Provable Algorithms for Machine Learning Problems

Rong Ge

A Dissertation
Presented to the Faculty
of Princeton University
in Candidacy for the Degree
of Doctor of Philosophy

Recommended for Acceptance
by the Department of
Computer Science
Adviser: Professor Sanjeev Arora

November 2013

Abstract

Modern machine learning algorithms can extract useful information from text, images and videos. All these applications involve solving NP-hard problems in average case using heuristics. What properties of the input allow it to be solved efficiently? Theoretically analyzing the heuristics is very challenging. Few results were known.

This thesis takes a different approach: we identify natural properties of the input, then design new algorithms that provably works assuming the input has these properties. We are able to give new, provable and sometimes practical algorithms for learning tasks related to text corpus, images and social networks.

The first part of the thesis presents new algorithms for learning thematic structure in documents. We show under a reasonable assumption, it is possible to provably learn many topic models, including the famous Latent Dirichlet Allocation. Our algorithm is the first provable algorithms for topic modeling. An implementation runs 50 times faster than latest MCMC implementation and produces comparable results.

The second part of the thesis provides ideas for provably learning deep, sparse representations. We start with sparse linear representations, and give the first algorithm for dictionary learning problem with provable guarantees. Then we apply similar ideas to deep learning: under reasonable assumptions our algorithms can learn a deep network built by denoising autoencoders.

The final part of the thesis develops a framework for learning latent variable models. We demonstrate how various latent variable models can be reduced to orthogonal tensor decomposition, and then be solved using tensor power method. We give a tight sample complexity analysis for tensor power method, which reduces the number of sample required for learning many latent variable models.

In theory, the assumptions in this thesis help us understand why intractable problems in machine learning can often be solved; in practice, the results suggest inherently new approaches for machine learning. We hope the assumptions and algorithms inspire new research problems and learning algorithms.

Acknowledgements

I would like to thank my parents for giving me life, love, and constant support.

I thank my advisor, Prof. Sanjeev Arora, for his advice and encouragements. His knowledge, way of thinking, passion, and optimism guided me through my five years in Princeton. He is always willing to spend lots of time to discuss about research. He also helped me with my writing and presentations with lots of patience. I could not have hoped for a better advisor.

I would also like to thank Sham Kakade, who mentored me when I was interning at Microsoft Research New England in summer 2012, and after I left Princeton University. His viewpoints and intuitions opened a different gate for me, which is important for the line of research presented in this thesis.

Thanks to Moses Charikar, Sham Kakade, Mark Bravermann, and Zeev Dvir for being in my thesis committee.

I thank the entire theory group in Princeton, IAS, and more generally in the Center for Computational Intractability. Their discussions and numerous talks give me the chance of learning new results and ideas.

I thank Princeton University and the Computer Science Department for providing a perfect environment for research. I especially thank Melissa Lawson and Mitra Kelly, for their help in all administrative work.

I also thank Andrew Chi-Chih Yao, who went to Tsinghua University when I was an undergraduate student there. The courses he taught introduced theoretical computer science to me, and lead me to the research path.

This thesis will not be possible without my collaborators. An incomplete list would be: Animashree Anandkumar, Boaz Barak, Aditya Bhaskara, Markus Brunnermeier, Decheng Dai, Yoni Halpern, Daniel Hsu, Ravindran Kannan, Tengyu Ma, David Mimno, Ankur

Moitra, Sushant Sachdeva, Grant Schoenebeck, Ali Kemal Sinop, David Sontag, Matus Telgarsky, Yichen Wu, and Michael Zhu.

This thesis is supported under NSF grants CCF-0832797, CCF-1117309, CCF-1302518, DMS-1317308, and Simons Investigator Grant.

Finally, I would like to thank my wife, who brought happiness and passion to my life.

Contents

List of Tables

List of Figures

Notations

$a, b, c, ...$	constants
$\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w}, ...$	column vectors
$\mathbf{0}_m$, $\mathbf{1}_m$	all 0's/1's vector of length m (m is omitted if it is clear from context)
$\mathrm{supp}(\boldsymbol{u})$	the support of vector $\boldsymbol{u}$ (set of nonzero indices)
$M, N, ...$	matrices
I_n	$n \times n$ identity matrix (n is omitted if it is clear from context)
$T, ...$	tensors
$\boldsymbol{u} \cdot \boldsymbol{v}$	inner-product of two vectors
$\mathrm{Diag}(\boldsymbol{u})$	diagonal matrix whose diagonal entries are from the vector u
$\\|\boldsymbol{u}\\|$	ℓ_2 norm of a vector
$\|\boldsymbol{v}\|_1$	ℓ_1 norm of a vector
$\\|M\\|$	spectral norm of a matrix
$\|M\|_1$	ℓ_1 norm of a matrix
$\\|M\\|_F$	Frobenius norm of a matrix
$\\|T\\|$	spectral norm of a tensor
$\\|T\\|_F$	Frobenius norm of a tensor
$\lambda_i(M)$	i-th largest eigenvalue of a matrix
$\kappa(M)$	condition number of a matrix
$M^\top$	Transpose of a matrix
M^{-1}	inverse of a matrix (M will always be invertible)
$M = UDV^\top$	the singular value decomposition of matrix M
$M^\dagger$	the Moore-Penrose pseudo-inverse of the matrix M
$\boldsymbol{u}_i, \boldsymbol{u}[i]$	the i-th entry of the vector
$\boldsymbol{u}_{i,j}, \boldsymbol{u}[i, j]$	the entry in i-th row and j-th column of the matrix

M_i	i-th column of a matrix
M^j	j-th row of a matrix
$T(U, V, W)$	tri-linear form of a tensor
$\boldsymbol{u} \otimes \boldsymbol{v}$	tensor product of vectors $\boldsymbol{u}$ and $\boldsymbol{v}$, equivalent to matrix $\boldsymbol{u}\boldsymbol{v}^\top$
$\boldsymbol{u} \otimes \boldsymbol{v} \otimes \boldsymbol{w}$	tensor of three vectors
$\boldsymbol{u}^{\otimes k}$	shorthand for $\boldsymbol{u} \otimes \boldsymbol{u} \otimes \cdots \otimes \boldsymbol{u}$ (k $\boldsymbol{u}$'s in the formula)
$[n]$	the set $\{1, 2, \ldots, n\}$

Chapter 1

Introduction

Most machine learning problems are computationally intractable. However in practice, many heuristic algorithms work well. How do we explain this mismatch? At a high level, this is a mismatch between worst case and average case complexity. Computational intractability only implies hardness on worst case inputs, which may not occur in practice. The heuristic algorithms work because they only need to solve the "easy" instances.

Investigating the details of this explanation raises more questions. What properties of the input make it "easy" for heuristic algorithms? Assuming the input satisfies these properties, can we prove current heuristic algorithms work? Can we design new algorithms that work provably on "easy" inputs?

In this thesis, we try to answer these problems using a few examples. For each learning task, we identify reasonable properties that are satisfied by real-life inputs, and design provable algorithms using these properties. In theory, these new results show why hard problems in machine learning can in fact be solved; in practice, these results lead to fundamentally different approaches.

We focus on *unsupervised learning* problems: given data points, the goal is to discover useful structure hiding in data. This is in contrast to supervised learning, where we are given data points and labels, and the goal is to find a function that maps data points to labels. Take the clustering problem as an example.

Clustering Problem:

Given: Data points

Goal: Group the data points, points within each group are close to each other.

Clustering is a classical unsupervised learning problem: we are only given data points but not which clusters they are in; the hidden structure is "points within each group are close to each other". In order to formally define the "correct" clustering, we specify a *probabilistic model*.

Probabilistic modeling is a general way of describing unsupervised learning tasks. The clustering problem can be modeled as *mixture of Gaussians*. Each data point is in one of r groups. Each group has its own mean $\boldsymbol{\mu_i}$. If a point is in i-th group, the point is generated according to a Gaussian distribution with mean $\boldsymbol{\mu_i}$. Doing clustering is then equivalent to learning the groups.

This probabilistic view is very useful because it specifies how the data is generated, and provides a "ground truth" for measuring qualities of solutions. Probabilistic models are also called *latent variable models*, as the observed data points depend on some latent variables (e.g. the group numbers). Learning a latent variable model amounts to finding the values of the latent variables.

There is a multitude of machine learning models in different application areas that at first glance seem quite different from one another. Can we have a unified framework for thinking about them? Here we propose a new framework that we call General Matrix Factorization (GMF) which encompasses PCA, ICA, NMF, dictionary learning etc. (see Figure 1.1 for illustration, and Table 1.1 for a list of problems in the framework).

Definition 1.1 (General Matrix Factorization). Each General Matrix Factorization (GMF) problem has three elements:

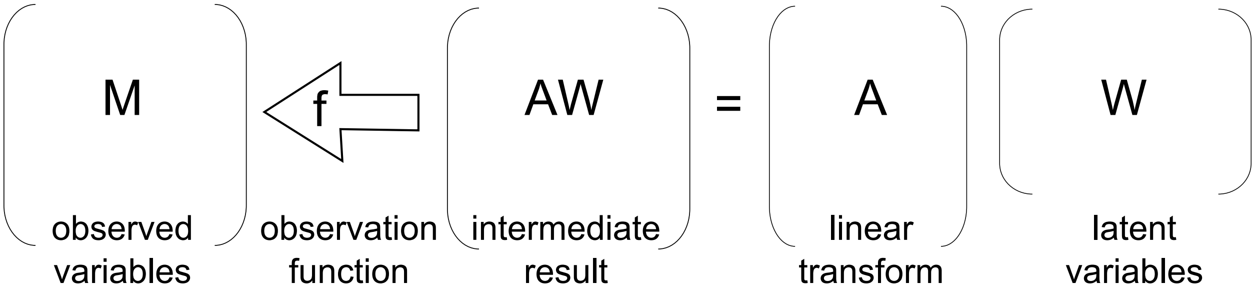


Figure 1.1: General Matrix Factorization Framework

Latent Variables

The latent variables form a matrix $W \in \mathbb{R}^{r\times m}$. The i-th column W_i of W contains latent variables related to the i-th data point. The columns are drawn independently.

Linear Transform

The i-th data point is closely related to AW_i, where the matrix $A \in \mathbb{R}^{n\times r}$ is an unknown linear transform.

Observation Function

The observed data forms a matrix M. Each data point is a column M_i of M, generated by applying a simple function f to AW_i ($M_i = f(AW_i)$).

Goal: Given M, learn matrices A and W.

Many classical learning tasks fit naturally in the GMF framework. For mixture of Gaussians, the columns of matrix W are basis vectors determining which group the data point belongs to (i.e. $W_i = \boldsymbol{e_j}$ if point i is in group j). The columns of matrix A correspond to the means of Gaussians $\boldsymbol{\mu_i}$. The observation function f simply adds Gaussian noise to the product AW.

The well-known Principal Component Analysis (PCA) tries to find a subspace where the data points have the largest variance. To reduce PCA to GMF, the hidden variables W_i are the coordinates of the data points in this subspace. The subspace itself is encoded in A (the

Problem	Observation Function	Constraints on A	Constraints on W
PCA	$M = AW + noise$	dimension $n \times r$	dimension $r \times m$
Clustering	$M = AW + noise$	dimension $n \times r$	Columns are basis vectors
ICA	$M = AW\ (+noise)$	full rank	independent non-Gaussian entries
Nonnegative Matrix Factorization	$M = AW\ (+noise)$	dimension $n \times r$ nonnegative entries	dimension $r \times m$ nonnegative entries
Topic Modeling	$M =$ sample AW	dimension $n \times r$ columns are distributions	dimension $r \times m$ priors
Dictionary Learning	$M = AW$	general	columns are sparse
Neural Net (1 layer)	$M = threshold(AW)$	general	0/1 entries
Mixed Membership Community Model	$M =$ sample$^* AW$	$A = W^\top P$	dimension $r \times m$ specific priors

Table 1.1: Machine Learning Problems in General Matrix Factorization Framework

columns of A form an orthonormal basis of the subspace). The observation function f adds small noise to the product AW.

Independent Component Analysis (ICA, [42]) is another famous example. The goal of ICA is to learn the matrix A given samples of the form $\boldsymbol{y} = A\boldsymbol{x}$, where the vector $\boldsymbol{x}$ has independent non-Gaussian entries. This is already in the GMF framework: the hidden variables are the vectors $\boldsymbol{x}$, the linear transform is matrix A, and the observation function is just the identity function $f(A\boldsymbol{x}) = A\boldsymbol{x}$.

In Table 1.1 we summarize how different machine learning problems can be reduced to the general matrix factorization framework (some problems are defined later in this Chapter).

Let us now go back to the questions we asked at the beginning. For mixture of Gaussians we want to know

> **Question:** What kind of data is easy for learning mixture of Gaussians? Are there algorithms that work provably for this kind of data?

These questions are already partly answered. The seminal work of Dasgupta[46] shows when the centers of Gaussians are far enough from each other, the problem can be solved efficiently. Follow-up works[31, 125, 141, 155] build on the same assumption, but reduce amount of separation required. In this thesis we answer similar questions for other problems.

The last five problems in Table 1.1 can be categorized into three classes according to the constraints on A and W. Each part of this thesis investigates a different class of General Matrix Factorization problem. We identify very different simplifying assumptions in each class, and design provable algorithms under such assumptions. We hope these assumptions explain why such GMF problems can be solved efficiently, and these algorithms inspire different approaches for machine learning problems.

1.1 Provable, Practical Topic Modeling Algorithm

The first part of the thesis focus on GMF problems with nonnegative constraints. *Topic modeling*, which tries to extract thematic information from large corpus of documents, is an important problem in this class.

A long line of work on topic modeling sets up the basic structure for topic models, starting from probabilistic latent semantic indexing (pLSI, [83]), followed by Latent Dirichlet Allocation[26], Correlated Topic Models[25] ,Pachinko allocation[112] and many others. According to these works, each topic is a probability distribution over words (e.g., in a topic related to “weather”, the words “snow”, “rain” and “sunny” have high probability); each document is a probability distribution over topics.

These models also assume the ordering of the words is not crucial in determining the topics (“bag-of-words” assumption). Hence words in a document are generated independently at random. To generate a word, first generate its topic according to the document-topic distribution, then pick a word from the corresponding topic-word distribution.

The process of generating a word can be summarized as a matrix product. The latent variables are the document-topic distributions. The linear transform is constructed from the topic-word distributions. That is, the (i, j)-th entry of matrix $W \in \mathbb{R}^{r\times m}$ corresponds to the probability that document j generates topic i. The (l,i)-th entry of $A \in \mathbb{R}^{n\times r}$ corresponds to the probability that the topic i generates the word l. The (l,j)-th entry of product AW is exactly the probability that document j generates word l. The observation function f samples N words according to the distribution AW_j. Therefore we have the following description of topic modeling in GMF framework.

Topic Modeling:

Given: There is an unknown topic matrix A with nonnegative entries that is dimension $n \times r$, and a stochastically generated unknown matrix W that is dimension $r \times m$. Each column of AW is viewed as a probability distribution on rows, and for each column we are given $N \ll n$ i.i.d. samples from the associated distribution.

Goal: Reconstruct A and parameters of the generating distribution for W.

Directly analyzing the General Matrix Factorization problem for topic modeling is not easy because much information is lost in the sampling process. In Chapter 2 We first focus on a simpler variant called Nonnegative Matrix Factorization (NMF).

Nonnegative Matrix Factorization:

Given: Matrix $M = AW + noise$, where matrices $A \in \mathbb{R}^{n\times r}$ and $W \in \mathbb{R}^{r\times m}$ have nonnegative entries.

Goal: Find A, W.

We show a natural assumption called "separability" [54] simplifies the problem, and give a polynomial time algorithm when the instance is separable.

Separability Assumption: A nonnegative factorization $M = AW$ is separable if for each i there is some row $r(i)$ of A that has a single nonzero entry and this entry is in the i-th column.

Under this assumption, the NMF problem has a nice geometric interpretation that leads to polynomial time algorithms.

In Chapter 3 we show how to apply similar techniques to topic modeling. For topic modeling the separability assumption naturally translates to the “anchor words assumption”.

> **Anchor Words Assumption:** Every topic has a word with probability at least p in that topic, and does not appear (has probability 0) in all other topics.

This assumption greatly simplifies the learning task. Even though we get samples from AW which form a very coarse approximation of the product, we show the noise can be reduced. Then utilizing the algorithm in Chapter 2 we give a polynomial time algorithm for topic modeling.

Although the algorithms in Chapters 2 and 3 are interesting in theory, they are very slow in practice. In Chapter 4 we redesign the algorithms seeking good performance in practice. The new algorithm is order of magnitude faster than classical topic modeling toolkit MALLET[122], and produces output with similar quality.

1.2 Towards Provably Learning Deep Representations

Deep learning is one of the hottest topics in machine learning in recent years. The basic idea of deep learning is using a multi-layer network to represent the mapping from data points to their labels. Although multi-layer neural nets have been around for a long time, they can only be effectively learned after Hinton’s fundamental work[80, 81]. Later, with new insights from many researchers (see survey by Bengio [19]), deep learning has broken many records in tasks related to voice, images and videos.

Despite its success in practice, theoretical analysis for deep learning algorithms is very limited. The main difficulties come from the nonlinearity in the network and correlations introduced by the layers of the network. Livni et al.[115] analyzed an algorithm of learning low degree polynomials, which can efficiently explain the input data. However there are no guarantees for recovering the ground truth parameters.

Traditionally, deep learning is considered a very different topic from what we've discussed so far. However, new developments have characterized the kinds of deep networks that are useful in practice, and made deep learning quite reminiscent of the other problems in GMF framework.

Although deep learning is a technique for supervised learning, its performance highly relies on an unsupervised pretraining step [81]. In this step, the algorithm tries to automatically discover hidden features in the input. These features are then used as a starting point for supervised training. The unsupervised step can be viewed as learning a "deep representation".

Researchers have also assumed that the net (or some modification) can be run *in reverse* to get a *generative model* for a distribution that is a close fit to the empirical input distribution. This idea was implicit in Hinton's work [81] as the Restricted Boltzmann Machines (RBM) he uses are naturally reversible. Later it is formalized as the idea of denoising autoencoders by Vincent et al.[156]. The generative model view of deep learning puts it into the General Matrix Factorization framework, where each layer provides a nonlinear code to represent the key structure of the data.

In the second part of the thesis we try to design algorithms for provably learning deep representations. In particular, our algorithms learn sparse representations that can be viewed as GMF problems with sparsity constraints.

In Chapter 5 we first look at the problem of learning a single-layer, sparse linear representation. This problem is usually called *dictionary learning* or *sparse coding.*

Dictionary Learning

Given: Samples $\boldsymbol{y} = A\boldsymbol{x}$ where $\boldsymbol{x}$ is a random sparse vector.

Goal: Learn matrix A and vectors $\boldsymbol{x}$.

Dictionary learning fits easily to the GMF framework: the latent variables W_i's are just the vectors $\boldsymbol{x}$; the linear transformation is the matrix A; the observation function is identity.

When the matrix A is given, dictionary learning becomes the simpler *sparse recovery* problem (closely related to *compressed sensing*[55]). Even this simpler problem is hard in the worst case ([48, 95]). However, if we look at the worst case instances more carefully, they are all in the regime where the vector $\boldsymbol{y}$ can have *multiple* sparse representations $\boldsymbol{x}$. Such ambiguity makes the representation less useful. This inspires the following assumption

> **Robustness Assumption:** The sparse representation for $\boldsymbol{y}$ is unique, and is stable when the vector $\boldsymbol{y}$ is perturbed.

Previous works [36, 53, 75] have identified sufficient conditions for the sparse representation to be unique. In Chapter 5 we work with *incoherent* assumption of Donoho and Huo [53]. Under this assumption, we show the correlation structure of samples contains enough information about the support graph of the vectors $\boldsymbol{x}$. This information is then used by a *graph recovery* algorithm to learn the supports of $\boldsymbol{x}$, and eventually leads to reconstructing the matrix A.

In Chapter 6, we show how to learn a network of *autoencoders*([82, 156]). An autoencoder is a one layer network with an encoding function E and a decoding function D. The functions D and E are usually of the form $D(\boldsymbol{x}) = s(A\boldsymbol{x} + \boldsymbol{b})$ and $E(\boldsymbol{y}) = s(A^\top \boldsymbol{y} + \boldsymbol{b'})$ where s is a nonlinear function that is applied coordinate-wise. We choose s to be the threshold function ($\mathrm{sgn}(x) = 1$ if $x > 0$ and $\mathrm{sgn}(x) = 0$ otherwise). The autoencoder is then just a one-layer neural network.

> **Learning Autoencoders (one-layer neural net)**
>
> Given: Samples $\boldsymbol{y} = \mathrm{sgn}(A\boldsymbol{x})$ where $\boldsymbol{x}$ is a random sparse vector.
>
> Goal: Learn matrix A and vectors $\boldsymbol{x}$.

Learning an autoencoder again reduces to the GMF problem: the latent variables are the vectors $\boldsymbol{x}$; the linear transformation is the matrix A; the observation function applies the threshold function sgn to the vector $A\boldsymbol{x}$.

The robustness assumption translates to the *denoising* property for autoencoders introduced by Vincent et al. [156].

> **Denoising Assumption:** Any valid code $\boldsymbol{x}$ can be recovered even if the output is perturbed: $E(D(\boldsymbol{x}) + noise) = \boldsymbol{x}$.

In Chapter 6 we show how to learn a denoising autoencoder when A is a random sparse matrix (this is inspired by Berinde et al. [21]). Our algorithm for the single-layer network is robust, so that even when the input vector $\boldsymbol{x}$ comes from a deep network, the algorithm can still learn the correct network A. Therefore we can apply this algorithm iteratively to learn the entire deep network.

1.3 Tensor Decomposition for General Matrix Factorization

In the final part of the thesis, we focus on a unified methodology for solving many GMF problems in the statistical recovery framework. The key assumption in this part is distributional:

> **Distribution Assumption:** The latent variables have a specific, simple prior. The data is generated exactly according to the model.

Statistical recovery algorithms are algorithms that work on synthetic data generated according to known distributions. There are many beautiful statistical recovery results in both theoretical computer science and machine learning, including algorithms for planted clique or dense subgraphs [125], Hidden Markov Models [88], mixture of Gaussians [128] and LDA model[9].

Most of these problems still fit into the General Matrix Factorization framework. However, the constraints on matrix A are not as strong as we have seen in the previous parts. For example, Anandkumar et al.[9] show how to learn the LDA model without using anchor words assumption[1]. However, previous statistical recovery algorithms use problem-specific techniques, and usually require a large number of samples.

In Chapter 7, we observe that in many cases it is possible to reduce General Matrix Factorization problem to the Orthogonal Tensor Decomposition problem. In fact, Orthogonal Tensor Decomposition can even be applied to settings where it is hard to formulate the problem as GMF.

Definition 1.2. Orthogonal Tensor Decomposition

Given: Tensor $\hat{T} \approx T = \sum_{i=1}^{r} \lambda_i v_i \otimes v_i \otimes v_i$, the vectors v_i are orthonormal.
Goal: Find estimates $\hat{\lambda}_i \approx \lambda_i$, $\hat{v}_i \approx v_i$.

The key insight in the reductions is the method-of-moments ([137]): first compute the moments of observed variables[2], then apply suitable linear transformation to get an orthogonal tensor. This reduction crucially relies on the Distribution Assumption.

After the reduction, the Orthogonal Tensor Decomposition problem can be solved using tensor power method. In Chapter 7 we give tight sample complexity bounds for this algorithm.

Many of the previous works (including [8, 9, 38, 87, 88, 130]) can be reinterpreted in the Orthogonal Tensor Decomposition framework. Our tight sample complexity analysis gives better bounds for learning many well known models, including Hidden Markov Model, Independent Component Analysis and simple topic models.

[1] Unlike the algorithms in Chapter 3, the algorithm does not generalize to other topic models such as Pachinko Allocation[112]

[2] The moments for an observed vector $\boldsymbol{y}$ include its mean $\mathbb{E}[\boldsymbol{y}]$, variance $\mathbb{E}[\boldsymbol{y}\boldsymbol{y}^\top]$ and higher order correlations

In Chapter 8, we highlight a new application of Orthogonal Tensor Decomposition: finding overlapping communities in social networks.

The most popular model for communities is the *stochastic block* model. In this model, every person is in a single community. Two persons in the same community know each other with probability p; two persons in different communities know each other with probability q. Many algorithms (e.g. [125], see a summary in [166]) can learn stochastic block models.

Stochastic block model is limited because every person can be in only one community. Airoldi et al.[4] purposed mixed membership stochastic block (MMSB) model. In this model, there are n persons and r communities. Each person belongs fractionally to different communities. If we use $W_{i,j}$ to denote the fraction that j-th person is involved in the i-th community, the model asserts the probability that two persons u, v know each other should be $W_u^\top P W_v$.

Learning MMSB model:

Given: There is an unknown matrix $W \in \mathbb{R}^{r\times n}$ with columns drawn from Dirichlet distribution, and an unknown matrix P. Given a graph G on n vertices, the edge (u, v) is in the graph with probability $W_u^\top P W_v$ independently.

Goal: Reconstruct W matrix.

Learning MMSB model can be reduced to GMF framework, if we assume the linear transformation has the form $W^\top P$, and the observation function f samples according to entries of AW to form the graph G. Using method-of-moments, we can also reduce the learning problem to orthogonal tensor decomposition.

The improved sample complexity analysis in Chapter 7 becomes crucial for learning MMSB model. Unlike topic models, where we can fix the topic matrix A and get as many documents as we want, in the community model the dimensions of both A and W increase as the number of persons increase. Using tensor decomposition, we can match the best known trade-off between number of communities and number of persons for the stochastic block model (even though our algorithm works for the more general MMSB model).

1.4 Other Related Works

Supervised Learning In supervised learning, the input includes data points and corresponding labels, the goal of learning is to predict the labels of new data points.

These problems are traditionally studied via the PAC (Probabilistic Approximately Correct) model of Valiant [152], and VC dimension and SVMs (Support Vector Machines) of Vapnik and others [29, 153]. However, our work has to go beyond these approaches. First, obtaining unlabeled data points has become much easier with vast data generated by the Internet, while data sets with good quality labels require significantly more effort. Also, unsupervised learning is very closely related to understanding the properties of the data. In fact, insights from deep learning [19, 81] hint that supervised learning tasks might be "easy" because the structure of the input can be learned by unsupervised learning algorithms. Without the correct structure of the input, most learning problems are computationally hard in PAC model or even simpler variants of PAC model (e.g. [93, 97]).

Semi-random Models A common critique to model assumptions and statistical recovery algorithms is that they are not robust to adversarial perturbations to the model. Semi-random models allow an adversary to perturb the instance after it is generated using a stochastic model. The first example is the planted bisection model of Feige and Kilian [60]. Recently, [118] generalized this result. Similar results are also known for graph coloring[27], finding hidden clique [61] and unique games [102].

Approximation Stability Assumptions Another line of work, initiated by Balcan, Blum and Gupta[16] focuses on problems where approximation algorithms are applied, such as k-means or k-median clustering. They assume the instances satisfy the property that if a solution has large objective function, then it should also be close to the "ground truth" solution. The reasoning behind this assumption is if it were false, then it would make no sense to apply approximation algorithms, and maybe people should seek a different objective function. Under this kind of assumption people were able to develop algorithms for various clustering-related problems.

These assumptions are quite different from our work. In particular, we try to make assumptions that natural input should satisfy, without looking into specific objective functions.

Tensor Decompositions There has been several lines of research studying tensor decomposition, including CP decomposition [78], Tucker decomposition [151] and HOSVD [50] (for a survey see [99]). Use of tensor decompositions in learning latent variable models also has a long history, some of the earlier works are summarized in the book [123].

For Orthogonal Tensor Decomposition [101], when the exact tensor is given, it is known how to decompose the tensor into sum of rank one tensors via variants of power method [100, 167]. However, before our work it is not clear whether any variant of power method works under sampling noise.

Part I

Provable, Practical Topic Modeling Algorithm

Chapter 2

Nonnegative Matrix Factorization

In the *Nonnegative Matrix Factorization* (NMF) problem, we are given an $n \times m$ matrix M with nonnegative real entries (such a matrix will be henceforth called "nonnegative") and an integer $r > 0$. Our goal is to express M as AW where A and W are nonnegative matrices of size $n \times r$ and $r \times m$ respectively. NMF arises naturally in the context of topic modeling, where the matrix product AW can concisely describe the process of generating a word from a document. In fact, NMF can be viewed as a limiting case of topic models where the number of words in a document is much larger than the size of the vocabulary.

In this chapter, we first give algorithms for solving exact NMF problem (that is, the matrix M is exactly equal to the product of AW), and an almost matching lowerbound. However, in order to apply NMF in the context of topic modeling, the algorithm needs to be robust to noise, and fully polynomial over all the parameters (n, m and r). Under a natural *separability* assumption [54], we show how to design an algorithm that meets these requirements.

2.1 Background and Main Results

2.1.1 History of NMF

Besides applications in topic modeling, NMF is itself an interesting problem that has been independently introduced in a number of different contexts and applications. Many interesting heuristics and local search algorithms (including the familiar Expectation Maximization or EM) have been proposed to find such factorizations. One compelling family of applications is data analysis, where a nonnegative factorization is computed in order to extract certain latent relationships in the data and has been applied to image segmentation [109], [110] information retrieval [83] and document clustering [162]. NMF also has applications in fields such as chemometrics [107] (where the problem has a long history of study under the name *self modeling curve resolution*) and biology (e.g. in vision research [32]): in some cases, the underlying physical model for a system has natural restrictions that force a corresponding matrix factorization to be nonnegative. In demography (see e.g., [79]), NMF is used to model the dynamics of marriage through a mechanism similar to the chemical laws of mass action. In combinatorial optimization, Yannakakis [164] characterized the number of extra variables needed to succinctly describe a given polytope as the nonnegative rank of an appropriate matrix (called the "slack matrix"). In communication complexity, Aho et al [3] showed that the log of the nonnegative rank of a Boolean matrix is polynomially related to its deterministic communication complexity - and hence the famous Log-Rank Conjecture of Lovasz and Saks [116] is equivalent to showing a quasi-polynomial relationship between real rank and nonnegative rank for Boolean matrices. In complexity theory, Nisan used nonnegative rank to prove lower bounds for non-commutative models of computation [133]. Additionally, the 1993 paper of Cohen and Rothblum [41] gives a long list of other applications in statistics and quantum mechanics. That paper also gives an exact algorithm that runs in exponential time.

2.1.2 Main Results

We give both an algorithm for accomplishing this algorithmic task that runs in polynomial time for any constant value of r and we complement this with an intractability result which states that assuming the Exponential Time Hypothesis [92] no algorithm can solve the exact NMF problem in time $(nm)^{o(r)}$.

Theorem 2.1. *There is an algorithm for the Exact NMF problem (where r is the target inner-dimension) that runs in time $O((nm)^{O(r^2 2^r)})$.*

This result is based on algorithms for deciding the first order theory of the reals - roughly the goal is to express the decision question of whether or not the matrix M has nonnegative rank at most r as a system of polynomial equations and then to apply algorithms in algebraic geometry to determine if this semi-algebraic set is non-empty. The complexity of these procedures is dominated by the number of distinct variables occurring in the system of polynomial equations. In fact, the number of distinct variables plays an analogous role to VC-dimension. The naive formulation of the NMF decision problem as a non-emptiness problem is to use $nr+mr$ variables, one for each entry in A or W [41]. Yet even for constant values of r, an algorithm based on such a formulation would run in time exponential in n and m.

At the heart of our algorithm is a structure theorem – based on a novel method for reducing the number of variables needed to define the associated semi-algebraic set. We are able to express the decision problem for nonnegative matrix factorization using $r^2 2^r$ distinct variables (and we make use of tools in geometry, such as the notion of a separable partition, to accomplish this [77], [6], [90]). Thus we obtain the algorithm quoted in the above theorem, and we note that prior to our work it was unknown whether even the case $r = 3$ was hard or solvable in polynomial time, and indeed it is the latter.

In fact, in a natural special case of the problem where the rank of M is equal to the target inner-dimension r we can obtain a further improvement: We refer to this problem as the *Simplicial Factorization* (SF) problem. See Section 2.2.1 for an explanation for why this special case is natural in the context of information retrieval and other applications where the goal is to learn latent statistical structure. Our algorithm is again based on the first order theory of the reals, but here the system of equations is much smaller so in practice one may be able to use heuristic approaches to solve this system (in which case, the validity of the solution can be easily checked).

Theorem 2.2. *There is an algorithm for the Exact SF problem (where r is the target inner-dimension) that runs in time $O((nm)^{O(r^2)})$*[1].

We complement these algorithms with a fixed parameter intractability result. We make use of a recent result of Patrascu and Williams [136] and engineer low-dimensional gadgets inspired by the gadgets of Vavasis [154] to show that under the Exponential Time Hypothesis [92], there is no exact algorithm for NMF that runs in time $(nm)^{o(r)}$. This intractability result holds also for the SF problem.

Theorem 2.3. *If there is an exact algorithm for the SF problem (or for the NMF problem) that runs in time $O((nm)^{o(r)})$ then 3-SAT can be solved in $2^{o(n)}$ time on instances with n variables.*

Although the algorithms mentioned above run in polynomial time for constant r, in practice they are still intractable even for small r. We consider the nonnegative matrix factorization problem under an assumption introduced by Donoho and Stodden [54] in the context of image segmentation called "separability". This assumption asserts that there are r rows of A that can be permuted to form a diagonal matrix. If we knew the names of these rows, then computing a nonnegative factorization would be easy. The challenge in this context, is to avoid brute-force search (which runs in time n^r) and to find these rows in time polynomial in n, m *and* r. To the best of our knowledge the following is the first example of a polynomial-time algorithm that provably works under a non-trivial condition on the input.

[1] This result is later improved in [127]

Theorem 2.4. *There is an exact algorithm that can compute a separable, nonnegative factorization $M = AW$ (where r is the inner-dimension) in time polynomial in n, m and r if such a factorization exists.*

Donoho and Stodden [54] argue that the separability condition is naturally met in the context of image segmentation. Additionally, Donoho and Stodden prove that separability in conjunction with some other conditions guarantees that the solution to the NMF problem is unique. Our theorem above is an algorithmic counterpart to their results, but requires only separability. Our algorithm can also be made noise tolerant, and hence works even when the separability condition only holds in an approximate sense. Indeed, an approximate separability condition is regarded as a fairly benign assumption and is believed to hold in many practical contexts in machine learning. For instance it is usually satisfied by model parameters fitted to various generative models (e.g. LDA [26] in information retrieval). We thank David Blei for this information.

Lastly, we consider the case in which the given matrix M does not have an exact low-rank NMF but rather can be approximated by a nonnegative factorization with small inner-dimension.

Theorem 2.5. *There is a $2^{\text{poly}(r\log(1/\epsilon))}\,\text{poly}(n,m)$-time algorithm that, given a M for which there is a nonnegative factorization AW (of inner-dimension r) which is an ϵ-approximation to M in Frobenius norm, computes A' and W' satisfying $\|M - A'W'\|_F \le O(\epsilon^{1/2}r^{1/4})\,\|M\|_F$.*

The rest of the chapter is organized as follows: In Section 2.2 we give an exact algorithm for the SF problem and in Section 2.3 we give an exact algorithm for the general NMF problem. In Section 2.4 we prove a fixed parameter intractability result for the SF problem. And in Section 2.5 we give algorithms for the separable nonnegative fatorization problems, which is particularly interesting for topic modeling. The same problem is solved again in later chapters and also in related follow-up works, these different algorithms provides a large variety of tradeoffs, which are summarized in Section 2.6. Finally we give an algorithm for approximate NMF in Section 2.7.

2.2 Simplicial Factorization

Here we consider the simplicial factorization problem, in which M has rank equal to the target inner-dimension r. Hence in any factorization, the factors A and W must have full column rank and full row rank respectively.

2.2.1 Justification for Simplicial Factorization

We first argue that the extra restriction imposed in simplicial factorization is natural in many contexts: Through a re-scaling we can assume that the columns of M, A and W all have unit ℓ_1 norm. The factorization $M = AW$ can be interpreted probabilistically: each column of M can be expressed as a convex combination (given by the corresponding column of W) of columns in A. In the example in the introduction, columns of M represent documents and the columns of A represent "topics". Hence a nonnegative factorization is an "explanation" : each document can be expressed as a convex combination of the topics. But if A does not have full column rank then this explanation is seriously deficient. This follows from a restatement of Radon's Lemma. Let $conv(A_U)$ be the convex hull of the columns A_i for $i \in U$.

Observation 1. *If A is an $n \times r$ (with $n \geq r$) matrix and $rank(A) < r$, then there are two disjoint sets of columns $U, V \subset [r]$ so that $conv(A_U) \cap conv(A_V) \neq \emptyset$.*

The observation implies that there can be a candidate document x that can be expressed as a convex combination of one set (U) of topics, or instead can be expressed as a convex combination of an entirely disjoint other set (V) of topics. The end goal of NMF is often to use the representation of documents as distributions on topics to perform various tasks, such as clustering or information retrieval. But if (even given the set of topics in a database) it is this ambiguous to determine how we should represent a given document as a convex combination of topics, then the topics we have extracted cannot be very useful for clustering! In fact, it seems unnatural to not require the columns of A to be linearly independent! One should consider the process (stochastic, presumably) that generates the columns of W. Any reasonable process would almost surely result in a matrix M whose rank is equal to the rank of A.

2.2.2 Algorithm for Simplicial Factorization

In this section we give an algorithm that solves the simplicial factorization problem in $(nm)^{O(r^2)}$ time. Let L be the maximum bit complexity of any coefficient in the input.

Theorem 2.6. *There is an $O((nm)^{O(r^2)})$ time algorithm for deciding if the simplicial factorization problem has a solution of inner-dimension at most r. Furthermore, we can compute a rational approximation to the solution up to accuracy δ in time* $\text{poly}(L, (nm)^{O(r^2)}, \log 1/\delta)$.

The above theorem is proved by using Lemma 2.7 below to reduce the problem of finding a simplicial factorization to finding a point inside a semi-algebraic set with *poly*(n) constraints and $2r^2$ real-valued variables (or deciding that this set is empty). The decision problem can be solved using the well-known algorithm of Basu et. al.[18] that solves this problem in $n^{O(r^2)}$ time. We can instead use the algorithm of Renegar [139] (and a bound of $\text{poly}(L, (nm)^{O(r^2)})$ on the bit complexity of the coefficients in the solution due to Grigor'ev and Vorobjov [73]) to compute a rational approximation to the solution up to accuracy δ in time $\text{poly}(L, (nm)^{O(r^2)}, \log 1/\delta)$.

This reduction uses the fact that since A, W have full rank they have "pseudo-inverses" A^+, W^+ which are $r \times n$ and $n \times r$ matrices respectively such that $A^+A = WW^+ = I_{r\times r}$.

Thus $A^+M_i = A^+AW_i = W_i$ and similarly $M^jW^+ = A^j$. Hence the columns of A can be obtained from linear combinations of the columns of M and similarly the rows of W can be obtain from linear combinations of rows of M. Let C be a maximal set of linearly independent columns of M and let R be a maximal set of linearly independent rows of M.

Lemma 2.7 (Structure Lemma for Simplicial Factorization). *Let M be a rank r matrix. Then M has a nonnegative matrix factorization of inner-dimension r if and only if there are linear transformations S and T such that: (i) $CSTR = M$ (ii) CS and TR are both non-negative*

Proof. ("if") Suppose the conditions in the theorem are met. Then set $A = CS$ and $W = TR$ and hese matrices are nonnegative and have size $n \times r$ and $r \times m$ respectively, and furthermore are a factorization for M. Since $rank(M) = r$, A and W are a simplicial factorization.

("only if") Conversely suppose that there is a simplicial factorization $M = AW$. Recall that because A and W have full column and row rank respectively, the columns of A can be obtained from linear combinations of the columns of M and also the rows of W can be obtain from linear combinations of rows of M. Since R and C span the row and column space of M respectively, we have that the columns of A can be obtained by a linear transformation of the columns of C and similarly W can be obtained by a linear transformation of R. □

2.3 General NMF

Now we consider the NMF problem where the factor matrices A, W need not have full rank. Although the simplicial factorization problem is a quite natural special case in applications in machine learning and statistics, indeed in other applications such as in extended formulations [164] it is crucial that the nonnegative rank of a matrix can be much different than its rank. We note that subsequent to our work, the fourth author gave a singly-exponential time algorithm for computing the nonnegative rank of a matrix [127] and coupled with our hardness results in Section 2.4 this almost characterizes the fastest algorithm we could hope for under the Exponential Time Hypothesis [92].

Theorem 2.8. *There is a $O((nm)^{O(r^2 2^r)})$ time deterministic algorithm that given an $n \times m$ nonnegative matrix M outputs a factorization AW of inner dimension r if such a factorization exists. Furthermore, we can compute a rational approximation to the solution up to accuracy δ in time* $\mathrm{poly}(L, (nm)^{O(r^2 2^r)}, \log 1/\delta)$.

As in the simplicial case the main idea will again be a reduction to an existence question for a semi-algebraic set, but this reduction is significantly more complicated than Lemma 2.7.

2.3.1 General Structure Theorem: Minimality

Our goal is to re-cast nonnegative matrix factorization (for constant r) as a system of polynomial inequalities where the number of variables is constant, the maximum degree is constant and the number of constraints is polynomially bounded in n and m. The main obstacle is that A and W are *large* - we cannot afford to introduce a new variable to represent each entry in these matrices. We will demonstrate there is always a "minimal" choice for A and W so that:

1. there is a collection of linear transformations $T_1, T_2, ...T_{g(r)}$ from the column-span of M to $\mathbb{R}^r$ and a choice function $\sigma_W : [m] \rightarrow [g(r)]$
2. and a collection of linear transformations $S_1, S_2, ...S_{g(r)}$ from the row-span of M to $\mathbb{R}^r$ and a choice function $\sigma_A : [n] \rightarrow [g(r)]$

And these linear transformations and choice functions satisfy the conditions:

1. for each $i \in [n]$, $W_i = T_{\sigma_W(i)} M_i$ and

2. for each $j \in [m]$, $A^j = M^j S_{\sigma_A(j)}$.

Furthermore, the number of possible choice functions σ_W is at most $m^{O(r^2 f(r))}$ and the number of possible choice functions for σ_A is at most $n^{O(r^2 g(r))}$. These choice functions are based on the notion of a simplicial partition, which we introduce later. We then give an algorithm for enumerating all simplicial partitions (this is the primary bottleneck in the algorithm). Fixing the choice functions σ_W and σ_A, the question of finding linear transformations $T_1, T_2, ...T_{g(r)}$ and $S_1, S_2, ...S_{g(r)}$ that satisfy the above constraints (and the constraint that $M = AW$, and A and W are nonnegative) is exactly a system of polynomial inequalities with a $O(r^2 g(r))$ variables (each matrix T_i or S_j is $r \times r$), degree at most four and furthermore there are at most $O(mn)$ polynomial constraints.

In this subsection, we will give a procedure (which given A and W) generates a "minimal" choice for A and W (call this minimal choice A' and W'), and we will later establish that this "minimal" choice satisfies the structural property stated informally above.

Definition 2.9. Let $\mathcal{C}(A) \subset 2^{[r]}$ denote the subsets of $[r]$ corresponding to maximal independent sets of columns (of A). Similarly let $\mathcal{R}(W) \subset 2^{[r]}$ denote the subsets of $[r]$ corresponding to maximal independent sets of rows (of W).

A basic fact from linear algebra is that all maximal independent sets of columns of A have exactly $rank(A)$ elements and all maximal independent sets of rows of W similarly have exactly $rank(W)$ elements.

Definition 2.10. Let $\succ_s$ be the total ordering on subsets of $[r]$ of size s so that if U and V are both subsets of $[r]$ of size s, $U \prec_s V$ iff U is lexicographically before V.

Definition 2.11. Given a column M_i, we will call a subset $U \in \mathcal{C}(A)$ a minimal basis for M_i (with respect to A) if $M_i \in cone(A_U)$ and for all $V \in \mathcal{C}(A)$ such that $M_i \in cone(A_V)$ we must have $U \prec_s V$.

Claim. *If $M_i \in cone(A)$, then there is some $U \in \mathcal{C}(A)$ such that $M_i \in cone(A_U)$.*

Definition 2.12. A proper chain (A, W, A', W') is a set of nonnegative matrices for which $M = AW$, $M = AW'$ and $M = A'W'$ (the inner dimension of these factorizations is r) and functions $\sigma_{W'} : [m] \to \mathcal{C}(A)$ and $\sigma_{A'} : [n] \to \mathcal{R}(W')$ such that

1. for all $i \in [m]$, $AW'_i = M_i$, $supp(W'_i) \subset \sigma_{W'}(i)$ and $\sigma_{W'}(i)$ is a minimal basis w.r.t. A for M_i
2. for all $j \in [n]$, $A'_j W' = M^j$, $supp(A^j) \subset \sigma_{A'}(j)$ and $\sigma_{A'}(j)$ is a minimal basis w.r.t. W' for M^j.

Note that the extra conditions on W' (i.e. the minimal basis constraint) is with respect to A and the extra conditions on A' are with respect to W'. This simplifies the proof that there is always some proper chain, since we can compute a W' that satisfies the above conditions with respect to A and then find an A' that satisfies the conditions with respect to W'.

Lemma 2.13. *If there is a nonnegative factorization $M = AW$ (of inner-dimension r), then there is a choice of nonnegative A', W' of inner-dimension r and functions $\sigma_{W'} : [m] \to \mathcal{C}(A)$ and $\sigma_{A'} : [n] \to \mathcal{R}(W')$ such that (A, W, A', W') and $\sigma_{W'}$, $\sigma_{A'}$ form a proper chain.*

Proof. The condition that there is some nonnegative W for which $M = AW$ is just the condition that for all $i \in [m]$, $M_i \in cone(A)$. Hence, for each vector M_i, we can choose a minimal basis $U \in \mathcal{C}(A)$ using Claim 2.3.1. Then $M_i \in cone(A_U)$ so there is some nonnegative vector W'_i supported on U such $AW'_i = M_i$ and we can set $\sigma_{W'}(i) = U$. Repeating this procedure for each column M_i, results in a nonnegative matrix W' that satisfies the condition $M = AW'$ and for each $i \in [m]$, by design $supp(W'_i) \subset \sigma_{W'}(i)$ and $\sigma_{W'}(i)$ is a minimal basis with respect to A for M_i.

We can re-use this argument above, setting $M^T = (W'^T)A^T$ and this interchanges the role of A and W. Hence we obtain a nonnegative matrix A' which satisfies $M = A'W'$ and for each $j \in [n]$, again by design we have that $supp(A^j) \subset \sigma_{A'}(j)$ and $\sigma_{A'}(j)$ is a minimal basis with respect to W for M^j. □

Definition 2.14. Let $\Pi(A, U)$ (for $U \in \mathcal{C}(A)$) denote the $r \times n$ linear transformation that is zero on all rows not in U (i.e. $\Pi(A, U)^j = \mathbf{0}$ for $j \notin U$) and restricted to U is $\Pi(A, U)^U = (A_U)^+$ (where the $+$ operation denotes the Moore-Penrose pseudoinverse).

Lemma 2.15. *Let (A, W, A', W') and $\sigma_{W'}$ and $\sigma_{A'}$ form a proper chain. For any index $i \in [m]$, let $U_i = \sigma_{W'}(i)$ and for any index $j \in [n]$ let $V_j = \sigma_{A'}(j)$. Then $W'_i = \Pi(A, U_i)M_i$ and $A'^j = M^j\Pi(W'^T, V_j)^T$.*

Notice that in the above lemma, the linear transformation that recovers the columns of W' is based on column subsets of A, while the linear transformation to recover the rows of A' is based on the row subsets of W' (not W).

Proof. Since (A, W, A', W') and $\sigma_{W'}$ and $\sigma_{A'}$ form a proper chain we have that $AW' = M$. Also $supp(W'_i) \subset U_i = \sigma_{W'}(i)$. Consider the quantity $\Pi(A, U_i)M_i$. For any $j \notin U_i$, $(\Pi(A, U_i)M_i)_j = 0$. So consider

$$(\Pi(A, U_i)M_i)_{U_i} = (A_{U_i})^+ AW'_i = (A_{U_i})^+ A_{U_i}(W'_i)_{U_i}$$

where the last equality follows from the condition $supp(W'_i) \subset U_i$. Since $U_i \in \mathcal{C}(A)$ we have that $(A_{U_i})^+A_{U_i}$ is the $|U_i| \times |U_i|$ identity matrix. Hence $W'_i = \Pi(A, U_i)M_i$. An identical argument with W' replaced with A' and with A replaced by W'^T (and i and U_i replaced with j and V_j) respectively implies that $A'^j = M^j\Pi(W'^T, V_j)^T$ too. □

Note that there are at most $|\mathcal{C}(A)| \leq 2^r$ linear transformations of the form $\Pi(A, U_i)$ and hence the columns of W' can be recovered by at most 2^r linear transformations of the column span of M, and similarly the rows of A' can also be recovered.

The remaining technical issue is we need to demonstrate that there are not too many choice functions $\sigma_{W'}$ and $\sigma_{A'}$ and that we can enumerate over this set efficiently. In principle, even if say $\mathcal{C}(A)$ is just two sets, there are exponentially many choices of which (of the two) linear transformation to use for each column of M. However, when we use lexicographic ordering to tie break (as in the definition of a minimal basis), the number of choice functions is polynomially bounded. We will demonstrate that the choice function $\sigma_{W'} : [m] \rightarrow \mathcal{C}(A)$ arising in the definition of a proper chain can be embedded in a restricted type of geometric partitioning of M which we call a simplicial partition.

2.3.2 General Structure Theorem: Simplicial Partitions

Here, we establish that the choice functions $\sigma_{W'}$ and $\sigma_{A'}$ in a proper chain are combinatorially simple. The choice function $\sigma_{W'}$ can be regarded as a partition of the columns of M into $|\mathcal{C}(A)|$ sets, and similarly the choice function $\sigma_{A'}$ is a partition of the rows of M into $\mathcal{R}(W')$ sets. Here we define a geometric type of partitioning scheme which we call a simplicial partition, which has the property that there are not too many simplicial partitions (by virtue of this class having small VC-dimension), and we show that the partition functions $\sigma_{W'}$ and $\sigma_{A'}$ arising in the definition of a proper chain are realizable as (small) simplicial partitions.

Definition 2.16. A (k, s)-simplicial partition of the columns of M is generated by a collection of k sets of s hyperplanes

$$\mathcal{H}^1 = \{h_1^1, h_2^1, ...h_s^1\}, \mathcal{H}^2 = \{h_1^2, h_2^2, ...h_s^2\}, ...\mathcal{H}^k = \{h_1^k, h_2^k, ...h_s^k\}.$$

Let $Q_i = \{i' \text{ s.t. for all } j \in [s], h^i_j \cdot M_{i'} \geq 0\}$. Then this collection of sets of hyperplanes results in the partition

- $P_1 = Q_1$
- $P_2 = Q_2 - P_1$
- $P_k = Q_k - P_1 - P_2 ... - P_{k-1}$
- $P_{k+1} = [m] - P_1 - P_2 ... - P_k$

If $rank(A) = s$, we will be interested in a $(\binom{r}{s}, s)$-simplicial partition.

Lemma 2.17. *Let (A, W, A', W') and $\sigma_{W'}$ and $\sigma_{A'}$ form a proper chain. Then the partitions corresponding to $\sigma_{W'}$ and to $\sigma_{A'}$ (of columns and rows of M respectively) are a $(\binom{r}{s}, s)$-simplicial partition and a $(\binom{r}{t}, t)$-simplicial partition respectively, where $rank(A) = s$ and $rank(W') = t$.*

Proof. Order the sets in $\mathcal{C}(A)$ according to the lexicographic ordering $\succ_s$, so that $V_1 \prec_s V_2 \prec_s ... V_k$ for $k = |\mathcal{C}(A)|$. Then for each j, let $\mathcal{H}^j$ be the rows of the matrix $(A_{V_j})^+$. Note that there are exactly $rank(A) = s$ rows, hence this defines a (k, s)-simplicial partition.

Claim. *$\sigma_{W'}(i) = j$ if and only if $M_i \in P_j$ in the (k, s)-simplicial partition generated by $\mathcal{H}^1, \mathcal{H}^2, ...\mathcal{H}^k$.*

Proof. Since (A, W, A', W') and $\sigma_{W'}$ and $\sigma_{A'}$ forms a proper chain, we have that $M = AW'$. Consider a column i and the corresponding set $V_i = \sigma_{W'}(i)$. Recall that V_j is the j^{th} set in $\mathcal{C}(A)$ according to the lexicographic ordering $\succ_s$. Also from the definition of a proper chain V_i is a minimal basis for M_i with respect to A. Consider any set $V_{j'} \in \mathcal{C}(A)$ with $j' < j$. Then from the definition of a minimal basis we must have that $M_i \notin cone(A_{V_{j'}})$. Since $V_{j'} \in \mathcal{C}(A)$, we have that the transformation $(A_{V_{j'}})(A_{V_{j'}})^+$ is a projection onto $span(A)$

which contains $span(M)$. Hence $(A_{V_{j'}})(A_{V_{j'}})^+M_i = M_i$, but $M_i \notin cone(A_{V_{j'}})$ so $(A_{V_{j'}})^+M_i$ cannot be a nonnegative vector. Hence M_i is not in $P_{j'}$ for any $j' < j$. Furthermore, M_i is in Q_j: using Lemma 2.15 we have $\Pi(A, V_j)M_i = \Pi(A, V_j)AW'_i = W'_i \geq \mathbf{0}$ and so $(A_{V_j})^+M_i = (\Pi(A, V_j)M_i)_{V_j} \geq \mathbf{0}$. □

We can repeat the above replacing A with W'^T and W' with A', and this implies the lemma. □

2.3.3 Enumerating Simplicial Partitions

Here we give an algorithm for enumerating all (k, s)-simplicial partitions (of, say, the columns of M) that runs in time $O(m^{ks(r+1)})$. An important observation is that the problem of enumerating all simplicial partitions can be reduced to enumerating all partitions that arise from a single hyperplane. Indeed, we can over-specify a simplicial partition by specifying the partition (of the columns of M) that results from each hyperplane in the set of ks total hyperplanes that generates the simplicial partition. From this set of partitions, we can recover exactly the simplicial partition.

A number of results are known in this domain, but surprisingly we are not aware of any algorithm that enumerates all partitions of the columns of M (by a single hyperplane) that runs in polynomial time (for $dim(M) \leq r$ and r is constant) without some assumption on M. For example, the VC-dimension of a hyperplane in r dimensions is $r + 1$ and hence the Sauer-Shelah lemma implies that there are at most $O(m^{r+1})$ distinct partitions of the columns of M by a hyperplane. In fact, a classic result of Harding (1967) gives a tight upper bound of $O(m^r)$. Yet these bounds do not yield an algorithm for efficiently enumerating this structured set of partitions without checking *all* partitions of the data.

A recent result of Hwang and Rothblum [90] comes close to our intended application. A separable partition into p parts is a partition of the columns of M into p sets so that the convex hulls of these sets are disjoint. Setting $p = 2$, the number of separable partitions is exactly the number of distinct hyperplane partitions. Under the condition that M is in general position (i.e. there are no t columns of M lying on a dimension $t-2$ subspace where $t = rank(M) - 1$), Hwang and Rothblum give an algorithm for efficiently enumerating all distinct hyperplane partitions [90].

Here we give an improvement on this line of work, by removing any conditions on M (although our algorithm will be slightly slower). The idea is to encode each hyperplane partition by a choice of not too many data points. To do this, we will define a slight generalization of a hyperplane partition that we will call a hyperplane separation:

Definition 2.18. To a hyperplane h we associate a mapping which we call a *hyperplane separation* that maps columns of M to $\{-1, 0, 1\}$ depending on the sign of $h \cdot M_i$ (where the sign function is 1 for positive values, -1 for negative values and 0 for zero).

A hyperplane partition can be regarded as a mapping from columns of M to $\{-1, 1\}$ where we adopt the convention that M_i such that $h \circ M_i$ is mapped to 1.

Definition 2.19. A hyperplane partition (defined by h) is an extension of a hyperplane separation (defined by g) if for all i, $g(M_i) \neq 0 \Rightarrow g(M_i) = h(M_i)$.

Lemma 2.20. *Let $rank(M) = s$, then for any hyperplane partition (defined by h), there is a hyperplane g that contains s affinely independent columns of M and for which h (as a partition) is an extension of g (as a separation).*

Proof. After an appropriate linear transformation (of the columns of M and the hyperplanes), we can assume that M is full rank. If the h already contains s affinely independent columns of M, then we can choose $g = h$. If not we can perturb h in some direction so that for any column with $h(M_i) = 0$, we maintain the invariant that M_i is contained on the perturbed hyperplane h'. Since $rank(M) = s$ this perturbation has non-zero inner product with some column in M and so this hyperplane h' will eventually contain a new column from M (without changing the sign of $h(M_i)$ for any other column). We can continue this argument until the hyperplane contains s affinely independent columns of M and by design on all remaining columns agrees in sign with h. □

Lemma 2.21. *Let $rank(M) = s$. For any hyperplane h (which defines a partition), there is a collection of $k \leq s$ sets of (at most s) columns of M, $S_1, S_2, ..S_k$ so that any hyperplanes $g_1, g_2, ..g_k$ which contain $S_1, S_2, ...S_k$ respectively satisfy: For all i, $h(M_i)$ (as a partition) is equal to the value of $g_j(M_i)$, where j is the smallest index for which $g_j(M_i) \neq 0$. Furthermore these subsets are nested: $S_1 \supset S_2 \supset ... \supset S_k$.*

Proof. We can apply Lemma 2.20 repeatedly. When we initially apply the lemma, we obtain a hyperplane g_1 that can be extended (as a separation) to the partition corresponding to h. In the above function (defined implicitly in the lemma) this fixes the partition of the columns except those contained in g_1. So we can then choose M' to be the columns of M that are contained in g_1, and recurse. If S_2 is the largest set of columns output from the recursive call, we can add columns of M contained in g_1 to this set until we obtain a set of $s + 1$ affinely independent columns contained in g_1, and we can output this set (as S_1). □

Theorem 2.22. *Let $rank(M) = s$. There is an algorithm that runs in time $O(m^s(s + 2)^s)$ time to enumerate all hyperplane partitions of the columns of M.*

Proof. We can apply Lemma 2.21 and instead enumerate the sets of points $S_1, S_2, ...S_s$. Since these sets are nested, we can enumerate all choices as follows:

- choose at most s columns corresponding to the set S_1
- initialize an active set $T = S_1$

- until T is empty either
  - choose a column to be removed from the active set
  - or indicate that the current active set represents the next set S_i and choose the sign of the corresponding hyperplane

There are at most $O(m^s(s+2)^s)$ such choices, and for each choice we can then run a linear program to determine if there is a corresponding hyperplane partition. (In fact, all partitions that result from the above procedure will indeed correspond to a hyperplane partition). The correctness of this algorithm follows from Lemma 2.21. □

This immediately implies:

Corollary 2.23. *There is an algorithm that runs in time $O(m^{ks^2})$ that enumerates a set of partitions of the columns of M that contains the set of all (k,s)-simplicial partitions (of the columns of M).*

2.3.4 Solving Systems of Polynomial Inequalities

The results of Basu et al [18] give an algorithm for finding a point in a semi-algebraic set defined by $O(mn)$ constraints on polynomials of total degree at most d, and $f(r)$ variables in time $O((mnd)^{cf(r)})$. Using our structure theorem for nonnegative matrix factorization, we will re-cast the decision problem of whether a nonnegative matrix M has nonnegative rank r as an existence question for a semi-algebraic set.

Theorem 2.24. *There is an algorithm for deciding if a $n \times m$ nonnegative matrix M has nonnegative rank r that runs in time $O((nm)^{O(r^2 2^r)})$. Furthermore, we can compute a rational approximation to the solution up to accuracy δ in time* $\mathrm{poly}(L, (nm)^{O(r^2 2^r)}, \log 1/\delta)$.

We first prove the first part of this theorem using the algorithm of Basu et al [18], and we instead use the algorithm of Renegar [139] to compute a rational approximation to the solution up to accuracy δ in time $\text{poly}(L, (nm)^{O(r^2 2^r)}, \log 1/\delta)$.

Proof. Suppose there is such a factorization. Using Lemma 2.13, there is also a proper chain. We can apply Lemma 2.17 and using the algorithm in Theorem 2.22 we can enumerate over a superset of simplicial partitions. Hence, at least one of those partitions will result in the choice functions $\sigma_{W'}$ and $\sigma_{A'}$ in the proper chain decomposition for $M = AW$.

Using Lemma 2.15 there is a set of at most 2^r linear transformations $T_1, T_2, ...T_{2^r}$ which recover columns of W' given columns of M, and similarly there is a set of at most 2^r linear transformations $S_1, S_2, ...S_{2^r}$ which recover the rows of A' given rows of M. Note that these linear transformations are from the column-span and row-span of M respectively, and hence are from subspaces of dimension at most r. So apply a linear transformation to columns of M and one to rows of M to to recover matrices M_C and M_R respectively (which are no longer necessarily nonnegative) but which are dimension $r \times m$ and $n \times r$ respectively. There will still be a collection of at most 2^r linear transformations from columns of M_C to columns of W', and similarly for M_R and A'.

We will choose r^2 variables for each linear transformation, so there are $2 * r^2 * 2^r$ variables in total. Then we can write a set of m linear constraints to enforce that for each column of $(M_C)_i$, the transformation corresponding to $\sigma_{W'}(i)$ recovers a nonnegative vector. Similarly we can define a set of n constraints based on rows in M_R.

Lastly we can define a set of constraints that enforce that we do recover a factorization for M: For all $i \in [m], j \in [n]$, let $i' = \sigma_{W'}(i)$ and $j' = \sigma_{A'}(j)$. Then we write the constraint $(M_C)^j S_{j'} T_{i'} (M_R)_i = M_i^j$. This constraint has degree at two in the variables corresponding to the linear transformations. Lemma 2.13 implies that there is some choice of these transformations that will satisfy these constraints (when we formulate these constraints using the correct choice functions in the proper chain decomposition). Furthermore, any set of transformations that satisfies these constraints does define a nonnegative matrix factorization of inner dimension r for M.

And of course, if there is no inner dimension r nonnegative factorization, then all calls to the algorithm of Basu et al [18] will fail and we can return that there is no such factorization.

□

The result in Basu et al. [18] is a quantifier elimination algorithm in the Blum, Shub and Smale (BSS) model of computation [28]. The BSS model is a model for real number computation and it is natural to ask what is the bit complexity of finding a rational approximation of the solutions. There has been a long line of research on the decision problem for first order theory of reals: given a quantified predicate over polynomial inequalities of reals, determine whether it is true or false. What we need for our algorithm is actually a special case of this problem: given a set of polynomial inequalities over real variables, determine whether there exists a set of values for the variables so that all polynomial inequalities are satisfied. In particular, all variables in our problem are quantified by existential quantifier and there are no alternations. For this kind of problem Grigor'ev and Vorobjov [73] first gave a singly-exponential time algorithm that runs in $(nd)^{O(f(r)^2)}$ where n is the number of polynomial inequalities, d is the maximum degree of the polynomials and $f(r)$ is the number of variables. The bit complexity of the algorithm is $\mathrm{poly}(L, (nd)^{O(f(r)^2)})$ where L is the maximum length of the coefficients in the input. Moreover, their algorithm also gives an upperbound of $\mathrm{poly}(L, (nd)^{O(f(r))})$ on the number of bits required to represent the solutions. Renegar[139] gave a better algorithm that for the special case we are interested in takes time $(nd)^{O(f(r))}$. Using his algorithm with binary search (with search range bounded by Grigor'ev et.al.[73]), we can find rational approximations to the solutions with accuracy up to δ in time $\mathrm{poly}(L, (nm)^{O(f(r))}, \log 1/\delta)$.

We note that our results on the SF problem are actually a special case of the theorem above (because our structural lemma for simplicial factorization is a special case of our general structure theorem):

Corollary 2.25. *There is an algorithm for determining whether the nonnegative rank of a nonnegative $n \times m$ matrix M equals the rank and this algorithm runs in time $O((nm)^{O(r^2)})$.*

Proof. If $rank(M) = r$, then we know that both A and W must be full rank. Hence $\mathcal{C}(A)$ and $\mathcal{R}(W)$ are both just the set $\{1, 2, ...r\}$. Hence we can circumvent the simplicial partition machinery, and set up a system of polynomial constraints in at most $2r^2$ variables. □

2.4 Strong Intractability of Simplicial Factorization

Here we prove that there is no algorithm for computing simplicial factorization of dimension r that runs in $(nm)^{o(r)}$ time unless 3-SAT can be solved in $2^{o(n)}$ time. Surprisingly, even the NP-hardness of the problem was only proved quite recently by Vavasis [154]. That reduction is the inspiration for our result, though unfortunately we were unable to use it directly to get low-dimensional instances. Instead we give a new reduction using the d-SUM Problem.

Definition 2.26 (d-SUM)**.** In the d-SUM problem we are given a set of N values $\{s_1, s_2, ...s_N\}$ each in the range $[0, 1]$, and the goal is to determine if there is a set of d numbers (not necessarily distinct) that sum to exactly $d/2$.

This definition for the d-SUM Problem is slightly unconventional in that here we allow repetition (i.e. the choice of d numbers need not be distinct). Patrascu and Williams [136] recently proved that if d-SUM can be solved in $N^{o(d)}$ time then 3-SAT has a sub-exponential time algorithm. In fact, in the instances constructed in [136] we can allow repetition of numbers without affecting the reduction since in these instances choosing any number more than once will never result in a sum that is exactly $d/2$. Hence we can re-state the results in [136] for our (slightly unconventional definition for) d-SUM.

Theorem 2.27. *If $d < N^{0.99}$ and if d-SUM instances of N distinct numbers each of $O(d \log N)$ bits can be solved in $N^{o(d)}$ time then* 3*-SAT on n variables can be solved in time $2^{o(n)}$.*

Given an instance of the d-SUM, we will reduce to an instance of the Intermediate Simplex problem defined in [154].

Definition 2.28 (Intermediate Simplex)**.** Given a polyhedron $P = \{x \in \mathbb{R}^{r-1} : Hx \geq b\}$ where H is an $n \times (r-1)$ size matrix and $b \in \mathbb{R}^n$ such that the matrix $[H, b]$ has rank r and a set S of m points in $\mathbb{R}^{r-1}$, the goal of the Intermediate Simplex Problem is to find a set of points T that form a simplex (i.e. T is a set of r affinely independent points) each in P such that the convex hull of T contains the points in S.

Vavasis [154] proved that Intermediate Simplex is equivalent to the Simplicial Factorization problem.

Theorem 2.29 (Vavasis [154])**.** *There is a polynomial time reduction from Intermediate Simplex problem to Simplicial Factorization problem and vice versa and furthermore both reductions preserve the value of r.*

2.4.1 The Gadget

Given the universe $U = \{s_1, s_2, \ldots, s_N\}$ for the d-SUM problem, we construct a two dimensional Intermediate Simplex instance as shown in Figure 2.1. We will show that the Intermediate Simplex instance has exactly N solutions, each representing a choice of s_i. Later in the reduction we use d such gadgets to represent the choice of d numbers in the set U.

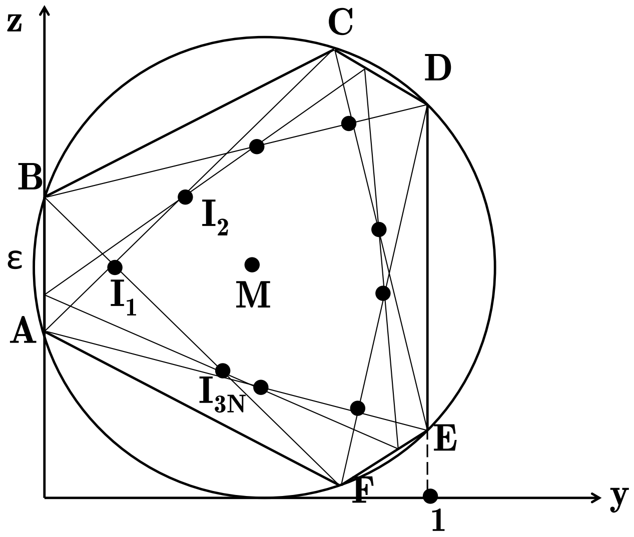


Figure 2.1: The Gadget

Recall for a two dimensional Intermediate Simplex problem, the input consists of a polygon $\mathcal{P}$ (which is the hexagon $ABCDEF$ in Figure 2.1) and a set of points $S = \{I_1, I_2, \ldots, I_{3N}\}$ inside $\mathcal{P}$ (which are the dots, except for M). A solution to this two dimensional Intermediate Simplex instance will be a triangle inside $\mathcal{P}$ such that all the points in S are contained in the triangle (in Figure 2.1 ACE is a valid solution).

We first specify the polygon $\mathcal{P}$ for the Intermediate Simplex instance. The polygon $\mathcal{P}$ is just the hexagon $ABCDEF$ inscribed in a circle with center M. All angles in the hexagon are $2\pi/3$, the edges $AB = CD = EF = \epsilon$ where ϵ is a small constant depending on N, d that we determine later. The other 3 edges also have equal lengths $BC = DE = FA$.

We use $y(A)$ and $z(A)$ to denote the y and z coordinates for the point A (and similarly for all other points in the gadget). The hexagon is placed so that $y(A) = y(B) = 0$, $y(D) = y(E) = 1$.

Now we specify the set S of $3N$ points for the Intermediate Simplex instance. To get these points first take N points in each of the 3 segements AB, CD, EF. On AB these N points are called A_1, A_2, ..., A_N, and $|AA_i| = \epsilon s_i$. Similarly we have points C_i's on CD

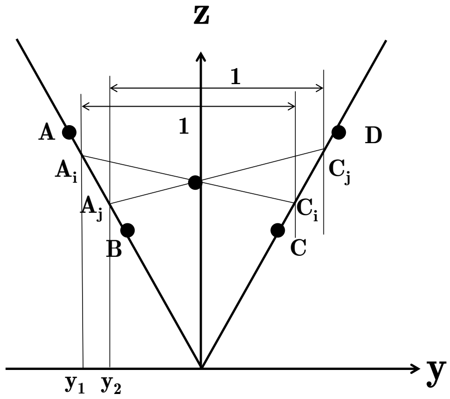


Figure 2.2: Proof of Lemma 2.30

and E_i's on EF, $|CC_i| = |EE_i| = \epsilon s_i$. Now we have N triangles $A_iC_iE_i$ (the thin lines in Figure 2.1). We claim (see Lemma 2.30 below) that the intersection of these triangles is a polygon with $3N$ vertices. The points in S are just the vertices of this intersection.

Lemma 2.30. *When $\epsilon < 1/50$, the points $\{A_i\}$, $\{C_i\}$, $\{E_i\}$ are on AB, CD, EF respectively and $AA_i = CC_i = EE_i = \epsilon s_i$, the intersection of the N triangles $\{A_iC_iE_i\}$ is a polygon with $3N$ vertices.*

Proof. Since the intersection of N triangles $A_iC_iE_i$ is the intersection of $3N$ halfplanes, it has at most $3N$ vertices. Therefore we only need to prove every edge in the triangles has a segment remaining in the intersection. Notice that the gadget is symmetric with respect to rotations of $2\pi/3$ around the center M. By symmetry we only need to look at edges A_iC_i. The situation here is illustrated in Figure 2.2.

Since all the halfplanes that come from triangles $A_iC_iE_i$ contain the center M, later when talking about halfplanes we will only specify the boundary line. For example, the halfplane with boundary A_iC_i and contains E_i (as well as M) is called halfplane A_iC_i.

The two thick lines in Figure 2.2 are extensions of AB and CD, now they are rotated so that they are $z = \pm\sqrt{3}y$. The two thin lines are two possible lines A_iC_i and A_jC_j. The differences between y coordinates of A_i and C_i are the same for all i (here normalized to 1) by the construction of the points A_i's and C_i's. Assume the coordinates for A_i, A_j are $(y_i, -\sqrt{3}y_i)$ and $(y_j, -\sqrt{3}y_j)$ respectively. Then the coordinates for the intersection is $(y_i+y_j+1, \sqrt{3}(1+y_i+y_j+2y_iy_j))$. This means if we have N segments with $y_1 < y_2 < \ldots < y_N$, segment i will be the highest one when y is in range $(y_{i-1} + y_i + 1, y_i + y_{i+1} + 1)$ (indeed, the lines with $j > i$ have higher slope and will win when $y > y_i + y_j + 1 \geq y_i + y_{i+1} + 1$; the lines with $j < i$ have lower slope and will win when $y < y_i + y_j + 1 \leq y_i + y_{i-1} + 1$).

We also want to make sure that all these intersection points are inside the halfplanes C_iE_i's and E_iA_i's. Since $\epsilon < 1/50$, all the y_i's are within $[-1/2 - 1/20, -1/2 + 1/20]$. Hence the intersection point is always close to the point $(0, \sqrt{3}/2)$, the distance is at most $1/5$. At the same time, since ϵ is small, the distances of this point $(0, \sqrt{3}/2)$ to all the C_iE_i's and E_iA_i's are all larger than $1/4$. Therefore all the intersection points are inside the other $2N$ halfplanes and the segments will indeed remain in the intersection. The intersection has $3N$ edges and $3N$ vertices. □

The Intermediate Simplex instance has N obvious solutions: the triangles $A_iC_iE_i$, each one corresponds to a value s_i for the d-SUM problem. In the following Lemma we show that these are the only possible solutions.

Lemma 2.31. *Let $\epsilon < 1/1000$. If the solution of the Intermediate Simplex problem is PQR, then PQR must be one of the $A_iC_iE_i$'s.*

Proof. Suppose PQR is a solution of the Intermediate Simplex problem, since M is in the convex hull of $\{I_1, I_2, \ldots, I_{3N}\}$, it must be in PQR. Thus one of the angles $\angle PMQ$, $\angle QMR$, $\angle RMP$ must be at least $2\pi/3$ (their sum is 2π). Without loss of generality we assume this angle is $\angle PMQ$ and by symmetry assume P is either on AB or BC. We shall show in

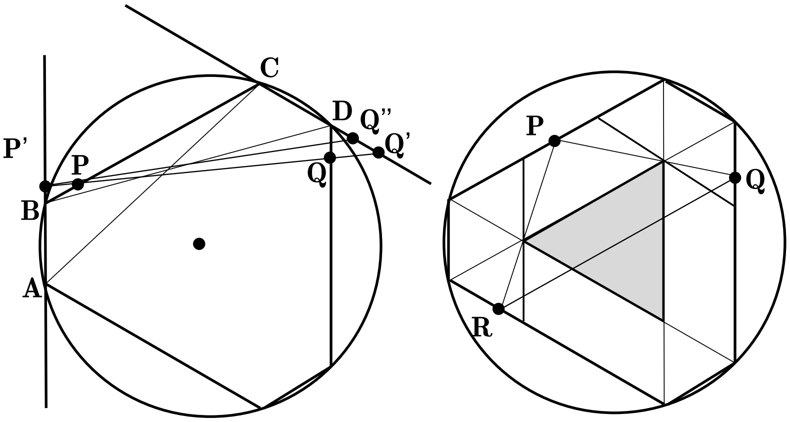


Figure 2.3: Proof of Lemma 2.31

either of the two cases, when P is not one of the A_i's, there will be some I_k that is not in the halfplane PQ (recall the halfplanes we are interested in always contain M so we don't specify the direction).

When P is on AB, since $\angle PMQ \geq 2\pi/3$, we have $CQ \geq AP$ (by symmetry when $CQ = AP$ the angle is exactly $2\pi/3$). This means we can move Q to Q' such that $CQ' = AP$. The intersection of halfplane PQ' and the hexagon $ABCDEF$ is at least as large as the intersection of halfplane PQ and the hexagon. However, if P is not any of the points $\{A_i\}$ (that is, $|PQ'|/\epsilon \notin \{s_1, s_2, ..., s_N\}$), then PQ' can be viewed as $A_{N+1}C_{N+1}$ if we add $s_{N+1} = |AP|/\epsilon$ to the set U. By Lemma 2.30 introducing PQ' must increase the number of vertices. One of the original vertices I_k is not in the hyperplane PQ', and hence not in PQR. Therefore when P is on AB it must coincide with one of the A_i's, by symmetry PQR must be one of $A_iC_iE_i$'s.

When P is on BC, there are two cases as shown in Figure 2.3.

First observe that if we take $U' = U \cup \{1 - s_1, 1 - s_2, \dots, 1 - s_N\}$, and generate the set $S = \{I_1, I_2, \dots, I_{6N}\}$ according to U', then the gadget is further symmetric with respect to flipping along the perpendicular bisector of BC. Now without loss of generality $BP \leq BC/2$. Since every I_k is now in the intersection of $2N$ triangles, in particular they are also in the intersection of the original N triangles, it suffices to show one of I_k ($k \in [6N]$) is outside halfplane PQ.

The first case (left part of Figure 2.3) is when $BP < \epsilon$. In this case we extend PQ to get intersection on AB (P') and intersection on CD (Q'). Again since $\angle PMQ \geq 2\pi/3$, we have $DQ \geq BP$. At the same time we know $\angle DQQ' \geq \angle P'PB$, so $DQ' > BP'$. Similar to the previous case, we take Q'' so that $CQ'' = AP'$. The intersection of hyperplane $P'Q''$ and the hexagon $ABCDEF$ is at least as large as the intersection of halfplane PQ and the hexagon. When $\epsilon < 1/1000$, we can check $AP' < 2\epsilon \ll 1/50$, therefore we can still view $P'Q''$ as some $A_{2N+1}C_{2N+1}$ for $s_{2N+1} < 2$. Now Lemma 2.30 shows there is some vertex I_k not in halfplane $P'Q''$ (and hence not in halfplane PQ).

The final case (right part of Figure 2.3) is when $BP \geq \epsilon$. In this case we notice the triangle with 3 edges AD, BE, CF (the shaded triangle in the figure) is contained in every $A_iC_iE_i$, thus it must also be in PQR. However, since $BC/2 \geq BP \geq \epsilon$, we know $AR \leq \epsilon$ and $DQ \leq \epsilon$. In this case PQR does not even contain the center M. □

2.4.2 The Reduction

Suppose we are given an instance of the d-SUM Problem with N values $\{s_1, s_2, ...s_N\}$. We will give a reduction to an instance of Intermediate Simplex in dimension $r - 1 = 3d + 1$.

To encode the choice of d numbers in the set $\{s_1, s_2, ..., s_N\}$, we use d gadgets defined in Section 2.4.1. The final solution of the Intermediate Simplex instance we constructed will include solutions to each gadget. As the solution of a gadget always corresponds to a number in $\{s_1, s_2, ..., s_N\}$ (Lemma 2.31) we can decode the solution and get d numbers, and we use an extra dimension w that "computes" the sum of these numbers and ensures the sum is equal to $d/2$.

We use three variables $\{x_i, y_i, z_i\}$ for the i^{th} gadget.

Variables 1. *We will use $3d + 1$ variables: sets $\{x_i, y_i, z_i\}$ for $i \in [d]$ and w.*

Constraints 1 (Box)**.** *For all* $i \in [d]$, $x_i, y_i \in [0,1]$, $z_i \in [0,2]$ *and also* $w \in [0,1]$.

Definition 2.32. Let $G \subset \mathbb{R}^2$ be the hexagon ABCDEF in the two-dimensional gadget given in the Section 2.4.1.Let $H \subset \mathbb{R}^3$ be the set $conv(\{(x_i, y_i, z_i) \in \mathbb{R}^3 | (y_i, z_i) \in G, x_i = 1\}, \mathbf{0})$.

H is a tilted-cone that has a hexagonal base G and has an apex at the origin.

Definition 2.33. Let R be a 7×3 matrix and $b \in \mathbb{R}^7$ so that $\{x | Rx \geq b\} = H$.

We will use these gadgets to define (some of the) constraints on the polyhedron P in an instance of intermediate simplex:

Constraints 2 (Gadget)**.** *For each* $i \in [d]$, $R(x_i, y_i, z_i) \geq b$.

Hence when restricted to dimensions x_i, y_i, z_i the i^{th} gadget G is on the plane $x_i = 1$.

We hope that in a gadget, if we choose three points corresponding to the triangle for some value s_i, that of these three points only the point on the AB line will have a non-zero value for w and that this value will be s_i. The points on the lines CD or EF will hopefully have a value close to zero. We add constraints to enforce these conditions:

Constraints 3 (CE)**.** *For all* $i \in [d]$, $w \leq 1 - y_i + (1 - x_i)$

These constraints make sure that points on CD or EF cannot have large w value.

Recall that we use $z(A)$ to denote the z coordinate of A in the gadget in Section 2.4.1.

Constraints 4 (AB)**.** *For all* $i \in [d]$: $w \in \left[\frac{(z_i - z(A)x_i)}{\epsilon} \pm (\frac{10}{\epsilon} y_i + (1 - x_i))\right]$

Theses constraints make sure that points on AB have values in $\{s_1, s_2, ..., s_N\}$.

The AB and CE constraints all have the property that when $x_i < 1$ (i.e. the corresponding point is off of the gadget on the plane $x_i = 1$) then these constraints gradually become relaxed.

To make sure the gadget still works, we don't want the extra constraints on w to rule out some possible values for x_i, y_i, z_i's. Indeed we show the following claim.

Claim. *For all points in $(x_i, y_i, z_i) \in H$, there is some choice of $w \in [0, 1]$ so that x_i, y_i, z_i and w satisfy the CE and AB Constraints.*

The proof is by observing that Constraints AB have almost no effect when $y > 0$ and Constraints CE have no effect when $y = 0$.

Constraints 1 to 4 define a polyhedron P in $3d + 1$-dimensional space and furthermore the set of constraints that define P have full rank (in fact even the inequalities in the Box Constraints have full rank). Thus this polyhedron is a valid polyhedron for the Intermediate Simplex problem.

Next we specify the points in S for the Intermediate Simplex problem(each of which will be contained in the polyhedron P). Let I_k (for $k \in [3N]$) be the set S in the gadget in Section 2.4.1. As before, let $z(I_k)$ and $y(I_k)$ be the z and y coordinates of I_k respectively.

Definition 2.34 (w-max(I_k))**.** Let w-max(I_k) be the maximum possible w-value of any point I with $x_i = 1$, $y_i = y(I_k)$, $z_i = z(I_k)$ and $x_j, y_j, z_j = 0$ for all $j \neq i$ so that I is still contained in P.

Definition 2.35 (O, W, I_k^i, Q)**.** The set S of points for the Intermediate Simplex problem is

O **points:** For all $i \in [d]$, $x_i, y_i, z_i = 0$ and $w = 0$

W **point:** For all $i \in [d]$, $x_i, y_i, z_i = 0$ and $w = 1$

I_k^i **points:** For each $i \in [d]$, for each $k \in [3N]$ set $x_i = 1/4$, $y_i = 1/4y(I_k)$, $z_i = 1/4z(I_k)$ and for $j \neq i$ set $x_j, y_j, z_j = 0$. Also set w to be the $1/4 \times w$-max(I_k).

Q **point:** For each $i \in [d]$, $x_i = 1/d$, $y_i = y(M)/d$, $z_i = z(M)/d$ and $w = 1/6$

This completes the reduction of 3-SUM to intermediate simplex, and next we establish the COMPLETENESS and SOUNDNESS of this reduction.

2.4.3 Completeness and Soundness

The completeness part is straight forward: for i^{th} gadget we just select the triangle that corresponds to s_{k_i}.

Lemma 2.36. *If there is a set $\{s_{k_1}, s_{k_2}, ...s_{k_d}\}$ of d values (not necessarily distinct) such that $\sum_{i\in[d]} s_{k_i} = d/2$ then there is a solution to the corresponding Intermediate Simplex Problem.*

Proof. We will choose a set of $3d+2$ points T: We will include the O and W points, and for each s_{k_i}, we will choose the triangle corresponding to the value s_{k_i} in the i^{th} gadget. Recall the triangle is $A_{k_i}C_{k_i}E_{k_i}$ in the gadget defined in Section 2.4.1. The points we choose have $x_i = 1$ and y_i, z_i equal to the corresponding point in the gadget. We will set w to be s_{k_i} for the point on the line AB and we will set w to be zero for the other two points not contained in the line AB. The rest of the dimensions are all set to 0.

Next we prove that the convex hull of this set of points T contains all the points in S: The points O and W are clearly contained in the convex hull of T (and are in fact in T!). Next consider some point I_k^i in S corresponding to some intersection point I_k in the gadget G. Since I_k is in the convex hull of the triangle corresponding to s_{k_i} in the gadget G, there is a convex combination of the these three points $A_{k_i}, C_{k_i}, E_{k_i}$ in T (which we call J) so that $1/4J$ matches I_k^i on all coordinates except possibly the w-coordinate. Furthermore the point J has some value in the coordinate corresponding to w and this must be at most the corresponding value in I_k^i (because we chose the w-value in I_k^i to be $1/4 \times w\text{-max}(I_k)$). Hence we can distribute the remaining $3/4$ weight among the O and W points to recover I_k^i exactly on all coordinates.

Lastly, we observe that if we equally weight all points in T (except O and W) we recover the point Q. In particular, the w coordinate of Q should be $\frac{1}{3d}\sum_{i=1}^{d} s_{k_i} = 1/6$. □

Next we prove SOUNDNESS for our reduction. Suppose the solution is T, which is a set of $3d+2$ points in the polyhedron P and the convex hull of points in T contains all the O, W, I_k^i, Q points (in Definition 2.35).

Claim. *The points O and W must be in the set T.*

Proof. The points O and W are vertices of the polyhedron P and hence cannot be expressed as a convex combination of any other set of points in P. □

Now we want to prove the rest of the $3d$ points in set T is partitioned into d triples, each triple belongs to one gadget. Set $T' = T - \{O\} - \{W\}$.

Definition 2.37. For $i \in [d]$, let

$$T'_i = \{Z \in T' | j \neq i \Rightarrow x_j(Z), y_j(Z), z_j(Z) = 0 \text{ and one of } x_i(Z), y_i(Z), z_i(Z) \neq 0\}$$

Claim. *The sets T'_i partition T' and each contain exactly 3 nodes.*

Proof. The sets T'_i are disjoint, and additionally each set T'_i must contain at least 3 nodes (otherwise the convex hull of T'_i even restricted to x_i, y_i, z_i cannot contain the points I_k^i). This implies the Claim. □

Recall the gadget in Section 2.4.1 is a two dimensional object, but it is represented as a three dimensional cone in our construction. We would like to apply Lemma 2.31 to points on the plane $x_i = 1$ (in this plane the coordinates y_i,z_i act the same as y, z in the gadget).

Definition 2.38. For each point $Z \in T_i'$, let $ext(Z) \in \mathbb{R}^3$ be the intersection of the line connecting the origin and $(x_i(Z), y_i(Z), z_i(Z))$ with the $x_i = 1$ base of the set $\{(x_i, y_i, z_i) | R(x_i, y_i, z_i) \geq b\}$. Let $ext(T_i')$ be the point-wise *ext* operation applied to each point in T_i'.

Since the points I_k^i are in the affine hull of T_i' when restricted to x_i, y_i, z_i , we know $ext(I_k^i)$ must be in the convex hull of $ext(T_i')$. Using Lemma 2.31 in Section 2.4.1, we get:

Corollary 2.39. *$ext(T_i')$ must correspond to some triangle $A_{k_i}C_{k_i}E_{k_i}$ for some value s_{k_i}.*

Now we know how to decode the solution T and get the numbers s_{k_i}. We will abuse notation and call the 3 points in T_i' A_{k_i}, C_{k_i}, E_{k_i} (they were used to denote the corresponding points in the 2-d gadget in Section 2.4.1).We still want to make sure the w coordinate correctly "computes" the sum of these numbers. As a first step we want to show that the x_i of all points in T_i' must be 1 (we need this because the Constraints AB and CE are only strict when $x_i = 1$).

Lemma 2.40. *For each point $Z \in T_i'$, $x_i(Z) = 1$*

Proof. Suppose, for the sake of contradiction, that $x_i(Z) < 1$ (for $Z \in T_i'$). Then consider the point Q. Since $\sum_{i\in[d]} x_i(Q) = 1$, and for any point in T $\sum_{i\in[d]} x_i \leq 1$, there is no convex combination of points in T that places non-zero weight on Z and equals Q.

Let T_i'' be $T_i' \backslash \{Z\}$, we observe that the points in T_i'' are the only points in T that have any contribution to (x_i, y_i, z_i) when we want to represent Q (using a convex combination). For now we restrict our attention to these three dimensions.When trying to represent Q we must have $1/d$ weight in the set T_i'' (because of the contribution in x_i coordinate). The y_i, z_i coordinates of Q are $y(M)/d$, $z(M)/d$ respectively. This means if we take projection to y_i, z_i plane M must be in the convex hull of T_i''. However that is impossible because no two points in $A_kC_kE_k$ contain M in their convex hull. This contradiction implies the lemma. □

Lemma 2.41. *Any convex combination of points in T that equals the point Q must place equal weight on all points in T'.*

Proof. Using Lemma 2.40, we conclude that the total weight on points in T_i' is exactly $1/d$, and there is a unique convex combination of the points T_i' (restricted to y_i, z_i) that recover the point M which is the $1/3, 1/3, 1/3$ combination. This implies the Lemma. □

Now we are ready to compute the w value of the point Q and show the sum of s_{k_i} is indeed $d/2$.

Lemma 2.42 (Soundness). *When $\epsilon < N^{-Cd}$ for some large enough constant C, if there is a solution to the Intermediate Simplex instance, then there is a choice of d values that sum up to exactly $d/2$.*

Proof. As we showed in previous Lemmas, the solution to the Intermediate Simplex problem must contain O, W, and for each gadget i the solution has 3 points T_i' that correspond to one of the solutions of the gadget. Suppose for gadget i the triangle we choose is $A_{k_i}C_{k_i}E_{k_i}$. By Constraints AB we know $w(A_{k_i}) = s_{k_i}$, by Constraints CE we know $w(C_{k_i}) \leq \epsilon$ and $w(E_{k_i}) \leq \epsilon$.

By Lemma 2.41 there is only one way to represent Q, and $w(Q) = \frac{1}{3d}\sum_{i=1}^{d}[w(A_{k_i}) + w(C_{k_i}) + w(E_{k_i})] = 1/6$. Then $\sum_{i=1}^{d} s_{k_i} = \sum_{i=1}^{d} w(A_{k_i}) = \frac{d}{2} - \sum_{i=1}^{d}[w(C_{k_i}) + w(E_{k_i})]$. Since $w(C_{k_i})$ and $w(E_{k_i})$'s are small, we have $\sum_{i=1}^{d} s_{k_i} \in [d/2 - 2d\epsilon, d/2]$. However the numbers only have $O(d \log N)$ bits and ϵ is so small, the only valid value in the range is $d/2$. Hence the sum $\sum_{i=1}^{d} s_{k_i}$ must be equal to $d/2$. □

2.5 Fully-Efficient Factorization under Separability

Earlier, we gave algorithms for NMF, and presented evidence that no $(nm)^{o(r)}$ time algorithm exists for determining if a matrix M has nonnegative rank at most r. Here we consider conditions on the input that allow the factorization to be found in time polynomial in n, m and r. (In Section 2.5.1, we give a noise-tolerant version of this algorithm). To the best of our knowledge this is the first example of an algorithm (that runs in time poly(n, m, r)) and provably works under a non-trivial condition on the input. Donoho and Stodden [54] in a widely-cited paper identified sufficient conditions for the factorization to be unique (motivated by applications of NMF to a database of images) but gave no algorithm for this task. We give an algorithm that runs in time poly(n, m, r) and assumes only one of their conditions is met (separability). We note that this separability condition is quite natural in its own right, since it is usually satisfied [23] by model parameters fitted to various generative models (e.g. LDA [26] in information retrieval).

Definition 2.43 (Separability)**.** A nonnegative factorization $M = AW$ is called *separable* if for each i there is some row $f(i)$ of A that has a single nonzero entry and this entry is in the i^{th} column.

Let us understand this condition at an intuitive level in context of clustering documents by topic, which was discussed in the introduction. Recall that there a column of M corresponds to a document. Each column of A represents a topic and its entries specify the probability that a word occurs in that topic. The NMF thus "explains" the i^{th} document as AW_i where the column vector W_i has (nonnegative) coordinates summing to one—in other words, W_i represents a convex combination of topics. In practice, the total number of words n may number in the thousands or tens of thousands, and the number of topics in the dozens. Thus it is not unusual to find factorizations in which each topic is flagged by a word that appears only in that topic and not in the other topics [23]. The separability condition asserts that this happens for every topic[2].

[2] More realistically, the word may appear in other topics only with negligible property instead of zero probability. This is allowed in our noise-tolerant algorithm later.

For simplicity we assume without loss of generality that the rows of M are normalized to have unit ℓ_1-norm. After normalizing M, we can still normalize W (while preserving the factorization) by re-writing the factorization as $M = AW = (AD)(D^{-1}W)$ for some $r \times r$ nonnegative matrix D. By setting $D_{i,i} = |W^i|_1$ the rows of $D^{-1}W$ will all have l_1 norm 1. When rows of M and W are all normalized the rows of A must also have unit ℓ_1-norm because

$$1 = |M^i|_1 = |\sum_{j=1}^{r} A_{i,j}W^j|_1 = \sum_{j=1}^{r} A_{i,j}|W^j|_1 = \sum_{j=1}^{r} A_{i,j}.$$

The third equality uses the nonnegativity of W. Notice that after this normalization, if a row of A has a unique nonzero entry (the rows in Separability), that particular entry must be one.

We also assume W is a simplicial matrix defined as below.

Definition 2.44 (simplicial matrix)**.** A nonnegative matrix W is *simplicial* if no row in W can be represented in the convex hull of the remaining rows in W.

The next lemma shows that without loss of generality we may assume W is simplicial.

Lemma 2.45. *If a nonnegative matrix M has a separable factorization AW of inner-dimension at most r then there is one in which W is simplicial.*

Proof. Suppose W is not simplicial, and let the j^{th} row W^j be in the convex hull of the remaining rows. Then we can represent $W^j = \boldsymbol{u}^T W$ where $\boldsymbol{u}$ is a nonnegative vector with $|\boldsymbol{u}|_1 = 1$ and the j^{th} coordinate is 0.

Now modify A as follows. For each row $A^{j'}$ in A that has a non-zero j^{th} coordinate, we zero out the j^{th} coordinate and add $A^{j'}_j \boldsymbol{u}$ to the row $A^{j'}$. At the end the matrix is still nonnegative but whose j^{th} column is all zeros. So delete the j^{th} column and let the resulting $n \times (r-1)$ matrix be A'. Let W' be the matrix obtained by deleting the j^{th} row of W. Then by construction we have $M = A'W'$. Now we claim A' is separable.

Since A was originally separable, for each column index i there is some row, say the $f(i)^{th}$ row, that has a non-zero entry in the i^{th} column and zeros everywhere else. If $i \neq j$ then by definition the above operation does not change the $f(i)^{th}$ row of A. If $i = j$ the j^{th} index is deleted at the end. In either case the final matrix A' satisfies the separability condition.

Repeating the above operation for all violations of the simplicial condition we end with a separable factorization of M (again with inner-dimension at most r) where W is simplicial.

□

Theorem 2.46. *There is an algorithm that runs in time polynomial in n, m and r and given a matrix M outputs a separable factorization with inner-dimension at most r (if one exists).*

Proof. We can apply Lemma 2.45 and assume without loss of generality that there is a factorization $M = AW$ where A is separable and W is simplicial. The separability condition implies that every row of W appears among the rows of M. Thus W is hiding in plain sight in M; we now show how to find it.

Say a row M^j is a *loner* if (ignoring other rows that are copies of M^j) it is not in the convex hull of the remaining rows. The simplicial condition implies that the rows of M that correspond to rows of W are loners.

Claim. *A row M^j is a loner iff M^j is equal to some row W^i*

Proof. Suppose (for contradiction) that a row in M^j is not a loner and but it is equal to some row W^i. Then there is a set S of rows of M so that M^j is in their convex hull and furthermore for all $j' \in S$, $M^{j'}$ is not equal to M^j. Thus there is a nonnegative vector $u \in \mathbb{R}^n$ that is 0 at the j^{th} coordinate and positive on indices in S such that $u^T M = M^j$.

Hence $u^T AW = M^j = W^i$, but $u^T A$ must have unit ℓ_1-norm (because $|u|_1 = 1$, all rows of A have unit ℓ_1-norm and are all nonnegative), also $u^T A$ is non-zero at position j'. Consequently W^i is in the convex hull of the other rows of W, which yields a contradiction.

Conversely if a row M^j is not equal to any row in W, we conclude that M^j is in the convex hull of the rows of W. Each row of W appears as a row of A (due to the separability condition). Hence M^j is not a loner because M^j is in the convex hull of rows of M that are equivalent to M^j itself. □

Using linear programming, we can determine which rows M^j are loners. Due to separability there will be exactly r different loner rows, each corresponds to one of the W^i. Thus we are able to recover W' that is equal to W after permutation over rows.We can compute a nonnegative A' such that $A'W' = M$, and such solution A' is necessarily separable (since it is just equal to A after permutation over columns). □

2.5.1 Adding Noise

In any practical setting the data matrix M will not have an exact NMF of low inner dimension since its entries are invariably subject to noise. Here we consider how to extend our separability-based algorithm to work in presence of noise. We assume that the input matrix M' is obtained by perturbing each row of M by adding a vector of ℓ_1-norm at most ϵ, where M has a separable factorization of inner-dimension r. Alternatively, $|M'^i - M^i|_1 \leq \epsilon$ for all i. Notice that the case in which the separability condition is only approximately satisfied is a subcase of this: If for each column there is some row in which that column's entry is at least $1 - \epsilon$ and the sum of the other row entries is less than ϵ then the matrix M' will satisfy the condition stated above. (Note that M, A, W have been scaled as discussed above.)

Our algorithm will require one more condition – namely, we require the unknown matrix W to be "robustly" simplicial instead of just simplicial.

Definition 2.47 (α-robust simplicial)**.** We call W *α-robust simplicial* if no row in W has ℓ_1 distance smaller than α to the convex hull of the remaining rows in W. (Here all rows have unit ℓ_1-norm.)

Recall from Lemma 2.45 that the simplicial condition can be assumed without loss of generality under separability. In general α-robust simplicial condition does not follow from separability. However, any reasonable generative model would surely posit that the matrix W —whose columns after all represents distributions—satisfies the condition above. For instance, if columns of W are picked randomly from the unit ℓ_1 ball then after normalization α is more than 1/10. Regardless of whether or not one self-identifies as a bayesian, it seems reasonable that any suitably generic way of picking column vectors would tend to satisfy the α-robust-simplicial property.

Theorem 2.48. *Suppose $M = AW$ where A is separable and W is α-robust simplicial. Let ϵ satisfy $20\epsilon/\alpha + 13\epsilon < \alpha$. Then there is a polynomial time algorithm that given M' such that for all rows $|M'^i - M^i|_1 < \epsilon$, finds a nonnegative matrix factorization $A'W'$ of the same inner dimension such that the ℓ_1 norm of each row of $M' - A'W'$ is at most $5\epsilon/\alpha + 4\epsilon$.*

Proof. Separability implies that for any column index i there is a row $f(i)$ in A whose only nonzero entry is in the i^{th} column. Then $M^{f(i)} = W^i$ and consequently $|M'^{f(i)} - W^i|_1 < \epsilon$. Let us call these rows $M'^{f(i)}$ for all i the *canonical rows.*

Similar to Theorem 2.46, we will show every row is close to convex hull of canonical rows. On the other hand, if we remove rows that are close to a canonical row, then the canonical row will be far from the convex hull of other rows. From the above description the proof of the following claim is immediate since the rows of M can be expressed as a convex combination of W^i's.

Claim. *Every row M'^j has ℓ_1-distance at most 2ϵ to the convex hull of canonical rows.*

Proof. Since M^j can be represented as $M^j = \sum_{k=1}^r A_{j,k} M^{f(k)}$, we have

$$\begin{aligned}|M'^j - \sum_{k=1}^r A_{j,k} M'^{f(k)}|_1 &\le |M'^j - M^j|_1 + |M^j - \sum_{k=1}^r A_{j,k} M^{f(k)}|_1 + |\sum_{k=1}^r A_{j,k}(M^{f(k)} - M'^{f(k)})|_1 \\ &\le |M'^j - M^j|_1 + \sum_{k=1}^r A_{j,k} |M^{f(k)} - M'^{f(k)}|_1 \le \epsilon + (\sum_{k=1}^r A_{j,k})\epsilon = 2\epsilon.\end{aligned}$$

Here we are just using triangle inequalities and the fact that rows of A have unit ℓ_1 norm. □

Next, we show how to find the canonical rows. For a row M'^j, we call it a *robust-loner* if upon ignoring rows whose ℓ_1 distance to M'^j is less than $d = 5\epsilon/\alpha + 2\epsilon$, the ℓ_1-distance of M'^j to the convex hull of the remaining rows is more than 2ϵ. Note that we can identify robust-loner rows using linear programming. The following two claims establish that a row of M'^j is a robust-loner if and only if it is close to some row W^i.

Claim. *If M'^j has distance more than $d + \epsilon$ to all of the W^i's, then it cannot be a robust loner.*

Proof. Such an M'^j has distance at least d to each of the canonical rows. The previous claim shows M'^j is close to the convex hull of the canonical rows and thus by definition it cannot be a robust-loner. □

Claim. *All canonical rows are robust-loners.*

Proof. Since $|M'^{f(i)} - W^i|_1 \le \epsilon$, when we check if $M'^{f(i)}$ is a robust-loner (using linear programming), we leave out of consideration all rows that have ℓ_1-distance at most $5\epsilon/\alpha + \epsilon$ to W^i. In particular, this omits any row M'^j such that $M^j = \sum_{k=1}^r A_{j,k} W^k$ and $A_{j,i} \ge 1 - 5\epsilon/\alpha$. All remaining rows have $A_{j,i} \le 1 - 5\epsilon/\alpha$.

Suppose the point in $conv(M^j : j$ is a remaining row) that is closest to $M^{f(i)}$ can be represented by $\sum_{t=1}^m c_t M^t = \sum_{k=1}^r a_k W^k$ (where $\sum_{t=1}^m c_t = 1$ and $c_t \geq 0$), then we know $a_i = \sum_{t=1}^m c_t A_{j,i} \leq 1 - 5\epsilon/\alpha$.

The distance between $M^{f(i)}$ and $conv(M^j : j$ is a remaining row) is therefore bounded by

$$|M^{f(i)} - \sum_{t=1}^m c_t M^t|_1 = |(1-a_i)W^i - \sum_{k\in[r],k\neq i} a_k W^k|_1 = (1-a_i)|W^i - \sum_{k\in[r],k\neq i} \frac{a_k}{1-a_i} W^k|_1 \geq \frac{5\epsilon}{\alpha} \cdot \alpha = 5\epsilon.$$

The inequality follows from α-robust simplicial property, because the vector inside the ℓ_1 norm is the difference between W_i and a vector in the convex hull of other columns.

Since rows of M' are close to M, let $\sum_{t=1}^m c'_t M'^t$ be the closest point of $M'^{f(i)}$ in $conv(M'^j :$ j is a remaining row), we must have

$$\begin{aligned}|M'^{f(i)} - \sum_{t=1}^m c'_t M'^t|_1 &\geq |M^{f(i)} - \sum_{t=1}^m c'_t M^t|_1 - |M'^{f(i)} - M^{f(i)}|_1 - \sum_{t=1}^m c'_t |M'^t - M^t|_1 \\ &\geq 5\epsilon - \epsilon - \epsilon = 3\epsilon.\end{aligned}$$

Therefore $M'^{f(i)}$ is a robust-loner. □

The previous claim implies that each robust-loner row is within ℓ_1-distance $d + \epsilon$ to some W^i and conversely, for every W^i there is at least one robust-loner row that is close to it. Since the ℓ_1-distances between W^i's are at least $4(d+\epsilon)$, we can apply distance based clustering on the robust-loner rows: place two robust-loner rows into the same cluster if and only if these rows are within ℓ_1-distance at most $2(d+\epsilon)$. Clearly two robust-loners corresponding to the same W^i has to be in the same cluster because their distance is smaller than $2(d+\epsilon)$. On the other hand, the distance between any two robust-loners corresponding to different W^i's is more than $4(d+\epsilon) - 2(d+\epsilon) = 2(d+\epsilon)$, they will never be in the same cluster.

Therefore we will obtain r clusters, one corresponding to each of the W^i's. Choose one row from each of the cluster (call the row that is close to W^i $M'^{g(i)}$), and using similar argument as Claim 2.5.1, we have

$$\begin{aligned}
|M'^j - \sum_{k=1}^{r} A_{j,k} M'^{g(k)}|_1 &\le |M'^j - M^j|_1 + |M^j - \sum_{k=1}^{r} A_{j,k} W^k|_1 + |\sum_{k=1}^{r} A_{j,k}(W^k - M'^{g(k)})|_1 \\
&\le |M'^j - M^j|_1 + 0 + \sum_{k=1}^{r} A_{j,k} |W^k - M'^{g(k)}|_1 \\
&\le \epsilon + (\sum_{k=1}^{r} A_{j,k})(d+\epsilon) = d + 2\epsilon.
\end{aligned}$$

This implies that every row of M' is within $d + 2\epsilon = 5\epsilon/\alpha + 4\epsilon$ to the convex hull of the rows we selected. Therefore these rows form a nonnegative W' and we can find A' so that $|M'^j - (A'W')^j|_1 \le 5\epsilon/\alpha + 4\epsilon$ for all j. □

2.6 Related Works for Separable-NMF

As the previous section pointed out, the NMF problem with separability assumption is equivalent to finding the vertices of a simplex, given noisy versions of the vertices and points within the simplex (but which data points came from vertices is unknown).

This problem has been studied in the context of hyperspectral unmixing, where separability assumption is called "pure pixel assumption". Since the separable NMF problem is very relevant to topic modeling, many subsequent works give new algorithms for separable NMF. Even within this thesis, we give two alternative algorithms in Chapter 3 and Chapter 4 with different guarantees and trade-offs.

Here we try to summarize the related works. As this is a widely studied problem and new algorithms are still being developed, the summary is not complete.

There are several aspects in the trade-offs between different algorithms

Provable Guarantee The best thing to hope for is that the algorithm works provably even when the input data is perturbed (all three algorithms in this paper have this kind of guarantee). Later in the table summarizing results we call such algorithms PROVABLE. However, there are some algorithms that work in practice, and we only know how to prove them when we are given exact points from the simplex (later in the table these algorithms are marked with EXACT). Some algorithms are not known to work even given exact points, we label these as UNKNOWN. There is an interesting special case between PROVABLE and EXACT, where the algorithm is guaranteed to find the vertices even when the input data contains perturbed points from the simplex, but the algorithm assumes an additional condition: any point that is not a vertex has some minimum distance to the closest vertex. These algorithms are labeled CONDITIONAL.

Norm of Perturbation/Guarantee Among the PROVABLE and CONDITIONAL algorithms, usually there is a theorem stating that "if the input is perturbed in A norm, the algorithm is guaranteed to find points that are close to the vertices in B norm". For example, A norm and B norm can be either ℓ_1 and ℓ_2. There is a special case for B norm, which is particularly relevant in the topic modeling setting: the norm is ϵ, if when the recovered vertex is represented as a convex combination of the true vertices, the largest component is at least $1-\epsilon$. In a sense this is the ℓ_1 norm when we perform linear transformation so that the simplex becomes the probability simplex. We call this norm the Probability norm.

Given the norm of perturbation, the exact amount of perturbation that can be tolerated is also different in different algorithms, however they usually only differ by a polynomial factor of r. Also, if an algorithm uses ℓ_1 perturbation, its amount of perturbation is usually related to the ℓ_1 condition number; if an algorithm uses ℓ_2 perturbation, its amount of perturbation is usually related to the traditional (ℓ_2) condition number. We don't go into the details of these guarantees. Among all the ℓ_1 algorithms, our LP-based algorithms achieves the optimal bound on error tolerance (the optimal bound appears in [66]), however they are not very practical.

The ℓ_2 algorithms are usually not applicable when the input is not from a simplex (all the vertices are in a lower dimensional space), the ℓ_1 algorithms usually work as long as the input forms a convex polytope. However, there are several ℓ_2 algorithms that can potentially work for non-simplex inputs.

Running Time Different algorithms have very different running time. The exact running time really depends on how the algorithm is implemented (for example several algorithms can be implemented in parallel), therefore it is hard to compare without running extensive simulations. Here we only give a rough idea on the running time, based on the number of linear programs used. The algorithms that uses polynomial number of LPs are usually not practical. The algorithms with a single LP are already practical, but may not be the fastest algorithms known. The algorithms that do not use any LPs (which are labeled COMBINATORIAL) are usually really fast in practice. However, keep in mind the real performance of the algorithms depend highly on implementation, and it is not clear that the LP-based algorithms are always slower than the combinatorial algorithms.

Parameters Required Different algorithms may require different parameters. Typical parameters to require are the noise ϵ and the number of topics r. There might be ways to remove dependency on various parameters for different algorithms (for example, it seems the algorithm in previous section depend on condition number α, but it can be removed). The parameter dependencies are from the algorithms written in individual papers, the author does not claim that there are no ways of removing any of them.

Table 2.1: Summary of NMF Algorithms

Algorithm	Provability	Perturbation/Guarantee	Running Time	Parameters
Section 2.5	PROVABLE	ℓ_1/ℓ_1	n LPs	ϵ
Section 3.5	PROVABLE	$\ell_1/Probability$	n^2 LPs	ϵ
Section 4.2	PROVABLE	ℓ_2/ℓ_2	Comb.	r^*
VCA[132]	EXACT		Comb.	r
NFINDR[69]	UNKNOWN		Comb.	r
Hottopixx[22]	COND	ℓ_1/ℓ_1	1 LP	ϵ, r
GillisVavasis[65]	PROVABLE	ℓ_2/ℓ_2	Comb.	r^*
Gillis[66]	PROVABLE	ℓ_1/ℓ_1	1 LP	ϵ, r
GillisLuce[67]	PROVABLE	ℓ_1/ℓ_1	1 LP	ϵ
XRAY[104]	EXACT $^+$	ℓ_2, non-simplex	Comb. $^+$	r
Ding et al.[52]	COND	ℓ_2/ℓ_2 non-simplex?	Comb.	$\epsilon, r^\dagger$

$*$: These dependencies on r can be removed in theory by spectral gap, but in practice the spectrum does not have a noticeable gap.

$+$: The authors claim there will be a proof for the perturbed case. Also, although this algorithm does not use LPs, it does use some other convex program (of smaller scale). The authors claim this algorithm is faster than Hottopixx.

$\dagger$: The parameter r appears in the input parameter of their algorithm description, but it is not used extensively. The authors also claim their algorithm works for the non-simplex case, this is indeed true but the proof only works on simplex case and has to be modified.

2.7 Approximate Nonnegative Matrix Factorization

Here we consider the case in which the given matrix does not have an exact low-rank NMF but rather can be approximated by a nonnegative factorization with small inner-dimension. We refer to this as *Approximate* NMF. Unlike the algorithm in Theorem 2.48, the algorithm here works with general nonnegative matrix factorization: we do not make any assumptions on matrices A and W.

Theorem 2.49. *Let M be an $n \times m$ nonnegative matrix such that there is a factorization AW satisfying $\|M - AW\|_F \le \epsilon \|M\|_F$, where A and W are nonnegative and have inner-dimension r. There is an algorithm that computes A' and W' satisfying*

$$\|M - A'W'\|_F \le O(\epsilon^{1/2} r^{1/4}) \|M\|_F$$

in time $2^{\text{poly}(r \log(1/\epsilon))} \text{poly}(n, m)$.

Note that the matrix M need not have low rank, but we will be able to assume M has rank at most r without loss of generality: Let M' be the best rank at most r approximation (in terms of Frobenius norm) to M. This can be computed using a truncated singular value decomposition (see e.g. [68]). Since A and W have inner-dimension r, we get:

Claim. $\|M' - M\|_F \leq \|M - AW\|_F$

Throughout this section, we will assume that the input matrix M has rank at most r - since otherwise we can compute M' and solve the problem for M'. Then using the triangle inequality, any good approximation to M' will also be a good approximation to M.

Throughout this section, we will use the notation A_t to denote the t^{th} column of A and W^t to denote the t^{th} row of W. Note that W^t is a row vector so we will frequently use $A_t W^t$ to denote an outer-product. Next, we apply a simple re-normalization that will allow us to state the main steps in our algorithm in a more friendly notation.

Lemma 2.50. *We can assume without loss of generality that for all t*

$$\|W^t\| = 1 \tag{2.1}$$

$$\|A_t\| \leq (1+\epsilon)\|M\|_F \tag{2.2}$$

and furthermore $\|A\|_F \leq (1+\epsilon)\|M\|_F$.

Proof. We can write $AW = \sum_{t=1}^r A_t W^t$. So we may scale A_t, W^t to ensure that $\|W^t\| = 1$. Next, since A and W are nonnegative we have $\|AW\|_F \geq \|A_t W^t\|_F = \|A_t\|\|W^t\|$ and $\|AW\|_F \leq (1+\epsilon)\|M\|_F$ and this implies the first condition in the lemma.

Next we observe that

$$\|AW\|_F^2 = \sum_{i=1}^{n}\sum_{j=1}^{m}\Big[\sum_{t=1}^{r}(A_tW^t)_{i,j}\Big]^2 \geq \sum_{t=1}^{r}\sum_{i=1}^{n}\sum_{j=1}^{m}[A_tW^t]_{i,j}^2 = \|A\|_F^2,$$

where the inequality follows because all entries in A and W are nonnegative, and the last equality follows because $\|W^t\| = 1$. □

Note that this lemma immediately implies that $\|W\|_F \leq \sqrt{r}$.

The intuition behind our algorithm is to decompose the unknown matrix W as the sum of two parts: $W = W_0 + W_1$. The first part W_0 is responsible for how good AW is as an approximation to M (i.e., $\|M - AW_0\|_F$ is small) but could be negative; the second part W_1 has little effect on the approximation but is important in ensuring the sum $W_0 + W_1$ is nonnegative. The algorithm will find good approximations to W_0, W_1.

What are W_0, W_1? Since removing W_1 has little effect on how good AW is as an approximation to M, this matrix should be roughly the projection of W onto the "less significant" singular vectors of A. Namely, let the singular value decomposition of A be $A = \sum_{t=1}^{r}\sigma_t \boldsymbol{u_t}\boldsymbol{v_t}^T$ and suppose that $\sigma_1 \geq \sigma_2 .... \geq \sigma_r$. Let t_0 be the largest t for which $|\sigma_t| \geq \delta\|M\|_F$ (where δ is a constant that is polynomially related to r and ϵ and will be specified later). Then set

$$W_0 = \sum_{t=1}^{t_0}(\boldsymbol{v_t}\boldsymbol{v_t}^T)W; \qquad W_1 = \sum_{t=t_0+1}^{r}(\boldsymbol{v_t}\boldsymbol{v_t}^T)W. \tag{2.3}$$

Lemma 2.51. $\|M - AW_0\|_F \leq \epsilon\|M\|_F + \delta\sqrt{r}\|M\|_F$

Proof. By the triangle inequality $\|M - AW_0\|_F \leq \|M - AW\|_F + \|AW_1\|_F$. Also $AW_1 = \sum_{t=t_0+1}^{r}\sigma_t(\boldsymbol{u_t}\boldsymbol{v_t}^T)W$, so we have

$$\|AW_1\|_F = \|\sum_{t=t_0+1}^{r}\sigma_t(\boldsymbol{u_t}\boldsymbol{v_t}^T)W\|_F \leq \|\sum_{t=t_0+1}^{r}\sigma_t(\boldsymbol{u_t}\boldsymbol{v_t}^T)\|_2\|W\|_F \leq \delta\|M\|_F\sqrt{r},$$

where the last inequality follows because $\|W\|_F \leq \sqrt{r}$ and the spectral norm of $\sum_{t=t_0+1}^{r}(\boldsymbol{u_t v_t^T})$ is one. □

Next, we establish a lemma that will be useful when searching for (an approximation to) W_0:

Lemma 2.52. *There is an $r \times m$ matrix W_0' such that each row is in the span of the rows of M and which satisfies $\|W_0' - W_0\|_F \leq 2\epsilon/\delta$.*

Proof. Consider the matrix $A^+ = \sum_{t=1}^{t_0} \frac{1}{\sigma_t}\boldsymbol{v_t u_t^T}$. Thus A^+ is a pseudo-inverse of the truncated SVD of A. Note that $W_0 = A^+AW$ and the spectral norm $\|A^+\|_2$ is at most $1/(\delta\|M\|_F)$. Then we can choose $W_0' = A^+M$. Clearly, each row of W_0' is in the span of the rows of M. Furthermore, we have

$$\|W_0' - W_0\|_F = \|A^+(M - AW)\|_F \leq \|A^+\|\|M - AW\|_F \leq \frac{1}{\delta\|M\|_F} \cdot 2\epsilon\|M\|_F \leq \frac{2\epsilon}{\delta}.$$

And this completes the proof of the lemma. □

Lemma 2.53. *There is an algorithm that in time $2^{\mathrm{poly}(r\log(1/\epsilon))}\,\mathrm{poly}(n,m)$ finds W_0'', W_1' and A' such that $W_0'' + W_1' \geq 0$, $A' \geq 0$ and $\|M - A'(W_0'' + W_1')\|_F \leq O(\frac{\epsilon}{\delta}\|A\|_F + \epsilon\|M\|_F + \delta\sqrt{r}\|M\|_F)$.*

Proof. We use exhaustive enumeration to find a close approximation to the matrix W_0' of Lemma 2.52, and then we use convex programming to find W_1', A':

The exhaustive enumeration is simple: try all vectors that lie in some ϵ_1-net in the span of the rows of M, where $\epsilon_1 = \epsilon/\delta$. Such an ϵ_1-net is easily enumerated in the provided time since the row vectors are smaller than $\|W\|_F = \sqrt{r}$ and their span is r-dimensional.

Contained in this net there must an W_0'' such that $\|A^+M - W_0''\|_F \le \epsilon_1$. Using Lemma 2.52, $\|W_0 - W_0'\| \le 2\epsilon/\delta$, so the triangle inequality implies $\|W_0 - W_0''\|_F \le 2\epsilon/\delta + \epsilon_1 \le 4\epsilon/\delta$.

Next, we give a method to find suitable substitutes W_1', A' for W_1, A respectively so that $W_0' + W_1' \ge 0$ and $A'(W_0' + W_1')$ is a good approximation to M.

Let us assume we know the vectors v_i appearing in the SVD of A and $\|A\|_F$. This is easy to guarantee since we can enumerate over all choices of the v_i's (which are unit vectors in $\mathbb{R}^r$) using a suitable ϵ_2-net where $\epsilon_2 = \min\{\frac{\epsilon}{\delta r}, 0.1\}$. Also, $\|A\|_F$ is a scalar value that can be easily guessed within multiplicative factor arbitrarily close to one.

Let $W_1' = Z$ be the optimal solution to the following convex program:

$$\min \quad \|A\|_F^2 \sum_{t=1}^{t_0} \|v_i^T Z\|^2 + \delta^2 \|M\|_F^2 \sum_{t=t_0+1}^{r} \|v_i^T Z\|^2 \tag{2.4}$$

$$s.t. \quad W_0'' + Z \ge 0. \tag{2.5}$$

This is optimization problem is convex since the constraints are linear and the objective function is quadratic but convex. (In fact this optimization problem can be separated into m smaller convex programs because the constraints between different columns of W_1' are independent).

When the vectors we enumerated (denoted as $\{v_i'\}$) are close enough to the true values $\{v_i\}$, that is, when $\sum_{i=1}^r \|v_i' - v_i\|^2 \le \min\{\frac{\epsilon^2}{\delta^2 r}, 0.01\}$, the value of the objective function after substituting v by v' can only change by at most $O(\frac{\epsilon^2}{\delta^2}\|A\|_F^2 + r\delta^2\|M\|_F^2)$. From now on we work with the true values of $\{v_i\}$. The claim below and arguments after will still be true although the vectors are not exact.

Claim. *The optimal value of this convex program is at most $O(\frac{\epsilon^2}{\delta^2}\|A\|_F^2 + r\delta^2\|M\|_F^2)$.*

Proof. We prove that $W_1' = W - W_0'' = (W_0 - W_0'') + W_1$ is a feasible solution and that the objective value of this solution is the value claimed in the lemma.

Since $W_1 = \sum_{t=t_0+1}^{r}(\boldsymbol{v_t v_t^T})W$ only contributes to the second term of the objective function in (2.4), we can upper bound the objective as

$$\|W_0 - W_0''\|_F^2\|A\|_F^2 + (\|W_0 - W_0''\|_F + \|W_1\|_F)^2\delta^2\|M\|_F^2.$$

The proof is completed because $\|W_0 - W_0''\|_F = O(\frac{\epsilon}{\delta})$ and $\|W_1\|_F \le \|W\|_F = \sqrt{r}$. □

After solving the convex program, we obtaine a candidate W_1'. Let $W' = W_0'' + W_1'$. To get the right A' (since W' is fixed) we can find the A' that minimizes $\|M - A'W'\|_F^2$ by solving a least-squares problem. Clearly such an A' satisfies $\|M - A'(W_0'' + W_1')\|_F \le \|M - A(W_0'' + W_1')\|_F$ and the latter quantity is bounded by $\|M - AW_0\|_F + \|A(W_0 - W_0'')\|_F + \|AW_1'\|_F$.

Lemma 2.51 bounds the first term and Lemma 2.52 bounds the second term. The square of the last term is bounded by the objective function of the convex program. □

Finally, by choosing $\delta = \frac{\sqrt{\epsilon}}{r^{1/4}}$ we get A', $W' = W_0'' + W_1'$ such that $\|M - A'W'\|_F \le O(\epsilon^{1/2}r^{1/4})\|M\|_F$.

Chapter 3

From NMF to Topic Modeling

Developing tools for automatic comprehension and classification of data —web pages, newspaper articles, images, genetic sequences, user ratings — is a holy grail of machine learning. *Topic Modeling* is an approach that has proved successful in all these settings. In this chapter we focus on the application on text corpus.

In Chapter 2, we have already shown how to learn the topic matrix A when the number of words per document is much larger than the vocabulary size. However, this is unrealistic in practice, where most articles have no more than 1000 words and the vocabulary contains more than a few thousand common words. In this chapter we shall see how to reduce the sampling noise, and adapt the algorithm in the previous chapter to design a provable algorithm for topic modeling.

3.1 Main Results

We first precisely define the topic modeling (learning) problem. There is an unknown *topic matrix* A which is dimension $n \times r$ (i.e. n is the dictionary size) and each column of A is a distribution on $[n]$. There is an unknown $r \times m$ matrix W whose each column is itself a distribution (aka convex combination) on $[r]$. The columns of W are i.i.d. samples from a distribution $\mathcal{T}$ which belongs to a known family, e.g., Dirichlet distributions, but whose parameters are unknown. Thus each column of AW (being a convex combination of distributions) is itself a distribution on $[n]$, and the algorithm's input consists of N i.i.d. samples for each column of AW. Here N is the document size and is assumed to be a constant for simplicity. Our algorithm can be easily adapted to work when the documents have different sizes.

The algorithm's running time will necessarily depend upon various model parameters, since distinguishing a very small parameter from 0 imposes a lower bound on the number of samples needed. The first such parameter is a quantitative version of *separability*, which was presented in Chapter 2 as a natural assumption in context of NMF and topic modeling.

Definition 3.1 (p-Separable Topic Matrix)**.** *An $n \times r$ matrix A is* p-separable *if for each i there is some row $\pi(i)$ of A that has a single nonzero entry which is in the i^{th} column and it is at least p.*

This assumption requires that each topic has some near-perfect indicator word – a word that we call the *anchor word* for this topic— that appears with reasonable probability in that topic but with negligible probability in all other topics (e.g., "soccer" could be an anchor word for the topic "sports"). This property is particularly natural in the context of topic modeling, where the number of distinct words (dictionary size) is very large compared to the number of topics. Note that separability does *not* mean that the anchor word always occurs (in fact, a typical document may be very likely to contain *no* anchor words). Instead, it dictates that when an anchor word does occur, it is a strong indicator that the corresponding topic is in the mixture used to generate the document.

The next parameter measures the lowest probability with which a topic occurs in the distribution that generates columns of W.

Definition 3.2 (Topic Imbalance). The *topic imbalance* of the model is the ratio between the largest and smallest expected entries in a column of W, in other words, $a = \max_{i,j \in [r]} \frac{\mathbb{E}[X_i]}{\mathbb{E}[X_j]}$ where $X \in \mathbb{R}^r$ is a random weighting of topics chosen from the distribution.

Finally, we require that topics stay identifiable despite sampling-induced noise. To formalize this, we define a matrix that will be important throughout this paper:

Definition 3.3 (Topic-Topic Covariance Matrix $R(\mathcal{T})$). If $\mathcal{T}$ is the distribution that generates the columns of W, then $R(\mathcal{T})$ is defined as an $r \times r$ matrix whose (i, j)th entry is $E[X_i X_j]$ where $X_1, X_2, ...X_r$ is a vector chosen from $\mathcal{T}$.

Let $\gamma > 0$ be a lower bound on the ℓ_1-condition number of the matrix $R(\mathcal{T})$. This is defined in Section 3.2, but for a $r \times r$ matrix it is within a factor of $\sqrt{r}$ of the smallest singular value. Our algorithm will work for any γ, but the number of documents we require will depend (polynomially) on $1/\gamma$:

Theorem 3.4 (Main). *There is a polynomial time algorithm that learns the parameters of a topic model if the number of documents is at least*

$$m = \max \left\{ O\left(\frac{\log n \cdot a^4 r^6}{\epsilon^2 p^6 \gamma^2 N} \right), O\left(\frac{\log r \cdot a^2 r^4}{\gamma^2} \right) \right\},$$

where the three numbers a, p, γ are as defined above. The algorithm learns the topic-term matrix A up to entry-wise additive error ϵ. Moreover, when the number of documents is also larger than $O\left(\frac{\log r \cdot r^2}{\epsilon^2}\right)$ the algorithm can learn the topic-topic covariance matrix $R(\mathcal{T})$ up to entry-wise additive error ϵ.

As noted earlier, we are able to recover the topic matrix even though we do not always recover the parameters of the column distribution $\mathcal{T}$. In some special cases we can also recover the parameters of $\mathcal{T}$, e.g. when this distribution is Dirichlet, as happens in the popular *Latent Dirichlet Allocation* (LDA) model [24, 26]. In Section 3.4.1 we compute a lower bound on the γ parameter for the Dirichlet distribution, which allows us to apply our main learning algorithm, and also the parameters of $\mathcal{T}$ can be recovered from the co-variance matrix $R(\mathcal{T})$ (see Section 3.4.2).

Recently the basic LDA model has been refined to allow correlation among different topics, which is more realistic. See for example the *Correlated Topic Model* (CTM) [25] and the *Pachinko Allocation Model* (PAM) [112]. A compelling aspect of our algorithm is that it extends to these models as well: we can learn the topic matrix, even though we cannot always identify $\mathcal{T}$. (Indeed, the distribution $\mathcal{T}$ in the Pachinko is not even identifiable: two different sets of parameters can generate exactly the same distribution.)

This Chapter is organized as follows: in Section 3.2 we introduce various parameters related to the A matrix. This prepares us for the proof of the main theorem in Section 3.3. The proof is adapted to the special case of LDA in Section 3.4. We give a better algorithm for NMF in Section 3.5 which is important for improving the sample complexity of the main theorem. Finally, Section 3.6 shows maximum likelihood estimator is NP-hard for topic modeling even if topics are separable, and justifies the need to assume data is actually generated according to the model.

3.2 Tools for (Noisy) Nonnegative Matrix Factorization

3.2.1 Various Condition Numbers

Central to our arguments will be various notions of matrices being "far" from being low-rank. The most interesting one for our purposes was introduced by Kleinberg and Sandler [96] in the context of collaborative filtering; and can be thought of as an ℓ_1-analogue to the smallest singular value of a matrix.

Definition 3.5 (ℓ_1 Condition Number)**.** If matrix B has nonnegative entries and all rows sum to 1 then its ℓ_1 Condition Number $\Gamma(B)$ is defined as:

$$\Gamma(B) = \min_{|\boldsymbol{x}|_1=1} |\boldsymbol{x}^\top B|_1.$$

If B does not have row sums of one then $\Gamma(B)$ is equal to $\Gamma(DB)$ where D is the diagonal matrix such that DB has row sums of one.

For example, if the rows of B have disjoint support then $\Gamma(B) = 1$ and in general the quantity $\Gamma(B)$ can be thought of a measure of how close two distributions on *disjoint* sets of rows can be. Note that, if $\boldsymbol{x}$ is an n-dimensional real vector, $\|\boldsymbol{x}\|_2 \leq |\boldsymbol{x}|_1 \leq \sqrt{n}\|\boldsymbol{x}\|_2$ and hence (if $\sigma_{min}(B)$ is the smallest singular value of B), we have:

$$\frac{1}{\sqrt{n}}\sigma_{min}(B) \leq \Gamma(B) \leq \sqrt{m}\sigma_{min}(B).$$

The above notions of condition number will be most relevant in the context of the topic-topic covariance matrix $R(\mathcal{T})$. We shall always use γ to denote the ℓ_1 condition number of $R(\mathcal{T})$. The definition of condition number will be preserved even when we estimate the topic-topic covariance matrix using random samples.

Lemma 3.6. *When $m > 5\log r/\epsilon_0^2$, with high probability the matrix $R = \frac{1}{m}WW^\top$ is entry-wise close to $R(\mathcal{T})$ with error ϵ_0. Further, when $\epsilon_0 < 1/4\gamma ar^2$ where a is topic imbalance, the matrix R has ℓ_1 condition number at least $\gamma/2$.*

Proof. Since $\mathbb{E}[W_iW_i^\top] = R(\mathcal{T})$, the first part is just by Chernoff bound and union bound. The further part follows because $R(\mathcal{T})$ is γ robustly simplicial, and the error can change the ℓ_1 norm of vR for any unit v by at most $ar \cdot r\epsilon_0$. The extra factor ar comes from the normalization to make rows of R sum up to 1. □

In Chapter 2 we defined a different measure of "distance" from singular which is essential to the polynomial time algorithm for NMF:

Definition 3.7 (β-robustly simplicial)**.** If each column of a matrix A has unit ℓ_1 norm, then we say it is *β-robustly simplicial* if no column in A has ℓ_1 distance smaller than β to the convex hull of the remaining columns in A.

The following claim clarifies the interrelationships of these latter condition numbers.

Claim. *(i) If A is p-separable then $A^\top$ has ℓ_1 condition number at least p. (ii) If $A^\top$ has all row sums equal to* 1 *then A is β-robustly simplicial for $\beta = \Gamma(A^\top)/2$.*

We shall see that the ℓ_1 condition number for product of matrices is at least the product of ℓ_1 condition number. The main application of this composition is to show that the matrix $R(\mathcal{T})A^\top$ (or the empirical version $RA^\top$) is at least $\Omega(\gamma p)$-robustly simplicial. The following lemma will play a crucial role in analyzing our main algorithm:

Lemma 3.8 (Composition Lemma)**.** *If B and C are matrices with ℓ_1 condition number $\Gamma(B) \geq \gamma$ and $\Gamma(C) \geq \beta$, then $\Gamma(BC)$ is at least $\beta\gamma$. Specificially, when A is p-separable the matrix $R(\mathcal{T})A^\top$ is at least $\gamma p/2$-robustly simplicial.*

Proof. The proof is straight forward because for any vector $\boldsymbol{x}$, we know $|\boldsymbol{x}^\top BC|_1 \leq \Gamma(C)|\boldsymbol{x}^\top B|_1 \leq \Gamma(C)\Gamma(B)|\boldsymbol{x}|_1$. For the matrix $R(\mathcal{T})A^\top$, by Claim 3.2.1 we know the matrix $A^\top$ has ℓ_1 condition number at least p. Hence $\Gamma(R(\mathcal{T})A^\top)$ is at least γp and again by Claim 3.2.1 the matrix is $\gamma p/2$-robustly simplicial. □

3.2.2 Noisy Nonnegative Matrix Factorization under Separability

A key ingredient is an approximate NMF algorithm in Chapter 2, which can recover an approximate nonnegative matrix factorization $\tilde{M} \approx AW$ when the ℓ_1 distance between each row of $\tilde{M}$ and the corresponding row in AW is small. We emphasize that this is not enough for our purposes, since the term-by-document matrix $\tilde{M}$ will have a substantial amount of noise (when compared to its expectation) precisely because the number of words in a document N is much smaller than the dictionary size n. Rather, we will apply the following algorithm (and an improvement that we give in Section 3.5) to the Gram matrix $\tilde{M}\tilde{M}^\top$.

For topic modeling we need a slightly different goal here than in Chapter 2. Instead of recovering estimates to the anchor words that are close in ℓ_1-norm, we would rather recover almost anchor words that have almost all its weights on a single coordinate. (this was called the *simplicial norm* in Section 2.6.) Hence, we will be able to achieve better bounds by treating this problem directly, we defer the proof to Section 3.5.

Theorem 3.9. *Suppose $M = AW$ where W and M are normalized to have rows sum up to 1, A is separable and W is γ-robustly simplicial. When $\epsilon < \gamma/100$ there is a polynomial time algorithm that given $\tilde{M}$ such that for all rows $|\tilde{M}^i - M^i|_1 < \epsilon$, finds r row (almost anchor words) in $\tilde{M}$. The i-th almost anchor word corresponds to a row in M that can be represented as $(1 - O(\epsilon/\gamma))W^i + O(\epsilon/\gamma)W^{-i}$. Here W^{-i} is a vector in the convex hull of other rows in W with unit length in ℓ_1 norm.*

3.3 Algorithm for Learning a Topic Model: Proof of Theorem 3.4

First it is important to understand why separability helps in nonnegative matrix factorization, and specifically, the exact role played by the anchor words. Suppose the NMF algorithm is given a matrix AB. If A is p-separable then this means that A contains a diagonal matrix (up to row permutations). Thus a scaled copy of each row of B is present as a row in AB. In fact, if we knew the anchor words of A, then by looking at the corresponding rows of AB we could“read off” the corresponding row of B (up to scaling), and use these in turn to recover all of A. Thus the anchor words constitute the “key” that “unlocks” the factorization, and indeed the main step of our earlier NMF algorithm was a geometric procedure to identify the anchor words. When one is given a noisy version of AB, the analogous notion is "almost anchor" words, which correspond to rows of AB that are "very close" to rows of B; see Theorem 3.9.

Now we sketch how to apply these insights to learning topic models. Let M denote the provided term-by-document matrix, whose each column describes the empirical word frequencies in the documents. It is obtained from sampling AW and thus is an extremely noisy approximation to AW. Our algorithm starts by forming the Gram matrix $MM^\top$, which can be thought of as an empirical word-word covariance matrix. In fact as the number of documents increases $\frac{1}{m}MM^\top$ tends to a limit $Q = \frac{1}{m}E[AWW^\top A]$, implying $Q = AR(\mathcal{T})A^\top$. (See Lemma 3.15.) Imagine that we are given the exact matrix Q instead of a noisy approximation. Notice that Q is a product of *three* nonnegative matrices, the first of which is p-separable and the last is the transpose of the first. NMF at first sight seems too weak to help find such factorizations. However, if we think of Q as a product of *two* nonnegative matrices, A and $R(\mathcal{T})A^\top$, then our NMF algorithm [11] can at least identify the anchor words of A. As noted above, these suffice to recover $R(\mathcal{T})A^\top$, and then (using the anchor words of A again) all of A as well. See Section 3.3.1 for details.

Of course, we are not given Q but merely a good approximation to it. Now our NMF algorithm allows us to recover "almost anchor" words of A, and the crux of the proof is Section 3.3.2 showing that these suffice to recover provably good estimates to A and $WW^\top$. This uses (mostly) bounds from matrix perturbation theory, and interrelationships of condition numbers mentioned in Section 3.2.

For simplicity we assume the following condition on the topic model, which we will see in Section 3.3.5 can be assumed without loss of generality:

(*) *The number of words, n, is at most $4ar/\epsilon$.*

Please see Algorithm 3.1: Main Algorithm for description of the algorithm. Note that R is our shorthand for $\frac{1}{m}WW^\top$, which as noted converges to $R(\mathcal{T})$ as the number of documents increases.

Algorithm 3.1. Learning Topic Models

```
input Samples from a topic model
output $R$ and $A$
Query the oracle for $m$ documents, where
    $m = \max\left\{O\left(\frac{\log n \cdot a^4 r^6}{\epsilon^2 p^6 \gamma^2 N}\right), O\left(\frac{\log r \cdot a^2 r^4}{\gamma^2}\right), O\left(\frac{\log r \cdot r^2}{\epsilon^2}\right)\right\}$
Split the words of each document into two halves, and let $\tilde{M}$, $\tilde{M}'$ be the term-by-document matrix with first and second half of words respectively.
Compute word-by-word matrix $Q = \frac{4}{N^2 m}\tilde{M}\tilde{M}'^\top$
Apply the "Robust NMF" algorithm of Theorem 3.9 to $Q$ which returns $r$ words that are "almost" the anchor words of $A$.
Use these $r$ words as input to Recover with Almost Anchor Words to compute $R = \frac{1}{m}WW^\top$ and $A$
```

3.3.1 Recover R and A with Anchor Words

We first describe how the recovery procedure works in an "idealized" setting (Algorthm 3.2,Recover with True Anchor Words), when we are given the exact value of $ARA^\top$ and a set of anchor words – one for each topic. We can permute the rows of A so that the anchor words are exactly the first r words. Therefore $A^\top = (D, U^\top)$ where D is a diagonal matrix. Note that D is not necessarily the identity matrix (nor even a scaled copy of the identity matrix), but we do know that the diagonal entries are at least p. We apply the same permutation to the rows and columns of Q. As shown in Figure 3.1, if we look at the submatrix formed by the first r rows and r columns, it is exactly DRD. Similarly, the submatrix consisting of the first r rows is exactly $DRA^\top$. We can use these two matrices to compute R and A, in this idealized setting (and we will use the same basic strategy in the general case, but need only be more careful about how we analyze how errors compound in our algorithm).

Algorithm 3.2. Recover with True Anchor Words

```
input  $r$ anchor words, empirical correlation matrix $Q$
output  $R$ and $A$
1: Permute the rows and columns of $Q$ so that the anchor words appear in the first $r$ rows and columns
2: Compute $DRA^\top \mathbf{1}$ (which is equal to $DR\mathbf{1}$)
3: Solve for $\boldsymbol{z}$: $DRD\boldsymbol{z} = DR\mathbf{1}$.
4: Output $A^\top = ((DRD\text{Diag}(z))^{-1}DRA^\top)$.
5: Output $R = (\text{Diag}(z)DRD\text{Diag}(z))$.
```

Our algorithm has exact knowledge of the matrices DRD and $DRA^\top$, and so the main task is to recover the diagonal matrix D. Given D, we can then compute A and R (for the Dirichlet Allocation we can also compute its parameters - i.e. the $\boldsymbol{\alpha}$ so that $R(\alpha) = R$). The key idea to this algorithm is that the row sums of DR and $DRA^\top$ are the same, and we can use the row sums of DR to set up a system of linear constraints on the diagonal entries of D^{-1}.

Lemma 3.10. *When the matrix Q is exactly equal to $ARA^\top$ and we know the set of anchor words,* Recover with True Anchor Words *outputs A and R correctly.*

Proof. The Lemma is straight forward from Figure 3.1 and the procedure. By Figure 3.1 we can find the exact value of $DRA^\top$ and DRD in the matrix Q. Step 2 of recover computes $DR\mathbf{1}$ by computing $DRA^\top\mathbf{1}$. The two vectors are equal because A is the topic-term matrix and its columns sum up to 1, in particular $A^\top\mathbf{1} = \mathbf{1}$.

In Step 3, since R is invertible by Lemma 3.6, D is a diagonal matrix with entries at least p, the matrix DRD is also invertible. Therefore there is a unique solution $\boldsymbol{z} = (DRD)^{-1}DR\mathbf{1} = D^{-1}\mathbf{1}$. Also $D\boldsymbol{z} = \mathbf{1}$ and hence $D\text{Diag}(z) = I$. Finally, using the fact that $D\text{Diag}(z) = I$, the output in step 4 is just $(DR)^{-1}DRA^\top = A^\top$, and the output in step 5 is equal to R. □

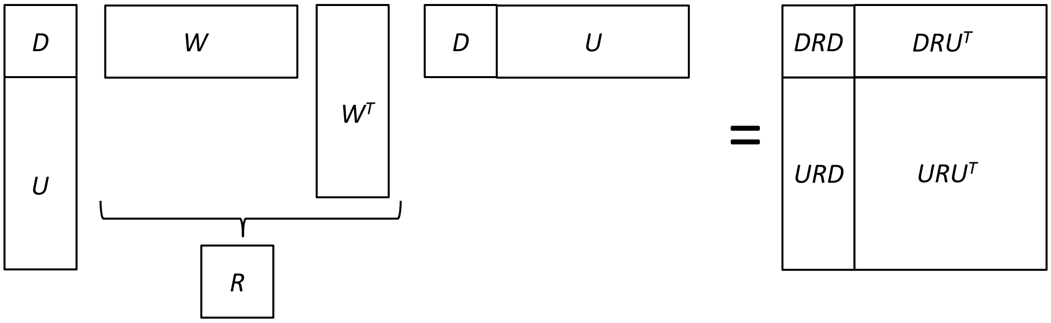


Figure 3.1: The matrix Q

3.3.2 Recover R and A with Almost Anchor Words

What if we are not given the exact anchor words, but are given words that are "close" to anchor words? As we noted, in general we cannot hope to recover the true anchor words, but even a good approximation will be enough to recover R and A.

When we restrict A to the rows corresponding to "almost" anchor words, the submatrix will not be diagonal. However, it will be close to a diagonal in the sense that the submatrix will be a diagonal matrix D multiplied by E, and E is close to the identity matrix (and the diagonal entries of D are at least $\Omega(p)$). Here we analyze the same procedure as above and show that it still recovers A and R (approximately) even when given "almost" anchor words instead of true anchor words. For clarity we state the procedure again in Algorithm 3.3: Recover with Almost Anchor Words. The guarantees at each step are different than before, but the implementation of the procedure is the same. Notice that here we permute the rows of A (and hence the rows and columns of Q) so that the "almost" anchor words returned by Theorem 3.9 appear first and the submatrix A on these rows is equal to DE.

Here, we still assume that the matrix Q is exactly equal to $ARA^\top$ and hence the first r rows of Q form the submatrix $DERA^\top$ and the first r rows and columns are $DERE^\top D$. The complication here is that $\mathrm{Diag}(z)$ is not necessarily equal to D^{-1}, since the matrix E is not necessarily the identity. However, we can show that $\mathrm{Diag}(z)$ is "close" to D^{-1} if E is suitably close to the identity matrix – i.e. given good enough proxies for the anchor words, we can bound the error of the above recovery procedure. We write $E = I + Z$. Intuitively when Z has only small entries E should behave like the identity matrix. In particular, E^{-1} should have only small off-diagonal entries. We make this precise through the following lemmas:

Lemma 3.11. *Let $E = I + Z$ and $\sum_{i,j} |Z_{i,j}| = \epsilon < 1/2$, then $E^{-1}\mathbf{1}$ is a vector with entries in the range $[1 - 2\epsilon, 1 + 2\epsilon]$.*

Proof. E is clearly invertible because the spectral norm of Z is at most $1/2$. Let $\boldsymbol{b} = E^{-1}\mathbf{1}$. Since $E = I + Z$ we multiply E on both sides to get $\boldsymbol{b} + Z\boldsymbol{b} = \mathbf{1}$. Let b_{max} be the largest absolute value of any entry of b ($b_{max} = \max |b_i|$). Consider the entry i where b_{max} is achieved, we know $b_{max} = |b_i| \leq 1 + |(Zb)_i| \leq 1 + \sum_j |Z_{i,j}||b_j| \leq 1 + \epsilon b_{max}$. Thus $b_{max} \leq 1/(1 - \epsilon) \leq 2$. Now all the entries in $Z\boldsymbol{b}$ are within 2ϵ in absolute value, and we know that $\boldsymbol{b} = \mathbf{1} + Z\boldsymbol{b}$. Hence all the entries of b are in the range $[1 - 2\epsilon, 1 + 2\epsilon]$, as desired. □

Lemma 3.12. *Let $E = I + Z$ and $\sum_{i,j} |Z_{i,j}| = \epsilon < 1/2$, then the columns of $E^{-1} - I$ have ℓ_1 norm at most 2ϵ.*

Proof. Without loss of generality, we can consider just the first column of $E^{-1} - I$, which is equal to $(E^{-1} - I)\boldsymbol{e_1}$, where $\boldsymbol{e_1}$ is the indicator vector that is one on the first coordinate and zero elsewhere.

The approach is similar to that in Lemma 3.11. Let $\boldsymbol{b} = (E^{-1} - I)\boldsymbol{e_1}$. Left multiply by $E = (I + Z)$ and we obtain $\boldsymbol{b} + Z\boldsymbol{b} = -Z\boldsymbol{e_1}$. Hence $\boldsymbol{b} = -Z(\boldsymbol{b} + \boldsymbol{e_1})$. Let b_{max} be the largest absolute value of entries of $\boldsymbol{b}$ ($b_{max} = \max |b_i|$). Let i be the entry in which b_{max} is achieved. Then

$$b_{max} = |b_i| \leq |(Z\boldsymbol{b})_i| + |(Z\boldsymbol{e_1})_i| \leq \epsilon b_{max} + \epsilon$$

Therefore $b_{max} \leq \epsilon/(1 - \epsilon) \leq 2\epsilon$. Further, the $|\boldsymbol{b}|_1 \leq |Z\boldsymbol{e_1}|_1 + |Z\boldsymbol{b}|_1 \leq \epsilon + 2\epsilon^2 \leq 2\epsilon$. □

Now we are ready to show that the procedure Recover with Almost Anchor Words succeeds when given ”almost” anchor words:

Algorithm 3.3. Recover with Almost Anchors

```
input  r "almost" anchor words, empirical correlation $Q$
output  $R$ and $A$.
1: Permute the rows and columns of $Q$ so that the "almost" anchor words appear in the first $r$ rows and columns.
2: Compute $DERA^\top \mathbf{1}$ (which is equal to $DER\mathbf{1}$)
3: Solve for $\boldsymbol{z}$: $DERE^\top D\boldsymbol{z} = DER\mathbf{1}$.
4: Output $A^\top = ((DERE^\top D\mathrm{Diag}(z))^{-1}DERA^\top)$.
5: Output $R = (\mathrm{Diag}(z)DERE^\top D\mathrm{Diag}(z))$.
```

Lemma 3.13. *When the matrix Q is exactly equal to $ARA^\top$, the matrix A restricted to almost anchor words is DE where $E - I$ has ℓ_1 norm $\epsilon < 1/10$ when viewed as a vector, procedure* Recover with Almost Anchor Words *outputs A such that each column of A has ℓ_1 error at most 6ϵ. The matrix R has additive error Z_R whose ℓ_1 norm when viewed as a vector is at most 8ϵ.*

Proof. Since Q is exactly $ARA^\top$, our algorithm is given $DERA^\top$ and $DERE^\top D$ with no error. In Step 3, since D, E and R are all invertible, we have

$$\boldsymbol{z} = (DERE^\top D)^{-1}DER\mathbf{1} = D^{-1}(E^\top)^{-1}\mathbf{1}$$

Ideally we would want $\mathrm{Diag}(z) = D^{-1}$, and indeed $D\mathrm{Diag}(z) = \mathrm{Diag}((E^\top)^{-1}\mathbf{1})$. From Lemma 3.11, the vector $(E^\top)^{-1}\mathbf{1}$ has entries in the range $[1-2\epsilon, 1+2\epsilon]$, thus each entry of $\mathrm{Diag}(z)$ is within a $(1 \pm 2\epsilon)$ multiplicative factor from the corresponding entry in D^{-1}.

Consider the output in Step 4. Since D, E, R are invertible, the first output is

$$(DERE^\top D\mathrm{Diag(z)})^{-1}DERA^\top = (D\mathrm{Diag}(z))^{-1}(E^\top)^{-1}A^\top$$

Our goal is to bound the ℓ_1 error of the columns of the output compared to the corresponding columns of A. Notice that it is sufficient to show that the j^{th} row of $(D\text{Diag}(z))^{-1}(E^\top)^{-1}$ is close (in ℓ_1 distance) to the indicator vector $\boldsymbol{e_j}^\top$.

Claim. *For each j, $|\boldsymbol{e_j}^\top(DDiag(z))^{-1}(E^\top)^{-1} - \boldsymbol{e_j}^\top|_1 \leq 5\epsilon$*

Proof. Again, without loss of generality we can consider just the first row. From Lemma 3.12 $\boldsymbol{e_1}^\top(E^\top)^{-1}$ has ℓ_1 distance at most 2ϵ to $\boldsymbol{e_1}^\top$. $(D\text{Diag}(z))^{-1}$ has entries in the range $[1-3\epsilon, 1+3\epsilon]$. And so

$$|\boldsymbol{e_1}^\top(D\text{Diag}(z))^{-1}(E^\top)^{-1} - \boldsymbol{e_1}^\top|_1 \leq |\boldsymbol{e_1}^\top(D\text{Diag}(z))^{-1}(E^\top)^{-1} - \boldsymbol{e_1}^\top(E^\top)^{-1}|_1 + |\boldsymbol{e_1}^\top(E^\top)^{-1} - \boldsymbol{e_1}^\top|_1$$

The last term can be bounded by 2ϵ. Consider the first term on the right hand side: The vector $\boldsymbol{e_1}^\top(D\text{Diag}(z))^{-1} - \boldsymbol{e_1}^\top$ has one non-zero entry (the first one) whose absolute value is at most 3ϵ. Hence, from Lemma 3.12 the first term can be bounded by $6\epsilon^2 \leq 3\epsilon$, and this implies the claim. □

The first row of $(D\text{Diag}(\boldsymbol{z}))^{-1}(E^\top)^{-1}A^\top$ is $A_1 + \boldsymbol{z}^\top A$ where $\boldsymbol{z}$ is a vector with ℓ_1 norm at most 5ϵ. So every column of A is recovered with ℓ_1 error at most 6ϵ.

Consider the second output of the algorithm. The output is $\text{Diag}(z)DERE^\top D\text{Diag}(z)$ and we can write $\text{Diag}(z)D = I + Z_1$ and $E = I + Z_2$. The leading error are $Z_1R + Z_2R + RZ_1 + RZ_2$ and hence the ℓ_1 norm of the leading error term (when treated as a vector) is at most 6ϵ and other terms are of order ϵ^2 and can safely be bounded by 2ϵ for suitably small ϵ). □

Finally we consider the general case (in which there is additive noise in Step 1): we are not given $ARA^\top$ exactly. We are given Q which is close to $ARA^\top$ (by Lemma 3.15). We will bound the accumulation of this last type of error. Suppose in Step 1 of RECOVER we obtain $DERA^\top + U$ and $DERE^\top D + V$ and furthermore the entries of U and $U\mathbf{1}$ have absolute value at most ϵ_1 and the matrix V has ℓ_1 norm ϵ_2 when viewed as a vector.

Lemma 3.14. *If $\epsilon, \epsilon_1, \epsilon_2$ are sufficiently small, RECOVER outputs A such that each entry of A has additive error at most $O(\epsilon + (ra\epsilon_2/p^3 + \epsilon_1 r/p^2)/\gamma)$. Also the matrix R has additive error Z_R whose ℓ_1 norm when viewed as a vector is at most $O(\epsilon + (ra\epsilon_2/p^3 + \epsilon_1 r/p^2)/\gamma)$.*

The main idea of the proof is to write $DERE^\top D + V$ as $DER(E^\top + V')D$. In this way the error V can be translated to an error V' on E and Lemma 3.13 can be applied. The error U can be handled similarly.

Proof. We shall follow the proof of Lemma 3.13. First can express the error term V instead as $V = (DER)V'(D)$. This is always possible because all of D, E, R are invertible. Moreover, the ℓ_1 norm of V' when viewed as a vector is at most $8ra\epsilon_2/\gamma p^3$, because this norm will grow by a factor of at most $1/p$ when multiplied by D^{-1}, a factor of at most 2 when multiplied by E^{-1} and at most $ra/\Gamma(R)$ when multiplied by R^{-1}. The bound of $\Gamma(R)$ comes from Lemma 3.6, we lose an extra ra because R may not have rows sum up to 1.

Hence $DERE^\top D + V = DER(E^\top + V')D$ and the additive error for $DERE^\top D$ can be transformed into error in E, and we will be able to apply the analysis in Lemma 3.13.

Similarly, we can express the error term U as $U = DERU'$. Entries of U' have absolute value at most $8\epsilon_1 r/\gamma p^2$. The right hand side of the equation in step 3 is equal to $DER\mathbf{1} + U\mathbf{1}$ so the error is at most ϵ_1 per entry. Following the proof of Lemma 3.13, we know $\mathrm{Diag}(z)D$ has diagonal entries within $1 \pm (2\epsilon + 16\epsilon_2/\gamma p^3 + 2\epsilon_1)$.

Now we consider the output. The output for $A^\top$ is equal to

$$(DER(E^\top + V')D\mathrm{Diag}(z))^{-1}DER(A^\top + U') = (D\mathrm{Diag}(z))^{-1}(E^\top + V')^{-1}(A^\top + U').$$

Here we know $(E^\top + V')^{-1} - I$ has ℓ_1 norm at most $O(\epsilon + ra\epsilon_2/\gamma p^3)$ per row, $(D\mathrm{Diag}(z))$ is a diagonal matrix with entries in $1 \pm O(\epsilon + ra\epsilon_2/\gamma p^3 + \epsilon_1)$, entries of U' has absolute value $O(\epsilon_1 r/\gamma p^2)$. Following the proof of Lemma 3.13 the final entry-wise error of A is roughly the sum of these three errors, and is bounded by $O(\epsilon + (ra\epsilon_2/p^3 + \epsilon_1 r/p^2)/\gamma)$ (Notice that Lemma 3.13 gives bound for ℓ_1 norm of rows, which is stronger. Here we switched to entry-wise error because the entries of U are bounded while the ℓ_1 norm of U might be large).

Similarly, the output of R is equal to $\mathrm{Diag}(z)(DERE^\top D + V)\mathrm{Diag}(z)$. Again we write $\mathrm{Diag}(z)D = I + Z_1$ and $E = I + Z_2$. The extra term $\mathrm{Diag}(z)V\mathrm{Diag}(z)$ is small because the entries of z are at most to $2/p$ (otherwise $\mathrm{Diag}(z)D$ won't be close to identity). The error can be bounded by $O(\epsilon + (ra\epsilon_2/p^3 + \epsilon_1 r/p^2)/\gamma)$. □

Now in order to prove our main theorem we just need to show that when number of documents is large enough, the matrix Q is close to the $ARA^\top$, and plug the error bounds into Lemma 3.14.

3.3.3 Error Bounds for Q

Here we show that the matrix Q indeed converges to $\frac{1}{m}AWW^\top A^\top = ARA^\top$ when m is large enough.

Lemma 3.15. *When $m > \frac{50\log n}{N\epsilon_Q^2}$, with high probability all entries of $Q - \frac{1}{m}AWW^\top A^\top$ have absolute value at most ϵ_Q. Further, the ℓ_1 norm of rows of Q are also ϵ_Q close to the ℓ_1 norm of the corresponding row in $\frac{1}{m}AWW^\top A^\top$.*

Proof. We shall first show that the expectation of Q is equal to $ARA^\top$ where R is $\frac{1}{m}WW^\top$. Then by concentration bounds we show that entries of Q are close to their expectations. Notice that we can also hope to show that Q converges to $AR(\mathcal{T})A^\top$. However in that case we will not be able to get the inverse polynomial relationship with N (indeed, even if N goes to infinity it is impossible to learn $R(\mathcal{T})$ with only one document). Replacing $R(\mathcal{T})$ with the empirical R allows our algorithm to perform better when the number of words per document is larger.

To show the expectation is correct we observe that conditioned on W, the entries of two matrices $\tilde{M}$ and $\tilde{M}'$ are independent. Their expectations are both $\frac{N}{2}AW$. Therefore,

$$\mathbb{E}[Q] = \frac{4}{mN^2}\mathbb{E}[\tilde{M}\tilde{M}'^{\top}] = \frac{1}{m}\left(\frac{2}{N}\mathbb{E}[\tilde{M}]\right)\left(\frac{2}{N}\mathbb{E}[\tilde{M}'^{\top}]\right) = \frac{1}{m}AWW^{\top}A^{\top} = ARA^{\top}.$$

We still need to show that Q is close to this expectation. This is not surprising because Q is the average of m independent samples (of $\frac{4}{N^2}\tilde{M}_i\tilde{M}_i'$). Further, the variance of each entry in $\frac{4}{N^2}\tilde{M}_i\tilde{M}_i'^{\top}$ can be bounded because $\tilde{M}$ and $\tilde{M}'$ also come from independent samples. For any i, j_1, j_2, let $v = AW_i$ be the probability distribution that $\tilde{M}_i$ and $\tilde{M}_i'$ are sampled from, then $\tilde{M}_i(j_1)$ is distributed as $Binomial(N/2, v(j_1))$ and $\tilde{M}_i'(j_2)$ is distributed as $Binomial(N/2, v(j_2))$. The variance of these two variables are less than $N/8$ no matter what v is by the properties of binomial distribution. Conditioned on the vector v these two variables are independent, thus the variance of their product is at most $\operatorname{Var}\tilde{M}_i(j_1)\mathbb{E}\tilde{M}_i'(j_2)^2 + \mathbb{E}\tilde{M}_i(j_1)^2\operatorname{Var}\tilde{M}_i'(j_2) + \operatorname{Var}\tilde{M}_i(j_1)\operatorname{Var}\tilde{M}_i'(j_2) \leq N^3/4 + N^2/64$. The variance of any entry in $\frac{4}{N^2}\tilde{M}_i\tilde{M}_i'^{\top}$ is at most $4/N + 1/16N^2 = O(1/N)$. Higher moments can be bounded similarly and they satisfy the assumptions of Bernstein inequalities. Thus by Bernstein inequalities the probability that any entry is more than ϵ_Q away from its true value is much smaller than $1/n^2$.

The further part follows from the observation that the ℓ_1 norm of a row in Q is proportional to the number of appearances of the word. As long as the number of appearances concentrates the error in ℓ_1 norm must be small. The words are all independent (conditioned on W) so this is just a direct application of Chernoff bounds. □

3.3.4 Proving the Main Theorem

We are now ready to prove Theorem 3.4:

Proof. By Lemma 3.15 we know when we have at least $50\log n/N\epsilon_Q^2$ documents, Q is entry wise close to $ARA^{\top}$. In this case error per row for Theorem 3.9 is at most $\epsilon_Q \cdot O(a^2r^2/p^2)$

because in this step we can assume that there are at most $4ar/p$ words (see Section 3.3.5) and to normalize the row we need a multiplicative factor of at most $10ar/p$ (we shall only consider rows with ℓ_1 norm at least $p/10ar$, with high probability all the anchor words are in these rows). The γ parameter for Theorem 3.9 is $p\gamma/4$ by Lemma 3.6. Thus the almost anchor words found by the algorithm has weight at least $1-O(\epsilon_Q a^2r^2/\gamma p^3)$ on diagonals. The error for $DERE^\top D$ is at most $\epsilon_Q r^2$, the error for any entry of $DERA^\top$ and $DERA^\top\mathbf{1}$ is at most $O(\epsilon_Q)$. Therefore by Lemma 3.14 the entry-wise error for A is at most $O(\epsilon_Q a^2r^3/\gamma p^3)$.

When $\epsilon_Q < \epsilon p^3\gamma/a^2r^3$ the error is bounded by ϵ . In this case we need

$$m = \max\left\{O\left(\frac{\log n \cdot a^4r^6}{\epsilon^2p^6\gamma^2N}\right), O\left(\frac{\log r \cdot a^2r^4}{\gamma^2}\right)\right\}.$$

The latter constraint comes from Lemma 3.6.

To get within ϵ additive error for the parameter α, we further need R to be close enough to the variance-covariance matrix of the document-topic distribution, which means m is at least

$$m = \max\left\{O\left(\frac{\log n \cdot a^4r^6}{\epsilon^2p^6\gamma^2N}\right), O\left(\frac{\log r \cdot a^2r^4}{\gamma^2}\right), O\left(\frac{\log r \cdot r^2}{\epsilon^2}\right)\right\}.$$

□

3.3.5 Reducing Dictionary Size

Above we assumed that the number of distinct words is small. Here, we give a simple gadget that shows in the general case we can assume that this is the case at the loss of an additional additive ϵ in our accuracy:

Lemma 3.16. *The general case can be reduced to an instance in which there are at most $4ar/\epsilon$ words all of which (with at most one exception) occur with probability at least $\epsilon/4ar$.*

Proof. In fact, we can collect all words that occur infrequently and "merge" all of these words into a aggregate word that we will call the *runoff word*. To this end, we call a word large if it appears more than $\epsilon mN/3ar$ times in $m = \frac{100ar \log n}{N\epsilon}$ documents, and otherwise we call it small. Indeed, with high probability all large words are words that occur with probability at least $\epsilon/4ar$ in our model. Also, all words that has a entry larger than ϵ in the corresponding row of A will appear with at least ϵ/ar probability, and is thus a large word with high probability. We can merge all small words (i.e. rename all of these words to a single, new word). Hence we can apply the above algorithm (which assumed that there are not too many distinct words). After we get a result with the modified documents we can ignore the *runoff words* and assign 0 weight for all the small words. The result will still be correct up to ϵ additive error. □

3.4 The Dirichlet Subcase

Here we demonstrate that the parameters of a Dirichlet distribution can be (robustly) recovered from just the covariance matrix $R(\mathcal{T})$. Hence an immediate corollary is that our main learning algorithm can recover both the topic matrix A and the distribution that generates columns of W in a *Latent Dirichlet Allocation* (LDA) Model [26], provided that A is separable. We believe that this algorithm may be of practical use, and provides the first alternative to local search and (unproven) approximation procedures for this inference problem [157], [51], [26].

The Dirichlet distribution is parametrized by a vector α of positive reals is a natural family of continuous multivariate probability distributions. The support of the Dirichlet Distribution is the unit simplex whose dimension is the same as the dimension of α. Let α be a r dimensional vector. Then for a vector $\theta \in \mathbb{R}^r$ in the r dimensional simplex, its probability density is given by

$$Pr[\theta|\alpha] = \frac{\Gamma(\sum_{i=1}^r \alpha_i)}{\prod_{i=1}^r \Gamma(\alpha_i)} \prod_{i=1}^r \theta_i^{\alpha_i - 1},$$

where Γ is the Gamma function. In particular, when all the α_i's are equal to one, the Dirichlet Distribution is just the uniform random distribution over the probability simplex.

The expectation and variance of θ_i's are easy to compute given the parameters α. We denote $\alpha_0 = |\alpha|_1 = \sum_{i=1}^r \alpha_i$, then the ratio α_i/α_0 should be interpreted as the "size" of the i-th variable θ_i, and α_0 shows whether the distributions is concentrated in the interior (when α_0 is large) or near the boundary (when α_0 is small). The first two moments of Dirichlet Distribution is listed as below:

$$\mathbb{E}[\theta_i] = \frac{\alpha_i}{\alpha_0}.$$

$$\mathbb{E}[\theta_i\theta_j] = \begin{cases} \frac{\alpha_i\alpha_j}{\alpha_0(\alpha_0+1)} & \text{when } i \neq j \\ \frac{\alpha_i(\alpha_i+1)}{\alpha_0(\alpha_0+1)} & \text{when } i = j \end{cases}.$$

Suppose the Dirichlet distribution has $\max \alpha_i / \min \alpha_i = a$ and the sum of parameters is α_0; we give an algorithm that computes close estimates to the vector of parameters α given a sufficiently close estimate to the co-variance matrix $R(\mathcal{T})$ (Theorem 3.19). Combining this with Theorem 3.4, we obtain the following corollary:

Theorem 3.17. *There is an algorithm that learns the topic matrix A with high probability up to an additive error of ϵ from at most*

$$m = \max\left\{O\left(\frac{\log n \cdot a^6 r^8 (\alpha_0+1)^4}{\epsilon^2 p^6 N}\right), O\left(\frac{\log r \cdot a^2 r^4 (\alpha_0+1)^2}{\epsilon^2}\right)\right\}$$

documents sampled from the LDA model and runs in time polynomial in n, m. Furthermore, we also recover the parameters of the Dirichlet distribution to within an additive ϵ.

Our main goal in this section is to bound the ℓ_1-condition number of the Dirichlet distribution (Section 3.4.1), and using this we show how to recover the parameters of the distribution from its covariance matrix (Section 3.4.2).

3.4.1 Condition Number of a Dirichlet Distribution

There is a well-known meta-principle that if a matrix W is chosen by picking its columns independently from a fairly diffuse distribution, then it will be far from low rank. However, our analysis will require us to prove an explicit lower bound on $\Gamma(R(\mathcal{T}))$. We now prove such a bound when the columns of W are chosen from a Dirichlet distribution with parameter vector α. We note that it is easy to establish such bounds for other types of distributions as well. Recall that we defined $R(\mathcal{T})$ in Section 3.1, and here we will abuse notation and throughout this section we will denote by $R(\alpha)$ the matrix $R(\mathcal{T})$ where $\mathcal{T}$ is a Dirichlet distribution with parameter α.

Let $\alpha_0 = \sum_{i=1}^{r} \alpha_i$. The mean, variance and co-variance for a Dirichlet distribution are well-known, from which we observe that $R(\alpha)_{i,j}$ is equal to $\frac{\alpha_i \alpha_j}{\alpha_0(\alpha_0+1)}$ when $i \neq j$ and is equal to $\frac{\alpha_i(\alpha_i+1)}{\alpha_0(\alpha_0+1)}$ when $i = j$.

Lemma 3.18. *The ℓ_1 condition number of $R(\alpha)$ is at least $\frac{1}{2(\alpha_0+1)}$.*

Proof. As the entries $R(\alpha)_{i,j}$ is $\frac{\alpha_i \alpha_j}{\alpha_0(\alpha_0+1)}$ when $i \neq j$ and $\frac{\alpha_i(\alpha_i+1)}{\alpha_0(\alpha_0+1)}$ when $i = j$, after normalization $R(\alpha)$ is just the matrix $D' = \frac{1}{\alpha_0+1}(\alpha \times (1, 1, ..., 1) + I)$ where × is outer product and I is the identity matrix.

Let $\boldsymbol{x}$ be a vector such that $|\boldsymbol{x}|_1 = 1$ and $|D'\boldsymbol{x}|_1$ achieves the minimum in $\Gamma(R(\alpha))$ and let $I = \{i | \boldsymbol{x}_i \geq 0\}$ and let $J = \bar{I}$ be the complement. We can assume without loss of generality that $\sum_{i \in I} \boldsymbol{x}_i \geq |\sum_{i \in J} \boldsymbol{x}_i|$ (otherwise just take $-\boldsymbol{x}$ instead). The product $D'\boldsymbol{x}$ is $\frac{\sum \boldsymbol{x}_i}{\alpha_0+1}\alpha + \frac{1}{\alpha_0+1}\boldsymbol{x}$. The first term is a nonnegative vector and hence for each $i \in I$, $(D'\boldsymbol{x})^i \geq 0$.

This implies that

$$|D'\boldsymbol{x}|_1 \geq \frac{1}{\alpha_0+1}\sum_{i\in I}\boldsymbol{x}_i \geq \frac{1}{2(\alpha_0+1)}.$$

□

3.4.2 Recovering the Parameters of a Dirichlet Distribution

When the variance covariance matrix $R(\alpha)$ is recovered with error ϵ_R in ℓ_1 norm when viewed as a vector, we can use Algorithm 3.4: Dirichlet to compute the parameters for the Dirichlet.

Algorithm 3.4. Dirichlet Parameters

```
input matrix $R$
output Dirichlet parameters $\alpha$
  Set $\alpha/\alpha_0 = R\mathbf{1}$.
  Let $i$ be the row with smallest $\ell_1$ norm, let $u = R_{i.i}$ and $v = \alpha_i/\alpha_0$.
  Set $\alpha_0 = \frac{1-u/v}{u/v-v}$.
  Output $\alpha = \alpha_0 \cdot (\alpha/\alpha_0)$.
```

Theorem 3.19. *When the variance covariance matrix $R(\alpha)$ is recovered with error ϵ_R in ℓ_1 norm when viewed as a vector, the procedure* Dirichlet(R) *learns the parameter of the Dirichlet distribution with error at most $O(ar(\alpha_0+1)\epsilon_R)$.*

Proof. The α_i/α_0's all have error at most ϵ_R. The value u is $\frac{\alpha_i}{\alpha_0}\frac{\alpha_i+1}{\alpha_0+1} \pm \epsilon_R$ and the value v is $\alpha_i/\alpha_0 \pm \epsilon_R$. Since $v \geq 1/ar$ we know the error for u/v is at most $2ar\epsilon_R$. Finally we need to bound the denominator $\frac{\alpha_i+1}{\alpha_0+1} - \frac{\alpha_i}{\alpha_0} > \frac{1}{2(\alpha_0+1)}$ (since $\frac{\alpha_i}{\alpha_0} \leq 1/r \leq 1/2$). Thus the final error is at most $5ar(\alpha_0+1)\epsilon_R$. □

3.5 Obtaining Almost Anchor Words

In this section, we prove Theorem 3.9, which we restate here:

Theorem 3.20. *Suppose $M = AW$ where W and M are normalized to have rows sum up to 1, A is separable and W is γ-robustly simplicial. When $\epsilon < \gamma/100$ there is a polynomial time algorithm that given $\tilde{M}$ such that for all rows $|\tilde{M}^i - M^i|_1 < \epsilon$, finds r row (almost anchor words) in $\tilde{M}$. The i-th almost anchor word corresponds to a row in M that can be represented as $(1 - O(\epsilon/\gamma))W^i + O(\epsilon/\gamma)W^{-i}$. Here W^{-i} is a vector in the convex hull of other rows in W with unit length in ℓ_1 norm.*

The major weakness of the algorithm in [11] is that it only considers the ℓ_1 norm. However in an r dimensional simplex there is another natural measure of distance more suited to our purposes: since each point is a unique convex combination of the vertices, we can view this convex combination as a probability distribution and use statistical distance (on this representation) as a norm for points inside the simplex. We will in fact need a slight modification to this norm, since we would like it to extend to points outside the simplex too:

Definition 3.21 ((δ, ϵ)-close)**.** A point M'^j is (δ, ϵ)-close to M'^i if and only if

$$\min_{c_k \ge 0, \sum_{k=1}^n c_k = 1, c_j \ge 1-\delta} |M'^i - \sum_{k=1}^n c_k M'^k|_1 \le \epsilon.$$

Intuitively think of point M'^j is (δ, ϵ)-close to M'^i if M'^i is ϵ close in ℓ_1 distance to some point Q, where Q is a convex combination of the rows of M' that places at least $1 - \delta$ weight on M'^j. Notice that this definition is not a distance since it is not symmetric, but we will abuse notation and nevertheless call it a distance function. We remark that this distance is easy to compute: To check whether M'^j is (δ, ϵ)-close to M'^i we just need to solve a linear program that minimizes the ℓ_1 distance when the c vector is a probability distribution with at least $1 - \delta$ weight on j (the constraints on c are clearly all linear).

We also consider all points that a row M'^j is close to, this is called the neighborhood of M'^j.

Definition 3.22 ((δ,ϵ)-neighborhood)**.** The (δ,ϵ)-neighborhood of M'^j are the rows M'^i such that M'^j is (δ,ϵ)-close to M'^i.

For each point M'^j, we know its original (unperturbed) point M_j is in a convex combination of W^i's: $M^j = \sum_{i=1}^r A_{j,i}W^i$. Separability implies that for any column index i there is a row $f(i)$ in A whose only nonzero entry is in the i^{th} column. Then $M^{f(i)} = W^i$ and consequently $|M'^{f(i)} - W^i|_1 < \epsilon$. Let us call these rows $M'^{f(i)}$ for all i the *canonical rows*. From the above description the following claim is clear.

Claim. *Every row M'^j has ℓ_1-distance at most 2ϵ to the convex hull of canonical rows.*

Proof. We have:

$$|M'^j - \sum_{k=1}^r A_{j,k}M'^{f(k)}|_1 \leq |M'^j - M^j|_1 + |M^j - \sum_{k=1}^r A_{j,k}M^{f(k)}|_1 + |\sum_{k=1}^r A_{j,k}(M^{f(k)} - M'^{f(k)})|_1$$

and we can bound the right hand side by 2ϵ. □

The algorithm will distinguish rows that are close to vertices and rows that are far by testing whether each row is close (in ℓ_1 norm) to the convex hull of rows outside its neighborhood. In particular, we define a robust loner as:

Definition 3.23 (robust loner)**.** We call a row M'^j a robust-loner if it has ℓ_1 distance at most 2ϵ to the convex hull of rows that are outside its $(6\epsilon/\gamma, 2\epsilon)$ neighborhood.

Our goal is to show that a row is a robust loner if and only if it is close to some row in W. The following lemma establishes one direction:

Lemma 3.24. *If $A_{j,t}$ is smaller than $1 - 10\epsilon/\gamma$, the point M'^j cannot be $(6\epsilon/\gamma, 2\epsilon)$-close to the canonical row that corresponds to $W^\top$.*

Proof. Assume towards contradiction that M'^j is $(6\epsilon/\gamma, 2\epsilon)$-close to the canonical row M'^i which is a perturbation of $W^\top$. By definition there must be probability distribution $c \in \mathbb{R}^n$

over the rows such that $c_j \geq 1-6\epsilon/\gamma$, and $|M'^i - \sum_{k=1}^n c_k M'^k|_1 \leq 2\epsilon$. Now we instead consider the unperturbed matrix M, since every row of M' is ϵ close (in ℓ_1 norm) to M we know $|M^i - \sum_{k=1}^n c_k M^k|_1 \leq 4\epsilon$. Now we represent M^i and $\sum_{k=1}^n c_k M^k$ as convex combinations of rows of W and consider the coefficient on $W^\top$. Clearly $M^i = W^\top$ so the coefficient is 1. But for $\sum_{k=1}^n c_k M^k$, since $c_j \geq 1 - 6\epsilon/\gamma$ and the coefficient $A_{j,t} \leq 1 - 10\epsilon/\gamma$, we know the coefficient of $W^\top$ in the sum must be strictly smaller than $1 - 10\epsilon/\gamma + 6\epsilon/\gamma = 1 - 4\epsilon/\gamma$. By the robustly simplicial assumption M^i and $\sum_{k=1}^n c_k M^k$ must be more than $4\epsilon/\gamma \cdot \gamma = 4\epsilon$ apart in ℓ_1 norm, which contradicts our assumption. □

As a corollary:

Corollary 3.25. *If $A_{j,t}$ is smaller than $1 - 10\epsilon/\gamma$ for all t, the row M'^j cannot be a robust loner.*

Proof. By the above lemma, we know the canonical rows are not in the $(6\epsilon/\gamma, 2\epsilon)$ neighborhood of M'^j. Thus by Claim 3.5 the row is close to the convex hull of canonical rows and cannot be a robust loner. □

Next we prove the other direction: a canonical row is necessarily a robust loner:

Lemma 3.26. *All canonical rows are robust loners.*

Proof. Suppose M'^i is a canonical row that corresponds to $W^\top$. We first observe that all the rows that are outside the $(6\epsilon/\gamma, 2\epsilon)$ neighborhood of M'^j must have $A_{j,t} < 1 - 6\epsilon/\gamma$. This is because when $A_{j,t} \geq 1 - 6\epsilon/\gamma$ we have $M^j - \sum_{k=1}^r A_{j,t} W^\top = \mathbf{0}$. If we replace M^j by M'^j and $W^\top$ by the corresponding canonical row, the distance is still at most 2ϵ and the coefficient on M'^i is at least $1 - 6\epsilon/\gamma$. By definition the corresponding row M'^j must be in the neighborhood of M'^i.

Now we try to represent M'^i with convex combination of rows that has $A_{j,t} < 1 - 6\epsilon/\gamma$. However this is impossible because every point in the convex combination will also have weight smaller than $1 - 6\epsilon/\gamma$ on $W^\top$, while M^i has weight 1 on $W^\top$. The ℓ_1 distance between M_i and the convex combination of the M_j's where $A_{j,t} < 1 - 6\epsilon/\gamma$ is a least 6ϵ by robust simplicial property. Even when the points are perturbed by ϵ (in ℓ_1) the distance can change by at most 2ϵ and is still more than 2ϵ. Therefore M'^i is a robust loner. □

Now we can prove the main theorem of this section:

Proof. Suppose we know γ and $100\epsilon < \gamma$.When γ is so small we have the following claim:

Claim. *If $A_{j,t}$ and $A_{i,l}$ is at least $1 - 10\epsilon/\gamma$, and $t \neq l$, then M'^j cannot be $(10\epsilon/\gamma, 2\epsilon)$-close to M'^i and vice versa.*

The proof is almost identical to Lemma 3.24. Also, the canonical row that corresponds to $W^\top$ is $(10\epsilon/\gamma, 2\epsilon)$ close to all rows with $A_{j,t} \geq 1 - 10\epsilon/\gamma$. Thus if we connect two robust loners when one is $(10\epsilon/\gamma, 2\epsilon)$ close to the other, the connected component of the graph will exactly be a partition according to the row in W that the robust loner is close to. We pick one robust loner in each connected component to get the almost anchor words.

Now suppose we don't know γ. In this case the problem is we don't know what is the right size of neighborhood to look at. However, since we know $\gamma > 100\epsilon$, we shall first run the algorithm with $\gamma = 100\epsilon$ to get r rows W' that are very close to the true rows in W. It is not hard to show that these rows are at least $\gamma/2$ robustly simplicial and at most $\gamma + 2\epsilon$ robustly simplicial. Therefore we can compute the $\gamma(W')$ parameter for this set of rows and use $\gamma(W') - 2\epsilon$ as the γ parameter. □

3.6 Maximum Likelihood Estimation is Hard

Here we prove that computing the Maximum Likelihood Estimate (MLE) of the parameters of a topic model is NP-hard. We call this problem the Topic Model Maximum Likelihoood Estimation (TM-MLE) problem:

Definition 3.27 (TM-MLE)**.** Given m documents and a target of r topics, the TM-MLE problem asks to compute the topic matrix A that has the largest probability of generating the observed documents (when the columns of W are generated by a uniform Dirichlet distribution).

Surprisingly, this appears to be the first proof that computing the MLE estimate in a topic model is indeed computationally hard, although its hardness is certainly to be expected. On a related note, Sontag and Roy [143] recently proved that *given the topic matrix* and a document, computing the Maximum A Posteriori (MAP) estimate for the distribution on topics that generated this document is NP-hard. Here we will establish that TM-MLE is NP-hard via a reduction from the MIN-BISECTION problem: In MIN-BISECTION the input is a graph with n vertices (n is an even integer), and the goal is to partition the vertices into two equal sized sets of $n/2$ vertices each so as to minimize the number of edges crossing the cut.

Theorem 3.28. *There is a polynomial time reduction from MIN-BISECTION to TM-MLE ($r = 2$).*

Proof. Suppose we are given an instance G of the MIN-BISECTION problem with n vertices and m edges. We will now define an instance of the TM-MLE problem. First, we set the number of words to be n. For each word i, we construct $N = \lceil 200m^3 \log n \rceil$ documents each of which contain the word i twice and no other words. For each edge in the graph G, we construct a document whose two words correspond to the endpoints of the edge.

Suppose that $x = (x_1, x_2)^\top$ is generated by the Dirichlet distribution $Dir(1, 1)$. Consequently the probability that words i and j appear in a document with only two words is exactly $(A^i x) \cdot (A^j x)$. We can take the expectation of this term over the Dirichlet distribution $Dir(1, 1)$ and hence the probability that a document (with exactly two words) contains the words i and j is

$$\mathbb{E}[(A^i x) \cdot (A^j x)] = \frac{1}{3}(A^i \cdot A^j) + \frac{1}{6}(A^i_1 A^j_2 + A^j_1 A^i_2)$$

In the TM-MLE problem, our goal is to maximize the following objective function (which is the log of the probability of generating the collection of documents):

$$OBJ = \sum_{\text{document} = \{i,j\}} \log \left[\frac{1}{3}(A^i \cdot A^j) + \frac{1}{6}(A_1^i A_2^j + A_1^j A_2^i) \right].$$

For any bisection, we define a canonical solution: the first topic is uniform on all words on one side of the bisection and the second topic is uniform on all words on the other side of the bisection.To prove the correctness of our reduction, a key step is to show that any candidate solution to the MLE problem must be close to a canonical solution. In particular, we show the following:

1. The rows A^i have almost the same ℓ_1 norm.
2. In each row A^i, almost all of the weight will be in one of the two topics.

Indeed, canonical solutions have large objective value. Any canonical solution has objective value at least $-Nn \log 3n^2/4 - m \log 3n^2/2$ (this is because documents with same words contribute $-\log 3n^2/4$ and documents with different words contribute at least $-\log 3n^2/2$).

Recall, in our reduction N is large. Roughly, if one of the rows has ℓ_1 norm that is bounded away from $2/n$ by at least $1/20nm$, the contribution (to the objective function) of documents with a repeated word will decrease significantly and the solution cannot be optimal. To prove this we use the fact that the function $\log x^2 = 2\log x$ is concave. Therefore when one of the rows has ℓ_1 norm more than $2/n + 1/20nm$, the optimal value for documents with a repeated word will be attained when all other rows have the same ℓ_1 norm $2/n - 1/20nm(n-1)$. Using a Taylor expansion, we conclude that the sum of terms for documents with a repeated word will decrease by at least $N/50m^2$ which is much larger than any effect the remaining m documents can recoup. In fact, an identical argument establishes that in each row A^i, the topic with smaller weight will always have weight smaller than $1/20nm$.

Now we claim that among canonical solutions, the one with largest objective value corresponds to a minimum bisection. The proof follows from the observation that the value of the objective function is $-Nn \log 3n^2/4 - k \log 3n^2/2 - (m-k) \log 3n^2/4$ for canonical solutions, where k is the number of edges cut by the bisection. In particular, the objective function of the minimum bisection will be at least an additive $\log 2$ larger than the objective function of a non-minimum bisection.

However, even if the canonical solution is perturbed by $1/20nm$, the objective function will only change by at most $m \cdot 1/10m = 1/10$, which is much smaller than $\frac{\log 2}{2}$. And this completes our reduction. □

We remark that the canonical solutions in our reduction are all *separable*, and hence this reduction applies even when the topic matrix A is known (and required) to be separable. So, even in the case of a separable topic matrix, it is NP-hard to compute the MLE.

Chapter 4

Practical Implementations and Experiments

In this chapter we show how to implement the algorithm in Chapter 3 efficiently. The algorithm produces results comparable to the best MCMC implementations while running orders of magnitude faster.

The algorithm in Chapter 3 is not very practical for several reasons. First, the algorithm requires solving many linear programs, which are very slow in practice. We need to replace linear programming with a combinatorial anchor-word selection algorithm. Second, the recovery algorithm relies on matrix inversion, which is notoriously unstable and can potentially produce negative entries. In this chapter we present a simple probabilistic interpretation of topic recovery given anchor words that replaces matrix inversion with a new gradient-based inference method.

After implementing the algorithm, we study both the empirical sample complexity of the algorithms on synthetic distributions and the performance of the algorithms on real-world document corpora. We find that our algorithm performs as well as collapsed Gibbs sampling on a variety of metrics, and runs at least an order of magnitude faster.

Algorithm 4.1. High Level Algorithm

```
input Textual corpus $\mathcal{D}$, Number of anchors $K$, Tolerance parameters $\epsilon_a, \epsilon_b > 0$.
output Word-topic matrix $A$, topic-topic matrix $R$
  $Q \leftarrow$ Word Co-occurences($\mathcal{D}$)
  Form $\{\bar{Q}_1, \bar{Q}_2, ...\bar{Q}_V\}$, the normalized rows of $Q$.
  $\mathbf{S} \leftarrow$ FastAnchorWords($\{\bar{Q}_1, \bar{Q}_2, ...\bar{Q}_V\}$, $K$, $\epsilon_a$) (Algorithm 4.3)
  $A, R \leftarrow$ RecoverKL($Q, \mathbf{S}, \epsilon_b$) (Algorithm 4.2)
  return $A, R$
```

The rest of the chapter is organized as follows: in Section 4.1 we introduce the main framework of the algorithm, and show how the *recovery* step can be replaced by a probabilistic approach. The proofs of new recovery algorithms are deferred to Section 4.4. In Section 4.2 we give a combinatorial NMF algorithm that is much faster than the ones in previous Chapters. The proof of the algorithm is deferred to Section 4.5. Section 4.3 introduces the methodology of experiments and present the results.

4.1 A Probabilistic Approach to Exploiting Separability

Recall that the algorithm in Chapter 3 has two steps: *anchor selection*, which identifies anchor words, and *recovery*, which recovers the parameters of A and of τ. Both anchor selection and recovery take as input the matrix Q (of size $V \times V$) of word-word co-occurrence counts, whose construction is described in the supplementary material. Q is normalized so that the sum of all entries is 1. The high-level flow of our complete learning algorithm is described in Algorithm 4.1, and follows the same two steps. In this section we will introduce a new recovery method based on a probabilistic framework. We defer the discussion of anchor selection to the next section, where we provide a purely combinatorial algorithm for finding the anchor words.

Algorithm 4.2. RecoverKL

```
input  Matrix $Q$, Set of anchor words $\mathbf{S}$, tolerance parameter $\epsilon$.
output  Matrices $A$,$R$
  Normalize the rows of $Q$ to form $\bar{Q}$.
  Store the normalization constants $\boldsymbol{p}_w = Q\mathbf{1}$.
  $\bar{Q}_{s_k}$ is the row of $\bar{Q}$ for the $k^{th}$ anchor word.
  for $i = 1, ..., V$ do
    Solve $C_{i\cdot} = \arg\min_{\boldsymbol{C}_i} D_{KL}(\bar{Q}_i || \sum_{k \in \mathbf{S}} C_{i,k} \bar{Q}_{s_k})$
    Subject to: $\sum_k C_{i,k} = 1$ and $C_{i,k} \geq 0$
    With tolerance: $\epsilon$
  end for
  $A' = \text{diag}(\boldsymbol{p}_w) C$
  Normalize the columns of $A'$ to form $A$.
  $R = A^{\dagger} Q A^{\dagger T}$
  return  $A, R$
```

Here we adopt a new *probabilistic* approach, which we describe below after introducing some notation. Consider any two words in a document and call them w_1 and w_2, and let z_1 and z_2 refer to their topic assignments. We will use $A_{i,k}$ to index the matrix of word-topic distributions, i.e. $A_{i,k} = \Pr(w_1 = i | z_1 = k) = \Pr(w_2 = i | z_2 = k)$. Given infinite data, the elements of the Q matrix can be interpreted as $Q_{i,j} = \Pr(w_1 = i, w_2 = j)$. The row-normalized Q matrix, denoted $\bar{Q}$, which plays a role in both finding the anchor words and the recovery step, can be interpreted as a conditional probability $\bar{Q}_{i,j} = \Pr(w_2 = j | w_1 = i)$.

Denoting the indices of the anchor words as $\mathbf{S} = \{s_1, s_2, ..., s_K\}$, the rows indexed by elements of $\mathbf{S}$ are special in that every other row of $\bar{Q}$ lies in the convex hull of the rows indexed by the anchor words. To see this, first note that for an anchor word s_k,

$$\begin{aligned} \bar{Q}_{s_k,j} &= \sum_{k'} \Pr(z_1 = k' | w_1 = s_k) \Pr(w_2 = j | z_1 = k') && (4.1) \\ &= \Pr(w_2 = j | z_1 = k), && (4.2) \end{aligned}$$

where (4.1) uses the fact that in an admixture model $w_2 \perp w_1 \mid z_1$, and (4.2) is because $\Pr(z_1 = k|w_1 = s_k) = 1$. For any other word i, we have

$$\bar{Q}_{i,j} \;=\; \sum_k \Pr(z_1 = k|w_1 = i) \Pr(w_2 = j|z_1 = k).$$

Denoting the probability $\Pr(z_1 = k|w_1 = i)$ as $C_{i,k}$, we have $\bar{Q}_{i,j} = \sum_k C_{i,k} \bar{Q}_{s_k,j}$. Since C is non-negative and $\sum_k C_{i,k} = 1$, we have that any row of $\bar{Q}$ lies in the convex hull of the rows corresponding to the anchor words. The mixing weights give us $\Pr(z_1|w_1 = i)$! Using this together with $\Pr(w_1 = i)$, we can recover the A matrix simply by using Bayes' rule:

$$\Pr(w_1 = i|z_1 = k) = \frac{\Pr(z_1 = k|w_1 = i) \Pr(w_1 = i)}{\sum_{i'} \Pr(z_1 = k|w_1 = i') p(w_1 = i')}.$$

Finally, we observe that $\Pr(w_1 = i)$ is easy to solve for since $\sum_j Q_{i,j} = \sum_j p(w_1 = i, w_2 = j) = \Pr(w_1 = i)$.

Our new algorithm finds, for each row of the empirical row normalized co-occurrence matrix, $\hat{Q}_i$, the coefficients $\Pr(z_1|w_1 = i)$ that best reconstruct it as a convex combination of the rows that correspond to anchor words. This step can be solved quickly and in parallel (independently) for each word using the exponentiated gradient algorithm. Once we have $\Pr(z_1|w_1)$, we recover the A matrix using Bayes' rule. The full algorithm using KL divergence as an objective is found in Algorithm 4.2. Further details of the exponentiated gradient algorithm are given in the supplementary material.

One reason to use KL divergence as the measure of reconstruction error is that the recovery procedure can then be understood as maximum likelihood estimation. In particular, we seek the parameters $\Pr(w_1)$, $\Pr(z_1|w_1)$, $\Pr(w_2|z_1)$ that maximize the likelihood of observing the *word co-occurence counts*, $\hat{Q}$. However, the optimization problem does not explicitly constrain the parameters to correspond an admixture model.

We can also define a similar algorithm using quadratic loss, which we call RecoverL2. This formulation has the extremely useful property that both the objective and gradient can be kernelized so that the optimization problem is independent of the vocabulary size. To see this, notice that the objective can be re-written as

$$||\overline{Q}_i - C_i^T \overline{Q}_{\mathbf{S}}||^2 = ||\overline{Q}_i||^2 - 2C_i(\overline{Q}_{\mathbf{S}} \overline{Q}_i^T) + C_i^T(\overline{Q}_{\mathbf{S}} \overline{Q}_{\mathbf{S}}^T)C_i,$$

where $\overline{Q}_{\mathbf{S}} \overline{Q}_{\mathbf{S}}^T$ is $K \times K$ and can be computed once and used for all words, and $\overline{Q}_{\mathbf{S}} \overline{Q}_i^T$ is $K \times 1$ and can be computed once prior to running the exponentiated gradient algorithm for word i.

To recover the R matrix for an admixture model, recall that $Q = ARA^T$. This may be an over-constrained system of equations with no solution for R, but we can find a least-squares approximation to R by pre- and post-multiplying Q by the pseudo-inverse $A^\dagger$. For the special case of LDA we can learn the Dirichlet hyperparameters. Recall that in applying Bayes' rule we calculated $\Pr(z_1) = \sum_{i'} \Pr(z_1|w_1 = i') \Pr(w_1 = i')$. These values for $\Pr(z)$ specify the Dirichlet hyperparameters up to a constant scaling. This constant could be recovered from the R matrix using Algorithm 3.4, but in practice we find it is better to choose it using a grid search to maximize the likelihood of the training data.

4.2 A Combinatorial Algorithm for Finding Anchor Words

Here we consider the *anchor selection* step of the algorithm where our goal is to find the anchor words. In the *infinite data* case where we have infinitely many documents, the convex hull of the rows in $\overline{Q}$ will be a simplex where the vertices of this simplex correspond to the anchor words.

Algorithm 4.3. FastAnchorWords

```
Input: $V$ points $\{\boldsymbol{d_1}, \boldsymbol{d_2}, ...\boldsymbol{d_V}\}$ in $V$ dimensions, almost in a simplex with $K$ vertices and $\epsilon > 0$
Output: $K$ points that are close to the vertices of the simplex.
  Project the points $\boldsymbol{d_i}$ to a randomly chosen $4 \log V/\epsilon^2$ dimensional subspace
  $S \leftarrow \{\boldsymbol{d_i}\}$ s.t. $\boldsymbol{d_i}$ is the farthest point from the origin.
  for $i = 1$ TO $K - 1$ do
    Let $d_j$ be the point in $\{\boldsymbol{d_1}, \ldots, \boldsymbol{d_V}\}$ that has the largest distance to span($S$).
    $S \leftarrow S \cup \{\boldsymbol{d_j}\}$.
  end for
  $S = \{\boldsymbol{v'_1}, \boldsymbol{v'_2}, ...\boldsymbol{v'_K}\}$.
  for $i = 1$ TO $K$ do
    Let $\boldsymbol{d_j}$ be the point that has the largest distance to span($\{\boldsymbol{v'_1}, \boldsymbol{v'_2}, ..., \boldsymbol{v'_K}\} \backslash \{\boldsymbol{v'_i}\}$)
    Update $\boldsymbol{v'_i}$ to $\boldsymbol{d_j}$
  end for
  return $\{\boldsymbol{v'_1}, \boldsymbol{v'_2}, ..., \boldsymbol{v'_K}\}$.
```

Notation: span(S) denotes the subspace spanned by the points in the set S. We compute the distance from a point x to the subspace span(S) by computing the norm of the projection of x onto the orthogonal complement of span(S).

Since we only have a finite number of documents, the rows of $\overline{Q}$ are only an approximation to their expectation. We are therefore given a set of V points $\boldsymbol{d_1}, \boldsymbol{d_2}, ...\boldsymbol{d_V}$ that are each a perturbation of $\boldsymbol{a_1}, \boldsymbol{a_2}, ...\boldsymbol{a_V}$ whose convex hull P defines a simplex. We would like to find an approximation to the vertices of P. This is the same as the problems we solved in Section 2.5 and Section 3.5.

In this section we describe a purely combinatorial algorithm for this task that avoids linear programming altogether. The new "FastAnchorWords" algorithm is given in Algorithm 4.3. To find all of the anchor words, our algorithm iteratively finds the furthest point from the subspace spanned by the anchor words found so far.

Since the points we are given are perturbations of the true points, we cannot hope to find the anchor words exactly. Nevertheless, the intuition is that even if one has only found r points S that are close to r (distinct) anchor words, the point that is furthest from span(S) will itself be close to a (new) anchor word. The additional advantage of this procedure is that when faced with many choices for a next anchor word to find, our algorithm tends to find the one that is most different than the ones we have found so far.

The main contribution of this section is a proof that the FastAnchorWords algorithm succeeds in finding K points that are close to anchor words. To precisely state the guarantees, we use a similar notion of robustness as Section 2.5 (but notice that they are different because of different norms used):

Definition 4.1. A simplex P is γ-robust if for every vertex v of P, the ℓ_2 distance between v and the convex hull of the rest of the vertices is at least γ.

In most reasonable settings the parameters of the topic model define lower bounds on the robustness of the polytope P. For example, in LDA, this lower bound is based on the largest ratio of any pair of hyper-parameters in the model [12]. Our goal is to find a set of points that are close to the vertices of the simplex, and to make this precise we introduce the following definition:

Definition 4.2. Let $\boldsymbol{a_1}, \boldsymbol{a_2}, ...\boldsymbol{a_V}$ be a set of points whose convex hull P is a simplex with vertices $\boldsymbol{v_1}, \boldsymbol{v_2}, ...\boldsymbol{v_K}$. Then we say $\boldsymbol{a_i}$ ϵ-covers $\boldsymbol{v_j}$ if when $\boldsymbol{a_j}$ is written as a convex combination of the vertices as $\boldsymbol{a_i} = \sum_j c_j \boldsymbol{v_j}$, then $c_j \geq 1 - \epsilon$. Furthermore we will say that a set of K points ϵ-covers the vertices if each vertex is ϵ covered by some point in the set.

We will prove the following theorem: suppose there is a set of points $\mathcal{A} = \boldsymbol{a_1}, \boldsymbol{a_2}, ...\boldsymbol{a_V}$ whose convex hull P is γ-robust and has vertices $\boldsymbol{v_1}, \boldsymbol{v_2}, ...\boldsymbol{v_K}$ (which appear in $\mathcal{A}$) and that we are given a perturbation $\boldsymbol{d_1}, \boldsymbol{d_2}, ...\boldsymbol{d_V}$ of the points so that for each i, $\|\boldsymbol{a_i} - \boldsymbol{d_i}\| \leq \epsilon$, then:

Theorem 4.3. *There is a combinatorial algorithm that runs in time $\tilde{O}(V^2 + VK/\epsilon^2)$ [1] and outputs a subset of $\{\boldsymbol{d_1}, \ldots, \boldsymbol{d_V}\}$ of size K that $O(\epsilon/\gamma)$-covers the vertices provided that $20K\epsilon/\gamma^2 < \gamma$.*

[1] In practice we find setting dimension to 1000 works well. The running time is then $O(V^2 + 1000VK)$.

This new algorithm not only helps us avoid linear programming altogether in inferring the parameters of a topic model, but also can be used to solve the nonnegative matrix factorization problem under the separability assumption, again without resorting to linear programming. Our analysis rests on the following lemmas, whose proof we defer to the supplementary material. Suppose the algorithm has found a set S of k points that are each δ-close to distinct vertices in $\{\boldsymbol{v_1}, \boldsymbol{v_2}, ..., \boldsymbol{v_K}\}$ and that $\delta < \gamma/20K$.

Lemma 4.4. *There is a vertex $\boldsymbol{v_i}$ whose distance from* $\mathrm{span}(S)$ *is at least $\gamma/2$.*

The proof of this lemma is based on a volume argument, and the connection between the volume of a simplex and the determinant of the matrix of distances between its vertices.

Lemma 4.5. *The point $\boldsymbol{d_j}$ found by the algorithm must be $\delta = O(\epsilon/\gamma^2)$ close to some vertex $\boldsymbol{v_i}$.*

This lemma is used to show that the error does not accumulate too badly in our algorithm, since δ only depends on ϵ, γ (not on the δ used in the previous step of the algorithm). This prevents the error from accumulating exponentially in the dimension of the problem, which would be catastrophic for our proof.

After running the first phase of our algorithm, we run a cleanup phase (the second loop in Alg. 4.3) that can reduce the error in our algorithm. When we have $K-1$ points close to $K-1$ vertices, only one of the vertices can be far from their span. The farthest point must be close to this missing vertex. The following lemma shows that this cleanup phase can improve the guarantees of Lemma 4.5:

Lemma 4.6. *Suppose $|S| = K-1$ and each point in S is $\delta = O(\epsilon/\gamma^2) < \gamma/20K$ close to some vertex $\boldsymbol{v_i}$, then the farthest point $\boldsymbol{v_j'}$ found by the algorithm is $1 - O(\epsilon/\gamma)$ close to the remaining vertex.*

This algorithm is a greedy approach to maximizing the volume of the simplex. The larger the volume is, the more words per document the resulting model can explain. Better anchor word selection is an open question for future work. We have experimented with a variety of other heuristics for maximizing simplex volume, with varying degrees of success.

4.3 Experimental Results

We compare three parameter recovery methods, Recover [12], RecoverKL and RecoverL2 to a fast implementation of Gibbs sampling [122].[2] Linear programming-based anchor word finding is too slow to be comparable, so we use FastAnchorWords for all three recovery algorithms. Using Gibbs sampling we obtain the word-topic distributions by averaging over 10 saved states, each separated by 100 iterations, after 1000 burn-in iterations.

4.3.1 Methodology

We train models on two synthetic data sets to evaluate performance when model assumptions are correct, and real documents to evaluate real-world performance. To ensure that synthetic documents resemble the dimensionality and sparsity characteristics of real data, we generate *semi-synthetic* corpora. For each real corpus, we train a model using MCMC and then generate new documents using the parameters of that model (these parameters are *not* guaranteed to be separable).

We use two real-world data sets, a large corpus of **New York Times** articles (295k documents, vocabulary size 15k, mean document length 298) and a small corpus of **NIPS** abstracts (1100 documents, vocabulary size 2500, mean length 68). Vocabularies were pruned with document frequency cutoffs. We generate semi-synthetic corpora of various sizes from models trained with $K = 100$ from NY Times and NIPS, with document lengths set to 300 and 70, respectively, and with document-topic distributions drawn from a Dirichlet with symmetric hyperparameters 0.03.

[2] We were not able to obtain [9]'s implementation of their algorithm, and our own implementation is too slow to be practical.

We use a variety of metrics to evaluate models: For the semi-synthetic corpora, we can compute **reconstruction error** between the true word-topic matrix A and learned topic distributions. Given a learned matrix $\hat{A}$ and the true matrix A, we use an LP to find the best matching between topics. Once topics are aligned, we evaluate ℓ_1 distance between each pair of topics. When true parameters are not available, a standard evaluation for topic models is to compute **held-out probability**, the probability of previously unseen documents under the learned model. This computation is intractable but there are reliable approximation methods [33, 158]. Topic models are useful because they provide interpretable latent dimensions. We can evaluate the **semantic quality** of individual topics using a metric called *Coherence*. Coherence is based on two functions, $D(w)$ and $D(w_1, w_2)$, which are number of documents with at least one instance of w, and of w_1 and w_2, respectively [126]. Given a set of words $\mathcal{W}$, coherence is

$$Coherence(\mathcal{W}) = \sum_{w_1, w_2 \in \mathcal{W}} \log \frac{D(w_1, w_2) + \epsilon}{D(w_2)}. \tag{4.3}$$

The parameter $\epsilon = 0.01$ is used to avoid taking the log of zero for words that never co-occur [146]. This metric has been shown to correlate well with human judgments of topic quality. If we perfectly reconstruct topics, all the high-probability words in a topic should co-occur frequently, otherwise, the model may be mixing unrelated concepts. Coherence measures the quality of individual topics, but does not measure redundancy, so we measure **inter-topic similarity**. For each topic, we gather the set of the N most probable words. We then count how many of those words do not appear in any other topic's set of N most probable words. Some overlap is expected due to semantic ambiguity, but lower numbers of unique words indicate less useful models.

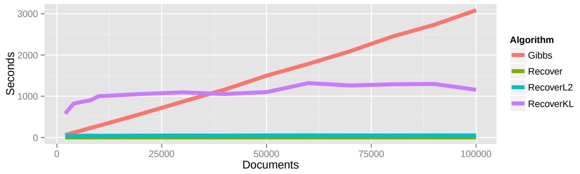


Figure 4.1: Training time on synthetic NIPS documents.

4.3.2 Efficiency

The Recover algorithms, in Python, are faster than a heavily optimized Java Gibbs sampling implementation [165]. Fig. 4.1 shows the time to train models on synthetic corpora on a single machine. Gibbs sampling is linear in the corpus size. RecoverL2 is also linear ($\rho = 0.79$), but only varies from 33 to 50 seconds. Estimating Q is linear, but takes only 7 seconds for the largest corpus. FastAnchorWords takes less than 6 seconds for all corpora.

4.3.3 Semi-synthetic documents

The new algorithms have good ℓ_1 reconstruction error on semi-synthetic documents, especially for larger corpora. Results for semi-synthetic corpora drawn from topics trained on NY Times articles are shown in Fig. 4.2 for corpus sizes ranging from 50k to 2M synthetic documents. In addition, we report results for the three Recover algorithms on "infinite data," that is, the true Q matrix from the model used to generate the documents. Error bars show variation between topics. Recover performs poorly in all but the noiseless, infinite data setting. Gibbs sampling has lower ℓ_1 with smaller corpora, while the new algorithms get better recovery and lower variance with more data (although more sampling might reduce MCMC error further).

Results for semi-synthetic corpora drawn from NIPS topics are shown in Fig. 4.3. Recover does poorly for the smallest corpora (topic matching fails for $D = 2000$, so ℓ_1 is not meaningful), but achieves moderate error for D comparable to the NY Times corpus. Re-

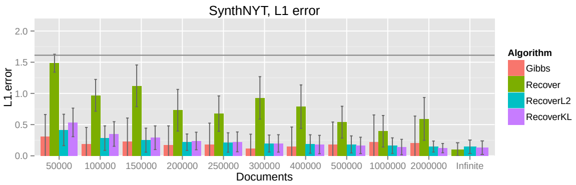


Figure 4.2: ℓ_1 error for a semi-synthetic model generated from a model trained on NY Times articles with $K = 100$. The horizontal line indicates the ℓ_1 error of K uniform distributions.

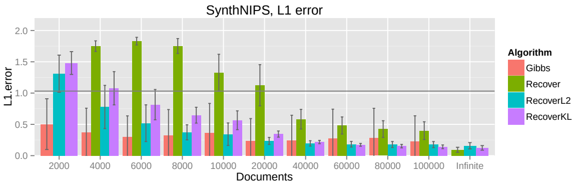


Figure 4.3: ℓ_1 error for a semi-synthetic model generated from a model trained on NIPS papers with $K = 100$. Recover fails for $D = 2000$.

coverKL and RecoverL2 also do poorly for the smallest corpora, but are comparable to or better than Gibbs sampling, with much lower variance, after 40,000 documents.

4.3.4 Effect of separability

The non-negative algorithms are more robust to violations of the separability assumption than the original Recover algorithm. In Fig. 4.3, Recover does not achieve zero ℓ_1 error even with noiseless "infinite" data. Here we show that this is due to lack of separability. In our semi-synthetic corpora, documents are generated from the LDA model, but the topic-word distributions are learned from data and may not satisfy the anchor words assumption. We test the sensitivity of algorithms to violations of the separability condition by adding a synthetic anchor word to each topic that is by construction unique to the topic. We assign the synthetic anchor word a probability equal to the most probable word in the original topic. This causes the distribution to sum to greater than 1.0, so we renormalize. Results are shown in Fig. 4.4. The ℓ_1 error goes to zero for Recover, and close to zero for RecoverKL

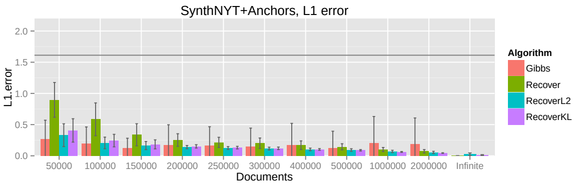


Figure 4.4: When we add artificial anchor words before generating synthetic documents, ℓ_1 error goes to zero for Recover and close to zero for RecoverKL and RecoverL2.

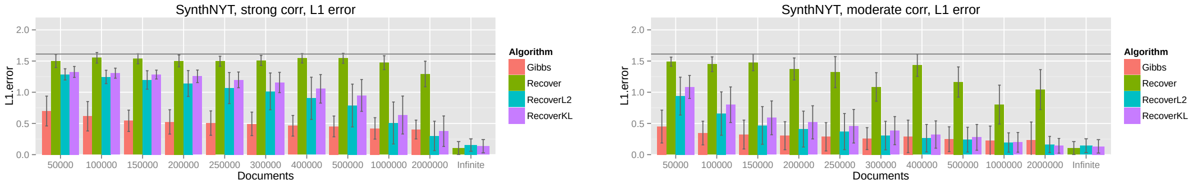


Figure 4.5: ℓ_1 error increases as we increase topic correlation. We use the same $K = 100$ topic model from NY Times articles, but add correlation: TOP $\rho = 0.05$, BOTTOM $\rho = 0.1$.

and RecoverL2. The reason RecoverKL and RecoverL2 do not reach exactly zero is because we do not solve the optimization problems to perfect optimality.

4.3.5 Effect of correlation

The theoretical guarantees of the new algorithms apply even if topics are correlated. To test how algorithms respond to correlation, we generated new synthetic corpora from the same $K = 100$ model trained on NY Times articles. Instead of a symmetric Dirichlet distribution, we use a logistic normal distribution with a block-structured covariance matrix. We partition topics into 10 groups. For each pair of topics in a group, we add a non-zero off-diagonal element to the covariance matrix. This block structure is not necessarily realistic, but shows the effect of correlation. Results for two levels of covariance ($\rho = 0.05, \rho = 0.1$) are shown in Fig. 4.5. Results for Recover are much worse in both cases than the Dirichlet-generated corpora in Fig. 4.2. The other three algorithms, especially Gibbs sampling, are more robust to correlation, but performance consistently degrades as correlation increases, and improves

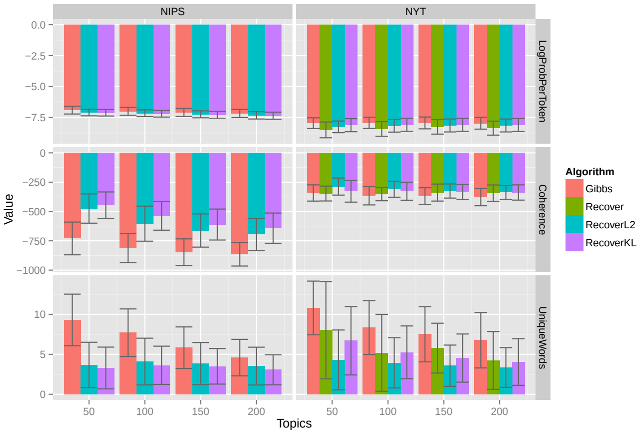


Figure 4.6: Held-out probability (per token) is similar for RecoverKL, RecoverL2, and Gibbs sampling. RecoverKL and RecoverL2 have better coherence, but fewer unique words than Gibbs. (Up is better for all three metrics.)

with larger corpora. With infinite data ℓ_1 error is equal to ℓ_1 error in the uncorrelated synthetic corpus (non-zero because of violations of the separability assumption).

4.3.6 Real documents

The new algorithms produce comparable quantitative and qualitative results on real data. Fig. 4.6 shows three metrics for both corpora. Error bars show the distribution of log probabilities across held-out *documents* (top panel) and coherence and unique words across *topics* (center and bottom panels). Held-out sets are 230 documents for NIPS and 59k for NY Times. For the small NIPS corpus we average over 5 non-overlapping train/test splits. The matrix-inversion in Recover failed for the smaller corpus (NIPS). In the larger corpus (NY Times), Recover produces noticeably worse held-out log probability per token than the other algorithms. Gibbs sampling produces the best average held-out probability ($p < 0.0001$ under a paired t-test), but the difference is within the range of variability between documents. We tried several methods for estimating hyperparameters, but the observed differences did not change the relative performance of algorithms. Gibbs sampling

Table 4.1: Example topic pairs from NY Times (closest ℓ_1), anchor words in bold. All 100 topics are in suppl. material.

RecoverL2	run inning game hit season zzz_anaheim_angel
Gibbs	run inning hit game ball pitch
RecoverL2	father family **zzz_elian** boy court zzz_miami
Gibbs	zzz_cuba zzz_miami cuban zzz_elian boy protest
RecoverL2	**file** sport read internet email zzz_los_angeles
Gibbs	web site com www mail zzz_internet

has worse coherence than the Recover algorithms, but produces more unique words per topic. These patterns are consistent with semi-synthetic results for similarly sized corpora (details are in supplementary material).

For each NY Times topic learned by RecoverL2 we find the closest Gibbs topic by ℓ_1 distance. The closest, median, and farthest topic pairs are shown in Table 4.1.[3] We observe that when there is a difference, recover-based topics tend to have more specific words (*Anaheim Angels* vs. *pitch*).

4.4 Proof for Nonnegative Recover Procedure

In order to show RecoverL2 learns the parameters even when the rows of $\bar{Q}$ are perturbed, we need the following lemma that shows when columns of $\bar{Q}$ are close to the expectation, the posteriors c computed by the algorithm is also close to the true value.

Lemma 4.7. *For a γ robust simplex S with vertices $\{\boldsymbol{v_1}, \boldsymbol{v_2}, ..., \boldsymbol{v_K}\}$, let $\boldsymbol{v}$ be a point in the simplex that can be represented as a convex combination $\boldsymbol{v} = \sum_{i=1}^K c_i \boldsymbol{v_i}$. If the vertices of S are perturbed to $S' = \{..., \boldsymbol{v_i'}, ...\}$ where $\|\boldsymbol{v_i'} - \boldsymbol{v_i}\| \leq \delta_1$ and $\boldsymbol{v}$ is perturbed to $\boldsymbol{v'}$ where $\|\boldsymbol{v} - \boldsymbol{v'}\| \leq \delta_2$. Let $\boldsymbol{v^*}$ be the point in S' that is closest to $\boldsymbol{v'}$, and $\boldsymbol{v^*} = \sum_{i=1}^K c_i' \boldsymbol{v_i}$, when $10\sqrt{K}\delta_1 \leq \gamma$ for all $i \in [K]$ $|c_i - c_i'| \leq 4(\delta_1 + \delta_2)/\gamma$.*

[3] The UCI NY Times corpus includes named-entity annotations, indicated by the *zzz* prefix.

Proof. Consider the point $\boldsymbol{u} = \sum_{i=1}^K c_i \boldsymbol{v_i'}$, by triangle inequality: $\|\boldsymbol{u} - \boldsymbol{v}\| \leq \sum_{i=1}^K c_i \|\boldsymbol{v_i} - \boldsymbol{v_i'}\| \leq \delta_1$. Hence $\|\boldsymbol{u} - \boldsymbol{v'}\| \leq \|\boldsymbol{u} - \boldsymbol{v}\| + \|\boldsymbol{v} - \boldsymbol{v'}\| \leq \delta_1 + \delta_2$, and $\boldsymbol{u}$ is in S'. The point $\boldsymbol{v^*}$ is the point in S' that is closest to $\boldsymbol{v'}$, so $\|\boldsymbol{v^*} - \boldsymbol{v'}\| \leq \delta_1 + \delta_2$ and $\|\boldsymbol{v^*} - \boldsymbol{u}\| \leq 2(\delta_1 + \delta_2)$.

Then we need to show when a point ($\boldsymbol{u}$) moves a small distance, its representation also changes by a small amount. Intuitively this is true because S is γ robust. By Lemma 4.4 when $10\sqrt{K}\delta_1 < \gamma$, the simplex S' is also $\gamma/2$ robust. For any i, let $Proj_i(\boldsymbol{v^*})$ and $Proj_i(\boldsymbol{u})$ be the projections of $\boldsymbol{v^*}$ and $\boldsymbol{u}$ in the orthogonal subspace of span$(S' \backslash \{\boldsymbol{v_i'}\})$, then

$$|c_i - c_i'| = \|Proj_i(\boldsymbol{v^*}) - Proj_i(\boldsymbol{u})\| / \text{dis}(\boldsymbol{v_i}, \text{span}(S' \backslash \{\boldsymbol{v_i'}\})) \leq 4(\delta_1 + \delta_2)/\gamma$$

and this completes the proof. □

With this lemma it is not hard to show that RecoverL2 has polynomial sample complexity.

Theorem 4.8. *When the number of documents M is at least*

$$\max\{O(aK^3 \log V/D(\gamma p)^6 \epsilon), O((aK)^3 \log V/D\epsilon^3(\gamma p)^4)\}$$

our algorithm using the conjunction of FastAnchorWords and RecoverL2 learns the A matrix with entry-wise error at most ϵ.

Proof. (sketch) We can assume without loss of generality that each word occurs with probability at least $\epsilon/4aK$ and furthermore that if M is at least $50 \log V/D\epsilon_Q^2$ then the empirical matrix $\tilde{Q}$ is entry-wise within an additive ϵ_Q to the true $Q = \frac{1}{M}\sum_{d=1}^M AW_dW_d^TA^T$ see [12] for the details. Also, the K anchor rows of $\bar{Q}$ form a simplex that is γp robust.

The error in each column of $\bar{Q}$ can be at most $\delta_2 = \epsilon_Q\sqrt{4aK/\epsilon}$. By Theorem 4.13 when $20K\delta_2/(\gamma p)^2 < \gamma p$ (which is satisfied when $M = O(aK^3 \log V/D(\gamma p)^6 \epsilon))$, the anchor words found are $\delta_1 = O(\delta_2/(\gamma p))$ close to the true anchor words. Hence by Lemma 4.7 every entry of C has error at most $O(\delta_2/(\gamma p)^2)$.

With such number of documents, all the word probabilities $p(w = i)$ are estimated more accurately than the entries of $C_{i,j}$, so we omit their perturbations here for simplicity. When we apply the Bayes rule, we know $A_{i,k} = C_{i,k}p(w = i)/p(z = k)$, where $p(z = k)$ is α_k which is lower bounded by $1/aK$. The numerator and denominator are all related to entries of C with positive coefficients sum up to at most 1. Therefore the errors δ_{num} and δ_{denom} are at most the error of a single entry of C, which is bounded by $O(\delta_2/(\gamma p)^2)$. Applying Taylor's Expansion to $(p(z = k, w = i) + \delta_{num})/(\alpha_k + \delta_{denom})$, the error on entries of A is at most $O(aK\delta_2/(\gamma p)^2)$. When $\epsilon_Q \le O((\gamma p)^2\epsilon^{1.5}/(aK)^{1.5})$, we have $O(aK\delta_2/(\gamma p)^2) \le \epsilon$, and get the desired accuracy of A. The number of document required is $M = O((aK)^3 \log V/D\epsilon^3(\gamma p)^4)$.

The sample complexity for R can then be bounded using matrix perturbation theory. □

4.5 Proof for Anchor-Words Finding Algorithm

Recall that the correctness of the algorithm depends on the following Lemmas:

Lemma 4.9. *There is a vertex $\boldsymbol{v_i}$ whose distance from* span(S) *is at least $\gamma/2$.*

Lemma 4.10. *The point Δ_j found by the algorithm must be $\delta = O(\epsilon/\gamma^2)$ close to some vertex $\boldsymbol{v_i}$.*

In order to prove Lemma 4.4, we use a volume argument. First we show that the volume of a robust simplex cannot change by too much when the vertices are perturbed.

Lemma 4.11. *Suppose $\{\boldsymbol{v_1}, \boldsymbol{v_2}, ..., \boldsymbol{v_K}\}$ are the vertices of a γ-robust simplex S. Let S' be a simplex with vertices $\{\boldsymbol{v'_1}, \boldsymbol{v'_2}, ..., \boldsymbol{v'_K}\}$, each of the vertices $\boldsymbol{v'_i}$ is a perturbation of $\boldsymbol{v_i}$ and $\|\boldsymbol{v'_i} - \boldsymbol{v_i}\|_2 \le \delta$. When $10\sqrt{K}\delta < \gamma$ the volume of the two simplices satisfy*

$$vol(S)(1 - 2\delta/\gamma)^{K-1} \le vol(S') \le vol(S)(1 + 4\delta/\gamma)^{K-1}.$$

Proof. As the volume of a simplex is proportional to the determinant of a matrix whose columns are the edges of the simplex, we first show the following perturbation bound for determinant.

Claim. *Let A, E be $K \times K$ matrices, the smallest eigenvalue of A is at least γ, the Frobenius norm $\|E\|_F \le \sqrt{K}\delta$, when $\gamma > 5\sqrt{K}\delta$ we have*

$$\det(A+E)/\det(A) \ge (1-\delta/\gamma)^K.$$

Proof. Since $\det(AB) = \det(A)\det(B)$, we can multiply both A and $A+E$ by A^{-1}. Hence $\det(A+E)/\det(A) = \det(I + A^{-1}E)$.

The Frobenius norm of $A^{-1}E$ is bounded by

$$\left\|A^{-1}E\right\|_F \le \left\|A^{-1}\right\| \|E\|_F \le \sqrt{K}\delta/\gamma.$$

Let the eigenvalues of $A^{-1}E$ be $\lambda_1, \lambda_2, ..., \lambda_K$, then by definition of Frobenius Norm $\sum_{i=1}^K \lambda_i^2 \le \|A^{-1}E\|_F^2 \le K\delta^2/\gamma^2$. The eigenvalues of $I + A^{-1}E$ are just $1+\lambda_1, 1+\lambda_2, ..., 1+\lambda_K$, and the determinant $\det(I + A^{-1}E) = \prod_{i=1}^K (1+\lambda_i)$. Hence it suffices to show

$$\min \prod_{i=1}^K (1+\lambda_i) \ge (1-\delta/\gamma)^K \text{ when } \sum_{i=1}^K \lambda_i^2 \le K\delta^2/\gamma^2.$$

To do this we apply Lagrangian method and show the minimum is only obtained when all λ_i's are equal. The optimal value must be obtained at a local optimum of

$$\prod_{i=1}^K (1+\lambda_i) + C\sum_{i=1}^K \lambda_i^2.$$

Taking partial derivatives with respect to λ_i's, we get the equations $-\lambda_i(1+\lambda_i) = -\prod_{i=1}^K (1+\lambda_i)/2C$ (here using $\sqrt{K}\delta/\gamma$ is small so $1+\lambda_i > 1/2 > 0$). The right hand

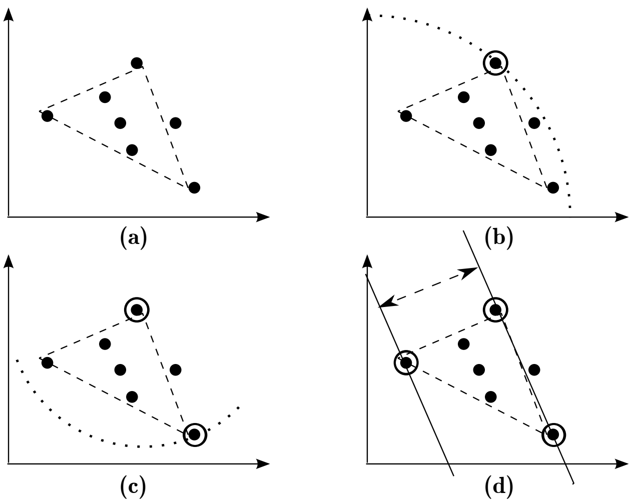


Figure 4.7: Illustration of the Algorithm

side is a constant, so each λ_i must be one of the two solutions of this equation. However, only one of the solution is larger than 1/2, therefore all the λ_i's are equal. □

For the lower bound, we can project the perturbed subspace to the $K-1$ dimensional space. Such a projection cannot increase the volume and the perturbation distances only get smaller. Therefore we can apply the claim directly, the columns of A are just $\boldsymbol{v_{i+1}} - \boldsymbol{v_1}$ for $i = 1, 2, ..., K-1$; columns of E are just $\boldsymbol{v'_{i+1}} - \boldsymbol{v_{i+1}} - (\boldsymbol{v'_1} - \boldsymbol{v_1})$. The smallest eigenvalue of A is at least γ because the polytope is γ robust, which is equivalent to saying after orthogonalization each column still has length at least γ. The Frobenius norm of E is at most $2\sqrt{K-1}\delta$. We get the lower bound directly by applying the claim.

For the upper bound, swap the two sets S and S' and use the argument for the lower bound. The only thing we need to show is that the smallest eigenvalue of the matrix generated by points in S' is still at least $\gamma/2$. This follows from Wedin's Theorem [160] and the fact that $\|E\| \le \|E\|_F \le \sqrt{K}\delta \le \gamma/2$. □

Now we are ready to prove Lemma 4.4.

Proof. The first case is for the first step of the algorithm, when we try to find the farthest point to the origin. Here essentially $S = \{\mathbf{0}\}$. For any two vertices $\boldsymbol{v_1}, \boldsymbol{v_2}$, since the simplex is γ robust, the distance between $\boldsymbol{v_1}$ and $\boldsymbol{v_2}$ is at least γ. Which means $\text{dis}(\mathbf{0}, \boldsymbol{v_1}) + \text{dis}(\mathbf{0}, \boldsymbol{v_2}) \geq \gamma$, one of them must be at least $\gamma/2$.

For the later steps, recall that S contains vertices of a perturbed simplex. Let S' be the set of original vertices corresponding to the perturbed vertices in S. Let $\boldsymbol{v}$ be any vertex in $\{\boldsymbol{v_1}, \boldsymbol{v_2}, ..., \boldsymbol{v_K}\}$ which is not in S. Now we know the distance between $\boldsymbol{v}$ and S is equal to $\text{vol}(S \cup \{\boldsymbol{v}\})/(|S| - 1)\text{vol}(S)$. On the other hand, we know $\text{vol}(S' \cup \{v\})/(|S'| - 1)\text{vol}(S') \geq \gamma$. Using Lemma 4.11 to bound the ratio between the two pairs $\text{vol}(S)/\text{vol}(S')$ and $\text{vol}(S \cup \{\boldsymbol{v}\})/\text{vol}(S' \cup \{\boldsymbol{v}\})$, we get:

$$\text{dis}(\boldsymbol{v}, S) \geq (1 - 4\epsilon'/\gamma)^{2|S|-2}\gamma > \gamma/2$$

when $\gamma > 20K\epsilon'$. □

Lemma 4.5 is based on the following observation: in a simplex the point with largest ℓ_2 is always a vertex. Even if two vertices have the same norm if they are not close to each other the vertices on the edge connecting them will have significantly lower norm.

Proof. (Lemma 4.5)

Since $\boldsymbol{d_j}$ is the point found by the algorithm, let us consider the point $\boldsymbol{a_j}$ before perturbation. The point $\boldsymbol{a_j}$ is inside the simplex, therefore we can write $\boldsymbol{a_j}$ as a convex combination of the vertices:

$$\boldsymbol{a_j} = \sum_{t=1}^{K} c_t \boldsymbol{v_t}$$

Let $\boldsymbol{v_t}$ be the vertex with largest coefficient c_t. Let Δ be the largest distance from some vertex to the space spanned by points in S ($\Delta = \max_l \text{dis}(\boldsymbol{v_l}, \text{span}(S))$. By Lemma 4.4 we know $\Delta > \gamma/2$. Also notice that we are not assuming $\text{dis}(\boldsymbol{v_t}, \text{span}(S)) = \Delta$.

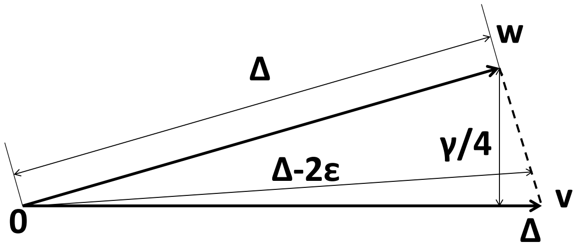


Figure 4.8: Proof of Lemma 4.5, after projecting to the orthogonal subspace of span(S).

Now we rewrite $\boldsymbol{a_j}$ as $c_t\boldsymbol{v_t} + (1-c_t)\boldsymbol{w}$, where $\boldsymbol{w}$ is a vector in the convex hull of vertices other than $\boldsymbol{v_t}$. Observe that $\boldsymbol{a_j}$ must be far from span(S), because $\boldsymbol{d_j}$ is the farthest point found by the algorithm. Indeed:

$$\mathrm{dis}(\boldsymbol{a_j}, \mathrm{span}(S)) \geq \mathrm{dis}(\boldsymbol{d_j}, \mathrm{span}(S)) - \epsilon \geq \mathrm{dis}(v_l, \mathrm{span}(S)) - 2\epsilon \geq \Delta - 2\epsilon$$

The second inequality is because there must be some point $\boldsymbol{d_l}$ that correspond to the farthest vertex v_l and have $\mathrm{dis}(\boldsymbol{d_l}, \mathrm{span}(S)) \geq \Delta - \epsilon$. Thus as $\boldsymbol{d_j}$ is the farthest point $\mathrm{dis}(\boldsymbol{d_j}, \mathrm{span}(S)) \geq \mathrm{dis}(\boldsymbol{d_l}, \mathrm{span}(S)) \geq \Delta - \epsilon$.

The point $\boldsymbol{a_j}$ is on the segment connecting $\boldsymbol{v_t}$ and w, the distance between $\boldsymbol{a_j}$ and span(S) is not much smaller than that of $\boldsymbol{v_t}$ and w. Following the intuition in ℓ_2 norm when $\boldsymbol{v_t}$ and w are far we would expect $\boldsymbol{a_j}$ to be very close to either $\boldsymbol{v_t}$ or w. Since $c_t \geq 1/K$ it cannot be really close to w, so it must be really close to $\boldsymbol{v_t}$. We formalize this intuition by the following calculation (see Figure 4.8):

Project everything to the orthogonal subspace of span(S) (points in span(S) are now at the origin). After projection distance to span(S) is just the ℓ_2 norm of a vector. Without loss of generality we assume $\|\boldsymbol{v_t}\|_2 = \|\boldsymbol{w}\| = \Delta$ because these two have length at most Δ, and extending these two vectors to have length Δ can only increase the length of $\boldsymbol{d_j}$.

The point $\boldsymbol{v_t}$ must be far from w by applying Lemma 4.4: consider the set of vertices $V' = \{v_i : v_i \text{ does not correspond to any point in } S \text{ and } i \neq t\}$. The set $V' \cup S$ satisfy the assumptions in Lemma 4.4 so there must be one vertex that is far from $\mathrm{span}(V' \cup S)$, and it can only be $\boldsymbol{v_t}$. Therefore even after projecting to orthogonal subspace of $\mathrm{span}(S)$, $\boldsymbol{v_t}$ is still far from any convex combination of V'. The vertices that are not in V' all have very small norm after projecting to orthogonal subspace (at most δ_0) so we know the distance of $\boldsymbol{v_t}$ and w is at least $\gamma/2 - \delta_0 > \gamma/4$.

Now the problem becomes a two dimensional calculation. When c_t is fixed the length of $\boldsymbol{a_j}$ is strictly increasing when the distance of $\boldsymbol{v_t}$ and w decrease, so we assume the distance is $\gamma/4$. Simple calculation (using essentially just pythagorean theorem) shows

$$c_t(1 - c_t) \leq \frac{\epsilon}{\Delta - \sqrt{\Delta^2 - \gamma^2/16}}.$$

The right hand side is largest when $\Delta = 2$ (since the vectors are in unit ball) and the maximum value is $O(\epsilon/\gamma^2)$. When this value is smaller than $1/K$, we must have $1 - c_t \leq O(\epsilon/\gamma^2)$. Thus $c_t \geq 1 - O(\epsilon/\gamma^2)$ and $\delta \leq (1 - c_t) + \epsilon \leq O(\epsilon/\gamma^2)$. □

The cleanup phase tries to find the farthest point to a subset of $K - 1$ vertices, and use that point as the K-th vertex. This will improve the result because when we have $K - 1$ points close to $K - 1$ vertices, only one of the vertices can be far from their span. Therefore the farthest point must be close to the only remaining vertex. Another way of viewing this is that the algorithm is trying to greedily maximize the volume of the simplex, which makes sense because the larger the volume is, the more words/documents the final LDA model can explain.

The following lemma makes the intuitions rigorous and shows how cleanup improves the guarantee of Lemma 4.5.

Lemma 4.12. *Suppose* $|S| = K - 1$ *and each point in* S *is* $\delta = O(\epsilon/\gamma^2) < \gamma/20K$ *close to some vertex* $\boldsymbol{v_i}$*, then the farthest point* $\boldsymbol{v'_j}$ *found by the algorithm is* $1 - O(\epsilon/\gamma)$ *close to the remaining vertex.*

Proof. We still look at the original point $\boldsymbol{a_j}$ and express it as $\sum_{t=1}^{K} c_t \boldsymbol{v_t}$. Without loss of generality let $\boldsymbol{v_1}$ be the vertex that does not correspond to anything in S. By Lemma 4.4 $\boldsymbol{v_1}$ is $\gamma/2$ far from span(S). On the other hand all other vertices are at least $\gamma/20r$ close to span(S). We know the distance $\mathrm{dis}(\boldsymbol{a_j}, \mathrm{span}(S)) \geq \mathrm{dis}(\boldsymbol{v_1}, \mathrm{span}(S)) - 2\epsilon$, this cannot be true unless $c_t \geq 1 - O(\epsilon/\gamma)$. □

These lemmas directly lead to the following theorem:

Theorem 4.13. *FastAnchorWords algorithm runs in time* $\tilde{O}(V^2 + VK/\epsilon^2)$ *and outputs a subset of* $\{\boldsymbol{d_1}, ..., \boldsymbol{d_V}\}$ *of size* K *that* $O(\epsilon/\gamma)$*-covers the vertices provided that* $20K\epsilon/\gamma^2 < \gamma$.

Proof. In the first phase of the algorithm, do induction using Lemma 4.5. When $20K\epsilon/\gamma^2 < \gamma$ Lemma 4.5 shows that we find a set of points that $O(\epsilon/\gamma^2)$-covers the vertices. Now Lemma 4.6 shows after cleanup phase the points are refined to $O(\epsilon/\gamma)$-cover the vertices. □

Part II

Towards Provably Learning Deep Networks

Chapter 5

Provable Sparse Coding: Learning Overcomplete Dictionaries

Dictionary learning, or *sparse coding*, tries to learn a sparse linear code to represent the given data succinctly. Such a code represents data points (e.g. image patches) using sparse linear combinations of *dictionary elements*. This sparse representation captures the useful structure in the input data, and can be used in various tasks including denoising the input data or providing features for supervised learning algorithms. Learning a sparse linear code can also be viewed as a preliminary for learning a sparse autoencoder (a sparse nonlinear code), which is a main building block for deep learning algorithms.

In this chapter we show under mild assumptions, it is possible to learn an overcomplete dictionary (i.e. the number of dictionary elements is larger than their dimension) in polynomial time.

5.1 Background and Results

5.1.1 Sparse Coding and Dictionary Learning

Sparse coding is widely used in machine learning. However, it is first introduced in neural science, trying to explain how evolution may have produced cells in the first layer of visual cortex: Olshausen and Field [134] observed the dictionary elements learned by sparse coding share similar features as the cells in the internal image representations in the visual cortex. Inspired by the neural analog, sparse coding is often used for feature selection [10] and for building classifiers atop sparse coding routines [94]. Dictionary learning is also a basic building block in design of deep learning systems [138].

Dictionary learning is also used in image processing, where experiments suggest that hand-designed dictionaries (such as sinusoids, wavelets, ridgelets, curvelets, etc. [120]) do not do as well as dictionaries learned from data. Dictionaries learned by sparse coding algorithms are used to perform denoising [57], edge-detection [117], super-resolution [163] and compression. See [1], [58] for further applications.

5.1.2 Incoherence Assumption

Designing provably correct algorithms for dictionary learning has proved challenging. Even if the dictionary is completely known, it can be NP-hard to represent a vector $\boldsymbol{u}$ as a sparse linear combination of the columns of A (even if such a representation exists) [47]. However for many natural types of dictionaries, the problem of finding a sparse representation is computationally easy. The pioneering work of Donoho and Huo [53] (building on the uncertainty principle of Donoho and Stark [56]) presented a number of important examples of dictionaries that are *incoherent* and gave an algorithm to find a sparse representation in a known, incoherent dictionary if one exists. More precisely:

Definition 5.1 (μ-incoherent)**.** An $n \times m$ matrix A whose columns are unit vectors is said to be *μ-incoherent* if for all $i \neq j$ we have

$$A_i \cdot A_j \leq \mu/\sqrt{n}.$$

We sometimes say just *incoherent* if μ is small, like $\log n$.

A randomly chosen dictionary is incoherent with high probability (even if $m = n^C$). Donoho and Huo [53] gave many other important examples of incoherent dictionaries, such as one constructed from *spikes* and *sines*, as well as those built up from wavelets and sines, or even wavelets and ridgelets. There is a rich body of literature devoted to incoherent dictionaries (see additional references in [64]). Donoho and Huo [53] proved that given $\boldsymbol{u} = A\boldsymbol{v}$ where $\boldsymbol{v}$ has k nonzero entries, where $k \leq \sqrt{n}/2\mu$, *basis pursuit* (solvable by a linear program) recovers $\boldsymbol{v}$ exactly and it is unique. Gilbert, Muthukrishnan and Strauss [64] (and subsequently [150]) gave algorithms for recovering v even in the presence of additive noise. Tropp [148] gave a more general *exact recovery condition* (ERC) under which the sparse recovery problem for incoherent dictionaries can be algorithmically solved. All of these require $n > k^2\mu^2$. In a foundational work, Candes, Romberg and Tao [35] showed that basis pursuit solves the sparse recovery problem even for $n = O(k \log(m/k))$ if A satisfies the weaker *restricted isometry property* [34]. Also if A is a full-rank square matrix, then we can compute $\boldsymbol{v}$ from $A^{-1}\boldsymbol{u}$, trivially. But our focus here will be on incoherent and overcomplete dictionaries.

5.1.3 Main Results

A range of results are possible which trade off more assumptions with better performance. We give two illustrative ones: the first makes the most assumptions but has the best performance; the second has the weakest assumptions and somewhat worse performance.

The theorem statements will be cleaner if we use asymptotic notation: the parameters k, n, m will go to infinity and the constants denoted as"$O(1)$" are arbitrary so long as they do not grow with these parameters.

Throughout this chapter, we will use $Y^{(i)}$ to denote the i^{th} sample and $X^{(i)}$ as the vector that generated it – i.e. $Y^{(i)} = AX^{(i)}$. When we are working with a graph G we will use $\Gamma_G(u)$ to denote the set of neighbors of u in G.

First we define the class of distributions that the k-sparse vectors must be drawn from. We will be interested in distributions on k-sparse vectors in $\mathbb{R}^m$ where each coordinate is nonzero with probability $\Theta(k/m)$ (the constant in $\Theta(\cdot)$ can differ among coordinates).

Definition 5.2 (Distribution class Γ and its moments)**.** The distribution is in class Γ if (i) each nonzero X_i has expectation 0 and lies in $[-C, 1] \cup [1, C]$ where $C = O(1)$. (ii) Conditional on any subset of coordinates in X being nonzero, the values X_i are independent of each other.

The distribution has *bounded ℓ-wise moments* if the probability that X is nonzero in any subset S of ℓ coordinates is at most c^ℓ times $\prod_{i \in S} \Pr[X_i \neq 0]$ where $c = O(1)$.

Remark 5.3. (i) The bounded moments condition trivially holds for any constant ℓ if the set of nonzero locations is a random subset of size k. The *values* of these nonzero locations are allowed to be distributed very differently from one another. (ii) The requirement that nonzero X_i's be bounded away from zero in magnitude is similar in spirit to the *Spike-and-Slab Sparse Coding* (S3C) model of Goodfellow et al. [70], which also encourages nonzero latent variables to be bounded away from zero.

Theorem 5.4. *There is a polynomial time algorithm to learn a μ-incoherent dictionary A from random examples of the form $Y = AX$, where $X \in \mathbb{R}^n$ is chosen according to some distribution in Γ and A is $\mathbb{R}^{m \times n}$.*

- *If $k \leq O(\min(m^{2/5}, \frac{\sqrt{n}}{\mu \log n}))$ and the distribution has bounded 3-wise moments, then the algorithm requires $p \geq \Omega(\max(m^2/k^2 \log m, \frac{m \log m}{\epsilon^2}))$ samples and with high probability* *returns a dictionary $\hat{A}$ so that for each i, $\|A_i - \hat{A}_i\| \leq \epsilon$. The algorithm runs in time $\tilde{O}(p^2 + m^2 p)$.*

- *If $k \leq c\min(m^{(\ell-1)/(2\ell-1)}, \frac{\sqrt{n}}{\mu \log n})$ and the distribution has bounded ℓ-wise moments, then there is a polynomial time algorithm with the same guarantees and sample complexity that runs in time $\tilde{O}(p^2 + k^{\ell-1}m + m^2 p)$.*

- *Even if we are given $Y^{(i)} = AX^{(i)} + \eta_i$, where η_i's are independent spherical Gaussian noise with standard deviation $\sigma = o(\sqrt{n})$, the algorithm still works with sample complexity $p \geq \Omega(m\sigma^2 \log m/\epsilon^2)$.*

Remark 5.5. (i) The sparsity that this algorithm can tolerate – roughly $\frac{\sqrt{n}}{\mu \log n}$ or $m^{1/2-\epsilon}$ – approaches the sparsity that the best known algorithms require *even if A is known.*

Furthermore since this procedure learns a dictionary that is close to the true dictionary, what it learns must also be incoherent. Hence:

Remark 5.6. This algorithm learns a dictionary for which we can actually solve sparse recovery problems.

Now we describe the other result with fewer assumptions on X but lower performance.

Definition 5.7 (Distribution class $\mathcal{D}$)**.** A distribution is in class $\mathcal{D}$ if (i) the events $X_i \neq 0$ have *weakly bounded* second moment and third moments, in the sense that $\Pr[X_i \neq 0 \text{ and } X_j \neq 0] \leq n^\epsilon \Pr[X_i \neq 0] \Pr[X_j \neq 0]$, $\Pr[X_i, X_j, X_t \neq 0] \leq o(n^{1/4}) \Pr[X_i \neq 0] \Pr[X_j \neq 0] \Pr[X_t \neq 0]$. (ii) Each nonzero X_i is in $[-C, -1] \cup [1, C]$ where $C = O(1)$.

Remark 5.8. A major difference from class Γ is that the X_i's do not have expectation 0 and are not forbidden from taking values close to 0 (provided they do have reasonable probability of taking values away from 0). Another major difference is that the distribution of X_i *can* depend upon the values of other nonzero coordinates. The weaker moment condition allows a fair bit of correlation among the set of nonzero coordinates.

Theorem 5.9. *There is a polynomial time algorithm to learn a μ-incoherent dictionary A from random examples of the form $Y = AX$, where X is chosen according to some distribution in $\mathcal{D}$.*

- *If $k \leq c\min(m^{1/4}, \frac{n^{1/4-\epsilon/2}}{\sqrt{\mu}})$ and we are given $p \geq \Omega(\max(m^2/k^2 \log m, \frac{mn^{3/2}\log m \log n}{k^2\mu}))$ samples, then the algorithm succeeds with high probability and returns a dictionary $\hat{A}$ so that for each i, $\|A_i - \hat{A}_i\| \leq O(k\sqrt{\mu}/n^{1/4-\epsilon/2})$. The algorithm runs in time $\tilde{O}(p^2 + m^2 p)$.*

The algorithm is also noise-tolerant as the one in Theorem 5.4.

Theorem 5.9 is proved similarly to Theorem 5.4, the proof can be found in [13].

Remark 5.10. It is also possible to relax the condition that each nonzero X_i is in $[-C, 1] \cup [1, C]$. Instead we require X_i has magnitude at most $O(1)$, and has a weak *anti-concentration* property: for every $\delta > 0$ it has probability at least $c_\delta > 0$ of exceeding δ in magnitude.

5.1.4 Previous Work

There has also been significant prior work on dictionary learning and sparse coding. Lewicki and Sejnowski [111] provided the first approach, and subsequently Engan, Aase and Husoy [59] introduce the *method of optimal directions* (MOD) and Aharon, Elad and Bruckstein [2] introduced *K-SVD*, both of which have had considerable success in practice. For completeness, we will describe these latter two approaches. Suppose the algorithm is given a matrix Y, generated as AX where A is the unknown dictionary and X is an unknown matrix with iid columns.

Method of Optimal Direction [59] : Start with an initial guess A, and then alternately update either A or X:

- Given A, compute a sparse X so that $AX \approx Y$ (using e.g. matching pursuit [119] or basis pursuit [39])

- Given X, compute the A that minimizes $\|AX - Y\|_F$

This algorithm converges to a local optimum, because in each step the error $\|AX - Y\|_F$ will only decrease.

K-SVD [2] Start with an initial guess for A. Then repeat the following procedure:

- Given A, compute a sparse X so that $AX \approx Y$ (again, using a pursuit method)
- Group all data points $Y^{(1)}$ where the corresponding X vector has a non-zero at index i. Subtract off components in the other directions

$$Y^{(1)} - \sum_{j \neq i} A_j X_j^{(1)}$$

- Compute the first singular vector v_1 of the residual matrix, and update the column A_i to v_1

However, these algorithms do not come with provable guarantees. After all, since the first step in the algorithm uses some arbitrary guess for A there may not be *any* sparse representation X so that $AX \approx Y$. Furthermore, even if the algorithm learns some dictionary there is no guarantee that it is incoherent, and consequently no guarantee that one can use it to solve sparse recovery.

Instead, our goal here is to give an algorithm for dictionary learning that has provable guarantees. Assuming the data is generated by an incoherent dictionary, we recover it to within arbitrary accuracy. The most closely related work is an elegant paper of Spielman, Wang and Wright [144], giving an algorithm that provably recovers A to arbitrary accuracy if it is a full column rank matrix, and X satisfies reasonable assumptions. However, requiring A to be full column rank precludes most interesting settings where the dictionary is redundant and hence cannot have full column rank (see e.g. [53], [58], [120]).

5.1.5 Our approach

The main idea is to recover the support of the vector X by a *graph recovery* algorithm. If given many samples, we can find all the samples for which $X_i \neq 0$ then we can recover the column A_i from a singular value decomposition. (Alternatively, we can instead find all the samples for which $X_i > 0$ and then the samples will be biased in the direction of A_i). This is exactly the approach taken in K-SVD [59]. The difference is that in K-SVD, the support of samples are determined after running matching pursuit on an intermediate guess for the dictionary. If this guess is not accurate, then the support could be far from correct. Our main observation is that for incoherent dictionaries, we can recover the support without knowing the dictionary! We build a graph – called the *connection graph*. We give a simple combinatorial algorithm for provably recovering the support graph of X from this graph, and from this we can immediately recover the true dictionary. In order to prove correctness of our combinatorial algorithm, we rely on tools from discrete geometry, namely the *piercing number* [121], [5].

Related Works The graph recovery problem we have is related to the *graph square root* problem ([106, 131]), which tries to reconstruct a graph given all pairs of distance at most 2 in the graph. Our setting is slightly different, and our algorithm only solves the problem in average case. There is also a connection between our graph recovery problem and those studied in community detection literature, where recent papers [14], [17] give provable algorithms for finding all of the overlapping communities in a graph. However these algorithms applied to our problem would run in time quasi-polynomial in the sparsity k.

We remark that algorithms for independent component analysis [42] can also provably learn an unknown dictionary A from random examples. Suppose we are given random samples of the form AX where the components of X are *exactly* independent. Frieze, Jerrum and Kannan [63] gave an algorithm with provable guarantees that recovers the dictionary

A up to arbitrary accuracy, provided that the random variables in the vector X are non-Gaussian (since in that case A is only determined up to rotations anyways). Subsequent work on this problem considers the over-determined case and gives provable algorithms even when A is $n \times m$ with m larger than n [49], [72]. However, these algorithms are very brittle to the assumption that the entries in X be independent; if this condition is even slightly violated, the algorithms can fail. The weakest conditions that we are aware of require that X be 4-wise independent, and typically even higher orders of independence are required for the overcomplete case [49], [72]. Hence these algorithms are not well-suited for dictionary learning where this condition seems too restrictive.

5.2 The Connection Graph

Let A be an unknown $n \times m$ dictionary whose columns are incoherent – i.e. each column is a unit vector and for all $i \neq j$ we have

$$A_i \cdot A_j \leq \mu/\sqrt{n}$$

Our goal is to recover the dictionary A given sparse linear combinations of its columns. In particular, suppose that there is a distribution Γ on k-sparse columns X of length m. We will assume the following generative model:

- each X has at most k non-zeroes and the probability that $X_i \neq 0$ is $\Theta(k/m)$ and furthermore $X_i \neq 0$ is non-positively correlated with other coordinates being non-zero
- if a coordinate is non-zero, its magnitude is in the range $[1, C]$ and its expectation is zero.

In fact, our analysis will immediately extend to even more general models (see the full version in [13]) but for simplicity we encourage the reader to restrict the distribution Γ so that the support of X is uniformly random over all k-tuples and that if a coordinate is non-zero it is a Rademacher random variable (equally likely to be $+1$ or -1) since this will allow us to not have to keep track of as many extraneous constants through the various steps of our analysis.

Recall our strategy is to find the support of X. Based on the support, we form an overlapping clustering where we gather together all the samples Y for which $X_i \neq 0$. We suggest two approaches for how to use this overlapping clustering to determine the dictionary in Section 5.4.1 and Section 5.4.2 respectively. The first approach is to further determine which set of samples have $X_i > 0$. This set of samples will be biased in exactly the direction of A_i and hence we can recover this column by taking an average. Alternatively, we can compute the direction of maximum variance. This latter approach recovers (roughly) the direction A_i and yields insight into why approaches like K-SVD [59] are effective in practice.

But how can we compute the correct support? As a starting point: if there are two samples $Y^{(1)} = AX^{(1)}$ and $Y^{(2)} = AX^{(2)}$ then we can determine if the support of $X^{(1)}$ and $X^{(2)}$ intersect, but with false negatives:

Lemma 5.11. *If $k^2\mu < \sqrt{n}/2$ then $|Y^{(1)} \cdot Y^{(2)}| > 1/2$ implies that he support of $X^{(1)}$ and $X^{(2)}$ intersect*

Proof. Suppose that the support of $X^{(1)}$ and $X^{(2)}$ are disjoint. Then the following upper bound holds:

$$|Y^{(1)} \cdot Y^{(2)}\rangle| \leq \sum_{i \neq j} |A_i \cdot A_j X_i^{(1)} X_j^{(2)}| \leq k^2\mu/\sqrt{n} < 1/2$$

and this implies the lemma. □

This would be a weak bound in our setting, since we would require that $k = O(n^{1/4}/\sqrt{\mu})$. Whereas if the dictionary is known, then classic results in compressed sensing show (algorithmically) how to recover a vector that is a linear combination of at most $\sqrt{n}/2\mu$ columns of A [53]. In fact we can impose weaker conditions on k, if we use the randomness of the values in X. We will appeal to the classic Hanson-Wright inequality:

Theorem 5.12 (Hanson-Wright)**.** *[76] Let X be a vector of independent, sub-Gaussian random variables with mean zero and variance one. Let M be a symmetric matrix. Then*

$$Pr[|X^\top MX - tr(M)| > t] \leq 2exp\{-c\min(t^2/\|M\|_F^2, t/\|M\|)\}$$

Lemma 5.13. *Suppose $k\mu < \frac{\sqrt{n}}{C'\log n}$ for large enough constant C'. Then if the support of $X^{(1)}$ and $X^{(2)}$ are disjoint, with high probability $|Y^{(1)} \cdot Y^{(2)}| < 1/2$*

Proof. Let N be the $k \times k$ submatrix resulting from restricting $A^\top A$ to the locations where $X^{(1)}$ and $X^{(2)}$ are non-zero. Set M to be a $2k \times 2k$ matrix where the $k \times k$ submatrices in the top-left and bottom-right are zero, and the $k \times k$ submatrices in the bottom-left and top-right are $(1/2)N$ and $(1/2)N^\top$ respectively. Here we think of the vector X as being a length $2k$ vector whose first k entries are the non-zero entries in $X^{(1)}$ and whose last k entries are the non-zero entries in $X^{(2)}$. And by construction, we have that

$$Y^{(1)} \cdot Y^{(2)} = X^\top MX$$

We can now appeal to the Hanson-Wright inequality (above). Note that since the support of $X^{(1)}$ and $X^{(2)}$ do not intersect, the entries in M are each at most $\mu/\sqrt{n}$ and so the Frobenius norm of M is at most $\frac{\mu k}{\sqrt{2n}}$. This is also an upper-bound on the spectral norm of M. We can set $t = 1/2$, and for $k\mu < \sqrt{n}/C' \log n$ both terms in the minimum are $\Omega(\log n)$ and this implies the lemma. □

We can build a graph of which pairs intersect, by connecting a pair of samples $Y^{(1)}$ and $Y^{(2)}$ if $|Y^{(1)} \cdot Y^{(2)}| > 1/2$.

Definition 5.14. Given p samples $Y^{(1)}, Y^{(2)}, ..., Y^{(p)}$, build a *connection graph* on p nodes where i and j are connected by an edge if and only if $Y^{(1)} \cdot Y^{(2)} > 1/2$.

This graph will "miss" some edges, since if a pair $X^{(1)}$ and $X^{(2)}$ have intersecting support we do not necessarily meet the above condition, but (with high probability) this graph will not have any false positives:

Corollary 5.15. *With high probability, each edge (i, j) that is present in the connection graph corresponds to a pair where the support of $X^{(i)}$ and $X^{(j)}$ intersect.*

It is important to note that a sample $Y^{(1)}$ will intersect many other samples (say, $Y^{(2)}$ and $Y^{(3)}$), each of which corresponds to a coordinate i (reps. i') where the support of $X^{(2)}$ (resp. $X^{(3)}$) intersect with the support of $X^{(1)}$. But the challenge is that we do not know if they intersect $X^{(1)}$ in the same coordinate or not. We will use the other samples as a way to "cross-check" and determine if $X^{(1)}, X^{(2)}$ and $X^{(3)}$ have a common intersection.

Remark 5.16. Even if we are given samples $Y = AX + \eta$ where η is additive noise, under natural conditions on the correlation of η with our samples (e.g. if it is spherical Gaussian noise whose expected norm is at most $o(\sqrt{n})$) this will not affect the construction of the connection graph, and hence will not affect the proof of correctness of our algorithms.

5.3 Graph Recovery

Our goal in this section is to determine which samples Y have $X_i \neq 0$ just from the connection graph. To do this, we reduce this problem to a *graph recovery* problem.

Definition 5.17 (Graph Recovery)**.** There is an unknown bipartite graph $G_0(U, V, E_0)$. Given a graph $G(V, E)$, constructed randomly based on G_0 satisfy the following properties:

1. For two vertices $(p, q) \in V$, if they have no common neighbor in G_0, then $(p, q) \notin E$.

2. For vertices $p, q_1, q_2, ..., q_l$, if there is a vertex $u \in U$ where $\{(u,p), (u,q_1), ..., (u,q_l)\} \subset E_0$, then $\Pr[\text{all edges } (p,q_i) \in E] \geq 2^{-l}$.

The goal is to recover the graph G_0.

Consider G_0 as the support graph of X, where U is the set of coordinates, V is the set of samples, and $(i,p) \in E_0$ iff $X_p^{(i)} \neq 0$. The connection graph we constructed in Definition 5.14 satisfies the requirement for graph recovery problem. In order to prove this we just need the following lemma.

Lemma 5.18. *Suppose the support of X intersects the support of each of $X^{(1)}, X^{(2)}, ..., X^{(\ell)}$. Then*

$$Pr[\textit{ for all } j = 1, 2, ..., \ell, |Y \cdot Y^{(j)}| > 1/2] \geq 2^{-\ell}$$

Proof. We have that with high probability that $\text{round}(Y \cdot Y^{(j)}) = X \cdot X^{(j)}$ for all j using Lemma 5.13, and hence we need to prove that with probability at least $2^{-\ell}$, each pair $X \cdot X^{(j)}$ is non-zero. We can prove this by induction, by considering the $X^{(j)}$'s in an order such that no set $S_j = \text{supp}(X) \cap \text{supp}(X^{(j)})$ is entirely contained in a earlier set S_i (for $i < j$). Then conditioned on all previous inner-products $X \cdot X^{(i)}$ being non-zero, the probability that $X^{(1)} \cdot X^{(j)}$ is non-zero is at least 1/2. □

Now given two samples $Y^{(1)}$ and $Y^{(2)}$ so that the support of $X^{(1)}$ and $X^{(2)}$ intersect uniquely at index i, we will prove that all the samples Y for which $X_i \neq 0$ can be recovered because the expected number of common neighbors between $Y^{(1)}, Y^{(2)}$ and Y will be much larger than in any other case (in particular if $X_i = 0$ instead).

Claim. *Suppose* $\text{supp}(X^{(1)}) \cap \text{supp}(X^{(2)}) \cap \text{supp}(X^{(3)}) \neq \emptyset$*, then*

$$Pr_Y[\textit{ for all } j = 1, 2, 3, |Y \cdot Y^{(j)}| > 1/2] \geq \frac{k}{8m}$$

Proof. Let $i \in supp(X^{(1)}) \cap supp(X^{(2)}) \cap supp(X^{(3)})$. Then the probability that $supp(X) \ni i$ is exactly k/m, and hence using the previous lemma this implies the claim. $\square$

Intuitively, if $\text{supp}(X^{(1)}) \cap \text{supp}(X^{(2)}) \cap \text{supp}(X^{(3)}) = \emptyset$ then the expected number of common neighbors of $Y^{(1)}, Y^{(2)}$ and $Y^{(3)}$ should be smaller. But in principle we should be concerned that the support of X could still intersect the support of each of $X^{(1)}, X^{(2)}$ and $X^{(3)}$. Let $a = |\text{supp}(X^{(1)}) \cap \text{supp}(X^{(2)})|$, $b = |\text{supp}(X^{(1)}) \cap \text{supp}(X^{(3)})|$ and $c = |\text{supp}(X^{(2)}) \cap \text{supp}(X^{(3)})|$. Then:

Lemma 5.19. *Suppose that* $\text{supp}(X^{(1)}) \cap \text{supp}(X^{(2)}) \cap \text{supp}(X^{(3)}) = \emptyset$*. Then the probability that the support of* X *intersects the support of each of* $X^{(1)}, X^{(2)}$ *and* $X^{(3)}$ *is at most*

$$\frac{k^6}{m^3} + \frac{3k^3(a+b+c)}{m^2}$$

Proof. We can break up the event whose probability we would like to bound into two (not necessarily disjoint) events: (1) the probability that X intersects each of $X^{(1)}, X^{(2)}$ and $X^{(3)}$ disjointly (i.e. it contains a point $i \in \text{supp}(X^{(1)})$ but $i \notin \text{supp}(X^{(2)}), \text{supp}(X^{(3)})$, and similarly for the other sets). (2) the probability that X contains a point in the common intersection of two of the sets, and one point from the remaining set. Clearly if the support of X intersects the support of each of $X^{(1)}, X^{(2)}$ and $X^{(3)}$ then at least one of these two events must occur.

The probability of the first event is at most the probability that the support of X contains at least one element from each of three disjoint sets of size at most k. The probability that the support of X contains an element of just one such set is at most the expected intersection which is $\frac{k^2}{m}$, and since the expected intersection of X with each of these sets are non-positively correlated (because they are disjoint) we have that the probability of the first event can be bounded by $\frac{k^6}{m^3}$.

Similarly, for the second event: consider the probability that the support of X contains an element in $\mathrm{supp}(X^{(1)}) \cap \mathrm{supp}(X^{(2)})$. Since $\mathrm{supp}(X^{(1)}) \cap \mathrm{supp}(X^{(2)}) \cap \mathrm{supp}(X^{(3)}) = \emptyset$, the support of X must also contain an element in $\mathrm{supp}(X^{(3)})$ too. The expected intersection of X and $\mathrm{supp}(X^{(1)}) \cap \mathrm{supp}(X^{(2)})$ is $\frac{ka}{m}$ and the expected intersection of X and $X^{(3)}$ is $\frac{k^2}{m}$, and again the expectations are non-positively correlated since the two sets $\mathrm{supp}(X^{(1)}) \cap \mathrm{supp}(X^{(2)})$ and $\mathrm{supp}(X^{(3)})$ are disjoint by assumption. Repeating this argument for the other pairs completes the proof of the lemma. □

Note that the probability that two sets of size k intersect in at least Q elements is at most $(\frac{k}{m})^Q$ by the same non-positive correlation argument. Hence we can assume that with high probability there is *no* pair of sets that intersect in more than Q locations.

And comparing the lemma and the claim above, we find that if $k \leq cm^{2/5}$ then the expected number of common neighbors is much larger if $\mathrm{supp}(X^{(1)}) \cap \mathrm{supp}(X^{(2)}) \cap \mathrm{supp}(X^{(3)}) \neq \emptyset$ than if the intersection is empty. Under this condition, if we take $p = O(m^2/k^2 \log n)$ samples each triple with a common intersection will have at least T common neighbors, and each triple whose intersection is empty will have less than $T/2$ common neighbors.

Hence we can search for a triple with a common intersection as follows: We can find a pair that intersects, since (with high probability) for any pair $|Y^{(1)} \cdot Y^{(2)}| > 1/2$ the support of $X^{(1)}$ and $X^{(2)}$ intersect. We can try neighbors of $Y^{(1)}$ at random, and we can use the number of common neighbors as a test to verify whether all three sets have a common intersection.

Definition 5.20. We will call a triple of samples $Y^{(1)}, Y^{(2)}$ and $Y^{(3)}$ an *identifying triple* for coordinate i if the support of $X^{(1)}, X^{(2)}$ and $X^{(3)}$ intersect and furthermore the support of $X^{(1)}$ and $X^{(2)}$ is exactly $\{i\}$.

Theorem 5.21. *Algorithm 5.1(RecoverGraph) output sets that each corresponds to some i and contains all $Y^{(j)}$ for which $i \in \mathrm{supp}(X^{(j)})$. The algorithm runs in time $\widetilde{O}(p^2 n)$ and succeeds with high probability if $k \leq c \min(m^{2/5}, \frac{\sqrt{n}}{\mu \log n})$ and if $p = \Omega(m^2/k^2 \log m)$*

Algorithm 5.1. RecoverGraph

```
input  $p$ samples $Y^{(1)}, Y^{(2)}, ..., Y^{(p)}$
output  Sets that are equal to the support of rows of $X$
  Compute a graph $G$ on $p$ nodes where there is an edge between $i$ and $j$ iff $|Y^{(1)} \cdot Y^{(2)}| > 1/2$

  Initialize list of triples to be empty
  Set $T = \frac{pk}{10m}$
  for $i = 1$ TO $\Omega(mk \log^2 m)$ do
    Choose a random edge $(u, v)$ in $G$ and a random neighbor $w$ of $u$
    if $|\Gamma_G(u) \cap \Gamma_G(v) \cap \Gamma_G(w)| \geq T$ then
              Add $(u, v, w)$ to the list of triples
    end if
  end for
  For each node $x$, add $x$ to each set $S_{u,v,w}$ where $(u, v, w)$ is a triple and $|\Gamma_G(u) \cap \Gamma_G(v) \cap \Gamma_G(x)| \geq T$
  Delete any set $S_{u,v,w}$ that contains another set $S_{a,b,c}$
  Output the remaining sets $S_{u,v,w} \cup \{u, v, w\}$
```

Proof. We can use Lemma 5.13 to conclude that each edge in G corresponds to a pair whose support intersects. We can appeal to Lemma 5.19 and Claim 5.3 to conclude that for $p = \Omega(m^2/k^2 \log m)$, with high probability each triple with a common intersection has at least T common neighbors, and each triple without a common intersection has at most $T/2$ common neighbors.

In fact, for a random edge (u, v), the probability that the common intersection of u and v (of the supports of their X's) is exactly $\{i\}$ is $\Omega(1/m)$ because we know that their X's do intersect, and that intersection has a constant probability of being size one and it is uniformly distributed over m possible locations. Furthermore the probability that a randomly chosen neighbor w of u contains i (in the support of its X) is at least k/m, and hence appealing to a coupon collector argument we conclude that if the inner loop is run at least $\Omega(km \log^2 m)$ times then the algorithm finds an identifying triple (u, v, w) for each column A_i with high probability.

Note that we may have triples that are not an identifying triple for some coordinate i. However, any other triple (u, v, w) found by the algorithm must have a common intersection.

Consider for example a triple (u, v, w) where u and v have a common intersection $\{i, j\}$. Then we know that there is some other triple (a, b, c) which is an identifying triple for i and hence $S_{a,b,c} \subset S_{u,v,w}$. (In fact this containment is strict, since $S_{u,v,w}$ will also contain a set corresponding to an identifying triple for j too). Then the second-to-last step in the algorithm will necessarily delete all such triples $S_{u,v,w}$. What is the running time of this algorithm? We need $O(p^2 n)$ time to build the connection graph, and the loop takes $\widetilde{O}(mpk)$ time. Finally, the deletion step requires time $\widetilde{O}(m^2 p)$ since there will be $\widetilde{O}(m^2)$ triples found in the previous step. This concludes the proof of correctness of the algorithm, and its running time analysis. □

5.4 Recovering the Dictionary

5.4.1 Finding the Relative Signs

Here we show how to recover the columns of A once we have learned the support of X. The key observation is that if the support of $X_i^{(1)}$ and $X_i^{(2)}$ uniquely intersect in index i then the sign of $Y^{(1)} \cdot Y^{(2)}$ is equal to the sign of $X_i^{(1)} X_i^{(2)}$. And if there are enough such pairs $X_i^{(1)}$ and $X_i^{(2)}$, then we can correctly determine the relative sign of every pair $X_i^{(1)}$ and $X_i^{(2)}$. We formalize this idea in the following lemma:

Lemma 5.22. *In Algorithm 5.2, S_i is either $\{u : X_i^{(u)} > 0\}$ or $\{u : X_i^{(u)} < 0\}$.*

Proof. It suffices to prove the lemma at the start of Step 12, since this step only takes the complement of S_i with respect to $\mathcal{C}_i$. Appealing to Lemma 5.13 we conclude that if the support of $X^{(u)}$ and $X^{(v)}$ uniquely intersect in coordinate i then the sign of $Y^{(u)} \cdot Y^{(v)}$ is equal to the sign of $X_i^{(u)} X_i^{(v)}$. Hence when Algorithm 5.2 adds an element to S_i it must have the same sign as the i^{th} component of $X^{(u_i)}$. What remains is to prove that each node $v \in \mathcal{C}_i$ is correctly labeled. We will do this by showing that for any such vertex, there is a

Algorithm 5.2. OverlappingAverage

```
input $p$ samples $Y^{(1)}, Y^{(2)}, ...Y^{(p)}$
output estimation $\hat{A}$ of $A$
  Run RecoverGraph (or RecoverGraphHighOrder) on the $p$ samples
  Let $\mathcal{C}_1, \mathcal{C}_2, ...\mathcal{C}_m$ be the $m$ returned sets
  for each $\mathcal{C}_i$ do
    for each pair $u, v \in \mathcal{C}_i$ such that $X^{(u)}$ and $X^{(v)}$ have a unique intersection do
      Label the pair $+1$ if $Y^{(u)} \cdot Y^{(v)} > 0$ and otherwise label it $-1$.
    end for
    Choose an arbitrary $u_i \in \mathcal{C}_i$, and set $S_i = \{u_i\}$
    for each $v \in \mathcal{C}_i$ do
      if the pair $u_i, v$ is labeled $+1$, or there is $w \in \mathcal{C}_i$ where the pairs $u_i, w$ and $v, w$ have
      the same label then
        add $v$ to $S_i$.
      end if
    end for
    if $|S_i| \leq |\mathcal{C}_i|/2$ then
      set $S_i = \mathcal{C}_i \backslash S_i$.
    end if
    Let $\hat{A}_i = \sum_{v \in S_i} X^{(v)} / \| \sum_{v \in S_i} X^{(v)} \|$
  end for
  return $\hat{A}$, where each column is $\hat{A}_i$ for some $i$
```

length two path of labeled pairs that connects u_i to v, and this is true because the number of labeled pairs is large. We need the following simple claim:

Claim. *If $p > m^2 \log m / k^2$ then with high probability any two rows of X share at most $2pk^2/m^2$ coordinates in common.*

This follows since the probability that a node is contained in any fixed pair of sets is at most k^2/m^2. Then for any node $u \in \mathcal{C}_i$, we would like to lower bound the number of labeled pairs it has in $\mathcal{C}_i$. Since u is in at most $k-1$ other sets $\mathcal{C}_{i_1}, ..., \mathcal{C}_{i_{k-1}}$, the number of pairs u, v where $v \in \mathcal{C}_i$ that are not labeled for $\mathcal{C}_i$ is at most

$$\sum_{t=1}^{k-1} |\mathcal{C}_{i_t} \cap \mathcal{C}_i| \leq k \cdot 2pk^2/m^2 \ll pk/3m = |\mathcal{C}_i|/3$$

Therefore for a fixed node u for at least a $2/3$ fraction of the other nodes $w \in \mathcal{C}_i$ the pair u, w is labeled. Hence we conclude that for each pair of nodes $u_i, v \in \mathcal{C}_i$ the number of w for which both u_i, w and w, v are labeled is at least $|\mathcal{C}_i|/3 > 0$ and so for every v, there is a labeled path of length two connecting u_i to v. □

Using this lemma, we are ready to prove Algorithm 5.2 correctly learns all columns of A.

Theorem 5.23. *Algorithm 5.2(OverlappingAverage) outputs a dictionary $\hat{A}$ so that for each i, $\|A_i - \hat{A}_i\| \leq \epsilon$ with high probability if $k \leq c \min(m^{2/5}, \frac{\sqrt{n}}{\mu \log n})$ and if*

$$p = \Omega \max(m^2/k^2 \log m, m \log m/\epsilon^2)$$

Furthermore the algorithm runs in time $O(p^2k^2/m)$.

Proof. We can invoke Lemma 5.22 and conclude that S_i is either $\{u : X_i^{(u)} > 0\}$ or $\{u : X_i^{(u)} < 0\}$, whichever set is larger. Let us suppose that it is the former. Then each $Y^{(u)}$ in S_i is an independent sample from the distribution conditioned on $X_i > 0$, which we call Γ_i^+. We have that $\mathbb{E}_{\Gamma_i^+}[AX] = cA_i$ where c is a constant in $[1, C]$ because $\mathbb{E}_{\Gamma_i^+}[X_j] = 0$ for all $j \neq i$.

Let us compute the variance:

$$\mathbb{E}_{\Gamma_i^+}[\|AX - \mathbb{E}_{\Gamma_i^+}AX\|^2] \leq \mathbb{E}_{\Gamma_i^+}X_i^2 + \sum_{j \neq i} \mathbb{E}_{\Gamma_i^+}[X_j^2] \leq C^2 + \sum_{j \neq i} C^2 k/m \leq C^2(k+1),$$

Note that there are no cross-terms because the signs of each X_j are independent. Furthermore we can bound the norm of each vector $Y^{(u)}$ via incoherence. We can conclude that if $|S_i| > C^2 k \log m/\epsilon^2$, then with high probability $\|\hat{A}_i - A_i\| \leq \epsilon$ using vector Bernstein's inequality ([74], Theorem 12). This latter condition holds because we set S_i to itself or its complement based on which one is larger. □

5.4.2 An Approach via SVD

Here we give an alternative algorithm for recovering the dictionary based instead on SVD. The advantage is that approaches such as K-SVD which are quite popular in practice also rely on finding directions of maximum variance, so the analysis we provide here yields insights into why these approaches work. However, we note that the crucial difference is that we rely on finding the correct support in the first step of our dictionary learning algorithms, whereas K-SVD and approaches like it attempt to alternate between updating the dictionary and fixing the estimation of the support.

Let us fix some notation: Let Γ_i be the distribution conditioned on $X_i \neq 0$. Then once we have found the support of X, each set from Algorithm 5.1 RecoverGraph is a set of random samples from Γ_i. Also let $\alpha = |\boldsymbol{u} \cdot A_i|$.

Definition 5.24. Let $R_i^2 = 1 + \sum_{j \neq i} (A_i \cdot A_j)^2 E_{\Gamma_i}[X_j^2]$.

Note that R_i^2 is the projected variance of Γ_i on the direction $\boldsymbol{u} = A_i$. Our goal is to show that for any $\boldsymbol{u} \neq A_i$ (i.e. $\alpha \neq 1$), the variance is strictly smaller.

Lemma 5.25. *The projected variance of* Γ_i *on* $\boldsymbol{u}$ *is at most*

$$\alpha^2 R_i^2 + \alpha\sqrt{(1-\alpha^2)}\frac{2\mu k}{\sqrt{n}} + (1-\alpha^2)(\frac{k}{m} + \frac{\mu k}{\sqrt{n}})$$

Proof. Let $\boldsymbol{u}^{||}$ and $\boldsymbol{u}^{\perp}$ be the components of $\boldsymbol{u}$ in the direction of A_i and perpendicular to A_i. Then we want bound $E_{\Gamma_i}[(\boldsymbol{u} \cdot Y)^2]$ where Y is sampled from Γ_i. Since the signs of each X_j are independent, we can write

$$E_{\Gamma_i}[(\boldsymbol{u} \cdot Y)^2] = \sum_j E_{\Gamma_i}[(\boldsymbol{u} \cdot A_j X_j)^2] = \sum_j E_{\Gamma_i}[((\boldsymbol{u}^{||} + \boldsymbol{u}^{\perp}) \cdot A_j X_j)^2]$$

Since $\alpha = \|\boldsymbol{u}^{||}\|$ we have:

$$E_{\Gamma_i}[(\boldsymbol{u} \cdot Y)^2] = \alpha^2 R_i^2 + E_{\Gamma_i}[\sum_{j \neq i} (2(\boldsymbol{u}^{||} \cdot A_j)(\boldsymbol{u}^{\perp} \cdot A_j) + (\boldsymbol{u}^{\perp} \cdot A_j)^2) X_j^2]$$

Also $E_{\Gamma_i}[X_j^2] = (k-1)/(m-1)$. Let v be the unit vector in the direction $\boldsymbol{u}^{\perp}$. We can write

$$E_{\Gamma_i}[\sum_{j \neq i} (\boldsymbol{u}^{\perp} \cdot A_j)^2 X_j^2] = (1 - \alpha^2)(\frac{k-1}{m-1}) v^{\top} A_{-i} A_{-i}^{\top} v$$

where A_{-i} denotes the dictionary A with the i^{th} column removed. The maximum over v of $v^{\top} A_{-i} A_{-i}^{\top} v$ is just the largest singular value of $A_{-i} A_{-i}^{\top}$ which is the same as the largest singular value of $A_{-i}^{\top} A_{-i}$ which by the Greshgorin Disk Theorem (see e.g. [85]) is at most $1 + \frac{\mu}{\sqrt{n}} m$. And hence we can bound

$$E_{\Gamma_i}[\sum_{j \neq i} (\boldsymbol{u}^{\perp} \cdot A_j)^2 X_j^2] \leq (1 - \alpha^2)(\frac{k}{m} + \frac{\mu k}{\sqrt{n}})$$

Also since $|\boldsymbol{u}^{||} \cdot A_j| = \alpha |A_i \cdot A_j| \leq \alpha \mu / \sqrt{n}$ we obtain:

$$E[\sum_{j \neq i} 2(\boldsymbol{u}^{||} \cdot A_j)(\boldsymbol{u}^{\perp} \cdot A_j) X_j^2] \leq \alpha \sqrt{(1 - \alpha^2)} \frac{2\mu k}{\sqrt{n}}$$

and this concludes the proof of the lemma. □

Definition 5.26. Let $\zeta = \max\{\frac{\mu k}{\sqrt{n}}, \sqrt{\frac{k}{m}}\}$, so the expression in Lemma 5.25 can be be an upper bounded by $\alpha^2 R_i^2 + 2\alpha\sqrt{1 - \alpha^2} \cdot \zeta + (1 - \alpha^2)\zeta^2$.

We will show that an approach based on SVD recovers the true dictionary up to additive accuracy $\pm\zeta$. Note that here ζ is a parameter that converges to zero as the size of the problem increases, but is not a function of the number of samples. So unlike the algorithm in the previous subsection, we cannot make the error in our algorithm arbitrarily small by increasing the number of samples, but this algorithm has the advantage that it succeeds even when $\mathbb{E}[X_i] \neq 0$.

Corollary 5.27. *The maximum singular value of* Γ_i *is at least* R_i *and the direction* $\boldsymbol{u}$ *satisfies* $\|\boldsymbol{u} - A_i\| \leq O(\zeta)$. *Furthermore the second largest singular value is bounded by* $O(R_i^2\zeta^2)$.

Proof. The bound in Lemma 5.25 is only an upper bound, however the direction $\alpha = 1$ has variance $R_i^2 > 1$ and hence the direction of maximum variance must correspond to $\alpha \in [1 - O(\zeta^2), 1]$. Then we can appeal to the variational characterization of singular values (see [85]) that

$$\sigma_2(\Sigma_i) = \max_{\boldsymbol{u} \perp A_i} \frac{\boldsymbol{u}^\top \Sigma_i \boldsymbol{u}}{\boldsymbol{u}^\top \boldsymbol{u}}$$

Then condition that $\alpha \in [-O(\zeta), O(\zeta)]$ for the second singular value implies the second part of the corollary. □

Since we have a lower bound on the separation between the first and second singular values of Σ_i, we can apply Wedin's Theorem (see Appendix A.1) and show that we can recover A_i approximately even in the presence of noise.

Hence even if we do not have access to Σ_i but rather an approximation to it $\hat{\Sigma}_i$ (e.g. an empirical covariance matrix computed from our samples), we can use the above perturbation bound to show that we can still recover a direction that is close to A_i – and in fact converges to A_i as we take more and more samples.

Algorithm 5.3. OverlappingSVD

```
Run RecoverGraph (or RecoverGraphHighOrder) on the $p$ samples
Let $\mathcal{C}_1, \mathcal{C}_2, ...\mathcal{C}_m$ be the $m$ returned sets
Compute $\hat{\Sigma}_i = \frac{1}{|\mathcal{C}_i|} \sum_{Y \in \mathcal{C}_i} YY^\top$
Compute the first singular value $\hat{A}_i$ of $\hat{\Sigma}_i$
**return** $\hat{A}$, where each column is $\hat{A}_i$ for some $i$
```

Theorem 5.28. *Algorithm 5.3 (OverlappingSVD) outputs a dictionary $\hat{A}$ so that for each i, $\|A_i - \hat{A}_i\| \leq \zeta$ with high probability if $k \leq c\min(m^{2/5}, \frac{\sqrt{n}}{\mu \log n})$ and if*

$$p \geq \max(m^2/k^2 \log m, \frac{mn \log m \log n}{\zeta^2})$$

Proof. Appealing to Theorem 5.21, we have that with high probability the call to Algorithm 5.1 (RecoverGraph) returns the correct support. Then given $\frac{n \log n}{\zeta^2}$ samples from the distribution Γ_i the classic result of Rudelson implies that the computed empirical covariance matrix $\hat{\Sigma}_i$ is close in spectral norm to the true co-variance matrix [140]. This, combined with the separation of the first and second singular values established in Corollary 5.27 and Wedin's Theorem A.2 imply that we recover each column of A up to an additive accuracy of ϵ and this implies the theorem. Note that since we only need to compute the first singular vector, this can be done via power iteration [68] and hence the bottleneck in the running time is the call to Algorithm 5.1(RecoverGraph). □

5.5 A Higher-Order Algorithm

Here we extend the Algorithm 5.1 (RecoverGraph) presented in Section 5.3 to succeed even when $k \leq c\min(m^{1/2-\epsilon}, \sqrt{n}/\mu \log n)$. We can then use the Algorithm 5.2 in conjunction with this new clustering algorithm and obtain the second part of Theorem 6.1.

The premise of Algorithm 5.1(RecoverGraph) is that we can distinguish whether or not a triple of sets $X^{(1)}, X^{(2)}, X^{(3)}$ has a common intersection based on their number of common neighbors in the connection graph. However for $k = \omega(m^{2/5})$ this is no longer true! But we will instead consider higher-order groups of sets. In particular, for any $\epsilon > 0$ there is an ℓ so that we can distinguish whether an ℓ-tuple of sets $X^{(1)}, X^{(2)}, ..., X^{(\ell)}$ has a common intersection or not based on their number of common neighbors even for $k = \Omega(m^{1/2-\epsilon})$.

The main technical challenge is in showing that if the sets $X^{(1)}, X^{(2)}, ..., X^{(\ell)}$ do not have a common intersection, that we can upper bound the probability that a random set X intersects each of them. To accomplish this, we will need to bound the number of ways of piercing ℓ sets $X^{(1)}, X^{(2)}, ..., X^{(\ell)}$ that have bounded pairwise intersections by at most s points (see Lemma 5.31), and from this an analysis of Algorithm 5.4(RecoverGraphHighOrder) will be immediate.

Algorithm 5.4. RecoverGraphHighOrder

```
Compute a graph $G$ on $p$ nodes where there is an edge between $i$ and $j$ iff $|Y^{(1)} \cdot Y^{(2)}| > 1/2$

Initialize list of $\ell$-tuples to be empty
Set $T = \frac{pk}{Cm2^\ell}$
for $i = 1$ TO $\Omega(k^{\ell-2} m \log^2 m)$ do
   Choose a random node $u$ in $G$, and $\ell - 1$ neighbors $u_1, u_2, ...u_{\ell-1}$
   if $|\Gamma_G(u) \cap \Gamma_G(u_1) \cap ... \cap \Gamma_G(u_{\ell-1})| \geq T$ then
      Add $(u, u_1, u_2, ...u_{\ell-1})$ to the list of $\ell$-tuples
   end if
end for
For each node $x$, add $x$ to each set $S_{u,u_1,u_2,...u_{\ell-1}}$ where $(u_1, u_2, ...u_{\ell-1})$ is an $\ell$-tuple and
$|\Gamma_G(u) \cap \Gamma_G(u_1) \cap ... \cap \Gamma_G(x)| \geq T$
Delete any set $S_{u,u_1,u_2,...u_{\ell-1}}$ that contains another set $S_{v,v_1,v_2,...v_{\ell-1}}$
return the remaining sets $S_{u,u_1,u_2,...u_{\ell-1}} \cup \{u, u_1, u_2, ...u_{\ell-1}\}$
```

What we need is an analogue of Claim 5.3 and Lemma 5.19. First the easy part:

Claim. *Suppose* $\mathrm{supp}(X^{(1)}) \cap \mathrm{supp}(X^{(2)}) \cap ... \cap \mathrm{supp}(X^{(\ell)}) \neq \emptyset$*, then*

$$Pr_Y[\textit{ for all } j = 1, 2, ..., \ell, |Y \cdot Y^{(j)}| > 1/2] \geq 2^{-\ell} \frac{k}{m}$$

The proof of this claim is identical to the proof of Claim 5.3. But what about an analogue of Lemma 5.19? To analyze the probability that a set X intersects each of the sets $X^{(1)}, X^{(2)}, ..., X^{(\ell)}$ we will rely on the following standard definition:

Definition 5.29. Given a collection of sets $supp(X^{(1)}), supp(X^{(2)}), ..., supp(X^{(\ell)})$, the *piercing number* is the minimum number of points $p_1, p_2, ..., p_r$ so that each set contains at least one point p_i.

The notion of piercing number is well-studied in combinatorics (see e.g. [121]). However, one is usually interested in upper-bounding the piercing number. For example, a classic result of Alon and Kleitman concerns the (p, q)-problem [5]: Suppose we are given a collection of sets that has the property that each choice of p of them has a subset of q which intersect. Then how large can the piercing number be? Alon and Kleitman proved that the piercing number is at most a fixed constant $c(p, q)$ independent of the number of sets [5].

However, here our interest in piercing number is not in bounding the minimum number of points needed but rather in analyzing how many ways there are of piercing a collection of sets with at most s points, since this will directly yield bounds on the probability that X intersects each of $X^{(1)}, X^{(2)}, ..., X^{(\ell)}$. We will need as a condition that each pair of sets has bounded intersection, and this holds in our model with high-probability.

Claim. *With high probability, the intersection of any pair* $\text{supp}(X^{(1)}), \text{supp}(X^{(2)})$ *has size at most* Q

Definition 5.30. We will call a set of ℓ sets a (k, Q) family if each set has size at most k and the intersection of each pair of sets has size at most Q.

Lemma 5.31. *The number of ways of piercing* (k, Q) *family (of* ℓ *sets) with* s *points is at most* $(\ell k)^s$*. And crucially if* $\ell \geq s + 1$*, then the number of ways of piercing it with* s *points is at most* $Qs(s + 1) + (\ell k)^{s-1}$*.*

Proof. The first part of the lemma is the obvious upper bound. Now let us assume $\ell \geq s+1$: Then given a set of s points that pierce the sets, we can partition the ℓ sets into s sets based on which of the s points is hits the set. (In general, a set may be hit by more than one point, but we can break ties arbitrarily). Let us fix any $s + 1$ of the ℓ sets, and let U be the the union of the pairwise intersections of each of these sets. Then U has size at most $Qs(s+1)$. Furthermore by the Pidgeon Hole Principle, there must be a pair of these sets that is hit by the same point. Hence one of the s points must belong to the set U, and we can remove this point and appeal to the first part of the lemma (removing any sets that are hit by this point). This concludes the proof of the second part of the lemma, too. □

Corollary 5.32. *The probability that the support of X hits each set in a (k, Q) family (of ℓ sets) is at most*

$$\sum_{2 \leq s \leq \ell - 1} (C_s + (\ell k)^{s-1}) \Big(\frac{k}{m}\Big)^s + \sum_{s \geq \ell} \Big(\frac{\ell k^2}{m}\Big)^s$$

where C_s is a constant depending polynomially on s.

Proof. We can break up the probability of the event that the support of X hits each set in a (k, Q) family into another family of events. Let us consider the probability that X pierces the family with $s \leq \ell - 1$ points or $s \geq \ell$ points. In the former case, we can invoke the second part of Lemma 5.31 and the probability that X hits any particular set of s points is at most $(k/m)^s$. In the latter case, we can invoke the first part of Lemma 5.31. □

Note that if $k \leq m^{1/2}$ then k/m is always greater than or equal to $k^{s-1}(k/m)^s$. And so asymptotically the largest term in the above sum is $(k^2/m)^\ell$ which we want to be asymptotically smaller than k/m which is the probability in Claim 5.5. So if $k \leq cm^{(\ell-1)/(2\ell-1)}$ then above bound is $o(k/m)$ which is asymptotically smaller than the probability that a given set of ℓ nodes that have a common intersection are each connected to a random (new) node in the connection graph. So again, we can distinguish between whether or not an ℓ-tuple has a common intersection or not and this immediately yields a new algorithm that works for k almost as large as $\sqrt{m}$, although the running time depends on how close k is to this bound.

Theorem 5.33. *Algorithm 5.4(RecoverGraphHighOrder(ℓ)) output sets that each corresponds to some i and contains all $Y^{(j)}$ for which $i \in \mathrm{supp}(X^{(j)})$. The algorithm runs in* *time* $\widetilde{O}(k^{\ell-2}mp + p^2n)$ *and succeeds with high probability if* $k \leq c\min(m^{(\ell-1)/(2\ell-1)}, \frac{\sqrt{n}}{\mu \log n})$ *and if* $p = \Omega(m^2/k^2 \log m)$

Hence this yields a polynomial time algorithm for finding an unknown incoherent dictionary whenever $k \leq c\min(\frac{\sqrt{n}}{\mu \log n}, m^{1/2-\epsilon})$ using Theorem 5.23. The proof of correctness of this algorithm is identical to one of Theorem 5.23 except that the definition of an *identifying* ℓ*-tuple* for coordinate i is a set of ℓ samples that have a common intersection and for which the first $\ell - 1$ have a common intersection that is exactly $\{i\}$. And note that the probability of finding an identifying ℓ-tuple for coordinate i is at least $\Omega(1/(mk^{\ell-1}))$.

The threshold of $k \leq \sqrt{n}/2\mu$ is a natural barrier: Suppose we are given a vector u of the form $u = Av$ where v has at most k non-zeros. Then if $k \leq \sqrt{n}/2\mu$, the vector v is uniquely defined and is the sparsest solution to the system $u = Ax$ (where x is the variable). However when $k > \sqrt{n}/2\mu$ this is no-longer true and there are incoherent dictionaries where a vector u admits more than one representation as a sparse linear combination of the columns of A. In fact, there are many known algorithms for recovering v from u up to the uniqueness threshold *when the dictionary* A *is known.* The above algorithm gives a method to recover the dictionary at almost the same threshold – i.e. if $k \leq c\min(\frac{\sqrt{n}}{\mu \log n}, m^{1/2-\epsilon})$.

Chapter 6

Learning Deep Representations

Can we provide theoretical explanation for the success of deep nets in applications such as vision, speech recognition etc. (see Bengio's survey [19])? Like most other ML tasks, learning deep nets seems NP-hard in any reasonable formulation, and in fact "badly NP-hard"because of many layers of hidden variables connected by nonlinear operations. Usually one imagines that NP-hardness is not a barrier to provable algorithms in ML because the input is drawn from some simple distribution and not worst-case. However, supervised learning of neural nets even on random inputs still seems as hard as cracking cryptographic schemes: this holds for depth-5 neural nets [93] and even ANDs of thresholds (a simple depth two network) [97].

However, modern deep nets are not "just"neural nets. They assume that the net (or some modification) can and should be run *in reverse* to get a *generative model* that produces a distribution that fits the empirical input distribution. Hinton promoted this viewpoint, and assumed that each level is a Restricted Boltzmann Machine (RBM), which is "reversible"in this sense. Vincent et al. [156] introduced *denoising autoencoder*, a generalization of an RBM (see Definition 6.3). These viewpoints allow a different methodology than classical backpropagation: *layerwise* learning. The bottom (observed) layer is learnt in *unsupervised fashion* using the provided data. This gives values for the next layer of hidden variables, which are used as the data to learn the next higher layer, and so on. The final net thus learned is also a good generative model for the distribution of the bottom layer[1].

Viewing layers of deep networks as reversible representations alludes to assumptions and techniques in the previous chapter. Each layer now is a sparse nonlinear code (as opposed to a sparse linear code in the previous section). We shall show a similar robustness assumption allows us to learn the nonlinear code under mild assumptions, and the learning algorithm can be composed to learn a random deep network. The rest of this chapter is organized as follows: in Section 6.1 we introduce the generative model and our main results. Every layer of our network will be a denoising autoencoder, and this is shown in Section 6.2. In Section 6.3 we illustrate the high-level algorithm, and prove it learns one layer networks. These results are extended in Section 6.4 to learn a multi-layer network. Section 6.5 describes the *graph recovery* algorithm, which is a crucial step in learning the network. The last layer of our network will be slightly different from the previous layers, and is discussed in Section 6.6. In Section 6.7 we show our network cannot be simplified: a two layer network cannot be represented by a one layer network. Finally we list the graph properties we used in Section 6.8.

6.1 Background and Main Results

6.1.1 Deep Networks

Here we describe the class of randomly chosen neural nets that can be learned by our algorithm. A network $\mathcal{D}(\ell, \rho_\ell, \{G_i\})$ has ℓ hidden layers of binary variables $\boldsymbol{h^{(\ell)}}$, $\boldsymbol{h^{(\ell-1)}}$, .., $\boldsymbol{h^{(1)}}$ from top to bottom and an observed layer $\boldsymbol{y}$ at bottom (see Figure 6.1). The set of nodes at layer $\boldsymbol{h^{(i)}}$ is denoted by N_i, and $|N_i| = n_i$. The nodes of the output layer are denoted by

[1] Recent work suggests that classical backpropagation-based learning of neural nets together with a few modern ideas like convolution and dropout training also performs very well [103], though the authors suggest that some unsupervised pretraining should help further.

Y (or N_0), and $|Y| = n_0$. For simplicity let $n = \max_i n_i$, and assume each $n_i > n^c$ for some positive constant c.

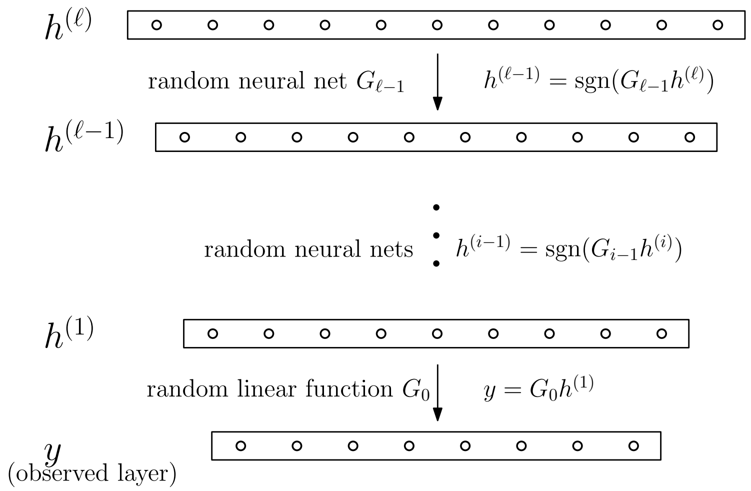

$h^{(\ell)}$
random neural net $G_{\ell-1}$
$h^{(\ell-1)} = \text{sgn}(G_{\ell-1}h^{(\ell)})$
$h^{(\ell-1)}$
random neural nets
$h^{(i-1)} = \text{sgn}(G_{i-1}h^{(i)})$
$h^{(1)}$
random linear function G_0
$y = G_0h^{(1)}$
y
(observed layer)

Figure 6.1: Example of a deep network

The graph between layers $i+1$ and i is a random bipartite graph $G_i(N_{i+1}, N_i, E_i, w)$, where every edge $(u, v) \in N_{i+1} \times N_i$ in E_i has probability p_i. We denote this distribution of graphs by $\mathcal{G}_{n_{i+1},n_i,p_i}$. Each edge $e \in E_i$ carries a random weight $w(e)$ in $\{-1, 1\}$.The set of positive edges are denoted by $E_i^+ = \{(u, v) \in N_{i+1} \times N_i : w(u, v) > 0\}$, and the set of negative edges by E^-.

The threshold for the generative direction (from top to bottom) is 0 for every node. The top layer $\boldsymbol{h^{(\ell)}}$ is a random vector where every component is an independent Bernoulli variable with probability ρ_ℓ[2]. Each node in layer $\ell - 1$ computes a weighted sum of its neighbors in layer ℓ, and becomes 1 iff that sum strictly exceeds 0. We use sgn$(\cdot)$ to denote the threshold function:

$$\text{sgn}(x) = 1 \text{ if } x > 0 \text{ and } 0 \text{ else.} \tag{6.1}$$

Applying sgn() to a vector involves applying it componentwise.

[2] We can also allow the top layer to be uniform over all the $\rho_\ell n_\ell$ sparse vectors, because in that case all the probabilities computed in the following sections changes only by a multiplicative factor of $(1 + o(1))$.

Hence the network generates the observed examples using the following equations: $\boldsymbol{h^{(i-1)}} = \mathrm{sgn}(G_{i-1}\boldsymbol{h^{(i)}})$ for all $i > 0$ and $\boldsymbol{h^{(0)}} = G_0\boldsymbol{h^{(1)}}$ (i.e., no threshold at the observed layer).Here (with slight abuse of notation) G_i stands for both the bipartite graph and the bipartite weight matrix of the graph at layer i.

Throughout this chapter, "with high probability" means the probability is at least $1-n^{-C}$ for arbitrary large constant C. Moreover, $f \gg g$, $f \ll g$ mean $f \geq n^{\epsilon}g$, $f \leq n^{-\epsilon}g$ where ϵ is a arbitrarily small constant[3].

More network notations. The expected degree from N_i to N_{i+1} is d_i, that is, $d_i \triangleq p_i|N_{i+1}| = p_in_{i+1}$, and the expected degree from N_{i+1} to N_i is denoted by $d_i' \triangleq p_i|N_i| = p_in_i$. The set of forward neighbors of $u \in N_{i+1}$ in graph G_i is denoted by $F_i(u) = \{v \in N_i : (u,v) \in E_i\}$, and the set of backward neighbors of $v \in N_i$ in G_i is denoted by $B_i(v) = \{u \in N_{i+1} : (u,v) \in E_i\}$. We use $F_i^+(u)$ to denote the positive neighbors: $F_i^+(u) \triangleq \{v, : (u,v) \in E_i^+\}$ (and similarly for $B_i^+(v)$). The expected density of the layers are defined as $\rho_{i-1} = \rho_i d_{i-1}/2$ (ρ_ℓ is given as a parameter of the model). For higher level ancestors, we use $B^{(j)}(u)$ to denote the all nodes that have a path to u at layer j, and $B_+^{(j)}(u)$ to denote all nodes that have a positive path to u at layer j.

Our analysis works while allowing network layers of different sizes and different degrees. For simplicity, we recommend first-time readers to assume all the n_i's are equal, and $d_i = d_i'$ for all layers.

Discussions Recent papers have given theoretical analysis of models with multiple levels of hidden features, including SVMs [40, 115]. However, none of these solves the task of recovering a ground-truth neural network given its output distribution.

Though real-life neural nets are not random, our consideration of random deep networks makes some sense for theory. Sparse denoising autoencoders are reminiscent of other objects

[3] It is possible to make the analysis tighter and allow $f \geq g \,\mathrm{poly}\log n$, we choose this to make the presentation simpler.

such as error-correcting codes, compressed sensing, etc. which were all first analysed in the random case. As mentioned, provable reconstruction of the hidden layer in a *known* autoencoder already seems a nonlinear generalization of compressed sensing (1-bit compressed sensing, [30]), and even the usual compressed sensing seems possible only if the adjacency matrix has "random-like" properties (low coherence or restricted isoperimetry or lossless expansion). In fact our result that a single layer of our generative model is a sparse denoising autoencoder can be seen as an analog of the fact that random matrices are good for compressed sensing/sparse reconstruction (see Donoho [55] for general matrices and Berinde et al. [21] for sparse matrices). Of course, in compressed sensing the matrix of edge weights is known whereas here it has to be learnt, which is the main contribution of our work. Furthermore, we show that our algorithm for learning a single layer of weights be extended to do layerwise learning of the entire network.

Along the way we show interesting properties of such randomly-generated neural nets: (a) Each pair of layers constitutes a denoising autoencoder in the sense of Vincent et al.; see Lemma 6.4. (b) The reverse computation (computing the top hidden layer given the value of the observed vector) can also be performed by the same network by appropriately changing the thresholds at the computation nodes (c) The reverse computation is stable to dropouts (d) the distribution generated by a two-layer net cannot be represented by any single layer neural net (see Appendix), which in turn suggests that a random t-layer network cannot be represented by *any* t/2-level neural net[4].

Note that properties (a) to (d) are *assumed* in modern deep network research (e.g., (b) is a heuristic trick called "weight tying") so the fact that they *provably* hold for a random generative model can be seen as some theoretical validation of those assumptions.

[4] Proving this for $t > 3$ is difficult however since showing limitations of even 2-layer neural nets is a major open problem in computational complexity theory. Some deep learning papers mistakenly cite an old paper for such a result, but the result that actually exists is far weaker.

6.1.2 Main Results

Theorem 6.1. *For a network $\mathcal{D}(\ell, \rho_\ell, \{G_i\})$, if all graphs G_i's are chosen according to $\mathcal{G}_{n_{i+1},n_i,p_i}$, weights are chosen at random, and the parameters satisfy:*

1. *All $d_i \gg 1, d_i' \gg 1$.*

2. *For all layers, $n_i^3(d_{i-1}')^8/n_{i-1}^8 \ll 1$.*

3. *For all but last layer ($i \geq 1$), $\rho_i^3 \ll \rho_{i+1}$.*

4. *For last layer, $\rho_1 d_0 = O(1)$, $d_0^{3/2}/d_1 d_2 \ll 1$, $\sqrt{d_0}/d_1 \ll 1$, $d_0^2 \ll n_1$*

Then there is an algorithm using $O(\log^2 n/\rho_\ell^2)$ samples, running in time $O(\sum_{i=1}^{\ell} n_i((d_{i-1}')^3 + n_{i-1}))$ that learns the network with high probability over both the graph and the samples.

Remark 6.2. We include the last layer whose output is real instead of 0/1, in order to get fully dense outputs. We can also learn a network without this layer, in which case the last layer needs to have density at most $1/\operatorname{poly}\log(n)$, and Condition 4 is no long needed.

Although we assume each layer of the network is a random graph, we are not using all the properties of the random graph. The properties we need are listed in Section 6.8.

6.2 Denoising Property

Experts feel that deep networks satisfy some intuitive properties. First, intermediate layers in a deep representation should approximately preserve the useful information in the input layer. Next, it should be possible to go back/forth easily between the representations in two successive layers, and in fact they should be able to use the neural net itself to do so. Finally, this process of translating between layers should be noise-stable to small amounts of random noise. All this was implicit in the early work on RBM and made explicit in the paper of Vincent et al. [156] on *denoising autoencoders*. For a theoretical justification of the notion of a denoising autoencoder based upon the "manifold assumption" of machine learning see the survey of Bengio [19].

Definition 6.3 (Denoising autoencoder)**.** An *autoencoder* consists of an *decoding function* $D(\boldsymbol{h}) = s(W\boldsymbol{h} + b)$ and a *encoding function* $E(\boldsymbol{y}) = s(W'\boldsymbol{y} + b')$ where W, W' are linear transformations, b, b' are fixed vectors and s is a nonlinear function that acts identically on each coordinate. The autoencoder is *denoising* if $E(D(\boldsymbol{h}) + \boldsymbol{\eta}) = \boldsymbol{h}$ with high probability where $\boldsymbol{h}$ is drawn from the input distribution, η is a noise vector drawn from the noise distribution, and $D(\boldsymbol{h}) + \boldsymbol{\eta}$ is a shorthand for "$E(\boldsymbol{h})$ corrupted with noise $\boldsymbol{\eta}$." The autoencoder is said to use *weight tying* if $W' = W^T$.

The popular choices of s includes logistic function, soft max, etc. In this work we choose s to be a simple threshold on each coordinate (i.e., the test > 0, this can be viewed as an extreme case of logistic function). Weight tying is a popular constraint and is implicit in RBMs. Our work also satisfies weight tying.

In practice the denoising autoencoder property is a "soft" constraint on the nonlinear optimization used in learning: it takes the form of a term that penalizes deviations from this ideal property. A denoising autoencoder is a network that makes this penalty term zero. Thus one could conceivably weaken the above definition to say $E(D(\boldsymbol{h}) + \boldsymbol{\eta}) \approx \boldsymbol{h}$ instead of exact equality. However, our networks satisfy the stronger equality constraint for most datapoints.

We will show that each layer of our network is a denoising autoencoder with very high probability. (Each layer can also be viewed as an RBM with an additional energy term to ensure sparsity of $\boldsymbol{h}$.) Later we will of course give efficient algorithms to learn such networks without recoursing to local search. In this section we just prove they satisfy Definition 6.3.

The single layer has m hidden and n output (observed) nodes. The connection graph between them is picked randomly by selecting each edge independently with probability p and putting a random weight on it in $\{-1, 1\}$. Then the linear transformation W corresponds simply to this matrix of weights. In our autoencoder we set $b = \mathbf{0}$ and $b' = 0.2d' \times \mathbf{1}$, where $d' = pn$ is the expected degree of the random graph on the hidden side. (By simple Chernoff bounds, every node has degree very close to d'.) The hidden layer $\boldsymbol{h}$ has the following prior: it is given a 0/1 assignment that is 1 on a random subset of hidden nodes of size ρm. This means the number of nodes in the output layer that are 1 is at most $\rho m d' = \rho n d$, where $d = pm$ is the expected degree on the observed side. We will see that since $b = \mathbf{0}$ the number of nodes that are 1 in the output layer is close to $\rho m d'/2$.

Lemma 6.4. *If $\rho m d' < 0.05n$ (i.e., the assignment to the observed layer is also fairly sparse) then the single-layer network above is a denoising autoencoder with high probability (over the choice of the random graph and weights), where the noise distribution is allowed to flip every output bit independently with probability* 0.01*.*

Remark: The parameters accord with the usual intuition that the information content must *decrease* when going from observed layer to hidden layer.

Proof. By definition, $D(\boldsymbol{h}) = \text{sgn}(W\boldsymbol{h})$. Let's understand what $D(\boldsymbol{h})$ looks like. If S is the subset of nodes in the hidden layer that are 1 in $\boldsymbol{h}$, then the unique neighbor property (Lemma 6.27) implies that (i) With high probability each node u in S has at least $0.9d'$ neighboring nodes in the observed layer that are neighbors to no other node in S. Furthermore, at least $0.44d'$ of these are connected to u by a positive edge and $0.44d'$ are connected by a negative edge. All $0.44d'$ of the former nodes must therefore have a value 1 in $D(\boldsymbol{h})$. Furthermore, it is also true that the total weight of these $0.44d'$ positive edges is at least $0.21d'$. (ii) Each v not in S has at most $0.1d'$ neighbors that are also neighbors of any node in S.

Now let's understand the encoder, specifically, $E(D(\boldsymbol{h}))$. It assigns 1 to a node in the hidden layer iff the weighted sum of all nodes adjacent to it is at least $0.2d'$. By (i), every node in S must be set to 1 in $E(D(\boldsymbol{h}))$ and no node in $\overline{S}$ is set to 1. Thus $E(D(\boldsymbol{h})) = \boldsymbol{h}$ for most $\boldsymbol{h}$'s and we have shown that the autoencoder works correctly. Furthermore, there is enough margin that the decoding stays stable when we flip 0.01 fraction of bits in the observed layer. □

For the last layer, since the output is not 0/1, we shall choose a different threshold.

Lemma 6.5. *For the last layer, with high probability the encoder* $E(\boldsymbol{y}) = \mathrm{sgn}(G^T\boldsymbol{y} - 0.5\mathbf{1})$ *and linear decoder* $D(\boldsymbol{h}) = G\boldsymbol{h}$ *form a denoising autoencoder, where the noise vector* η *has independent components each with variance at most* $o(d_0'/\log^2 n)$*.*

Proof. When there is no noise, we know $E(D(\boldsymbol{h})) = \mathrm{sgn}(G^TG\boldsymbol{h} - 0.5d_0'\mathbf{1})$. With high probability the matrix G^TG has value at least $0.9d_0'$ on diagonals. For any fixed $\boldsymbol{h}$, the support of the i-th row of G^TG and the support of $\boldsymbol{h}$ have a intersection of size at most $d_0'\log^2 n$ with high probability. Also, at all these intersections, the entries of G^TG has random signs, and variance bounded by $O(1)$. Hence if $\boldsymbol{h}_i = 1$ $(G^TG\boldsymbol{h})_i \geq 0.9d_0' - O(\sqrt{d_0'}\log^2 n)$; if $\boldsymbol{h}_i = 0$ $(G^TG\boldsymbol{h})_i \leq O(\sqrt{d_0'}\log^2 n)$. Setting the threshold at $0.5d_0'$ easily distinguishes between these two cases.

Even if we add noise, since the inner product of G_i and the noise vector is bounded by $o(d_0)$ with high probability, the autoencoder still works. □

6.3 Learning a Single Layer Network

We first consider the question of learning a single layer network, which as noted amounts to learning *nonlinear* dictionaries. It perfectly illustrates how we leverage the sparsity and the randomness of the *support graph.*

The overall algorithm is illustrated in Algorithm 6.1.

We start with the simplest subcase of the first step: all edge weights are 1, and consider only pairwise correlations. Then we shall consider 3-wise correlations to figure out which triples share a common neighbor in the hidden layer. The correlation structure is used by

Algorithm 6.1. High Level Algorithm

```
input samples $\boldsymbol{y}$'s generated by a deep network described in Section 6.1
output Output the network/encoder and decoder functions
for $i = 1$ TO $\ell$ do
    Call LastLayerGraph/3-Wise Graph on $\boldsymbol{h^{(i-1)}}$ to construct the correlation graph
    Call RecoverGraph3Wise to learn the positive edges $E_i^+$
    Use PartialEncoder to encode all $\boldsymbol{h^{(i-1)}}$ to $\boldsymbol{h^{(i)}}$
    Call LearnGraphLast/LearnGraph to learn the graph/decoder between layer $i$ and $i-1$.
end for
```

the Graph Recovery procedure (described later in Section 6.5) to learn the support of the graph.

In Section 6.3.1 we show how to generalize these ideas to learn positive edges when edges can have weights $\{\pm 1\}$.

In Section 6.3.2 we construct a partial encoder, which can do encoding given only the support of positive edges. The result there is general and works in the multi-layer setting.

Finally we give a simple algorithm for learning the negative edges.

6.3.1 Correlation Implies Common Cause

Warm up: 0/1 weights

In this part we work with the simplest setting: a single level network with m hidden nodes, n observed nodes, and a random (but unknown) bipartite graph $G(U, V, E)$ connecting them (see Figure 6.2). The expected backdegree of observed nodes are d. All edge weights are 1, so learning G is equivalent to finding the edges. Recall that we denote the hidden variables by $\boldsymbol{h}$ and the observed variables by $\boldsymbol{y}$, and the neural network implies $\boldsymbol{y} = \mathrm{sgn}(G\boldsymbol{h})$.

Theorem 6.6. *Let G be a graph satisfying properties $\mathcal{P}_{sing}$. Suppose $\rho \ll 1/d^2$, with high probability over the samples, Algorithm 6.2 construct a graph $\hat{G}$, where u, v are connected in $\hat{G}$ iff they have a common neighbor in G.*

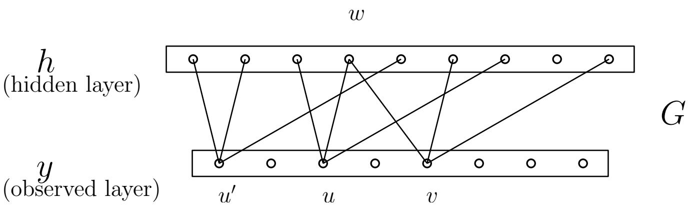


Figure 6.2: Single layered network

Algorithm 6.2. PairwiseGraph

```
input  $N = O(\log^2 n/\rho^2)$ samples of $\boldsymbol{y} = \mathrm{sgn}(G\boldsymbol{h})$
output  Graph $\hat{G}$ on vertices $V$, $u, v$ are connected if $u, v$ share a positive neighbor in $G$
  for each $u, v$ in the output layer do
    if there are at least $2\rho N/3$ samples of $\boldsymbol{y}$: $\boldsymbol{y}_u = \boldsymbol{y}_v = 1$ then
      connect $u$ and $v$ in $\hat{G}$
    end if
  end for
```

As mentioned, the crux of the algorithm is to compute the correlations between observed variables. The following lemma shows pairs of variables with a common parent both get value 1 more frequently than a typical pair. To simplify notation let $\rho_y = \rho d$ be the approximate expected density of output layer.

Lemma 6.7. *Under the assumptions of Theorem 6.6, if two observed nodes u, v have a common neighbor in the hidden layer then*

$$\Pr_{\boldsymbol{h}}[\boldsymbol{y}_u = 1, \boldsymbol{y}_v = 1] \geq \rho$$

otherwise,

$$\Pr_{\boldsymbol{h}}[\boldsymbol{y}_u = 1, \boldsymbol{y}_v = 1] \leq 2\rho_y^2$$

Proof. When u and v have a common neighbor z in the input layer, as long as z is fired both u and v are fired. Thus $\Pr[\boldsymbol{y}_u = 1, \boldsymbol{y}_v = 1] \geq \Pr[\boldsymbol{h}_z = 1] = \rho$.

On the other hand, suppose the neighbor of u ($B(u)$) and the neighbors of v ($B(v)$) are disjoint. Then to make them both 1, we need $\mathrm{supp}(\boldsymbol{h})$ to intersect both $B(u)$ and $B(v)$.

These two events are independent (because the sets are disjoing), the probability is at most $(\rho|B(u)|)(\rho|B(v)|) \leq 2\rho_{\boldsymbol{y}}^2$. □

The lemma implies that when $\rho_y^2 \ll \rho$(which is equivalent to $\rho \ll 1/(d^2)$), we can find pairs of nodes with common neighbors by estimating the probability that they are both 1. The theorem follos directly from the lemma.

Algorithm 6.3. 3-WiseGraph

```
input $N = O(\log n/\rho^2)$ samples of $\boldsymbol{y} = \mathrm{sgn}(Gh)$, where $\boldsymbol{h}$ is unknown and chosen from
  uniform $\rho m$-sparse distribution
output Hypergraph $\hat{G}$ on vertices $V$. $\{u, v, s\}$ is an edge if and only if they share a positive
  neighbor in $G$
  for each $u, v, s$ in the observed layer of $\boldsymbol{y}$ do
    if there are at least $\rho N/3$ samples of $\boldsymbol{y}$ satisfying all $u, v$ and $s$ are fired then
      add $\{u, v, s\}$ as an hyperedge for $\hat{G}$
    end if
  end for
```

In order to weaken the assumption that $\rho \ll 1/d^2$, we can consider higher order correlations using Algorithm 6.3 3-WiseGraph. In the following lemma we show how 3-wise correlation works when $\rho \ll d^{-3/2}$ ($\rho_y^3 \ll \rho$).

Lemma 6.8. *If the graph G satisfies $\mathcal{P}_{sing}$ and $\mathcal{P}_{sing+}$, for any u, v, s in the observed layer,*

1. $\Pr_{\boldsymbol{h}}[\boldsymbol{y}_u = \boldsymbol{y}_v = \boldsymbol{y}_s = 1] \geq \rho$*, if u, v, s have a common neighbor*

2. $\Pr_{\boldsymbol{h}}[\boldsymbol{y}_u = \boldsymbol{y}_v = \boldsymbol{y}_s = 1] \leq 2\rho_y^3 + O(\rho_y \rho \log n)$ *otherwise.*

Proof. The proof is very similar to the proof of Lemma 6.7.

If u,v and s have a common neighbor z, then $\Pr[\boldsymbol{y}_u = \boldsymbol{y}_v = \boldsymbol{y}_s = 1] \leq \Pr[\boldsymbol{h}_u = 1] = \rho$.

On the other hand, if they don't share a common neighbor, then $\Pr_{\boldsymbol{h}}[\boldsymbol{y}_u = \boldsymbol{y}_v = \boldsymbol{y}_s = 1] = \Pr[\mathit{supp}(\boldsymbol{h})$ intersects with $B(u), B(v), B(s)]$. There are two ways $\mathrm{supp}(\boldsymbol{h})$ can intersect with three sets: 1. it intersects with the three sets independently or 2. it intersects with the intersection of two. The probability of the first type is easily bound by $2\rho_y^3$. Notice that the property $\mathcal{P}_{sing+}$ bounds the pairwise intersections of these sets by $O(\log n)$, so the probability of second type is bounded by $O(\rho_y \rho \log n)$. □

General case: finding common positive neighbors

The previous two algorithms still work even when weights can be both positive and negative, as long as the weights are random. We will show this in the next lemma. A difference in notation here is $\rho_y = \rho d/2$ (because only half of the edges have positive weights).

Lemma 6.9. *When the graph G satisfies properties $\mathcal{P}_{sing}$ and $\mathcal{P}_{sing+}$, for any u, v, s in the observed layer,*

1. $\Pr_{\boldsymbol{h}}[\boldsymbol{y}_u = \boldsymbol{y}_v = \boldsymbol{y}_s = 1] \geq \rho/2$*, if u, v, s have a common positive neighbor*

2. $\Pr_{\boldsymbol{h}}[\boldsymbol{y}_u = \boldsymbol{y}_v = \boldsymbol{y}_s = 1] \leq O(\rho_y^3 + \rho_y \rho \log n)$*, otherwise.*

Proof. The proof is similar to the proof of Lemma 6.8.

First, when u, v, s have a common positive neighbor z, let U be the neighbors of u, v, s except z. Hence $U = B(u) \cup B(v) \cup B(s) \setminus \{z\}$. By property $\mathcal{P}_{sing}$, we know the size of U is at most $3.3d$. With at least $1 - 3.3\rho d \geq 0.9$ probability, none of them is 1. Hence

$$\Pr[\boldsymbol{y}_u = \boldsymbol{y}_v = \boldsymbol{y}_s = 1] \geq \Pr[\boldsymbol{h}_z = 1] \Pr[\boldsymbol{h}_U = \boldsymbol{0}] \geq \rho/2.$$

On the other hand, if u, v and s don't have a positive common neighbor, then we have $\Pr_{\boldsymbol{h}}[\boldsymbol{y}_u = \boldsymbol{y}_v = \boldsymbol{y}_s = 1] \leq \Pr[supp(\boldsymbol{h})$ intersects with $B^+(u), B^+(v), B^+(s)]$. This case is the same as Lemma 6.8. □

6.3.2 Paritial Encoder: Finding h given y

Although we only know the positive edges of G, it is enough for the encoding process. This is based on *unique neighbor property* (see Section 6.8 for more details). Intuitively, when

$\rho d \ll 1$ any $\boldsymbol{h}_z = 1$ will have a large fraction of *unique positive neighbors* (that are not neighbors of any z' with $\boldsymbol{h}_{z'} = 1$). These unique neighbors cannot be canceled by any -1 edges, so they must be 1 in $\boldsymbol{y}$.

Algorithm 6.4. PartialEncoder

```
input  positive edges $E^+$, sample $\boldsymbol{y} = \mathrm{sgn}(G\boldsymbol{h})$, threshold $\theta$
output  the hidden variable $\boldsymbol{h}$
  return  $\boldsymbol{h} = \mathrm{sgn}((E^+)^\top \boldsymbol{y} - \theta \mathbf{1})$
```

Lemma 6.10. *If the support of vector* $\boldsymbol{h}$ *has the* $11/12$*-strong unique neighbor property in* G*, then Algorithm 6.4 returns* $\boldsymbol{h}$ *given input* E^+ *and* $\theta = 0.3d'$*.*

Proof. Let $S = \mathrm{supp}(\boldsymbol{h})$.

If $u \in S$, at most $d'/12$ of its neighbors (in particular that many of its positive neighbors) can be shared with other vertices in S. Thus u has at least $(0.3)d'$ unique positive neighbors, and these are all “on”.

Now if $u \notin S$, it can have an intersection at most $d'/12$ with $F(S)$ (by the definition of strong unique neighbors). Anything outside $F(S)$ must be 0 in $\boldsymbol{y}$. Thus there cannot be $(0.3)d'$ of its neighbors that are 1. □

Remark 6.11. Notice that Lemma 6.10 only depends on the unique neighbor property, which holds for the support of any vector $\boldsymbol{h}$ with high probability over the randomness of the graph. Therefore this ParitialEncoder can be used even when we are learning the layers of deep network (and $\boldsymbol{h}$ is not a i.i.d. random sparse vector).

6.3.3 Learning the Graph: Finding -1 edges.

Now that we can find $\boldsymbol{h}$ given $\boldsymbol{y}$, the idea is to use many such pairs $(\boldsymbol{h}, \boldsymbol{y})$ and the partial graph E^+ to determine all the non-edges (i.e., edges of 0 weight) of the graph. Since we know all the $+1$ edges, we can thus find all the -1 edges.

Consider some $(\boldsymbol{h}, \boldsymbol{y})$, and suppose $\boldsymbol{y}_v = 1$, for some output v. Now suppose we knew that precisely one element of $B^+(v)$ is 1 in $\boldsymbol{h}$ (recall: B^+ denotes the back edges with weight $+1$). Note that this is a condition we can *verify*, since we know both $\boldsymbol{h}$ and E^+. In this case, it must be that there is no edge between v and $S \setminus B^+$, since if there had been an edge, it must be with weight -1, in which case it would cancel out the contribution of $+1$ from the B^+. Thus we ended up "discovering" that there is no edge between v and several vertices in the hidden layer.

We now claim that observing polynomially many samples $(\boldsymbol{h}, \boldsymbol{y})$ and using the above argument, we can discover every non-edge in the graph. Thus the complement is precisely the support of the graph, which in turn lets us find all the -1 edges.

Algorithm 6.5. Learning Graph

```
input positive edges $E^+$, samples of $(\boldsymbol{h}, \boldsymbol{y})$
output $E^-$
$R \leftarrow (U \times V) \setminus E^+$.
for each of the samples $(\boldsymbol{h}, \boldsymbol{y})$, and each $v$ do
    Let $S$ be the support of $\boldsymbol{h}$
    if $\boldsymbol{y}_v = 1$ and $S \cap B^+(v) = \{u\}$ for some $u$ then
        for $s \in S$ do
            remove $(s, v)$ from $R$.
        end for
    end if
end for
return $R$
```

Note that the algorithm R maintains a set of candidate E^-, which it initializes to $(U \times V) \setminus E^+$, and then removes all the non-edges it finds (using the argument above). The main lemma is now the following.

Lemma 6.12. *Suppose we have $N = O(\log^2 n/(\rho^2 d))$ samples $(\boldsymbol{h}, \boldsymbol{y})$. Then with high probability over choice of the samples, Algorithm 6.5 outputs the set E^-.*

The lemma follows from the following proposition, which says that the probability that a non-edge (z, u) is identified by one sample $(\boldsymbol{h}, \boldsymbol{y})$ is at least $\rho^2 d/3$. The probability that this particular non-edge is not identified after $O(\log^2 n/(\rho^2 d))$ samples is $< 1/n^{O(\log n)}$. Therefore all non-edges must be found with high probability by union bound.

Proposition 6.13. *Let (z, u) be a non-edge, then with probability at least $\rho^2 d/3$, all of the followings hold: 1. $\boldsymbol{h}_z = 1$, 2. $|B^+(u) \cap supp(\boldsymbol{h})| = 1$, 3. $|B^-(u) \cap supp(\boldsymbol{h})| = 0$.*

If such $(\boldsymbol{h}, \boldsymbol{y})$ is one of the samples we consider, (z, u) will be removed from R by Algorithm 6.5.

Proof. The latter part of the proposition follows from the description of the algorithm. Hence we only need to bound the probability of the three events.

The three events are independent because $\{z\}$, $B^+(u)$, $B^-(u)$ are three disjoint sets. The first event has probability ρ, the second event have probability roughly $\rho|B^+(u)|$ (since $\rho d \ll 1$) and the third event has probability at least $1 - \rho|B^-(u)|$. Therefore the probability that all three events happen is at least $\rho^2/3$. □

6.4 Learning a Multi-layer Network

Among the four steps of the algorithm, graph recovery and partial encoding are not assuming any distribution on the hidden variable $\boldsymbol{h}$. Hence these two steps can still be used for learning a multi-layer network. On the other hand, building the correlation graph and learning the -1 edges all depend on the distribution of $\boldsymbol{h}$. In this section we shall show the effect of correlation between hidden variables are bounded. Although in a multi-layer network the hidden variable $\boldsymbol{h^{(1)}}$ does not have independent components, the first and last step of the algorithm can still work. The key idea here is that although the maximum correlation between two nodes in z, t in layer $\boldsymbol{h^{(i+1)}}$ can be large, there are only very few pairs with such high correlation. Since the graph G_i is random and independent of the upper layers, we don't expect to see many such pairs in the $B(u)$, $B(v)$ for any u, v in $\boldsymbol{h^{(i)}}$.

6.4.1 Correlation Grapph in a Multilayer Network

For the first step of the algorithm, we show correlation does not change the behavior of the algorithm by the following theorem.

Theorem 6.14. *For any $1 \leq i \leq \ell-1$, and if the network satisfies Property $\mathcal{P}_{mul}+$ (which are satisfied by random graphs if $\rho_{i+1}^3 \ll \rho_i$), then given $O(\log^2 n/\rho_{i+1}^2)$ samples, Algorithm 6.3 3-WiseGraph constructs a hypergraph $\hat{G}$, where (u, v, s) is an edge if and only if they share a positive neighbor in G_i.*

The theorem follows directly from the lemma below.

Lemma 6.15. *Under the assumption of Theorem 6.14, for any $i \leq \ell - 1$ and any u, v, s in the layer of $\boldsymbol{h}^{(i)}$, if they have a common positive neighbor(parent) in layer of $\boldsymbol{h}^{(i+1)}$*

$$\Pr[\boldsymbol{h}^{(i)}{}_u = \boldsymbol{h}^{(i)}{}_v = \boldsymbol{h}^{(i)}{}_s = 1] \geq \rho_{i+1}/3,$$

otherwise

$$\Pr[\boldsymbol{h}^{(i)}{}_u = \boldsymbol{h}^{(i)}{}_v = \boldsymbol{h}^{(i)}{}_s = 1] \leq 0.2\rho_{i+1}$$

Proof. The idea is to go all the way back to the ℓ-th layer and use the assumption on the input distribution there.

Consider first the case when u, v and s have a common positive neighbor z in the layer of $\boldsymbol{h}^{(i+1)}$. Similar to the proof of Lemma 6.9, when $\boldsymbol{h}^{(i+1)}{}_z = 1$ and none of other neighbors of u, v and s in the layer of $\boldsymbol{h}^{(i+1)}$ is 1, we know $\boldsymbol{h}^{(i)}{}_u = \boldsymbol{h}^{(i)}{}_v = \boldsymbol{h}^{(i)}{}_s = 1$. Going back to level ℓ, this is implied by the event $B_+^{(\ell)}(z)$ has at least one component being 1, and everything else in $B^{(\ell)}(u) \cup B^{(\ell)}(v) \cup B^{(\ell)}(s)$ are all 0. By Property $\mathcal{P}_{mul}$, $\left|B_+^{(\ell)}(z)\right| \geq 0.8\rho_{i+1}/\rho_\ell$, and $\left|B^{(\ell)}(u) \cup B^{(\ell)}(v) \cup B^{(\ell)}(s)\right| \leq 2^{\ell+1}\rho_i/\rho_\ell$. Let $S = B^{(\ell)}(u) \cup B^{(\ell)}(v) \cup B^{(\ell)}(s) \setminus B_+^{(\ell)}(z)$, we know (using independence at the first layer $\boldsymbol{h}^{(\ell)}$)

$$\Pr[\boldsymbol{h}^{(i)}{}_u = \boldsymbol{h}^{(i)}{}_v = \boldsymbol{h}^{(i)}{}_s = 1] \geq \Pr[\text{supp}(\boldsymbol{h}^{(\ell)}) \cap B_+^{(\ell)}(z) \neq \emptyset] \Pr[\text{supp}(\boldsymbol{h}^{(\ell)}) \cap B_+^{(\ell)}(z) = \emptyset] \geq \rho_{i+1}/3.$$

On the other hand, if u, v and s don't have a common positive neighbor in layer of $\boldsymbol{h^{(i+1)}}$, consider event E: S intersects each of $B_+^{(\ell)}(u)$, $B_+^{(\ell)}(v)$, $B_+^{(\ell)}(s)$. Clearly, the target probability can be upperbounded by the probability of E. There are three ways the support of $\boldsymbol{h^{(\ell)}}$ can intersect with all three sets: 1. intersect with the intersection of three sets; 2. intersect with the intersection of two of the sets, and intersect at a different place with the third set; 3. intersect all three sets at different places.

By graph Property $\mathcal{P}_{mul+}$, the size of pairwise and three-wise intersections can be bounded, let $A \le 2^{\ell+1}\rho_i/\rho_\ell$ be the size of the largest set, $B \le O(\log n)\rho_{i+1}/\rho_\ell$ be the size of the largest pairwise intersection, and let $C = o(\rho_{i+1}/\rho_\ell)$ be the size of 3-wise intersection, we can bound the probability of event E by

$$\Pr[E] \le \rho_\ell C + 3\rho_\ell^2 AB + \rho_\ell^3 A^3 \le 0.2\rho_{i+1}.$$

The bound on the third term uses the assumption that $\rho_i^3 \ll \rho_{i+1}$.

□

6.4.2 Learning the -1 Edges

In order to show Algorithm 6.5 still works, we prove the following lemma that replaces Lemma 6.12.

Lemma 6.16. *If the network satisfy $\mathcal{P}_{neg+}$, given $O(\log^2 n/(\rho_{i+1}^2))$ samples, with high probability over the choice of the samples, Algorithm 6.5 returns E^-.*

Proof. Similar as Lemma 6.12, it suffices to prove the following Proposition 6.17 (which replaces Proposition 6.13). □

Proposition 6.17. *For any $(x, u) \notin E_i$, with probability $\Omega(\rho_{i+1}^2)$ over the choice of $\boldsymbol{h^{(i+1)}}$, the following events happen simultaneously: 1. $x \in$ supp($\boldsymbol{h^{(i+1)}}$), 2. $\left|B^+(u) \cap \mathrm{supp}(\boldsymbol{h^{(i+1)}})\right| = 1$, 3. $\left|B^-(u) \cap \mathrm{supp}(\boldsymbol{h^{(i+1)}})\right| = 0$.*

When these events happen, (x,u) is removed from R by Algorithm 6.5.

Proof. Let t be an arbitrary node in $B^+(u)$, and let $S = B(u) \setminus \{t\}$. For simplicity, we shall bound probability of the following three events happen simultaneously: 1. $\boldsymbol{h}_x^{(\boldsymbol{i+1})} = 1$; 2. $\boldsymbol{h}_t^{(\boldsymbol{i+1})} = 1$; 3. $\mathrm{supp}(\boldsymbol{h}^{(\boldsymbol{i+1})}) \cap S = \emptyset$.

Clearly, these three events imply the three events in the lemma. The set S does not depend on graphs G_{i+1} to $G_{\ell-1}$. Going back to the first layer, let $S_1 = B_+^{(\ell)}(x) \setminus B^{(\ell)}(\{x\} \cup S)$, $S_2 = B_+^{(\ell)}(t) \setminus B^{(\ell)}(\{x\} \cup S)$, $S_3 = B^{(\ell)}(t) \cup B^{(\ell)}(x) \cup B^{(\ell)}(S) \setminus (S_1 \cup S_2)$. By Property $\mathcal{P}_{neg+}$, $|S_1| \geq \rho_{i+1}/2\rho_\ell$, $|S_2| \geq \rho_{i+1}/2\rho_\ell$, and $|S_3| \leq 2^{\ell+2}\rho_i/\rho_\ell$. Let $H = \mathrm{supp}(\boldsymbol{h}^{(\boldsymbol{\ell})})$, the probability of the three events is at least

$$\Pr[H \cap S_1 \neq \emptyset] \Pr[H \cap S_2 \neq \emptyset] \Pr[H \cap S_3 = \emptyset] \geq (\rho_{i+1}/3)^2 (1 - O(\rho_i)) = \Omega(\rho_{i+1}^2).$$

□

6.5 Graph Recovery

By Algorithms 6.2 PairwiseGraph and 6.3 3-WiseGraph, we can construct a graph/hypergraph that is related to the support graph of the positive edges in the network. How can we find the positive edges? The problem reduces to a *graph recovery* problem, which is very similar to the problem we solved in Section 5.3.

For the pairwise correlation graph, the problem is formulated as follows:

Definition 6.18 (Graph Recovery Problem)**.** There is an unknown random bipartite graph $G_1(U, V, E_1)$ between $|U| = m$ and $|V| = n$ vertices. Each edge is in E_1 with probability d'/n.

Given: Graph $\hat{G}(V, E)$ where $(v_1, v_2) \in E$ iff v_1 and v_2 share a common parent in G_1 (i.e. $\exists u \in U$ where $(u, v_1) \in E_1$ and $(u, v_2) \in E_1$).

Goal: Find the bipartite graph G_1.

This problem is really similar to the problem we solved in Section 5.3. In fact, it is even simpler because whenever two vertices share a common parent they are connected (in Section 5.3 such pairs are only connected with probability /12). We can directly apply Algorithm 5.1 or the more complicated Algorithm 5.4 to solve this problem.

However, there is a limit in the sparsity of the graphs for these algorithms to work. In particular, they would require $d' \ll \sqrt{n}$. This is not very satisfying for our application. In this case, we can rely on the hypergraph generated by 3WiseGraph. Such higher-order correlation structure was not available for algorithms in the previous chapter.

Definition 6.19 (Graph Recovery with 3-wise Correlation)**.** There is an unknown random bipartite graph $G_1(U, V, E_1)$ between $|U| = m$ and $|V| = n$ vertices. Each edge is in E_1 with probability d'/n.

Given: Hypergraph $\hat{G}(V, E)$ where $(v_1, v_2, v_3) \in E$ iff there exists $u \in U$ where $(u, v_1), (u, v_2)$ and (u, v_3) are all in E_1.

Goal: Find the bipartite graph G_1.

Algorithm 6.6. RecoverGraph3Wise

```
input Hypergraph $\hat{G}$ in Definition 6.19
output Graph $G_1$ in Defintion 6.19
  repeat
    Pick a random hyperedge $(v_1, v_2, v_3)$ in $E$
    Let $S = \{v : (v, v_1, v_2), (v, v_1, v_3), (v, v_2, v_3) \in E\}$
    if $|S| < 1.3d'$ then
      Let $S' = \{v \in S : v$ is correlated with at least $\binom{0.8d'-1}{2}$ pairs in $S\}$
      In $G_1$, create a vertex $u$ and connect $u$ to every $v \in S'$.
      Mark all hyperedges $(v_1, v_2, v_3)$ for $v_1, v_2, v_3 \in S'$
    end if
  until all hyperedges are marked
```

The intuitions behind Algorithm 6.6 are very similar to the algorithms in previous chapter: since 3-wise correlations are rare, not many vertices should have 3-wise correlation with all three pairs (v_1, v_2), (v_1, v_3) and (v_2, v_3) unless they are all in the same community.

The performance of Algorithm 6.6 is better than the previous algorithms because 3-wise correlations are rarer.

Theorem 6.20. *When the graph G_1 satisfy properties $\mathcal{P}_{rec+}$, $\hat{G}$ is constructed according to Definition 6.19, and when $m^3 d'^8/n^8 \ll 1$, Algorithm 6.6 solves Graph Recovery with 3-wise Correlation. The expected running time is $O(m(d'^3+n))$ over the randomness of the algorithm.*

Proof. If (v_1, v_2, v_3) has more than one common neighbor, then Property 2 shows the condition in if statement must be false (as $|S| \geq |F(B(v_1) \cap B(v_2) \cap B(v_3))| \geq 1.5d'$.

When (v_1, v_2, v_3) has only one common neighbor u, then Property 1 shows $S = F(u) \cup T$ where $|T| \leq d'/20$.

Now consider S', for any $v \in F(u)$, it is correlated with all other pairs in $F(u)$. Hence it must be correlated with at least $\binom{0.8d'-1}{2}$ pairs in S, which implies v is in S.

For any $v' \notin F(u)$, by Property 3 it can only be correlated with $d'^2/40$ pairs in $F(u)$. Therefore, the total number of correlated pairs it can have in S is bounded by $|T|\,|F(u)| + \binom{|T|}{2} + d'^2/40 < \binom{0.8d'-1}{2}$. This implies v' is not in S.

The argument above shows $S' = F(u)$, so the algorithm correctly learns the edges related to u.

Finally, to bound the running time, notice that Property 4 shows the algorithm finds a new vertex u in 10 iterations in expectation. Each iteration takes at most $n + d'^3$ time. Therefore the algorithm takes $O(m(d'^3 + n))$ expected time. □

6.6 Layer with Real-valued Output

In previous sections, we considered hidden layers with sparse binary input and output. However, in most applications of deep learning, the observed vector is dense and real-valued.

Bengio et al.[20] suggested a variant auto-encoder with linear decoder, which is particularly useful in this case.

We use similar ideas as in Section 6.3 to learn the last layer. The algorithm collects the correlation-structure of the observed variables, and use this information to reconstruct the edges.

When we observe real-valued output in the last layer, we can learn whether three nodes u,v and s have a common parent from their correlation measured by $\mathbb{E}[x_u x_v x_s]$, even if the output vector is fully dense. Note that compared to Theorem 6.6 in Section 6.3, the main difference here is that we allow ρpm to be any constant (before $\rho pm \ll 1/d$).

Theorem 6.21. *When $\rho_1 d_0 = O(1)$, $d_0^2 \ll n_1$, $d_0^{3/2}/(d_1 d_2) \ll 1$ and $\sqrt{d_0}/d_1 \ll 1$, the last layer is generated at random and the rest of the network satisfy $\mathcal{P}_{mul+}$, then with high probability over the choice of the weights and the choice of the graph, for any three nodes u, v, s at the observed vector y*

1. *If u, v and s have no common neighbor, then $|\mathbb{E}_{\boldsymbol{h}}[\boldsymbol{y}_u \boldsymbol{y}_v \boldsymbol{y}_s]| \leq 0.6\rho_1$*
2. *If u, v and s have a unique common neighbor, then $|\mathbb{E}_{\boldsymbol{h}}[\boldsymbol{y}_u \boldsymbol{y}_v \boldsymbol{y}_s]| \geq 0.1\rho_1$*

Before proving the theorem we state a concentration bounds for trilinear forms:

Lemma 6.22. *If $\boldsymbol{x}$, $\boldsymbol{y}$ and $\boldsymbol{z}$ are three independent uniformly random vectors in $\{\pm 1\}^n$, with high probability*

$$\left|\sum_{i,j,k} A_{i,j,k} \boldsymbol{x}_i \boldsymbol{y}_j \boldsymbol{z}_k\right| \leq O(\sqrt{\sum_{i,j,k} A_{i,j,k}^2} \log^3 n).$$

This bound follows easily from the McDiarmid's inequality (see Appendix B) and union bound. This is not tight (for example compared to the Hanson-Wright inequality in the previous chapter) but only loses poly log factors and is good enough for our need.

Proof. In this proof we shall use $\boldsymbol{h}$ for $\boldsymbol{h}^{(1)}$ to simplify notations.

By the structure of the network we know

$$\mathbb{E}[\boldsymbol{y}_u\boldsymbol{y}_v\boldsymbol{y}_s] \;=\; \sum_{i\in B(u),j\in B(v),k\in B(s)} w_{i,u}w_{j,v}w_{k,s}\,\mathbb{E}[\boldsymbol{h}_i\boldsymbol{h}_j\boldsymbol{h}_k] \tag{6.2}$$

Observe that $w_{i,u}$ and $w_{j,v}$ are different random variables, no matter whether i is equal to j. Thus by concentration bounds for trilinear forms, with high probability over the choice of the weights

$$|\,\mathbb{E}[\boldsymbol{y}_u\boldsymbol{y}_v\boldsymbol{y}_s]| \le \sqrt{\sum_{i\in B(u),j\in B(v),k\in B(s)} \mathbb{E}[\boldsymbol{h}_i\boldsymbol{h}_j\boldsymbol{h}_k]^2\log^3 n} \tag{6.3}$$

Let V be the main term in the bound:

$$V \triangleq \sum_{i\in B(u),j\in B(v),k\in B(s)} \mathbb{E}[\boldsymbol{h}_i\boldsymbol{h}_j\boldsymbol{h}_k]^2.$$

We need to bound the size of V.

From Property $\mathcal{P}_{mul+}$ we can bound the expectations of $\boldsymbol{h}$ variables.

Proposition 6.23.

$$\mathbb{E}[\boldsymbol{h}_i\boldsymbol{h}_j\boldsymbol{h}_k] \textit{ is } \begin{cases} \le \quad O(\rho_1^3+\rho_2\rho_1+\rho_2)\log n \quad \textit{if } B^+(i)\cap B^+(j)\cap B^+(k)\neq\emptyset, \\ \le \quad O(\rho_1^3+\log n\rho_2\rho_1+\rho_3)\log n \quad \textit{if } B^+(i)\cap B^+(j)\cap B^+(k)=\emptyset, \end{cases}$$

and

$$\mathbb{E}[\boldsymbol{h}_i\boldsymbol{h}_j] \textit{ is } \begin{cases} \le \quad O(\rho_1^2+\rho_2)\log n \quad \textit{if } B^+(i)\cap B^+(j)\neq\emptyset, \\ \le \quad O(\rho_1^2+\rho_3)\log n \quad \textit{if } B^+(i)\cap B^+(j)=\emptyset, \end{cases}$$

When $d_0^2 \ll n_1$, with high probability we can also bound the intersection of neighbors $|B(u)\cap B(v)|\le \log n$ for any two u, v in the observed layer.

Combining all these, we can carefully bound the size of V when u, v, s do not share a common neighbor. In particular $\sqrt{V} \le \sqrt{v_1 + v_2} \le O(d^{3/2}(\rho_1^3 + \rho_2\rho_1 + \rho_3) + 10d^{1/2}(\rho_1^2 + \rho_2)) \operatorname{poly}\log n$.

Under the assumption of theorem, $\sqrt{V \log^3 n} \le \rho_1/3$. With high probability, when u, v, s do not share a common neighbor, we know $|\mathbb{E}[\boldsymbol{y}_u\boldsymbol{y}_v\boldsymbol{y}_s]| \le \rho_1/10$.

When u, v, s share a unique neighbor z, we take the term $\mathbb{E}[\boldsymbol{h}_z^3]w_{z,u}w_{z,v}w_{z,s}$ out of the sum. The remaining terms can still be bounded by $\rho_1/3$. Hence $|\mathbb{E}[\boldsymbol{y}_u\boldsymbol{y}_v\boldsymbol{y}_s]| \ge \mathbb{E}[\boldsymbol{h}_z^3] - \rho_1/10 \ge 0.6\rho_1$ (we argue that $\mathbb{E}[\boldsymbol{h}_z] \ge 0.7\rho$ by going back to the ℓ-th layer as before).

□

Although the hypergraph generated by Theorem 6.21 is not exactly the same as the hypergraph required in Definiton 6.19, the same Algorithm 6.6 can find the edges of the graph. This is because the two hypergraphs only differ when u, v, s share more than one common neighbor, which happens with extremely low probability (Algorithm 6.6 can actually tolerate this effect even if the probability is a small constant).

Note that different from the previous sections, here Algorithm 6.3 actually returns a graph that contains both the positive edges and negative edges. In order to reconstruct the weighted bipartite graph we need to differentiate between positive and negative edges. This is not hard: from the proof of Theorem 6.21, when u, v, s share a unique neighbor x, $\mathbb{E}_{\boldsymbol{h}}[\boldsymbol{y}_u\boldsymbol{y}_v\boldsymbol{y}_s] > 2\rho/3$ if an even number of edges (x, u), (x, v), (x, s) are negative. Therefore Algorithm 6.7 can distinguish positive and negative edges.

To prove the algorithm works we first consider the case when any triple share at most one parent. In that case all nodes in S will have edges with the same sign to x, therefore the sign of t is exactly the sign of $\mathbb{E}_{\boldsymbol{h}}[\boldsymbol{y}_t\boldsymbol{y}_{u'}\boldsymbol{y}_{v'}]$ for any $u', v' \in S$. The algorithm is extremely noise tolerant, so it still works even if there is a small number of triples that share more than one parent (and the test $\mathbb{E}_{\boldsymbol{h}}[\boldsymbol{y}_u\boldsymbol{y}_v\boldsymbol{y}_s] > 0$ is not perfect).

Algorithm 6.7. LearnLastLayer

```
input $E$, the set of edges
output $E^+$, $E^-$, the set of positive and negative edges
  for each node $x \in \boldsymbol{h^{(1)}}$ do
    Pick $u, v$ in $F(x)$
    Let $S$ be the set $\{s : \mathbb{E}_{\boldsymbol{h}}[\boldsymbol{y}_u \boldsymbol{y}_v \boldsymbol{y}_s] > 0\}$.
    for each $t : (x, t) \in E$ do
      if most triples $(t, u', v')(u', v' \in S)$ have positive $\mathbb{E}_{\boldsymbol{h}}[\boldsymbol{y}_t \boldsymbol{y}_{u'} \boldsymbol{y}_{v'}]$ then
        add $(x, t)$ to $E^+$
      else
        add $(x, t)$ to $E^-$
      end if
    end for
  end for
```

6.7 Lower Bound

In this section we show that a two-layer network with ±1 weights is more expressive than one layer network with arbitrary weights. A two-layer network (G_1, G_2) consists of random graphs G_1 and G_2 with random ±1 weights on the edges. Viewed as a generative model, its input is $\boldsymbol{h^{(3)}}$ and the output is $\boldsymbol{h^{(1)}} = \operatorname{sgn}(G_1 \operatorname{sgn}(G_2 \boldsymbol{h^{(3)}}))$. We will show that a single-layer network even with arbitrary weights and arbitrary threshold functions must generate a fairly different distribution.

Lemma 6.24. *For almost all choices of (G_1, G_2), the following is true. For every one layer network with matrix A and vector $\boldsymbol{b}$, if $\boldsymbol{h^{(3)}}$ is chosen to be a random $\rho_3 n$-sparse vector with $\rho_3 d_2 d_1 \ll 1$, the probability (over the choice of $\boldsymbol{h^{(3)}}$) is at least $\Omega(\rho_3^2)$ that* $\operatorname{sgn}(G_1 \operatorname{sgn}(G_1 \boldsymbol{h^{(3)}})) \neq \operatorname{sgn}(A\boldsymbol{h^{(3)}} + \boldsymbol{b})$.

The idea is that the cancellations possible in the two-layer network simply cannot all be accomodated in a single-layer network even using arbitrary weights. More precisely, even the bit at a single output node v cannot be well-represented by a simple threshold function.

First, observe that the output at v is determined by values of $d_1 d_2$ nodes at the top layer that are its ancestors. Wlog, in the one layer net $(A, \boldsymbol{b})$, there should be no edge between v and any node u that is not its ancestor. The reason is that these edges between v and its ancestors in $(A, \boldsymbol{b})$ can be viewed collectively as a single random variables that is not correlated with values at v's ancestors, and either these edges are "influential" with probability at least $\rho_3^2/4$ in which case it causes a wrong bit at v; or else it is not influential and removing it will not change the function computed on $\rho_3^2/4$ fraction of probability mass. Similarly, if there is a path from u to v then there must be a corresponding edge in the one-layer network. The question is what weight it should have, and we show that no weight assignment can avoid producing erroneous answers.

The reason is that with probability at least $\rho_3/2$, among all ancestors of v in the input layer, only u is 1. Thus in order to produce the same output in all these cases, in the one-layer net the edge between u and v should be positive iff the path from u to v consists of two positive edges. But now we show that with high probability there is a cancellation effect involving a local structure in the two layer net whose effect cannot be duplicated by such a single-layer net (See the Figure 6.3 and 6.4).

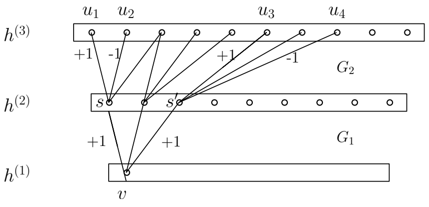


Figure 6.3: Two-layer network(G_1, G_2)

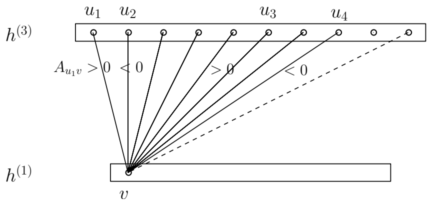


Figure 6.4: Single-layer network $(A, \boldsymbol{b})$

As drawn in Figure 6.3, suppose the nodes u_1, u_2 connect to s in $\boldsymbol{h^{(2)}}$ via $+1$ and -1 edge, and s connects to v via a $+1$ edge. Similarly, the nodes u_3, u_4 connect to s' in $\boldsymbol{h^{(2)}}$ via $+1$ and -1 edge, and s connects to v via a $+1$ edge.

Now assume all other ancestors of v are off, and consider the following four values of (u_1, u_2, u_3, u_4): $(1, 1, 0, 0)$, $(0, 0, 1, 1)$, $(1, 0, 0, 1)$, $(0, 1, 1, 0)$. In the two-layer network, $\boldsymbol{h_v^{(1)}}$ should be 0 for the first two inputs and 1 for the last two inputs. Now we are going to see the contradiction. For single-layer network, these values imply constraints $A_{u_1,v} + A_{u_2,v} + \boldsymbol{b}_v \leq 0$,

$A_{u_3,v} + A_{u_4,v} + \boldsymbol{b}_v \leq 0$, $A_{u_1,v} + A_{u_4,v} + \boldsymbol{b}_v > 0$, $A_{u_2,v} + A_{u_3,v} + \boldsymbol{b}_v > 0$. However, there can be no choices of $(A, \boldsymbol{b})$ that satisfies all four constraints! To see that, simply add the first two and the last two, the left-hand-sides are all $\sum_{i=1}^4 A_{u_i,v} + 2\boldsymbol{b}_v$, but the right-hand-sides are ≤ 0 and > 0. The single-layer network cannot agree on all four inputs. Each of the four inputs occurs with probability at least $\Omega(\rho_3^2)$. Therefore the outputs of two networks must disagree with probability $\Omega(\rho_3^2)$.

Remark: It is well-known in complexity theory that such simple arguments do not suffice to prove lowerbounds on neural nets with more than one layer.

6.8 Random Graph Properties

In this Section we state the properties of random graphs that are used by the algorithm. We first describe the unique-neighbor property, which is central to our analysis. In the next part we list the properties required by different steps of the algorithm.

6.8.1 Unique neighbor property

Recall that in bipartite graph $G(U, V, E, w)$, the set $F(u)$ denotes the neighbors of $u \in U$, and the set $B(v)$ denotes the neighbors of $v \in V$.

For any node $u \in U$ and any subset $S \subset U$, let $UF(u, S)$ be the sets of unique neighbors of u with respect to S,

$$UF(u, S) \triangleq \{v \in V : v \in F(u), v \notin F(S \setminus \{u\})\}$$

Definition 6.25. In a bipartite graph $G(U, V, E, w)$, a node $u \in U$ has $(1-\epsilon)$-unique neighbor property with respect to S if $|UF(u, S)| \geq (1-\epsilon)\,|F(u)|$.

The set S has $(1-\epsilon)$-strong unique neighbor property if for every $u \in U$, u has $(1-\epsilon)$-unique neighbor property with respect to S.

Remark 6.26. Notice that in our definition, u does not need to be inside S. This is not much stronger than the usual definition where u has to be in S: if u is not in S we are simply saying u has unique neighbor property with respect to $S \cup \{u\}$. When all (or most) sets of size $|S| + 1$ have the "usual" unique neighbor properties, all (or most) sets of size $|S|$ have the unique neighbor property according to our definition.

Lemma 6.27. *If $G(U, V, E)$ is from distribution $\mathcal{G}_{m,n,p}$, for every subset S of U with $(1 - p)^{|S|} > 1 - \epsilon/2$ (note that $p|S| = d'|S|/n$ is the expected density of $F(S)$), with probability $1 - 2m\exp(-\epsilon^2 d')$ over the randomness of the graph, S has the $(1-\epsilon)$-strong unique neighbor property.*

Proof. Fix the vertex u, first sample the edges incident to u. Without loss of generality assume u has neighbors $v_1, \dots, v_k$ (where $k \approx pn$). Now sample the edges incident to the v_i's. For each v_i, with probability $(1-p)^{|S|} \geq 1 - \epsilon/2$, v_i is a unique neighbor of u. Call this event $Good_i$ (and we also use $Good_i$ as the indicator for this event).

By the construction of the graph we know $Good_i$'s are independent, hence by Chernoff bounds, we have that with probability $1 - 2\exp(-\epsilon^2 d')$ the node u has unique neighbor property with respect to S.

By union bound, every u satisfies this property with probability at least $1 - 2m\exp(-\epsilon^2 C)$. □

6.8.2 Properties required by each steps

We now list the properties required in our analysis. These properties hold with high probability for random graphs.

Properties for Correlation Graph: Single-layer

The algorithm PairwisGraph requires the following properties $\mathcal{P}_{sing}$

1. For any u in the observed layer, $|B(u)| \in [0.9d, 1.1d]$, (if G has negative weights, we also need $|B^+(u)| \in [0.9d/2, 1.1d/2]$)

2. For any z in the hidden layer, $|F(z)| \in [0.9d', 1.1d']$, (if G has negative weights, we also need $|F^+(z)| \in [0.45d', 0.55d']$)

The algorithm 3-WiseGraph needs $\mathcal{P}_{sing}$, and additionally $\mathcal{P}_{sing+}$

1. For any u, v in the observed layer, $|B^+(u) \cup B^+(v)| \leq 10 \log n$.

Lemma 6.28. *If graph G is chosen from $\mathcal{G}_{m,n,p}$ with expected degrees $d, d' \gg \log n$, with high probability over the choice of the graph, $\mathcal{P}_{sing}$ is satisfied. If in addition $d^2 \ll n$, the property $\mathcal{P}_{sing+}$ is also satisfied with high probability.*

The lemma is a straight-forward application of concentration bounds.

Properties for Correlation Graph: Multi-layer

For the multi-layer setting, the algorithm PairwisGraph requires the following expansion properties $\mathcal{P}_{mul}$.

1. For any node u at the layer i, $|F_{i-1}(u)| \in [0.9d'_{i-1}, 1.1d'_{i-1}]$, $|B_i(u)| \in [0.9d_i, 1.1d_i]$, $\left|F^+_{i-1}(u)\right| \in [0.45d'_{i-1}, 0.55d'_{i-1}]$, $\left|B^+_i(u)\right| \in [0.45d_i, 0.55d_i]$

2. For any node u at the layer i, $|B^{(t)}_+(u)| \geq= 0.8\rho_i/\rho_t$, and $|B^{(\ell)}(u)| \leq 2^{\ell-i+1}\rho_i/\rho_\ell$.

3. For any pair of nodes u, v at layer i,

$$\left|B^{(\ell)}_+(u) \cap B^{(\ell)}_+(v)\right| \leq O\left(\rho_{i+1}/\rho_\ell \cdot \left(1/d_{i+1} + \log n/(\rho_\ell n_\ell) + \left|B^{(i+1)}_+(u) \cap B^{(i+1)}_+(v)\right|\right)\right)$$

   In particular, if u and v have no common positive parent at layer $i+1$ ($\left|B^{(i+1)}_+(u) \cap B^{(i+1)}_+(v)\right| = 0$), then

$$\left|B^{(\ell)}_+(u) \cap B^{(\ell)}_+(v)\right| \leq o(\rho_{i+1}/\rho_\ell)$$

The algorithm 3-WiseGraph needs the additional property $\mathcal{P}_{mul+}$:

1. For any pair of nodes u, v at layer i, $\left|B_+^{(\ell)}(u) \cap B_+^{(\ell)}(v)\right| \le O(\rho_{i+1} \log n/\rho_\ell)$
2. For any three nodes u, v and s at layer i, if they don't have a common positive neighbor in layer $i+1$,

$$\begin{aligned} & \left|B_+^{(\ell)}(u) \cap B_+^{(\ell)}(v) \cap B_+^{(\ell)}(s)\right| \\ \le \;& O\left(\rho_{i+1} \log n/\rho_\ell \cdot \left(1/d_{i+1} + \rho_\ell/(\rho_\ell n_\ell) + 1/(\rho_\ell^2 n_\ell^2)\right)\right) \le o(\rho_{i+1}/\rho_\ell) \end{aligned}$$

Property 1 can be relaxed but we choose to present this simpler condition.

Lemma 6.29. *If the network $\mathcal{D}(\ell, \rho_\ell, \{G_i\})$ have parameters satisfying $d_i \gg \log n$, and $\rho_i^2 \ll \rho_{i+1}$, then with high probability over the randomness of the graphs, $\{G_i\}$'s satisfy $\mathcal{P}_{mul}$. Additionally, if $g_i \gg \log n$ and $\rho_i^3 \ll \rho_{i+1}$, then $\{G_i\}$'s satisfy $\mathcal{P}_{mul+}$ with high probability.*

In order to prove Lemma 6.29 we need the following claim:

Claim. *If the graph $G \sim \mathcal{G}_{m,n,p}$ with $d = pm$ being the expected back degree, and $d \gg \log n$. For two arbitrary sets T_1 and T_2, with $d|T_1| \ll m, d|T_2| \ll m$, we have with high probability*

$$|B(T_1) \cap B(T_2)| \le (1+\epsilon)d|T_1 \cap T_2| + (1+\epsilon)d^2|T_1||T_2|/m + 5\log n$$

This Claim simply follows from simple concentration bounds. Now we are ready to prove Lemma 6.29.

Proof of Lemma 6.29. Property 1 in $\mathcal{P}_{mul}$ follows directly from $d_i \gg \log n$.

Property 2 in $\mathcal{P}_{mul}$ follows from unique neighbor properties (when we view the bipartite graph from N_i to N_{i+1}).

For Property 3, we prove the following proposition by induction on t:

Proposition 6.30. *For any two nodes u, v at layer 1,*

$$\left|B_+^{(t)}(u)\cap B_+^{(t)}(v)\right| \le (1+\epsilon)^t t\rho_1^2/(\rho_t^2 n_t)+6t(1+\epsilon)^t\rho_3\log n/\rho_t+(1+\epsilon)^t\rho_2\left|B_+^{(2)}(u)\cap B^{(2)}(v)\right|/\rho_t$$

This is true for $t=2$ (simply because of the third term). Suppose it is true for all the values at most t. When we are considering $t+1$-th level, by Claim 6.8.2, we know with high probability (notice that we only need to do union bound on n^2 pairs)

$$\begin{aligned}
\left|B_+^{(t+1)}(u)\cap B_+^{(t+1)}(v)\right| &\le (1+\epsilon)d_t/2\cdot\rho_2\left|B_+^{(t)}(u)\cap B_+^{(t)}(v)\right|/\rho_t \\
&\quad +(1+\epsilon)d_t^2/4\cdot|B_+^{(t)}(u)||B_+^{(t)}(v)|+5\log n \\
&\le (1+\epsilon)^{t+1}\rho_2\left|B_+^{(2)}(u)\cap B^{(2)}(v)\right|/\rho_{t+1}+6t(1+\epsilon)^{t+1}\rho_3\log n/\rho_{t+1} \\
&\quad +(1+\epsilon)^t t\rho_1^2/(\rho_t^2 n_t)+(1+\epsilon)d_t^2/4\cdot\rho_1^2/(\rho_t^2 n_{t+1})+5\log \\
&\le (1+\epsilon)^{t+1}\rho_2\left|B_+^{(2)}(u)\cap B^{(2)}(v)\right|/\rho_{t+1}+6(t+1)(1+\epsilon)^{t+1}\rho_3\log n/\rho_{t+1} \\
&\quad +2(1+\epsilon)^{t+1}(t+1)\rho_1^2/(\rho_{t+1}^2 n_{t+1}),
\end{aligned}$$

where the last inequality uses the fact that $\rho_1^2/(\rho_t^2 n_t)\le d_t^2/4\cdot\rho_1^2/(\rho_t^2 n_{t+1})$. This is because $n_t d_t/n_{t+1}=d'_t\gg 1$.

Proposition 6.30 implies that when $\rho_1^2\ll\rho_2$, and ℓ is a constant,

$$\left|B_+^{(t)}(u)\cap B_+^{(t)}(v)\right|\le O\left(\rho_{i+1}/\rho_\ell\cdot\left(1/d_{i+1}+\log n/(\rho_\ell n_\ell)+\left|B_+^{(i+1)}(u)\cap B_+^{(i+1)}(v)\right|\right)\right)$$

Property 2 in $\mathcal{P}_{mul+}$ is similar but more complicated. □

Properties for Graph Reovery

For the algorithm RecoverGraph3Wise to work, the hypergraph generated from the random graph should have the following properties $\mathcal{P}_{rec+}$.

1. For any $(v_1, v_2, v_3) \in E$, if S is the set defined as in the algorithm, then $|S \backslash F(B(v_1) \cap B(v_2) \cap B(v_3))| < d'/20$.

2. For any $u_1, u_2 \in U$, $|F(u_1) \cup F(u_2)| > 1.5d'$.

3. For any $u \in U$, $v \in V$ and $v \notin F(u)$, v is correlated with at most $d'^2/40$ pairs in $F(u)$.

4. For any $u \in U$, at least 0.1 fraction of triples $v_1, v_2, v_3 \in F(u)$ does not have a common neighbor other than u.

5. For any $u \in U$, its degree $d_u \in [0.9d', 1.1d']$.

Lemma 6.31. *When $m^3 d'^8/n^8 \ll 1$, $d' \gg 1$, with high probability a random graph satisfies Property $\mathcal{P}_{rec+}$.*

Proof. Property 1: Fix any v_1, v_2 and v_3, Consider the graph G_1 to be sampled in the following order. First sample the edges related to v_1, v_2 and v_3, then sample the rest of the graph.

At step 2, let $S_1 = B(v_2) \cap B(v_3) \backslash B(v_1)$, $S_2 = B(v_1) \cap B(v_3) \backslash B(v_2)$ and $S_3 = B(v_1) \cap B(v_2) \backslash B(v_3)$. By the construction of the graph $\hat{G}$, every vertex in S must be in $F(S_1) \cap F(S_2) \cap F(S_3)$. With high probability ($e^{-\Omega(d')}$) we know $|S_1| \le 5m(d'/n)^2$ (this is by Chernoff bound, because each vertex u is connected to two vertices v_2, v_3 with probability $(d'/n)^2$). Similar things hold for S_2, S_3.

Now $F(S_1)$, $F(S_2)$ and $F(S_3)$ are three random sets of size at most $10m(d')^3/n^2$, thus again by Chernoff bound we know their intersection has size smaller than $10(10m(d')^3/n^2)^3/n^2$, which is smaller than $d'/20$ by assumption.

Property 2: This follows directly from the unique neighbor property of sets of size 2.

Property 3: First sample the edges related to u and v, then sample edges related to vertices in $B(v)$. For any vertex in $B(v)$, the expected number of pairs is $(d'^2/n)^2$. Since $|B(v)| \le md'/n$ with high probability, again by Chernoff bound we know the number of pairs is smaller than $O((d'^2/n)^2 md'/n) = O(md'^5/n^3)$. This is much smaller than $d'^2/40$ under assumption (notice that $(md'^3)/n^3 = ((m^3d'^8/n^8)(d'/n))^{1/3} \ll 1$). Property 4: Again change the sampling process: first sample all the edges not related to u, then sample 1/2 of the edges connecting to u, and finally sample the second half.

Let S_1 be the first half of $F(u)$. For a vertex outside S_1, similar to Property 3 we know every $v \notin S_1$ has at most $d'/40$ neighboring pairs in S_1, therefore any new sample in the second half is going to introduce many new triples of correlation. The total number of new correlations is at least 0.1 fraction.

Property 5: This simply follows from $d' \gg 1$. □

Properties for Partial Encoder

The partial encoder only relies on the strong unique-neighbor property.

Properties for Learning -1 Edges

In order to learn the -1 edges, we need Property $\mathcal{P}_{neg+}$. It includes some properties in $\mathcal{P}_{mul}$, and additionally the property that nodes in i-th layer cannot be too negatively correlated, in particular

1. 1 and 2 in $\mathcal{P}_{mul}$.

2. For fixed node u and set S at layer i, if $|S| \le 2d_{i-1}$, then with high probability $\left|B_+^{(\ell)}(u) \setminus B^{(\ell)}(S)\right| \ge \rho_i/2\rho_\ell$.

The proof of second property is very similar to Lemma 6.29.

Part III

Tensor Decomposition for General Matrix Factorization

Chapter 7

Orthogonal Tensor Decomposition

In this chapter we discuss Orthogonal Tensor Decomposition, which is a unified methodology for solving General Matrix Factorization problems (or learning latent variable models in general). This methodology provides computationally and statistically efficient parameter estimation methods for a wide class of latent variable models—including simple topic models, independent component analysis and hidden Markov models—which exploits a certain tensor structure in their low-order observable moments. Specifically, parameter estimation is reduced to the problem of extracting a certain (orthogonal) decomposition of a symmetric tensor derived from the moments; this decomposition can be viewed as a natural generalization of the singular value decomposition for matrices.

Although tensor decompositions are generally intractable to compute, the decomposition of these specially structured tensors can be efficiently obtained by a variety of approaches, including power iterations and maximization approaches (similar to the case of matrices). We provide a detailed analysis of a robust tensor power method, establishing an analogue of Wedin's perturbation theorem for the singular vectors of matrices. This implies a robust and computationally tractable estimation approach for several popular latent variable models.

7.1 Orthogonal Tensors

We call the decomposition

$$T = \sum_{i=1}^{k} \lambda_i \boldsymbol{v_i}^{\otimes 3} \tag{7.1}$$

an orthogonal decomposition of tensor T if all the vectors $\boldsymbol{v_i}$'s are orthonormal. And we call the tensor T orthogonally decomposable. Notice that this is similar to the singular value decomposition (SVD) of symmetric matrices: $M = \sum_{i=1}^{k} \sigma_i \boldsymbol{v_i} \boldsymbol{v_i}^T$. All symmetric matrices can be decomposed into orthogonal components, but only very special tensors are orthogonally decomposable.

Given a tensor $\hat{T}$ which is close to an orthogonally decomposable tensor T

$$\hat{T} \approx T = \sum_{i=1}^{k} \lambda_i \boldsymbol{v_i}^{\otimes 3},$$

the goal of orthogonal tensor decomposition is to find $\hat{\lambda}_i$'s and $\boldsymbol{\hat{v}_i}$'s that are close to λ_i's and $\boldsymbol{v_i}$'s.

Our main result is

Theorem 7.1. *Let $\hat{T} = T + E \in \mathbb{R}^{k \times k \times k}$, where T is a symmetric tensor with orthogonal decomposition $T = \sum_{i=1}^{k} \lambda_i \boldsymbol{v_i}^{\otimes 3}$ where each $\lambda_i > 0$, $\{\boldsymbol{v_1}, \boldsymbol{v_2}, \ldots, \boldsymbol{v_k}\}$ is an orthonormal basis, and E has operator norm $\epsilon := \|E\|$. Define $\lambda_{\min} := \min\{\lambda_i : i \in [k]\}$, and $\lambda_{\max} := \max\{\lambda_i : i \in [k]\}$.*

There exists universal constant C such that when $\epsilon \le C \frac{\lambda_{\min}}{k}$, there is an algorithm that returns $(\boldsymbol{\hat{v}_1}, \hat{\lambda}_1), (\boldsymbol{\hat{v}_2}, \hat{\lambda}_2), \ldots, (\boldsymbol{\hat{v}_k}, \hat{\lambda}_k)$. With high probability, there exists a permutation π on $[k]$ such that

$$\|\boldsymbol{v_{\pi(j)}} - \boldsymbol{\hat{v}_j}\| \le 8\epsilon/\lambda_{\pi(j)}, \qquad |\lambda_{\pi(j)} - \hat{\lambda}_j| \le 5\epsilon, \quad \forall j \in [k],$$

and

$$\left\| T - \sum_{j=1}^{k} \hat{\lambda}_j \hat{\boldsymbol{v}}_{\boldsymbol{j}}^{\otimes 3} \right\| \leq 55\epsilon.$$

The orthogonal constraint may seem very unnatural, as the parameters of models do not usually have such strong structure. However in Section 7.2 we show that a more general case reduces to orthogonal tensor decomposition via commonly applied *whitening*.

In Section 7.3 we give a few examples to show how this problem is crucial in learning many different latent variable models. In Section 7.4 we present the tensor power method. This is followed by discussions in Section 7.5, and the proof of the main theorem appears in Section 7.6.

7.2 Whitening: Getting Orthogonality from General Vectors

We now demonstrate how orthogonality can be enforced using the whitening operation.For concreteness, we take the following specific form, which appears in the exchangeable single topic model (Theorem 7.2):

$$\begin{aligned} M_2 &= \sum_{i=1}^{k} w_i \; \boldsymbol{\mu_i} \otimes \boldsymbol{\mu_i}, \\ M_3 &= \sum_{i=1}^{k} w_i \; \boldsymbol{\mu_i} \otimes \boldsymbol{\mu_i} \otimes \boldsymbol{\mu_i}. \end{aligned}$$

(The more general case allows the weights w_i in M_2 to differ in M_3, but for simplicity we keep them the same in the following discussion.) We now show how to reduce these forms to an orthogonally decomposable tensor from which the w_i and $\boldsymbol{\mu_i}$ can be recovered.

Throughout, we assume the following non-degeneracy condition.

Condition 7.2.1 (Non-degeneracy). *The vectors $\boldsymbol{\mu_1}, \boldsymbol{\mu_2}, \ldots, \boldsymbol{\mu_k} \in \mathbb{R}^d$ are linearly independent, and the scalars $w_1, w_2, \ldots, w_k > 0$ are strictly positive.*

Observe that Condition 7.2.1 implies that $M_2 \succeq 0$ is positive semidefinite and has rank-k. This is a mild condition; furthermore, when this condition is not met, learning is conjectured to be hard for both computational [130] and information-theoretic reasons [128].

7.2.1 The reduction

First, let $W \in \mathbb{R}^{d \times k}$ be a linear transformation such that

$$M_2(W, W) \;=\; W^\top M_2 W \;=\; I$$

where I is the $k \times k$ identity matrix (*i.e.*, W whitens M_2). Since $M_2 \succeq 0$, we may for concreteness take $W := UD^{-1/2}$, where $U \in \mathbb{R}^{d \times k}$ is the matrix of orthonormal eigenvectors of M_2, and $D \in \mathbb{R}^{k \times k}$ is the diagonal matrix of positive eigenvalues of M_2. Let

$$\tilde{\boldsymbol{\mu}}_{\boldsymbol{i}} := \sqrt{w_i}\; W^\top \boldsymbol{\mu_i}.$$

Observe that

$$M_2(W, W) \;=\; \sum_{i=1}^{k} W^\top (\sqrt{w_i} \boldsymbol{\mu_i})(\sqrt{w_i} \boldsymbol{\mu_i})^\top W \;=\; \sum_{i=1}^{k} \tilde{\boldsymbol{\mu}}_{\boldsymbol{i}} \tilde{\boldsymbol{\mu}}_{\boldsymbol{i}}^\top \;=\; I,$$

so the $\tilde{\boldsymbol{\mu}}_{\boldsymbol{i}} \in \mathbb{R}^k$ are orthonormal vectors.

Now define $\widetilde{M_3} := M_3(W, W, W) \in \mathbb{R}^{k \times k \times k}$, so that

$$\widetilde{M_3} \;=\; \sum_{i=1}^{k} w_i \; (W^\top \boldsymbol{\mu_i})^{\otimes 3} \;=\; \sum_{i=1}^{k} \frac{1}{\sqrt{w_i}} \; \tilde{\boldsymbol{\mu}}_{\boldsymbol{i}}^{\otimes 3}.$$

When applying this reduction to learning latent variable models, it is important to bound the error of the orthogonal tensor. Loose bounds follow from McDiarmid's inequality (see Appendix B). Tighter bounds can be obtained by matrix bernstein inequalities, see Chapter 8 for an example.

7.3 Tensor structure in latent variable models

In this section, we give several examples of latent variable models whose low-order moments can be written as symmetric tensors of low symmetric rank. This form is demonstrated in Theorem 7.2 for the first example. The general pattern will emerge from subsequent examples.

7.3.1 Exchangeable single topic models

We first consider a simple bag-of-words model for documents in which the words in the document are assumed to be *exchangeable*. Recall that a collection of random variables $\boldsymbol{x}_1, \boldsymbol{x_2}, \ldots, \boldsymbol{x}_\ell$ are exchangeable if their joint probability distribution is invariant to permutation of the indices. The well-known De Finetti's theorem [15] implies that such exchangeable models can be viewed as mixture models in which there is a latent variable h such that $\boldsymbol{x}_1, \boldsymbol{x_2}, \ldots, \boldsymbol{x}_\ell$ are conditionally i.i.d. given h (see Figure 7.1(a) for the corresponding graphical model) and the conditional distributions are identical at all the nodes.

In our simplified topic model for documents, the latent variable h is interpreted as the (sole) topic of a given document, and it is assumed to take only a finite number of distinct values. Let k be the number of distinct topics in the corpus, d be the number of distinct words in the vocabulary, and $\ell \geq 3$ be the number of words in each document. The generative process for a document is as follows: the document's topic is drawn according to the discrete distribution specified by the probability vector $w := (w_1, w_2, \ldots, w_k) \in \Delta^{k-1}$. This is modeled as a discrete random variable h such that

$$\Pr[h = j] = w_j, \quad j \in [k].$$

Given the topic h, the document's ℓ words are drawn independently according to the discrete distribution specified by the probability vector $\mu_h \in \Delta^{d-1}$. It will be convenient to represent the ℓ words in the document by d-dimensional random *vectors* $\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_\ell} \in \mathbb{R}^d$. Specifically, we set

$$\boldsymbol{x_t} = \boldsymbol{e_i} \quad \text{if and only if} \quad \text{the } t\text{-th word in the document is } i, \quad t \in [\ell],$$

where $\boldsymbol{e_1}, \boldsymbol{e_2}, \ldots \boldsymbol{e_d}$ is the standard coordinate basis for $\mathbb{R}^d$.

One advantage of this encoding of words is that the (cross) moments of these random vectors correspond to joint probabilities over words. For instance, observe that

$$\begin{aligned}\mathbb{E}[\boldsymbol{x_1} \otimes \boldsymbol{x_2}] &= \sum_{1 \le i,j \le d} \Pr[\boldsymbol{x_1} = \boldsymbol{e_i}, \boldsymbol{x_2} = \boldsymbol{e_j}]\ \boldsymbol{e_i} \otimes \boldsymbol{e_j} \\ &= \sum_{1 \le i,j \le d} \Pr[\text{1st word} = i, \text{2nd word} = j]\ \boldsymbol{e_i} \otimes \boldsymbol{e_j},\end{aligned}$$

so the (i, j)-the entry of the matrix $\mathbb{E}[\boldsymbol{x_1} \otimes \boldsymbol{x_2}]$ is $\Pr[\text{1st word} = i, \text{2nd word} = j]$. More generally, the $(i_1, i_2, \ldots, i_\ell)$-th entry in the tensor $\mathbb{E}[\boldsymbol{x_1} \otimes \boldsymbol{x_2} \otimes \cdots \otimes \boldsymbol{x_\ell}]$ is $\Pr[\text{1st word} = i_1, \text{2nd word} = i_2, \ldots, \ell\text{-th word} = i_\ell]$. This means that estimating cross moments, say, of $\boldsymbol{x_1} \otimes \boldsymbol{x_2} \otimes \boldsymbol{x_3}$, is the same as estimating joint probabilities of the first three words over all documents. (Recall that we assume that each document has at least three words.)

The second advantage of the vector encoding of words is that the conditional expectation of $\boldsymbol{x_t}$ given $h = j$ is simply $\boldsymbol{\mu_j}$, the vector of word probabilities for topic j:

$$\mathbb{E}[\boldsymbol{x_t}|h = j] \;=\; \sum_{i=1}^{d} \Pr[t\text{-th word} = i|h = j]\; e_i \;=\; \sum_{i=1}^{d} [\boldsymbol{\mu_j}]_i \; \boldsymbol{e_i} \;=\; \boldsymbol{\mu_j}, \quad j \in [k]$$

(where $[\boldsymbol{\mu_j}]_i$ is the i-th entry in the vector $\boldsymbol{\mu_j}$). Because the words are conditionally independent given the topic, we can use this same property with conditional cross moments, say, of $\boldsymbol{x_1}$ and $\boldsymbol{x_2}$:

$$\mathbb{E}[\boldsymbol{x_1} \otimes \boldsymbol{x_2}|h = j] \;=\; \mathbb{E}[\boldsymbol{x_1}|h = j] \otimes \mathbb{E}[\boldsymbol{x_2}|h = j] \;=\; \boldsymbol{\mu_j} \otimes \boldsymbol{\mu_j}, \quad j \in [k].$$

This and similar calculations lead one to the following theorem.

Theorem 7.2 ([8]). *If*

$$\begin{aligned} M_2 &:= \mathbb{E}[\boldsymbol{x_1} \otimes \boldsymbol{x_2}] \\ M_3 &:= \mathbb{E}[\boldsymbol{x_1} \otimes \boldsymbol{x_2} \otimes \boldsymbol{x_3}], \end{aligned}$$

then

$$\begin{aligned} M_2 &= \sum_{i=1}^{k} w_i \; \boldsymbol{\mu_i} \otimes \boldsymbol{\mu_i} \\ M_3 &= \sum_{i=1}^{k} w_i \; \boldsymbol{\mu_i} \otimes \boldsymbol{\mu_i} \otimes \boldsymbol{\mu_i}. \end{aligned}$$

As we will see in Section 7.2, the structure of M_2 and M_3 revealed in Theorem 7.2 implies that the topic vectors $\boldsymbol{\mu_1}, \boldsymbol{\mu_2}, \ldots, \boldsymbol{\mu_k}$ can be estimated by computing a certain symmetric tensor decomposition. Moreover, due to exchangeability, all triples (resp., pairs) of words in a document—and not just the first three (resp., two) words—can be used in forming M_3 (resp., M_2).

7.3.2 Beyond raw moments

In the single topic model above, the raw (cross) moments of the observed words directly yield the desired symmetric tensor structure. In some other models, the raw moments do not explicitly have this form. Here, we show that the desired tensor structure can be found through various manipulations of different moments.

Independent component analysis (ICA)

The standard model for ICA [37, 42, 44, 91], in which independent signals are linearly mixed and corrupted with Gaussian noise before being observed, is specified as follows. Let $\boldsymbol{h} \in \mathbb{R}^k$ be a latent random *vector* with independent coordinates, $A \in \mathbb{R}^{d \times k}$ the mixing matrix, and $\boldsymbol{z}$ be a multivariate Gaussian random vector. The random vectors $\boldsymbol{h}$ and $\boldsymbol{z}$ are assumed to be independent. The observed random vector is

$$\boldsymbol{x} := A\boldsymbol{h} + \boldsymbol{z}.$$

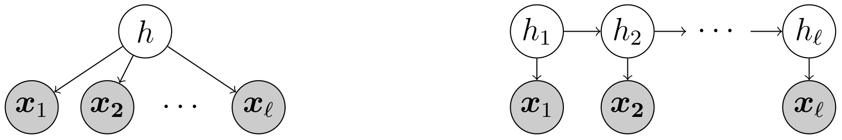
Let $\boldsymbol{\mu_i}$ denote the i-th column of the mixing matrix A.

Theorem 7.3 ([44]). *Define*

$$M_4 \quad := \quad \mathbb{E}[\boldsymbol{x} \otimes \boldsymbol{x} \otimes \boldsymbol{x} \otimes \boldsymbol{x}] - T$$

where T is the fourth-order tensor with

$$[T]_{i_1,i_2,i_3,i_4} := \mathbb{E}[\boldsymbol{x}_{i_1}\boldsymbol{x}_{i_2}]\mathbb{E}[\boldsymbol{x}_{i_3}\boldsymbol{x}_{i_4}] + \mathbb{E}[\boldsymbol{x}_{i_1}\boldsymbol{x}_{i_3}]\mathbb{E}[\boldsymbol{x}_{i_2}\boldsymbol{x}_{i_4}] + \mathbb{E}[\boldsymbol{x}_{i_1}\boldsymbol{x}_{i_4}]\mathbb{E}[\boldsymbol{x}_{i_2}\boldsymbol{x}_{i_3}], \quad 1 \le i_1, i_2, i_3, i_4 \le k$$



(a) Multi-view models (b) Hidden Markov model

Figure 7.1: Examples of latent variable models.

*(*i.e.*, T is the fourth derivative tensor of the function $v \mapsto 8^{-1}\mathbb{E}[(v^\top x)^2]^2$. Let $\kappa_i := \mathbb{E}[h_i^4] - 3$ for each $i \in [k]$. Then*

$$M_4 \;=\; \sum_{i=1}^{k} \kappa_i \; \boldsymbol{\mu_i} \otimes \boldsymbol{\mu_i} \otimes \boldsymbol{\mu_i} \otimes \boldsymbol{\mu_i}.$$

See [87] for a proof of this theorem in this form. Note that κ_i corresponds to the excess kurtosis, a measure of non-Gaussianity as $\kappa_i = 0$ if h_i is a standard normal random variable. Furthermore, note that A is not identifiable if h is a multivariate Gaussian.

We may derive forms similar to that of M_2 and M_3 from Theorem 7.2 using M_4 by observing that

$$M_4(I, I, \boldsymbol{u}, \boldsymbol{u}) = \sum_{i=1}^{k} \kappa_i (\boldsymbol{\mu_i}^\top \boldsymbol{u})(\boldsymbol{\mu_i}^\top \boldsymbol{u}) \; \boldsymbol{\mu_i} \otimes \boldsymbol{\mu_i},$$

$$M_4(I, I, I, \boldsymbol{v}) = \sum_{i=1}^{k} \kappa_i (\boldsymbol{\mu_i}^\top \boldsymbol{v}) \; \boldsymbol{\mu_i} \otimes \boldsymbol{\mu_i} \otimes \boldsymbol{\mu_i}$$

for any vectors $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^d$.

7.3.3 Multi-view models

Multi-view models (also sometimes called naïve Bayes models) are a special class of Bayesian networks in which observed variables $\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_\ell}$ are conditionally independent given a latent variable h. This is similar to the exchangeable single topic model, but here we do not require the conditional distributions of the $\boldsymbol{x_t}, t \in [\ell]$ to be identical. Techniques developed for this class can be used to handle a number of widely used models including hidden Markov models (HMMs) [8, 130], phylogenetic tree models [38, 130], certain tree mixtures [7], and certain probabilistic grammar models [86].

As before, we let $h \in [k]$ be a discrete random variable with $\Pr[h = j] = w_j$ for all $j \in [k]$. Now consider random vectors $\boldsymbol{x_1} \in \mathbb{R}^{d_1}$, $\boldsymbol{x_2} \in \mathbb{R}^{d_2}$, and $\boldsymbol{x_3} \in \mathbb{R}^{d_3}$ which are conditionally independent given h, and

$$\mathbb{E}[\boldsymbol{x_t} | h = j] = \boldsymbol{\mu_{t,j}}, \quad j \in [k],\ t \in \{1, 2, 3\}$$

where the $\boldsymbol{\mu_{t,j}} \in \mathbb{R}^{d_t}$ are the conditional means of the $\boldsymbol{x_t}$ given $h = j$. Thus, we allow the observations $\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_\ell}$ to be random vectors, parameterized only by their conditional means. Importantly, these conditional distributions may be discrete, continuous, or even a mix of both.

We first note the form for the raw (cross) moments.

Proposition 7.4. *We have that:*

$$\begin{aligned}
\mathbb{E}[\boldsymbol{x_t} \otimes \boldsymbol{x_{t'}}] &= \sum_{i=1}^{k} w_i\ \boldsymbol{\mu_{t,i}} \otimes \boldsymbol{\mu_{t',i}}, \quad \{t, t'\} \subset \{1, 2, 3\}, t \neq t' \\
\mathbb{E}[\boldsymbol{x_1} \otimes \boldsymbol{x_2} \otimes \boldsymbol{x_3}] &= \sum_{i=1}^{k} w_i\ \boldsymbol{\mu_{1,i}} \otimes \boldsymbol{\mu_{2,i}} \otimes \boldsymbol{\mu_{3,i}}.
\end{aligned}$$

The cross moments do not possess a symmetric tensor form when the conditional distributions are different. Nevertheless, the moments can be “symmetrized” via a simple linear transformation of $\boldsymbol{x_1}$ and $\boldsymbol{x_2}$ (roughly speaking, this relates $\boldsymbol{x_1}$ and $\boldsymbol{x_2}$ to $\boldsymbol{x_3}$); this leads to an expression from which the conditional means of $\boldsymbol{x_3}$ (*i.e.*, $\mu_{3,1}, \mu_{3,2}, \ldots, \mu_{3,k}$) can be recovered. For simplicity, we assume $d_1 = d_2 = d_3 = k$; the general case (with $d_t \geq k$) is easily handled using low-rank singular value decompositions.

Theorem 7.5 ([9])**.** *Assume that the vectors* $\{\boldsymbol{\mu_{v,1}}, \boldsymbol{\mu_{v,2}}, \dots, \boldsymbol{\mu_{v,k}}\}$ *are linearly independent for each* $v \in \{1, 2, 3\}$*. Define*

$$\begin{aligned}
\tilde{\boldsymbol{x}}_1 &:= \mathbb{E}[\boldsymbol{x_3} \otimes \boldsymbol{x_2}]\mathbb{E}[\boldsymbol{x_1} \otimes \boldsymbol{x_2}]^{-1}\boldsymbol{x_1} \\
\tilde{\boldsymbol{x}}_2 &:= \mathbb{E}[\boldsymbol{x_3} \otimes \boldsymbol{x_1}]\mathbb{E}[\boldsymbol{x_2} \otimes \boldsymbol{x_1}]^{-1}\boldsymbol{x_2} \\
M_2 &:= \mathbb{E}[\tilde{\boldsymbol{x}}_1 \otimes \tilde{\boldsymbol{x}}_2] \\
M_3 &:= \mathbb{E}[\tilde{\boldsymbol{x}}_1 \otimes \tilde{\boldsymbol{x}}_2 \otimes \boldsymbol{x_3}].
\end{aligned}$$

Then

$$\begin{aligned}
M_2 &= \sum_{i=1}^{k} w_i \ \boldsymbol{\mu_{3,i}} \otimes \boldsymbol{\mu_{3,i}} \\
M_3 &= \sum_{i=1}^{k} w_i \ \boldsymbol{\mu_{3,i}} \otimes \boldsymbol{\mu_{3,i}} \otimes \boldsymbol{\mu_{3,i}}.
\end{aligned}$$

We now discuss three examples (taken mostly from [8]) where the above observations can be applied. The first two concern mixtures of product distributions, and the last one is the time-homogeneous hidden Markov model.

Hidden Markov models

Our last example is the time-homogeneous HMM for sequences of vector-valued observations $\boldsymbol{x_1}, \boldsymbol{x_2}, \dots \in \mathbb{R}^d$. Consider a Markov chain of discrete hidden states $y_1 \to y_2 \to y_3 \to \cdots$ over k possible states $[k]$; given a state y_t at time t, the observation $\boldsymbol{x_t}$ at time t (a random vector taking values in $\mathbb{R}^d$) is independent of all other observations and hidden states. See Figure 7.1(b).

Let $\pi \in \Delta^{k-1}$ be the initial state distribution (*i.e.*, the distribution of y_1), and $T \in \mathbb{R}^{k \times k}$ be the stochastic transition matrix for the hidden state Markov chain: for all times t,

$$\Pr[y_{t+1} = i | y_t = j] = T_{i,j}, \quad i, j \in [k].$$

Finally, let $M \in \mathbb{R}^{d \times k}$ be the matrix whose j-th column is the conditional expectation of $\boldsymbol{x_t}$ given $y_t = j$: for all times t,

$$\mathbb{E}[\boldsymbol{x_t} | y_t = j] = M\boldsymbol{e_j}, \quad j \in [k].$$

Proposition 7.6 ([8]). *Define $h := y_2$, where y_2 is the second hidden state in the Markov chain. Then*

- *$\boldsymbol{x_1}, \boldsymbol{x_2}, \boldsymbol{x_3}$ are conditionally independent given h;*
- *the distribution of h is given by the vector $w := T\pi \in \Delta^{k-1}$;*
- *for all $j \in [k]$,*

$$\begin{aligned}
\mathbb{E}[\boldsymbol{x_1} | h = j] &= M \operatorname{diag}(\pi) T^\top \operatorname{diag}(w)^{-1} \boldsymbol{e_j} \\
\mathbb{E}[\boldsymbol{x_2} | h = j] &= M \boldsymbol{e_j} \\
\mathbb{E}[\boldsymbol{x_3} | h = j] &= MT \boldsymbol{e_j}.
\end{aligned}$$

Note the matrix of conditional means of $\boldsymbol{x_t}$ has full column rank, for each $t \in \{1, 2, 3\}$, provided that: (i) M has full column rank, (ii) T is invertible, and (iii) π and $T\pi$ have positive entries.

7.4 Tensor power method

In this section, we consider the tensor power method of [105, Remark 3] for orthogonal tensor decomposition. We first state a simple convergence analysis for an orthogonally decomposable tensor T.

When only an approximation $\hat{T}$ to an orthogonally decomposable tensor T is available (*e.g.*, when empirical moments are used to estimate population moments), an orthogonal decomposition need not exist for this perturbed tensor (unlike for the case of matrices), and a more robust approach is required to extract the approximate decomposition. Here, we propose such a variant in Algorithm 7.1 and provide a detailed perturbation analysis. We note that alternative approaches such as simultaneous diagonalization can also be employed.

7.4.1 Convergence analysis for orthogonally decomposable tensors

The following lemma establishes the quadratic convergence of the tensor power method (*i.e.*, repeated iteration of (7.2)) for extracting a single component of the orthogonal decomposition. Note that the initial vector $\boldsymbol{\theta_0}$ determines which robust eigenvector will be the convergent point. Computation of subsequent eigenvectors can be computed with deflation, *i.e.*, by subtracting appropriate terms from T.

Lemma 7.7. *Let $T \in \bigotimes^3 \mathbb{R}^k$ have an orthogonal decomposition as given in* (7.1)*. For a vector $\boldsymbol{\theta_0} \in \mathbb{R}^k$, suppose that the set of numbers $|\lambda_1 \boldsymbol{v_1}^\top \boldsymbol{\theta_0}|, |\lambda_2 \boldsymbol{v_2}^\top \boldsymbol{\theta_0}|, \ldots, |\lambda_k \boldsymbol{v_k}^\top \boldsymbol{\theta_0}|$ has a unique largest element. Without loss of generality, say $|\lambda_1 \boldsymbol{v_1}^\top \boldsymbol{\theta_0}|$ is this largest value and $|\lambda_2 \boldsymbol{v_2}^\top \boldsymbol{\theta_0}|$ is the second largest value. For $t = 1, 2, \ldots,$ let*

$$\boldsymbol{\theta_t} \;:=\; \frac{T(I, \boldsymbol{\theta_{t-1}}, \boldsymbol{\theta_{t-1}})}{\|T(I, \boldsymbol{\theta_{t-1}}, \boldsymbol{\theta_{t-1}})\|}. \tag{7.2}$$

Then

$$\|\boldsymbol{v_1} - \boldsymbol{\theta_t}\|^2 \leq \left(2\lambda_1^2 \sum_{i=2}^{k} \lambda_i^{-2}\right) \cdot \left|\frac{\lambda_2 \boldsymbol{v_2}^\top \boldsymbol{\theta_0}}{\lambda_1 \boldsymbol{v_1}^\top \boldsymbol{\theta_0}}\right|^{2^{t+1}}.$$

That is, repeated iteration of (7.2) *starting from* $\boldsymbol{\theta_0}$ *converges to* $\boldsymbol{v_1}$ *at a quadratic rate.*

To obtain all eigenvectors, we may simply proceed iteratively using deflation, executing the power method on $T - \sum_j \lambda_j \boldsymbol{v_j}^{\otimes 3}$ after having obtained robust eigenvector / eigenvalue pairs $\{(\boldsymbol{v_j}, \lambda_j)\}$.

Proof. Let $\overline{\boldsymbol{\theta_0}}, \overline{\boldsymbol{\theta_1}}, \overline{\boldsymbol{\theta_2}}, \ldots$ be the sequence given by $\overline{\boldsymbol{\theta_0}} := \boldsymbol{\theta_0}$ and $\overline{\boldsymbol{\theta_t}} := T(I, \boldsymbol{\theta_{t-1}}, \boldsymbol{\theta_{t-1}})$ for $t \geq 1$. Let $c_i := \boldsymbol{v_i}^\top \boldsymbol{\theta_0}$ for all $i \in [k]$. It is easy to check that (i) $\boldsymbol{\theta_t} = \overline{\boldsymbol{\theta_t}} / \|\overline{\boldsymbol{\theta_t}}\|$, and (ii) $\overline{\boldsymbol{\theta_t}} = \sum_{i=1}^k \lambda_i^{2^t-1} c_i^{2^t} \boldsymbol{v_i}$. (Indeed, $\overline{\boldsymbol{\theta_{t+1}}} = \sum_{i=1}^k \lambda_i (\boldsymbol{v_i}^\top \overline{\boldsymbol{\theta_t}})^2 \boldsymbol{v_i} = \sum_{i=1}^k \lambda_i (\lambda_i^{2^t-1} c_i^{2^t})^2 \boldsymbol{v_i} = \sum_{i=1}^k \lambda_i^{2^{t+1}-1} c_i^{2^{t+1}} \boldsymbol{v_i}$.) Then

$$1 - (\boldsymbol{v_1}^\top \boldsymbol{\theta_t})^2 = 1 - \frac{(\boldsymbol{v_1}^\top \overline{\boldsymbol{\theta_t}})^2}{\|\overline{\boldsymbol{\theta_t}}\|^2} = 1 - \frac{\lambda_1^{2^{t+1}-2} c_1^{2^{t+1}}}{\sum_{i=1}^k \lambda_i^{2^{t+1}-2} c_i^{2^{t+1}}} \leq \frac{\sum_{i=2}^k \lambda_i^{2^{t+1}-2} c_i^{2^{t+1}}}{\sum_{i=1}^k \lambda_i^{2^{t+1}-2} c_i^{2^{t+1}}} \leq \lambda_1^2 \sum_{i=2}^{k} \lambda_i^{-2} \cdot \left|\frac{\lambda_2 c_2}{\lambda_1 c_1}\right|^{2^{t+1}}.$$

Since $\lambda_1 > 0$, we have $\boldsymbol{v_1}^\top \boldsymbol{\theta_t} > 0$ and hence $\|\boldsymbol{v_1} - \boldsymbol{\theta_t}\|^2 = 2(1 - \boldsymbol{v_1}^\top \boldsymbol{\theta_t}) \leq 2(1 - (\boldsymbol{v_1}^\top \boldsymbol{\theta_t})^2)$ as required. □

In this proof, the key observation is if $\boldsymbol{u} = \sum_{i=1}^k c_i \boldsymbol{v_i}$, then $T(I, \boldsymbol{u}, \boldsymbol{u} = \sum_{i=1}^k \lambda_i c_i^2 \boldsymbol{v_i}$, which is the same as the result of left multiplying u by a matrix M whose eigenvectors are $\boldsymbol{v_i}$'s and eigenvalues are $\lambda_i c_i$'s. Therefore we call $|\lambda_i c_i|$ the *effective eigenvalue*.

Definition 7.8 (Effective Eigenvalue)**.** Given $T = \sum_{i=1}^k \lambda_i \boldsymbol{v_i}$ and vector $\boldsymbol{u} = \sum_{i=1}^k c_i \boldsymbol{v_i}$, the effective eigenvalue with respect to $\boldsymbol{v_i}$ is $|\lambda_i c_i|$.

7.4.2 Perturbation analysis of a robust tensor power method

Now we consider the case where we have an approximation $\hat{T}$ to an orthogonally decomposable tensor T. Here, a more robust approach is required to extract an approximate decomposition. We propose such an algorithm in Algorithm 7.1, and provide a detailed perturbation analysis.

Algorithm 7.1. Robust tensor power method

```
input symmetric tensor $\tilde{T} \in \mathbb{R}^{k\times k\times k}$, number of iterations $L$, $N$.
output the estimated eigenvector/eigenvalue pair; the deflated tensor.
for $\tau = 1$ to $L$ do
    Draw $\boldsymbol{\theta}_{\mathbf{0}}^{(\boldsymbol{\tau})}$ uniformly at random from the unit sphere in $\mathbb{R}^k$.
    for $t = 1$ to $N$ do
        Compute power iteration update
            $\boldsymbol{\theta}_{\boldsymbol{t}}^{(\boldsymbol{\tau})} := \frac{\tilde{T}(I, \boldsymbol{\theta}_{\boldsymbol{t-1}}^{(\boldsymbol{\tau})}, \boldsymbol{\theta}_{\boldsymbol{t-1}}^{(\boldsymbol{\tau})})}{\|\tilde{T}(I, \boldsymbol{\theta}_{\boldsymbol{t-1}}^{(\boldsymbol{\tau})}, \boldsymbol{\theta}_{\boldsymbol{t-1}}^{(\boldsymbol{\tau})})\|}$    (7.3)
    end for
end for
Let $\tau^* := \arg\max_{\tau\in[L]}\{\tilde{T}(\boldsymbol{\theta}_{\boldsymbol{N}}^{(\boldsymbol{\tau})}, \boldsymbol{\theta}_{\boldsymbol{N}}^{(\boldsymbol{\tau})}, \boldsymbol{\theta}_{\boldsymbol{N}}^{(\boldsymbol{\tau})})\}$.
Do $N$ power iteration updates starting from $\boldsymbol{\theta}_{\boldsymbol{N}}^{(\boldsymbol{\tau^*})}$ to obtain $\hat{\boldsymbol{\theta}}$, and set $\hat{\lambda} := \tilde{T}(\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\theta}})$.
return the estimated eigenvector/eigenvalue pair $(\hat{\boldsymbol{\theta}}, \hat{\lambda})$; the deflated tensor $\tilde{T} - \hat{\lambda}\, \hat{\boldsymbol{\theta}}^{\otimes 3}$.
```

Algorithm 7.2. Robust tensor power method: Main Loop

```
input symmetric tensor $\tilde{T} \in \mathbb{R}^{k\times k\times k}$, number of iterations $L$, $N$.
output the estimated eigenvector/eigenvalue pairs
  for $i = 1$ to $k$ do
    Call Algorithm 7.1, get an eigenvector/eigenvalue pair, and replace $\tilde{T}$ with the deflated tensor
  end for
  return the estimated eigenvector/eigenvalue pairs
```

Assume that the symmetric tensor $T \in \mathbb{R}^{k\times k\times k}$ is orthogonally decomposable, and that $\hat{T} = T + E$, where the perturbation $E \in \mathbb{R}^{k\times k\times k}$ is a symmetric tensor with small spectral norm.

In our latent variable model applications, $\hat{T}$ is the tensor formed by using empirical moments, while T is the orthogonally decomposable tensor derived from the population moments for the given model.

The following theorem is similar to Wedin's perturbation theorem for singular vectors of matrices [160] in that it bounds the error of the (approximate) decomposition returned by Algorithm 7.1 on input $\hat{T}$ in terms of the size of the perturbation, provided that the perturbation is small enough.

Theorem 7.9. *Let $\hat{T} = T + E \in \mathbb{R}^{k\times k\times k}$, where T is a symmetric tensor with orthogonal decomposition $T = \sum_{i=1}^{k} \lambda_i \boldsymbol{v_i}^{\otimes 3}$ where each $\lambda_i > 0$, $\{\boldsymbol{v_1}, \boldsymbol{v_2}, \ldots, \boldsymbol{v_k}\}$ is an orthonormal basis, and E has operator norm $\epsilon := \|E\|$. Define $\lambda_{\min} := \min\{\lambda_i : i \in [k]\}$, and $\lambda_{\max} := \max\{\lambda_i : i \in [k]\}$. For any $\delta > 0$, there exists universal constants $C_1, C_2, C_3 > 0$ such that the following holds. Pick any $\eta \in (0,1)$, and suppose*

$$\epsilon \leq C_1 \cdot \frac{\lambda_{\min}}{k}, \qquad N \geq C_2 \cdot \left(\log(k) + \log\log\Big(\frac{\lambda_{\max}}{\epsilon}\Big)\right),$$

and $L = k^{1+\delta}\log(k/\eta)$. If Algorithm 7.2 returned eigenvector/eigenvalue pairs $(\boldsymbol{\hat{v}_1}, \hat{\lambda}_1), (\boldsymbol{\hat{v}_2}, \hat{\lambda}_2), \ldots, (\boldsymbol{\hat{v}_k}, \hat{\lambda}_k)$, then with probability at least $1-\eta$, there exists a permutation π on $[k]$ such that

$$\|\boldsymbol{v_{\pi(j)}} - \boldsymbol{\hat{v}_j}\| \leq 8\epsilon/\lambda_{\pi(j)}, \qquad |\lambda_{\pi(j)} - \hat{\lambda}_j| \leq 5\epsilon, \quad \forall j \in [k],$$

and

$$\left\| T - \sum_{j=1}^{k} \hat{\lambda}_j \boldsymbol{\hat{v}_j}^{\otimes 3} \right\| \leq 55\epsilon.$$

The proof of Theorem 7.9 is given in Section 7.6.

One important difference from Wedin's theorem is that this is an algorithm dependent perturbation analysis, specific to Algorithm 7.1 (since the perturbed tensor need not have an orthogonal decomposition). Furthermore, note that Algorithm 7.1 uses multiple restarts to ensure (approximate) convergence—the intuition is that by restarting at multiple points, we eventually start at a point in which the initial contraction towards some eigenvector dominates the error E in our tensor. The proof shows that we find such a point with high probability within $L = \text{poly}(k)$ trials. It should be noted that for large k, the required bound on L is very close to linear in k.

In general, it is possible, when run on a general symmetric tensor (*e.g.*, $\hat{T}$), for the tensor power method to exhibit oscillatory behavior [98, Example 1]. This is not in conflict with Theorem 7.9, which effectively bounds the amplitude of these oscillations; in particular, if $\hat{T} = T + E$ is a tensor built from empirical moments, the error term E (and thus the amplitude of the oscillations) can be driven down by drawing more samples. The practical value of addressing these oscillations and perhaps stabilizing the algorithm is an interesting direction for future research [100].

A final consideration is that for specific applications, it may be possible to use domain knowledge to choose better initialization points. For instance, in the topic modeling applications (*cf.* Section 7.3.1), the eigenvectors are related to the topic word distributions, and many documents may be primarily composed of words from just single topic. Therefore, good initialization points can be derived from these single-topic documents themselves, as these points would already be close to one of the eigenvectors.

7.5 Discussion

7.5.1 Computational complexity

It is interesting to consider the computational complexity of the tensor power method in the dense setting where $T \in \mathbb{R}^{k \times k \times k}$ is orthogonally decomposable but otherwise unstructured. Each iteration requires $O(k^3)$ operations, and assuming at most $k^{1+\delta}$ random restarts for extracting each eigenvector (for some small $\delta > 0$) and $O(\log(k) + \log\log(1/\epsilon))$ iterations per restart, the total running time is $O(k^{5+\delta}(\log(k) + \log\log(1/\epsilon)))$ to extract all k eigenvectors and eigenvalues.

An alternative approach to extracting the orthogonal decomposition of T is to reorganize T into a matrix $M \in \mathbb{R}^{k \times k^2}$ by flattening two of the dimensions into one. In this case, if $T = \sum_{i=1}^{k} \lambda_i \boldsymbol{v}_{\boldsymbol{i}}^{\otimes 3}$, then $M = \sum_{i=1}^{k} \lambda_i \boldsymbol{v}_{\boldsymbol{i}} \otimes \mathrm{vec}(\boldsymbol{v}_{\boldsymbol{i}} \otimes \boldsymbol{v}_{\boldsymbol{i}})$. This reveals the singular value decomposition of M (assuming the eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_k$ are distinct), and therefore can be computed with $O(k^4)$ operations. Therefore it seems that the tensor power method is less efficient than a pure matrix-based approach via singular value decomposition. However, it should be noted that this matrix-based approach fails to recover the decomposition when eigenvalues are repeated, and it is unstable when the gap between eigenvalues is small.

It is worth noting that the running times differ by roughly a factor of $\Theta(k^{1+\delta})$, which can be accounted for by the random restarts. This gap can potentially be alleviated or removed by using a more clever method for initialization. Moreover, using special structure in the problem (as discussed above) can also improve the running time of the tensor power method.

7.5.2 Sample complexity bounds

Previous work on using linear algebraic methods for estimating latent variable models crucially rely on matrix perturbation analysis for deriving sample complexity bounds [8, 9, 87, 88, 130]. The learning algorithms in these works are plug-in estimators that use empirical moments in place of the population moments, and then follow algebraic manipulations that result in the desired parameter estimates. As long as these manipulations can tolerate small perturbations of the population moments, a sample complexity bound can be obtained by exploiting the convergence of the empirical moments to the population moments via the law of large numbers.

Using the perturbation analysis for the tensor power method, improved sample complexity bounds can be obtained for all of the examples discussed in Section 7.3. The underlying analysis remains the same as in previous works (*e.g.*, [9, 87]), the main difference being the accuracy of the orthogonal tensor decomposition obtained via the tensor power method.

Relative to the previously cited works, the sample complexity bound will be considerably improved in its dependence on the rank parameter k, as Theorem 7.9 implies that the tensor estimation error (*e.g.*, error in estimating $\widetilde{M}_3$ from Section 7.2) is not amplified by any factor explicitly depending on k (there is a requirement that the error be smaller than some factor depending on k, but this only contributes to a lower-order term in the sample complexity bound).

7.6 Analysis of robust power method

In this section, we prove Theorem 7.9.

The proof is divided into several steps. First in Section 7.6.1 we define what are "good" initializazers for the tensor power iterations, and show that a random vector is good with significant probability. Then in Section 7.6.2 we show given a good initializer, the tensor power method will quickly reach the neighborhood of the top eigenvector. Finally in Section 7.6.3, we show that the error in deflation step does not accrue very badly. These three steps is assembled in Section 7.6.4 to complete the proof.

The vectors during the tensor power method will be called $\boldsymbol{\theta_0}, \boldsymbol{\theta_1}, ...$, and we express each vector in the basis of $\{\boldsymbol{v_1}, \boldsymbol{v_2}, \ldots, \boldsymbol{v_k}\}$:

$$\boldsymbol{\theta_t} = \sum_{i=1}^{k} \theta_{i,t} \boldsymbol{v_i}.$$

.

Recall the update rule used in the power method,

$$\boldsymbol{\theta_{t+1}} = \sum_{i=1}^{k} \theta_{i,t+1} \boldsymbol{v_i} := \tilde{T}(I, \boldsymbol{\theta_t}, \boldsymbol{\theta_t}) / \|\tilde{T}(I, \boldsymbol{\theta_t}, \boldsymbol{\theta_t})\|.$$

In this subsection, we assume that $\tilde{T}$ has the form

$$\tilde{T} = \sum_{i=1}^{k} \tilde{\lambda}_i \boldsymbol{v_i}^{\otimes 3} + \tilde{E} \tag{7.4}$$

where $\{\boldsymbol{v_1}, \boldsymbol{v_2}, \ldots, \boldsymbol{v_k}\}$ is an orthonormal basis, and, without loss of generality,

$$\tilde{\lambda}_1 |\theta_{1,t}| = \max_{i \in [k]} \tilde{\lambda}_i |\theta_{i,t}| > 0.$$

Also, define

$$\tilde{\lambda}_{\min} := \min\{\tilde{\lambda}_i : i \in [k],\ \tilde{\lambda}_i > 0\}, \quad \tilde{\lambda}_{\max} := \max\{\tilde{\lambda}_i : i \in [k]\}.$$

We do not use the original eigenvalues λ_i's here, because throughout the deflation process, some eigenvalues will be deflated and becomes 0.

We further assume the error $\tilde{E}$ is a symmetric tensor such that, for some constant $p > 1$,

$$\|\tilde{E}(I, \boldsymbol{u}, \boldsymbol{u})\| \leq \tilde{\epsilon}, \quad \forall \boldsymbol{u} \in S^{k-1}; \tag{7.5}$$

$$\|\tilde{E}(I, \boldsymbol{u}, \boldsymbol{u})\| \leq \tilde{\epsilon}/p, \quad \forall \boldsymbol{u} \in S^{k-1} \text{ s.t. } (\boldsymbol{u}^\top \boldsymbol{v_1})^2 \geq 1 - (3\tilde{\epsilon}/\tilde{\lambda}_1)^2. \tag{7.6}$$

All these settings may seem very unnatural at this point, but they are important for the later deflation analysis. In particular, after some deflation steps, the error $\tilde{E}$ will be the original error E plus the additional error introduced in the deflation process. Later we will show that the spectral norm of the additional error is bounded by $\tilde{\epsilon}$ which is a constant factor more than ϵ. However that alone is not enough because if the error grows by a constant factor at every deflation step, it will be huge at the end of iteration. The additional insight is that the error introduced by deflation is not *uniform* along all directions, it is only large when the current vector has large projection on the previously found eigenvectors, and will be small when the current vector is close to a new eigenvector (which itself is orthogonal to all the previously found eigenvectors). That is why constant p is introduced here.

7.6.1 Initialization

For $\gamma \in (0,1)$, we say a unit vector $\boldsymbol{\theta_0} \in \mathbb{R}^k$ is γ-*separated* relative to $i^* \in [k]$ if

$$|\theta_{i^*,0}| - \max_{i \in [k] \setminus \{i^*\}} |\theta_{i,0}| \geq \gamma |\theta_{i^*,0}|.$$

Intuitively, a γ-separated vector is a good initializer for the tensor power method, because it has significantly larger correlation with one of the vectors, and we can hope that this initial advantage will be amplified and in the end the process converges to the eigenvector.

The following lemma shows that for any constant γ, with probability at least $1 - \eta$, at least one out of $\operatorname{poly}(k) \log(1/\eta)$ i.i.d. random vectors (uniformly distributed over the unit sphere S^{k-1}) is γ-separated relative to $\arg\max_{i \in [k]} \tilde{\lambda}_i$. (For small enough γ and large enough k, the polynomial is close to linear in k.)

Lemma 7.10. *For any constant γ, there exists a δ such that with probability at least $1-\eta$ over $k^{1+\delta} \cdot \log_2(1/\eta)$ i.i.d. uniform random unit vectors, at least one of the vectors is γ-separated relative to* $\arg\max_{i \in [k]} \tilde{\lambda}_i$.

Proof. Without loss of generality, assume $\arg\max_{i \in [k]} \tilde{\lambda}_i = 1$. Suppose we take $L = k^{1+\delta}$ vectors. Consider a random matrix $Z \in \mathbb{R}^{k \times L}$ whose entries are independent $\mathcal{N}(0,1)$ random variables; we take the j-th column of Z to be comprised of the random variables used for the j-th random vector (before normalization). Specifically, for the j-th random vector,

$$\theta_{i,0} := \frac{Z_{i,j}}{\sqrt{\sum_{i'=1}^{k} Z_{i',j}^2}}, \quad i \in [n].$$

It suffices to show that with probability at least 1/2, there is a column $j^* \in [L]$ such that

$$|Z_{1,j^*}| \geq \frac{1}{1-\gamma} \max_{i \in [k] \setminus \{1\}} |Z_{i,j^*}|.$$

Since $\max_{j \in [L]} |Z_{1,j}|$ is a 1-Lipschitz function of L independent $\mathcal{N}(0,1)$ random variables, it follows that

$$\Pr\left[\left|\max_{j \in [L]} |Z_{1,j}| - \text{median}\left[\max_{j \in [L]} |Z_{1,j}|\right]\right| > \sqrt{2\ln(8)}\right] \leq 1/4.$$

Moreover,

$$\text{median}\left[\max_{j \in [L]} |Z_{1,j}|\right] \geq \text{median}\left[\max_{j \in [L]} Z_{1,j}\right] =: m.$$

Observe that the cumulative distribution function of $\max_{j \in [L]} Z_{1,j}$ is given by $F(z) = \Phi(z)^L$, where Φ is the standard Gaussian CDF. Since $F(m) = 1/2$, it follows that $m = \Phi^{-1}(2^{-1/L})$. It can be checked that

$$\Phi^{-1}(2^{-1/L}) \geq \sqrt{2\ln(L)} - \frac{\ln(\ln(L)) + c}{2\sqrt{2\ln(L)}}$$

for some absolute constant $c > 0$. Also, let $j^* := \arg\max_{j \in [L]} |Z_{1,j}|$.

Now for each $j \in [L]$, let $|Z_{2:k,j}| := \max\{|Z_{2,j}|, |Z_{3,j}|, \ldots, |Z_{k,j}|\}$. Again, since $|Z_{2:k,j}|$ is a 1-Lipschitz function of $k-1$ independent $\mathcal{N}(0,1)$ random variables, it follows that

$$\Pr\left[|Z_{2:k,j}| > \mathbb{E}\left[|Z_{2:k,j}|\right] + \sqrt{2\ln(4)}\right] \leq 1/4.$$

Moreover, by a standard argument,

$$\mathbb{E}\left[|Z_{2:k,j}|\right] \leq \sqrt{2\ln(k)}.$$

Since $|Z_{2:k,j}|$ is independent of $|Z_{1,j}|$ for all $j \in [L]$, it follows that the previous two displayed inequalities also hold with j replaced by j^*.

Therefore we conclude with a union bound that with probability at least $1/2$,

$$|Z_{1,j^*}| \geq \sqrt{2\ln(L)} - \frac{\ln(\ln(L)) + c}{2\sqrt{2\ln(L)}} - \sqrt{2\ln(8)} \quad \text{and} \quad |Z_{2:k,j^*}| \leq \sqrt{2\ln(k)} + \sqrt{2\ln(4)}.$$

When this even happens, and when δ is large enough, the j^*-th random vector is γ-separated. The probability of success can easily be boosted from $1/2$ to $1-\eta$ by taking $\log(1/\eta)L$ samples, as all the samples are independent.

□

7.6.2 Tensor power iterations

In this section we show a good initializer will converge to the neighborhood of the top eigenvector if the error on the tensor is small enough.

In the next two propositions (Propositions 7.11 and 7.12) and next two lemmas (Lemmas 7.13 and 7.14), we analyze the power method iterations using $\tilde{T}$ at some arbitrary iterate $\boldsymbol{\theta_t}$ using only the property (7.5) of $\tilde{E}$. But throughout, the quantity $\tilde{\epsilon}$ can be replaced by $\tilde{\epsilon}/p$ if $\boldsymbol{\theta_t}$ satisfies $(\boldsymbol{\theta_t}^\top \boldsymbol{v_1})^2 \geq 1 - (3\tilde{\epsilon}/\tilde{\lambda}_1)^2$ as per property (7.6).

We first define several quantities to measure the progress and simplify calculations.

$$\begin{aligned} R_\tau &:= \left(\frac{\theta_{1,\tau}^2}{1-\theta_{1,\tau}^2} \right)^{1/2}, \qquad & r_{i,\tau} &:= \frac{\tilde{\lambda}_1 \theta_{1,\tau}}{\tilde{\lambda}_i |\theta_{i,\tau}|}, \\ \gamma_\tau &:= 1 - \frac{1}{\min_{i\neq 1} |r_{i,\tau}|}, \qquad & \delta_\tau &:= \frac{\tilde{\epsilon}}{\tilde{\lambda}_1 \theta_{1,\tau}^2} \end{aligned} \tag{7.7}$$

for $\tau \in \{t, t+1\}$.

Here R is used to measure progress (in the end we hope R would be very large). The quantity γ is similar to the "eigengap" for matrices (recall the definition of effective eigenvalue in Definition 7.8. Initially γ should be a small constant for good initial vectors, and will remain at least a constant. The quantity δ should be considered as the relative size of the error (at the beginning it is a small constant and decreases as the algorithm proceeds). The two propositions below are really just calculations.

Proposition 7.11.

$$\min_{i\neq 1} |r_{i,t}| \geq R_t, \qquad \gamma_t \geq 1 - \frac{1}{R_t}, \qquad \theta_{1,t}^2 = \frac{R_t^2}{1+R_t^2}.$$

Proposition 7.12.

$$r_{i,t+1} \geq r_{i,t}^2 \cdot \frac{1-\delta_t}{1+\delta_t r_{i,t}^2} = \frac{1-\delta_t}{\frac{1}{r_{i,t}^2}+\delta_t}, \quad i \in [k], \tag{7.8}$$

$$R_{t+1} \geq R_t \cdot \frac{1-\delta_t}{1-\gamma_t+\delta_t R_t} \geq \frac{1-\delta_t}{\frac{1}{R_t^2}+\delta_t}. \tag{7.9}$$

Proof. Let $\check{\boldsymbol{\theta}}_{\boldsymbol{t+1}} := \tilde{T}(I, \boldsymbol{\theta_t}, \boldsymbol{\theta_t})$, so $\boldsymbol{\theta_{t+1}} = \check{\boldsymbol{\theta}}_{\boldsymbol{t+1}} / \|\check{\boldsymbol{\theta}}_{\boldsymbol{t+1}}\|$. Since $\check{\theta}_{i,t+1} = \tilde{T}(\boldsymbol{v_i}, \boldsymbol{\theta_t}, \boldsymbol{\theta_t}) = T(\boldsymbol{v_i}, \boldsymbol{\theta_t}, \boldsymbol{\theta_t}) + E(\boldsymbol{v_i}, \boldsymbol{\theta_t}, \boldsymbol{\theta_t})$, we have

$$\check{\theta}_{i,t+1} = \tilde{\lambda}_i \theta_{i,t}^2 + E(\boldsymbol{v_i}, \boldsymbol{\theta_t}, \boldsymbol{\theta_t}), \quad i \in [k].$$

Using the triangle inequality and the fact $\|E(\boldsymbol{v_i}, \boldsymbol{\theta_t}, \boldsymbol{\theta_t})\| \leq \tilde{\epsilon}$, we have

$$\check{\theta}_{i,t+1} \geq \tilde{\lambda}_i \theta_{i,t}^2 - \tilde{\epsilon} \geq |\theta_{i,t}| \cdot \Big(\tilde{\lambda}_i |\theta_{i,t}| - \tilde{\epsilon}/|\theta_{i,t}|\Big) \tag{7.10}$$

and

$$|\check{\theta}_{i,t+1}| \leq |\tilde{\lambda}_i \theta_{i,t}^2| + \tilde{\epsilon} \leq |\theta_{i,t}| \cdot \Big(\tilde{\lambda}_i |\theta_{i,t}| + \tilde{\epsilon}/|\theta_{i,t}|\Big) \tag{7.11}$$

for all $i \in [k]$. Combining (7.10) and (7.11) gives

$$r_{i,t+1} = \frac{\tilde{\lambda}_1 \theta_{1,t+1}}{\tilde{\lambda}_i |\theta_{i,t+1}|} = \frac{\tilde{\lambda}_1 \check{\theta}_{1,t+1}}{\tilde{\lambda}_i |\check{\theta}_{i,t+1}|} \geq r_{i,t}^2 \cdot \frac{1-\delta_t}{1+\frac{\tilde{\epsilon}}{\tilde{\lambda}_i \theta_{i,t}^2}} = r_{i,t}^2 \cdot \frac{1-\delta_t}{1+(\tilde{\lambda}_i/\tilde{\lambda}_1)\delta_t r_{i,t}^2} \geq r_{i,t}^2 \cdot \frac{1-\delta_t}{1+\delta_t r_{i,t}^2}.$$

Moreover, by the triangle inequality and Hölder's inequality,

$$\begin{aligned}\Big(\sum_{i=2}^{n}[\check{\theta}_{i,t+1}]^2\Big)^{1/2} &= \Big(\sum_{i=2}^{n}\Big(\tilde{\lambda}_i\theta_{i,t}^2+E(\boldsymbol{v_i},\boldsymbol{\theta_t},\boldsymbol{\theta_t})\Big)^2\Big)^{1/2}\\ &\le \Big(\sum_{i=2}^{n}\tilde{\lambda}_i^2\theta_{i,t}^4\Big)^{1/2}+\Big(\sum_{i=2}^{n}E(\boldsymbol{v_i},\boldsymbol{\theta_t},\boldsymbol{\theta_t})^2\Big)^{1/2}\\ &\le \max_{i\neq 1}\tilde{\lambda}_i|\theta_{i,t}|\Big(\sum_{i=2}^{n}\theta_{i,t}^2\Big)^{1/2}+\tilde{\epsilon}\\ &= (1-\theta_{1,t}^2)^{1/2}\cdot\Big(\max_{i\neq 1}\tilde{\lambda}_i|\theta_{i,t}|+\tilde{\epsilon}/(1-\theta_{1,t}^2)^{1/2}\Big). \qquad (7.12)\end{aligned}$$

Combining (7.10) and (7.12) gives

$$\frac{|\theta_{1,t+1}|}{(1-\theta_{1,t+1}^2)^{1/2}}=\frac{|\check{\theta}_{1,t+1}|}{\Big(\sum_{i=2}^{n}[\check{\theta}_{i,t+1}]^2\Big)^{1/2}}\ge\frac{|\theta_{1,t}|}{(1-\theta_{1,t}^2)^{1/2}}\cdot\frac{\tilde{\lambda}_1|\theta_{1,t}|-\tilde{\epsilon}/|\theta_{1,t}|}{\max_{i\neq 1}\tilde{\lambda}_i|\theta_{i,t}|+\tilde{\epsilon}/(1-\theta_{1,t}^2)^{1/2}}.$$

In terms of R_{t+1}, R_t, γ_t, and δ_t, this reads

$$R_{t+1}\ge\frac{1-\delta_t}{(1-\gamma_t)\Big(\frac{1-\theta_{1,t}^2}{\theta_{1,t}^2}\Big)^{1/2}+\delta_t}=R_t\cdot\frac{1-\delta_t}{1-\gamma_t+\delta_t R_t}=\frac{1-\delta_t}{\frac{1-\gamma_t}{R_t}+\delta_t}\ge\frac{1-\delta_t}{\frac{1}{R_t^2}+\delta_t}$$

where the last inequality follows from Proposition 7.11. □

The following Lemma shows how the vector changes during tensor power iterations. There are several phases, in the first phase Theorem 7.13 shows we make progress as long as $R_t \le 9$.

Lemma 7.13 (First phase)**.** *Assume $0 \le \delta_t < 1/18$, and $\gamma_t > 18\delta_t$.*

1. *If $r_{i,t}^2 \le 8$, then $r_{i,t+1} \ge |r_{i,t}|\big(1+\frac{\gamma_t}{2}\big)$.*

2. *If $4 < r_{i,t}^2$, then $r_{i,t+1} \ge \min\{r_{i,t}^2/2,\ \frac{1/2-\delta_t}{\delta_t}\}$.*

3. *"Spectral gap" always increases until it reaches 1/2: $\gamma_{t+1} \ge \min\{\gamma_t, 1/2\}$.*

4. *Make progress whenever R_t is small: If $R_t \le 9$, then $R_{t+1} \ge R_t\big(1 + \frac{\gamma_t}{3}\big)$, $\theta_{1,t+1}^2 \ge \theta_{1,t}^2$ and $\delta_{t+1} \le \delta_t$.*

Proof. Consider two (overlapping) cases depending on $r_{i,t}^2$.

- Case 1: $r_{i,t}^2 \le 2\rho^2$. By (7.8) from Proposition 7.12,

  $$r_{i,t+1} \ge r_{i,t}^2 \cdot \frac{1-\delta_t}{1+\kappa\delta_t r_{i,t}^2} \ge |r_{i,t}| \cdot \frac{1}{1-\gamma_t} \cdot \frac{1-\delta_t}{1+2\kappa\rho^2\delta_t} \ge |r_{i,t}|\Big(1+\frac{\gamma_t}{2}\Big)$$

  where the last inequality uses the assumption $\gamma_t > 2(1+2\kappa\rho^2)\delta_t$. This proves the first claim.

- Case 2: $\rho^2 < r_{i,t}^2$. We split into two sub-cases. Suppose $r_{i,t}^2 \le (\rho(1-\delta_t)-1)/(\kappa\delta_t)$. Then, by (7.8),

  $$r_{i,t+1} \ge r_{i,t}^2 \cdot \frac{1-\delta_t}{1+\kappa\delta_t r_{i,t}^2} \ge r_{i,t}^2 \cdot \frac{1-\delta_t}{1+\kappa\delta_t\frac{\rho(1-\delta_t)-1}{\kappa\delta_t}} = \frac{r_{i,t}^2}{\rho}.$$

  Now suppose instead $r_{i,t}^2 > (\rho(1-\delta_t)-1)/(\kappa\delta_t)$. Then

  $$r_{i,t+1} \ge \frac{1-\delta_t}{\frac{\kappa\delta_t}{\rho(1-\delta_t)-1} + \kappa\delta_t} = \frac{1-\delta_t-1/\rho}{\kappa\delta_t}. \tag{7.13}$$

Observe that if $\min_{i\ne 1} r_{i,t}^2 \le (\rho(1-\delta_t)-1)/(\kappa\delta_t)$, then $r_{i,t+1} \ge |r_{i,t}|$ for all $i \in [k]$, and hence $\gamma_{t+1} \ge \gamma_t$. Otherwise we have $\gamma_{t+1} > 1 - \frac{\kappa\delta_t}{1-\delta_t-1/\rho} > 1 - 1/\rho$. This proves the third claim.

Finally, for the last claim, if $R_t \le 1+2\kappa\rho^2$, then by (7.9) from Proposition 7.12 and the assumption $\gamma_t > 2(1+2\kappa\rho^2)\delta_t$,

$$R_{t+1} \ge R_t \cdot \frac{1-\delta_t}{1-\gamma_t+\delta_t R_t} \ge R_t \cdot \frac{1-\frac{\gamma_t}{2(1+2\kappa\rho^2)}}{1-\gamma_t/2} \ge R_t\bigg(1+\gamma_t\cdot\frac{\kappa\rho^2}{1+2\kappa\rho^2}\bigg) \ge R_t\Big(1+\frac{\gamma_t}{3}\Big).$$

This in turn implies that $\theta_{1,t+1}^2 \ge \theta_{1,t}^2$ via Proposition 7.11, and thus $\delta_{t+1} \le \delta_t$. □

Next Lemma deals with the second phase, when R is larger than 9. It shows R_t grows in a quadratic speed until it becomes really large, and once it is really large it stays large.

Lemma 7.14 (Second Phase)**.** *When* $9 \le R_t \le \frac{\tilde{\lambda}_1}{3\tilde{\epsilon}}$, $R_{t+1} \ge \min\{R_t^2/2, \frac{\tilde{\lambda}_1}{3\tilde{\epsilon}}\}$.

When $R_t \ge \frac{\tilde{\lambda}_1}{3\tilde{\epsilon}}$, $R_{t+1} \ge \frac{\tilde{\lambda}_1}{3\tilde{\epsilon}}$

Proof. Observe that for any $c > 0$,

$$R_t \ge \frac{1}{c} \quad \Leftrightarrow \quad \theta_{1,t}^2 \ge \frac{1}{1+c^2} \quad \Leftrightarrow \quad \delta_t \le \frac{(1+c^2)\tilde{\epsilon}}{\tilde{\lambda}_1}. \tag{7.14}$$

Now consider the following cases depending on R_t.

- Case 1: $R_t \ge \frac{\tilde{\lambda}_1}{3\tilde{\epsilon}} = 1/\alpha$. In this case, we have

  $$\delta_t \le \frac{(1+\alpha^2)\tilde{\epsilon}}{\tilde{\lambda}_1} \le \frac{\alpha\gamma_t}{1+\alpha}$$

  by (7.14) (with $c = \alpha$). Combining this with (7.9) from Proposition 7.12 gives

  $$R_{t+1} \ge \frac{1-\delta_t}{\frac{1-\gamma_t}{R_t} + \delta_t} \ge \frac{1 - \frac{\alpha\gamma_t}{1+\alpha}}{(1-\gamma_t)\alpha + \frac{\alpha\gamma_t}{1+\alpha}} = \frac{1}{\alpha}.$$

- Case 2: $9 \le R_t < \frac{\tilde{\lambda}_1}{3\tilde{\epsilon}}$. In this case, we have

  $$\delta_t \le \frac{(1+1/9^2)\tilde{\epsilon}}{\tilde{\lambda}_1} \le \frac{1.1\tilde{\epsilon}}{\tilde{\lambda}_1}$$

  by (7.14), we have if $\delta_t \ge 1/(2+R_t^2)$, then (7.9) implies

  $$R_{t+1} \ge \frac{1-\delta_t}{\frac{1}{R_t^2} + \delta_t} \ge \frac{1-2\delta_t}{2\delta_t} \ge \frac{\tilde{\lambda}_1}{3\tilde{\epsilon}}.$$

If instead $\delta_t < 1/(2 + R_t^2)$, then (7.9) implies

$$R_{t+1} \geq \frac{1 - \delta_t}{\frac{1}{R_t^2} + \delta_t} > \frac{1 - \frac{1}{2+R_t^2}}{\frac{1}{R_t^2} + \frac{1}{2+R_t^2}} = \frac{R_t^2}{2}. \qquad \square$$

Approximate recovery of a single eigenvector

We now state the main result regarding the approximate recovery of a single eigenvector using the tensor power method on $\tilde{T}$. Here, we exploit the special properties of the error $\tilde{E}$ (both (7.5) and (7.6)).

Lemma 7.15 (Convergence for Top Eigenvector)**.** *There exists a universal constant $C > 0$ such that the following holds.*

If the initializer is γ_0-biased ($0 < \gamma_0 < 1$) with respect to $i^* = \arg\max \tilde{\lambda}_i$, when $\tilde{\epsilon} < \tilde{\lambda}_{\max} \cdot \theta_{i^*,0}^2 \cdot \gamma_0/18$, and $N \geq C \cdot \left(\log(k/\gamma_0) + \log\log \frac{p\tilde{\lambda}_{i^*}}{\tilde{\epsilon}}\right)$, after $t \geq N$ iterations of the tensor power method on tensor $\tilde{T}$ as defined in (7.4) and satisfying (7.5) and (7.6), the final vector $\boldsymbol{\theta_t}$ satisfies

$$\theta_{i^*,t} \geq \sqrt{1 - \left(\frac{3\tilde{\epsilon}}{p\tilde{\lambda}_{i^*}}\right)^2}, \quad \|\boldsymbol{\theta_t} - v_{i^*}\| \leq \frac{4\tilde{\epsilon}}{p\tilde{\lambda}_{i^*}}, \quad |\tilde{T}(\boldsymbol{\theta_t}, \boldsymbol{\theta_t}, \boldsymbol{\theta_t}) - \tilde{\lambda}_{i^*}| \leq \left(27\Big(\frac{\tilde{\epsilon}}{p\lambda_{i^*}}\Big)^2 + 2\right)\frac{\tilde{\epsilon}}{p}.$$

Proof. Assume without loss of generality that $i^* = 1$. We consider three phases: (i) iterations before the first time t such that $R_t > 9$, (ii) the subsequent iterations before the first time t such that $R_t \geq \frac{\tilde{\lambda}_1}{3\tilde{\epsilon}}$ and (iii) If $p > 1$, the subsequent iterations before the first time t such that $R_t \geq \frac{p\tilde{\lambda}_1}{3\tilde{\epsilon}}$.

We begin by analyzing the first phase, *i.e.*, the iterates in $T_1 := \{t \geq 0 : R_t \leq 9\}$. Observe that the condition on $\tilde{\epsilon}$ implies

$$\delta_0 = \frac{\tilde{\epsilon}}{\tilde{\lambda}_1 \theta_{1,0}^2} < \gamma_0/18,$$

and hence the preconditions on δ_t and γ_t of Lemma 7.13 hold for $t = 0$. For all $t \in T_1$ satisfying the preconditions, Lemma 7.13 implies that $\delta_{t+1} \leq \delta_t$ and $\gamma_{t+1} \geq \min\{\gamma_t, 1/2\}$, so the next iteration also satisfies the preconditions. Hence by induction, the preconditions hold for all iterations in T_1. Moreover, for all $i \in [k]$, we have

$$|r_{i,0}| \geq \frac{1}{1-\gamma_0};$$

and while $t \in T_1$: (i) $|r_{i,t}|$ increases at a linear rate while $r_{i,t}^2 \leq 8$, and (ii) $|r_{i,t}|$ stays above 2 once it is at least 2.(The specific rates are given, respectively, in Lemma 7.13, claims 1 and 2.)

It follows that $\min_{i \neq 1} r_{i,t}^2 \leq 4$ for at most

$$\frac{2}{\gamma_0} \ln\left(\frac{\sqrt{8}}{\frac{1}{1-\gamma_0}}\right) = O(\log 1/\gamma_0) \tag{7.15}$$

iterations in T_1.

We know $R_0 \geq 1/\sqrt{k}$ at the beginning, and it increases by a factor of $(1 + \gamma_t/3)$ at every step, therefore there are at most an additional $O(\log k)$ steps after γ reaches 1/2.

Therefore, by combining the counts, we have that the number of iterations in the first phase is at most $|T_1| = O\Bigg(\log k/\gamma\Bigg)$.

We now analyze the second phase, *i.e.*, the iterates in $T_2 := \{t \geq 0 : t \notin T_1,\ R_t < \frac{\tilde{\lambda}_1}{3\tilde{\epsilon}}\}$.

Note that for the initial iteration $t' := \min T_2$, we have that $R_{t'} \geq 1+2\kappa\rho^2 = 1+8\kappa = 1/\beta$, Lemma 7.14 implies that $R_{t+1} \geq \min\{R_t, \frac{\tilde{\lambda}_1}{3\tilde{\epsilon}}\}$.

To bound the number of iterations in T_2, observe that R_t increases at a quadratic rate until $R_t \geq \frac{\tilde{\lambda}_1}{3\tilde{\epsilon}}$, so $|T_2| = O\Bigg(\log\log \frac{\tilde{\lambda}_1}{3\tilde{\epsilon}}\Bigg)$.

After $R_{t''} \geq \frac{\tilde{\lambda}_1}{3\tilde{\epsilon}}$, we have

$$\theta_{1,t''}^2 \geq 1 - \left(\frac{3\tilde{\epsilon}}{\tilde{\lambda}_1}\right)^2.$$

Therefore, the vector $\theta_{t''}$ satisfies the condition for property (7.6) of $\tilde{E}$ to hold. Now we apply Lemma 7.14 using $\tilde{\epsilon}/p$ in place of $\tilde{\epsilon}$, including in the definition of δ_t (which we call $\overline{\delta}_t$):

$$\overline{\delta}_t := \frac{\tilde{\epsilon}}{p\tilde{\lambda}_1 \theta_{1,t}^2};$$

Similar as before, by Lemma 7.14, we have that R_t increases at a quadratic rate in this final phase until $R_t \geq \frac{p\tilde{\lambda}_1}{3\tilde{\epsilon}}$. So the number of iterations before $R_t \geq \frac{\tilde{\lambda}_1}{3\tilde{\epsilon}}$ can be bounded as $O\left(\log\log \frac{p\tilde{\lambda}_1}{\tilde{\epsilon}}\right)$.

Once $R_t \geq \frac{p\tilde{\lambda}_1}{3\tilde{\epsilon}}$, we have $\theta_{1,t}^2 \geq 1 - \left(\frac{3\tilde{\epsilon}}{p\tilde{\lambda}_1}\right)^2$.

Since $\operatorname{sign}(\theta_{1,t}) = r_{1,t} \geq r_{1,t-1}^2 \cdot (1 - \overline{\delta}_{t-1})/(1 + \overline{\delta}_{t-1} r_{1,t-1}^2) > 0$ by Proposition 7.12, we have $\theta_{1,t} > 0$. Therefore we can conclude that

$$\|\boldsymbol{\theta_t} - \boldsymbol{v_1}\| = \sqrt{2(1-\theta_{1,t})} \leq \sqrt{2\left(1 - \sqrt{1 - (3\tilde{\epsilon}/(p\tilde{\lambda}_1))^2}\right)} \leq 4\tilde{\epsilon}/(p\tilde{\lambda}_1).$$

Finally,

$$\begin{aligned} |\tilde{T}(\boldsymbol{\theta_t}, \boldsymbol{\theta_t}, \boldsymbol{\theta_t}) - \tilde{\lambda}_1| &= \left| \tilde{\lambda}_1(\theta_{1,t}^3 - 1) + \sum_{i=2}^{k} \tilde{\lambda}_i \theta_{i,t}^3 + \tilde{E}(\boldsymbol{\theta_t}, \boldsymbol{\theta_t}, \boldsymbol{\theta_t}) \right| \\ &\leq \tilde{\lambda}_1\big(1 - \theta_{1,t} + |\theta_{1,t}(1 - \theta_{1,t}^2)|\big) + \max_{i \neq 1} \tilde{\lambda}_i \sqrt{1 - \theta_{1,t}^2} \sum_{i=2}^{k} \theta_{i,t}^2 + \|\tilde{E}(I, \boldsymbol{\theta_t}, \boldsymbol{\theta_t})\| \\ &\leq \frac{(27\kappa \cdot (\tilde{\epsilon}/p\tilde{\lambda}_1)^2 + 2)\tilde{\epsilon}}{p}. \end{aligned}$$

□

7.6.3 Deflation

In this section we give a bound on the additional error caused by deflation. Assuming all previous steps found vectors that are close to the true vectors, then the Lemma below claims the error introduced by deflation depends on the projection of the current vector into the space spanned by the previously found eigenvectors. The benefit of a direction-dependent bound is that once we approach the end of the power method iterations, the vector will be close to the new eigenvector, and therefore has very low projection to the previously found eigenvectors. This allows the algorithm to find more accurate vectors, in particular the guarantee will not deteriorate as the deflation process continues.

Lemma 7.16. *Fix some $\tilde{\epsilon} \geq 0$. Let $\{\boldsymbol{v_1}, \boldsymbol{v_2}, \ldots, \boldsymbol{v_k}\}$ be an orthonormal basis for $\mathbb{R}^k$, and $\lambda_1, \lambda_2, \ldots, \lambda_k \geq 0$ with $\lambda_{\min} := \min_{i \in [k]} \lambda_i$. Also, let $\{\boldsymbol{\hat{v}_1}, \boldsymbol{\hat{v}_2}, \ldots, \boldsymbol{\hat{v}_k}\}$ be a set of unit vectors in $\mathbb{R}^k$ (not necessarily orthogonal), $\hat{\lambda}_1, \hat{\lambda}_2, \ldots, \hat{\lambda}_k \geq 0$ be non-negative scalars, and define*

$$\mathcal{E}_i := \lambda_i \boldsymbol{v_i}^{\otimes 3} - \hat{\lambda}_i \boldsymbol{\hat{v}_i}^{\otimes 3}, \quad i \in [k].$$

Pick any $t \in [k]$. If

$$|\hat{\lambda}_i - \lambda_i| \leq \tilde{\epsilon},$$
$$\|\boldsymbol{\hat{v}_i} - \boldsymbol{v_i}\| \leq \min\{\sqrt{2},\ 2\tilde{\epsilon}/\lambda_i\}$$

for all $i \in [t]$, then for any unit vector $u \in S^{k-1}$, $\tilde{\epsilon} \leq \lambda_{\min}/10000\sqrt{k}$ implies

$$\left\| \sum_{i=1}^{t} \mathcal{E}_i(I, \boldsymbol{u}, \boldsymbol{u}) \right\|_2^2 \leq \left(0.01 + 100 \sum_{i=1}^{t} (\boldsymbol{u}^\top \boldsymbol{v_i})^2 \right) \tilde{\epsilon}^2.$$

Proof. For any unit vector u and $i \in [t]$, the error term

$$\mathcal{E}_i(I, \boldsymbol{u}, \boldsymbol{u}) = \lambda_i (\boldsymbol{u}^\top \boldsymbol{v_i})^2 \boldsymbol{v_i} - \hat{\lambda}_i (\boldsymbol{u}^\top \boldsymbol{\hat{v}_i})^2 \boldsymbol{\hat{v}_i}$$

lives in $\operatorname{span}\{\boldsymbol{v_i}, \boldsymbol{\hat{v}_i}\}$; this space is the same as $\operatorname{span}\{\boldsymbol{v_i}, \boldsymbol{\hat{v}_i}^\perp\}$, where

$$\boldsymbol{\hat{v}_i}^\perp := \boldsymbol{\hat{v}_i} - (\boldsymbol{v_i}^\top \boldsymbol{\hat{v}_i}) \boldsymbol{v_i}$$

is the projection of $\hat{\boldsymbol{v}}_{\boldsymbol{i}}$ onto the subspace orthogonal to $\boldsymbol{v}_{\boldsymbol{i}}$. Since $\|\hat{\boldsymbol{v}}_{\boldsymbol{i}} - \boldsymbol{v}_{\boldsymbol{i}}\|^2 = 2(1 - \boldsymbol{v}_{\boldsymbol{i}}^\top \hat{\boldsymbol{v}}_{\boldsymbol{i}})$, it follows that

$$c_i := \boldsymbol{v}_{\boldsymbol{i}}^\top \hat{\boldsymbol{v}}_{\boldsymbol{i}} = 1 - \|\hat{\boldsymbol{v}}_{\boldsymbol{i}} - \boldsymbol{v}_{\boldsymbol{i}}\|^2/2 \geq 0$$

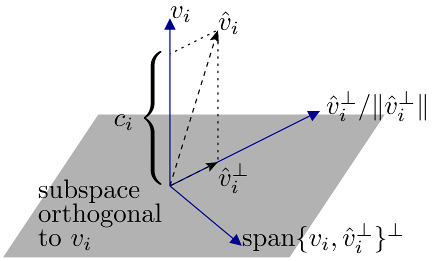
(the inequality follows from the assumption $\|\hat{\boldsymbol{v}}_{\boldsymbol{i}} - \boldsymbol{v}_{\boldsymbol{i}}\| \leq \sqrt{2}$, which in turn implies $0 \leq c_i \leq 1$). By the Pythagorean theorem and the above inequality for c_i,

$$\|\hat{\boldsymbol{v}}_{\boldsymbol{i}}^\perp\|^2 = 1 - c_i^2 \leq \|\hat{\boldsymbol{v}}_{\boldsymbol{i}} - \boldsymbol{v}_{\boldsymbol{i}}\|^2.$$

Later, we will also need the following bound, which is easily derived from the above inequalities and the triangle inequality:

$$|1 - c_i^3| = |1 - c_i + c_i(1 - c_i^2)| \leq 1 - c_i + |c_i(1 - c_i^2)| \leq 1.5\|\hat{\boldsymbol{v}}_{\boldsymbol{i}} - \boldsymbol{v}_{\boldsymbol{i}}\|^2.$$

We now express $\mathcal{E}_i(I, u, u)$ in terms of the coordinate system defined by $\boldsymbol{v}_{\boldsymbol{i}}$ and $\hat{\boldsymbol{v}}_{\boldsymbol{i}}^\perp$, depicted below.


v_i
$\hat{v}_i$
$\hat{v}_i^\perp/\|\hat{v}_i^\perp\|$
c_i
$\hat{v}_i^\perp$
subspace orthogonal to v_i
span$\{v_i, \hat{v}_i^\perp\}^\perp$

Define

$$a_i := \boldsymbol{u}^\top \boldsymbol{v}_{\boldsymbol{i}} \quad \text{and} \quad b_i := \boldsymbol{u}^\top \big(\hat{\boldsymbol{v}}_{\boldsymbol{i}}^\perp / \|\hat{\boldsymbol{v}}_{\boldsymbol{i}}^\perp\|\big).$$

(Note that the part of u living in $\operatorname{span}\{\boldsymbol{v_i}, \boldsymbol{\hat{v}_i^\perp}\}^\perp$ is irrelevant for analyzing $\mathcal{E}_i(I,u,u)$.) We have

$$\begin{aligned}
\mathcal{E}_i(I,u,u) &= \lambda_i(\boldsymbol{u}^\top \boldsymbol{v_i})^2\boldsymbol{v_i} - \hat{\lambda}_i(\boldsymbol{u}^\top \boldsymbol{\hat{v}_i})^2\boldsymbol{\hat{v}_i} \\
&= \lambda_i a_i^2 \boldsymbol{v_i} - \hat{\lambda}_i\big(a_ic_i + \|\boldsymbol{\hat{v}_i^\perp}\|b_i\big)^2\big(c_i\boldsymbol{v_i} + \boldsymbol{\hat{v}_i^\perp}\big) \\
&= \lambda_i a_i^2 \boldsymbol{v_i} - \hat{\lambda}_i\big(a_i^2c_i^2 + 2\|\boldsymbol{\hat{v}_i^\perp}\|a_ib_ic_i + \|\boldsymbol{\hat{v}_i^\perp}\|^2b_i^2\big)c_i\boldsymbol{v_i} - \hat{\lambda}_i\big(a_ic_i + \|\boldsymbol{\hat{v}_i^\perp}\|b_i\big)^2\boldsymbol{\hat{v}_i^\perp} \\
&= \underbrace{\Big((\lambda_i - \hat{\lambda}_ic_i^3)a_i^2 - 2\hat{\lambda}_i\|\boldsymbol{\hat{v}_i^\perp}\|a_ib_ic_i^2 - \hat{\lambda}_i\|\boldsymbol{\hat{v}_i^\perp}\|^2b_i^2c_i\Big)}_{=:A_i}\boldsymbol{v_i} - \underbrace{\hat{\lambda}_i\|\boldsymbol{\hat{v}_i^\perp}\|\big(a_ic_i + \|\boldsymbol{\hat{v}_i^\perp}\|b_i\big)^2}_{=:B_i}\big(\boldsymbol{\hat{v}_i^\perp}/\|\boldsymbol{\hat{v}_i^\perp}\|\big) \\
&= A_i\boldsymbol{v_i} - B_i\big(\boldsymbol{\hat{v}_i^\perp}/\|\boldsymbol{\hat{v}_i^\perp}\|\big).
\end{aligned}$$

The overall error can also be expressed in terms of the A_i and B_i:

$$\begin{aligned}
\Big\|\sum_{i=1}^t \mathcal{E}_i(I,\boldsymbol{u},\boldsymbol{u})\Big\|_2^2 &= \Big\|\sum_{i=1}^t A_i\boldsymbol{v_i} - \sum_{i=1}^t B_i(\boldsymbol{\hat{v}_i^\perp}/\|\boldsymbol{\hat{v}_i^\perp}\|)\Big\|_2^2 \\
&\le 2\Big\|\sum_{i=1}^t A_i\boldsymbol{v_i}\Big\|^2 + 2\Big\|\sum_{i=1}^t B_i(\boldsymbol{\hat{v}_i^\perp}/\|\boldsymbol{\hat{v}_i^\perp}\|)\Big\|_2^2 \\
&\le 2\sum_{i=1}^t A_i^2 + 2\Big(\sum_{i=1}^t |B_i|\Big)^2 \qquad (7.16)
\end{aligned}$$

where the first inequality uses the fact $(x+y)^2 \le 2(x^2+y^2)$ and the triangle inequality, and the second inequality uses the orthonormality of the $\boldsymbol{v_i}$ and the triangle inequality.

It remains to bound A_i^2 and $|B_i|$ in terms of $|a_i|$, λ_i, and $\tilde{\epsilon}$. The first term, A_i^2, can be bounded using the triangle inequality and the various bounds on $|\lambda_i - \hat{\lambda}_i|$, $\|\boldsymbol{\hat{v}_i} - \boldsymbol{v_i}\|$, $\|\boldsymbol{\hat{v}_i^\perp}\|$, and c_i:

$$
\begin{aligned}
|A_i| &\leq (|\lambda_i - \hat{\lambda}_i|c_i^3 + \lambda_i|c_i^3 - 1|)a_i^2 + 2(\lambda_i + |\lambda_i - \hat{\lambda}_i|)\|\boldsymbol{\hat{v}_i^\perp}\||a_i b_i|c_i^2 + (\lambda_i + |\lambda_i - \hat{\lambda}_i|)\|\boldsymbol{\hat{v}_i^\perp}\|^2 b_i^2 c_i \\
&\leq (|\lambda_i - \hat{\lambda}_i| + 1.5\lambda_i\|\boldsymbol{\hat{v}_i} - \boldsymbol{v_i}\|^2 + 2(\lambda_i + |\lambda_i - \hat{\lambda}_i|)\|\boldsymbol{\hat{v}_i} - \boldsymbol{v_i}\|)|a_i| + (\lambda_i + |\lambda_i - \hat{\lambda}_i|)\|\boldsymbol{\hat{v}_i} - \boldsymbol{v_i}\|^2 \\
&\leq (\tilde{\epsilon} + 7\tilde{\epsilon}^2/\lambda_i + 4\tilde{\epsilon} + 4\tilde{\epsilon}^2/\lambda_i)|a_i| + 4\tilde{\epsilon}^2/\lambda_i + \tilde{\epsilon}^3/\lambda_i^2 \\
&= (5 + 11\tilde{\epsilon}/\lambda_i)\tilde{\epsilon}|a_i| + 4(1 + \tilde{\epsilon}/\lambda_i)\tilde{\epsilon}^2/\lambda_i,
\end{aligned}
$$

and therefore (via $(x+y)^2 \leq 2(x^2+y^2)$)

$$
A_i^2 \leq 2(5 + 11\tilde{\epsilon}/\lambda_i)^2\tilde{\epsilon}^2 a_i^2 + 32(1 + \tilde{\epsilon}/\lambda_i)^2\tilde{\epsilon}^4/\lambda_i^2.
$$

The second term, $|B_i|$, is bounded similarly:

$$
\begin{aligned}
|B_i| &\leq 2(\lambda_i + |\lambda_i - \hat{\lambda}_i|)\|\boldsymbol{\hat{v}_i^\perp}\|^2(a_i^2 + \|\boldsymbol{\hat{v}_i^\perp}\|^2) \\
&\leq 2(\lambda_i + |\lambda_i - \hat{\lambda}_i|)\|\boldsymbol{\hat{v}_i} - \boldsymbol{v_i}\|^2(a_i^2 + \|\boldsymbol{\hat{v}_i} - \boldsymbol{v_i}\|^2) \\
&\leq 8(1 + \tilde{\epsilon}/\lambda_i)(\tilde{\epsilon}^2/\lambda_i)a_i^2 + 32(1 + \tilde{\epsilon}/\lambda_i)\tilde{\epsilon}^4/\lambda_i^3.
\end{aligned}
$$

Therefore, using the inequality from (7.16) and again $(x+y)^2 \leq 2(x^2+y^2)$,

$$
\begin{aligned}
\left\|\sum_{i=1}^t \mathcal{E}_i(I, \boldsymbol{u}, \boldsymbol{u})\right\|_2^2 &\leq 2\sum_{i=1}^t A_i^2 + 2\left(\sum_{i=1}^t |B_i|\right)^2 \\
&\leq \left(0.01 + 100\sum_{i=1}^t (\boldsymbol{u}^\top \boldsymbol{v_i})^2\right)\tilde{\epsilon}^2
\end{aligned}
\qquad \square
$$

7.6.4 Proof of the main theorem

Finally we are ready to prove the main theorem in this Chapter. The proof is basically an induction on Theorem 7.15, plugging in the error bound in Theorem 7.16. Ideally we would find the t-th largest eigenvector at the t-th iteration of the algorithm. Theorem 7.10

will guarantee that in one of the random initial vectors we indeed find the top eigenvector. However, additional complication arises because we don't know which random initialization worked!

The solution to this problem is to pick the one vector with the largest $\tilde{T}(v, v, v)$, because the value $\tilde{T}(v, v, v)$ is very similar to the quadratic form $v^T M v$ for matrices. Analog to matrices suggest that having large $\tilde{T}(v, v, v)$ value should imply closeness to a large eigenvector. We show that this intuition is correct during the induction step in the proof below.

Theorem 7.9 restated. *Let $\hat{T} = T + E \in \mathbb{R}^{k\times k\times k}$, where T is a symmetric tensor with orthogonal decomposition $T = \sum_{i=1}^{k} \lambda_i \boldsymbol{v_i}^{\otimes 3}$ where each $\lambda_i > 0$, $\{\boldsymbol{v_1}, \boldsymbol{v_2}, \ldots, \boldsymbol{v_k}\}$ is an orthonormal basis, and E has operator norm $\epsilon := \|E\|$. Define $\lambda_{\min} := \min\{\lambda_i : i \in [k]\}$, and $\lambda_{\max} := \max\{\lambda_i : i \in [k]\}$. For any $\delta > 0$, there exists universal constants $C_1, C_2, C_3 > 0$ such that the following holds. Pick any $\eta \in (0, 1)$, and suppose*

$$\epsilon \le C_1 \cdot \frac{\lambda_{\min}}{k}, \qquad N \ge C_2 \cdot \left(\log(k) + \log\log\Big(\frac{\lambda_{\max}}{\epsilon}\Big)\right),$$

and

$$L = k^{1+\delta} \log(k/\eta)$$

Suppose that Algorithm 7.1 is iteratively called k times, where the input tensor is $\hat{T}$ in the first call, and in each subsequent call, the input tensor is the deflated tensor returned by the previous call. Let $(\hat{v}_1, \hat{\lambda}_1), (\hat{v}_2, \hat{\lambda}_2), \ldots, (\hat{v}_k, \hat{\lambda}_k)$ be the sequence of estimated eigenvector/eigenvalue pairs returned in these k calls. With probability at least $1 - \eta$, there exists a permutation π on $[k]$ such that

$$\|\boldsymbol{v_{\pi(j)}} - \boldsymbol{\hat{v}_j}\| \le 8\epsilon/\lambda_{\pi(j)}, \qquad |\lambda_{\pi(j)} - \hat{\lambda}_j| \le 5\epsilon, \quad \forall j \in [k],$$

and

$$\left\| T - \sum_{j=1}^{k} \hat{\lambda}_j \hat{\boldsymbol{v}}_{\boldsymbol{j}}^{\otimes 3} \right\| \leq 55\epsilon.$$

Proof. We prove by induction that for each $i \in [k]$ (corresponding to the i-th call to Algorithm 7.1), with probability at least $1 - i\eta/k$, there exists a permutation π on $[k]$ such that the following assertions hold.

1. For all $j \leq i$, $\|\boldsymbol{v}_{\boldsymbol{\pi(j)}} - \hat{\boldsymbol{v}}_{\boldsymbol{j}}\| \leq 8\epsilon/\lambda_{\pi(j)}$ and $|\lambda_{\pi(j)} - \hat{\lambda}_j| \leq 12\epsilon$.

2. The error tensor

$$\tilde{E}_{i+1} := \left(\hat{T} - \sum_{j \leq i} \hat{\lambda}_j \hat{\boldsymbol{v}}_{\boldsymbol{j}}^{\otimes 3} \right) - \sum_{j \geq i+1} \lambda_{\pi(j)} \boldsymbol{v}_{\boldsymbol{\pi(j)}}^{\otimes 3} = E + \sum_{j \leq i} \left(\lambda_{\pi(j)} \boldsymbol{v}_{\boldsymbol{\pi(j)}}^{\otimes 3} - \hat{\lambda}_j \hat{\boldsymbol{v}}_{\boldsymbol{j}}^{\otimes 3} \right)$$

satisfies

$$\|\tilde{E}_{i+1}(I, \boldsymbol{u}, \boldsymbol{u})\| \leq 56\epsilon, \quad \forall \boldsymbol{u} \in S^{k-1}; \tag{7.17}$$

$$\|\tilde{E}_{i+1}(I, \boldsymbol{u}, \boldsymbol{u})\| \leq 2\epsilon, \quad \forall u \in S^{k-1} \text{ s.t. } \exists j \geq i+1, (\boldsymbol{u}^\top \boldsymbol{v}_{\boldsymbol{\pi(j)}})^2 \geq 1 - (168\epsilon/\lambda_{\pi(j)})^2. \tag{7.18}$$

We actually take $i = 0$ as the base case, so we can ignore the first assertion, and just observe that for $i = 0$,

$$\tilde{E}_1 = \hat{T} - \sum_{j=1}^{k} \lambda_i \boldsymbol{v}_{\boldsymbol{i}}^{\otimes 3} = E.$$

We have $\|\tilde{E}_1\| = \|E\| = \epsilon$, and therefore the second assertion holds.

Now fix some $i \in [k]$, and assume as the inductive hypothesis that, with probability at least $1 - (i-1)\eta/k$, there exists a permutation π such that two assertions above hold for $i - 1$ (call this Event_{i-1}). The i-th call to Algorithm 7.1 takes as input

$$\tilde{T}_i := \hat{T} - \sum_{j \leq i-1} \hat{\lambda}_j \hat{\boldsymbol{v}}_{\boldsymbol{j}}^{\otimes 3},$$

which is intended to be an approximation to

$$T_i := \sum_{j \geq i} \lambda_{\pi(j)} \boldsymbol{v}_{\boldsymbol{\pi(j)}}^{\otimes 3}.$$

Observe that

$$\tilde{T}_i - T_i = \tilde{E}_i,$$

which satisfies the second assertion in the inductive hypothesis. We may write $T_i = \sum_{l=1}^{k} \tilde{\lambda}_l v_l^{\otimes 3}$ where $\tilde{\lambda}_l = \lambda_l$ whenever $\pi^{-1}(l) \geq i$, and $\tilde{\lambda}_l = 0$ whenever $\pi^{-1}(l) \leq i-1$. This form is used when referring to $\tilde{T}$ or the $\tilde{\lambda}_i$ in preceding lemmas (in particular, Lemma 7.10 and Lemma 7.15).

By Lemma 7.10, with conditional probability at least $1 - \eta/k$ given Event_{i-1}, at least one of $\boldsymbol{\theta}_{\boldsymbol{0}}^{(\tau)}$ for $\tau \in [L]$ is γ-separated relative to $\pi(j_{\max})$, where $j_{\max} := \arg\max_{j \geq i} \lambda_{\pi(j)}$, (for $\gamma = 0.01$; call this Event_i'; note that the application of Lemma 7.10 determines C_3). Therefore $\Pr[\mathsf{Event}_{i-1} \cap \mathsf{Event}_i'] = \Pr[\mathsf{Event}_i' | \mathsf{Event}_{i-1}] \Pr[\mathsf{Event}_{i-1}] \geq (1 - \eta/k)(1 - (i-1)\eta/k) \geq 1 - i\eta/k$. It remains to show that $\mathsf{Event}_{i-1} \cap \mathsf{Event}_i' \subseteq \mathsf{Event}_i$; so henceforth we condition on $\mathsf{Event}_{i-1} \cap \mathsf{Event}_i'$.

Set C_1 to be a universal constant that is small enough. For all $\tau \in [L]$ such that $\boldsymbol{\theta}_{\boldsymbol{0}}^{(\tau)}$ is γ-separated relative to $\pi(j_{\max})$, we have (i) $|\theta_{j_{\max},0}^{(\tau)}| \geq 1/\sqrt{k}$, and (ii) that by Lemma 7.15 (using $\tilde{\epsilon}/p := 2\epsilon$, and $i^* := \pi(j_{\max})$, $C = C_2$),

$$|\tilde{T}_i(\boldsymbol{\theta}_{\boldsymbol{N}}^{(\tau)}, \boldsymbol{\theta}_{\boldsymbol{N}}^{(\tau)}, \boldsymbol{\theta}_{\boldsymbol{N}}^{(\tau)}) - \lambda_{\pi(j_{\max})}| \leq 5\epsilon$$

(notice by definition that $\gamma \geq 1/100$ implies $\gamma_0 \geq 1 - /(1+\gamma) \geq 1/101$, thus it follows from the bounds on the other quantities that $\tilde{\epsilon} = 2p\epsilon \leq 56 C_1 \cdot \frac{\lambda_{\min}}{k} < \frac{\gamma_0}{2(1+8\kappa)} \cdot \tilde{\lambda}_{\min} \cdot \theta_{i^*,0}^2$ as necessary). Therefore $\boldsymbol{\theta_N} := \boldsymbol{\theta_N^{(\tau^*)}}$ must satisfy

$$\tilde{T}_i(\boldsymbol{\theta_N}, \boldsymbol{\theta_N}, \boldsymbol{\theta_N}) = \max_{\tau \in [L]} \tilde{T}_i(\boldsymbol{\theta_N^{(\tau)}}, \boldsymbol{\theta_N^{(\tau)}}, \boldsymbol{\theta_N^{(\tau)}}) \geq \max_{j \geq i} \lambda_{\pi(j)} - 5\epsilon = \lambda_{\pi(j_{\max})} - 5\epsilon.$$

On the other hand, by the triangle inequality,

$$\begin{aligned}\tilde{T}_i(\boldsymbol{\theta_N}, \boldsymbol{\theta_N}, \boldsymbol{\theta_N}) &\leq \sum_{j \geq i} \lambda_{\pi(j)} \theta^3_{\pi(j),N} + |\tilde{E}_i(\boldsymbol{\theta_N}, \boldsymbol{\theta_N}, \boldsymbol{\theta_N})| \\ &\leq \sum_{j \geq i} \lambda_{\pi(j)} |\theta_{\pi(j),N}| \theta^2_{\pi(j),N} + 56\epsilon \\ &\leq \lambda_{\pi(j^*)} |\theta_{\pi(j^*),N}| + 56\epsilon\end{aligned}$$

where $j^* := \arg\max_{j \geq i} \lambda_{\pi(j)} |\theta_{\pi(j),N}|$. Therefore

$$\lambda_{\pi(j^*)} |\theta_{\pi(j^*),N}| \geq \lambda_{\pi(j_{\max})} - 5\epsilon - 56\epsilon \geq \frac{4}{5} \lambda_{\pi(j_{\max})}.$$

Squaring both sides and using the fact that $\theta^2_{\pi(j^*),N} + \theta^2_{\pi(j),N} \leq 1$ for any $j \neq j^*$,

$$\begin{aligned}\left(\lambda_{\pi(j^*)} \theta_{\pi(j^*),N}\right)^2 &\geq \frac{16}{25} \left(\lambda_{\pi(j_{\max})} \theta_{\pi(j^*),N}\right)^2 + \frac{16}{25} \left(\lambda_{\pi(j_{\max})} \theta_{\pi(j),N}\right)^2 \\ &\geq \frac{16}{25} \left(\lambda_{\pi(j^*)} \theta_{\pi(j^*),N}\right)^2 + \frac{16}{25} \left(\lambda_{\pi(j)} \theta_{\pi(j),N}\right)^2\end{aligned}$$

which in turn implies

$$\lambda_{\pi(j)} |\theta_{\pi(j),N}| \leq \frac{3}{4} \lambda_{\pi(j^*)} |\theta_{\pi(j^*),N}|, \quad j \neq j^*.$$

This means that θ_N is (1/4)-separated relative to $\pi(j^*)$. Also, observe that

$$|\theta_{\pi(j^*),N}| \geq \frac{4}{5} \cdot \frac{\lambda_{\pi(j_{\max})}}{\lambda_{\pi(j^*)}} \geq \frac{4}{5}, \quad \frac{\lambda_{\pi(j_{\max})}}{\lambda_{\pi(j^*)}} \leq \frac{5}{4}.$$

Therefore we are in a situation that is very similar to Theorem 7.14, except that now the dominating $|\lambda_{j^*}\theta_{j^*}|$ may not have the same j^* as the largest $j_{\max}$. The proof of Theorem 7.14 can be easily adapted in this case because we still know $\frac{\lambda_{\pi(j_{\max})}}{\lambda_{\pi(j^*)}} \leq \frac{5}{4}$, hence we can still apply Theorem 7.15 and get

$$\|\hat{\boldsymbol{\theta}} - \boldsymbol{v_{\pi(j^*)}}\| \leq \frac{8\epsilon}{\lambda_{\pi(j^*)}}, \qquad |\hat{\lambda} - \lambda_{\pi(j^*)}| \leq 5\epsilon.$$

Since $\hat{\boldsymbol{v}}_{\boldsymbol{i}} = \hat{\boldsymbol{\theta}}$ and $\hat{\lambda}_i = \hat{\lambda}$, the first assertion of the inductive hypothesis is satisfied, as we can modify the permutation π by swapping $\pi(i)$ and $\pi(j^*)$ without affecting the values of $\{\pi(j) : j \leq i-1\}$ (recall $j^* \geq i$).

We now argue that $\tilde{E}_{i+1}$ has the required properties to complete the inductive step. By Lemma 7.16 (using $\tilde{\epsilon} := 5\epsilon$), we have for any unit vector $u \in S^{k-1}$,

$$\left\| \left(\sum_{j\leq i} \Big(\lambda_{\pi(j)} \boldsymbol{v}_{\boldsymbol{\pi(j)}}^{\otimes 3} - \hat{\lambda}_j \hat{\boldsymbol{v}}_{\boldsymbol{j}}^{\otimes 3} \Big) \right) (I, \boldsymbol{u}, \boldsymbol{u}) \right\| \leq \left(1/50 + 100 \sum_{j=1}^{i} (\boldsymbol{u}^\top \boldsymbol{v}_{\boldsymbol{\pi(j)}})^2 \right)^{1/2} 5\epsilon \leq 55\epsilon. \quad (7.19)$$

Therefore by the triangle inequality,

$$\|\tilde{E}_{i+1}(I, \boldsymbol{u}, \boldsymbol{u})\| \leq \|E(I, \boldsymbol{u}, \boldsymbol{u})\| + \left\| \left(\sum_{j\leq i} \Big(\lambda_{\pi(j)} \boldsymbol{v}_{\boldsymbol{\pi(j)}}^{\otimes 3} - \hat{\lambda}_j \hat{\boldsymbol{v}}_{\boldsymbol{j}}^{\otimes 3} \Big) \right) (I, \boldsymbol{u}, \boldsymbol{u}) \right\| \leq 56\epsilon.$$

Thus the bound (7.17) holds.

To prove that (7.18) holds, pick any unit vector $u \in S^{k-1}$ such that there exists $j' \geq i+1$ with $(\boldsymbol{u}^\top \boldsymbol{v}_{\boldsymbol{\pi(j')}})^2 \geq 1 - (168\epsilon/\lambda_{\pi(j')})^2$. We have (via the assumed bound $\epsilon \leq C_1 \cdot \frac{\lambda_{\min}}{k}$)

$$100 \sum_{j=1}^{i} (\boldsymbol{u}^\top \boldsymbol{v}_{\boldsymbol{\pi(j)}})^2 \leq 100 \Big(1 - (\boldsymbol{u}^\top \boldsymbol{v}_{\boldsymbol{\pi(j')}})^2 \Big) \leq 100 \left(\frac{168\epsilon}{\lambda_{\pi(j')}} \right)^2 \leq \frac{1}{50},$$

and therefore

$$\left(1/50 + 100\sum_{j=1}^{i}(\boldsymbol{u}^\top \boldsymbol{v}_{\boldsymbol{\pi(j)}})^2\right)^{1/2} 5\epsilon \le (1/50 + 1/50)^{1/2} 5\epsilon \le \epsilon.$$

By the triangle inequality, we have $\|\tilde{E}_{i+1}(I, \boldsymbol{u}, \boldsymbol{u})\| \le 2\epsilon$. Therefore (7.18) holds, so the second assertion of the inductive hypothesis holds. Thus $\mathsf{Event}_{i-1} \cap \mathsf{Event}'_i \subseteq \mathsf{Event}_i$, and $\Pr[\mathsf{Event}_i] \ge \Pr[\mathsf{Event}_{i-1} \cap \mathsf{Event}'_i] \ge 1 - i\eta/k$. We conclude that by the induction principle, there exists a permutation π such that two assertions hold for $i = k$, with probability at least $1 - \eta$.

From the last induction step ($i = k$), it is also clear from (7.19) that $\|T - \sum_{j=1}^{k} \hat{\lambda}_j \hat{\boldsymbol{v}}_{\boldsymbol{j}}^{\otimes 3}\| \le 55\epsilon$ (in $\mathsf{Event}_{k-1} \cap \mathsf{Event}'_k$). This completes the proof of the theorem. □

7.7 Better Initializers and Adaptive Deflation

If we take a closer look at the analysis of the tensor power method, it is easy to see the requirement of error smaller than $\lambda_{\min}/k$ is only caused by Theorem 7.15, in particular in the requirement that the relative error δ_0 should be a small constant. For example, the deflation step, Theorem 7.16 actually requires error only smaller than $\lambda_{\min}/\sqrt{k}$.

In some situations, it is possible to obtain much better initializers. The exchangeable single topic model is a good example for this idea: for a single document, the expected word frequency vector $\boldsymbol{v}$ will be $\boldsymbol{\mu_i}$ if the document has topic i. When the document is long enough it is easy to show that $\boldsymbol{v}$ is actually very close to its expectation. If we apply the reduction in Section 7.2.1, then $W^T\boldsymbol{v}$ will be close to one of the component in the tensor decomposition. If we have many such good initializers, then it is possible to do orthogonal tensor decomposition even when the error is more than $\lambda_{\min}/k$.

To make this precise, define a vector to be (γ, Q)-good respect to i if $\boldsymbol{u} \cdot \boldsymbol{v_i} > Q$ and $|\boldsymbol{u} \cdot \boldsymbol{v_i}| - \max_{j\neq i} |\boldsymbol{u} \cdot \boldsymbol{v_j}| > \gamma|\boldsymbol{u} \cdot \boldsymbol{v_i}|$. Theorem 7.15 shows if the initializer is (γ, Q)-good with respect to the top eigenvector, we only require $\tilde{\epsilon} < \lambda_{\min} Q^2 \gamma/18$. This is really good as if Q and γ are both constants, $\tilde{\epsilon}$ is only required to be smaller than a small constant times $\hat{\lambda}_{\max}$.

When Q and γ are large, it is possible that the deflation step becomes the bottleneck as it still requires ϵ to be smaller than $\lambda_{\min}/\sqrt{k}$. In order to avoid this we need to do deflation in a more careful way, as in Algorithm 7.3

Algorithm 7.3. $\{\lambda, \Phi\} \leftarrow$TensorEigen$(T, \{\boldsymbol{v_i}\}_{i\in[L]}, N)$

Input: Tensor $T \in \mathbb{R}^{k\times k\times k}$, set of L initialization vectors $\{\boldsymbol{v_i}\}_{i\in L}$, number of iterations N.
Output: the estimated eigenvalue/eigenvector pairs $\{\hat{\lambda}_i, \hat{\boldsymbol{v}}_{\boldsymbol{i}}\}$, where $\hat{\lambda}$ are the eigenvalues and $\hat{\boldsymbol{v}}_{\boldsymbol{i}}$'s are the eigenvectors.

```
for $i = 1$ to $k$ do
  for $\tau = 1$ to $L$ do
    $\boldsymbol{\theta_0} \leftarrow v_\tau$.
    for $t = 1$ to $N$ do
      $\tilde{T} \leftarrow T$.
      for $j = 1$ to $i - 1$ (when $i > 1$) do
        if $|\hat{\lambda}_j \boldsymbol{\theta_t^{(\tau)}} \cdot \hat{\boldsymbol{v}}_{\boldsymbol{j}}| > \xi = 5\tilde{\epsilon}$ then
          $\tilde{T} \leftarrow \tilde{T} - \hat{\lambda}_j \hat{\boldsymbol{v}}_j^{\otimes 3}$.
        end if
      end for
      Compute power iteration update $\boldsymbol{\theta_t}^{(\tau)} := \frac{\tilde{T}(I, \boldsymbol{\theta_{t-1}^{(\tau)}}, \boldsymbol{\theta_{t-1}^{(\tau)}})}{\|\tilde{T}(I, \boldsymbol{\theta_{t-1}^{(\tau)}}, \boldsymbol{\theta_{t-1}^{(\tau)}})\|}$
    end for
  end for
  Let $\tau^* := \arg\max_{\tau\in L}\{\tilde{T}(\boldsymbol{\theta_N^{(\tau)}}, \boldsymbol{\theta_N^{(\tau)}}, \boldsymbol{\theta_N^{(\tau)}})\}$.
  Do $N$ power iteration updates starting from $\boldsymbol{\theta_N^{(\tau^*)}}$ to obtain eigenvector estimate $\hat{\boldsymbol{v}}_{\boldsymbol{i}}$,
  and set $\hat{\lambda}_i := \tilde{T}(\hat{\boldsymbol{v}}_{\boldsymbol{i}}, \hat{\boldsymbol{v}}_{\boldsymbol{i}}, \hat{\boldsymbol{v}}_{\boldsymbol{i}})$.
end for
return the estimated eigenvalue/eigenvectors $(\hat{\lambda}_i, \hat{\boldsymbol{v}}_{\boldsymbol{i}})$.
```

The main difference between Algorithm 7.3 and Algorithm 7.1 is in the deflation step: instead of the one shot deflation in Algorithm 7.1 (just subtract the found component from the tensor), in Algorithm 7.3 we are more careful in that we only subtract the component when it "threats" the current iteration (that is, the $|\lambda_i \theta_{i,t}|$ value is larger or almost as large as $|\lambda_j \theta_{j,t}|$ when we are trying to converge to $\boldsymbol{v_j}$).

Lemma 7.17 (Deflation analysis)**.** *Let $\tilde{\epsilon} > 0$ and let $\{\boldsymbol{v_1}, \dots, \boldsymbol{v_k}\}$ be an orthonormal basis for $\mathbb{R}^k$ and $\lambda_i \geq 0$ for $i \in [k]$. Let $\{\boldsymbol{\hat{v}_1}, \dots, \boldsymbol{\hat{v}_k}\} \in \mathbb{R}^k$ be a set of unit vectors and $\hat{\lambda}_i \geq 0$. Define third order tensor $\mathcal{E}_i$ such that*

$$\mathcal{E}_i := \lambda_i \boldsymbol{v_i}^{\otimes 3} - \hat{\lambda}_i \boldsymbol{\hat{v}_i}^{\otimes 3}, \quad \forall\, i \in k.$$

For some $t \in [k]$ and a unit vector $u \in S^{k-1}$ such that $\boldsymbol{u} = \sum_{i \in [k]} \theta_i \boldsymbol{v_i}$ and $\hat{\theta}_i := \boldsymbol{u} \cdot \boldsymbol{\hat{v}_i}$, we have for $i \in [t]$,

$$|\hat{\lambda}_i \hat{\theta}_i| \geq \xi \geq 5\tilde{\epsilon},$$

$$|\hat{\lambda}_i - \lambda_i| \leq \tilde{\epsilon},$$

$$\|\boldsymbol{\hat{v}_i} - \boldsymbol{v_i}\| \leq \min\{\sqrt{2},\ 2\tilde{\epsilon}/\lambda_i\},$$

then when $\tilde{\epsilon} \leq 10^{-5}\lambda_{\min}$ implies

$$\left\| \sum_{i=1}^{t} \mathcal{E}_i(I, \boldsymbol{u}, \boldsymbol{u}) \right\|_2^2 \leq 200\tilde{\epsilon}^2 \sum_{i=1}^{t} \theta_i^2.$$

Proof: The proof is on lines of deflation analysis in Theorem 7.16, but we improve the bounds based on additional properties of vector u. From Theorem 7.16, we have that for all $i \in [t]$, and any unit vector u,

$$\left\| \sum_{i=1}^{t} \mathcal{E}_i(I, \boldsymbol{u}, \boldsymbol{u}) \right\|_2^2 \leq \left(O(\sum_{i=1}^{t} \tilde{\epsilon}^2/\lambda_i^2) + 100 \sum_{i=1}^{t} \theta_i^2 \right) \tilde{\epsilon}^2 \tag{7.20}$$

Let $\hat{\lambda}_i = \lambda_i + \delta_i$ and $\hat{\theta}_i = \theta_i + \beta_i$. We have $\delta_i \leq \tilde{\epsilon}$ and $\beta_i \leq 2\tilde{\epsilon}/\lambda_i$, and that $|\hat{\lambda}_i \hat{\theta}_i| \geq \xi$.

$$||\hat{\lambda}_i \hat{\theta}_i| - |\lambda_i \theta_i|| \leq |\hat{\lambda}_i \hat{\theta}_i - \lambda_i \theta_i| \leq |(\lambda_i + \delta_i)(\theta_i + \beta_i) - \lambda_i \theta_i| \leq 4\tilde{\epsilon}. \tag{7.21}$$

Thus, we have that $|\lambda_i \theta_i| \geq 5\tilde{\epsilon} - 4\tilde{\epsilon} = \tilde{\epsilon}$. Thus $\sum_{i=1}^{t} \tilde{\epsilon}^2/\lambda_i^2 \leq \sum_i \theta_i^2$. Substituting in (7.20), we have the result. □

We also need to show that the deflation process indeed deflated all the previously found eigenvectors that has a large effective eigenvalue. This is again by (7.21), if $|\lambda_i \theta_i| \geq 9\tilde{\epsilon}$, then $|\hat{\lambda}_i \hat{\theta}_i| \geq 5\tilde{\epsilon}$ and will get deflated. On the other hand, the effective eigenvalue of the vector we want to converge to is at least $\lambda_{min} Q \gg 5\tilde{\epsilon}$.

Chapter 8

Tensor Decomposition in Community Detection

Using the tensor decomposition framework, it is possible to learn many latent variable models. In this chapter we use *mixed membership community model* as an example. The new algorithm in this section illustrates many useful techniques for proving sharp sample complexity bounds using tensor power method. In particular, the sample complexity bounds we get almost match the state-of-art for stochastic block model (which is a special case of mixed membership model), even though our algorithm works for more general models.

8.1 Background and Main Results

8.1.1 History of Overlapping Communities

Studying communities forms an integral part of social network analysis. A community generally refers to a group of individuals with shared interests (e.g. music, sports), or relationships (e.g. friends, co-workers). Community formation in social networks has been studied by many sociologists, e.g. [45, 108, 124, 129], starting with the seminal work of [129].

Various probabilistic and non-probabilistic network models attempt to explain community formation. In addition, they also attempt to quantify interactions and the extent of overlap between different communities, relative sizes among the communities, and various other network properties. Studying such community models are also of interest in other domains, e.g. in biological networks.

While there exists a vast literature on community models, learning these models is typically challenging, and various heuristics such as Markov Chain Monte Carlo (MCMC) or variational expectation maximization (EM) are employed in practice. Such heuristics tend to scale poorly for large networks. On the other hand, community models with guaranteed learning methods tend to be restrictive. A popular class of probabilistic models, termed as *stochastic blockmodels*, have been widely studied and enjoy strong theoretical learning guarantees, e.g. [62, 84, 125, 142, 159, 161]. On the other hand, they posit that an individual belongs to a single community, which does not hold in most real settings [135].

The mixed membership community model of [4] has a number of attractive properties. It retains many of the convenient properties of the stochastic block model. For instance, conditional independence of the edges is assumed, given the community memberships of the nodes in the network. At the same time, it allows for communities to overlap, and for every individual to be fractionally involved in different communities.

8.1.2 Main Result

In this section, we describe the mixed membership community model based on Dirichlet priors for the community draws by the individuals. Then we state our main result.

We first introduce the special case of the popular stochastic block model, where each node belongs to a single community.

Notation: Let G be the $\{0,1\}$ adjacency matrix for the random network and let $G_{A,B}$ be the submatrix of G corresponding to rows $A \subseteq [n]$ and columns $B \subseteq [n]$. We consider models with k underlying (hidden) communities. For node i, let $\boldsymbol{\pi_i} \in \mathbb{R}^k$ denote its *community membership vector*, i.e., the vector is supported on the communities to which the node belongs. In the special case of the popular stochastic block model described below, $\boldsymbol{\pi_i}$ is a basis coordinate vector, while the more general mixed membership model relaxes this assumption and a node can be in multiple communities with fractional memberships. Define $\Pi := [\boldsymbol{\pi_1}|\boldsymbol{\pi_2}|\cdots|\boldsymbol{\pi_n}] \in \mathbb{R}^{k\times n}$. and let $\Pi_A := [\boldsymbol{\pi_i} : i \in A] \in \mathbb{R}^{k\times|A|}$ denote the set of column vectors restricted to $A \subseteq [n]$. For a matrix M, recall that $(M)_i$ and $(M)^i$ denote its i^{th} column and row respectively.

Stochastic block model (special case): In this model, each individual is independently assigned to a single community, chosen at random: each node i chooses community j independently with probability $\widehat{\alpha}_j$, for $i \in [n], j \in [k]$, and we assign $\boldsymbol{\pi_i} = \boldsymbol{e_j}$ in this case, where $\boldsymbol{e_j} \in \{0,1\}^k$ is the j^{th} coordinate basis vector. Given the community assignments Π, every directed[1] edge in the network is independently drawn: if node u is in community i and node v is in community j (and $u \neq v$), then the probability of having the edge (u,v) in the network is $P_{i,j}$. Here, $P \in [0,1]^{k\times k}$ and we refer to it as the *community connectivity matrix*. This implies that given the community membership vectors $\boldsymbol{\pi_u}$ and $\boldsymbol{\pi_v}$, the probability of an edge from u to v is $\boldsymbol{\pi_u}^\top P \boldsymbol{\pi_v}$ (since when $\boldsymbol{\pi_u} = \boldsymbol{e_i}$ and $\boldsymbol{\pi_v} = \boldsymbol{e_j}$, we have $\boldsymbol{\pi_u}^\top P \boldsymbol{\pi_v} = P_{i,j}$.).

Throughout this Chapter we work in the special case where $P_{i,i} = p$ and $P_{i,j} = q$ when $i \neq j$. Here p, q are two probabilities that two individual know each other when they are in/not in the same community respectively. Also, we assume all the communities have equal size, that is, $\widehat{\alpha}_i = 1/k$ for all $i \in [k]$.

[1] We limit our discussion to directed networks in this paper, but note that the results also hold for undirected community models, where P is a symmetric matrix, and an edge (u,v) is formed with probability $\boldsymbol{\pi_u}^\top P \boldsymbol{\pi_v} = \boldsymbol{\pi_v}^\top P \boldsymbol{\pi_u}$.

Mixed membership model: We now consider the extension of the stochastic block model which allows for an individual to belong to multiple communities and yet preserves some of the convenient independence assumptions of the block model. In this model, the community membership vector $\boldsymbol{\pi_u}$ at node u is a probability vector, i.e., $\sum_{i\in[k]} \boldsymbol{\pi_u}(i) = 1$, for all $u \in [n]$. Given the community membership vectors, the generation of the edges is identical to the block model: given vectors $\boldsymbol{\pi_u}$ and $\boldsymbol{\pi_v}$, the probability of an edge from u to v is $\boldsymbol{\pi_u}^\top P \boldsymbol{\pi_v}$, and the edges are independently drawn. This formulation allows for the nodes to be in multiple communities, and at the same time, preserves the conditional independence of the edges, given the community memberships of the nodes.

Dirichlet prior for community membership: The only aspect left to be specified for the mixed membership model is the distribution from which the community membership vectors Π are drawn. We consider the popular setting of [4, 71], where the community vectors $\{\boldsymbol{\pi_u}\}$ are i.i.d. draws from the Dirichlet distribution, denoted by $\mathrm{Dir}(\alpha)$, with parameter vector $\alpha \in \mathbb{R}^k_{>0}$. The probability density function of the Dirichlet distribution is given by

$$\mathbb{P}[\pi] = \frac{\prod_{i=1}^k \Gamma(\alpha_i)}{\Gamma(\alpha_0)} \prod_{i=1}^k \pi_i^{\alpha_i - 1}, \quad \pi \sim \mathrm{Dir}(\alpha), \alpha_0 := \sum_i \alpha_i, \tag{8.1}$$

where $\Gamma(\cdot)$ is the Gamma function and the ratio of the Gamma function serves as the normalization constant.

The Dirichlet distribution is widely employed for specifying priors in Bayesian statistics, e.g. latent Dirichlet allocation [26]. The Dirichlet distribution is the conjugate prior of the multinomial distribution which makes it attractive for Bayesian inference.

Let $\widehat{\alpha}$ denote the normalized parameter vector α/α_0, where $\alpha_0 := \sum_i \alpha_i$. In particular, note that $\widehat{\alpha}$ is a probability vector: $\sum_i \widehat{\alpha}_i = 1$. Intuitively, $\widehat{\alpha}$ denotes the relative expected sizes of the communities (since $\mathbb{E}[n^{-1} \sum_{u\in[n]} \boldsymbol{\pi_u}[i]] = \widehat{\alpha}_i$). Again in this Chapter we work with the simplified case where $\alpha_i = \alpha_0/k$ for all i.

The stochastic block model is a limiting case of the mixed membership model when the Dirichlet parameter is $\alpha = \alpha_0 \cdot \widehat{\alpha}$, where the probability vector $\widehat{\alpha}$ is held fixed and $\alpha_0 \to 0$. In the other extreme when $\alpha_0 \to \infty$, the Dirichlet distribution becomes peaked around a single point, for instance, if $\alpha_i \equiv c$ and $c \to \infty$, the Dirichlet distribution is peaked at $k^{-1} \cdot \mathbf{1}$, where $\mathbf{1}$ is the all-ones vector. Thus, the parameter α_0 serves as a measure of the average sparsity of the Dirichlet draws or equivalently, of how concentrated the Dirichlet measure is along the different coordinates. This in effect, controls the extent of overlap among different communities.

Specific Parameter Choices in Homophilic Models A sub-class of community model are those satisfying *homophily.* Homophily or the tendency to form edges within the members of the same community has been posited as an important factor in community formation, especially in social settings. Many of the existing learning algorithms, e.g. [166] require this assumption to provide guarantees in the stochastic block model setting.

Throughout this Chapter we deal with a standard special case, where the diagonal entries of P are all equal to p, and off-diagonal entries are all equal to $q < p$. All the communities have the same (expected size), which means $\widehat{\alpha}_i = 1/k$ for all $i \in [k]$. The parameter α_0 should generally be thought of as a constant (which correspond to the case that everyone is expected to be in constant number of communities). In particular, our $\tilde{O}$ and $\tilde{\Omega}$ notations will ignore polynomial powers of $\alpha_0 + 1$.

Main Result Informally, the main result of this chapter is

Theorem 8.1 (Informal)**.** *There is an algorithm for estimating the parameters of Mixed Membership Stochastic Block model, when everyone is expected to be in constant number of communities, and communities have similar sizes, the guarantee of the algorithm is matches the state-of-art result for the simpler Stochastic Block model.*

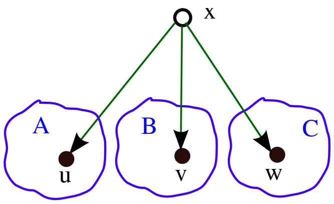


Figure 8.1: Our moment-based learning algorithm uses 3-star count tensor from set X to sets A, B, C (and the roles of the sets are interchanged to get various estimates). Specifically, T is a third order tensor, where $\mathrm{T}(u, v, w)$ is the normalized count of the 3-stars with u, v, w as leaves over all $x \in X$.

The rest of this chapter is structured as follows: we show what graph moments we use and give the structure of the graph moments in Section 8.2. In Section 8.3, we describe learning algorithms when we have exact moments (just to give intuition) and how this can be adapted to the case when we are only given samples. The next Section 8.4 states the performance guarantee of our algorithms, and outlines the proof. All the details of the proof are deferred to the final Section 8.5.

Implementation In a subsequent work, Huang et al. [89] implemented the algorithm using GPUs. The algorithm scales up to networks with more than 100,000 nodes and 500 communities, and produces reasonable results.

8.2 Graph Moments Under Mixed Membership Models

Our approach for learning a mixed membership community model relies on the form of the graph moments[2] under the mixed membership model. We now describe the specific graph moments used by our learning algorithm (based on 3-star and edge counts) and provide explicit forms for the moments, assuming draws from a mixed membership model.

[2] We interchangeably use the term first order moments for edge counts and third order moments for 3-star counts.

To simplify notation we define

$$F := \Pi^\top P^\top = [\boldsymbol{\pi_1}|\boldsymbol{\pi_2}|\cdots|\boldsymbol{\pi_n}]^\top P^\top. \tag{8.2}$$

For a subset $A \subseteq [n]$ of individuals, let $F_A \in \mathbb{R}^{|A|\times k}$ denote the submatrix of F corresponding to nodes in A, *i.e.*, $F_A := \Pi_A^\top P^\top$.

Graph Moments under Mixed Membership Dirichlet Model

3-star counts: The primary quantity of interest is a third-order tensor which counts the number of 3-stars. A 3-star is a star graph with three leaves $\{a, b, c\}$ and we refer to the internal node x of the star as its "head", and denote the structure by $x \to \{a, b, c\}$ (see figure 8.1). We partition the network into four[3] parts and consider 3-stars such that each node in the 3-star belongs to a different partition. Specifically, consider[4] a partition A, B, C, X of the network. We count the number of 3-stars from X to A, B, C and our quantity of interest is

$$\mathrm{T}_{X\to\{A,B,C\}} := \frac{1}{|X|}\sum_{i\in X}[G_{i,A}^\top \otimes G_{i,B}^\top \otimes G_{i,C}^\top]. \tag{8.3}$$

$\mathrm{T} \in \mathbb{R}^{|A|\times|B|\times|C|}$ is a third order tensor, and an element of the tensor is given by

$$\mathrm{T}_{X\to\{A,B,C\}}(a, b, c) = \frac{1}{|X|}\sum_{x\in X} G(x, a)G(x, b)G(x, c), \quad \forall a \in A, b \in B, c \in C, \tag{8.4}$$

which is the normalized count of the number of 3-stars with leaves a, b, c such that its "head" is in set X.

[3] For sample complexity analysis, we require dividing the graph into more than four partitions to deal with statistical dependency issues, and we outline it in Section 8.3.

[4] For our theoretical guarantees to hold, the partitions A, B, C, X can be randomly chosen and are of size $\Theta(n)$.

We now analyze the graph moments for the general mixed membership Dirichlet model. Instead of the raw moments (i.e. edge and 3-star counts), we consider modified moments to obtain similar expressions as in the case of the stochastic block model.

Let $\mu_{X\to A} \in \mathbb{R}^{|A|}$ denote a vector which gives the normalized count of edges from X to A:

$$\boldsymbol{\mu_{X\to A}} := \frac{1}{|X|}\sum_{i\in X}[G_{i,A}^{\top}]. \tag{8.5}$$

We now define a modified adjacency matrix[5] $G^{\alpha_0}_{X,A}$ as

$$G^{\alpha_0}_{X,A} := \left(\sqrt{\alpha_0+1}G_{X,A} - (\sqrt{\alpha_0+1}-1)\mathbf{1}\boldsymbol{\mu}^{\top}_{\boldsymbol{X\to A}}\right). \tag{8.6}$$

In the special case of the stochastic block model ($\alpha_0 \to 0$), $G^{\alpha_0}_{X,A} = G_{X,A}$ is the submatrix of the adjacency matrix G. Similarly, we define modified third-order statistics,

$$\begin{aligned}\mathrm{T}^{\alpha_0}_{X\to\{A,B,C\}} :=& (\alpha_0+1)(\alpha_0+2)\,\mathrm{T}_{X\to\{A,B,C\}} + 2\,\alpha_0^2\,\boldsymbol{\mu_{X\to A}}\otimes\mu_{X\to B}\otimes\mu_{X\to C}\\ &-\frac{\alpha_0(\alpha_0+1)}{|X|}\sum_{i\in X}\left[G_{i,A}^{\top}\otimes G_{i,B}^{\top}\otimes\mu_{X\to C} + G_{i,A}^{\top}\otimes\mu_{X\to B}\otimes G_{i,C}^{\top} + \boldsymbol{\mu_{X\to A}}\otimes G_{i,B}^{\top}\otimes G_{i,C}^{\top}\right],\end{aligned} \tag{8.7}$$

and it reduces to (a scaled version of) the 3-star count $\mathrm{T}_{X\to\{A,B,C\}}$ defined in (8.3) for the stochastic block model ($\alpha_0 \to 0$). The modified adjacency matrix and the 3-star count tensor can be viewed as a form of "centering" of the raw moments which simplifies the expressions for the moments. The following relationships hold between the modified graph moments $G^{\alpha_0}_{X,A}$, T^{α_0} and the model parameters P and $\widehat{\alpha}$ of the mixed membership model.

Proposition 8.2 (Moments in Mixed Membership Model)**.** *Given partitions A, B, C, X and $G^{\alpha_0}_{X,A}$ and T^{α_0}, as in* (8.6) *and* (8.7)*, normalized Dirichlet concentration vector $\widehat{\alpha}$, and*

[5] To compute the modified moments G^{α_0}, and T^{α_0}, we need to know the value of the scalar $\alpha_0 := \sum_i \alpha_i$, which is the concentration parameter of the Dirichlet distribution and is a measure of the extent of overlap between the communities. We assume its knowledge here.

$F := \Pi^\top P^\top$, where P is the community connectivity matrix and Π is the matrix of community memberships, we have

$$\mathbb{E}[(G^{\alpha_0}_{X,A})^\top|\Pi_A, \Pi_X] = F_A \operatorname{Diag}(\widehat{\alpha}^{1/2})\Psi_X, \tag{8.8}$$

$$\mathbb{E}[\mathrm{T}^{\alpha_0}_{X\to\{A,B,C\}}\,|\Pi_A, \Pi_B, \Pi_C] = \sum_{i=1}^{k} \widehat{\alpha}_i (F_A)_i \otimes (F_B)_i \otimes (F_C)_i, \tag{8.9}$$

where $(F_A)_i$ corresponds to i^{th} column of F_A and Ψ_X relates to the community membership matrix Π_X as

$$\Psi_X := \operatorname{Diag}(\widehat{\alpha}^{-1/2})\left(\sqrt{\alpha_0+1}\Pi_X - (\sqrt{\alpha_0+1}-1)\left(\frac{1}{|X|}\sum_{i\in X}\boldsymbol{\pi_i}\right)\mathbf{1}^\top\right).$$

Moreover, we have that

$$|X|^{-1}\mathbb{E}_{\Pi_X}[\Psi_X\Psi_X^\top] = I. \tag{8.10}$$

Proof: Recall $\mathbb{E}[G^\top_{i,A}|\boldsymbol{\pi_i}, \Pi_A] = F_A\boldsymbol{\pi_i}$ for a mixed membership model, and $\boldsymbol{\mu_{X\to A}} := \frac{1}{|X|}\sum_{i\in X} G^\top_{i,A}$, therefore $\mathbb{E}[\boldsymbol{\mu_{X\to A}}|\Pi_A, \Pi_X] = F_A\left(\frac{1}{|X|}\sum_{i\in X}\boldsymbol{\pi_i}\right)\mathbf{1}^\top$. Equation (8.8) follows directly. For Equation (8.10), we note the Dirichlet moment, $\mathbb{E}[\pi\pi^\top] = \frac{1}{\alpha_0+1}\operatorname{Diag}(\widehat{\alpha}) + \frac{\alpha_0}{\alpha_0+1}\widehat{\alpha}\widehat{\alpha}^\top$, when $\pi \sim \operatorname{Dir}(\alpha)$ and

$$\begin{aligned}
|X|^{-1}\mathbb{E}[\Psi_X\Psi_X^\top] &= \operatorname{Diag}(\widehat{\alpha}^{-1/2})\left[(\alpha_0+1)\mathbb{E}[\pi\pi^\top] + (-2\sqrt{\alpha_0+1}(\sqrt{\alpha_0+1}-1)\right.\\
&\quad \left. +(\sqrt{\alpha_0+1}-1)^2)\mathbb{E}[\pi]\mathbb{E}[\pi]^\top\right]\operatorname{Diag}(\widehat{\alpha}^{-1/2})\\
&= \operatorname{Diag}(\widehat{\alpha}^{-1/2})\left(\operatorname{Diag}(\widehat{\alpha}) + \alpha_0\widehat{\alpha}\widehat{\alpha}^\top + (-\alpha_0)\widehat{\alpha}\widehat{\alpha}^\top\right)\operatorname{Diag}(\widehat{\alpha}^{-1/2})\\
&= I.
\end{aligned}$$

The expectation in (8.9) involves multi-linear map of the expectation of the tensor products $\boldsymbol{\pi} \otimes \boldsymbol{\pi} \otimes \boldsymbol{\pi}$ among other terms. Collecting these terms, we have that

$$(\alpha_0 + 1)(\alpha_0 + 2)\mathbb{E}[\boldsymbol{\pi} \otimes \boldsymbol{\pi} \otimes \boldsymbol{\pi}] - (\alpha_0)(\alpha_0 + 1)(\mathbb{E}[\boldsymbol{\pi} \otimes \boldsymbol{\pi} \otimes \mathbb{E}[\boldsymbol{\pi}]]$$
$$+\mathbb{E}[\boldsymbol{\pi} \otimes \mathbb{E}[\boldsymbol{\pi}] \otimes \boldsymbol{\pi}] + \mathbb{E}[\mathbb{E}[\boldsymbol{\pi}] \otimes \boldsymbol{\pi} \otimes \boldsymbol{\pi}]) + 2\alpha_0^2\mathbb{E}[\boldsymbol{\pi}] \otimes \mathbb{E}[\boldsymbol{\pi}] \otimes \mathbb{E}[\boldsymbol{\pi}]$$

is a diagonal tensor, in the sense that its (p,p,p)-th entry is $\widehat{\alpha}_p$, and its (p,q,r)-th entry is 0 when p,q,r are not all equal. With this, we have (8.9). □

Note that the form of the tensor in this Lemma is very similar to Theorem 7.2. However, because F_A, F_B, F_C are three different matrices, we need a more complicated reduction than Section 7.2.1.

8.3 Algorithm for Learning Mixed Membership Models

The simple form of the graph moments derived in the previous section is now utilized to recover the community vectors Π and model parameters $P, \widehat{\alpha}$ of the mixed membership model. We first analyze the simpler case when exact moments are available in Section 8.3.1 and then extend the method to handle empirical moments computed from the network observations in Section 8.3.2.

8.3.1 Learning Mixed Membership Models Under Exact Moments

We first describe the learning approach when exact moments are available. In Section 8.3.2, we suitably modify the approach to handle perturbations, which are introduced when only empirical moments are available.

We first describe a "simultaneous whitening" procedure similar to Section 7.2.1 to convert the graph moment tensor T^{α_0} to a orthogonally decomposable tensor through a multi-linear transformation of T^{α_0}. We then employ the power method to obtain the eigenvalues and eigenvectors of the tensor. Finally, by reversing the multi-linear transform in the whitening step we get the parameters of interest. This yields a guaranteed method for obtaining the decomposition of graph moment tensor T^{α_0} under exact moments.

Reduction of the graph-moment tensor to symmetric orthogonal form (Whitening): Recall from Proposition 8.2 that the modified 3-star count tensor T^{α_0} has the form

$$\mathbb{E}[\mathrm{T}^{\alpha_0}|\Pi_A, \Pi_B, \Pi_C] = \sum_{i=1}^{k} \widehat{\alpha}_i (F_A)_i \otimes (F_B)_i \otimes (F_C)_i.$$

In order to whiten the tensor, we make use of the modified adjacency matrix G^{α_0}, defined in (8.6). Consider the singular value decomposition (SVD) of the modified adjacency matrix G^{α_0} under exact moments:

$$|X|^{-1/2}\mathbb{E}[(G^{\alpha_0}_{X,A})^\top|\Pi] = U_A D_A V_A^\top.$$

Define $W_A := U_A D_A^{-1}$, and similarly define W_B and W_C using the corresponding matrices $G^{\alpha_0}_{X,B}$ and $G^{\alpha_0}_{X,C}$ respectively. Now define

$$R_{A,B} := \frac{1}{|X|} W_B^\top \mathbb{E}[(G^{\alpha_0}_{X,B})^\top|\Pi] \cdot \mathbb{E}[(G^{\alpha_0}_{X,A})|\Pi] W_A, \quad \tilde{W}_B := W_B R_{A,B}, \tag{8.11}$$

and similarly define $\tilde{W}_C$. We establish that a multilinear transformation (as defined in (A.1)) of the graph-moment tensor T^{α_0} using matrices $W_A, \tilde{W}_B$, and $\tilde{W}_C$ results in a orthogonally decomposable tensor.

Lemma 8.3 (Reducing to Orthogonally Decomposable Tensor)**.** *Assume that the matrices F_A, F_B, F_C and Π_X have rank k, where k is the number of communities. We have an orthogonal symmetric tensor form for the modified 3-star count tensor T^{α_0} in (8.7) under a multilinear transformation using matrices $W_A, \tilde{W}_B$, and $\tilde{W}_C$:*

$$\mathbb{E}[\mathrm{T}^{\alpha_0}(W_A, \tilde{W}_B, \tilde{W}_C)|\Pi_A, \Pi_B, \Pi_C] = \sum_{i\in[k]} \lambda_i(\Phi)_i^{\otimes 3} \in \mathbb{R}^{k\times k\times k}, \tag{8.12}$$

where $\lambda_i := \widehat{\alpha}_i^{-0.5}$ and $\Phi \in \mathbb{R}^{k\times k}$ is an orthogonal matrix, given by

$$\Phi := W_A^\top F_A \operatorname{Diag}(\widehat{\alpha}^{0.5}). \tag{8.13}$$

Remark: Note that the matrix W_A orthogonalizes F_A under exact moments, and is referred to as a *whitening matrix.* Similarly, the matrices $\tilde{W}_B = R_{A,B}W_B$ and $\tilde{W}_C = R_{A,C}W_C$ consist of whitening matrices W_B and W_C, and in addition, the matrices $R_{A,B}$ and $R_{A,C}$ serve to symmetrize the tensor.

Proof: Recall that the modified adjacency matrix G^{α_0} satisfies

$$\mathbb{E}[(G^{\alpha_0}_{X,A})^\top|\Pi_A, \Pi_X] = F_A \operatorname{Diag}(\widehat{\alpha}^{1/2})\Psi_X.$$

$$\Psi_X := \operatorname{Diag}(\widehat{\alpha}^{-1/2}) \left(\sqrt{\alpha_0+1}\Pi_X - (\sqrt{\alpha_0+1}-1)\left(\frac{1}{|X|}\sum_{i\in X}\boldsymbol{\pi_i}\right)\mathbf{1}^\top\right).$$

From the definition of Ψ_X above, we see that it has rank k when Π_X has rank k. Using the Sylvester's rank inequality, we have that the rank of $F_A \operatorname{Diag}(\widehat{\alpha}^{1/2})\Psi_X$ is at least $2k - k = k$. This implies that the whitening matrix W_A also has rank k. Notice that

$$|X|^{-1}W_A^\top \mathbb{E}[(G^{\alpha_0}_{X,A})^\top|\Pi] \cdot \mathbb{E}[(G^{\alpha_0}_{X,A})|\Pi]W_A = D_A^{-1}U_A^\top U_A D_A^2 U_A^\top U_A D_A^{-1} = I \in \mathbb{R}^{k\times k},$$

or in other words, $|X|^{-1}MM^\top = I$, where $M := W_A^\top F_A \operatorname{Diag}(\widehat{\alpha}^{1/2})\Psi_X$. We now have that

$$I = |X|^{-1}\mathbb{E}_{\Pi_X}\left[MM^\top\right] = |X|^{-1}W_A^\top F_A \operatorname{Diag}(\widehat{\alpha}^{1/2})\mathbb{E}[\Psi_X\Psi_X^\top] \operatorname{Diag}(\widehat{\alpha}^{1/2})F_A^\top W_A$$
$$= W_A^\top F_A \operatorname{Diag}(\widehat{\alpha})F_A^\top W_A,$$

since $|X|^{-1}\mathbb{E}_{\Pi_X}[\Psi_X\Psi_X^\top] = I$ from (8.10), and we use the fact that the sets A and X do not overlap. Thus, W_A whitens $F_A \operatorname{Diag}(\widehat{\alpha}^{1/2})$ under exact moments (up on taking expectation over Π_X) and the columns of $W_A^\top F_A \operatorname{Diag}(\widehat{\alpha}^{1/2})$ are orthonormal. Now note from the definition of $\tilde{W}_B$ that

$$\tilde{W}_B^\top \mathbb{E}[(G_{X,B}^{\alpha_0})^\top|\Pi] = W_A^\top \mathbb{E}[(G_{X,A}^{\alpha_0})^\top|\Pi],$$

since W_B satisfies

$$|X|^{-1}W_B^\top \mathbb{E}[(G_{X,B}^{\alpha_0})^\top|\Pi] \cdot \mathbb{E}[(G_{X,B}^{\alpha_0})|\Pi]W_B = I,$$

and similar result holds for $\tilde{W}_C$. The final result in (8.12) follows by taking expectation of tensor T^{α_0} over Π_X. □

Overview of the learning approach under exact moments: With the above result in place, we are now ready to describe the high-level approach for learning the mixed membership model under exact moments. First, whiten the graph-moment tensor T^{α_0} as described above and then apply the tensor power method described in the previous Chapter. This enables us to obtain the vector of eigenvalues $\lambda := \widehat{\alpha}^{-1/2}$ and the matrix of eigenvectors $\Phi = W_A^\top F_A \operatorname{Diag}(\widehat{\alpha}^{0.5})$ using tensor power iterations. We can then recover the community membership vectors of set A^c (i.e., nodes not in set A) under exact moments as

$$\Pi_{A^c} \leftarrow \operatorname{Diag}(\lambda)^{-1}\Phi^\top W_A^\top \mathbb{E}[G_{A^c,A}^\top|\Pi],$$

since $\mathbb{E}[G^\top_{A^c,A}|\Pi] = F_A\Pi_{A^c}$ (since A and A^c do not overlap) and $\mathrm{Diag}(\lambda)^{-1}\Phi^\top W_A^\top = \mathrm{Diag}(\widehat{\alpha})F_A^\top W_A W_A^\top$ under exact moments. In order to recover the community membership vectors of set A, viz., Π_A, we can reverse the direction of the 3-star counts, i.e., consider the 3-stars from set A to X, B, C and obtain Π_A in a similar manner. Once all the community membership vectors Π are obtained, we can obtain the community connectivity matrix P, using the relationship: $\Pi^\top P \Pi = \mathbb{E}[G|\Pi]$ and noting that we assume Π to be of rank k. Thus, we are able to learn the community membership vectors Π and the model parameters $\widehat{\alpha}$ and P of the mixed membership model using edge counts and the 3-star count tensor. We now describe modifications to this approach to handle empirical moments.

8.3.2 Learning Algorithm Under Empirical Moments

In the previous section, we explored a tensor-based approach for learning mixed membership model under exact moments. However, in practice, we only have samples (i.e. the observed network), and the method needs to be robust to perturbations when empirical moments are employed.

Algorithm 8.1. $\{\hat{\Pi}, \hat{P}, \widehat{\alpha}\} \leftarrow \mathrm{LearnMixedMembership}(G, k, \alpha_0)$

```
Input: Adjacency matrix $G \in \mathbb{R}^{n\times n}$, $k$ is the number of communities, $\alpha_0 := \sum_i \alpha_i$, where
  $\alpha$ is the Dirichlet parameter vector. Let $A^c := [n] \setminus A$ denote the set of nodes not in $A$.
Output: Estimates of the community membership vectors $\Pi \in \mathbb{R}^{n\times k}$, community connec-
  tivity matrix $P \in [0,1]^{k\times k}$, and the normalized Dirichlet parameter vector $\widehat{\alpha}$.
  Partition the vertex set $[n]$ into 5 parts $X$, $Y$, $A$, $B$, $C$.
  Compute moments $G^{\alpha_0}_{X,A}$, $G^{\alpha_0}_{X,B}$, $G^{\alpha_0}_{X,C}$, $\mathrm{T}^{\alpha_0}_{Y\to\{A,B,C\}}$ using (8.6) and (8.7).
  $\{\hat{\Pi}_{A^c}, \widehat{\alpha}\} \leftarrow \mathrm{LearnPartitionCommunity}(G^{\alpha_0}_{X,A}, G^{\alpha_0}_{X,B}, G^{\alpha_0}_{X,C}, \mathrm{T}^{\alpha_0}_{Y\to\{A,B,C\}}, G)$.
  Interchange roles of $Y$ and $A$ to obtain $\hat{\Pi}_{Y^c}$.
  Define $\hat{Q}$ such that its $i$-th row is $\hat{Q}^i := (\alpha_0+1)\frac{\hat{\Pi}^i}{|\hat{\Pi}^i|_1} - \frac{\alpha_0}{n}\mathbf{1}^\top$. {We will establish that
  $\hat{Q} \approx (\Pi^\dagger)^\top$.}
  Estimate $\hat{P} \leftarrow \hat{Q} G \hat{Q}^\top$. {Recall that $\mathbb{E}[G] = \Pi^\top P \Pi$ in our model.}
  Return $\hat{\Pi}, \hat{P}, \widehat{\alpha}$
```

Pre-processing steps

```
Algorithm 8.2. $\{\hat{\Pi}_{A^c}, \widehat{\alpha}\} \leftarrow$ LearnPartitionCommunity($G^{\alpha_0}_{X,A}$, $G^{\alpha_0}_{X,B}$, $G^{\alpha_0}_{X,C}$, $\mathrm{T}^{\alpha_0}_{Y\to\{A,B,C\}}$, $G$)
Input: Require modified adjacency submatrices $G^{\alpha_0}_{X,A}$, $G^{\alpha_0}_{X,B}$, $G^{\alpha_0}_{X,C}$, 3-star count tensor $\mathrm{T}^{\alpha_0}_{Y\to\{A,B,C\}}$, adjacency matrix $G$
Output: Estimates of $\Pi_{A^c}$ and $\widehat{\alpha}$.
    Compute rank-$k$ SVD: $(|X|^{-1/2}G^{\alpha_0}_{X,A})^\top_{k-svd} = U_A D_A V_A^\top$ and compute whitening matrices $\hat{W}_A := U_A D_A^{-1}$. Similarly, compute $\hat{W}_B, \hat{W}_C$ and $\hat{R}_{AB}, \hat{R}_{AC}$ using (8.14).
    Compute whitened and symmetrized tensor $T \leftarrow \mathrm{T}^{\alpha_0}_{Y\to\{A,B,C\}}(\hat{W}_A, \hat{W}_B\hat{R}_{AB}, \hat{W}_C\hat{R}_{AC})$.
    $\{\hat{\lambda}, \hat{\Phi}\} \leftarrow$ Algorithm 7.3 TensorEigen($T, \{\hat{W}_A^\top G_{i,A}^\top\}_{i\in B}, 10\log n$). {$\hat{\Phi}$ is a $k\times k$ matrix with each columns being an estimated eigenvector and $\hat{\lambda}$ is the vector of estimated eigenvalues. $\hat{W}_A^\top G_{i,A}^\top$'s are initial vectors}
    $\hat{\Pi}_{A^c} \leftarrow$ Thres(Diag$(\hat{\lambda})^{-1}\hat{\Phi}^\top\hat{W}_A^\top G_{A^c,A}^\top$ , $\tau$) and $\hat{\alpha}_i \leftarrow \hat{\lambda}_i^{-2}$, for $i \in [k]$.
    Return $\hat{\Pi}_{A^c}$ and $\hat{\alpha}$.
```

Partitioning: In the previous section, we partitioned the nodes into four sets A, B, C, X for learning under exact moments. However, we require more partitions under empirical moments to avoid statistical dependency issues and obtain stronger reconstruction guarantees. We now divide the network into five non-overlapping sets A, B, C, X, Y. The set X is employed to compute whitening matrices $\hat{W}_A$, $\hat{W}_B$ and $\hat{W}_C$, described in detail subsequently, the set Y is employed to compute the 3-star count tensor T^{α_0} and sets A, B, C contain the leaves of the 3-stars under consideration. The roles of the sets can be interchanged to obtain the community membership vectors of all the sets.

Whitening: The whitening procedure is along the same lines as described in the previous section, except that now empirical moments are used. Specifically, consider the k-rank singular value decomposition (SVD) of the modified adjacency matrix G^{α_0} defined in (8.6),

$$(|X|^{-1/2}G^{\alpha_0}_{X,A})^\top_{k-svd} = U_A D_A V_A^\top.$$

Define $\hat{W}_A := U_A D_A^{-1}$, and similarly define $\hat{W}_B$ and $\hat{W}_C$ using the corresponding matrices $G^{\alpha_0}_{X,B}$ and $G^{\alpha_0}_{X,C}$ respectively. Now define

$$\hat{R}_{A,B} := \frac{1}{|X|}\hat{W}_B^\top (G^{\alpha_0}_{X,B})^\top_{k-svd} \cdot (G^{\alpha_0}_{X,A})_{k-svd}\hat{W}_A, \tag{8.14}$$

and similarly define $\hat{R}_{AC}$. The whitened and symmetrized graph-moment tensor is now computed as

$$\mathrm{T}^{\alpha_0}_{Y\to\{A,B,C\}}(\hat{W}_A, \hat{W}_B\hat{R}_{AB}, \hat{W}_C\hat{R}_{AC}),$$

where T^{α_0} is given by (8.7) and the multi-linear transformation of a tensor is defined in (A.1).

Efficient Initialization: For a mixed membership model in the sparse regime, recall that the community membership vectors Π are sparse (with high probability). Under this regime of the model, we note that the whitened neighborhood vectors contain good initializers for the power iterations. Specifically, in Algorithm 7.1, we initialize with the whitened neighborhood vectors $\hat{W}_A^\top G_{i,A}^\top$, for $i \notin A$. The intuition behind this is as follows: for a suitable choice of parameters (such as the scaling of network size n with respect to the number of communities k), we expect neighborhood vectors $G_{i,A}^\top$ to concentrate around their mean values, viz., , $F_A\boldsymbol{\pi_i}$. Since $\boldsymbol{\pi_i}$ is sparse (w.h.p) for the model regime under consideration, this implies that there exist vectors $\hat{W}_A^\top F_A\boldsymbol{\pi_i}$, for $i \in A^c$, which concentrate (w.h.p) on only along a few eigen-directions of the whitened tensor, and hence, serve as an effective initializer.

Reconstruction after tensor power method

Recall that previously in Section 8.3.1, when exact moments are available, estimating the community membership vectors Π is straightforward, once we recover all the stable tensor eigen-pairs. However, in case of empirical moments, we can obtain better guarantees with the following modification: the estimated community membership vectors Π are further subject to thresholding so that the weak values are set to zero. Since we are limiting ourselves to the regime of the mixed membership model, where the community vectors Π are sparse (w.h.p), this modification strengthens our reconstruction guarantees. This thresholding step is incorporated in Algorithm 8.1.

Moreover, recall that under exact moments, estimating the community connectivity matrix P is straightforward, once we recover the community membership vectors since $P \leftarrow (\Pi^\top)^\dagger \mathbb{E}[G|\Pi]\Pi^\dagger$. However, when empirical moments are available, we are able to establish better reconstruction guarantees through a different method, outlined in Algorithm 8.1. We define $\hat{Q}$ such that its i-th row is

$$\hat{Q}^i := (\alpha_0 + 1)\frac{\hat{\Pi}^i}{|\hat{\Pi}^i|_1} - \frac{\alpha_0}{n}\mathbf{1}^\top,$$

based on estimates $\hat{\Pi}$, and the matrix $\hat{P}$ is obtained as $\hat{P} \leftarrow \hat{Q}G\hat{Q}^\top$. We subsequently establish that $\hat{Q}\hat{\Pi}^\top \approx I$.

Improved support recovery estimates in homophilic models: We describe the post-processing method in Algorithm 8.3, which gives improved estimates by averaging. Specifically, consider nodes in set C and edges going from C to nodes in B. First, consider the special case of the stochastic block model: for each node $c \in C$, compute the number of neighbors in B belonging to each community (as given by the estimate $\hat{\Pi}$ from Algorithm 8.1), and declare the community with the maximum number of such neighbors as the community of node c. Intuitively, this provides a better estimate for Π_C since we average over the edges in B. This method has been used before in the context of spectral clustering [125]. The same idea can be extended to the general mixed membership models: declare communities to be significant if they exceed a certain threshold, as evaluated by the average number of edges to each community. The details are provided in Algorithm 8.3. In the next section, we establish that in certain regime of parameters, this procedure can lead to zero-error support recovery of significant community memberships of the nodes and also rule out communities where a node does not have a strong presence.

Algorithm 8.3. $\{\hat{S}\} \leftarrow$ SupportRecoveryHomophilicModels$(G, k, \alpha_0, \xi, \hat{\Pi})$

```
Input: Adjacency matrix $G \in \mathbb{R}^{n \times n}$, $k$ is the number of communities, $\alpha_0 := \sum_i \alpha_i$, where $\alpha$
    is the Dirichlet parameter vector, $\xi$ is the threshold for support recovery, corresponding to
    significant community memberships of an individual. Get estimate $\hat{\Pi}$ from Algorithm 8.1.
Output: $\hat{S} \in \{0,1\}^{n \times k}$ is the estimated support for significant community memberships
    (see Theorem 8.6 for guarantees).
    Consider partitions $A, B, C, X, Y$ as in Algorithm 8.1.
    Define $\hat{Q}$ on lines of definition in Algorithm 8.1, using estimates $\hat{\Pi}$. Let the $i$-th row for
    set $B$ be $\hat{Q}_B^i := (\alpha_0 + 1)\frac{\hat{\Pi}_B^i}{|\hat{\Pi}_B^i|_1} - \frac{\alpha_0}{n}\mathbf{1}^\top$. Similarly define $\hat{Q}_C^i$.
    Estimate $\hat{F}_C \leftarrow G_{C,B}\hat{Q}_B^\top$, $\hat{P} \leftarrow \hat{Q}_C\hat{F}_C$.
    Let $H$ be the average of diagonals of $\hat{P}$, $L$ be the average of off-diagonals of $\hat{P}$
    for $x \in C$, $i \in [k]$ do
        $\hat{S}(i,x) \leftarrow 1$ if $\hat{F}_C(x,i) \geq L + (H - L) \cdot \frac{3\xi}{4}$ and zero otherwise.{Identify large entries}
    end for
    Permute the roles of the sets $A, B, C, X, Y$ to get results for remaining nodes.
```

Computational complexity: We note that the computational complexity of the method is $O(n^2k)$. This is because the time for computing whitening matrices is dominated by SVD of the top k singular vectors of $n \times n$ matrix, which takes $O(n^2k)$ time. We then compute the whitened tensor T which requires time $O(n^2k + k^3n) = O(n^2k)$, since for each $i \in Y$, we multiply $G_{i,A}, G_{i,B}, G_{i,C}$ with the corresponding whitening matrices, and this step takes $O(nk)$ time. We then average this $k \times k \times k$ tensor over different nodes $i \in Y$ to the result, which takes $O(k^3)$ time in each step.

For the tensor power method, the time required for a single iteration is $O(k^3)$. We need at most $\log n$ iterations per initial vector, and we need to consider k^{γ_0} initial vectors (recall that γ_0 is a constant close to 1). Hence the total running time of tensor power method is $O(k^{4+\gamma})$ which is dominated by $O(n^2k)$.

In the process of estimating Π and P, the dominant operation is multiplying $k \times n$ matrix by $n \times n$ matrix, which takes $O(n^2k)$ time. For support recovery, the dominant operation is computing the "average degree", which again takes $O(n^2k)$ time. Thus, we have that the overall computational time is $O(n^2k)$.

Note that the above bound on complexity of our method nearly matches the bound for spectral clustering method [125], since computing the k-rank SVD requires $O(n^2k)$ time. Another method for learning stochastic block models is based on convex optimization involving semi-definite programming (SDP) [166], and it provides the best scaling bounds (for both the network size n and the separation $p-q$ for edge connectivity) known so far. The specific convex problem can be solved via the method of *augmented Lagrange multipliers* [114], where each step consists of an SVD operation and q-linear convergence is established by [114]. This implies that the method has complexity $O(n^3)$, since it involves taking SVD of a general $n \times n$ matrix, rather than a k-rank SVD. Thus, our method has significant advantage in terms of computational complexity, when the number of communities is much smaller than the network size ($k \ll n$).

8.4 Sample Analysis for Proposed Learning Algorithm

8.4.1 Sufficient Conditions and Recovery Guarantees

Recall that for a matrix M, $(M)^i$ and $(M)_i$ denote the i^{th} row and column respectively. We say that an event holds with high probability, if it occurs with probability $1 - n^{-c}$ for some constant $c > 0$. Recall that $\tilde{O}$ and $\tilde{\Omega}$ hides polylog factors and polynomials of $(1 + \alpha_0)$.

Theorem 8.4 (Guarantees on Estimating P, Π)**.** *When* $p > q > \operatorname{poly}\log n$, $\frac{p-q}{\sqrt{p}} = \tilde{\Omega}(\frac{k}{\sqrt{n}})$, *with high probability*

$$\varepsilon_{\pi,\ell_1} := \max_i \|\hat{\Pi}^i - \Pi^i\|_1 = \tilde{O}\left(\frac{\sqrt{np}}{(p-q)}\right)$$
$$\varepsilon_P := \max_{i,j\in[n]} |\hat{P}_{i,j} - P_{i,j}| = \tilde{O}\left(\frac{k\sqrt{p}}{\sqrt{n}}\right).$$

The proofs are given in next section and a proof outline is provided in Section 8.4.2.

The main ingredient in establishing the above result is the tensor concentration bound and additionally, recovery guarantees under the tensor power method in Procedure 7.1. We now provide these results below.

Recall that $F_A := \Pi_A^\top P^\top$ and $\Phi = W_A^\top F_A \operatorname{Diag}(\widehat{\alpha}^{1/2})$ denotes the set of tensor eigenvectors under exact moments in (8.13), and $\hat{\Phi}$ is the set of estimated eigenvectors under empirical moments, obtained using Algorithm 8.2. We establish the following guarantees.

Lemma 8.5 (Perturbation bound for estimated eigen-pairs)**.** *The recovered eigenvector-eigenvalue pairs $(\hat{\Phi}_i, \hat{\lambda}_i)$ from the tensor power method satisfies with high probability, for a permutation θ, such that*

$$\max_{i\in[k]} \|\hat{\Phi}_i - \Phi_{\theta(i)}\| \leq 8k^{-1/2}\varepsilon_T, \qquad \max_i |\lambda_i - \widehat{\alpha}_{\theta(i)}^{-1/2}| \leq 5\varepsilon_T, \tag{8.15}$$

The tensor perturbation bound ε_T is given by

$$\begin{aligned}\varepsilon_T &:= \left\| \mathrm{T}^{\alpha_0}_{Y\to\{A,B,C\}}(\hat{W}_A, \hat{W}_B\hat{R}_{AB}, \hat{W}_C\hat{R}_{AC}) - \mathbb{E}[\mathrm{T}^{\alpha_0}_{Y\to\{A,B,C\}}(W_A, \tilde{W}_B, \tilde{W}_C)|\Pi_{A\cup B\cup C}] \right\| \\ &= \tilde{O}\left(\frac{k^{3/2}\sqrt{p}}{(p-q)\sqrt{n}}\right). \end{aligned}\tag{8.16}$$

Notice that the power on k is 3/2 here, the extra $\sqrt{k}$ factor is due to all eigenvalues of this tensor are of order $\sqrt{k}$.

Zero-error guarantees for support recovery

Recall that we proposed Procedure 8.3 as a post-processing step to provide improved support recovery estimates. We now provide guarantees for this method.

We now specify the threshold ξ for support recovery in Algorithm 8.3.

Theorem 8.6 (Support Recovery)**.** *The support recovery method in Algorithm 8.3 has the following guarantees on the estimated support set $\hat{S}$ when $\xi = \tilde{\Omega}(\frac{\sqrt{p}k}{(p-q)\sqrt{n}})$: with high probability,*

$$\Pi(i,j) \geq \xi \Rightarrow \hat{S}(i,j) = 1 \quad \textit{and} \quad \Pi(i,j) \leq \frac{\xi}{2} \Rightarrow \hat{S}(i,j) = 0, \quad \forall i \in [k], j \in [n], \tag{8.17}$$

where Π is the true community membership matrix.

Thus, the above result guarantees that the Algorithm 8.3 correctly recovers all the "large" entries of Π and also correctly rules out all the "small" entries in Π. In other words, we can correctly infer all the significant memberships of each node and also rule out the set of communities where a node does not have a strong presence.

The only shortcoming of the above result is that there is a gap between the "large" and "small" values, and for an intermediate set of values (in $[\xi/2, \xi]$), we cannot guarantee correct inferences about the community memberships. Note this gap depends on ε_P, the error in estimating the P matrix. This is intuitive, since as the error ε_P decreases, we can infer the community memberships over a large range of values.

For the special case of stochastic block models (i.e. $\lim \alpha_0 \to 0$), we can improve the above result and give a zero error guarantee at all nodes (w.h.p). Note that we no longer require a threshold ξ in this case, and only infer one community for each node.

8.4.2 Proof Outline

We now summarize the main techniques involved in proving Theorems 8.4 and 8.6. The details are in Section 8.5. The main ingredient is the concentration of the adjacency matrix: since the edges are drawn independently conditioned on the community memberships, we establish that the adjacency matrix concentrates around its mean under the stated assumptions. See Section 8.5.1 for details. With this in hand, we can then establish concentration of various quantities used by our learning algorithm.

Step 1: Whitening matrices. We first establish concentration bounds on the whitening matrices $\hat{W}_A$, $\hat{W}_B$, $\hat{W}_C$ computed using empirical moments, described in Section 8.3.2. With this in hand, we can approximately recover the span of matrix F_A since $\hat{W}_A^\top F \operatorname{Diag}(\widehat{\alpha}_i)^{1/2}$ is a rotation matrix. The main technique employed is the Matrix Bernstein's inequality [149, thm. 1.4]. See Appendix B for details.

Step 2: Tensor concentration bounds Recall that we use the whitening matrices to obtain a symmetric orthogonal tensor. We establish that the whitened and symmetrized tensor concentrates around its mean. This is done in several stages and we carefully control the tensor perturbation bounds. See Section 8.5.1 for details.

Step 3: Tensor power method analysis. We utilize the algorithm in Section 7.7. Here we need to establish that there exist good initializers for the power method among (whitened) neighborhood vectors, see Section 8.5.2 for details.

Step 4: Thresholding of estimated community vectors In Step 3, we provide guarantees for recovery of each eigenvector in ℓ_2 norm. Direct application of this result only allows us to obtain ℓ_2 norm bounds for row-wise recovery of the community matrix Π. In order to strengthen the result to an ℓ_1 norm bound, we threshold the estimated Π vectors. Here, we exploit the sparsity in Dirichlet draws and carefully control the contribution of weak entries in the vector. Finally, we establish perturbation bounds on P through rather straightforward concentration bound arguments. See Section 8.5.3 for details.

Step 5: Support recovery guarantees. It is convenient to consider the case of in stochastic block model here in the canonical setting of Section 8.4.1. Recall that Procedure 8.3 readjusts the community membership estimates based on degree averaging. For each vertex, if we count the average degree towards these "approximate communities", for the correct community the result is concentrated around value p and for the wrong community the result is around value q. Therefore, we can correctly identify the community memberships of all the nodes, when $p - q$ is sufficiently large. The argument can be easily extended to general mixed membership models. See Section 8.5.3 for details.

8.5 Proof of the Main Theorems

In this section we prove the main theorems. The order of the proof follows the order of the algorithm: we will first show concentration bounds for the whitening matrices and the tensor; then we give the guarantee for tensor power method (which mainly involves showing the initializers are good); finally we prove the reconstruction guarantees.

The proof is simplified due to the assumptions (all communities are of the same size, P has the special structure), but is not oversimplified and can still be easily extended to the general case.

Throughout the section we heavily use the fact that the columns of Π are chosen according to Dirichlet distribution, in particular the moments of the distribution is stated as follows

Proposition 8.7 (Moments under Dirichlet distribution). *Suppose $\boldsymbol{\pi} \sim Dir(\alpha)$, the moments of $\boldsymbol{\pi}$ satisfies the following formulas:*

$$\begin{aligned}
\mathbb{E}[\boldsymbol{\pi}_i] &= \frac{\alpha_i}{\alpha_0} \\
\mathbb{E}[\boldsymbol{\pi}_i^2] &= \frac{\alpha_i(\alpha_i+1)}{\alpha_0(\alpha_0+1)} \\
\mathbb{E}[\boldsymbol{\pi}_i\boldsymbol{\pi}_j] &= \frac{\alpha_i\alpha_j}{\alpha_0(\alpha_0+1)}, \quad i \neq j.
\end{aligned}$$

More generally, if $a^{(t)} = \prod_{i=0}^{t-1}(a+i)$*, then we have*

$$\mathbb{E}[\prod_{i=1}^{k} \boldsymbol{\pi}_i^{(a_i)}] = \frac{\prod_{i=1}^{k} \alpha_i^{(a_i)}}{\alpha_0^{(\sum_{i=1}^{k} a_i)}}.$$

8.5.1 Concentration Bounds

Let's first look at the concentration bounds for the matrices and tensors used in the algorithm.

Bounding Π **and** F

Lemma 8.8 (Bounds for Π and F)**.** *For* $\boldsymbol{\pi_i} \overset{iid}{\sim} \mathrm{Dir}(\alpha)$ *with high probability, we have*

$$\begin{aligned}\sigma_{\min}(\Pi_A) &\geq \sqrt{\frac{|A|}{k(\alpha_0+1)}} - \tilde{O}(n^{1/4}),\\ \|\Pi_A\| &\leq \sqrt{|A|/k} + \tilde{O}(n^{1/4}),\\ \kappa(\Pi_A) &\leq 2\sqrt{(\alpha_0+1)}.\end{aligned}$$

Moreover, with high probability

$$|F_A|_1 \leq 2p|A| \tag{8.18}$$

Proof: Consider $\Pi_A \Pi_A^\top = \sum_{i\in A} \boldsymbol{\pi_i}\boldsymbol{\pi_i}^\top$.

$$\begin{aligned}\frac{1}{|A|}\mathbb{E}[\Pi_A^\top \Pi_A] =& \mathbb{E}_{\pi\sim Dir(\alpha)}[\boldsymbol{\pi}\boldsymbol{\pi}^\top]\\ =& \frac{\alpha_0}{\alpha_0+1}\widehat{\alpha}\widehat{\alpha}^\top + \frac{1}{\alpha_0+1}\mathrm{Diag}(\widehat{\alpha}),\end{aligned}$$

from Proposition 8.7. The first term is positive semi-definite so the eigenvalues of the sum are at least the eigenvalues of the second component. Smallest eigenvalue of second component gives lower bound on $\sigma_{\min}(\mathbb{E}[\Pi_A \Pi_A^\top])$. The spectral norm of the first component is bounded by $\frac{\alpha_0}{\alpha_0+1}\|\hat{\alpha}\| \leq \frac{\alpha_0}{\alpha_0+1}\widehat{\alpha}_{\max}$, the spectral norm of second component is $\frac{1}{\alpha_0+1}\alpha_{\max}$. Thus $\left\|\mathbb{E}[\Pi_A\Pi_A^\top]\right\| \leq |A| \cdot \widehat{\alpha}_{\max}$.

Now applying Matrix Bernstein's inequality to $\frac{1}{|A|}\sum_i \left(\boldsymbol{\pi_i}\boldsymbol{\pi_i}^\top - \mathbb{E}[\boldsymbol{\pi}\boldsymbol{\pi}^\top]\right)$. We have that the variance is $O(1/|A|)$. Thus with probability $1-\delta$,

$$\left\|\frac{1}{|A|}\left(\Pi_A\Pi_A^\top - \mathbb{E}[\Pi_A\Pi_A^\top]\right)\right\| = O\left(\sqrt{\frac{\log(k/\delta)}{|A|}}\right).$$

To show bound on $|F_A|_1$, note that each column of F_A satisfies $\mathbb{E}[(F_A)_i] = (\widehat{\alpha}^T(P)_i)\mathbf{1}^\top$, and thus $|\mathbb{E}[F_A]|_1 \leq |A|\max_i(P\widehat{\alpha})_i \leq p|A|$. Using Bernstein's inequality, for each column of F_A, we have, with probability $1-\delta$,

$$\left|\,|(F_A)_i|_1 - |A|\widehat{\alpha}^\top(P)_i\right| = O\left(\sqrt{p|A|\log\frac{|A|}{\delta}}\right),$$

by applying Bernstein's inequality, since $|\widehat{\alpha}^\top(P)_i| \leq p$. □

Bounding the Modified Adjacency Matrix The amount of error in the whitening step largely depends on the spectral norm of the difference between the true modified adjacency matrix $G^{\alpha_0}_{X,A}$ and its expectation.

Lemma 8.9 (Concentration of adjacency submatrices)**.** *With high probability*

$$\|G_{X,A} - \mathbb{E}[G_{X,A}|\Pi]\| = \tilde{O}\left(\sqrt{pn}\right). \tag{8.19}$$

$$\|\boldsymbol{\mu_{X\to A}} - \mathbb{E}[\boldsymbol{\mu_{X\to A}}|\Pi]\| = \tilde{O}(\sqrt{p/n}). \tag{8.20}$$

As a corollary, the modified adjacency matrix has bound

$$\epsilon_G := \|G^{\alpha_0}_{X,A} - \mathbb{E}[(G^{\alpha_0}_{X,A})^\top|\Pi]\| = \tilde{O}(\sqrt{pn}) \tag{8.21}$$

Proof: Recall $\mathbb{E}[G_{X,A}|\Pi] = F_A\Pi_X$ and $G_{A,X} = \text{Ber}(F_A\Pi_X)$ where $\text{Ber}(\cdot)$ denotes the Bernoulli random matrix with independent entries. Let

$$Z_i := (G_{i,A}^\top - F_A\boldsymbol{\pi_i})e_i^\top.$$

We have $G_{X,A}^\top - F_A\Pi_X = \sum_{i\in X} Z_i$. We apply matrix Bernstein's inequality.

We compute the variances $\sum_i \mathbb{E}[Z_iZ_i^\top|\Pi]$ and $\sum_i \mathbb{E}[Z_i^\top Z_i|\Pi]$. We have that $\sum_i \mathbb{E}[Z_iZ_i^\top|\Pi]$ only the diagonal terms are non-zero due to independence of Bernoulli variables, and

$$\mathbb{E}[Z_iZ_i^\top|\Pi] \leq \text{Diag}(F_A\boldsymbol{\pi_i}) \tag{8.22}$$

entry-wise. Thus,

$$\begin{aligned}
\|\sum_{i\in X}\mathbb{E}[Z_iZ_i^\top|\Pi]\| &\leq \max_{a\in[k]} \sum_{i\in X, b\in[k]} F_A(a,b)\boldsymbol{\pi_i}(b) \\
&= \max_{a\in[k]} \sum_{i\in X, b\in[k]} F_A(a,b)\Pi_X(b,i) \\
&\leq \max_{c\in[k]} \sum_{i\in X, b\in[k]} P(c,b)\Pi_X(b,d) \\
&= \|P\Pi_X\|_\infty = |F_X|_1.
\end{aligned} \tag{8.23}$$

Similarly $\sum_{i\in X}\mathbb{E}[Z_i^\top Z_i] = \sum_{i\in X}\text{Diag}(\mathbb{E}[\|G_{i,A}^\top - F_A\boldsymbol{\pi_i}\|^2]) \leq |F_X|_1$. From Lemma 8.8, we have $|F_X|_1 = \tilde{O}(p|X|)$.

We now bound $\|Z_i\|$. First note that the entries in $G_{i,A}$ are independent and we can use the vector Bernstein's inequality to bound $\|G_{i,A} - F_A\boldsymbol{\pi_i}\|$. We have $\max_{j\in A}|G_{i,j} - (F_A\boldsymbol{\pi_i})_j| \leq 2$ and $\sum_j \mathbb{E}[G_{i,j} - (F_A\boldsymbol{\pi_i})_j]^2 \leq \sum_j (F_A\boldsymbol{\pi_i})_j \leq |F_A|_1$. Thus with high probability

$$\|G_{i,A} - F_A\boldsymbol{\pi_i}\| = \tilde{O}(\sqrt{|F_A|_1}).$$

Thus, we have the bound that $\|\sum_i Z_i\| = \tilde{O}(\max(\sqrt{|F_A|_1}, \sqrt{|F_X|_1})) = \tilde{O}(\sqrt{pn})$. The concentration of the mean term follows from this result.

The statement for μ is just a straight forward application of Bernstein's inequality.

Finally, for the modified adjacency matrix $G^{\alpha_0}_{X,A}$, we have

$$\epsilon_G \leq \sqrt{\alpha_0 + 1}\|G_{X,A} - \mathbb{E}[G_{X,A}|\Pi]\| + (\sqrt{\alpha_0 + 1} - 1)\sqrt{|X|}\|\boldsymbol{\mu_{X\to A}} - \mathbb{E}[\boldsymbol{\mu_{X\to A}}|\Pi]\|.$$

substituting in the bounds for adjacency matrix and mean vector gives the result.

□

Another factor in the whitening error is the smallest singular value of the expectation of $G^{\alpha_0}_{X,A}$, which we estimate below.

Define

$$\psi_i := \mathrm{Diag}(\hat{\alpha})^{-1/2}(\sqrt{\alpha_0 + 1}\boldsymbol{\pi_i} - (\sqrt{\alpha_0 + 1} - 1)\widehat{\alpha}). \tag{8.24}$$

Let Ψ_X be the matrix with columns ψ_i, for $i \in X$. We have

$$\mathbb{E}[(G^{\alpha_0}_{X,A})^\top|\Pi] = F_A \, \mathrm{Diag}(\hat{\alpha})^{1/2}\Psi_X,$$

from definition of $\mathbb{E}[(G^{\alpha_0}_{X,A})^\top|\Pi]$.

Lemma 8.10 (Smallest Singular Value)**.** *With hith probability,*

$$\varepsilon_1 := \|I - |X|^{-1}\Psi_X\Psi_X^\top\| \leq \tilde{O}(\sqrt{k/n})$$
$$\sigma_{\min}\left(\mathbb{E}[(G^{\alpha_0}_{X,A})^\top|\Pi]\right) = \tilde{\Omega}((p-q)n/k).$$

Proof: Note that ψ_i is a random vector with norm bounded by $O(\sqrt{(\alpha_0+1)/\widehat{\alpha}_{min}})$ from Lemma 8.8 and $\mathbb{E}[\psi_i\psi_i^\top] = I$. We know bound ε_1 using Matrix Bernstein Inequality. Each matrix $\psi_i\psi_i^\top/|X|$ has spectral norm at most $O((\alpha_0+1)/\widehat{\alpha}_{min}|X|)$. The variance σ^2 is bounded by

$$\left\| \frac{1}{|X|^2} \mathbb{E}[\sum_{i \in X} \|\psi_i\|^2 \psi_i \psi_i^\top] \right\| \leq \left\| \frac{1}{|X|^2} \max \|\psi_i\|^2 \mathbb{E}[\sum_{i \in X} \psi_i \psi_i^\top] \right\| \leq O((\alpha_0 + 1)/\widehat{\alpha}_{\min}|X|).$$

Since $O((\alpha_0+1)/\alpha_{\min}|X|) < 1$, the variance dominates in Matrix Bernstein's inequality.

Let $B := |X|^{-1}\Psi_X \Psi_X^\top$. We have with probability $1-\delta$,

$$\begin{aligned} \sigma_{\min}(\mathbb{E}[(G^{\alpha_0}_{X,A})^\top|\Pi]) &= \sqrt{|X|\sigma_{\min}(F_A \operatorname{Diag}(\hat{\alpha})^{1/2} B \operatorname{Diag}(\hat{\alpha})^{1/2} F_A^\top)}, \\ &= \Omega(\sqrt{\widehat{\alpha}_{\min}|X|(1-\epsilon_1)} \cdot \sigma_{\min}(F_A)). \end{aligned}$$

From Lemma 8.8, with probability $1-\delta$,

$$\sigma_{\min}(F_A) \geq \left(\sqrt{\frac{|A|\widehat{\alpha}_{\min}}{\alpha_0 + 1}} - O((|A| \log k/\delta)^{1/4}) \right) \cdot \sigma_{\min}(P).$$

Similarly other results follow. □

Whitening Error Consider rank-k SVD of $|X|^{-1/2}(G^{\alpha_0}_{X,A})^\top_{k-svd} = \hat{U}_A \hat{D}_A \hat{V}_A^\top$, and the whitening matrix is given by $\hat{W}_A := \hat{U}_A \hat{D}_A^{-1}$ and thus $|X|^{-1}\hat{W}_A^\top (G^{\alpha_0}_{X,A})^\top (G^{\alpha_0}_{X,A}) \hat{W}_A = I$. Now consider the singular value decomposition of

$$|X|^{-1}\hat{W}_A^\top \mathbb{E}[(G^{\alpha_0}_{X,A})^\top|\Pi] \cdot \mathbb{E}[(G^{\alpha_0}_{X,A})|\Pi] \hat{W}_A = \Phi \tilde{D} \Phi^\top.$$

$\hat{W}_A$ does not whiten the exact moments in general. On the other hand, consider

$$W_A := \hat{W}_A \Phi_A \tilde{D}_A^{-1/2} \Phi_A^\top.$$

Observe that W_A whitens $|X|^{-1/2}\mathbb{E}[(G^{\alpha_0}_{X,A})|\Pi]$

$$|X|^{-1}W_A^\top \mathbb{E}[(G^{\alpha_0}_{X,A})^\top|\Pi]\mathbb{E}[(G^{\alpha_0}_{X,A})|\Pi]W_A = (\Phi_A \tilde{D}_A^{-1/2}\Phi_A^\top)^\top \Phi_A \tilde{D}_A \Phi_A^\top \Phi_A \tilde{D}_A^{-1/2}\Phi_A^\top = I$$

Now the ranges of W_A and $\hat{W}_A$ may differ and we control the perturbations below.

Also note that $\hat{R}_{A,B}$, $\hat{R}_{A,C}$ are given by

$$\hat{R}_{AB} := |X|^{-1}\hat{W}_B^\top (G^{\alpha_0}_{X,B})^\top_{k-svd}(G^{\alpha_0}_{X,A})_{k-svd}\hat{W}_A. \tag{8.25}$$

$$R_{AB} := |X|^{-1}W_B^\top \mathbb{E}[(G^{\alpha_0}_{X,B})^\top|\Pi] \cdot \mathbb{E}[G^{\alpha_0}_{X,A}|\Pi] \cdot W_A. \tag{8.26}$$

Recall ϵ_G is given by (8.21), and $\sigma_{\min}\left(\mathbb{E}[G^{\alpha_0}_{X,A}|\Pi]\right)$ is given in Theorem 8.10.

Remark 1: Here it is really important to define W_A using $\hat{W}_A$, as in general there are infinitely many matrices that whitens $\mathbb{E}[(G^{\alpha_0}_{X,A}|\Pi]$, and these matrices are not close in spectral norm.

Remark 2: It is tempting to define the whitening error as $\left\|\hat{W}_A - W_A\right\|$, however for tight results in the more general settings, it is better to define the whitening error as the following Lemma.

Remark 3: The whitening matrix perturbation is very central in the perturbation bounds, in our algorithm works when $\tilde{O}(\varepsilon_{W_A}) < 1$.

Lemma 8.11 (Whitening matrix perturbations). *With high probability,*

$$\epsilon_{W_A} := \|\operatorname{Diag}(\widehat{\alpha})^{1/2}F_A^\top(\hat{W}_A - W_A)\| = O\left(\frac{(1-\varepsilon_1)^{-1/2}\epsilon_G}{\sigma_{\min}\left(\mathbb{E}[G^{\alpha_0}_{X,A}|\Pi]\right)}\right) \tag{8.27}$$

$$\epsilon_{\tilde{W}_B} := \|\operatorname{Diag}(\widehat{\alpha})^{1/2}F_B^\top(\hat{W}_B\hat{R}_{AB} - W_B R_{AB})\| = O\left(\frac{(1-\varepsilon_1)^{-1/2}\epsilon_G}{\sigma_{\min}\left(\mathbb{E}[G^{\alpha_0}_{X,B}|\Pi]\right)}\right) \tag{8.28}$$

Thus,

$$\epsilon_{W_A} = \epsilon_{\tilde{W}_B} = \tilde{O}\left(\frac{\sqrt{p}k}{(p-q)\sqrt{n}}\right) \tag{8.29}$$

Proof: Using the fact that $W_A = \hat{W}_A \Phi_A \tilde{D}_A^{-1/2} \Phi_A^\top$ or $\hat{W}_A = W_A \Phi_A \tilde{D}_A^{1/2} \Phi_A^\top$ we have that

$$\begin{aligned}
\| \operatorname{Diag}(\widehat{\alpha})^{1/2} F_A^\top (\hat{W}_A - W_A)\| &\leq \| \operatorname{Diag}(\widehat{\alpha})^{1/2} F_A^\top W_A (I - \Phi_A \tilde{D}_A^{1/2} \Phi_A^\top)\| \\
&= \| \operatorname{Diag}(\widehat{\alpha})^{1/2} F_A^\top W_A (I - \tilde{D}_A^{1/2})\| \\
&\leq \| \operatorname{Diag}(\widehat{\alpha})^{1/2} F_A^\top W_A (I - \tilde{D}_A^{1/2})(I + \tilde{D}_A^{1/2})\| \\
&\leq \| \operatorname{Diag}(\widehat{\alpha})^{1/2} F_A^\top W_A\| \cdot \|I - \tilde{D}_A\|
\end{aligned}$$

using the fact that $\tilde{D}_A$ is a diagonal matrix.

Now note that W_A whitens $|X|^{-1/2}\mathbb{E}[G_{X,A}^{\alpha_0}|\Pi] = |X|^{-1/2} F_A \operatorname{Diag}(\alpha^{1/2}) \Psi_X$, where Ψ_X is defined in (8.24). Further it is shown in Lemma 8.10 that Ψ_X satisfies with probability $1-\delta$ that

$$\varepsilon_1 := \|I - |X|^{-1} \Psi_X \Psi_X^\top\| \leq O\left(\sqrt{\frac{(\alpha_0+1)}{\widehat{\alpha}_{\min}|X|}} \cdot \log\frac{k}{\delta}\right)$$

Since $\varepsilon_1 \ll 1$ when X, A are of size $\Omega(n)$, we have that $|X|^{-1/2}\Psi_X$ has singular values around 1. Since W_A whitens $|X|^{-1/2}\mathbb{E}[G_{X,A}^{\alpha_0}|\Pi]$, we have

$$|X|^{-1} W_A^\top F_A \operatorname{Diag}(\alpha^{1/2}) \Psi_X \Psi_X^\top \operatorname{Diag}(\alpha^{1/2}) F_A^\top W_A = I.$$

Thus, with probability $1-\delta$,

$$\| \operatorname{Diag}(\widehat{\alpha})^{1/2} F_A^\top W_A\| = O((1-\varepsilon_1)^{-1/2}).$$

Let $\mathbb{E}[(G^{\alpha_0}_{X,A})|\Pi] = (G^{\alpha_0}_{X,A})_{k-svd} + \Delta$. We have

$$\begin{aligned}
\|I - \tilde{D}_A\| &= \|I - \Phi_A \tilde{D}_A \Phi_A^\top\| \\
&= \|I - |X|^{-1} \hat{W}_A^\top \mathbb{E}[(G^{\alpha_0}_{X,A})^\top|\Pi] \cdot \mathbb{E}[(G^{\alpha_0}_{X,A})|\Pi] \hat{W}_A\| \\
&= O\left(|X|^{-1} \|\hat{W}_A^\top \left(\Delta^\top (G^{\alpha_0}_{X,A})_{k-svd} + \Delta (G^{\alpha_0}_{X,A})^\top_{k-svd}\right) \hat{W}_A\|\right) \\
&= O\left(|X|^{-1/2} \|\hat{W}_A^\top \Delta^\top \hat{V}_A + \hat{V}_A^\top \Delta \hat{W}_A\|\right), \\
&= O\left(|X|^{-1/2} \|\hat{W}_A\| \|\Delta\|\right) \\
&= O\left(|X|^{-1/2} \|W_A\| \epsilon_G\right),
\end{aligned}$$

since $\|\Delta\| \leq \epsilon_G + \sigma_{k+1}(G^{\alpha_0}_{X,A}) \leq 2\epsilon_G$, using Weyl's theorem for singular value perturbation and the fact that $\epsilon_G \cdot \|W_A\| \ll 1$ and $\|W_A\| = |X|^{1/2} / \sigma_{\min}\left(\mathbb{E}[G^{\alpha_0}_{X,A}|\Pi]\right)$.

We now consider perturbation of $W_B R_{AB}$. By definition, we have that

$$\mathbb{E}[G^{\alpha_0}_{X,B}|\Pi] \cdot W_B R_{AB} = \mathbb{E}[G^{\alpha_0}_{X,A}|\Pi] \cdot W_A.$$

and

$$\|W_B R_{AB}\| = |X|^{1/2} \sigma_{\min}(\mathbb{E}[G^{\alpha_0}_{X,B}|\Pi])^{-1}.$$

Along the lines of previous derivation for ϵ_{W_A}, let

$$|X|^{-1} (\hat{W}_B \hat{R}_{AB})^\top \cdot \mathbb{E}[(G^{\alpha_0}_{X,B})^\top|\Pi] \cdot \mathbb{E}[G^{\alpha_0}_{X,B}|\Pi] \hat{W}_B \hat{R}_{AB} = \Phi_B \tilde{D}_B \Phi_B^\top.$$

Again using the fact that $|X|^{-1} \Psi_X \Psi_X^\top \approx I$, we have

$$\| \operatorname{Diag}(\widehat{\alpha})^{1/2} F_B^\top W_B R_{AB}\| \approx \| \operatorname{Diag}(\widehat{\alpha})^{1/2} F_A^\top W_A\|,$$

and the rest of the proof follows. □

Expected Error in a Single Whitened Vector Before going to the concentration bound of the tensor, let us first take a look at the expected error in a whitened tensor $\hat{W}^\top G_{i,A}$. This is used in both the tensor concentration bounds and for showing the existence of good initilizers for tensor power method.

Lemma 8.12 (Concentration of a random whitened vector)**.** *Conditioned on* Π *matrix, for all* $i \notin X$*,*

$$\sqrt{\mathbb{E}[\left\|\hat{W}_A^T G_{i,A}^T - W_A^T F_A \boldsymbol{\pi_i}\right\|^2 |\Pi]} \leq O(\varepsilon_{W_A} \widehat{\alpha}_{min}^{-1/2}) = \tilde{O}(\sqrt{p}k^{3/2}/(p-q)\sqrt{n}).$$

In particular, with probability at least $1/4$*,*

$$\left\|\hat{W}_A^T G_{i,A}^T - W_A^T F_A \boldsymbol{\pi_i}\right\|^2 \leq O(\varepsilon_{W_A} \widehat{\alpha}_{min}^{-1/2}) = \tilde{O}(\sqrt{p}k^{3/2}/(p-q)\sqrt{n}).$$

Proof. We have

$$\left\|\hat{W}_A^T G_{i,A}^T - W_A^T F_A \boldsymbol{\pi_i}\right\| \leq \left\|(\hat{W}_A - W_A)^T F_A \boldsymbol{\pi_i}\right\| + \left\|\hat{W}_A^T (G_{i,A}^T - \hat{W}_A^T F_A \pi)\right\|.$$

The first term is bounded by $\varepsilon_{W_A} \widehat{\alpha}_{min}^{-1/2}$ by Lemma 8.11.

Now we bound the second term. Note that $G_{i,A}^\top$ is independent of $\hat{W}_A^\top$, since they are related to disjoint subset of edges. The whitened neighborhood vector can be viewed as a sum of vectors:

$$\hat{W}_A^\top G_{i,A}^\top = \sum_{j \in A} G_{i,j} (\hat{W}_A^\top)_j = \sum_{j \in A} G_{i,j} (\hat{D}_A \hat{U}_A^\top)_j = \hat{D}_A \sum_{j \in A} G_{i,j} (\hat{U}_A^\top)_j.$$

Conditioned on $\boldsymbol{\pi_i}$ and F_A, $G_{i,j}$ are Bernoulli variables with probability $(F_A\boldsymbol{\pi_i})_j$. The goal is to compute the variance of the sum, $\sum_{j\in A}(F_A\boldsymbol{\pi_i})_j \left\|(\hat{U}_A^\top)_j\right\|^2$, and then use Chebyshev's inequality.

By Wedin's theorem, we know the span of columns of $\hat{U}_A$ is $O(\epsilon_G/\sigma_{min}(G_X^{\alpha_0}, A)) = O(\epsilon_{W_A})$ close to the span of columns of F_A. The span of columns of F_A is the same as the span of rows in Π_A. In particular, let $Proj_\Pi$ be the projection matrix of the span of rows in Π_A, we have

$$\left\|\hat{U}_A\hat{U}_A^\top - Proj_\Pi\right\| \leq O(\epsilon_{W_A}).$$

Using the spectral norm bound, we have the Frobenius norm

$$\left\|\hat{U}_A\hat{U}_A^\top - Proj_\Pi\right\|_F \leq O(\epsilon_{W_A}\sqrt{k})$$

since they are rank k matrices. This implies that

$$\sum_{j\in A}\left(\left\|(\hat{U}_A^T)_j\right\| - \|Proj_\Pi^j\|\right)^2 = O(\epsilon_{W_A}^2 k).$$

We also know

$$\|Proj_\Pi^j\| \leq \frac{\|\pi_j\|}{\sigma_{\min}(\Pi_A)} = O\left(\sqrt{\frac{(\alpha_0+1)}{n\widehat{\alpha}_{\min}}}\right),$$

by the SVD of Π_A.

Now we can bound the variance of the vectors $\sum_{j\in A} G_{i,j}(\hat{U}_A^T)_j$, since the variance of $G_{i,j}$ is bounded by $(F_A\boldsymbol{\pi_i})_j$ (its probability), and the variance of the vectors is at most

$$\begin{aligned}\sum_{j\in A}(F_A\boldsymbol{\pi_i})_j\left\|(\hat{U}_A^\top)_j\right\|^2 &\leq 2\sum_{j\in A}(F_A\boldsymbol{\pi_i})_j\left\|Proj_\Pi^j\right\|^2 + 2\sum_{j\in A}(F_A\boldsymbol{\pi_i})_j\left(\left\|(\hat{U}_A^T)_j\right\| - \left\|Proj_\Pi^j\right\|\right)^2\\ &\leq 2\sum_{j\in A}(F_A\boldsymbol{\pi_i})_j\max_{j\in A}\left(\left\|Proj_\Pi^j\right\|^2\right) + \max_{i,j}P_{i,j}\sum_{j\in A}\left(\left\|(\hat{U}_A^T)_j\right\| - \left\|Proj_\Pi^j\right\|\right)^2\\ &\leq O\left(\frac{|F_A|_1(\alpha_0+1)}{n\widehat{\alpha}_{\min}}\right)\end{aligned}$$

Chebyshev's inequality implies that with probability at least 1/4 (or any other constant),

$$\left\| \sum_{j \in A} (G_{i,j} - F_A \boldsymbol{\pi_i})(\hat{U}_A^T)_j \right\|^2 \leq O\left(\frac{|F_A|_1(\alpha_0 + 1)}{n\widehat{\alpha}_{\min}} \right).$$

And thus, we have

$$\hat{W}_A^T (G_{i,A} - F_A \boldsymbol{\pi_i}) \leq \sqrt{\frac{|F_A|_1(\alpha_0 + 1)}{n\widehat{\alpha}_{\min}}} \cdot \left\| \hat{W}_A^T \right\| \leq O\left(\epsilon_{W_A} \widehat{\alpha}_{\min}^{-1/2} \right).$$

Combining the two terms, we have the result. □

Concentration of the Tensor Finally we are ready to prove the tensor concentration bound.

Theorem 8.13 (Perturbation of whitened tensor)**.** *With high probability,*

$$\begin{aligned} \varepsilon_T &:= \left\| \mathrm{T}^{\alpha_0}_{Y \to \{A,B,C\}}(\hat{W}_A, \hat{W}_B \hat{R}_{AB}, \hat{W}_C \hat{R}_{AC}) - \mathbb{E}[\mathrm{T}^{\alpha_0}_{Y \to \{A,B,C\}}(W_A, \tilde{W}_B, \tilde{W}_C) | \Pi_A, \Pi_B, \Pi_C] \right\| \\ &= \tilde{O}\left(\frac{\sqrt{p} k^{3/2}}{(p-q)\sqrt{n}} \right). \end{aligned} \tag{8.30}$$

Proof: In tensor T^{α_0} in (8.7), the first term is

$$(\alpha_0 + 1)(\alpha_0 + 2) \sum_{i \in Y} \left(G_{i,A}^\top \otimes G_{i,B}^\top \otimes G_{i,C}^\top \right).$$

We claim that this term dominates in the perturbation analysis since the mean vector perturbation is of lower order. We now consider perturbation of the whitened tensor

$$\Lambda_0 = \frac{1}{|Y|} \sum_{i \in Y} \left((\hat{W}_A^\top G_{i,A}^\top) \otimes (\hat{R}_{AB}^\top \hat{W}_B^\top G_{i,B}^\top) \otimes (\hat{R}_{AC}^\top \hat{W}_C^\top G_{i,C}^\top) \right).$$

We show that this tensor is close to the corresponding term in the expectation in three steps.

First we show it is close to

$$\Lambda_1 = \frac{1}{|Y|}\sum_{i\in Y}\Big((\hat{W}_A^\top F_A\boldsymbol{\pi_i})\otimes(\hat{R}_{AB}^\top\hat{W}_B^\top F_B\boldsymbol{\pi_i})\otimes(\hat{R}_{AC}^\top\hat{W}_C^\top F_C\boldsymbol{\pi_i})\Big).$$

Then this vector is close to the expectation over Π_Y.

$$\Lambda_2 = \mathbb{E}_{\pi\sim\mathrm{Dir}(\alpha)}\Big((\hat{W}_A^\top F_A\pi)\otimes(\hat{R}_{AB}^\top\hat{W}_B^\top F_B\pi)\otimes(\hat{R}_{AC}^\top\hat{W}_C^\top F_C\pi)\Big).$$

Finally we replace the estimated whitening matrix $\hat{W}_A$ with W_A.

$$\Lambda_3 = \mathbb{E}_{\pi\sim\mathrm{Dir}(\alpha)}\Big((W_A^\top F_A\pi)\otimes(\tilde{W}_B^\top F_B\pi)\otimes(\tilde{W}_C^\top F_C\pi)\Big).$$

For $\Lambda_0-\Lambda_1$, we write $\hat{W}_A^\top G_{i,A}^\top$ as the sum of two terms $\hat{W}_A^\top F_A\boldsymbol{\pi_i}$ and $\hat{W}_A^\top G_{i,A}^\top-\hat{W}_A^\top F_A\boldsymbol{\pi_i}$ (and similarly for B and C), then expand Λ_0 into eight terms. The first term will be equal to Λ_1. In the remaining terms, the dominant term in the perturbation bound is,

$$\frac{1}{|Y|}\sum_{i\in Y}\hat{W}_A^\top(G_{i,A}^\top - F_A\boldsymbol{\pi_i})\otimes(\hat{R}_{AB}^\top\hat{W}_B^\top F_B\boldsymbol{\pi_i})\otimes(\hat{R}_{AC}^\top\hat{W}_C^\top F_C\boldsymbol{\pi_i}).$$

We can view the tensor as a matrix (and by definition the spectral norm can only go down), then apply matrix Bernstein bound. Both terms of the variance can be bounded by $|Y|\mathbb{E}[\left\|\hat{W}_A^\top(G_{i,A}^\top - F_A\boldsymbol{\pi_i}\right\|)^2]\left\|\tilde{W}_B^\top F_B\right\|^2\left\|\tilde{W}_C^\top F_C\right\|^2 = O(n\varepsilon_{W_A}^2k^3)$, the norm of each vector can also be bounded. By matrix Bernstein with high probability this term is bounded by $O(\varepsilon_{W_A}k^{3/2}/\sqrt{n})\le O(\varepsilon_{W_A}\sqrt{k})$ because $k^2<n$.

For $\Lambda_1-\Lambda_2$, since $\hat{W}_AF_A\,\mathrm{Diag}(\widehat{\alpha})^{1/2}$ has spectral norm almost 1, by Lemma 8.14 the spectral norm of the perturbation is at most

$$\left\|\hat{W}_A F_A \operatorname{Diag}(\widehat{\alpha})^{1/2}\right\|^3 \left\|\frac{1}{|Y|}\sum_{i\in Y}(\operatorname{Diag}(\widehat{\alpha})^{-1/2}\boldsymbol{\pi_i})^{\otimes 3} - \mathbb{E}_{\pi\sim \operatorname{Dir}(\alpha)}(\operatorname{Diag}(\widehat{\alpha})^{-1/2}\boldsymbol{\pi_i})^{\otimes 3}\right\|$$
$$\leq O(1/\widehat{\alpha}_{min}\sqrt{n}\sqrt{\log n/\delta}).$$

For the final term $\Lambda_2 - \Lambda_3$, the dominating term is

$$(\hat{W}_A - W_A)F_A \operatorname{Diag}(\widehat{\alpha})^{1/2}\left\|\Lambda_3\right\| \leq \varepsilon_{W_A}\left\|\Lambda_3\right\|$$

Putting all these together, the third term $\|\Lambda_2 - \Lambda_3\|$ dominates and we get the result. □

Lemma 8.14. *With high probability,*

$$\left\|\frac{1}{|Y|}\sum_{i\in Y}(\operatorname{Diag}(\widehat{\alpha})^{-1/2}\boldsymbol{\pi_i})^{\otimes 3} - \mathbb{E}_{\boldsymbol{\pi}\sim \operatorname{Dir}(\alpha)}(\operatorname{Diag}(\widehat{\alpha})^{-1/2}\boldsymbol{\pi})^{\otimes 3}\right\| \leq O(\cdot\frac{1}{\widehat{\alpha}_{min}\sqrt{n}}\sqrt{\log n/\delta})$$
$$= \tilde{O}(k/\sqrt{n})$$

Proof. The spectral norm of this tensor cannot be larger than the spectral norm of a $k\times k^2$ matrix that we obtain be "collapsing" the last two dimensions (by definitions of norms). Let $\boldsymbol{\phi_i} = \operatorname{Diag}(\hat{\alpha})^{-1/2}\boldsymbol{\pi_i}$, the "collapsed" tensor is just the matrix $\boldsymbol{\phi_i}(\boldsymbol{\phi_i}\otimes\boldsymbol{\phi_i})^\top$ (here we view $\boldsymbol{\phi_i}\otimes\boldsymbol{\phi_i}$ as a vector in $\mathbb{R}^{k^2}$). We apply Matrix Bernstein on the matrices $Z_i = \boldsymbol{\phi_i}(\boldsymbol{\phi_i}\otimes\boldsymbol{\phi_i})^\top$.

Clearly, $\left\|\sum_{i\in Y}\mathbb{E}[Z_i Z_i^\top]\right\| \leq |Y|\max\|\boldsymbol{\phi}\|^4\left\|\mathbb{E}[\boldsymbol{\phi}\boldsymbol{\phi}^\top]\right\| \leq |Y|\widehat{\alpha}_{min}^{-2}$ because $\left\|\mathbb{E}[\boldsymbol{\phi}\boldsymbol{\phi}^\top]\right\| \leq 2$.

For the other variance term $\left\|\sum_{i\in Y}\mathbb{E}[Z_i^\top Z_i]\right\|$, we have

$$\left\|\sum_{i\in Y}\mathbb{E}[Z_i^\top Z_i]\right\| \leq |Y|\widehat{\alpha}_{min}\left\|\mathbb{E}[(\boldsymbol{\phi}\otimes\boldsymbol{\phi})(\boldsymbol{\phi}\otimes\boldsymbol{\phi})^\top]\right\|$$

.

It remains to bound the norm of $\mathbb{E}[(\boldsymbol{\phi} \otimes \boldsymbol{\phi})(\boldsymbol{\phi} \otimes \boldsymbol{\phi})^\top]$. This is almost a fixed matrix for constant α_0, the details are lengthy because of the number of different sized entries, and are omitted here. □

8.5.2 Good Initializer for Tensor Power Method

The eigenvectors of the orthogonally decomposable tensor obtained in Lemma 8.3 are columns of Φ.

Recall that a vector $\boldsymbol{u}$ is (γ, Q)-good with respect to Φ_i if $|\boldsymbol{u}^\top \Phi_i| > Q$, and $|\lambda_i \boldsymbol{u}^\top \Phi_i| \geq (1+\gamma)|\lambda_j \boldsymbol{u}^\top \Phi_j|$ for all $\lambda_j \leq \lambda_i$. The following Lemma shows there exists very good initial vectors in among $\hat{W}_A^\top G_{i,A}$ for $i \in B$.

Lemma 8.15. *With high probability, for any* $10/9 > \gamma > 1$ *there are* $(\gamma, \tilde{\Omega(1)})$*-good initializers with respect to all* Φ_i*'s within the set* $\{\hat{W}_A^\top G_{i,A} : i \in B\}$.

Intuitively, this Lemma should be true because by Lemma 8.12, $\hat{W}_A^\top G_{i,A}$ is close to $W_A^\top F_A \boldsymbol{\pi_i} = (\mathrm{Diag}(\widehat{\alpha})^{-1/2}\boldsymbol{\pi_i})^\top \Phi$. This shows when this vector is expressed in the basis of Φ, the coefficients are $\mathrm{Diag}(\widehat{\alpha})^{-1/2}\boldsymbol{\pi}$. The vector $\boldsymbol{\pi}$ is chosen from Dirichlet distribution so we expect $\boldsymbol{\pi}$ to have a single large entry, thus the vector should be a good initialzer.

We first prove the intuition that Dirichlet vectors should have a single large entry with good probability. Without loss of generality we assume $\widehat{\alpha}_1 \leq \widehat{\alpha}_2 \leq \cdots \leq \widehat{\alpha}_k$ (this is trivially true in the special case, but the Lemma below applies to more general cases).

Lemma 8.16. *Let* $\boldsymbol{v}$ *be the unit vector* $\mathrm{Diag}(\hat{\alpha})^{-1/2}\boldsymbol{\pi} / \left\|\mathrm{Diag}(\hat{\alpha})^{-1/2}\boldsymbol{\pi}\right\|$ *where* $\boldsymbol{\pi}$ *is chosen according to* $\mathrm{Dir}(\alpha)$. *For any* $1 < \gamma \leq 5/4$, *with probability* $\Omega(\widehat{\alpha}_i / k^\gamma)$, *the i-th entry of* $\boldsymbol{v}$ *is at least* $\widehat{\alpha}_i^{-1/2}/(\widehat{\alpha}_i^{-1/2} + \sqrt{(\alpha_0+1)k})$, *the j-th entry of* $\boldsymbol{v}$ *is at most* $1/\gamma$ *of the i-th entry for all* $j > i$, *and* $\boldsymbol{\pi}_i > 9/10(\alpha_0+1)$.

Proof. We are assuming $\alpha_i < 1$ throughout. This is the reasonable case where Dirichlet distributions are sparse.

When $\alpha_0 < 1/10$, we derive the bound directly from properties of Dirichlet distribution. Let $(X_1, X_2, ..., X_k) \sim \mathrm{Dir}(\alpha)$, by properties of Dirichlet distribution, we know $\mathbb{E}[\boldsymbol{\pi}_i] = \widehat{\alpha}_i$ and $\mathbb{E}[\boldsymbol{\pi}_i^2] = \widehat{\alpha}_i \frac{\alpha_i+1}{\alpha_0+1}$. Let $p_i = \Pr[\boldsymbol{\pi}_i \geq 9/10(\alpha_0+1)]$. We have

$$\begin{aligned}\mathbb{E}[\boldsymbol{\pi}_i^2] &= p_i \mathbb{E}[\boldsymbol{\pi}_i^2 | \boldsymbol{\pi}_i \geq \frac{9}{10(\alpha_0+1)}] + (1-p_i)\mathbb{E}[\boldsymbol{\pi}_i^2 | \boldsymbol{\pi}_i < \frac{9}{10(\alpha_0+1)}] \\ &\leq p_i + (1-p_i)\frac{9}{10(\alpha_0+1)}\mathbb{E}[\boldsymbol{\pi}_i | \boldsymbol{\pi}_i < \frac{9}{10(\alpha_0+1)}] \\ &\leq p_i + (1-p_i)\frac{9}{10(\alpha_0+1)}\mathbb{E}[\boldsymbol{\pi}_i]\end{aligned}$$

Here the second line is using the fact that when $\boldsymbol{\pi}_i < t$, $\boldsymbol{\pi}_i^2 \leq t\boldsymbol{\pi}_i$ (let $t = 9/10(\alpha_0+1)$ and take expectation). Substitute $\mathbb{E}[\boldsymbol{\pi}_i]$ and $\mathbb{E}[\boldsymbol{\pi}_i^2]$, we know $p_i \geq \widehat{\alpha}_i/10(\alpha_0+1) = \Omega(\widehat{\alpha}_i)$. Also, when $\boldsymbol{\pi}_i \geq 9/10(\alpha_0+1) \geq 9/11$, all other π_j's are at most $2/11$, so $\widehat{\alpha}_i^{-1/2}\boldsymbol{\pi}_i$ is at least $4.5 \cdot \widehat{\alpha}_j^{-1/2}\pi_j$ for $j > i$.

The only thing left in this case is to show $\widehat{\alpha}_i^{-1/2}\boldsymbol{\pi}_i / \sqrt{\sum_{j=1}^k \widehat{\alpha}_j^{-1}\boldsymbol{\pi}_j^2}$ is large.

Notice that for Dirichlet distributions, $\boldsymbol{\pi}_i$ is independent of $\frac{1}{1-\boldsymbol{\pi}_i}(\boldsymbol{\pi}_1, \boldsymbol{\pi}_2, ..., \boldsymbol{\pi}_{i-1}, \boldsymbol{\pi}_{i+1}, ..., \boldsymbol{\pi}_k)$. When we condition on $\boldsymbol{\pi}_i \geq 9/11$, the expectation $\mathbb{E}[\sum_{j\neq i} \widehat{\alpha}_i^{-1}\boldsymbol{\pi}_j^2 | \boldsymbol{\pi}_i \geq 9/11]$ can only be smaller than $\mathbb{E}[\sum_{j\neq i} \widehat{\alpha}_i^{-1}\boldsymbol{\pi}_j^2] \leq 2k/(\alpha_0+1) \leq 2k$. By Markov, with probability 1/4 the true value is below 4 times the expectation, so $\Pr[\sum_{j\neq i} \widehat{\alpha}_i^{-1}\boldsymbol{\pi}_j^2 \leq 8k | \boldsymbol{\pi}_i \geq 9/11] \geq 1/4$.

Now, with probability $p_i/4$ we know $\boldsymbol{\pi}_i \geq 9/11$, and $\sum_{j\neq i} \widehat{\alpha}_i^{-1}\boldsymbol{\pi}_j^2 \leq 8k$, which means

$$\widehat{\alpha}_i^{-1/2}\boldsymbol{\pi}_i / \sqrt{\sum_{j=1}^k \widehat{\alpha}_j^{-1}\boldsymbol{\pi}_j^2} \geq \widehat{\alpha}_i^{-1/2}/(\widehat{\alpha}_i^{-1/2} + \sqrt{k})$$

The second case is when $\alpha_0 \geq 1/10$. In this case we need to several properties of Dirichlet distribution, Gamma distribution and Gamma functions in Proposition 8.21.

Let $Y_i \sim \Gamma(\alpha_i, 1)$. By the proposition it is clear that $\boldsymbol{v}$ is just the normalized version of $(..., \widehat{\alpha}_i^{-1/2} Y_i, ...)$.

By property 5 in Proposition 8.21, if we pick $t = \log k$, then each Y_i is bigger than t with probability at most $1/k$. Since they are independent, the probability that there exists some $j \neq i$ such that $Y_j > t$ is at most $(1 - 1/k)^k \leq e^{-1}$. Also, the expected value of $\sum_{j\neq i} \widehat{\alpha}_j^{-1/2} Y_j^2 \leq 2k\alpha_0$, the probability that it is larger than $20k\alpha_0$ is at most $1/10$. The expected value of $\sum_{j\neq i} Y_j \leq \alpha_0$, the probability that it is larger than $10\alpha_0$ is at most $1/10$. By union bound, with probability at least $e^{-1} - 1/10 - 1/10 > 0.1$, for all $j \neq i$, $Y_j < \log k$, $\sum_{j\neq i} \widehat{\alpha}_j^{-1} Y_j^2 \leq 20k\alpha_0$, and $\sum_{j\neq i} Y_j \leq 10\alpha_0$.

However, when we pick $t = \gamma \log k$, we know with probability $C\alpha_i/4tk^\gamma = \Omega(\widehat{\alpha}_i/k^\gamma \log k)$ (here we know $\alpha_i \geq \widehat{\alpha}_i/10$ because $\alpha_0 > 1/10$), Y_i is at least $\gamma \log k$.

Since Y_i and all the other Y's are independent, with probability at least $\Omega(\widehat{\alpha}_i/k^\gamma \log k)$, we have $Y_i \geq \gamma \log k$, $Y_j \leq \log k (j \neq i)$, and

$$\widehat{\alpha}_i^{-1/2} Y_i / \sqrt{\sum_{j=1}^{k} \widehat{\alpha}_i^{-1} Y_j^2} \geq \widehat{\alpha}_i^{-1/2} \log k / \sqrt{\widehat{\alpha}_i^{-1} \log^2 k + 10k\alpha_0} \leq \widehat{\alpha}_i^{-1/2} / (\widehat{\alpha}_i^{-1/2} + \sqrt{\alpha_0 k}).$$

Finally, $\boldsymbol{\pi}_i \geq \frac{Y_i}{Y_i + \sum_{j\neq i} Y_j} \geq \log t / (\log t + 10\alpha_0) > 9/10(\alpha_0 + 1)$.

□

Remark: The probability can be improved when $\widehat{\alpha}_{max}/\widehat{\alpha}_{min}$ is small. In particular for the special case we are interested in, the probability of a good initializer is $\tilde{O}(k^{-\gamma_0})$. We will use this bound.

Now we are ready to prove Lemma 8.15.

Proof of Lemma 8.15: By Lemma 8.16, the probability that π_j gives a good vector for index i ($i \in k$) is at least $1/k^{\gamma_0}$. When that happens the normalization of $\boldsymbol{s_j} = W^T F_A \boldsymbol{\pi_j}$ is a $(\gamma_0, \widetilde{\Omega(1)})$-good initializer.

Before normalization, by Lemma 8.16 we know $\boldsymbol{\pi_j}(i) \geq 9/10(\alpha_0 + 1)$, therefore every good vector $\boldsymbol{s_j}$ for index i had projection at least $9\widehat{\alpha}_i^{-1/2}/10(\alpha_0 + 1)$ on Φ_i, and at most $\boldsymbol{s_j}^\top \Phi_i/\gamma_0$ projection on all $\Phi_t (t > i)$.

By Lemma 8.12, with constant probability, conditioned on $\hat{W}_A, \boldsymbol{\pi_j}$, $\left\| \hat{W}_A^\top G_{j,A}^\top - W_A^\top F_A \boldsymbol{\pi_j} \right\| \leq O(\varepsilon_{W_A}\sqrt{k})$. Therefore with constant probability the inner product with Φ_i can change by at most $O(\varepsilon_{W_A}\sqrt{k})$, when that happens $\hat{W}^\top G_{j,A}$ is a $(\gamma_0 - \Delta, \tilde{\Omega}(1))$-good intializer for arbitrary small Δ.

Since the constant probability in Lemma 8.12 is conditioned on π_j, we can multiply the probability in Lemma 8.12 and Lemma 8.16 (we use the bound in the Remark following that Lemma). Therefore, the probability that a vector is $(\gamma_0 - \Delta, \tilde{\Omega}(1))$-good is at least $\tilde{\Omega}(k^{-\gamma_0})$. The size $|B|$ is much larger than $k^{\gamma_0} \log n$ (because $n > k^2$), by union bound the set $\hat{W}_A^\top G_{i,A}$ has good initializer for all i with high probability. □

8.5.3 Reconstruction After Tensor Power Method

Reconstructing Π Let $(M)^i$ and $(M)_i$ denote the i^{th} row and i^{th} column in matrix M respectively. Let $Z \subseteq A^c$ denote any subset of nodes not in A, considered in Procedure LearnPartition Community. Define

$$\tilde{\Pi}_Z := \mathrm{Diag}(\lambda)^{-1} \hat{\Phi}^\top \hat{W}_A^\top G_{Z,A}^\top. \tag{8.31}$$

Recall that the final estimate $\hat{\Pi}_Z$ is obtained by thresholding $\tilde{\Pi}_Z$ element-wise with threshold τ in Procedure 8.2. We first analyze perturbation of $\tilde{\Pi}_Z$.

Lemma 8.17 (Reconstruction Guarantees for $\tilde{\Pi}_Z$)**.** *With high probability,*

$$\varepsilon_\pi := \max_{i \in Z} \|(\tilde{\Pi}_Z)^i - (\Pi_Z)^i\| = O\left(\frac{\sqrt{p}k}{(p-q)\sqrt{n}} \|\Pi_Z\| \right).$$

Proof: We have $(\tilde{\Pi}_Z)^i = \lambda_i^{-1}((\hat{\Phi})_i)^\top \hat{W}_A^\top G_{Z,A}^\top$.

Notice that $\Pi_Z^i = \mathrm{Diag}(\widehat{\alpha}_i^{1/2})\Phi_i^\top W_A^\top F_A \Pi_Z$, we can write the difference between $\tilde{\Pi}_Z^i$ and Π_Z^i in four terms

$$
\begin{aligned}
\tilde{\Pi}_Z^i - \Pi_Z^i =& (\mathrm{Diag}(\lambda_i)^{-1} - Diag(\widehat{\alpha}_i^1/2))\Phi_i^\top W_A^\top F_A \Pi_Z + \mathrm{Diag}(\lambda_i)^{-1}(\hat{\Phi}_i - \Phi_i)^\top W_A^\top F_A \Pi_Z \\
&+ \mathrm{Diag}(\lambda_i)^{-1}\hat{\Phi}_i^\top (\hat{W_A}^\top - W_A^\top) F_A \Pi_Z + \mathrm{Diag}(\lambda_i)^{-1}\hat{\Phi}_i^\top \hat{W_A}^\top (G_{A,Z} - F_A \Pi_Z)
\end{aligned}
$$

We will now use perturbation bounds for each of the terms to get the result.

The first term is bounded by

$$
\| \mathrm{Diag}(\lambda_i)^{-1} - \mathrm{Diag}(\widehat{\alpha}_i^{1/2})\| \cdot \|\Phi\| \cdot \|W_A^\top F_A\| \cdot \|\Pi_Z\| \le \varepsilon_T / k \cdot 2 \cdot 2\sqrt{k} \cdot \|\Pi_Z\| \le O(\varepsilon_{W_A}) \|\Pi_Z\|).
$$

The second term is bounded by

$$
\| \mathrm{Diag}(\lambda_i)^{-1}\| \cdot \|(\Phi)_i - \hat{\Phi}_i\| \cdot \|W_A^\top F_A\| \cdot \|\Pi_Z\| \le 2/\sqrt{k} \cdot \varepsilon_T/\sqrt{k} \cdot 2\sqrt{k} \cdot \|\Pi_Z\| \le O(\varepsilon_{W_A} \|\Pi_Z\|).
$$

The third term is bounded by

$$
\| \mathrm{Diag}(\lambda_i)^{-1}\| \cdot \|\hat{\Phi}_i\| \cdot \|(\hat{W}_A^\top - W_A^\top) F_A\| \cdot \|\Pi_Z\| \le 2/\sqrt{k} \cdot 1 \cdot \varepsilon_{W_A}\sqrt{k} \cdot \|\Pi_Z\| = O(\varepsilon_{W_A} \|\Pi_Z\|),
$$

from Lemma 8.11 and finally, we bound the fourth term by

$$
\begin{aligned}
\| \mathrm{Diag}(\lambda_i)^{-1}\| \cdot \|\hat{\Phi}_i\| \cdot \left\|\hat{W}_A^\top\right\| \cdot \|G_{A,Z} - F_A \Pi_Z\| &\le O\left(1/\sqrt{k} \cdot 1/\sigma_{\min}(F_A) \cdot \sqrt{pn}\right) \\
&\le O(\varepsilon_{W_A}/\sqrt{k})
\end{aligned}
$$

from Lemma 8.9 and Lemma 8.10.

The first three terms dominates the last term (by Lemma 8.8), therefore we get the result. □

We now show that if we threshold the entries of $\tilde{\Pi}_Z$, the the resulting matrix $\hat{\Pi}_Z$ has rows close to those in Π_Z in ℓ_1 norm. Which is stronger than the ℓ_2 guarantee we had in the previous Lemma.

Lemma 8.18 (Guarantees after thresholding)**.** *For* $\hat{\Pi}_Z := \text{Thres}(\tilde{\Pi}_Z, \tau)$*, where* $\tau = \tilde{(\Theta)}(\varepsilon_\pi \sqrt{k}/\sqrt{n}) = \tilde{O}(\varepsilon_W)$ *is the threshold, we have with high probability that*

$$\varepsilon_{\pi,\ell_1} := \max_{i\in[k]} |(\hat{\Pi}_Z)^i - (\Pi_Z)^i|_1 = \tilde{O}(\varepsilon_{W_A} n/k) = \tilde{O}\left(\frac{\sqrt{p}k}{(p-q)\sqrt{n}} \cdot \frac{n}{k}\right).$$

Remark: Notice that the ℓ_1 norm of a row of Π should be around n/k by Lemma 8.8, so this bound makes sense as long as $\varepsilon_{W_A} \ll 1$.

Proof: Let $S_i := \{j : \hat{\Pi}_Z(i,j) > 0\} = \{j : \tilde{\Pi}_Z(i,j) > \tau\}$, $S_i^0 = \{j : \Pi_Z(i,j) > \tau/2\}$ and $S_i^1 = \{j : \Pi_Z(i,j) > 2\tau\}$. The idea is S_i^0 should contain almost all entries in S_i, and S_i should contain almost all entries of S_i^1 (approximately $S_i^1 \subset S_i \subset S_i^0$).

For a vector v, let v_S denote the sub-vector by considering entries in set S. We now have

$$|(\hat{\Pi}_Z)^i - (\Pi_Z)^i|_1 \le |(\hat{\Pi}_Z)^i_{S_i} - (\Pi_Z)^i_{S_i}|_1 + |(\Pi_Z)^i_{S_i^c}|_1 + |(\hat{\Pi}_Z)^i_{S_i^c}|_1$$

Let us first assume $\alpha_0 < 1$ for simplicity.

By thresholding procedure $(\hat{\Pi}_Z)^i_{S_i} = \mathbf{0}$ so that term disappears.

For the first term, from Lemma 8.20, we have $\mathbb{P}[\Pi(i,j) \ge \tau/2] \le 8\widehat{\alpha}_i \log(2/\tau)$. Since $\Pi(i,j)$ are independent for $j \in Z$, by Chernoff bound $\max_{i\in[k]} |S_i^0| < \tilde{O}(n/k)$.

We claim that most of S_i should be in S_i^0, because for every $j \in S_i \backslash S_i^0$, we have $|\Pi_Z(i,j) - \tilde{\Pi}_Z(i,j)| \ge \tau/2$, for $j \in S_i$, and number of such terms is at most $4\varepsilon_\pi^2/\tau^2 = \tilde{O}(n/k)$. Thus,

$$|(\Pi_Z)^i_{S_i} - (\hat{\Pi}_Z)^i_{S_i}|_1 \le \varepsilon_\pi \sqrt{|S_i|} \le \tilde{O}(\varepsilon_\pi \sqrt{n/k}).$$

For the other term, from Lemma 8.20, we have

$$\mathbb{E}[\Pi_Z(i,j) \cdot \delta(\Pi_Z(i,j) \le 2\tau)] \le \widehat{\alpha}_i(2\tau).$$

Concentration bounds show that $\max_{i \in [k]} (\Pi_Z)^i_{S_i^{1,c}} \le \tilde{O}(\tau \cdot n/k) \le \tilde{O}(\varepsilon_\pi \sqrt{n/k})$.

Again we claim that most indices in S_i^c are also in $S_i^{1,c}$, because for every $j \in S_i^c$ and $j \notin S_i^{1,c}$, we have $|\Pi_Z(i,j) - \tilde{\Pi}_Z(i,j)| \ge \tau$. This implies that there are at most $\tilde{O}(n/k)$ such entries and

$$\begin{aligned}
\sum_{j \in S_i^c \backslash S_i^{1,c}} \Pi_Z(i,j) &\le \sqrt{|S_i^c \backslash S_i^{1,c}|} \sqrt{\sum_{j \in S_i^c \backslash S_i^{1,c}} [\Pi_Z(i,j)]^2} \\
&\le \tilde{O}(\sqrt{n/k}) \cdot \sqrt{4 \sum_{j \in S_i^c \backslash S_i^{1,c}} [\Pi_Z(i,j) - \tilde{\Pi}_Z(i,j)]^2} \\
&\le \tilde{O}(\sqrt{n/k}\varepsilon_\pi).
\end{aligned}$$

Case $\alpha_0 \in [1,k)$: From Lemma 8.20, we see that the results hold when we replace $\widehat{\alpha}_{\max}$ with $\alpha_{\max}$, losing factors of $(\alpha_0 + 1)$. □

Reconstruction of P Finally we would like to use the community vectors Π and the adjacency matrix G to estimate the P matrix. Recall that in the generative model, we have $\mathbb{E}[G] = \Pi^\top P \Pi$. Thus, a straightforward estimate is to use $(\hat{\Pi}^\dagger)^\top G \hat{\Pi}^\dagger$. However, our guarantees on $\hat{\Pi}$ are not strong enough to control the error on $\hat{\Pi}^\dagger$ (since we only have row-wise ℓ_1 guarantees).

We propose an alternative estimator $\hat{Q}$ for $\hat{\Pi}^\dagger$ and use it to find $\hat{P}$ in Algorithm 8.1. Recall that the i-th row of $\hat{Q}$ is given by

$$\hat{Q}^i := (\alpha_0 + 1) \frac{\hat{\Pi}^i}{|\hat{\Pi}^i|_1} - \frac{\alpha_0}{n} \mathbf{1}^\top.$$

Define Q using exact communities, i.e.

$$Q^i := (\alpha_0 + 1)\frac{\Pi^i}{|\Pi^i|_1} - \frac{\alpha_0}{n}\mathbf{1}^\top.$$

Note that if we define Q as $(Q')^i := \frac{\alpha_0+1}{n\widehat{\alpha}_i}\Pi^i - \frac{\alpha_0}{n}\mathbf{1}^\top$, then $\mathbb{E}_\Pi[Q'\Pi^\top] = I$, and we show that Q' is close to Q since $\mathbb{E}[|\Pi^i|_1] = n\widehat{\alpha}_i$. For Q, we normalize by $|\Pi^i|_1$ in order to ensure that the first term of Q has equal column norms (which will be used in our proofs subsequently). We show below that $\hat{Q}$ is close to $\Pi^\dagger$, and therefore, $\hat{P} := \hat{Q}^\top G\hat{Q}$ is close to P w.h.p.

Lemma 8.19 (Reconstruction of P)**.** *With probability* $1 - 5\delta$*,*

$$\varepsilon_P := \max_{i,j\in[n]} |\hat{P}_{i,j} - P_{i,j}| \leq \tilde{O}(\varepsilon_{W_A}(p-q))$$

Remark: Note that again this bound is meaningful when $\varepsilon_{W_A} \ll 1$.

Proof: The proof goes in three steps:

$$P \approx Q\Pi^\top P\Pi Q^\top \approx \hat{Q}\Pi^\top P\Pi\hat{Q}^\top \approx \hat{Q}G\hat{Q}^\top.$$

Note that $\mathbb{E}_\Pi[\Pi Q^\top] = I$ and by Bernstein's bound, we can claim that $\Pi Q^\top$ is close to I and can show that the i-th row of $Q\Pi^\top$ satisfies

$$\Delta_i := |(Q\Pi^\top)^i - e_i^\top|_1 = \tilde{O}(k/\sqrt{n}).$$

Moreover,

$$\begin{aligned}|(\Pi^\top P\Pi Q^\top)_{i,j} - (\Pi^\top P)_{i,j}| &\leq |(\Pi^\top P)^i((Q)_j - e_j)| = |(\Pi^\top P)^i\Delta_j| \\ &\leq O(pk/\sqrt{n}) \leq O(\varepsilon_{W_A}(p-q)).\end{aligned}$$

using the fact that $(\Pi^\top P)_{i,j} \leq p$.

This can be translated to a bound between P and $Q\Pi^\top P \Pi Q^\top$ because the dominating term would be $|Q|_1 \left\|(\Pi^\top P\Pi Q^\top) - (\Pi^\top P)\right\|_\infty$.

Now we claim that $\hat{Q}$ is close to Q and by Lemma 8.17 $|Q_B^i - \hat{Q}_B^i|_1 \leq O(\varepsilon_{W_A})$, which implies

$$\begin{aligned}
|(\Pi^\top P\Pi Q^\top)_{i,j} - (\Pi^\top P\Pi\hat{Q}^\top)_{i,j}| &= |(\Pi^\top P\Pi)^i(Q^\top - \hat{Q}^\top)_j| \\
&= ((\Pi^\top P\Pi)^i - q\mathbf{1}^\top)|(Q^\top - \hat{Q}^\top)_j|_1 \\
&\leq O(\varepsilon_{W_A}(p-q)).
\end{aligned}$$

using the fact that $(Q^j - \hat{Q}^j)\mathbf{1} = 0$, due to the normalization.

Finally, $|(G\hat{Q}^\top)_{i,j}(\Pi^\top P\Pi\hat{Q}^\top)_{i,j}|$ are small by standard concentration bounds (and the differences are of lower order). Combining these $|\hat{P}_{i,j} - P_{i,j}| \leq O(\varepsilon_P)$.

□

Support Recovery *Proof of Theorem 8.6:* From Lemma 8.19,

$$|\hat{P}_{i,j} - P_{i,j}| \leq O(\varepsilon_P)$$

which implies bounds for the average of diagonals H and average of off-diagonals L:

$$|H - p| = O(\varepsilon_P), \quad |L - q| = O(\varepsilon_P).$$

In particular, $|L + (H - L)\cdot\frac{3\xi}{4} - (q + (p-q)\frac{3\xi}{4})| \leq O(\varepsilon_P)$, which means the threshold is very close to $(q + (p-q)\frac{3\xi}{4})$.

On similar lines as the proof of Lemma 8.19 and from independence of edges used to define $\hat{F}$ from the edges used to estimate $\hat{\Pi}$, we also have

$$|\hat{F}(j,i) - F(j,i)| \leq O(\varepsilon_P).$$

Note that $F_{j,i} = q + \Pi_{i,j}(p - q)$. The threshold ξ satisfies $\xi = \Omega(\varepsilon_{W_A})$ and in particular $(p - q)\xi > O(\varepsilon_P)$.

If $\Pi_{i,j} > xi$, then $F_{i,j} > q+(p-q)\xi$ and $\hat{F}_{j,i} > q+(p-q)\xi-O(\varepsilon_P) > (q+(p-q)\frac{3\xi}{4})+O(\varepsilon_P)$, thus all large entries are identified. Similarly, if $\Pi_{i,j} < xi/2$, $\hat{F}_{j,i}$ cannot be larger than the threshold. □

8.5.4 Dirichlet Properties

Lemma 8.20 (Marginal Dirichlet distribution in sparse regime)**.** *For $Z \sim B(a, b)$, the following results hold:*

Case $b \leq 1$, $C \in [0, 1/2]$:

$$\Pr[Z \geq C] \leq 8 \log(1/C) \cdot \frac{a}{a+b} \tag{8.32}$$

$$\mathbb{E}[Z \cdot \delta(Z \leq C)] \leq C \cdot \mathbb{E}[Z] = C \cdot \frac{a}{a+b} \tag{8.33}$$

Case $b \geq 1$, $C \leq (b+1)^{-1}$: *we have*

$$\Pr[Z \geq C] \leq a \log(1/C) \tag{8.34}$$

$$\mathbb{E}[Z \cdot \delta(Z \leq C)] \leq 6aC \tag{8.35}$$

Remark: The guarantee for $b \geq 1$ is worse and this agrees with the intuition that the Dirichlet vectors are more spread out (or less sparse) when $b = \alpha_0 - \alpha_i$ is large.

Proof. We have

$$
\begin{aligned}
\mathbb{E}[Z \cdot \delta(Z \leq C)] &= \int_0^C \frac{1}{B(a,b)} x^a (1-x)^{b-1} dx \\
&\leq \frac{(1-C)^{b-1}}{B(a,b)} \int_0^C x^a dx \\
&= \frac{(1-C)^{b-1} C^{a+1}}{(a+1)B(a,b)}
\end{aligned}
$$

For $\mathbb{E}[Z \cdot \delta(Z \geq C)]$, we have,

$$
\begin{aligned}
\mathbb{E}[Z \cdot \delta(Z \geq C)] &= \int_C^1 \frac{1}{B(a,b)} x^a (1-x)^{b-1} dx \\
&\geq \frac{C^a}{B(a,b)} \int_C^1 (1-x)^{b-1} dx \\
&= \frac{(1-C)^b C^a}{bB(a,b)}
\end{aligned}
$$

The ratio between these two is at least

$$
\frac{\mathbb{E}[Z \cdot \delta(Z \geq C)]}{\mathbb{E}[Z \cdot \delta(Z \leq C)]} \geq \frac{(1-C)(a+1)}{bC} \geq \frac{1}{C}.
$$

The last inequality holds when $a, b < 1$ and $C < 1/2$. The sum of the two is exactly $\mathbb{E}[Z]$, so when $C < 1/2$ we know $\mathbb{E}[Z \cdot \delta(Z \leq C)] < C \cdot \mathbb{E}[Z]$.

Next we bound the probability $\Pr[Z \geq C]$. Note that $\Pr[Z \geq 1/2] \leq 2\mathbb{E}[Z] = \frac{2a}{a+b}$ by Markov's inequality. Now we show $\Pr[Z \in [C, 1/2]]$ is not much larger than $\Pr[Z \geq 1/2]$ by bounding the integrals.

$$
A = \int_{1/2}^1 x^{a-1}(1-x)^{b-1} dx \geq \int_{1/2}^1 (1-x)^{b-1} dx = (1/2)^b / b.
$$

$$
\begin{aligned}
B = \int_C^{1/2} x^{a-1}(1-x)^{b-1} &\leq (1/2)^{b-1} \int_C^{1/2} x^{a-1} dx \\
&\leq (1/2)^{b-1} \frac{0.5^a - C^a}{a} \\
&\leq (1/2)^{b-1} \frac{1-(1-a\log 1/C)}{a} \\
&= (1/2)^{b-1} \log(1/C).
\end{aligned}
$$

The last inequality uses the fact that $e^x \geq 1 + x$ for all x. Now

$$
\Pr[Z \geq C] = (1 + \frac{B}{A}) \Pr[Z \geq 1/2] \leq (1 + 2b\log(1/C))\frac{2a}{a+b} \leq 8\log(1/C) \cdot \frac{a}{a+b}
$$

and we have the result.

Case 2: When $b \geq 1$, we have an alternative bound. We use the fact that if $X \sim \Gamma(a,1)$ and $Y \sim \Gamma(b,1)$ then $Z \sim X/(X+Y)$. Since Y is distributed as $\Gamma(b,1)$, its PDF is $\frac{1}{\Gamma(b)} x^{b-1} e^{-x}$. This is proportional to the PDF of $\Gamma(1)$ (e^{-x}) multiplied by a increasing function x^{b-1}.

Therefore we know $\Pr[Y \geq t] \geq \Pr_{Y' \sim \Gamma(1)}[Y' \geq t] = e^{-t}$.

Now we use this bound to compute the probability that $Z \leq 1/R$ for all $R \geq 1$.

This is equivalent to

$$
\begin{aligned}
\Pr[\frac{X}{X+Y} \leq \frac{1}{R}] &= \int_0^\infty Pr[X = x] Pr[Y \geq (R-1)X] dx \\
&\geq \int_0^\infty \frac{1}{\Gamma(a)} x^{a-1} e^{-Rx} dx \\
&= R^{-a} \int_0^\infty \frac{1}{\Gamma(a)} y^{a-1} e^{-y} dy \\
&= R^{-a}
\end{aligned}
$$

In particular, $\Pr[Z \leq C] \geq C^a$, which means $\Pr[Z \geq C] \leq 1 - C^a \leq a\log(1/C)$.

For $\mathbb{E}[Z\delta(Z < C)]$, the proof is similar as before:

$$P = \mathbb{E}[Z\delta(Z < C)] = \int_0^C \frac{1}{B(a,b)} x^a (1-x)^b dx \leq \frac{C^{a+1}}{B(a,b)(a+1)}$$

$$Q = \mathbb{E}[Z\delta(Z \geq C)] = \int_C^1 \frac{1}{B(a,b)} x^a (1-x)^b dx \geq \frac{C^a (1-C)^{b+1}}{B(a,b)(b+1)}$$

Now $\mathbb{E}[Z\delta(Z \leq C)] \leq \frac{P}{Q}\mathbb{E}[Z] \leq 6aC$ when $C < 1/(b+1)$. □

Properties of Gamma and Dirichlet Distributions

Recall Gamma distribution $\Gamma(\alpha, \beta)$ is a distribution on nonnegative real values with density function $\frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} e^{-\beta x}$.

Proposition 8.21 (Dirichlet and Gamma distributions)**.** *The following facts are known for Dirichlet distribution and Gamma distribution.*

1. *Let $Y_i \sim \Gamma(\alpha_i, 1)$ be independent random variables, then the vector $(Y_1, Y_2, ..., Y_k)/\sum_{i=1}^k Y_k$ is distributed as $Dir(\alpha)$.*
2. *The Γ function satisfies Euler's reflection formula: $\Gamma(1-z)\Gamma(z) \leq \pi/\sin \pi_z$.*
3. *The $\Gamma(z) \geq 1$ when $0 < z < 1$.*
4. *There exists a universal constant C such that $\Gamma(z) \leq C/z$ when $0 < z < 1$.*
5. *For $Y \sim \Gamma(\alpha, 1)$ and $t > 0$ and $\alpha \in (0,1)$, we have*

$$\frac{\alpha}{4C} t^{\alpha-1} e^{-t} \leq \Pr[Y \geq t] \leq t^{\alpha-1} e^{-t}, \tag{8.36}$$

and for any $\eta, c > 1$, we have

$$\mathbb{P}[Y > \eta t | Y \geq t] \geq (c\eta)^{\alpha-1} e^{-(\eta-1)t}. \tag{8.37}$$

Proof: The bounds in (8.36) is derived using the fact that $1 \leq \Gamma(\alpha) \leq C/\alpha$ when $\alpha \in (0,1)$ and

$$\int_t^\infty \frac{1}{\Gamma(\alpha_i)} x^{\alpha_i - 1} e^{-x} dx \leq \frac{1}{\Gamma(\alpha_i)} \int_t^\infty t^{\alpha_i - 1} e^{-x} dx \leq t^{\alpha_i - 1} e^{-t},$$

and

$$\int_t^\infty \frac{1}{\Gamma(\alpha_i)} x^{\alpha_i - 1} e^{-x} dx \geq \frac{1}{\Gamma(\alpha_i)} \int_t^{2t} x^{\alpha_i - 1} e^{-x} dx \geq \alpha_i/C \int_t^{2t} (2t)^{\alpha_i - 1} e^{-x} dx \geq \frac{\alpha_i}{4C} t^{\alpha_i - 1} e^{-t}.$$

□

Appendix A

Matrices and Tensors

A.1 Matrix Notations

A.1.1 Basic Notations, Rows and Columns

A matrix is a rectangular array of numbers. Throughout this thesis we only consider real matrices.

Most notations in this paper are commonly used. We used bold letters for vectors like $\boldsymbol{v}$. All the vectors are by default column vectors. The entries of a vector is usually denoted as $\boldsymbol{u}_i$, but sometimes we also use $\boldsymbol{u}[i]$ (especially when the vector itself has many superscripts or subscripts). The all 1's and all 0's vectors of length m are denoted as $\mathbf{0}_m$ and $\mathbf{1}_m$, but the length is usually omitted as it is clear from the context. The basis vectors $\boldsymbol{e}_i$ are vectors which has only one entry $e_i[i] = 1$ and all other entries are 0, again the length of the vector will be clear from context.

Inner products of vectors is denoted as $\boldsymbol{u} \cdot \boldsymbol{v}$, although sometimes equivalent matrix notation $\boldsymbol{u}^\top \boldsymbol{v}$ is also used (mainly in manipulations that also involves matrices).

We use I to denote the identity matrix, $M^\top$ to denote the transpose of a matrix, and M^{-1} to denote the inverse of a matrix. A matrix is symmetric if $M = M^\top$. It is orthonormal if $MM^\top = M^\top M = I$. The entries of a matrix M is usually denoted as $M_{i,j}$, however sometimes we also use $M[i,j]$. To represent a diagonal matrix, we use $\mathrm{Diag}(\boldsymbol{u})$ or $\mathrm{Diag}(\boldsymbol{u}_i)$ where $\boldsymbol{u}$ is the vector of the diagonal entries.

We use M_i to denote the i-th column of a matrix, and M^j to denote the j-th row of a matrix. To denote a submatrix, let M be an $n \times m$ matrix, $A \subset [n]$ and $B \subset [m]$, then $M_{A,B}$ or $M[A,B]$ denote the submatrix of M with indices in A and B. When either A or B is a singleton we also abuse notation to use $M_{i,B}$ instead of $M_{\{i\},B}$.

A.1.2 Norms and Eigenvalues

We again use commonly accepted definitions for vector and matrix norms. The ℓ_2 norm of a vector $\|\boldsymbol{u}\| = \sqrt{\sum_{i=1}^k \boldsymbol{u}_i^2}$ (notice that the subscript 2 is omitted). The ℓ_1 norm is $|\boldsymbol{u}|_1 = \sum_{i=1}^k |\boldsymbol{u}_i|$. Infinity norm is also sometimes used but we will simply write $\max_k |\boldsymbol{u}_k|$.

For a matrix M, if $M\boldsymbol{v} = \lambda\boldsymbol{v}$, then λ is an eigenvalue and $\boldsymbol{v}$ is an eigenvector of the matrix. We will only talk about eigenvalues and eigenvectors of real symmetric matrices, so the eigenvalues and eigenvectors are themselves real.

The singular values of a matrix M are the square roots of the eigenvalues of $M^\top M$, usually denoted by $\sigma_i(M)$'s (and usually we assume $\sigma_1 \ge \sigma_2 \ge \cdots \ge \sigma_m$).

The spectral norm of a matrix is the same as the operator norm, and has several equivalent definitions:

$$\|M\| = \sigma_1(M) = \max_{\|\boldsymbol{u}\| \le 1} \|M\boldsymbol{u}\| = \max_{\|\boldsymbol{u}\| \le 1} \boldsymbol{u}^\top M\boldsymbol{u}.$$

Here the last equality only holds when M is symmetric.

The Frobenius norm of a matrix is defined as follows

$$\|M\|_F = \sqrt{\sum_{i,j} M_{i,j}^2} = \sqrt{\sum_i \sigma_i(M)^2}.$$

There are several standard facts about these norms

1. $\|A + B\| \leq \|A\| + \|B\|$.

2. $\|AB\| \leq \|A\| \|B\|$

3. $\|AB\|_F \leq \|A\| \|B\|_F$.

4. For rank r matrices $\|A\| \leq \|A\|_F \leq \sqrt{r} \|A\|$.

We call a symmetric matrix is positive semidefinite (PSD) if all its eigenvalues are non-negative. The smallest singular value of a matrix is usually denoted as $\sigma_{min}(M)$.

The condition number of a matrix is equal to the ratio between its largest and smallest singular values, $\kappa(M) = \|M\| / \sigma_{min}(M)$. When the condition number is small we say the matrix is well conditioned.

A.1.3 Singular Value Decomposition and Eigenspace Perturbation

Every matrix can be decomposed into the form $M = UDV^\top$, where $U^\top U = I$ and $VV^\top = I$, and D is a diagonal matrix whose values are the singular values of M. The columns of U and V are called left and right singular vectors. When M is symmetric PSD we have $U = V$.

Often we use the top-k singular values to approximate a matrix M, $M_{k-svd} = \sum_{i=1}^{k} U_i \sigma_i(M) V_i^\top$. This is the optimal approximation using rank k matrices in terms of both spectral norm and Frobenius norm.

The Moore-Penrose pseudo-inverse of an $n \times m$ $(n > m)$ full rank matrix M is $M^\dagger = VD^{-1}U^\top$, which always satisfies $M^\dagger M = I$ (but keep in mind that there are more than one matrices that satisfy this equation). When a matrix M is symmetric PSD that has singular value decomposition $M = UDU^\top$, the p-th power $p \in R$ of M is defined as $M^p = U \operatorname{Diag}(D_{i,i}^p) U^\top$.

There are several theorems controlling the eigenvalues and eigenvectors of matrices when they are perturbed.

Weyl's Theorem Weyl's Theorem bounds the perturbation in the singular values when a noise with bounded spectral norm is added to the matrix.

Theorem A.1 (Weyl's theorem; Theorem 4.11, p.204 in [147].)**.** *Let $A, E \in \mathbb{R}^{m\times n}$ with $m > n$, then*

$$\max_{i\in[n]} |\sigma_i(A+E) - \sigma_i(A)| \leq \|E\| \,.$$

Wedin's Theorem: Wedin's Theorem bounds the perturbation of subspaces spanned by singular vectors when a noise with bounded spectral norm is added to the matrix. When the spectral norm of the noise is much smaller than the top-k singular values, the space spanned by the top-k singular vectors does not change much.

Theorem A.2 (Wedin's theorem; Theorem 4.4, p. 262 in [147].)**.** *Let $A, E \in \mathbb{R}^{m\times n}$ with $m \geq n$ be given. Let A have the singular value decomposition*

$$\begin{bmatrix} U_1^\top \\ U_2^\top \\ U_3^\top \end{bmatrix} A \begin{bmatrix} V_1 & V_2 \end{bmatrix} = \begin{bmatrix} \Sigma_1 & 0 \\ 0 & \Sigma_2 \\ 0 & 0 \end{bmatrix}.$$

Let $\tilde{A} := A + E$, with analogous singular value decomposition $(\tilde{U_1}, \tilde{U_2}, \tilde{U_3}, \tilde{\Sigma_1}, \tilde{\Sigma_2}, \tilde{V_1}\tilde{V_2})$. Let Φ be the matrix of canonical angles between $\text{range}(U_1)$ *and* $\text{range}(\tilde{U_1})$*, and Θ be the matrix of canonical angles between* $\text{range}(V_1)$ *and* $\text{range}(\tilde{V_1})$*. If there exists $\delta, \alpha > 0$ such that* $\min_i \sigma_i(\tilde{\Sigma}_1) \geq \alpha + \delta$ *and* $\max_i \sigma_i(\Sigma_2) \leq \alpha$*, then*

$$\max\{\|\sin\Phi\|_2, \|\sin\Theta\|_2\} \leq \frac{\|E\|_2}{\delta}.$$

A.2 Tensor Notations

A real *p-th order tensor* $T \in \bigotimes_{i=1}^{p}$ is a member of the tensor product of Euclidean spaces $\mathbb{R}^{n_i}$, $i \in [p]$. We generally restrict to the case where $n_1 = n_2 = \cdots = n_p = n$, and simply write $T \in \bigotimes^p \mathbb{R}^n$. For a vector $\boldsymbol{v} \in \mathbb{R}^n$, we use $\boldsymbol{v}^{\otimes p} := \boldsymbol{v} \otimes \boldsymbol{v} \otimes \cdots \otimes \boldsymbol{v} \in \bigotimes^p \mathbb{R}^n$ to denote its p-th tensor power. As is the case for vectors (where $p = 1$) and matrices (where $p = 2$), we may identify a p-th order tensor with the p-way array of real numbers $[T_{i_1,i_2,\dots,i_p} \colon i_1, i_2, \dots, i_p \in [n]]$, where $T_{i_1,i_2,\dots,i_p}$ is the $(i_1, i_2, \dots, i_p)$-th coordinate of A (with respect to a canonical basis).

An useful way to work with tensors is to view the tensor as a multilinear form. We use the following notation[1]:

Notation (Multilinear Form)**.** We can consider T to be a multilinear map in the following sense: for a set of matrices $\{V_i \in \mathbb{R}^{n \times m_i} : i \in [p]\}$, the $(i_1, i_2, \dots, i_p)$-th entry in the p-way array representation of $T(V_1, V_2, \dots, V_p) \in \mathbb{R}^{m_1 \times m_2 \times \cdots \times m_p}$ is

$$[T(V_1, V_2, \dots, V_p)]_{i_1,i_2,\dots,i_p} := \sum_{j_1,j_2,\dots,j_p \in [n]} T_{j_1,j_2,\dots,j_p} \, [V_1]_{j_1,i_1} \, [V_2]_{j_2,i_2} \cdots [V_p]_{j_p,i_p}. \tag{A.1}$$

Note that if T is a matrix ($p = 2$), then

$$T(V_1, V_2) = V_1^\top T V_2.$$

Similarly, for a matrix M and vector $\boldsymbol{v} \in \mathbb{R}^n$, we can express $M\boldsymbol{v}$ as

$$M(I, \boldsymbol{v}) = M\boldsymbol{v} \in \mathbb{R}^n,$$

[1] the notation first appeared in [113]

where I is the $n \times n$ identity matrix. As a final example of this notation, observe

$$T(\boldsymbol{e_{i_1}}, \boldsymbol{e_{i_2}}, \ldots, \boldsymbol{e_{i_p}}) = T_{i_1,i_2,\ldots,i_p},$$

where $\{\boldsymbol{e_1}, \boldsymbol{e_2}, \ldots, \boldsymbol{e_n}\}$ is the canonical basis for $\mathbb{R}^n$.

Most tensors $T \in \bigotimes^p \mathbb{R}^n$ considered in this thesis will be *symmetric* (sometimes called *supersymmetric*), which means that their p-way array representations are invariant to permutations of the array indices: *i.e.*, for all indices $i_1, i_2, \ldots, i_p \in [n]$, $T_{i_1,i_2,\ldots,i_p} = T_{i_{\pi(1)},i_{\pi(2)},\ldots,i_{\pi(p)}}$ for any permutation π on $[p]$. It can be checked that this reduces to the usual definition of a symmetric matrix for $p = 2$.

Definition A.3 (Tensor Rank)**.** The *rank* of a p-th order tensor $T \in \bigotimes^p \mathbb{R}^n$ is the smallest non-negative integer k such that $T = \sum_{j=1}^k \boldsymbol{u}_{1,j} \otimes \boldsymbol{u}_{2,j} \otimes \cdots \otimes \boldsymbol{u}_{p,j}$ for some $\boldsymbol{u}_{i,j} \in \mathbb{R}^n, i \in [p], j \in [k]$, and the *symmetric rank* of a symmetric p-th order tensor T is the smallest non-negative integer k such that $T = \sum_{j=1}^k \boldsymbol{u}_j^{\otimes p}$ for some $\boldsymbol{u}_j \in \mathbb{R}^n, j \in [k]$.

The notion of rank readily reduces to the usual definition of matrix rank when $p = 2$, as revealed by the singular value decomposition. Similarly, for symmetric matrices, the symmetric rank is equivalent to the matrix rank as given by the spectral theorem.

The notion of tensor (symmetric) rank is considerably more delicate than matrix (symmetric) rank. For instance, it is not clear *a priori* that the symmetric rank of a tensor should even be finite [43]. In addition, removal of the best rank-1 approximation of a (general) tensor may increase the tensor rank of the residual [145].

Definition A.4 (Spectral Norm of Tensor)**.** The spectral norm of a tensor T is defined by viewing it as a linear operator.

$$\|T\| = \sup_{\|\boldsymbol{v}_i\| \le 1, i \in [p-1]} \|T(\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_{p-1}, I)\| = \sup_{\|\boldsymbol{v}_i\| \le 1, i \in [p]} \|T(\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_p)\|.$$

For symmetric tensors it is easy to show that the maximum is always achieved when all the vectors are the same, so for symmetric T, $\|T\| = \sup_{\|\boldsymbol{v}\| \leq 1} T(\boldsymbol{v}, \boldsymbol{v}, ..., \boldsymbol{v})$.

Appendix B

Concentration Bounds

B.1 Concentration Bounds for Single Variables or Functions

Given a random variable X, concentration bounds for X tries to bound the probability that X is far from its expectation $\mathbb{E}[X]$. All the random variables in this thesis will have bounded moments.

The most basic inequalities are the Markov's inequality and Chebyshev's inequality:

Theorem B.1 (Markov's Inequality)**.** *If X is a nonnegative random variable, then for any $k \geq 1$*

$$\Pr[X \geq k\mathbb{E}[X]] \leq 1/k.$$

Theorem B.2 (Chebyshev's Inequality)**.** *For any random variable X and for any $k \geq 1$*

$$\Pr[|X - \mathbb{E}[X]| > k\sqrt{\operatorname{Var} X}] \leq 1/k^2.$$

Much stronger guarantees can be obtained for sum of independent random variables, including Chernoff bounds and Bernstein's inequality.

Theorem B.3 (Chernoff Bounds). *Let $X_1, ..., X_n$ be independent random variables taking values in $\{0, 1\}$, $\Pr[X_i = 1] = p_i$. Then if we let $X = \sum_{i=1}^n X_i$ and $\mu = \mathbb{E}[X] = \sum_{i=1}^n p_i$, for any $\delta > 0$,*

$$\Pr[X > (1+\delta)\mu] \le \left(\frac{e^\delta}{(1+\delta)^{(1+\delta)}}\right)^\mu,$$
$$\Pr[X < (1-\delta)\mu] \le \left(\frac{e^{-\delta}}{(1-\delta)^{(1-\delta)}}\right)^\mu.$$

Theorem B.4 (Bernstein's Inequality). *Let $X_1, ..., X_n$ be independent random variables with mean 0. Suppose $|X_i| \le M$ almost surely for all i, then for all $t > 0$,*

$$\Pr[|\sum_{i=1}^n X_i| > t] \le 2\exp\left(-\frac{t^2/2}{\sum_{i=1}^n \mathrm{Var}[X_i] + Mt/3}\right).$$

Finally, when the random variables are not independent, or when we are interested in other functions of random variables, the common tool is martingales and Azuma's inequality. However here we just state the bounded differences/McDiarmid's inequality as it is the simplest way to obtain polynomial bounds for the quantities of interest in Chapter 7.

Theorem B.5 (McDiarmid's Inequality). *Consider independent random variables $X_1, ..., X_n \in \mathcal{X}$, and a real valued function $f(X_1, X_2, ..., X_n)$. If for all index $i \in [n]$, and for all $x_1, x_2, ..., x_n, x_i' \in \mathcal{X}$, the function satisfies*

$$|f(x_1, x_2, ..., x_{i-1}, x_i, x_{i+1}, ..., x_n) - f(x_1, x_2, ..., x_{i-1}, x_i', x_{i+1}, ..., x_n)| \le c_i,$$

then we have for all $t > 0$

$$\Pr[|f(X_1, ..., X_n) - E[f(X_1, ..., X_n)]| \ge t] \le 2\exp\left(\frac{2t^2}{\sum_{i=1}^n c_i^2}\right).$$

For example, the simplest way to show the sum of independent matrices/tensors have concentrated spectral norm is to let f be the spectral norm of the sum. However, this usually won't give the tightest bound. In order to better understand the concentration behavior we need more precise bounds in the next Section.

B.2 Concentration Bounds for Vectors and Matrices

Vectors and matrices also have concentration bounds that behave similarly to the Bernstein's inequality.

Proposition B.6 (Vector Bernstein Inequality). *Let $z = (z_1, z_2, ..., z_n) \in \mathbb{R}^n$ be a random vector with independent entries, $\mathbb{E}[z_i] = 0$, $\mathbb{E}[z_i^2] = \sigma_i^2$, and $\Pr[|z_i| \leq 1] = 1$. Let $A = [a_1|a_2|\cdots|a_n] \in \mathbb{R}^{m\times n}$ be a matrix, then*

$$\Pr[\|Az\| \leq (1+\sqrt{8t})\sqrt{\sum_{i=1}^{n} \|a_i\|^2 \sigma_i^2} + (4/3)\max_{i\in[n]} \|a_i\| \, t] \geq 1 - e^{-t}.$$

Proposition B.7 (Matrix Bernstein Inequality). *Suppose $Z = \sum_j W_j$ where*

1. *W_j are independent random matrices with dimension $d_1 \times d_2$,*
2. *$\mathbb{E}[W_j] = 0$ for all j,*
3. *$\|W_j\| \leq R$ almost surely.*

Let $d = d_1 + d_2$, and $\sigma^2 = \max\left\{\left\|\sum_j \mathbb{E}[W_j W_j^\top]\right\|, \left\|\sum_j \mathbb{E}[W_j^\top W_j]\right\|\right\}$, then we have

$$\Pr[\|Z\| \geq t] \leq d \cdot exp\left\{\frac{-t^2/2}{\sigma^2 + Rt/3}\right\}.$$

Bibliography

[1] M. Aharon. *Overcomplete Dictionaries for Sparse Representation of Signals.* PhD thesis, Technion - Israel Institute of Technology, 2006.

[2] M. Aharon, M. Elad, and A. Bruckstein. K-svd: An algorithm for designing overcomplete dictionaries for sparse representation. *Trans. Sig. Proc.*, 54(11):4311–4322, November 2006.

[3] A. Aho, J. Ullman, and M. Yannakakis. On notions of information transfer in vlsi circuits. In *Proceedings of the 15th symposium on Theory of Computing*, pages 133–139, 1983.

[4] Edoardo M. Airoldi, David M. Blei, Stephen E. Fienberg, and Eric P. Xing. Mixed membership stochastic blockmodels. *Journal of Machine Learning Research*, 9:1981–2014, June 2008.

[5] N. Alon and D. Kleitman. Piercing convex sets and the hadwigder debrunner (p, q)-problem. *Advances in Mathematics*, 96:103–112, 1992.

[6] N. Alon and S. Onn. Separable partitions. *Discrete Applied Math*, pages 39–51, 1999.

[7] A. Anandkumar, D. Hsu, F. Huang, and S.M. Kakade. Learning mixtures of tree graphical models. In *Advances in Neural Information Processing Systems 25*, 2012.

[8] A. Anandkumar, D. Hsu, and S.M. Kakade. A method of moments for mixture models and hidden markov models. In *Proc. of Conf. on Learning Theory*, June 2012.

[9] Anima Anandkumar, Dean P. Foster, Daniel Hsu, Sham M. Kakade, and Yi-Kai Liu. A spectral algorithm for latent Dirichlet allocation. In *Advances in Neural Information Processing Systems 25*, 2012.

[10] Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. Multi-task feature learning. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 41–48, Cambridge, MA, 2007. MIT Press.

[11] Sanjeev Arora, Rong Ge, Ravindran Kannan, and Ankur Moitra. Computing a nonnegative matrix factorization – provably. In *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19 - 22*, pages 145–162, 2012.

[12] Sanjeev Arora, Rong Ge, and Ankur Moitra. Learning topic models – going beyond svd. In *Proceedings of the 53rd Annual IEEE Symposium on Foundations of Computer Science*, pages 1–10, 2012.

[13] Sanjeev Arora, Rong Ge, and Ankur Moitra. New algorithms for learning incoherent and overcomplete dictionaries. *arXiv*, 1308.6273, 2013.

[14] Sanjeev Arora, Rong Ge, Sushant Sachdeva, and Grant Schoenebeck. Finding overlapping communities in social networks: toward a rigorous approach. In *Proceedings of the 13th ACM Conference on Electronic Commerce*, EC '12, pages 37–54, New York, NY, USA, 2012. ACM.

[15] T. Austin. On exchangeable random variables and the statistics of large graphs and hypergraphs. *Probab. Survey*, 5:80–145, 2008.

[16] Maria-Florina Balcan, Avrim Blum, and Anupam Gupta. Clustering under approximation stability. *J. ACM*, 60(2):8:1–8:34, May 2013.

[17] Maria-Florina Balcan, Christian Borgs, Mark Braverman, Jennifer T. Chayes, and Shang-Hua Teng. Finding endogenously formed communities. In *Proceedings of the 24th annual ACM-SIAM symposium on Discrete algorithms*, volume abs/1201.4899, 2013.

[18] S. Basu, R. Pollack, and M. Roy. On the combinatorial and algebraic complexity of quantifier elimination. *Journal of the ACM*, pages 1002–1045, 1996.

[19] Yoshua Bengio. Learning deep architectures for ai. *Found. Trends Mach. Learn.*, 2(1):1–127, January 2009.

[20] Yoshua Bengio, Aaron C. Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(8):1798–1828, 2013.

[21] R. Berinde, A.C. Gilbert, P. Indyk, H. Karloff, and M.J. Strauss. Combining geometry and combinatorics: a unified approach to sparse signal recovery. In *46th Annual Allerton Conference on Communication, Control, and Computing*, pages 798–805, 2008.

[22] V. Bittorf, B. Recht, C. Re, and J. Tropp. Factoring nonnegative matrices with linear programs. In *NIPS*, pages 1223–1231, 2012.

[23] D. Blei. Personal communication.

[24] D. Blei. Introduction to probabilistic topic models. *Communications of the ACM*, pages 77–84, 2012.

[25] David M. Blei and John D. Lafferty. A correlated topic model of science. *AAS*, 1(1):17–35, 2007.

[26] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, March 2003.

[27] Avrim Blum and Joel Spencer. Coloring random and semi-random k-colorable graphs. *J. Algorithms*, 19(2):204–234, September 1995.

[28] L. Blum, F. Cucker, M. Shub, and S. Smale. *Complexity of Real Computations.* Springer Verlag, 1998.

[29] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, COLT '92, pages 144–152, New York, NY, USA, 1992. ACM.

[30] Petros T Boufounos and Richard G Baraniuk. 1-bit compressive sensing. In *Information Sciences and Systems, 2008. CISS 2008. 42nd Annual Conference on*, pages 16–21. IEEE, 2008.

[31] Spencer Charles Brubaker and Santosh Vempala. Isotropic pca and affine-invariant clustering. In *Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '08, pages 551–560, Washington, DC, USA, 2008. IEEE Computer Society.

[32] G. Buchsbaum and O. Bloch. Color categories revealed by non-negative matrix factorization of munsell color spectra. *Vision Research*, pages 559–563, 2002.

[33] Wray L. Buntine. Estimating likelihoods for topic models. In *Asian Conference on Machine Learning*, 2009.

[34] E. J. Candes and T. Tao. Decoding by linear programming. *IEEE Trans. Inf. Theor.*, 51(12):4203–4215, December 2005.

[35] Emmanuel J. Candès, Justin K. Romberg, and Terence Tao. Stable signal recovery from incomplete and inaccurate measurements. *Comm. Pure Appl. Math.*, 59(8):1207–1223, August 2006.

[36] Emmanuel J Candes and Terence Tao. Decoding by linear programming. *Information Theory, IEEE Transactions on*, 51(12):4203–4215, 2005.

[37] J.-F. Cardoso and Pierre Comon. Independent component analysis, a survey of some algebraic methods. In *IEEE International Symposium on Circuits and Systems*, pages 93–96, 1996.

[38] Joseph T. Chang. Full reconstruction of Markov models on evolutionary trees: Identifiability and consistency. *Mathematical Biosciences*, 137:51–73, 1996.

[39] Scott Shaobing Chen, David L. Donoho, and Michael A. Saunders. Atomic decomposition by basis pursuit. *SIAM Rev.*, 43(1):129–159, January 2001.

[40] Youngmin Cho and Lawrence Saul. Kernel methods for deep learning. In *Advances in Neural Information Processing Systems 22*, pages 342–350. 2009.

[41] J. Cohen and U. Rothblum. Nonnegative ranks, decompositions and factorizations of nonnegative matices. *Linear Algebra and its Applications*, pages 149–168, 1993.

[42] P. Comon. Independent component analysis, a new concept? *Signal Processing*, 36(3):287–314, 1994.

[43] P. Comon, G. Golub, L.-H. Lim, and B. Mourrain. Symmetric tensors and symmetric tensor rank. *SIAM Journal on Matrix Analysis Appl.*, 30(3):1254–1279, 2008.

[44] P. Comon and C. Jutten. *Handbook of Blind Source Separation: Independent Component Analysis and Applications.* Academic Press. Elsevier, 2010.

[45] S. Currarini, M.O. Jackson, and P. Pin. An economic model of friendship: Homophily, minorities, and segregation. *Econometrica*, 77(4):1003–1045, 2009.

[46] Sanjoy Dasgupta. Learning mixtures of gaussians. In *Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science*, pages 634–644, 1999.

[47] G. Davis, S. Mallat, and M. Avellaneda. Adaptive greedy approximations. *Constructive Approximation*, 13(1):57–98, March 1997.

[48] Geoff Davis, Stephane Mallat, and Marco Avellaneda. Adaptive greedy approximations. *Constructive approximation*, 13(1):57–98, 1997.

[49] L. De Lathauwer, J. Castaing, and J.-F. Cardoso. Fourth-order cumulant-based blind identification of underdetermined mixtures. *Signal Processing, IEEE Transactions on*, 55(6):2965–2973, 2007.

[50] Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle. A multilinear singular value decomposition. *SIAM journal on Matrix Analysis and Applications*, 21(4):1253–1278, 2000.

[51] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum-likelihood from incomplete data via the EM algorithm. *J. Royal Statist. Soc. Ser. B*, 39:1–38, 1977.

[52] Weicong Ding, Mohammad H. Rohban, Prakash Ishwar, and Venkatesh Saligrama. Topic discovery through data dependent and random projections. In *ICMLProceedings of the 30th International Conference on Machine Learning (ICML-13)*, 2013.

[53] D. L. Donoho and X. Huo. Uncertainty principles and ideal atomic decomposition. *IEEE Trans. Inf. Theor.*, 47(7):2845–2862, September 2006.

[54] David Donoho and Victoria Stodden. When does non-negative matrix factorization give a correct decomposition into parts? In *NIPS*, 2003.

[55] David L Donoho. Compressed sensing. *Information Theory, IEEE Transactions on*, 52(4):1289–1306, 2006.

[56] David L. Donoho and Philip B. Stark. Uncertainty principles and signal recovery. *SIAM J. Appl. Math.*, 49(3):906–931, June 1989.

[57] M. Elad and M. Aharon. Image denoising via sparse and redundant representations over learned dictionaries. *Image Processing, IEEE Transactions on*, 15(12):3736–3745, December 2006.

[58] Michael Elad. *Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing*. Springer Publishing Company, Incorporated, 1st edition, 2010.

[59] K. Engan, S. O. Aase, and J. Hakon Husoy. Method of optimal directions for frame design. In *Proceedings of the Acoustics, Speech, and Signal Processing, 1999. on 1999 IEEE International Conference - Volume 05*, ICASSP '99, pages 2443–2446, Washington, DC, USA, 1999. IEEE Computer Society.

[60] Uriel Feige and Joe Kilian. Heuristics for semirandom graph problems. *Journal of Computing and System Sciences*, 63:639–671, 2001.

[61] Uriel Feige and Robert Krauthgamer. Finding and certifying a large hidden clique in a semi-random graph. Technical report, Jerusalem, Israel, Israel, 1999.

[62] S.E. Fienberg, M.M. Meyer, and S.S. Wasserman. Statistical analysis of multiple sociometric relations. *Journal of the american Statistical association*, 80(389):51–67, 1985.

[63] Alan M. Frieze, Mark Jerrum, and Ravindran Kannan. Learning linear transformations. In *Proceedings of the 37th Annual IEEE Symposium on Foundations of Computer Science*, 1996.

[64] Anna C. Gilbert, S. Muthukrishnan, and Martin J. Strauss. Approximation of functions over redundant dictionaries using coherence. In *Proceedings of the 14th annual ACM-SIAM symposium on Discrete algorithms*, SODA '03, pages 243–252, Philadelphia, PA, USA, 2003. Society for Industrial and Applied Mathematics.

[65] N. Gillis and S. Vavasis. Fast and robust recursive algorithms for separable nonnegative matrix factorization. *arXiv*, 1208.1237, 2012.

[66] Nicolas Gillis. Robustness analysis of hottopixx, a linear programming model for factoring nonnegative matrices. *SIAM Journal on Matrix Analysis and Applications*, 34(3):1189–1212, 2013.

[67] Nicolas Gillis and Robert Luce. Robust near-separable nonnegative matrix factorization using linear optimization. *arXiv*, 1302.4385, 2013.

[68] G. H. Golub and C. F. van Loan. *Matrix Computations.* Johns Hopkins University Press, 1996.

[69] C. Gomez, H. Le Borgne, P. Allemand, C. Delacourt, and P. Ledru. N-findr method versus independent component analysis for lithological identification in hyperspectral imagery. *Int. J. Remote Sens.*, 28(23), January 2007.

[70] I. J. Goodfellow, A. Courville, and Y. Bengio. Large-scale feature learning with spike-and-slab sparse coding. In *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, 2012.

[71] P. Gopalan, D. Mimno, S. Gerrish, M. Freedman, and D. Blei. Scalable inference of overlapping communities. In *Advances in Neural Information Processing Systems 25*, pages 2258–2266, 2012.

[72] N. Goyal, S. Vempala, and Y. Xiao. Fourier pca. *arXiv*, 1306.5825, 2013.

[73] D. Yu. Grigor'ev and N.N. Vorobjov Jr. Solving systems of polynomial inequalities in subexponential time,. *Journal of Symbolic Computation*, 5-1-2:37–64, 1988.

[74] David Gross. Recovering low-rank matrices from few coefficients in any basis. *Information Theory, IEEE Transactions on*, 57(3):1548–1566, 2011.

[75] V. Guruswami and P. Indyk. Expander-based constructions of efficiently decodable codes. In *Proceedings of the 42nd IEEE symposium on Foundations of Computer Science*, FOCS '01, pages 658–, Washington, DC, USA, 2001. IEEE Computer Society.

[76] D.L. Hanson and F.T. Wright. A bound on tail probabilities for quadratic forms in independent random variables. *The Annals of Mathematical Statistics*, 42(3):1079–1083, 1971.

[77] E. Harding. The number of partitions of a set of n points in k dimensions induced by hyperplanes. *Edinburgh Math. Society*, pages 285–289, 1967.

[78] Richard A Harshman. Foundations of the parafac procedure: models and conditions for an" explanatory" multimodal factor analysis. 1970.

[79] L. Henry. Schémas de nuptialité: déséquilibre des sexes et célibat. *Population*, 24:457–486, 1969.

[80] Geoffrey E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Comput.*, 14(8):1771–1800, August 2002.

[81] Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural Comput.*, 18(7):1527–1554, July 2006.

[82] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[83] Thomas Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '99, pages 50–57, New York, NY, USA, 1999. ACM.

[84] P.W. Holland, K.B. Laskey, and S. Leinhardt. Stochastic blockmodels: first steps. *Social networks*, 5(2):109–137, 1983.

[85] Roger A. Horn and Charles R. Johnson, editors. *Matrix analysis*. Cambridge University Press, New York, NY, USA, 1986.

[86] D. Hsu, S. Kakade, and P. Liang. Identifiability and unmixing of latent parse trees. In *Advances in Neural Information Processing Systems 25*, 2012.

[87] Daniel Hsu and Sham M. Kakade. Learning mixtures of spherical gaussians: moment methods and spectral decompositions. In *Proceedings of the 4th conference on Innovations in Theoretical Computer Science*, ITCS '13, pages 11–20, New York, NY, USA, 2013. ACM.

[88] Daniel Hsu, Sham M. Kakade, and Tong Zhang. A spectral algorithm for learning hidden Markov models. *Journal of Computer and System Sciences*, 78(5):1460–1480, 2012.

[89] Furong Huang, Mohammad Umar Hakeem, Prateek Verma, Animashree Anandkumar, et al. Fast detection of overlapping communities via online tensor methods on gpus. *arXiv*, 1309.0787, 2013.

[90] F. Hwang and U. Rothblum. On the number of separable partitions. *Journal of Combinatorial Optimization*, pages 423–433, 2011.

[91] A. Hyvärinen and E. Oja. Independent component analysis: algorithms and applications. *Neural Networks*, 13(4–5):411–430, 2000.

[92] Russel Impagliazzo and Ramamohan Paturi. On the complexity of k-sat. *J. Comput. Syst. Sci.*, 62(2):367–375, March 2001.

[93] Jeffrey C Jackson, Adam R Klivans, and Rocco A Servedio. Learnability beyond ac^0. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 776–784. ACM, 2002.

[94] Koray Kavukcuoglu, Marc'Aurelio Ranzato, and Yann Lecun. Fast inference in sparse coding algorithms with applications to object recognition. Technical report, Technical report, Computational and Biological Learning Lab, Courant Institute, NYU, 2008.

[95] Leonid Khachiyan. On the complexity of approximating extremal determinants in matrices. *Journal of Complexity*, 11(1):138–153, 1995.

[96] J. Kleinberg and M. Sandler. Using mixture models for collaborative filtering. *JCSS*, pages 49–69, 2008.

[97] Adam R Klivans and Alexander A Sherstov. Cryptographic hardness for learning intersections of halfspaces. *Journal of Computer and System Sciences*, 75(1):2–12, 2009.

[98] E. Kofidis and P. A. Regalia. On the best rank-1 approximation of higher-order super-symmetric tensors. *SIAM Journal on Matrix Analysis and Applications*, 23(3):863–884, 2002.

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

[100] T. G. Kolda and J. R. Mayo. Shifted power method for computing tensor eigenpairs. *SIAM Journal on Matrix Analysis and Applications*, 32(4):1095–1124, October 2011.

[101] Tamara G Kolda. Orthogonal tensor decompositions. *SIAM Journal on Matrix Analysis and Applications*, 23(1):243–255, 2001.

[102] Alexandra Kolla, Konstantin Makarychev, and Yury Makarychev. How to play unique games against a semi-random adversary. In *Proceedings of the IEEE 52nd Annual Symposium on Foundations of Computer Science*, FOCS '11, pages 443–452, Washington, DC, USA, 2011. IEEE Computer Society.

[103] Alex Krizhevsky, Ilya Sutskever, and Geoff Hinton. Imagenet classification with deep convolutional neural networks. In P. Bartlett, F.C.N. Pereira, C.J.C. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1106–1114. 2012.

[104] A. Kumar, V. Sindhwani, and P. Kambadur. Fast conical hull algorithms for near-separable non-negative matrix factorization. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, 2013.

[105] L. De Lathauwer, B. De Moor, and J. Vandewalle. On the best rank-1 and rank-$(R_1, R_2, ..., R_n)$ approximation and applications of higher-order tensors. *SIAM J. Matrix Anal. Appl.*, 21(4):1324–1342, 2000.

[106] Lap Chi Lau. Bipartite roots of graphs. In *In Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 952–961, 2004.

[107] W. Lawton and E. Sylvestre. Self modeling curve resolution. *Technometrics*, pages 617–633, 1971.

[108] P.F. Lazarsfeld, R.K. Merton, et al. Friendship as a social process: A substantive and methodological analysis. *Freedom and control in modern society*, 18(1):18–66, 1954.

[109] D. Lee and H. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, pages 788–791, 1999.

[110] Daniel D. Lee and H. Sebastian Seung. Algorithms for non-negative matrix factorization. In *NIPS*, pages 556–562, 2000.

[111] Michael S. Lewicki and Terrence J. Sejnowski. Learning overcomplete representations. *Neural Comput.*, 12(2):337–365, February 2000.

[112] Wei Li and Andrew McCallum. Pachinko allocation: Dag-structured mixture models of topic correlations. In *Proceedings of the 23rd international conference on Machine learning*, ICML '06, pages 577–584, New York, NY, USA, 2006. ACM.

[113] Lek-Heng Lim. Singular values and eigenvalues of tensors: a variational approach. In *Computational Advances in Multi-Sensor Adaptive Processing, 2005 1st IEEE International Workshop on*, pages 129–132. IEEE, 2005.

[114] Zhouchen Lin, Minming Chen, and Yi Ma. The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices. *Mathematical Programming*, 2010.

[115] Roi Livni, Shai Shalev-Shwartz, and Ohad Shamir. A provably efficient algorithm for training deep networks. *arXiv*, 1304.7045, 2013.

[116] L. Lovász and M. Saks. Communication complexity and combinatorial lattice theory. *JCSS*, pages 322–349, 1993.

[117] Julien Mairal, Marius Leordeanu, Francis Bach, Martial Hebert, and Jean Ponce. Discriminative sparse image models for class-specific edge detection and image interpretation. In *Proceedings of the 10th European Conference on Computer Vision: Part III*, ECCV '08, pages 43–56, Berlin, Heidelberg, 2008. Springer-Verlag.

[118] Konstantin Makarychev, Yury Makarychev, and Aravindan Vijayaraghavan. Approximation algorithms for semi-random partitioning problems. In *Proceedings of the 44th symposium on Theory of Computing*, STOC '12, pages 367–384, New York, NY, USA, 2012. ACM.

[119] S.G. Mallat and Zhifeng Zhang. Matching pursuits with time-frequency dictionaries. *Trans. Sig. Proc.*, 41(12):3397–3415, December 1993.

[120] Stphane Mallat. *A Wavelet Tour of Signal Processing, Third Edition: The Sparse Way.* Academic Press, 3rd edition, 2008.

[121] Jiri Matousek. *Lectures on Discrete Geometry.* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2002.

[122] Andrew Kachites McCallum. Mallet: A machine learning for language toolkit, 2002.

[123] Peter McCullagh. *Tensor methods in statistics*, volume 161. Chapman and Hall London, 1987.

[124] M. McPherson, L. Smith-Lovin, and J.M. Cook. Birds of a feather: Homophily in social networks. *Annual review of sociology*, pages 415–444, 2001.

[125] F. McSherry. Spectral partitioning of random graphs. In *Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science*, 2001.

[126] David Mimno, Hanna Wallach, Edmund Talley, Miriam Leenders, and Andrew McCallum. Optimizing semantic coherence in topic models. In *EMNLP*, 2011.

[127] A. Moitra. An almost optimal algorithm for computing the nonnegative rank. In *Proceedings of the 24th annual ACM-SIAM symposium on Discrete algorithms*, 2013.

[128] Ankur Moitra and Gregory Valiant. Settling the polynomial learnability of mixtures of gaussians. In *Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science*, 2010.

[129] J.L. Moreno. *Who shall survive?: A new approach to the problem of human interrelations.* Nervous and Mental Disease Publishing Co, 1934.

[130] Elchanan Mossel and Sébastian Roch. Learning nonsingular phylogenies and hidden Markov models. *Annals of Applied Probability*, 16(2):583–614, 2006.

[131] Rajeev Motwani and Madhu Sudan. Computing roots of graphs is hard. *DISCRETE APPLIED MATHEMATICS*, 54:54–81, 1994.

[132] J.M. P. Nascimento and J. M. B. Dias. Vertex component analysis: A fast algorithm to unmix hyperspectral data. *IEEE TRANS. GEOSCI. REM. SENS*, 43:898–910, 2004.

[133] N. Nisan. Lower bounds for non-commutative computation (extended abstract). In *Proceedings of the 23rd symposium on Theory of Computing*, pages 410–418, 1991.

[134] B. A. Olshausen and D. J. Field. Sparse coding with an overcomplete basis set: a strategy employed by v1? *Vision Res*, 37:3311–25, 1997.

[135] G. Palla, I. Derényi, I. Farkas, and T. Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435(7043):814–818, 2005.

[136] M. Patrascu and R. Williams. On the possibility of faster sat algorithms. In *Proceedings of the 21st annual ACM-SIAM symposium on Discrete algorithms*, pages 1065–1075, 2010.

[137] K. Pearson. Contributions to the mathematical theory of evolution. *Philosophical Transactions of the Royal Society, London, A.*, page 71, 1894.

[138] Marc'Aurelio Ranzato, Y-Lan Boureau, and Yann LeCun. Sparse feature learning for deep belief networks. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 1185–1192, Cambridge, MA, 2008. MIT Press.

[139] J. Renegar. On the computational complexity and geometry of the first-order theory of the reals. *Journal of Symbolic Computation*, 13-3:255–352., 1992.

[140] M. Rudelson. Random vectors in the isotropic position. *J. Funct. Anal.*, 164(1):60–72, 1999.

[141] Arora Sanjeev and Ravi Kannan. Learning mixtures of arbitrary gaussians. In *Proceedings of the 33rd annual ACM symposium on Theory of computing*, STOC '01, pages 247–257, New York, NY, USA, 2001. ACM.

[142] T.A.B. Snijders and K. Nowicki. Estimation and prediction for stochastic blockmodels for graphs with latent block structure. *Journal of Classification*, 14(1):75–100, 1997.

[143] D. Sontag and D. Roy. Complexity of inference in latent dirichlet allocation. In *NIPS*, pages 1008–1016, 2011.

[144] Daniel A. Spielman, Huan Wang, and John Wright. Exact recovery of sparsely-used dictionaries. In *COLT*, 2012.

[145] A. Stegeman and P. Comon. Subtracting a best rank-1 approximation may increase tensor rank. *Linear Algebra and Its Applications*, 433:1276–1300, 2010.

[146] Keith Stevens, Philip Kegelmeyer, David Andrzejewski, and David Buttler. Exploring topic coherence over many models and many topics. In *EMNLP*, 2012.

[147] G.W. Stewart and J. Sun. *Matrix perturbation theory*, volume 175. Academic press New York, 1990.

[148] J. A. Tropp. Greed is good: algorithmic results for sparse approximation. *IEEE Trans. Inf. Theor.*, 50(10):2231–2242, September 2006.

[149] J.A. Tropp. User-friendly tail bounds for sums of random matrices. *Foundations of Computational Mathematics*, 12(4):389–434, 2012.

[150] Joel A. Tropp, Anna C. Gilbert, S. Muthukrishnan, and Martin Strauss. Improved sparse approximation over quasiincoherent dictionaries. In *IEEE International Conf. on Image Processing*, pages 37–40, 2003.

[151] Ledyard R Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3):279–311, 1966.

[152] L. G. Valiant. A theory of the learnable. In *Proceedings of the 16th annual ACM symposium on Theory of computing*, STOC '84, pages 436–445, New York, NY, USA, 1984. ACM.

[153] Vladimir N. Vapnik and Alexey Chervonenkis. On the Uniform Convergence of Relative Frequencies of Events to Their Probabilities. *Theory of Probability and its Applications*, 16(2):264–280, 1971.

[154] Stephen A. Vavasis. On the complexity of nonnegative matrix factorization. *SIAM J. on Optimization*, 20(3):1364–1377, October 2009.

[155] Santosh Vempala and Grant Wang. A spectral algorithm for learning mixture models. *J. Comput. Syst. Sci.*, 68(4):841–860, June 2004.

[156] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *ICML*, pages 1096–1103, 2008.

[157] M. Wainwright and M. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, pages 1–305, 2008.

[158] Hanna Wallach, Iain Murray, Ruslan Salakhutdinov, and David Mimno. Evaluation methods for topic models. In *Proceedings of the 26th International Conference on Machine Learning (ICML-09)*, 2009.

[159] Y.J. Wang and G.Y. Wong. Stochastic blockmodels for directed graphs. *Journal of the American Statistical Association*, 82(397):8–19, 1987.

[160] P. Wedin. Perturbation bounds in connection with singular value decomposition. *BIT Numerical Mathematics*, 12(1):99–111, 1972.

[161] H.C. White, S.A. Boorman, and R.L. Breiger. Social structure from multiple networks. i. blockmodels of roles and positions. *American journal of sociology*, pages 730–780, 1976.

[162] Wei Xu, Xin Liu, and Yihong Gong. Document clustering based on non-negative matrix factorization. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, SIGIR '03, pages 267–273, New York, NY, USA, 2003. ACM.

[163] Jianchao Yang, John Wright, Thomas S. Huang, and Yi Ma. Image super-resolution via sparse representation. *Trans. Img. Proc.*, 19(11):2861–2873, November 2010.

[164] M. Yannakakis. Expressing combinatorial optimization problems by linear programs. *JCSS*, pages 441–466, 1991.

[165] Limin Yao, David Mimno, and Andrew McCallum. Efficient methods for topic model inference on streaming document collections. In *KDD*, 2009.

[166] Chen Yudong, Sujay Sanghavi, and Huan Xu. Clustering sparse graphs. In *Advances in Neural Information Processing Systems 25*, 2012.

[167] Tong Zhang and Gene H. Golub. Rank-one approximation to high order tensors. *SIAM J. Matrix Anal. Appl.*, 23(2):534–550, February 2001.