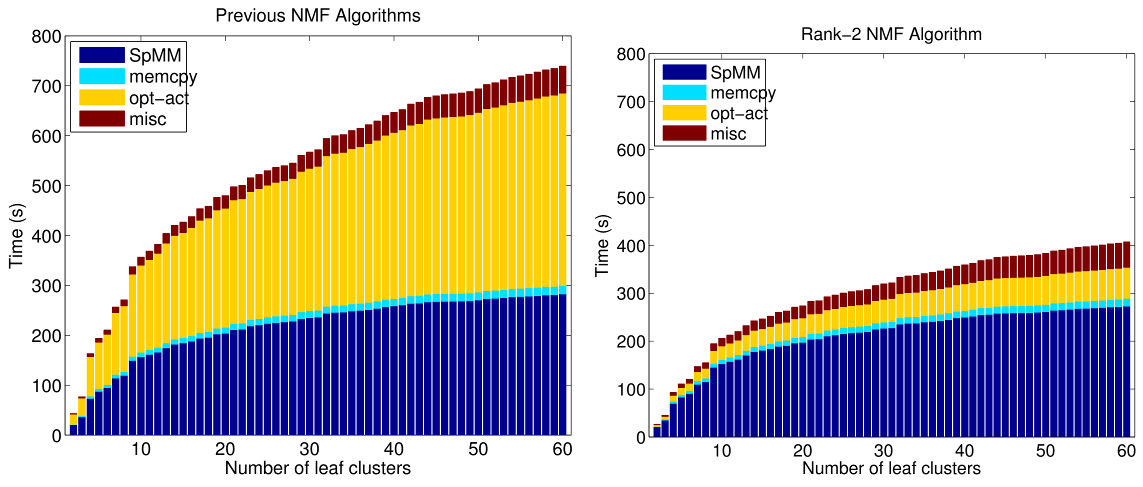
In reality, $X = WH$ can only be approximately satisfied, that is,

$$X \approx WH. \tag{66}$$

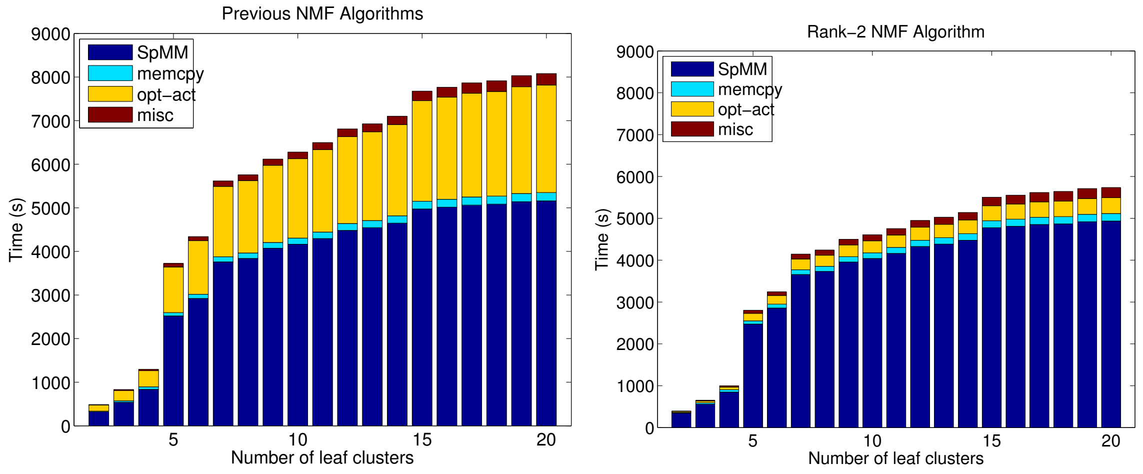
Though (66) loses probabilistic interpretation of the generative process of topics and documents, many studies have shown that it is also very useful for clustering and topic modeling [4, 3, 62]. Note that $X, W, H$ are all nonnegative matrices and (66) is the formulation of NMF. We typically normalize both the columns of $A$ and $W$ and scale the rows of $H$ accordingly in the case of text data. When each column of $W$ is interpreted as a vector of "importance" values for the $m$ terms and viewed as a topic, each column of $H$ can be interpreted as topic mixture coefficients that approximately reconstruct the corresponding document though it does not sum to one. Therefore, in the framework of NMF, clustering and topic modeling can be viewed as two different names of the same analytics problem.

NMF as a topic modeling method has several advantages over LDA. First, without the need for a probabilistic interpretation, we can provide a term-document matrix with *tf-idf* weighting as an input to NMF instead of raw frequencies of word occurrences, just as in most text classification methods [77]. Tf-idf weighting has been widely shown to improve classification or clustering accuracy. Second, numerous algorithms have been proposed to efficiently compute the solutions of NMF [71, 53, 57, 21], making it desirable to apply NMF for web-scale clustering and topic modeling. The hierarchical clustering algorithm based on rank-2 NMF proposed in Chapter 5, which we call *HierNMF2*, is much more efficient than previous NMF-based clustering methods, generating 60 leaf nodes on the 800K RCV1 data set [68] in minutes. More importantly, HierNMF2 proceeds by carefully determining which node in the hierarchy should be split further and which should not, and its clustering quality often retains or even exceeds that of the standard NMF.

However, as we discussed in Section 5.7, one major bottleneck of HierNMF2, which is also a common bottleneck in many other machine learning applications, is the multiplication a large sparse matrix with a tall-and-skinny dense matrix (SpMM). Fig. 22 shows the timing of various steps of the entire workflow of HierNMF2 on the 800K RCV1 and 4.5M Wikipedia articles data sets. SpMM costs 67% and 86% of the total runtime for these two data sets respectively.



(a) 800K RCV1 data set



(b) 4.5M Wikipedia data set

Figure 22: Timing of the major algorithmic steps in NMF-based hierarchical clustering shown in different colors. The legends are: **SpMM** – Sparse-dense matrix multiplication, where the dense matrix has two columns; **memcpy** – Memory copy for extracting a submatrix of the term-document matrix for each node in the hierarchy; **opt-act** – Searching for the optimal active set in active-set-type algorithms (refer to Section 5.2); **misc** – Other algorithmic steps altogether. "Previous NMF algorithms" refer to active-set based algorithms [53, 56, 57]. The Rank-2 NMF algorithm greatly reduced the cost of *opt-act*, leaving SpMM as the major bottleneck.

In data analytics where a sparse data matrix is involved as in term-document matrices, user-by-item rating matrices, and graph adjacency matrices, it often has an irregular sparsity structure and is most commonly stored in a generic format such as the Compressed Sparse Row format. However, existing numerical libraries that implement SpMM are often tuned towards other applications such as structural mechanics, and thus cannot exploit the full computing capability for machine learning applications. In this chapter, we accelerate SpMM on the graphics processing units (GPU), especially for a dense matrix with two columns for further increasing the efficiency of HierNMF2, and propose a cache blocking strategy that can take advantage of memory locality and increase memory throughput. The sparse matrix such as a term-document matrix representing large-scale text data is often larger than the global memory of GPUs. Our routine is an *out-of-core* implementation and can stream the sparse matrix onto the GPUs for computation. To the best of our knowledge, this is the first out-of-core SpMM routine on the GPUs for sparse matrices with an irregular sparsity structure. Therefore, we expect that our work has a broader impact on the efficiency of many data analytics problems as the sizes of sparse data matrices are steadily growing.

In addition, the hierarchy generated by HierNMF2 does not fit well into a flat topic modeling [15] or hierarchical topic modeling [13] framework. We will introduce our way to flatten a hierarchy of clusters in order to produce both hierarchical clusters as well as flat clusters and topics. We will show that our overall framework for flat clustering and topic modeling is orders of magnitude faster than existing methods such as the standard NMF and LDA.

The rest of this chapter is organized as follows. In Section 6.2, we conduct an overview of the SpMM routine in machine learning applications. We introduce the challenges in achieving high throughput of the SpMM kernel on GPUs in Section 6.3. We present a performance model to analyze the SpMM kernel in Section 6.4. We describe our implementation of SpMM in Section 6.5. We show our benchmarking results on a variety of sparse matrices originating from machine learning applications in Section 6.6. Finally, we incorporate our SpMM routine into a C++ software for HierNMF2 and run large-scale topic modeling experiments in Section 6.7.

## *6.2 SpMM in Machine Learning Applications*

Sparse matrix is a common form of data representation in machine learning. In text analysis, a collection of documents can be represented as a term-document matrix where rows correspond to words, columns correspond to documents, and each entry is the raw or weighted frequency of a word in a document. In a movie recommendation problem, we often use a sparse matrix to represent the movie ratings where rows correspond to users, columns correspond to movies, and each nonzero entry is one observed rating. In graph analytics, the adjacency matrix that represent the graph edges between nodes is often sparse in a real-world setting such as social networks. These matrices are all inherently sparse, and a dedicated sparse format uses considerably less space in storing them and is computationally more efficient than storing them as an ordinary dense matrix.

In numerical computing and machine learning problems that involve a sparse data matrix, a common operation is to multiply the sparse matrix $A$ with a dense vector $x$ (SpMV), or with a tall-and-skinny dense matrix $X$ (SpMM). For example, SpMV is a building block and the most expensive step in iterative eigensolvers [6]; SpMM is accountable for the majority of computation in alternating least squares algorithms for collaborative filtering [44] and alternating nonnegative least squares algorithms for nonnegative matrix factorization [55]. Computing the pairwise Euclidean distances between a large set of sparse vectors and a small set of dense vectors, such as centroids in K-means clustering, can also be reduced to an SpMM operation [83]. Recent graph mining frameworks such as PEGASUS [48], GraphChi [63], and TurboGraph [39] rely on the SpMV routine for many tasks such as finding connected components and modules. As the size of data sets becomes increasingly larger nowadays, SpMV/SpMM is often a bottleneck in machine learning applications, and it is important to have an efficient implementation of SpMV/SpMM for the overall efficiency of an algorithm.

Following the convention in the high-performance computing community, we will call SpMV/SpMM as *kernels*. There was plenty of research in accelerating these kernels, especially for SpMV [103, 108, 104, 72]. However, most of previous research was focused on their use in other science and engineering disciplines that require solving large sparse linear systems, for example, in finite element methods in structural mechanics. The sparse matrices in those disciplines often have special sparsity structures such as small dense blocks in finite element methods. However, the sparse matrices that represent data in data-intensive applications often carry different characteristics: Their sparsity structures are often irregular, and therefore the kernels presented in previous research are difficult to achieve the highest possible throughput on such matrices [20]. Existing off-the-shelf numerical libraries such as MATLAB and Intel Math Kernel Library [2] (MKL) are not tuned towards data-intensive applications either.

In the sections to follow, we present our approach to building an efficient SpMM kernel for sparse matrices with an irregular sparsity structure. We view SpMV as a special case of SpMM. We accelerate SpMM on the graphic processing units (GPU) because GPU is a suitable platform for operations requiring high memory bandwidth including SpMM. The basic idea is to build a performance model for the memory throughput of SpMM and determine when to use a cache blocking strategy. Distributed SpMM kernel is also very important for large sparse matrices such as the adjacency matrix for extreme-scale graphs; however, our aim is to accelerate SpMM on a single shared-memory machine which is a required component in any distributed implementation.

## 6.3 The SpMM Kernel and Related Work

We follow previous work [7, 20] and define the SpMM kernel as $Y \leftarrow A \cdot X + Y$, where $A \in \mathbb{R}^{m \times n}$ is a large sparse matrix and $X \in \mathbb{R}^{n \times k}$ is a tall-and-skinny dense matrix in the row-major order to improve the locality of memory access. The number

[2] `https://software.intel.com/en-us/intel-mkl` (retrieved in June 2014)

of floating point operations (flops) of SpMM is $2 \cdot Z \cdot k$, where $Z$ is the number of nonzero entries in $A$. The flop:byte ratio in single precision is $2k$ flops per 4 bytes if only the memory read for the sparse matrix entries is taken into account. Thus, SpMV and SpMM with a few columns are considered as a memory bandwidth bound operation. A tremendous number of techniques were proposed to accelerate SpMM on various hardware platforms. Since the graphic processing unit (GPU) has a much larger peak bandwidth than the host memory, SpMV/SpMM kernels on GPUs have received great attention in recent years [7, 20].

### 6.3.1 Our Assumption

Most of existing work on the SpMV/SpMM kernel on GPUs made several implicit assumptions: (1) The sparse matrix $A$ is small enough to fit into the GPU on-chip memory called the global memory; (2) $A$ has certain substructures such as small dense blocks that can be exploited for higher throughput. Therefore, it is considered useful to reorder the rows and columns of $A$ for better memory locality, and a variety of storage formats for sparse matrices were proposed in previous work to adapt to specific sparsity structures and increase thread and bandwidth utilization [7, 20]. In addition, the size of $A$ was not a factor in the optimization of these kernels; that is, a large matrix $A$ was not broken into smaller chunks for the purpose of SpMM.

However, we pose the following assumptions for data-intensive applications in this paper, which has a clear distinction from the previous assumptions: (1) $A$ is arbitrarily large and can be larger than the size of GPU global memory; (2) $A$ has an irregular sparsity structure without row and column reordering. Therefore, we consider it expensive to reorder the rows and columns of $A$. Most of the special formats proposed earlier will not work well either since they would enforce far more zero-paddings for a matrix with an irregular sparsity structure [20].

Therefore, in this chapter, we do not consider reordering the rows and columns of

$A$, but instead analyze the performance of reading and writing the dense matrices $X$ and $Y$. Previous work has focused on improving the sparse storage format for $A$ to achieve higher memory throughput. On the contrary, we assume a fixed sparse format for $A$ and improve the memory throughput of accessing $X$ and $Y$. Our implementation is an optimized routine for SpMM for a given sparse format. We focus on analyzing the in-core performance where $A$ is assumed to reside on the global memory; however, the implementation supports out-of-core computation that streams chunks of $A$ onto the GPU when $A$ is larger than the size of global memory.

In our performance model, we consider a generic sparse format, where one floating point number and at least one integer index are needed for storing each nonzero in $A$. Thus, the memory space required to store $A$ is $4 \cdot 2Z$ bytes using single precision and 32-bit integer indices. Compressed Sparse Row (CSR) is a widely-used and generic format for storing a sparse matrix and the default format in many software packages such as MATLAB and GraphLab [74]. In the CSR format, the nonzeros in each row are stored contiguously. Thus, it is a natural choice for many applications, for example in graph analytics where the adjacency list of a node is contained in one row. We use the CSR format as a running example; however, our performance model is not restricted to any specific sparse format.

### 6.3.2 Cache Blocking for SpMM

We focus on cache reuse of the dense matrix $X$ to improve the memory throughput. Consider the operations associated with a nonzero `A[i,j]`, the $(i, j)$-th entry of $A$, on the GPU:

$$\texttt{Y[i,:]  += A[i,j] * X[j,:].} \tag{67}$$

Generic storage formats such as CSR store at least one matrix index, which we assume is the column index, along with each nonzero entry. The specific row `X[j,:]` in the operation (67) is read from the global memory only when the program knows the column index `j` that is associated with `A[i,j]`. When an SpMM kernel goes through all the nonzeros row by row and executes (67), it needs one memory read for each nonzero entry in $A$ along with its column index, one read and one write for each row `Y[i,:]`, and multiple reads for each row `X[j,:]` (once for each nonzero in the `j`-th column of $A$). Thus, we often cache the row `X[j,:]` so that later accesses to this row of $X$ can reuse `X[j,:]` in the cache rather than loading it from global memory. When $X$ is large and cannot fit into the cache, we can divide the columns of $A$ into several blocks, called column blocking, such that each corresponding row block of $X$ can fit into the cache in order to have the fewest cache misses.

*Cache blocking* is a standard technique for implementing dense matrix computation [101]. The idea of applying cache blocking for SpMV kernels was also explored on the CPU to reduce cache misses [45]; however, its effect on the actual timing results was not as dramatic as the case for dense matrices. The same idea did not prove effective on the GPU at all in previous work where the focus was placed on sparse matrices with small dense blocks [20].

We will, on the contrary, show that cache blocking impacts the overall efficiency of SpMM kernels on the GPU under our assumptions above, where reordering the rows and columns is not considered for a sparse matrix arising from data-intensive applications. The essence of this chapter is to show that while the memory-bandwidth-bound nature of SpMM usually refers to the memory reads for $A$, the memory reads for $X$ can impose additional bandwidth requirement and need to be accounted for in the performance analysis. Moreover, contrary to previous SpMM kernels, the size of $A$ becomes a factor in the optimization of SpMM; that is, a large matrix $A$ can be broken into smaller chunks to facilitate cache reuse in SpMM.

Table 18: Symbols and their units in the performance model for SpMM.

| Symbol | Description | Unit |
|---|---|---|
| $m$ | Number of rows of $A$ | - |
| $n$ | Number of columns of $A$ | - |
| $k$ | Number of columns of $X$ and $Y$ | - |
| $Z$ | Number of nonzeros of $A$ | - |
| $\rho_A$ | Density of nonzeros in $A$ | - |
| $T$ | Total time for SpMM | seconds |
| $T_{\text{pci}}$ | Time for host/device PCI-e transfer | seconds |
| $T_{\text{comp}}$ | Time for computation on GPU cores | seconds |
| $T_{\text{mem}}$ | Time for global memory operations | seconds |
| $T_{\text{global}}$ | Time for global memory operations using no cache | seconds |
| $T_{\text{tex}}$ | Time for global memory operations using texture cache | seconds |
| $T_{\text{shr1}}, T_{\text{shr2}}$ | Time for global memory operations using shared memory | seconds |
| $T_{\text{L2}}$ | Time for global memory operations using L2 cache | seconds |
| $T_{\text{final}}$ | Time for global memory operations in our SpMM routine | seconds |
| $\beta_{\text{pci,h2d}}$ | Host-to-device PCI-e bandwidth (pinned) | seconds |
| $\beta_{\text{pci,d2h}}$ | Device-to-host PCI-e bandwidth (pinned) | seconds |
| $\beta_{\text{pci}}$ | Average host/device PCI-e bandwidth (pinned) | seconds |
| $\beta_{\text{mem}}$ | Peak global memory bandwidth | bytes/second |
| $\beta_{\text{csr}}$ | Achievable memory bandwidth for $A$ | bytes/second |
| $r_b$ | Number of row blocks | - |
| $c_b$ | Number of column blocks | - |
| $N_{\text{SM}}$ | Number of multiprocessors (SMs) | - |
| $L_{\text{tex}}$ | Texture cache size / SM | bytes |
| $L_{\text{shr}}$ | Shared memory / threadblock | bytes |
| $L_{\text{L2}}$ | L2 cache size | bytes |
| $N_{\text{thread}}$ | Maximum number of active threads / SM | - |
| $N_{\text{kernel}}$ | Actual number of threads / threadblock | - |
| $N_{\text{vector}}$ | Number of threads assigned to each row of $A$ | - |
| $O$ | Maximum number of active threadblocks | - |

## 6.4 Performance Analysis for SpMM

In this section, we build a performance model to analyze the SpMM kernel and guide our implementation. The goal of our model is to reveal when cache blocking should be used, which type of cache on the GPU should be used for caching in SpMM, and whether changing the sparse storage format helps with the performance. The symbols that will be used in the performance model are listed in Table 18.

Without loss of generality, we focus our discussion on single-precision floating point numbers and 32-bit integers as matrix indices. GPUs are connected to the

CPU host through a Peripheral Component Interconnect Express (PCI-e) bus. A typical GPU kernel works by first transferring the input data from the host to a GPU, computing in the GPU cores, and tranferring the results back to the host. For an out-of-core SpMM routine, the lower bound of the total time $T$ can be written as:

$$T \geq \max\{T_{\text{pci}}, T_{\text{comp}}, T_{\text{mem}}\}, \tag{68}$$

$T_{\text{pci}}$ is generally the dominant part because the peak PCI-e bandwidth $\beta_{\text{pci}}$ is much less than the peak global memory bandwidth and the peak achievable computing power. Hence, we can use $T_{\text{pci}}$ as a tight bound for the total time of out-of-core SpMM unless $k$ is very large (we will quantify the upper bound of $k$ such that $T_{\text{pci}}$ dominates the performance later in this section):

$$T \geq T_{\text{pci}} = 4 \cdot \left( \frac{2Z + nk}{\beta_{\text{pci,h2d}}} + \frac{mk}{\beta_{\text{pci,d2h}}} \right). \tag{69}$$

However, we would like to optimize both the out-of-core and the in-core performances. To simplify the setting, we will ignore the host/device transfer and focus on the in-core section. It might be natural to believe that SpMM with a small $k$ is memory bandwidth bound and SpMM with a large $k$ above some threshold can become compute-bound. However, as it turns out, SpMM with a large $k$ is still memory bandwidth bounded; and contrary to common understanding, we will show that in the case with a small $k$, the memory bandwidth bound sometimes can be attributed to reading the dense matrix rather than the sparse matrix. In the following text, we focus on analyzing the memory throughput of in-core SpMM.

We formulate our problem as follows: Given an $m \times n$ sparse matrix $A$ with $Z$ nonzeros, $n \times k$ dense matrix $X$, and assuming the storage format for $A$ is fixed, we would like to estimate the time for accessing global memory for SpMM. Some relevant machine-specific parameters for the K20x GPU we used in experiments are listed in Table 19. The peak global memory bandwidth was measured by `bandwidthTest` in the CUDA SDK. We employ several additional kernel-specific symbols: $r_b, c_b, \beta_{\text{csr}}, N_{\text{kernel}}, O$

Table 19: Specifications for NVIDIA K20x GPU.

| | |
|---|---|
| Global memory | 5760 MB |
| PCI-e bandwidth (pinned, H2D) | 6.08 GB/s |
| PCI-e bandwidth (pinned, D2H) | 6.53 GB/s |
| Peak global memory bandwidth | 170 GB/s |
| Number of multiprocessors (SMs) | 14 |
| L2 cache | 1.5 MB |
| Texture cache / SM | $\leq$ 48 KB |
| Shared memory / threadblock | 48 KB |
| Maximum number of threads / SM | 2048 |

(refer to Table 18). The important parameters are $r_b$ and $c_b$, the numbers of row blocks and column blocks of $A$ that characterize the cache blocking scheme. $\beta_{\text{csr}}$ denotes the achievable global memory bandwidth for the memory read of $A$, where $A$ can be in the CSR format or other formats. The occupancy $O$ can be calculated as:

$$O = \frac{N_{\text{thread}}}{N_{\text{kernel}}} \cdot N_{\text{SM}}. \tag{70}$$

The key assumption in our performance model is similar to that for dense matrix multiplication: The large sparse matrix $A$ is arranged as a $r_b \times c_b$ grid of small matrices in order to fit the reusable dense matrices into the cache. In a SpMM kernel on GPU, the memory reads of $X$ can be serviced from the global memory directly; alternatively, we can exploit one of the caches available on the GPU including the texture cache, the shared memory, and L2 cache. We will introduce their functionality for each of them in our discussion below. We consider three scenarios separately – no cache reuse, using texture cache, and using shared memory – where L2 cache is not considered because it cannot be controlled by the programmer and the NVIDIA whitepaper[3] did not provide sufficient information for L2 cache.

*(1) Reading $X$ directly from global memory.* In this case, no caching is enabled, and we need to load a row of $X$ from global memory for each nonzero encountered in

[3]`http://www.nvidia.com/content/PDF/kepler/NVIDIA-Kepler-GK110-Architecture-Whitepaper.pdf` (retrieved in June 2014)

$A$. The time for global memory traffic is:

$$T_{\text{global}} = 4\left(\frac{2Z}{\beta_{\text{csr}}} + \frac{Z\cdot k}{\beta_{\text{mem}}} + \frac{mk}{\beta_{\text{mem}}}\right). \tag{71}$$

In this expression, the three terms correspond to the time for memory read of $A$, memory read of $X$, and memory write of $Y$, respectively. Changing the sparse storage format would help with the performance only when the first term is the dominant part of $T_{\text{global}}$, that is,

$$\frac{2Z}{\beta_{\text{csr}}} >> \frac{Z\cdot k}{\beta_{\text{mem}}} + \frac{mk}{\beta_{\text{mem}}}. \tag{72}$$

Under a mild assumption that $Z > m$, (72) becomes

$$k << 2\cdot\frac{Z}{m}\cdot\frac{\beta_{\text{mem}}}{\beta_{\text{csr}}} \tag{73}$$

Hence, changing the sparse format only impacts the in-core performance for SpMV and SpMM with a small number of dense columns, and is especially beneficial when the average number of nonzeros per row is small, i.e. when $A$ is sparser. The achievable bandwidth $\beta_{\text{csr}}$ for reading $A$ is larger for a more efficient sparse format and is constrained by $\beta_{\text{csr}} \leq \beta_{\text{mem}}$. When $A$ is fixed, it makes more sense to design a "better" sparse format than designing the "best" sparse format.

Comparing $T_{\text{global}}$ with $T_{\text{pci}}$ and using the approximation that $\beta_{\text{pci,h2d}} \approx \beta_{\text{pci,d2h}} \approx \beta_{\text{pci}} << \beta_{mem}$, and due to $\beta_{\text{csr}} \leq \beta_{\text{mem}}$, we derive the condition under which $T_{\text{pci}}$ dominates the out-of-core performance:

$$k << \frac{\frac{2}{\beta_{\text{pci}}}}{\frac{1}{\beta_{\text{mem}}} - \frac{(m+n)/Z}{\beta_{\text{pci}}}} \tag{74}$$

*(2) Reading X from texture cache.* Texture memory on the GPU has much higher bandwidth than the global memory and can be used for caching. Each multiprocessor (SM) has its own working set of texture memory, and we treat it as a dedicated on-chip cache for each SM. The time for global memory traffic when using texture memory for caching $X$ is:

$$T_{\text{tex}} = 4\left(\frac{2Z}{\beta_{\text{csr}}} + \frac{N_{\text{SM}}\cdot nk}{\beta_{\text{mem}}} + \frac{c_b\cdot mk}{\beta_{\text{mem}}}\right). \tag{75}$$

The second term in (75) accounts for memory read of $X$ by all the SMs. Theoretically, $X$ needs to be read only once for each SM since the data in texture cache survive between different threadblocks. The third term in (75) accounts for memory write of $Y$. Because we can only read but not write the texture memory, all accesses of $Y$ must be serviced from global memory. $Y$ need to be written for $c_b$ times, once for each column block.

We need to minimize $c_b$ for an optimistic estimate of $T_{\text{tex}}$. The expression for $T_{\text{tex}}$ assumes no cache misses once a block of $X$ is loaded into the texture cache, thus the size of each block of $X$ satisfies

$$4(n/c_b)k \leq L_{\text{tex}}. \tag{76}$$

We argue that in order to have no cache misses, $4(n/c_b)k$ cannot be larger than twice the texture cache size $2L_{\text{tex}}$. For a sparse matrix with an irregular sparsity structure, we assume a uniform distribution of nonzeros. Hence, when the size of each block of $X$ is twice the texture cache size, we have 100% cache misses and the performance can degrade to reading one row of $X$ from global memory for each nonzero of $A$. Therefore, we calculate $c_b$ as:

$$c_b = \left\lceil \frac{2kn}{L_{\text{tex}}} \right\rceil . \tag{77}$$

Comparing $T_{\text{tex}}$ with $T_{\text{global}}$, in order to benefit from caching in texture memory, we need to have

$$N_{\text{SM}}n + c_b m \leq Z + m. \tag{78}$$

For a large sparse matrix $A$, we can take the dominant term for both sides of (78) and drop the constant $N_{\text{SM}}$ and reach:

$$\frac{2kn \cdot m}{L_{\text{tex}}} \leq Z. \tag{79}$$

Thus we obtain a requirement for $\rho_A$, the density of $A$:

$$\rho_A = \frac{Z}{mn} \geq \frac{2k}{L_{\text{tex}}}. \tag{80}$$

This condition implies that we can benefit from texture caching for $A$ sufficiently dense above a threshold. For a fixed matrix $A$, this threshold goes up and texture caching will become not favorable as $k$ increases.

*(3) Reading $X$ from shared memory.* Each SM on the GPU has a built-in fast memory shared by all the threads in a threadblock, called *shared memory*, which is frequently used for caching. Unlike texture cache which is user transparent, data in shared memory needs to be loaded explicitly from global memory for each individual threadblock.

We discuss two possible implementations using shared memory. The first one uses shared memory only for caching $X$, and the time for global memory traffic is:

$$T_{\text{shr1}} = 4\left(\frac{2Z}{\beta_{\text{csr}}} + \frac{O \cdot nk}{\beta_{\text{mem}}} + \frac{c_b \cdot mk}{\beta_{\text{mem}}}\right). \tag{81}$$

The second term in (81) accounts for memory read of $X$ by all the threadblocks, and the third term accounts for memory write of $Y$ similar to (75). Unlike the texture cache, "cache miss" is not automatically handled by the hardware of shared memory. Thus, we only consider the case of no cache misses, and the size of each block of $X$ satisfies

$$4(n/c_b)k \le L_{\text{shr}}. \tag{82}$$

Alternatively, we can simultaneously cache $X$ and $Y$ in the shared memory to reduce the global memory traffic for $Y$. Thus in a second possible implementation, each threadblock proceeds through all the columns for each row block before getting to the next row block. In this case, $X$ needs to be read for $r_b$ times instead of $O$ times, and the performance gain due to column blocking is reduced. The time for global memory traffic is:

$$T_{\text{shr2}} = 4\left(\frac{2Z}{\beta_{\text{csr}}} + \frac{r_b \cdot nk}{\beta_{\text{mem}}} + \frac{mk}{\beta_{\text{mem}}}\right), \tag{83}$$

where $r_b$ is subject to

$$4(m/r_b + n/c_b)k \le L_{\text{shr}}. \tag{84}$$

Table 20: Text data matrices for benchmarking after preprocessing. $\rho$ denotes the density of each matrix.

|  | $m$ | $n$ | $Z$ | $\rho$ |
|---|---|---|---|---|
| RCV1 | 149,113 | 764,751 | 59,851,107 | $5.2 \times 10^{-4}$ |
| Wikipedia | 2,361,566 | 4,126,013 | 468,558,693 | $4.8 \times 10^{-5}$ |

We set $c_b = \infty$ and relax the constraint on $r_b$ to

$$4(m/r_b)k \leq L_{\text{shr}}. \tag{85}$$

Comparing $T_{\text{shr1}}, T_{\text{shr2}}$ with $T_{\text{global}}$, in order to benefit from caching in shared memory, we need to have

$$On + c_b m \leq Z + m, \tag{86}$$

$$r_b n + m \leq Z + m, \tag{87}$$

for the two implementations we discussed respectively. For a large sparse matrix $A$, we take the dominant terms and drop the constants and reach the same condition for both the implementations:

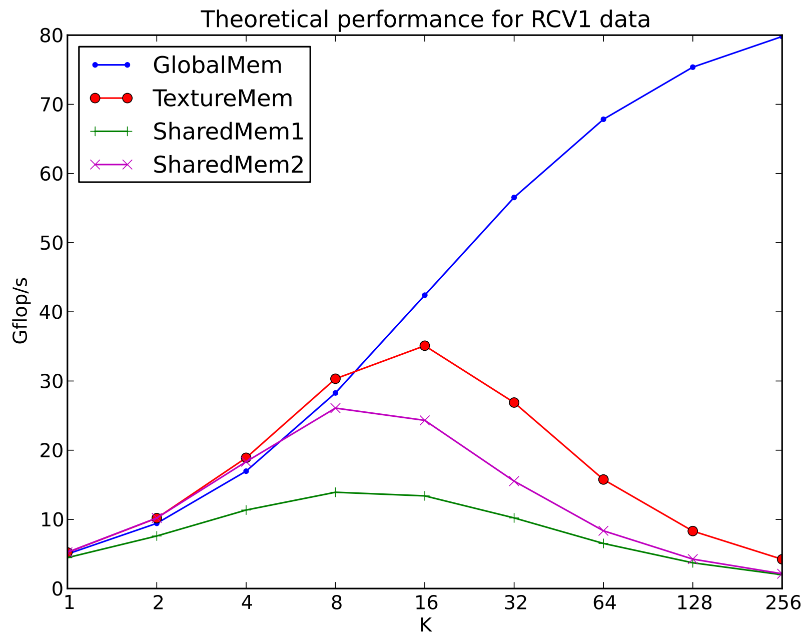
$$\frac{4mnk}{L_{\text{shr}}} \leq Z. \tag{88}$$

The condition on the density of $A$ in order to benefit from shared memory is:

$$\rho_A = \frac{Z}{mn} \geq \frac{4k}{L_{\text{shr}}}, \tag{89}$$

which implies that for a fixed $A$, it is more difficult to have efficiency benefit from shared memory than from texture memory for caching, given that the sizes of shared memory per threadblock and texture cache per SM are similar. The reason is that data need to be explicitly loaded into shared memory and the hardware is not designed to handle "cache miss" in shared memory, and thus the lower bound of $\rho_A$ for shared memory is smaller than that for texture cache by a factor of two.

Using the machine characteristics in Table 19, our performance model concludes that the smallest density of $A$ such as to make caching worthwhile is $\rho_A = 4 \times 10^{-5} \cdot k$



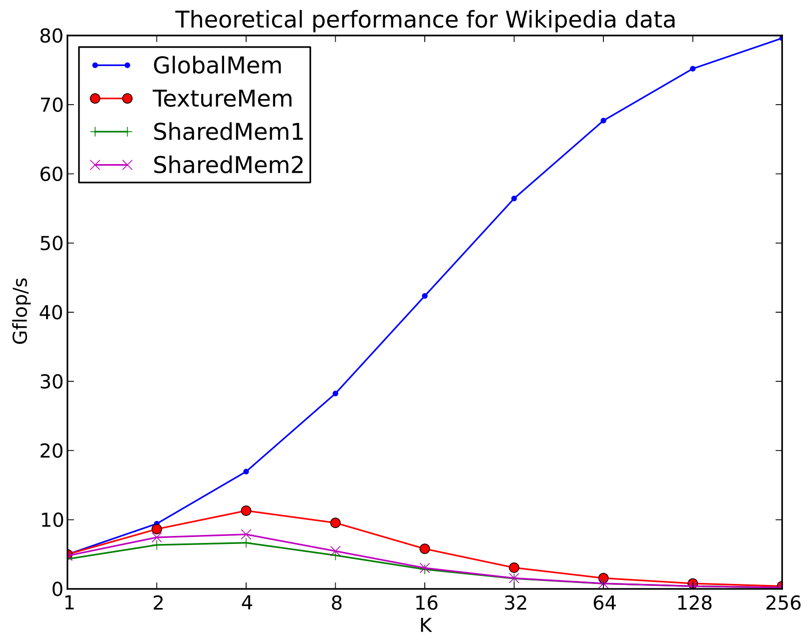


Figure 23: Theoretical performance bounds associated with no caching, texture sharing, and shared memory caching (with two possible implementations in Section 6.4).

for texture cache and $\rho_A = 8 \times 10^{-5} \cdot k$ for using shared memory as cache. We test our model on two real-world text data sets, whose statistics is listed in Table 20. The occupancy of threadblocks $O$ is 224, and we measure the empirical throughtput of reading a CSR sparse matrix by replacing the memory read of $X$ and $Y$ with dummy operations, which yields $\beta_{\text{csr}} = 21.21GB/s$. The achievable memory bandwidth using the CSR format is much smaller than the peak global memory bandwidth (170 GB/s) because the rows are stored contiguously in the CSR format and thus global memory access is not fully coalesced.

We plot the performance bounds determined by global memory traffic as a function of $k$ in Fig. 23. The theoretical performances when using texture cache and shared memory for caching could be slightly larger than that with no caching when $k$ is small, and degrade dramatically when $k$ is large. The asymptotic condition on $\rho_A$ suggests that for the RCV1 data set, the upper limit of $k$ is 13 in order to have efficient texture caching and is 6 in order to have efficient shared memory caching; for the Wikipedia data set, no caching is always more efficient. This result is consistent with visual inspection on Fig. 23.

These results suggest that SpMM is generally a memory bandwidth bound operation on the GPU, and using texture memory or shared memory for caching will not bring extra values for its overall performance. In fact, caching via texture memory and shared memory incurs additional overhead. Typically, the sizes of each texture cache and shared memory are small and therefore the number of column blocks $c_b$ would be large when caching is enabled. For example, using texture cache, the average number of nonzeros per row in each column block is $\rho L/(2k)$, which equals to 13 for RCV1 data and 1 for Wikipedia data. These numbers are too small to effectively utilize all the concurrent threads on the GPU. If multiple column blocks of $A$ were processed concurrently, much overhead would be incurred for atomic add operations to accumulate the partial sums associated with each column block. We tested caching via texture memory for SpMM, and its performance agreed with our reasoning above, degrading quickly when $c_b$ increases.

For out-of-core performance, we determine the upper bound of $k$ such that the time for host/device transfer dominates based on (74). For both the text data sets, this upper bound is about 90.

Note that we did not consider caching the sparse matrix which would be a better choice for a very large value of $k$. This is beyond the scope of this chapter since we put our emphasis on SpMM with a tall-and-skinny dense matrix $X$. It will an interesting question to find the upper limit of columns in $X$ such that no caching of $A$ is perferable over explicit caching of $A$ and/or replacing $A$ with a dense matrix.

We can still exploit the much larger L2 cache for cache blocking. In the next section, we implemented our SpMM routine, and estimate the performance bound taking L2 cache into account as well as the upper limit of $k$ where cache blocking is preferable to no caching.

## *6.5 Implementation*

We describe our implementation of an out-of-core routine for SpMM on the GPU. We store the sparse matrix in the CSR format, and when column blocking is used for better L2 cache performance, we store the entire matrix in consecutive submatrices each representing a column block in the CSR format. We wrote our routine based on the open-source package CUSP[4] by adding the functionality of column blocking, constructing dense matrices in the row-major order, and using CUDA streams. We also performed basic code optimization such as loop unrolling.

To cover the latency of host/device transfer and make our routine applicable to sparse matrices larger than the global memory size of the GPU, we break up the sparse matrix into $r_b$ large chunks of rows, and use one CUDA stream to compute the matrix product of one row chunk and the dense matrix $X$. The CUDA streams run concurrently and are independent with each other. We pre-allocate $M/r_b$ bytes of memory for each CUDA stream, where $M$ is the size of available global memory in

[4]`http://cusplibrary.github.io/` (retrieved in June 2014)

bytes. One iteration in each CUDA stream consists of three steps: 1. streaming the largest number of consecutive rows of $A$ that the preallocated memory can hold to the GPU; 2. invoking the SpMM kernel; 3. streaming the result matrix $Y$ corresponding to the rows of $A$ in this iteration back to the host. It turns out that the value of $r_b$ is not very influential for the wall-clock performance. If there are multiple column blocks, each stream processes the column blocks sequentially to avoid atomic adds for the partial sums.

We can perform column blocking on $A$ to leverage the L2 cache for reusing the memory of $X$. The L2 cache is built on top of the global memory and shared by all the SMs. All the memory reads of $A$ and $X$ first check the L2 cache; however, we expect that the memory for $A$ is not persistent in the L2 cache because there is no reuse of $A$ and L2 cache employs the *least-recently-used* (LRU) replacement policy.

Similar to Section 6.4, we develop a performance model to predict the performance bound of SpMM with L2 cache enabled. The model extends the previous one for global memory traffic $T_{\text{global}}$ (71) when reading $X$ from the global memory directly. The time for global memory traffic is:

$$T_{\text{L2}} = 4\left(\frac{2Z}{\beta_{\text{csr}}} + \frac{nk}{\beta_{\text{mem}}} + \frac{c_b \cdot mk}{\beta_{\text{mem}}}\right), \tag{90}$$

where the size of each block of $X$ satisfies

$$4(n/c_b)k \leq L_{\text{L2}}. \tag{91}$$

Note that (90) is an optimistic estimate because we assume the entire L2 cache is dedicated to caching $X$. As a practical performance model, we further require that the average number of nonzeros per row in each column block cannot be smaller than the number of threads assigned to each sparse matrix row in the SpMM kernel, that is,

$$\lfloor Z/(mc_b) \rfloor \geq N_{\text{vector}}. \tag{92}$$

In our implementation, $N_{\text{vector}} = 32$. Otherwise, it is difficult to achieve the maximum occupancy of threads on the GPU (see Section 6.4), and we resort to the strategy of no column blocking and reading $X$ from global memory directly. Therefore, our final model of a practical implementation depends on both $T_{\text{global}}$ and $T_{\text{L2}}$:

$$T = \begin{cases} T_{\text{L2}}, & \text{if } \lfloor Z/(mc_b) \rfloor \geq N_{\text{vector}} \text{ and } T_{\text{L2}} < T_{\text{global}}; \\ T_{\text{global}}, & \text{otherwise.} \end{cases} \tag{93}$$

Using similar estimation techniques presented in Section 6.4, we can express (93) in terms of $\rho_A$, the density of $A$:

$$T = \begin{cases} T_{\text{L2}}, & \text{if } \rho_A \geq 4kN_{\text{vec}}/L_{\text{L2}}; \\ T_{\text{global}}, & \text{otherwise.} \end{cases} \tag{94}$$

## *6.6 Benchmarking Results*

Now we present benchmarking results to evaluate the in-core performance of SpMM on large sparse matrices arising from machine learning applications, especially text data. We use a K20x GPU with CUDA 5.5 in the benchmarking.

First, we examine the validity of our practical performance model in Section 6.5. In Fig. 24, we compare the actual Gflop/s achieved by NVIDIA CUSPARSE[5] with the performance predicted by our model in Gflop/s. Our model closely matches the actual CUSPARSE performance, especially for a small $k$. It also leaves some room for improvement when $k$ is small.

For the RCV1 data, our model (94) suggests column blocking when $k <= 6$; for the Wikipedia data, our model suggest no column blocking for any $k$. Thus, in Fig. 25, we compare the performance given by column blocking via L2 cache with the CUSPARSE performance for the RCV1 data. We can see that our model correctly

[5]`http://docs.nvidia.com/cuda/cusparse/index.html` (retrieved in June 2014) We benchmark the `csrmm2` routine.

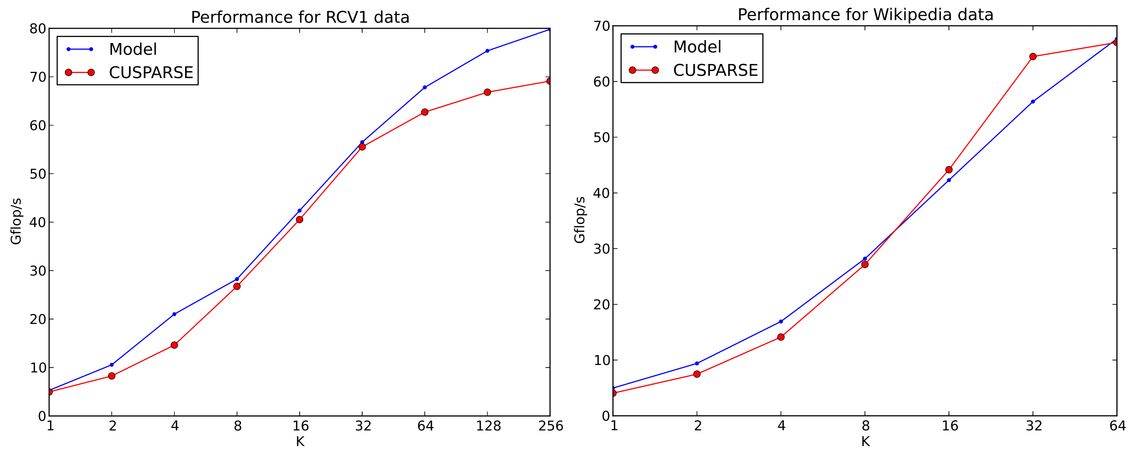


Figure 24: Performance comparisons between CUSPARSE and our model.

identifies the cases where column blocking would help, namely $k = 1, 2, 4$. Among the three cases, $k = 2$ corresponds to the largest speedup factor (1.33x), which is important for HierNMF2 based on rank-2 NMF. Note that the performances of our routine when $k = 1$ is slightly worse than previously reported results [7]. We can potentially obtain better performance specifically for SpMV using texture cache as suggested by Fig. 23.

## 6.7 Large-Scale Topic Modeling Experiments

In this section, we introduce a new method for large-scale topic modeling, called *HierNMF2-flat*, based on the techniques we have studied in previous sections. First, we apply the accelerated SpMM routine on the GPU to HierNMF2 that generates a hierarchy of clusters. Then we transform the hierarchical clustering result back to a flat clustering and flat topic model. The outcome is a scalable topic modeling method providing better-quality topics and significantly faster than previous methods.

We formulate the problem of flattening a hierarchy of clusters as a nonnegative least squares (NNLS) problem. Assume that at the end of HierNMF2, we obtain a hierarchy with $k$ leaf nodes. As explained in Section 5.4, each node $\mathcal{N}$ in the tree is associated with a multinomial term distribution represented as a vector $\mathbf{w}$ of length $m$. This vector is one of the two columns of $W$ given by the NMF result that

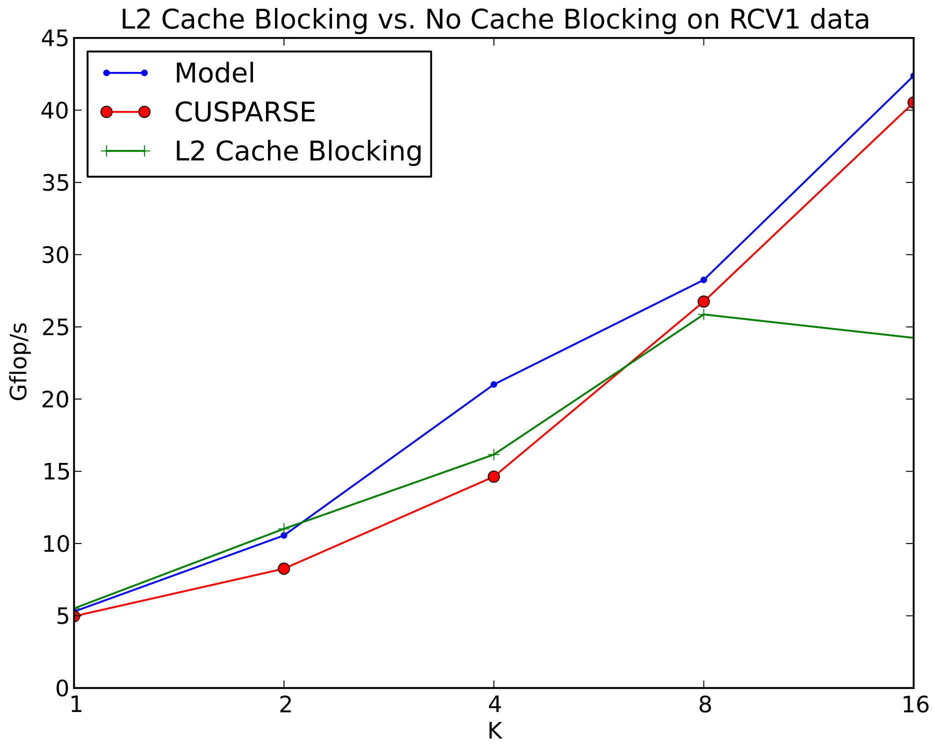


Figure 25: Performance comparisons between CUSPARSE and our routine on the RCV1 data set.

generates the node $\mathcal{N}$ along with its sibling. Now we treat each vector associated with a leaf node as a *topic* and collect all these vectors, forming a term-topic matrix $\overline{W} \in \mathbb{R}^{m\times k}$. This matrix can be seen as a topic model after each column is normalized. We compute an approximation of the term-document matrix $X$ using $\overline{W}$:

$$\min_{H\geq 0} \|\overline{W}H - X\|_F^2. \tag{95}$$

This is an NNLS problem and can be solved by many existing algorithms [53, 56, 57, 55]. This NNLS problem needs to be computed only once. Just as in the original NMF, the matrix $H$ in the solution of (95) can be treated as soft clustering assignments, and we can obtain a hard clustering assignment for the $i$-th document by selecting the index associated with the largest element in the $i$-th column of $H$.

We evaluated the clustering quality of HierNMF2-flat on text data sets with ground-truth labels. The basic information of the data sets can be found in Section 5.6. Fig. 26 shows the normalized mutual information given by HierNMF2-flat,

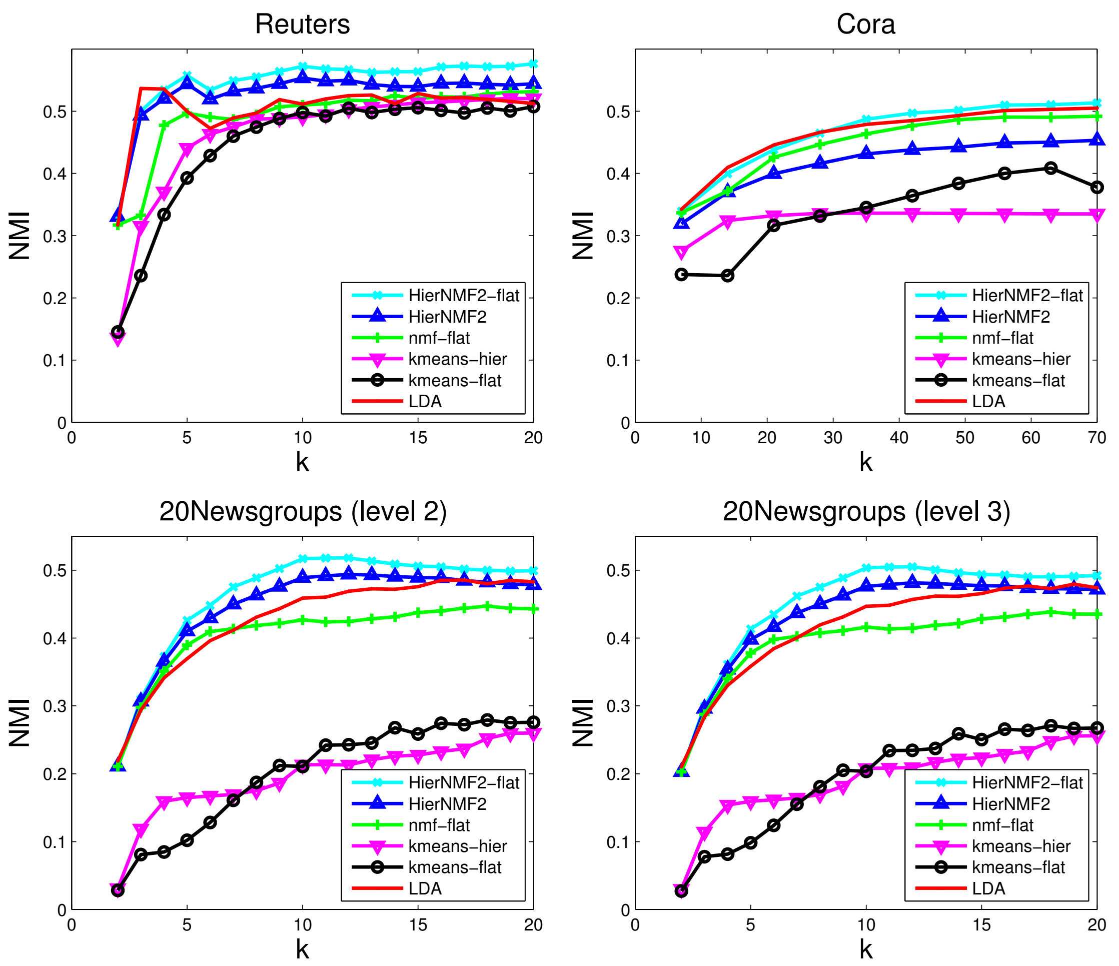


Figure 26: Evaluation of clustering quality of HierNMF2-flat on labeled text data sets.

Table 21: Timing results of HierNMF2-flat (in seconds).

| | Matlab (double) | C++ (double) | C++ (single) | GPU (single) |
|---|---|---|---|---|
| 800K RCV1 | 317.7 | 117.3 | 97.4 | 78.2 |
| 4.5M Wikipedia | 4704 | 1667.5 | 1374.8 | 763.8 |

HierNMF2, other widely-used methods. We can see that HierNMF2-flat gives consistently better topic/cluster quality than other methods. One possible reason for its better performance is that documents that appear to be outliers are removed when building the hierarchy in HierNMF2, and thus the topics at the leaf nodes are more meaningful and represent salient topics than those generated by a flat topic modeling method that takes every document into account.

We also evaluated the efficiency of HierNMF2-flat on large, unlabeled text data sets listed in Table 20. Table 21 shows the timing results of an Matlab implementation, a highly-optimized C++ implementation [59], and a CPU/GPU hybrid implementation. The Matlab performance is based on double precision since Matlab does not support single-precision sparse matrices. The C++ implementation includes an SpMM routine faster than Intel MKL for the $k = 2$ case, and therefore we did not show the timing result of HierNMF2-flat implemented with Intel MKL. The GPU routine for SpMM is based on single precision only.

The HierNMF2-flat algorithm acclerated by our SpMM routine on the GPU achieves about 5 times the efficiency of the Matlab version. On the RCV1 data set, compared to standard NMF and latent Dirichlet allocation (LDA) in double precision, our CPU/GPU hybrid implementation runs more than 200x faster. Therefore, we can conclude that it is at least 100x faster than standard NMF and LDA in single precision.

# CHAPTER VII

# CONCLUSIONS AND FUTURE WORK

Nonnegative matrix factorization (NMF) is a dimension reduction method with unique characteristics: The factor matrices it finds are nonnegative. Dimension reduction and clustering are closely related. In the following formulation of low-rank approximation

$$X \approx WH, \tag{96}$$

where $X \in \mathbb{R}_+^{m\times n}, W \in \mathbb{R}_+^{m\times k}, H \in \mathbb{R}_+^{k\times n}$, each column of $H$ is the $k$-dimensional representation of the corresponding column of $X$. If we can use $H$ to derive an assignment of the $n$ data points represented as the $n$ columns of $X$ into $k$ groups, clustering can be viewed as a special type of dimension reduction. One example is the classical K-means clustering:

$$\min \sum_{i=1}^{n} \|\mathbf{x}_i - \mathbf{w}_{g_i}\|_2^2, \tag{97}$$

where $\mathbf{x}_1, \cdots, \mathbf{x}_n$ are the columns of $X$, $\mathbf{w}_1, \cdots, \mathbf{w}_k$ are the $k$ centroids, and $g_i = j$ when the $i$-th data point is assigned to the $j$-th cluster ($1 \le j \le k$). Consider K-means formulated as a dimension reduction problem [54]:

$$\min_{H\in\{0,1\}^{k\times n}, \mathbf{1}_k^T H = \mathbf{1}_n^T} \|X - WH\|_F^2, \tag{98}$$

where $\mathbf{1}_k \in \mathbb{R}^{k\times 1}, \mathbf{1}_n \in \mathbb{R}^{n\times 1}$ are column vectors whose elements are all 1's. In the formulation (98), columns of $W$ are the cluster centroids, and the single nonzero element in each column of $H$ indicates the clustering assignment. Following this line of reasoning, NMF can be viewed as a clustering method that is formulated as a dimension reduction problem subject to a different set of constraints on $W$ and $H$:

$$\min_{W\ge 0, H\ge 0} \|X - WH\|_F^2. \tag{99}$$

In the formulation (99), columns of $W$ are the cluster representatives, and the largest element in each column of $H$ indicates the hard clustering assignment.

In this dissertation, we justified our solution to several issues of NMF related to its correctness and efficiency, and extended NMF to a general and efficient clustering method. First, we showed the limitation of the original NMF as a clustering method and proposed Symmetric NMF as a general clustering method that applies to a similarity graph. Second, we pointed out a critical flaw in the widely-used consensus NMF, and proposed a novel measure for cluster validation called Gap of Prediction Strength along with a new framework for cancer subtype discovery. Third, we devised a hierarchical clustering strategy that greatly improved the efficiency of NMF-based clustering. The performance boost came from our new algorithm for rank-2 NMF as well as a new way to transform a hierarchy of clusters into a flat clusterng result. Finally, we further accelerated rank-2 NMF by implementing sparse-dense matrix multiplication on the GPUs, and our performance model showed that we achieved the best optimal Gflop/s in the rank-2 case (that is, when the dense matrix in the multiplication has two columns).

For future work, a promising direction will be combining the high clustering quality of Symmetric NMF and the high efficiency of rank-2 NMF, and accelerating community detection on large graphs that runs orders of magnitudes faster and achieving comparable or even better quality of communities. Since the sizes of online social networks have become extremely large with billions of nodes, this direction is potentially important for the scalability of community detection algorithms.

Another direction is to apply our framework for cancer subtype discovery, which combines Gap of Prediction Strength and affine NMF, to more cancer studies and investigate the discovered cancer subtypes. We have seen at the end of Section 4 that the original NMF and affine NMF drew different conclusions on the number of *lung adenocarcinoma* subtypes. New knowledge of cancer biology could be generated by analyzing the correlation of the patient groups found by the original NMF and affine NMF with clinical data. As of this writing, there are 30 different cancers with genomic data available in the TCGA project. Putting the new framework into practice for a wider range of cancers would potentially bring better treatment for cancer patients and have an impact on peoples' lives.

# REFERENCES

[1] Arbelaez, P., Maire, M., Fowlkes, C., and Malik, J., "Contour detection and hierarchical image segmentation," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 33, no. 5, pp. 898–916, 2011.

[2] Arora, R., Gupta, M. R., Kapila, A., and Fazel, M., "Clustering by left-stochastic matrix factorization," in *ICML '11: Proc. of the 28th Int. Conf. on Machine learning*, 2011.

[3] Arora, S., Ge, R., Halpern, Y., Mimno, D. M., Moitra, A., Sontag, D., Wu, Y., and Zhu, M., "A practical algorithm for topic modeling with provable guarantees.".

[4] Arora, S., Ge, R., Kannan, R., and Moitra, A., "Computing a nonnegative matrix factorization – provably," in *STOC '12: Proc. of the 44th Symp. on Theory of Computing*, pp. 145–162, 2012.

[5] Banerjee, A., Dhillon, I. S., Ghosh, J., and Sra, S., "Clustering on the unit hypersphere using von Mises-Fisher distributions," *J. Mach. Learn. Res.*, vol. 6, pp. 1345–1382, 2005.

[6] Barrett, R., Berry, M., Chan, T. F., Demmel, J., Donato, J., Dongarra, J., Eijkhout, V., Pozo, R., Romine, C., and der Vorst, H. V., *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*. Philadelphia, PA: SIAM, 2nd ed., 1994.

[7] Bell, N. and Garland, M., "Efficient sparse matrix-vector multiplication on CUDA," Tech. Rep. NVR-2008-004, NVIDIA, 2008.

[8] Ben-Hur, A., Elisseeff, A., and Guyon, I., "A stability based method for discovering structure in clustered data," in *Pacific Symp. on Biocomputing 7*, pp. 6–17, 2002.

[9] Berman, A. and Plemmons, R. J., *Nonnegative matrices in the mathematical sciences*. Philadelphia, PA: SIAM, 1994.

[10] Berman, A. and Shaked-Monderer, N., *Completely positive matrices*. River Edge, NJ: World Scientific, 2003.

[11] Bertsekas, D. P., "Projected newton methods for optimization problems with simple constraints," *SIAM J. Control and Optimization*, vol. 20, no. 2, pp. 221–246, 1982.

[12] Bertsekas, D. P., *Nonlinear programming.* Belmont, MA: Athena Scientific, 2nd ed., 1999.

[13] Blei, D. M., Griffiths, T. L., Jordan, M. I., and Tenenbaum, J. B., "Hierarchical topic models and the nested Chinese restaurant process," in *Advances in Neural Information Processing Systems 16*, 2003.

[14] Blei, D. M., "Probabilistic topic models," *Commun. ACM*, vol. 55, pp. 77–84, 2012.

[15] Blei, D. M., Ng, A. Y., and Jordan, M. I., "Latent Dirichlet allocation," *J. Mach. Learn. Res.*, vol. 3, pp. 993–1022, 2003.

[16] Brunet, J.-P., Tamayo, P., Golub, T. R., and Mesirov, J. P., "Metagenes and molecular pattern discovery using matrix factorization," *Proc. Natl. Acad. Sci.*, vol. 101, no. 12, pp. 4164–4169, 2004.

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

[18] Catral, M., Han, L., Neumann, M., and Plemmons, R. J., "On reduced rank nonnegative matrix factorization for symmetric matrices," *Linear Algebra and Its Applications*, vol. 393, pp. 107–126, 2004.

[19] Chan, P., Schlag, M., and Zien, J., "Spectral k-way ratio-cut partitioning and clustering," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 13, no. 9, pp. 1088–1096, 1994.

[20] Choi, J. W., Singh, A., and Vuduc, R. W., "Model-driven autotuning of sparse matrix-vector multiply on GPUs," in *PPoPP '10: Proc. of the 15th ACM SIGPLAN Symp. on Principles and Practice of Parallel Programming*, pp. 115–126, 2010.

[21] Cichocki, A. and Phan, A. H., "Fast local algorithms for large scale nonnegative matrix and tensor factorizations," *IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences*, vol. E92A, no. 3, pp. 708–721, 2009.

[22] Cour, T., Benezit, F., and Shi, J., "Spectral segmentation with multiscale graph decomposition," in *CVPR '05: Proc. of the 2005 IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 1124–1131, 2005.

[23] Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R., "Indexing by latent semantic analysis," *J. American Soc. for Info. Sci.*, vol. 41, no. 6, pp. 391–407, 1990.

[24] Dhillon, I., Guan, Y., and Kulis, B., "A unified view of kernel k-means, spectral clustering and graph cuts," Tech. Rep. TR-04-25, University of Texas at Austin, 2005.

[25] Dhillon, I. and Modha, D. S., "Concept decompositions for large sparse text data using clustering," *Mach. Learn.*, vol. 42, pp. 143–175, 2001.

[26] Ding, C., Li, T., and Jordan, M., "Nonnegative matrix factorization for combinatorial optimization: Spectral clustering, graph matching, and clique finding," in *ICDM '08: Proc. of the 8th IEEE Int. Conf. on Data Mining*, pp. 183–192, 2008.

[27] Ding, C. and He, X., "Cluster merging and splitting in hierarchical clustering algorithms," in *ICDM '02: Proc. of the 2nd IEEE Int. Conf. on Data Mining*, pp. 139–146, 2002.

[28] Ding, C., He, X., and Simon, H. D., "On the equivalence of nonnegative matrix factorization and spectral clustering," in *SDM '05: Proc. of SIAM Int. Conf. on Data Mining*, pp. 606–610, 2005.

[29] Ding, C., Li, T., and Jordan, M. I., "Convex and semi-nonnegative matrix factorization," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 32, no. 1, pp. 45–55, 2010.

[30] Donoho, D. and Stodden, V., "When does non-negative matrix factorization give correct decomposition into parts?," in *Advances in Neural Information Processing Systems 16*, 2003.

[31] Dubes, R. and Jain, A. K., "Clustering techniques: The user's dilemma," *Pattern Recognition*, vol. 8, no. 4, pp. 247–260, 1976.

[32] Duda, R. O., Hart, P. E., and Stork, D. G., *Pattern Classification.* Wiley-Interscience, 2000.

[33] Eisen, M. B., Spellman, P. T., Brown, P. O., and Botstein, D., "Cluster analysis and display of genome-wide expression patterns," *Proc. Natl. Acad. Sci. USA*, vol. 95, no. 25, pp. 14863–14868, 1998.

[34] Fowlkes, C. and Malik, J., "How much does globalization help segmentation?," Tech. Rep. UCB/CSD-4-1340, University of California, Berkeley, 2004.

[35] Gao, Y. and Church, G., "Improving molecular cancer class discovery through sparse non-negative matrix factorization," *Bioinformatics*, vol. 21, no. 21, pp. 3970–3975, 2005.

[36] Globerson, A., Chechik, G., Pereira, F., and Tishby, N., "Euclidean embedding of co-occurrence data," *J. Mach. Learn. Res.*, vol. 8, pp. 2265–2295, 2007.

[37] Gonzales, E. F. and Zhang, Y., "Accelerating the lee-seung algorithm for non-negative matrix factorization," Tech. Rep. TR05-02, Rice University, 2005.

[38] Grippo, L. and Sciandrone, M., "On the convergence of the block nonlinear gauss-seidel method under convex constraints," *Operations Research Letters*, vol. 26, pp. 127–136, 2000.

[39] Han, W.-S., Lee, S., Park, K., Lee, J.-H., Kim, M.-S., Kim, J., and Yu, H., "TurboGraph: A fast parallel graph engine handling billion-scale graphs in a single PC," in *KDD '13: Proc. of the 19th ACM Int. Conf. on Knowledge Discovery and Data Mining*, pp. 77–85, 2013.

[40] Handl, J., Knowles, J., and Kell, D. B., "Computational cluster validation in post-genomic data analysis," *Bioinformatics*, vol. 21, no. 15, pp. 3201–3212, 2005.

[41] He, Z., Xie, S., Zdunek, R., Zhou, G., and Cichocki, A., "Symmetric nonnegative matrix factorization: Algorithms and applications to probabilistic clustering," *IEEE Trans. on Neural Networks*, vol. 22, no. 12, pp. 2117–2131, 2011.

[42] Ho, N.-D., *Nonnegative matrix factorization algorithms and applications*. PhD thesis, Universite catholique de Louvain, 2008.

[43] Hoyer, P. O., "Non-negative matrix factorization with sparseness constraints," *Journal of Machine Learning Research (JMLR)*, vol. 5, pp. 1457–1469, 2004.

[44] Hu, Y., Koren, Y., and Volinsky, C., "Collaborative filtering for implicit feedback datasets," in *ICDM '08: Proc. of the 8th IEEE Int. Conf. on Data Mining*, pp. 263–272, 2008.

[45] Im, E.-J., Yelick, K., and Vuduc, R., "SPARSITY: Optimization framework for sparse matrix kernels," *Int. J. High Perform. Comput. Appl.*, vol. 18, no. 1, pp. 135–158, 2004.

[46] Järvelin, K. and Kekäläinen, J., "Cumulated gain-based evaluation of IR techniques," *ACM Trans. Inf. Syst.*, vol. 20, no. 4, pp. 422–446, 2002.

[47] Jung, Y., Park, H., Du, D.-Z., and Drake, B. L., "A decision criterion for the optimal number of clusters in hierarchical clustering," *J. of Global Optimization*, vol. 25, no. 1, pp. 91–111, 2003.

[48] Kang, U., Tsourakakis, C. E., and Faloutsos, C., "PEGASUS: A peta-scale graph mining system implementation and observations," in *ICDM '09: Proc. of the 9th IEEE Int. Conf. on Data Mining*, pp. 229–238, 2009.

[49] Karypis, G., "Cluto – a clustering toolkit," Tech. Rep. TR02-017, University of Minnesota, 2002.

[50] Kelley, C. T., *Iterative Methods for Linear and Nonlinear Equations.* Philadelphia, PA: SIAM, 1995.

[51] Kim, D., Sra, S., and Dhillon, I., "Fast newton-type methods for the least squares nonnegative matrix approximation problem," in *SDM '07: Proc. of SIAM Int. Conf. on Data Mining*, pp. 343–354, 2007.

[52] Kim, H. and Park, H., "Sparse non-negative matrix factorizations via alternating non-negativity-constrained least squares for microarray data analysis," *Bioinformatics*, vol. 23, no. 12, pp. 1495–1502, 2007.

[53] Kim, H. and Park, H., "Nonnegative matrix factorization based on alternating non-negativity-constrained least squares and the active set method," *SIAM J. on Matrix Analysis and Applications*, vol. 30, no. 2, pp. 713–730, 2008.

[54] Kim, J. and Park, H., "Sparse nonnegative matrix factorization for clustering," Tech. Rep. GT-CSE-08-01, Georgia Inst. of Technology, 2008.

[55] Kim, J., He, Y., and Park, H., "Algorithms for nonnegative matrix and tensor factorizations: A unified view based on block coordinate descent framework," *Journal of Global Optimization*, 2013.

[56] Kim, J. and Park, H., "Toward faster nonnegative matrix factorization: A new algorithm and comparisons," in *Proc. of the 8th IEEE Int. Conf. on Data Mining*, pp. 353–362, 2008.

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

[58] Kleinberg, J., "An impossibility theorem for clustering," in *Advances in Neural Information Processing Systems 15*, pp. 446–453, 2002.

[59] Kuang, D., Boyd, R., Drake, B., and Park, H., "Fast hierarchical clustering and topic modeling based on rank-2 nonnegative matrix factorization." unpublished, 2014.

[60] Kuhn, H. W., "The hungarian method for the assignment problem," *Nav. Res. Logist. Q.*, vol. 2, pp. 83–97, 1955.

[61] Kulis, B., Basu, S., Dhillon, I., and Mooney, R., "Semi-supervised graph clustering: a kernel approach," in *ICML '05: Proc. of the 22nd Int. Conf. on Machine learning*, pp. 457–464, 2005.

[62] Kumar, A., Sindhwani, V., and Kambadur, P., "Fast conical hull algorithms for near-separable non-negative matrix factorization," in *ICML '13: Proc. of the 30th Int. Conf. on Machine Learning*, 2013.

[63] Kyrola, A., Blelloch, G., and Guestrin, C., “GraphChi: Large-scale graph computation on just a PC,” in *OSDI ’12: Proc. of the 10th USENIX Conf. on Operating Systems Design and Implementation*, pp. 31–46, 2012.

[64] Laurberg, H. and Hansen, L. K., “On affine non-negative matrix factorization,” in *ICASSP ’07: Proc. of the 2007 IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, pp. 653–656, 2007.

[65] Lawson, C. L. and Hanson, R. J., *Solving least squares problems.* Englewood Cliffs, NJ: Prentice-Hall, 1974.

[66] Lee, D. D. and Seung, H. S., “Learning the parts of objects by non-negative matrix factorization,” *Nature*, vol. 401, pp. 788–791, 1999.

[67] Lee, K.-C., Ho, J., and Kriegman, D., “Acquiring linear subspaces for face recognition under variable lighting,” *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 27, no. 5, pp. 684–698, 2005.

[68] Lewis, D. D., Yang, Y., Rose, T. G., and Li, F., “RCV1: A new benchmark collection for text categorization research,” *J. Mach. Learn. Res.*, vol. 5, pp. 361–397, 2004.

[69] Li, T., Ding, C., and Jordan, M. I., “Solving consensus and semi-supervised clustering problems using nonnegative matrix factorization,” in *ICDM ’07: Proc. of the 7th IEEE Int. Conf. on Data Mining*, pp. 577–582, 2007.

[70] Lin, C.-J., “On the convergence of multiplicative update algorithms for nonnegative matrix factorization,” *Trans. Neur. Netw.*, vol. 18, no. 6, pp. 1589–1596, 2007.

[71] Lin, C.-J., “Projected gradient methods for nonnegative matrix factorization,” *Neural Computation*, vol. 19, no. 10, pp. 2756–2779, 2007.

[72] Liu, X., Chow, E., Vaidyanathan, K., and Smelyanskiy, M., “Improving the performance of dynamical simulations via multiple right-hand sides,” in *IPDPS ’12: Proc. of the 26th Int. Parallel Distributed Processing Symp.*, pp. 36–47, 2012.

[73] Lloyd, S., “Least squares quantization in PCM,” *IEEE Trans. Inf. Theor.*, vol. 28, no. 2, pp. 129–137, 1982.

[74] Low, Y., Gonzalez, J., Kyrola, A., Bickson, D., and Guestrin, C., “GraphLab: A new parallel framework for machine learning,” in *UAI ’10: Proc. of the 26th Conf. on Uncertainty in Artificial Intelligence*, pp. 340–349, 2010.

[75] Ma, X., Gao, L., Yong, X., and Fu, L., “Semi-supervised clustering algorithm for community structure detection in complex networks,” *Physica A: Statistical Mechanics and its Applications*, vol. 389, no. 1, pp. 187–197, 2010.

[76] Malik, J., Belongie, S., Leung, T., and Shi, J., "Contour and texture analysis for image segmentation," *Int. J. Computer Vision*, vol. 43, no. 1, pp. 7–27, 2001.

[77] Manning, C. D., Raghavan, P., and Schütze, H., *Introduction to Information Retrieval.* New York, NY: Cambridge University Press, 2008.

[78] Martin, D., Fowlkes, C., Tal, D., and Malik, J., "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *ICCV '01: Proc. of the 8th IEEE Int. Conf. on Computer Vision, Vol. 2*, pp. 416–423, 2001.

[79] Martin, D., Fowlkes, C., and Malik, J., "Learning to detect natural image boundaries using local brightness, color, and texture cues," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 26, no. 5, pp. 530–549, 2004.

[80] McCallum, A. K., Nigam, K., Rennie, J., and Seymore, K., "Automating the construction of Internet portals with machine learning," *Inf. Retr.*, vol. 3, no. 2, pp. 127–163, 2000.

[81] Mimno, D., Wallach, H. M., Talley, E., Leenders, M., and McCallum, A., "Optimizing semantic coherence in topic models," in *EMNLP '11: Proc. of the Conf. on Empirical Methods in Natural Language Processing*, pp. 262–272, 2011.

[82] Monti, S., Tamayo, P., Mesirov, J., and Golub, T., "Consensus clustering: A resampling-based method for class discovery and visualization of gene expression microarray data," *Mach. Learn.*, vol. 52, no. 1-2, pp. 91–118, 2003.

[83] Nabney, I. T., *NETLAB: Algorithms for Pattern Recognition.* London, United Kingdom: Springer, 2004.

[84] Nepusz, T., Petróczi, A., Négyessy, L., and Bazsó, F., "Fuzzy communities and the concept of bridgeness in complex networks," *Phys. Rev. E*, vol. 77, no. 1, p. 016107, 2008.

[85] Ng, A. Y., Jordan, M. I., and Weiss, Y., "On spectral clustering: Analysis and an algorithm," in *Advances in Neural Information Processing Systems 14*, pp. 849–856, 2001.

[86] Owen, S., Anil, R., Dunning, T., and Friedman, E., *Mahout in Action.* Shelter Island, NY: Manning Publications, 2011.

[87] Paatero, P. and Tapper, U., "Positive matrix factorization: a non-negative factor model with optimal utilization of error estimates of data values," *Environmetrics*, vol. 5, pp. 111–126, 1994.

[88] Pauca, V. P., Shahnaz, F., Berry, M. W., and Plemmons, R. J., "Text mining using non-negative matrix factorizations," in *SDM '04: Proc. of SIAM Int. Conf. on Data Mining*, pp. 452–456, 2004.

[89] Rajaraman, A. and Ullman, J. D., *Mining of massive datasets.* Cambridge, United Kingdom: Cambridge University Press, 2011.

[90] Shahnaz, F., Berry, M. W., Pauca, V. P., and Plemmons, R. J., "Document clustering using nonnegative matrix factorization," *Inf. Process. Manage.*, vol. 42, pp. 373–386, 2006.

[91] Shi, J. and Malik, J., "Normalized cuts and image segmentation," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 888–905, 2000.

[92] Sim, T., Baker, S., and Bsat, M., "The CMU pose, illumination, and expression database," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 25, no. 12, pp. 1615–1618, 2003.

[93] Sokal, R. R. and Rohlf, F. J., "The comparison of dendrograms by objective methods," *Taxon*, vol. 11, no. 2, pp. 33–40, 1962.

[94] Stewart, G. W. and Sun, J.-G., *Matrix perturbation theory.* Academic Press, 1990.

[95] Strogatz, S. H., "Exploring complex networks," *Nature*, vol. 410, pp. 268–276, 2001.

[96] Tibshirani, R. and Walther, G., "Cluster validation by prediction strength," *J. Computational and Graphical Statistics*, vol. 14, no. 3, pp. 511–528, 2005.

[97] Tibshirani, R., Walther, G., and Hastie, T., "Estimating the number of clusters in a data set via the gap statistic," *J. Royal Stat. Soc.: Series B*, vol. 63, no. 2, pp. 411–423, 2001.

[98] Van Benthem, M. H. and Keenan, M. R., "Fast algorithm for the solution of large-scale non-negativity constrained least squares problems," *Journal of Chemometrics*, vol. 18, pp. 441–450, 2004.

[99] van der Maaten, L. and Hinton, G., "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, pp. 2579–2605, 2008.

[100] Virtanen, T., "Monaural sound source separation by nonnegative matrix factorization with temporal continuity and sparseness criteria," *IEEE Trans. on Audio, Speech, and Language Processing*, vol. 15, no. 3, pp. 1066–1074, 2007.

[101] Volkov, V. and Demmel, J. W., "Benchmarking GPUs to tune dense linear algebra," in *SC '08: Proc. of the 2008 ACM/IEEE Conf. on Supercomputing*, pp. 31:1–31:11, 2008.

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

[103] Vuduc, R., *Automatic performance tuning of sparse matrix kernels.* PhD thesis, University of California, Berkeley, 2003.

[104] Vuduc, R., Demmel, J. W., and Yelick, K. A., "OSKI: A library of automatically tuned sparse matrix kernels," *J. Phys.: Conf. Series*, vol. 16, pp. 521–530, 2005.

[105] Wang, F., Li, T., Wang, X., Zhu, S., and Ding, C., "Community discovery using nonnegative matrix factorization," *Data Min. Knowl. Discov.*, vol. 22, no. 3, pp. 493–521, 2011.

[106] Wild, S., Curry, J., and Dougherty, A., "Improving non-negative matrix factorizations through structured initialization," *Pattern Recognition*, vol. 37, pp. 2217–2232, 2004.

[107] Willett, P., "Recent trends in hierarchic document clustering: a critical review," *Inf. Process. Manage.*, vol. 24, no. 5, pp. 577–597, 1988.

[108] Williams, S., Oliker, L., Vuduc, R., Shalf, J., Yelick, K., and Demmel, J., "Optimization of sparse matrix-vector multiplication on emerging multicore platforms," in *SC '07: Proc. of the 2007 ACM/IEEE Conf. on Supercomputing*, pp. 38:1–38:12, 2007.

[109] Xie, B., Song, L., and Park, H., "Topic modeling via nonnegative matrix factorization on probability simplex," in *NIPS Workshop on Topic Models: Computation, Application, and Evaluation*, 2013.

[110] Xu, W., Liu, X., and Gong, Y., "Document clustering based on non-negative matrix factorization," in *SIGIR '03: Proc. of the 26th Int. ACM Conf. on Research and development in informaion retrieval*, pp. 267–273, 2003.

[111] Yang, Z., Hao, T., Dikmen, O., Chen, X., and Oja, E., "Clustering by nonnegative matrix factorization using graph random walk," in *Advances in Neural Information Processing Systems 25*, pp. 1088–1096, 2012.

[112] Yang, Z. and Oja, E., "Clustering by low-rank doubly stochastic matrix decomposition," in *ICML '12: Proc. of the 29th Int. Conf. on Machine learning*, 2012.

[113] Yu, S. X. and Shi, J., "Multiclass spectral clustering," in *ICCV '03: Proc. of the 9th IEEE Int. Conf. on Computer Vision*, pp. 313–319, 2003.

[114] Yun, S., Tseng, P., and Toh, K.-C., "A block coordinate gradient descent method for regularized convex separable optimization and covariance selection," *Mathematical Programming*, vol. 129, pp. 331–355, 2011.

[115] Zass, R. and Shashua, A., "A unifying approach to hard and probabilistic clustering," in *ICCV '05: Proc. of the 10th IEEE Int. Conf. on Computer Vision*, pp. 294–301, 2005.

[116] Zelnik-Manor, L. and Perona, P., "Self-tuning spectral clustering," in *Advances in Neural Information Processing Systems 17*, pp. 1601–1608, 2004.

[117] Zhang, S., Wang, W., Ford, J., and Makedon, F., "Learning from incomplete ratings using non-negative matrix factorization," in *SDM '06: Proc. of SIAM Int. Conf. on Data Mining*, pp. 548–552, 2006.

[118] Zhang, Y. and Yeung, D.-Y., "Overlapping community detection via bounded nonnegative matrix tri-factorization," in *KDD '12: Proc. of the 18th ACM Int. Conf. on Knowledge Discovery and Data Mining*, pp. 606–614, 2012.

[119] Zhang, Z.-Y., Wang, Y., and Ahn, Y.-Y., "Overlapping community detection in complex networks using symmetric binary matrix factorization," *Phys. Rev. E*, vol. 87, no. 6, p. 062803, 2013.